Meridian IVR

Application Development Guide

Publication number:	555-9001-310
Product release:	Meridian IVR 2.0/I
Document release:	Standard 1.0
Date:	February 1996

Printed in the United States of America

Information is subject to change without notice. Northern Telecom reserves the right to make changes in design or components as progress in engineering and manufacturing may warrant.

Publication history

February 1996

This document is the first standard issue for Meridian IVR release 2.0/I.

Contents

List of figures

List of tables

List of procedures

About this guide

Who should use this guide

This guide is written for Meridian IVR 2.0/I application developers who will be developing and customizing Meridian IVR applications.

This guide assumes that you are familiar with creating voice applications with Meridian IVR 2.0/I. You should also be familiar with “C” programming language and have a knowledge of telecommunications.

How to use this guide

This manual contains the following chapters and appendices:

Chapter 1: Understanding applications

Introduces you to the basic concepts of the Meridian IVR application.

Chapter 2: Understanding cells

Provides an overview of the components of a cell and introduces you to the basic cell types.

Chapter 3: Building applications

Explains the rules of designing and building effective applications.

Chapter 4: Creating a sample application

Teaches you how to create a sample application that you can load and run.

Chapter 5: Creating user functions

Teaches you how to create user functions, and provides an overview on how the user function process works.

Chapter 6: Using information databases

Explains the concepts of creating and editing information databases.

Chapter 7: Cell catalog

Describes the cell types in a Meridian IVR application. It lists the buffers used, buffers updated, cell parameters, next cells, and tables.

Appendix A: Standard prompts

Lists the standard Meridian IVR prompts.

Appendix B: Meridian ACCESS return codes

Lists the Meridian ACCESS return codes.

Appendix C: Sample cell chart

Provides you a sample cell chart that you can use for each cell if you want to plan your application on paper.

Appendix D: Converting applications

Shows you how to convert Meridian IVR applications from Release 1 to Release 2.

Appendix E: Upgrading databases

Shows you how to upgrade Meridian IVR databases from Release 1 to Release 2/I.

Additional Meridian IVR guides

You may find the following documentation useful while reading this manual.

Manual	NTP Number
Meridian IVR Product Guide	555-9001-010
Meridian IVR Planning and Engineering Guide	555-9001-200
Meridian IVR Installation Guide	555-9001-210
Meridian IVR Getting Started Guide	555-9001-301
Meridian IVR System Administration Guide	555-9001-300
Meridian IVR 3270 Gateway Development Guide	555-9001-312
Meridian IVR SQL Server Guide	555-9001-314
Meridian IVR VT100 Gateway Development Guide	555-9001-316
Meridian IVR 5250 Guide	555-9001-318
Meridian IVR Fax Application Guide	555-9001-350
Meridian IVR Maintenance and Diagnostics Guide	555-9001-500

Third-party documentation

The information supplied with your vendor’s host connectivity package will be necessary for hardware and software installation and configuration.

Conventions used in this guide

Throughout this guide, several typographic conventions have been used to highlight certain types of information.

- Buffer names are shown in all upper case characters, for example, the CURRENT MESSAGE buffer.
- Commands you must type are shown in bold, for example, type **sam** at the prompt.
- Keynames you press are enclosed in angle brackets, for example, the <Enter> key.
- Softkeys shown on the window which are mapped to function keys are enclosed in square brackets, for example, the [Save] softkey.
- Variables shown in command lines appear in italics, for example, the host_cfg*n* file, where *n* is a variable representing a board number.
- Screen output is shown in `courier`.

Chapter 1: Understanding applications

This chapter introduces you to the Meridian IVR application concepts and shows you how to integrate the elements of the application into an effective call flow.

The elements of a Meridian IVR application

To build a voice application with Meridian IVR, you need to use a variety of components that include the following:

- cells
- branches
- buffers
- information databases
- Direct Number Information System (DNIS) support
- Agent Whisper
- identification numbers for prompts and messages

A Meridian IVR application consists of cells that perform a set of interconnected activities during a telephone call. The cell types determine the activities you perform at each step.

You can run an application on one channel or several voice channels at the same time. The application usually begins by answering or making a telephone call, and ends by hanging up the call. What happens in between is entirely up to you as the application developer.

Cells

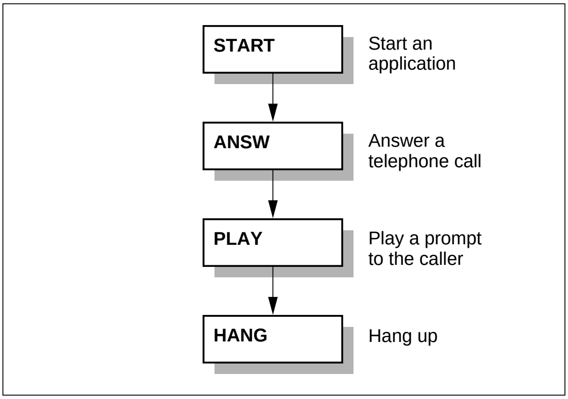
Cells are the building blocks of your application. Each cell performs an action. The cells are linked together by pathways. An application follows these pathways and performs a series of actions.

