

Telephony Application Development Platform

Evaluation System for Windows 95 and NT

User's Guide

[Table of Contents](#)



[Next Pg](#)

Table of Contents

CHAPTER 1 INTRODUCTION 3

- What is MasterVox 3
- Evaluation System Overview 3
- Installing the MasterVox Evaluation 4
- About this Manual & Sample Applications 4

CHAPTER 2 THE 4 STEPS TO CREATING AN APPLICATION.. 5

CHAPTER 3 DESIGN PROCESS OVERVIEW 6

- All About Actions 6
- Most Common Non-Switching Actions 8
- Action Logic Paths 10
- Audiotex Actions 12
- Disjointed Connectors 13
- How Actions Work in Running Applications 13
- Application Starting Point and Default Actions 14
- Action Comments 14
- Main Applications and Sub-Applications 15
- Reducing Actions to Sub-Applications 16
- MasterVox Variables 18
- What are MasterVox Variables 18
- How are MasterVox Variables Created 18
- How are MasterVox Variables Used 18
- SQL Based Database Support 20
- Application Files 20
- Voice Files 21

CHAPTER 4 STEP 1: Design An Application 22

- Starting the Designer 22
- Examine a Sample Application - Playrec_demo 22

CHAPTER 5 STEP 2: Configure the MasterVox Environment 25

CHAPTER 6 STEP 3: Record Voice Prompts 27

CHAPTER 7 STEP 4: Run the Simulated Application 28
 Playrec_wav_demo

Table of Contents



Chapter 1 Introduction

What is MasterVox?

MasterVox® is an extremely powerful computer-telephony application development and deployment system for the Natural MicroSystems (NMS) family of voice-processing hardware and other MVIP-compatible telephony products. It provides dramatic savings in the time and cost associated with computer-telephony system development (reducing development time from months to days) and serves as a stable and robust production platform on which to deploy your telephony systems. While it is an invaluable tool for the seasoned developer, MasterVox also opens the door to non-programmers who want to create computer-telephony applications — either for their own use or value-added resale.

MasterVox is not a new language to learn, but rather an intuitive and highly visual system for defining and implementing industrial-strength computer-telephony applications. Using MasterVox, you will work with abstract, high-level design elements (called 'Actions') without having to worry about low-level software coding. Actions are simple, easy-to-conceptualize, discrete activities you want your system to perform — such as Answer Call, Record Message, Speak Date, or Transfer Call.

Evaluation Overview

Welcome to the MasterVox Evaluation System. The full-blown version of MasterVox is made up of three distinct parts: The Designer, which is the visual toolbox for creating and viewing computer telephony applications; The Configuration Manager, in which you tell the system which MasterVox applications to run, and allows you to configure the MasterVox product for your particular hardware and software environments, as well as adjust how your voice-processing hardware behaves; And finally the Engine, which runs your MasterVox applications (simulated in this evaluation). This Evaluation includes a fully functioning version of the MasterVox Designer and Configuration Manager and a simulation version of the Engine.

The limitations in this Evaluation are that the greater part of the Configuration Manager is not applicable in simulation mode (since telephony hardware does not play a part in simulation mode), the Designer will only save up to 25 Actions of a design to disk, and the Engine will only run in simulation mode for up to 20 minutes, and on a maximum of 8 lines.

[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)

[MasterVox Evaluation User's Guide](#)

Chapter 1 Introduction

Installing The MasterVox Evaluation

Please insert the CD-ROM into your CD-ROM drive. If the setup program does not start automatically, please click on the 'start' button on the Windows Task Bar, select 'run' and type in 'x:\setup', where 'x' is the letter of your CD-ROM drive. Select the default options when prompted by the installation program.

About This Evaluation and Sample Application

The purpose of this manual is to walk the user through the steps involved in taking an application from design to execution, and to highlight some of MasterVox's more powerful features. In order to do this, we will be examining a fully functioning MasterVox sample application ([playrec_demo.dat](#)) which can be used to record prompts for use in other MasterVox applications. There are a variety of sample applications included in the MasterVox Evaluation package, and they can be found in the following directory:

X:\mstrvox\apps

(where 'x' is the drive upon which you installed the MasterVox Evaluation)

Certainly, this example application will not make use of all the MasterVox features and Actions, as it is intended only to help the user feel comfortable with the design process, and to show how simple it can be to take an application from design to execution. It is highly suggested that you peruse the sample applications in order to gain some insight on the design process.

[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)

MasterVox Evaluation User's Guide

Chapter 2 4 Steps From Design to Execution

The application process can be divided into four distinct steps:

STEP 1 - Design the Application

This step is completed in the MasterVox Designer which provides all the tools you need to create a robust, fully-functioning telephony application.

STEP 2 - Configure the Environment

This step refers to both hardware and software configuration and is accomplished through the MasterVox Configuration Manager. For the purposes of this evaluation, where telephony hardware is not a consideration, the Configuration Manager will only be touched upon with respect to software configuration - telling the system which application(s) to run. In a real application situation it is necessary to be somewhat familiar with the telephony hardware installed in your PC.

STEP 3 - Record Prompts

The prompts used in the playrec_wav_demo application which we will be creating in this manual have been provided for your use. In a real application situation you can use one of the playrec applications to create your own prompts for your applications.

STEP 4 - Run the Engine

This final step will actually run the [playrec_wav_demo](#) application for evaluation.

[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)

[MasterVox Evaluation User's Guide](#)

Chapter 3 Design Process Overview



Wait For Call



Answer Call



Hang Up

**Three Action Icons:
Wait for Call, Answer Call
& Hang Up Line**

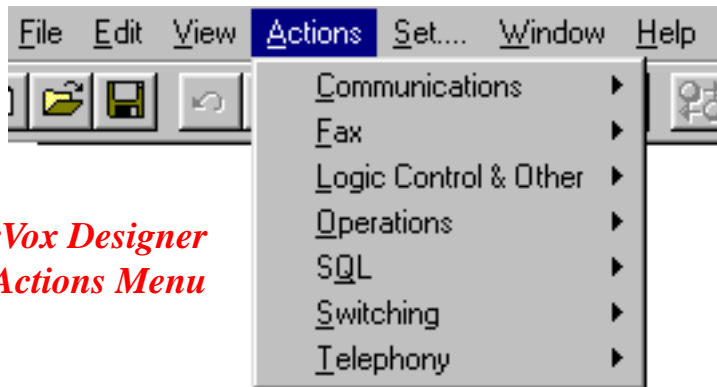
Table of Contents



The following sections will help you to quickly get started building simple MasterVox applications and will explore the MasterVox Designer environment. Before we go full-steam into our design, it will be helpful if you are familiar with some general concepts about MasterVox Applications.