Meridian IVR cells can perform many different activities within an application. For each kind of activity, there is a cell type. When you are building an application, you choose cell types that perform the activities you want.

An example of an application

The following call-flow diagram, Figure 1-1, represents a simple application that answers a telephone call, plays a prompt to the caller, then hangs up the call.

Figure 1-1
Example of a simple application



In the call flow diagram, each rectangle represents a cell. Inside the rectangle is the cell type such as ANSW for Answer, PLAY for Play Prompts, and HANG for Hang Up. The cell at the top of the call flow is the first cell in the application. The arrows guide you from one cell to the next.

Branches

A branch is a pathway that connects one cell to another. Each branch includes a source and a destination connector (also referred to as an entry connector). A cell can have zero or many source connectors (that is, exception or success connectors), but only one destination connector. You can designate a branch from only a source connector.

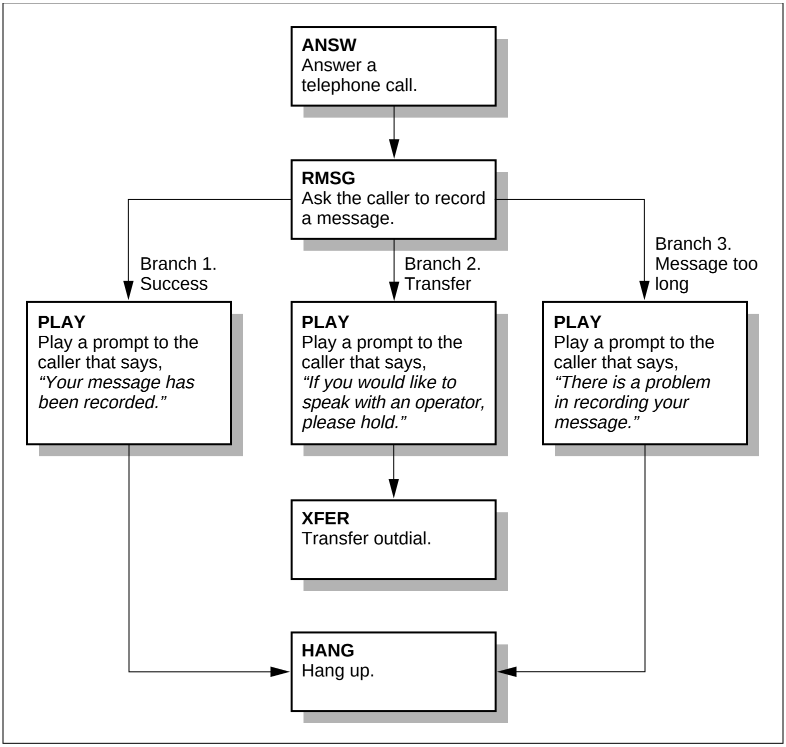
An application must be able to respond to unexpected results. For example, suppose an application asks a caller to leave a message. There is a cell type named Record Message (RMSG) that can do this. However, it is not sufficient to leave it at that; you need to anticipate what a caller can do.

The RMSG cell type has three branches to cover all of the possibilities:

- The caller can successfully leave a message.
- There can be a problem that prevents the system from recording the message.
- The caller can take too long to record the message.

As shown in Figure 1-2, each branch leads to a PLAY cell that plays a prompt appropriate to the situation.

Figure 1-2
An application with branching


ANSW
Answer a telephone call.
RMSG
Ask the caller to record a message.
Branch 1. Success
Branch 2. Transfer
Branch 3. Message too long
PLAY
Play a prompt to the caller that says, *"Your message has been recorded."*
PLAY
Play a prompt to the caller that says, *"If you would like to speak with an operator, please hold."*
PLAY
Play a prompt to the caller that says, *"There is a problem in recording your message."*
XFER
Transfer outdial.
HANG
Hang up.

Buffers

Meridian IVR uses buffers to store information and pass it from one cell to another. The information stored in a Meridian IVR buffer is a string of up to 31 characters. These characters can be letters, numbers, spaces, or any characters that you type on a keyboard, except for commas.

For example, a cell in an application can ask a caller to enter a password by pressing the keys on the telephone. If the caller presses the 3, 7, 2, and 9 keys, the cell accepts the number 3729 and stores it in a buffer. Now that the password is in a buffer, any other cell can use it. You can add another cell that validates the password.

You can use two different kinds of buffers in your application: system buffers that Meridian IVR creates for you, and user-defined buffers that you create yourself.

System buffers

A set of predefined system buffers is available to each application during each phone call. Each buffer is designed for a special purpose; for example, the buffer called PASSWORD is generally used to store a password that the caller enters. However, you are not restricted to use it for this purpose. Generally, you can use buffers in any way you choose.

The following is a list of system buffers and an explanation of their typical use.

ANI DIGITS

The number of the person who is calling the application.

CURRENT MESSAGE

The identification number of the current message.

DATA EXCHANGE #1

Information that a DELV cell can pass to a scheduled application. You can also this buffer to store generic data during an application.

DATA EXCHANGE #2

Information that a DELV cell can pass to a scheduled application. You can also use this buffer to store generic data during an application.

DATA EXCHANGE #3

Information that a DELV cell can pass to a scheduled application. You can also use this buffer to store generic data during an application.

DATA EXCHANGE #4

Information that a DELV cell can pass to a scheduled application. You can also use this buffer to store generic data during an application.

DATA EXCHANGE #5

Information that a DELV cell can pass to a scheduled application. You can also use this buffer to store generic data during an application.

DATA EXCHANGE #6

Area where you can store generic data during an application.

DATA EXCHANGE #7

Area where you can store generic data during an application.

DATA EXCHANGE #8

Area where you can store generic data during an application.

DATA EXCHANGE #9

Area where you can store generic data during an application.

DATA EXCHANGE #10

Area where you can store generic data during an application.

DATE

The current date in the form *mmddyyyy*.

DAY

The numeric form of the current day of the week, where 1 is Monday and 7 is Sunday.

DELIVERY ATTEMPTS

The number of times a delivery application should try to make the delivery.

DELIVERY EVENT ID

An identification number for a specific delivery event.

DELIVERY INTERVAL

The length of time between delivery attempts.

DELIVERY TIME

The scheduled date and time for delivery.

DIGITS

The number being called.

LENGTH

The length of the string in a buffer.

MAILBOX ID

The identification number of a mailbox which stores messages. The mailbox ID is often a phone number or extension.

NUMBER OF ANI DIGITS

The number of digits in the ANI DIGITS buffer.

NUMBER OF DIGITS

The number of digits in the DIGITS buffer.

NUMBER OF MESSAGES

The number of messages currently stored in a particular mailbox.

PASSWORD

A password associated with a mailbox.

SENDER ID

The identification of the message sender.

TERMINATION DIGIT

The terminating digit from the digit string requested in the GDAT cell.

TIME

The current time in the form of *hhmm.*

Certain system buffers that are automatically updated by Meridian IVR can also be referred to as system-defined buffers.

The following buffers are automatically updated by the corresponding calls. The DIGITS and NUMBER OF DIGITS buffers are updated in the MENU cell. The CURRENT MESSAGE buffer is updated in the RMSG cell and is used by the PMSG and DMSG cells.

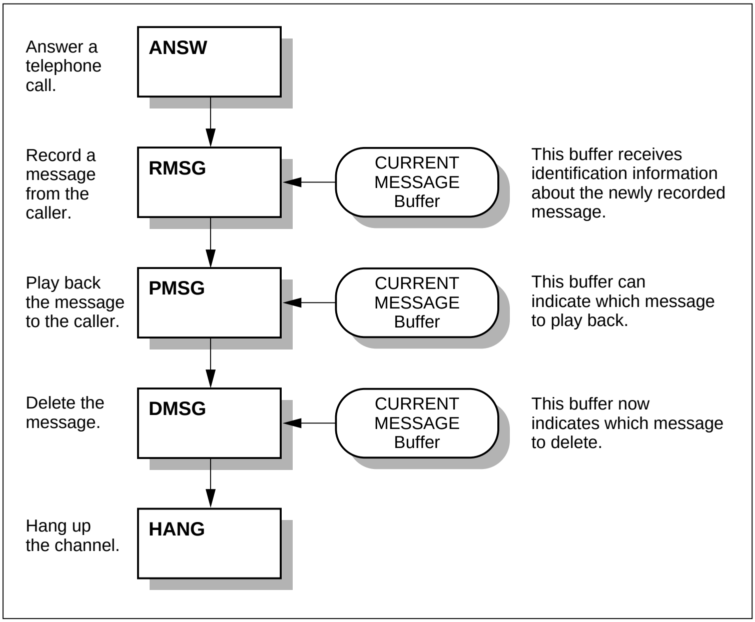
Two of the system buffers deserve special attention: DIGITS and ANI DIGITS. At the beginning of a call, before the application executes the first cell, Meridian Mail can receive digits that represent the called number and the calling number. Meridian IVR automatically takes the called number and stores it in the DIGITS buffer. The number of digits in the DIGITS buffer is automatically stored in the NUMBER OF DIGITS buffer.