ACTIONS

The MasterVox Designer allows you to design your telephony applications by visually piecing together high-level Actions into a logical call-flow diagram. The available Actions are broken down by category and are displayed as sub-menus under the 'Actions' pull-down menu available on the main Designer window. In addition, the 10 last used unique Actions also appear on a movable button bar.



Every MasterVox application is made up of a series of Actions. The MasterVox Designer is simply a toolbox of Actions - in which you put together the Actions you wish your application to perform. You may put Actions together in an endless number of ways to create an endless number of applications. Think of the Actions in the toolbox as a set of Lego blocks, and one design differs from the next by exactly which blocks you choose to use and how you put them together. You may use the same blocks to build a bridge as you would to build a pyramid - the only difference is how the pieces fit together.

Every MasterVox Action performs one task or a small set of logically related tasks. The following list of MasterVox Actions should give you an idea about Actions:

Action Name Task(s) Performed

- Wait For CallWait for an incoming call, collect incoming digits
- Answer CallAnswer an incoming call
- Hang UpHang up the line
- AudiotexPlay a message to a caller and/or collect caller input
- Compare DataCompare two pieces of data

Chapter 3 Design Process Overview

You could begin by selecting the Action Wait for Call from the 'Telephony' sub-menu. Upon selecting the Action, a corresponding icon appears in the upper left corner of your application workspace. At any time you may move the Action icons on the screen to any desired position, in order to keep the application readable, by clicking on and dragging the Action icon.

Associated with each Action in the MasterVox system is a 'Configuration Dialog Box', where you enter data and/or variables into fields which define how the Action behaves. After positioning an Action icon to the desired position on your workspace, you must next Configure the Action. To do this, simply double click on the Action icon to pull up its configuration dialog box.

Assign Value to Variable

Variable: @Today Value: []

Assign value Options: Date representation for current locale

Comment: Get today's date

Buttons: Cancel, OK

Configuration Box for 'Assign Value to Variable' Action

For example, in the 'Assign Value to Variable' Action dialog, you enter both the name of the variable that is to be assigned a value, as well as the value to be assigned. In the 'Speak Data' Action dialog, you select the type of data to be spoken - digits ('one', 'two', 'three'), number ('one hundred twenty three'), money ('one hundred twenty three dollars'), etc - and you enter the actual digits, number, dollars to be spoken. Once finished, simply click on the OK button to save the Action. The dialog box will disappear and you will be ready to select your next Action.

[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)

MasterVox Evaluation User's Guide

Chapter 3 Design Process Overview



Wait For Call

Unselected 'Wait for Call'
Action icon



Wait For Call

Selected 'Wait for Call'
Action icon

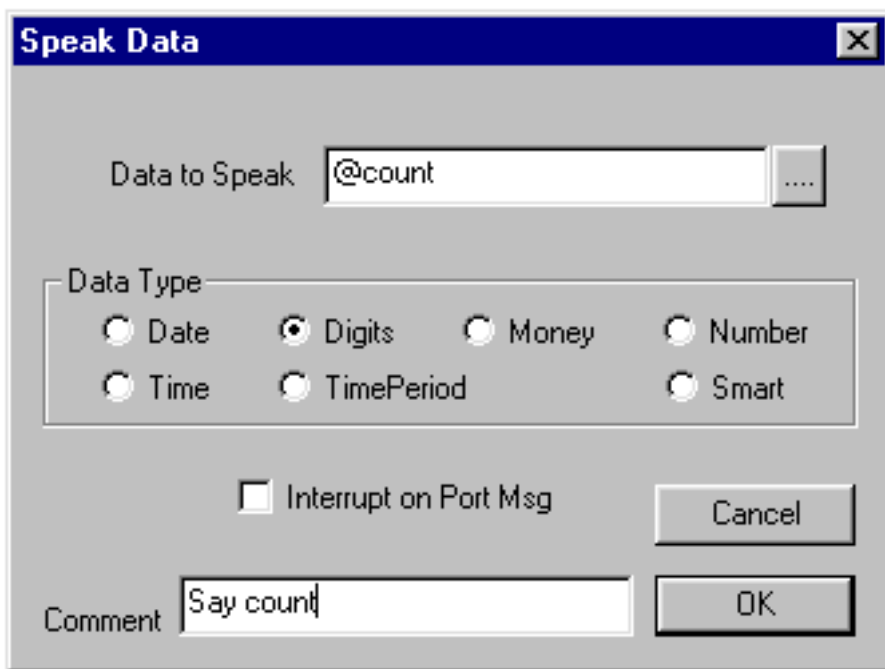
[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)



Configuration Dialog Box for 'Speak Data' Action

You may move Actions in your workspace and open their configuration dialog boxes at any time in the designer in order to modify your design.

You may also delete an Action from your design at any time by simply clicking once on the Action icon to **select it**, and pressing the delete key.

MOST COMMON NON-SWITCHING ACTIONS

In order to introduce you to the MasterVox Action Toolbox, the following list includes the most commonly used non-telephony actions and the tasks they perform.

Action Group	Action	Description
Logical Control & Other Actions		
	Assign Value to Variable	assign a literal value, or many system-determined variables such as date, time, line number, etc.
	Call Sub-App	call another MatserVox application (sub-app)
	Sub-App Return	return to the calling application
	Compare Data	compares strings, numbers, dates and times
Operations		
	Date	perform date operations
	Time	perform time operations
	Numeric	perform numeric operations
	String	perform string operations
	File	open/close/rename files, create directories, etc
SQL		
	Connect	connect to a database
	Disconnect	disconnect from a database
	Begin Transaction	start a SQL transaction
	Commit Transaction	ends and commits a SQL transaction
	Rollback	rolls back an uncommitted transaction
	Query	query a SQL database
	Next record	get the next record in the query
	Exec SQL	execute a SQL statment (insert, update, etc.)
Telephony		
	Wait for Call	wait for an incoming call
	Answer Call	answer incoming call
	Hang Up	hang up the line
	Place Call	place internal or external call
	Speak Data	speak a number, digits, money, date, time, etc.
	Audiotex	play a recorded message file, get user touch-tone input
	Record Message	record a message
	Dial Digits	dial touch-tone digits
	Select Language	change default language

Chapter 3 Design Process Overview

ACTION LOGIC PATHS

To create applications, you will be putting Actions together into logical paths to define a call flow. There are up to 4 different types of path's that a call may take - depending upon what events take place during the call. A path is the direction a call takes from one Action to another. Every Action has **one entry point** and from **two to four exit points** (the only exception to this is the *Audiotex* Action which can have up to 16 exit paths - more on that later).

Not all Actions have 4 possible exit points, but all have at least 2. The 4 main types of exit points that any Action may have are listed below, and in each case, the exit path is represented by a colored line and arrow connecting one Action to another:

- Success** *All* Actions have a Success exit point represented by a **green** arrow
- Failure** *All* Actions have a Failure exit point represented by a **red** arrow
- Time-out** *Most* Actions have a Time-out exit point represented by a **blue** arrow
- Hang-up** *Some* Actions have a Hang-up exit point represented by a **purple** arrow

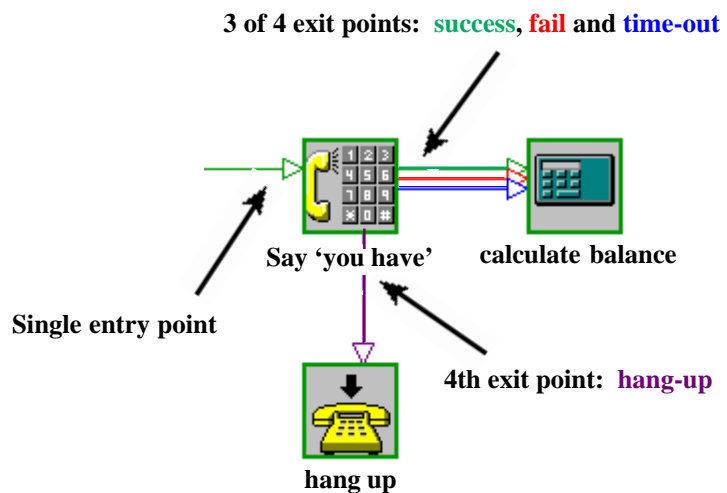


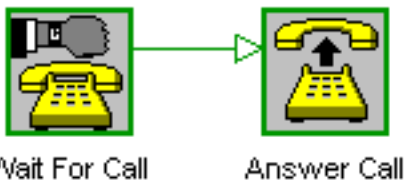
Illustration of an action with one entry point and four exit points (success, fail, time-out and hang-up)

[Table of Contents](#)



Chapter 3 Design Process Overview

Change the state of any action by right clicking on the action's icon - it's outline will change color



Application execution goes from 'wait for call' to 'Answer call'

Table of Contents



Once the first Action is in place on your workspace, you will place a second Action into your design. MasterVox will automatically connect the first Action to the second Action with a green arrow (success path). You may add, alter or delete any path from one Action to another as follows:

Select the type of path you wish to delete or alter

Right click any Action in your design. Notice that with each click the outline color of the Action changes from green to red to blue and so on. The color of the Action's outline determines which exit path you will be changing



Compare Data

Action in a success state signified by green outline



Compare Data

Action in a fail state signified by red outline

Delete a path

To delete a success path from one Action to another. Right click the 'connect-from' Action until a green outline appears (success path). Hold the right button and drag the mouse out of the icon and release the mouse. A confirmation alert will appear to confirm that you wish to delete the path. Simply press OK and the path will disappear.

Join two Actions

To join two Actions with a success path, simply right click the 'connect-from' Action icon until the outline is green. Hold the right mouse button down and drag the mouse to the 'connect-to' Action. Release the mouse on the destination Action and a green line will appear from the first Action to the second Action.

The steps to change other types of paths (failure, time-out, hang-up) are the same. Simply right click the Action until the outline appears in the color for the desired type of path you wish to change.

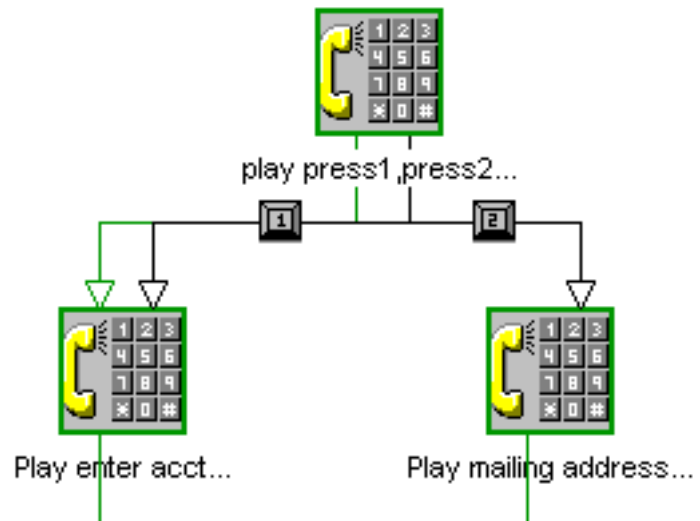
Chapter 3 Design Process Overview

The Exception - Audiotex Actions

The exception to the above is the *Audiotex Action*. The purpose of the Audiotex Action is to play a prompt to a caller and/or collect the caller's response to the prompt (e.g. entering an account number or making a selection from a menu - press 1 for this, press 2 for that and so on). In addition to the four basic exits discussed above, a caller could be asked to enter any number from 0 - 9, star or pound. Consequently, the Audiotex Action could have up to 16 different exit paths (e.g. success, failure, time-out, hang-up, press1, press2, press3, etc).



*Audiotex Tone Pad
with '1' selected*



Branch from an Audiotex Action, caller presses 1 to enter account, caller presses 2 for mailing address

To define the paths for SUCCESS, FAILURE, TIME-OUT and HANG-UP from an Audiotex Action to another Action, the steps are the same as any other Action. After that, the difference is apparent.

To branch from an Audiotex Action to another Action for user input from 0-9, star or pound, right click the Audiotex icon until the state of the Action indicates 'tone' - the Action will have a black border and a touch-tone window will appear. On the touch-tone window that appears, select the tone for which you wish to define a path (e.g. to define the path for user selects one, click on the 1 in the touch-tone pad). Next, place the mouse on the Audiotex Action, then click and hold the right mouse button. Finally, drag the mouse to the 'connect-to' Action and release the mouse. A black path with a touch-tone 1 will appear connecting the Audiotex Action to the destination Action when a caller enters 1.

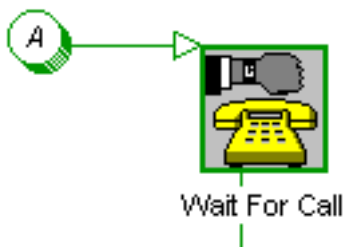
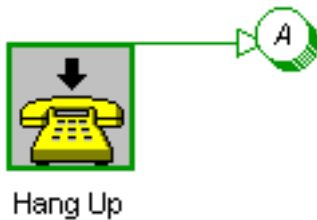
Table of Contents



Chapter 3 Design Process Overview



*Disjointed Connector Icon
from Designer Toolbar*



Two actions connected with a disjointed connector - call flow proceeds from 'Hang-up' to 'Wait for call' upon success (green path)

Table of Contents



To specify another branch path from the same Audiotex Action, select the next desired key from the Tone Pad and repeat the right mouse point-and-click process from the Audiotex Action to the desired branch Action. You may specify up to twelve different branches from a single Audiotex Action.

You can delete a branch connection from an Audiotex Action by first making the desired key active on the Tone Pad, then holding down the right mouse button and dragging the mouse pointer outside the Application Workspace. You will be prompted to confirm that you wish to delete the branch connection.

This application building process continues until you have entered all the Actions required to implement your application. In a typical telephony application, the logic flow loops back to the beginning upon termination of a telephone call. If this is the case with your application, don't forget to make the connection between the end of the call (e.g., the Hang Up Action) and the beginning of the call (e.g., the Wait for Call Action).

Hint: This is a good place to use the 'Create Disjointed Connector' feature – in order to keep your design neat.

Disjointed Connectors

The disjointed connector is an excellent MasterVox feature that allows you to connect two Actions which appear long distances from each other in your workspace area - to avoid long paths between Actions which may span the length you're your design. This allows you to keep your design neat and easy to read.

This is accomplished by first right clicking on the '**connect-from**' Action to set the desired path type you are creating (success, fail, etc.). Next, click on the disjointed connector tool on the toolbar, and finally click on the '**connect-to**' Action.

How Action Paths Work in a Running Application

After execution of an Action, the exit point which is taken by the call is dependent upon the outcome of the Action. For example, A SQL Query Action has 3 possible exit points: Success, Failure and Time-out. If the execution of a SQL Query Action ends successfully by finding a matching record in the database, the call will exit through the Success point and follow the green path to where it leads. If the queried record is not found, the Failure exit (red path) will be followed, and if the query times out before a match is found, the Time-out exit (blue path) will be followed.

Chapter 3 Design Process Overview

Before saving an application design the first time, you MUST define a starting point and default actions



Application Starting Point Icon

Default Action Icons:



Success



Failure



Time-Out



Hang-Up

Table of Contents



APPLICATION STARTING POINT & DEFAULT ACTIONS

Before you can save your design, you must specify the application starting point (where execution of the application begins) and the default logic Actions.

For the application starting point, simply click on the Begin Here icon on the toolbar. Next click on the Action which is to be the application starting point. The 'Begin Here' tag will attach itself to the specified Action.

Typically when designing an application, all possible exit points for any Action are explicitly defined and lead to another Action. In some cases you may overlook setting an exit point for an Action. Or, for instance, you may wish to send all Time-Outs to one central action. All MasterVox applications must have a default path defined for all 4 basic exit types (Success, Failure, Time-out and Hang-up). This way, when a call must take an exit which is undefined, the default path for that type of exit can be followed.

For example, you may have a SQL Query Action for which there is no Time-Out path defined. If a call reaches this Action and the Action times out waiting for the result of the query, the call must have a place to go based on a Time-Out result. If such a situation arises, the call will proceed to the Action defined as the 'Time-Out Default Action'.

To define the Default Success Action simply click on the 'Default Success' icon on the toolbar, then click on the Action which will serve as the default Action for the application flow to proceed to upon successful completion of an Action where no other success path is indicated. Use the same procedure to define the 'Default Failure', 'Default Time-Out', and 'Default Hang-Up' Actions with the appropriate toolbar buttons. Note that multiple default logic tags can be attached to the same Action. Now you can save your design and exit the Designer using the 'File Save' or 'File Save As' pull-down menu options.

ACTION COMMENTS

Every MasterVox Action has a comment associated with it to make your design more readable. Comments for Actions can be entered into the 'comment' field of any Action's dialog box. Comments should describe what an Action does in the context of your application. For example, in an Action where you are playing a message from a voice file, you might employ a comment such as 'Play Main Menu' to make it obvious with a glance at your design which message is to be played at that point.

Chapter 3 Design Process Overview

MAIN APPLICATIONS & SUB-APPLICATIONS (SUB-APPS)

To promote good design habits, MasterVox Applications can actually be broken into a Main Application and any number of Sub-Applications, which may be loaded statically (when the engine is started) or dynamically (during execution of the main program). By breaking an application into sub-apps, you create a program which is reusable (if a certain group of Actions are repeated multiple times), easy to maintain and easy to edit (sometimes, applications simply become too large). Typically, a sub-app is an application which performs one task or a group of logically-related tasks.

When this is done, the *calling* application is referred to as the Main Application and the *called* application is referred to as the sub-application. When one sub-app calls another sub-app, this is referred to as sub-application 'nesting'. MasterVox allows unlimited nesting of sub-apps.

For example, you may have a program which will answer a call, request an account number from the caller, offer a list of options to the caller, then perform a task based on the caller's selection from the list. This example may be broken up into a main applications and any number of sub-apps, as follows:

Application/Sub-App Task(s)

Main Application	Answer a call, request an account number, call sub-app to look up account
Call Acct Sub-App...	Look for account information in a database, return to main app
Main Application	Play list of options to caller, wait for caller input and call sub-app which corresponds to caller's selection
Call Sub-App1	Perform a task for option 1, return to main application
or	
Call Sub-App2	Perform a task for option 2, return to main application
or	
Call Sub-App3	Perform a task for option 3, return to main application
Etc...	
Main Application	Hang up and wait for next call (return to the top)

[Table of Contents](#)



[Prev Pg](#)

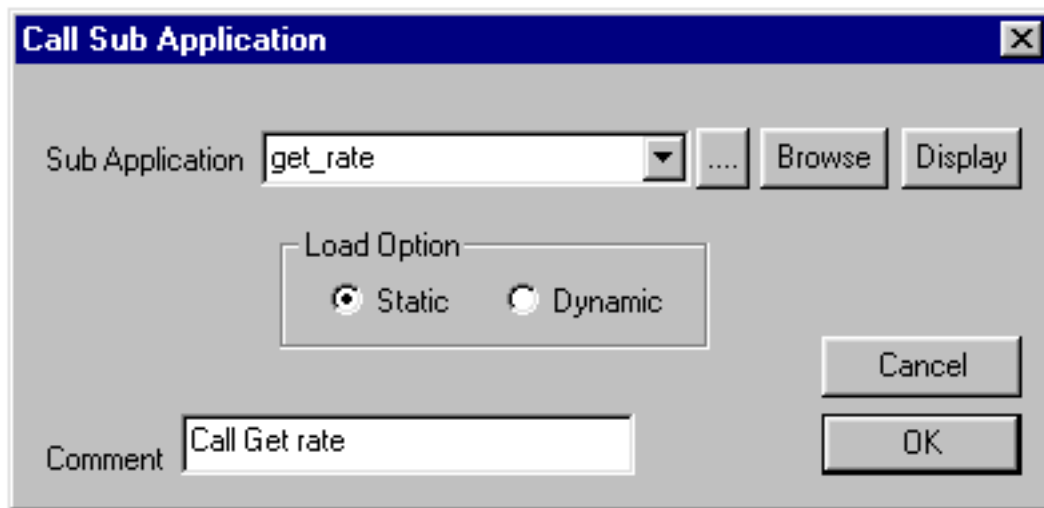


[Next Pg](#)

[MasterVox Evaluation User's Guide](#)

Chapter 3 Design Process Overview

This, of course, is a very simplistic view of how you may break up an application into pieces or sub-apps. The possibilities are endless. However, when an application is designed in this modular fashion, you may find that many of your sub-apps can be reused over and over again from one application to the next. Also, when you need to make a change to, say, task1 above, you need only to edit a small sub-app. Rather than open a large file, where, if the application is large, you not only must find the Actions to be edited, but you also run the risk of making unwanted changes to other parts of the application.



Call Sub-Application Action Configuration Dialog Box

REDUCING ACTIONS TO SUB_APPLICATIONS

You may reduce any set of Actions in a MasterVox application to a Sub-Application by positioning the cursor above the upper left-most Action in the group to be reduced, clicking with the left mouse, and dragging an edit box around the group of Actions to be reduced. Then click the right mouse (or chose from the 'Edit' drop down menu) and select 'Reduce to Sub-App'. MasterVox then prompts you to enter a Sub App name. This process automatically saves the selected Actions to the new .dat file, as well as attaches a default 'Begin Here' icon to the first Action, and adds a 'Sub-App Return' Action with 'Success, Failure, Time-out and Hang-up icons attached. It also places a 'Call Sub-App' Action in their place in your original application.

[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)

[MasterVox Evaluation User's Guide](#)

Chapter 3 Design Process Overview



Call Sub-App

Call Sub-App Action Icon



Sub-App Return

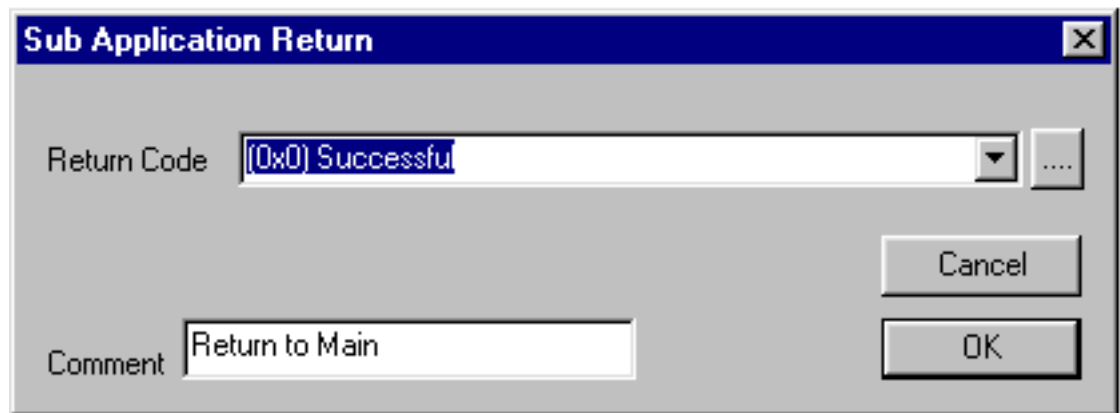
Sub-App Return Action Icon

CALLING SUB-APPLICATIONS

Sub-applications are invoked by means of the Call Sub-Application Action. This Action accepts a parameter which is the name of the application you want to invoke. This is the file name (with or without the suffix) of another MasterVox application which you have either previously designed or plan to design.

SUB-APPLICATION RETURN CODES

MasterVox provides a special Action called 'Sub-Application Return' which allows you to directly return from a sub-application to the calling application. Included in the Configuration Dialog Box for this Action is a Return Code Field in which you specify the type of return is made (e.g., SUCCESS, FAILURE, TIME OUT, HANG UP). The logic path followed upon return from a sub-app is based upon the return code selected for the Sub-Application Return Action.



Sub-Application Return Action Dialog Box with return code = successful

[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)

MasterVox Evaluation User's Guide

Chapter 3 Design Process Overview

*MasterVox Variables are
CASE SENSITIVE*

*ALL MasterVox Variables
must begin with an
ampersand '@'*

Table of Contents



Prev Pg



Next Pg

MASTERVox VARIABLES

What are MasterVox Variables?

MasterVox variables are general-purpose named 'data buffers' which can be used in a variety of ways. For example, you might need to perform a mathematical operation, such as subtracting a value from an account balance or a comparison such as checking the current time against the end of business hours. Variables can be used to accomplish these tasks.

How are variables created in MasterVox?

Variables are created when they are first referenced in any MasterVox Action. However, to avoid problems caused by using variables whose contents are undefined, variables should first be used in an Action where they are the recipient of data, rather than the source of data. In other words, variables should first be used in Actions where they are assigned a value.

Examples of MasterVox Actions where a variable is assigned a value are:

- Wait For Call** Designate variables to hold the calling number, called number, and caller ID of an inbound call.
- Audiotex** Designate a variable to collect digits entered by a caller
- Assign Value to Variable**.... Designate a variable to which you assign a value

Additionally, the SQL Query Action, in which you query a database for information, automatically defines variables for you – one variable per field returned in the query.

How are variables used in MasterVox?

Variables are distinguished from literal data by means of the 'at' sign (@). Consequently, variable names must always begin with the @ (e.g., where MyVar would be taken literally, @MyVar defines a variable name). Once a variable has been defined (either by using the Action Assign Value to Variable or by simply using the variable for the first time in any other Action), its value can be used or changed in other Actions by specifying the variable name.

Variables resulting from a database query take on a slightly different format from regular variables. They all begin with the 'at sign' (@), followed by the name of the query, a colon (:), and finally the name of the database field retrieved. For example, you define a query called CALLERINFO in which you retrieve an account number and a balance.

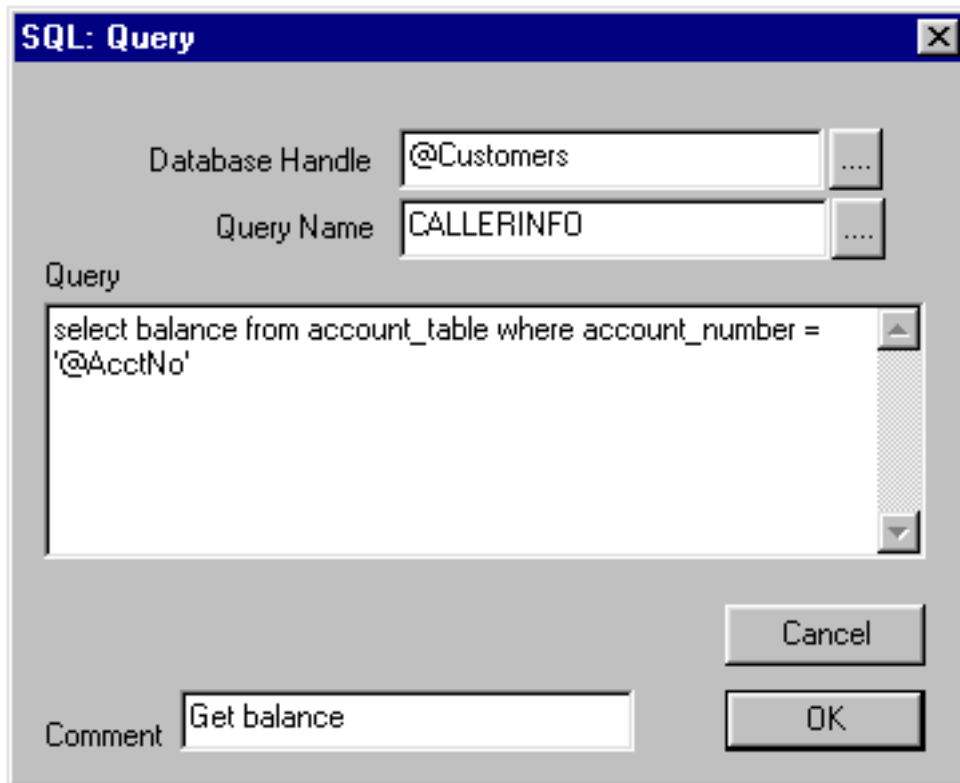
Chapter 3 Design Process Overview

The CALLERINFO query may read:

```
select balance from account_table where account = '@AcctNo'
```

The variable created as a result of the query would be as follows:

```
@CALLERINFO:balance
```



SQL Query Dialog Box

the result of this query will create the variable: @CALLERINFO:balance

An example of the usage of MasterVox variables is as follows: Consider an application which uses the MasterVox Action Speak Data. When filling in the Dialog Box for the *Speak Data* Action, suppose you select 'Digits' as Data Type to speak. For the 'Data to Speak' field, you have two options:

You can enter a literal value in this field, e.g., '123', and the system will speak those digits directly when this Action is reached during processing. That is, it will speak 'one', 'two', 'three.'

Alternately, you can enter a variable in this field and the system will speak the contents of the variable when this Action is reached during processing. For example, suppose you had previously defined a variable called '@Count' in an earlier Action, and had assigned it the value '999'. You can enter '@Count' in the 'Data to Speak' field, and the application will speak 'nine', 'nine', 'nine' when this Action is reached during processing.

Table of Contents



Prev Pg



Next Pg

MasterVox Evaluation User's Guide

Chapter 3 Design Process Overview

Virtually all MasterVox Action Dialog Box input fields accept variable inputs (except some cases where it is necessary to select from a drop-down list). Use of variables represents the most powerful and flexible means for the MasterVox developer to obtain, evaluate, modify, and make decisions based upon information and/or data used in an application.

SQL-BASED DATABASE SUPPORT

Increasingly, organizations are choosing to establish a central database repository for their data (a database server) and access it across a network from various applications on various types of machines around the organization (clients). This database approach is typically deployed using one of the many available client-server, structured query language (SQL)-based relational database management systems (RDBMS). MasterVox supports the most powerful and popular of these high-end database systems operating in a client-server architecture or standalone on a telephony server computer.

MasterVox SQL database support complies with the industry-standard ODBC protocol. In order to use the SQL-related Actions provided by MasterVox, you will need to select and obtain: 1) a database system for your server's operating system and for which an ODBC driver for Windows NT is available, 2) the appropriate network software where applicable from the database vendor, and 3) the corresponding ODBC driver for a Windows NT 4.0 client. ODBC drivers are available from a number of sources, including Microsoft (the author of the ODBC specification) and Intersolv. You can also obtain the ODBC specification and write your own ODBC driver if you wish to use a non-standard or proprietary database system. A small sample of the databases compatible with MasterVox for NT, with corresponding ODBC drivers (from Intersolv or Microsoft), are provided on our Web Page. Note that this is a small subset of available databases and drivers, and you should contact the various ODBC driver suppliers for additional and more up-to-date information regarding database and driver compatibility and availability.

APPLICATION FILES

MasterVox applications are stored in application files. Application files are compact binary files and are stored in the application directory (usually 'MSTRVOX\APPS'). They are identified by the suffix '.DAT'. These files are very small, yet contain complete specifications for your applications. Consequently, they facilitate support for applications that are distributed over a large geographic area due to their small size and resulting speed and ease with which they can be transmitted via modems and networks.

[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)

[MasterVox Evaluation User's Guide](#)

Chapter 3 Design Process Overview

*MasterVox is Compatible with
VOX, WAV and OKI
audio file types*

*DO NOT include the file
extension (.vox, .wav, .oki)
when referencing a voice
file in an Audiotex Action*

VOICE FILES

As MasterVox is compatible with a variety of voice file types, VOX, WAV and OKI, the sample application we will be examining refers to a file of the VOX type - an NMS proprietary voice file type. One thing which makes VOX files unique, is the ability to encapsulate more than one prompt file into one large file, each prompt is referred to as a single message in the file. So, you can record all your prompts separately, then place them all into one file - assigning each prompt a different message number. This is useful for managing large numbers of voice prompts used in one application.

For our example application, playrec_demo.dat, only one voice file is used, playrec.vox. However, this file contains 10 separate voice prompts, numbered 1 through 10. You will notice that almost all Audiotex Actions refer to this one VOX file, but each refers to a different message number within the file.

[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)

[MasterVox Evaluation User's Guide](#)

Chapter 4 Design an Application

*To start the MasterVox Designer...
select it from
/Programs/MasterVox Evaluation
from the Windows Start Menu*

In this example, we will be examining the sample application `playrec_demo.dat`. A finished version can be found in the MasterVox \Apps subdirectory for reference. The purpose of the `playrec_demo` is to facilitate recording messages (voice prompts) to be used in MasterVox applications. There are two versions of the `playrec` application - `playrec_demo`, which can be used for recording 'vox' format messages through the use of telephony hardware, and `playrec_wav_demo`, which performs the same Actions as `playrec_demo`, but uses 'wav' format messages for use in simulation mode where telephony hardware does not play a part.

Start the Designer

Run the MasterVox Designer by selecting it from Programs/MasterVox Evaluation in the Windows (NT or 95) 'Start' menu or by double clicking the MasterVox Designer icon in the MasterVox Evaluation program group.

Upon startup, the designer will open a new, untitled design and displays a fresh, blank workspace. Open the `playrec_demo.dat` file from the *file menu* by selecting *open*.

Let's Examine Sample Application: `Playrec_demo.dat`

The following will walk you through the `playrec_demo` application, highlighting some important points about the application.

Action: Wait for Call

This is the application's starting action and will wait for an incoming call. Upon successfully receiving an incoming call, the application proceeds to the next action along a *success (green) path*. Notice that no *failure path* has been defined for the action. If the wait for call action should fail, then execution would proceed to the *default failure action* - identified by the *default failure tag* attached to the 'hang-up action'.

Action: Answer Call

As the name implies, this action will answer the call. It is configured to answer after one ring. Upon a successful answer, the call will proceed along the *success path* to the first recorded prompt. Once again, notice that no *failure path* has been defined for the action. If the answer call action should fail, then execution would proceed to the *default failure action* - identified by the *default failure tag* attached to the 'hang-up action'.

Table of Contents



Chapter 4 Design an Application

Action: Introduction

This is an Audiotex Action which will play a recorded voice prompt to the caller. In this example, the action plays a VOX type message file called *playrec*. It prompts the user to press the pound (#) key for help. It will wait up to 2 seconds for the caller to enter something, and is expecting only one digit as input. It will accept any keys as input from the caller, but will only act upon the pound. If the caller presses pound, the call will proceed along the *pound* (black) *action path* to an action which will play another prompt to the caller explaining the application. If no input is received from the caller within 2 seconds, the action will time-out and execution will proceed along the *time-out* (blue) *action path*.

If the action fails or the caller hangs up, execution will proceed to the hang-up action which is designated as both the *default failure* action, as well as the *default hang-up* action.

Action: Get Msg

Another Audiotex Action, this action plays a prompt to the caller asking for a 2-digit message number to be played or recorded. If no input is received from the caller, it will repeat itself up to three times before a failure condition is assumed. It will wait up to 4 seconds for the caller to respond and is expecting a 2-digit response from the caller. If the action is successful, it will place the digits entered by the caller into a variable called **@MsgNbr**, and will proceed to the next action along the *success path*.

Action: Msg

This action simply plays a message to the caller which basically says 'the message number you entered is'. It does not expect any input from the caller and proceeds to the next action after the message has been played.

Action: Speak Data

This action will speak the message number entered by the caller back to the caller. Upon examination, the number to be spoken is contained in the variable **@MsgNbr** and the action has been configured to say the data as whole number, as opposed to separate digits (e.g. 'one hundred twenty three' instead of 'one two three'). Upon successful completion of the action, the call will proceed to the Main Menu along the green success path.

Table of Contents



Prev Pg



Next Pg

MasterVox Evaluation User's Guide

Chapter 4 Design an Application

Action: Main Menu

This action will play a menu to the caller, from which the caller must make a selection. Press 1 to record the message, press 2 to play the message, press 3 to enter a new message number, press 4 for help, and press 5 to exit. The action is expecting only 1 digit from the caller, it will wait up to 4 seconds for the caller to respond, and it will repeat itself up to 3 times waiting for the caller to respond. If the caller presses 1 or 2, the call will follow the appropriate path for the number entered. If a 3 is pressed, the call will jump back to a previous Audiotex Action asking for a message number. If a 4 is entered, a help message will be played, and the call will jump back to the Main Menu again - via a green *disjointed connector* labeled 'E'. If a 5 is entered, the call will end, the line will be hung up, and the application will jump back to the 'Wait for call' Action at the top - via the green *disjointed connector* labeled 'A'.

With the exception of the 'press 5' path leading from the 'Main Menu' Action to the 'Hang up' Action, all branches leading out of the 'Main Menu' Action wind up back at the 'Main menu' Action eventually, and all use *disjointed connectors* to get there.

If at any point an action fails, times out or the caller hangs up, the call will proceed to the default for these actions, in this case, the Hang-Up Action - note the default tags connected to the Hang-up Action. Once the line is hung up, the application returns to the top to wait for the next call.

THE OTHER ACTIONS

The rest of the application should be self-explanatory at this point. Take a moment to examine the other actions in playrec_demo and step through the logic of the call flow.

In a real telephony application setup, this step refers to both hardware and software configuration and is accomplished through the MasterVox Configuration Manager. For the purposes of this evaluation, where telephony hardware is not a consideration, the Configuration Manager will only be used to tell the system which application(s) to run.

[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)

[MasterVox Evaluation User's Guide](#)

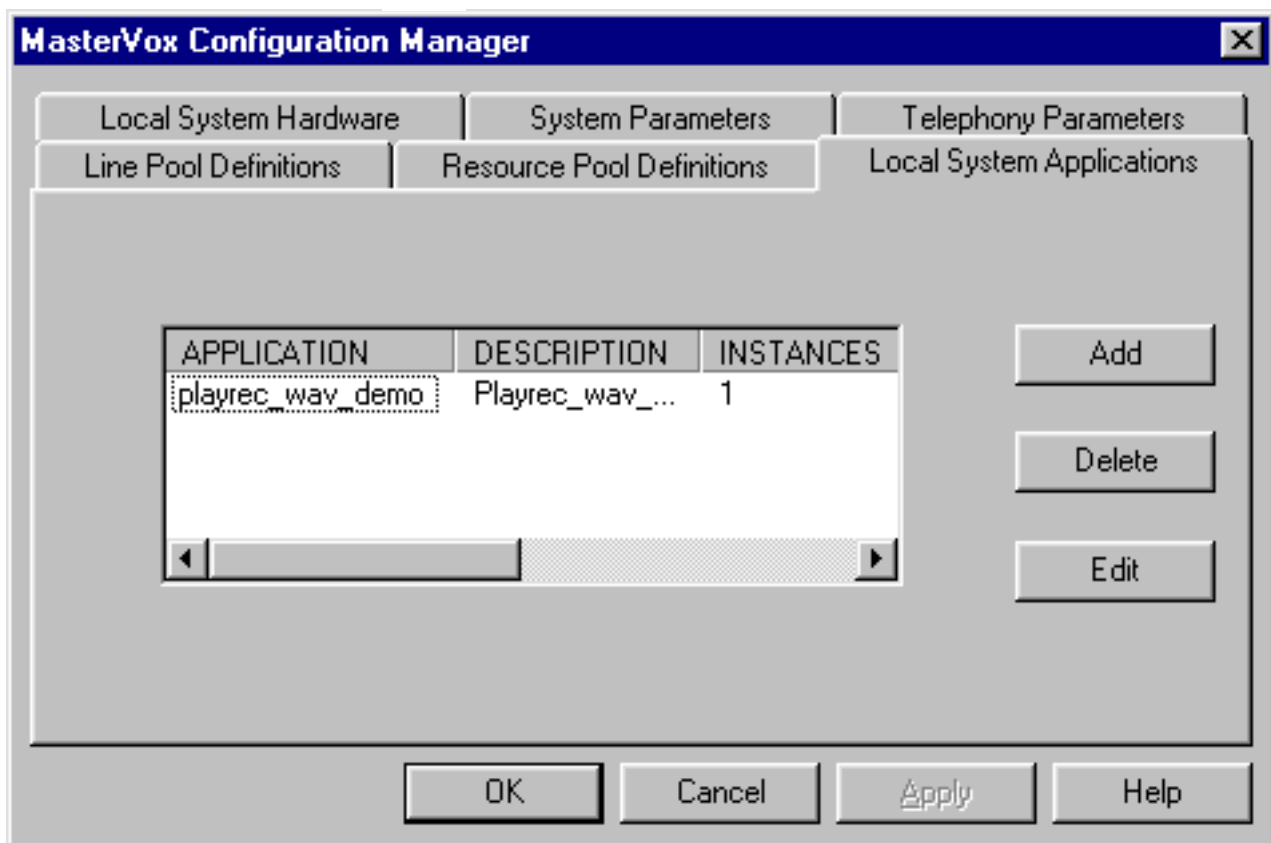
Chapter 5 Configure the Environment

Run the MasterVox Configuration Manager by selecting it from Programs/MasterVox Evaluation in the Windows Start Menu

START THE CONFIGURATION MANAGER

Run the MasterVox Configuration Manager by selecting it from **Programs/MasterVox Evaluation** in the Windows (NT or 95) **Start Menu** or by double clicking the MasterVox Configuration Manager icon in the MasterVox Evaluation program group.

Select the '*Local System Applications*' tab and confirm that **Playrec_wav_demo** is designated as the application. This tells the Engine which application to run at startup. We will be running the WAV demo so that you may hear the recorded prompts over your computer's speakers during the simulation. If you do not have a sound card and speakers installed on your computer, you will not be able to hear the messages. The **playrec_demo** we have been examining is identical in design and function to the **playrec_wav_demo**, except it uses VOX files (rather than WAV files), which require telephony hardware.



MasterVox Configuration Manager showing the Local System Applications Tab

Table of Contents



Chapter 5 Configure the Environment

*Among other things...
The MasterVox Configuration
Manager tells the system what
MasterVox Applications to run*

To execute some other application you have created, simply click on the **edit button** and enter the name of the MasterVox application you wish to run. The description field should be used for a meaningful description of the application being executed. This comment will appear in the MasterVox Engine Title Bar when you run your application.

In addition to indicating the application(s) to be executed by the MasterVox Engine, the 'Local System Applications' tab allows you to turn on/off diagnostic logging, the level of logging (there are 4 levels of diagnostic logging), and the frequency with which the log files are flushed (does not apply to simulation mode). Also applicable to real-time execution of MasterVox applications, from this tab you will indicate the number of instances of the application to start up, and which line and resource pools are to be used for the application.

The remaining tabs in the MasterVox Configuration Manager deal with defining your telephony hardware, breaking your resources into logical pools, and adjusting the telephony parameters related to your NMS hardware (this determines how your hardware communicates to you telephone carrier). Since these issues does not apply to the evaluation version of MasterVox, they will not be discussed here. Feel free to peruse the Configuration Manager in order get a feel for the extent to which you may configure the MasterVox environment.

[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)

[MasterVox Evaluation User's Guide](#)

Chapter 6

Record Voice Prompts

DO NOT use file extensions (.vox, .wav or .oki) when referencing voice files in Audiotex Actions

This step is not applicable to the MasterVox Evaluation Version since, not only are the prompts needed to run the demo applications provided with this evaluation, but also, without telephony hardware installed in your PC you will not be able to record prompts.

If you have a sound card installed in your computer, it is possible, to record WAV file messages with your PC and use them in other applications you may create with this evaluation version, since MasterVox is compatible with WAV type audio files. To do this you will need to use the **Sound Recorder** which came with your Windows Operating System, and you will need a microphone. Please note, the sound quality of recordings made this way will typically not be that good.

The Sound Recorder can usually be found by clicking the **Start Menu**, and selecting **Programs\Accessories\Multimedia**. Once the sound recorder is started, you will need to customize the Audio Properties of the file you will record to match the following:

File Format: PCM
File Attributes: 11,025 Hz, 16 Bit, Mono

Finally, to use your messages in your MasterVox Demo Applications, be sure to set the '**File Type**' parameter in your Audiotex Actions to 'PCM', and when referencing the file name, **DO NOT USE THE FILE EXTENSION**.

Finally, make certain that you place the files in the correct MasterVox sub-directory, usually **\Audio**.

[Table of Contents](#)



[Prev Pg](#)



[Next Pg](#)

MasterVox Evaluation User's Guide

Chapter 7

Run the Application

In order to play and record WAV files for use in this demo, it is necessary that you have a sound card and speakers installed in your PC

To start the MasterVox Engine... select it from /Programs/MasterVox Evaluation from the Windows Start Menu

Table of Contents



Prev Pg



In Telephony Simulation Mode the Engine will run on both Windows 95 and Windows NT without any telephony hardware - allowing you to do some of your development on a laptop or another computer that does not have any telephony hardware installed. Note that only the telephony actions are simulated in this mode – all other actions (including database actions) are executed normally (i.e., in “live” mode) by the Engine.

With the application ready to go, the Configuration Manager set up to run the correct application, and voice prompts in place, it is now time to run the demo application. In simulation mode, since telephony hardware does not play a part, you will use your mouse and an on-screen telephone keypad to interact with the application - as if you were a caller.

In simulation mode, as actions are executed, a dialog box pops up for each action asking for input from you regarding the outcome of the action - SUCCESS, FAILURE, etc. - and for input in cases where input is necessary (e.g. enter your account number or press 1 for....). By determining the outcome of any action, you can simulate all possible call flow paths in your application - without any telephony hardware. As for database interaction, all SQL related actions will execute normally in simulation mode. Please Note: Be sure to register your database as a data source with your ODBC application first !

To start the Engine, double click on the **MasterVox Engine icon** to launch the MasterVox Runtime Engine. The engine will come up in Telephony Simulation Mode.

In the Engine window each line that you have configured to run will be represented by one line in the Engine, and a **red phone**, indicating an *on-hook state*, will appear on each line.

To start a simulated call, double click on the red phone icon.

Next to the phone, on each line, the activity being performed will show in text - notice that the text appearing is the comment for each action.

You can also view the contents of your variables. To do this, right click on the activity text of the active line. A pop-up menu will appear allowing you to select the application variable contents and/or display a trace window that shows very detailed diagnostics information about each action as it is being executed.

Now try selecting “Application Statistics” from the “View” menu and watch the display change from the textual activity format to a graphical representation of application activity.