Meridian IVR takes the calling number and stores it in the ANI DIGITS buffer. The number of digits in the ANI DIGITS buffer is automatically updated in the NUMBER OF ANI DIGITS buffer. If the Meridian 1 PBX is connected to the telephone network with ISDN trunks, the ANI DIGITS buffer contains Calling Line Identification (CLID) of the caller. For non-ISDN calls, the ANI DIGITS buffer contains the trunk route access code and the trunk route member number. For internal calls, the ANI DIGITS buffer contains the Directory NUMBER (DN) of the caller.

The following call flow, Figure 1-3, shows an application that uses system-defined buffers. This application answers a telephone call, records a message from the caller, plays the message back to the caller, erases the message from Meridian Mail, and hangs up the call.

In this call flow, each oval represents a buffer. An arrow leading from a buffer to a cell indicates that the cell is using the information in the buffer. A black arrow is the pathway from one cell to the next.

Figure 1-3
An application using buffers


Answer a telephone call.
ANSW
Record a message from the caller.
RMSG
CURRENT MESSAGE Buffer
This buffer receives identification information about the newly recorded message.
Play back the message to the caller.
PMSG
CURRENT MESSAGE Buffer
This buffer can indicate which message to play back.
Delete the message.
DMSG
CURRENT MESSAGE Buffer
This buffer now indicates which message to delete.
Hang up the channel.
HANG

User-defined buffers

Meridian IVR allows you to use as many as 50 buffers of your own in any single application. You can create a user-defined buffer at any point in the application. Once it is created, the user-defined buffer is available to all the cells in the application.

There are three reasons why you may want to create user-defined buffers:

- You find that your application needs many buffers, and there are not enough system buffers for your purposes.
- You want to have a buffer whose name reminds you of its specific function in the application, and none of the system buffers seems appropriate.
- You want to have a buffer that is entirely under your control. Some cell types use buffers automatically. However, you can prefer just to create a buffer that you know will not be used unless you explicitly reference it.

If you are not a programmer, you only need to remember that a buffer can store up to 31 characters. These characters can be letters, numbers, spaces, or any of the other characters that you can type on a keyboard except commas.

If you are a programmer, you need to remember that each buffer contains a null-terminated string of up to 31 characters which can be any ASCII character except commas.

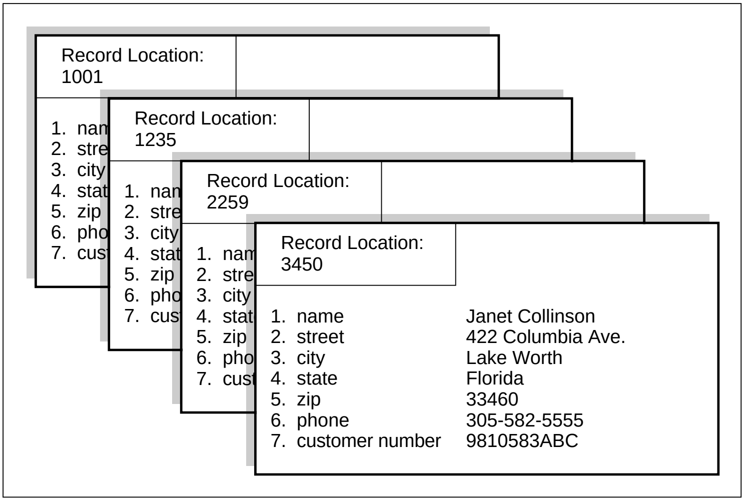
Buffers are created by accessing a cell where a buffer resides and renaming the buffer.

Information databases

An information database consists of a set of records, each of which stores strings of characters. Figure 1-4 illustrates an information database. You can create more than one information database. Up to 20 information databases can be in use by all of the applications that are running at any one time.

To use the information, an application can have a Check database cell (CHEK) to copy data from one record to a set of buffers you have chosen. Once the data has been placed in buffers, the application can use the data just as it would any other buffer data. For example, the application could look up a given record in a database, copy a phone number from the record to a buffer in the application, and then read the phone number to the caller.

Figure 1-4
Records in an information database



Directory Number Identification System

Directory Number Identification System (DNIS) identifies to the called system the last three or four digits of the number dialed by the caller.

One example of the use of DNIS is with 800 numbers. A company has set up several 800 numbers, one for testing your advertisements on TV stations in Phoenix, another for testing your advertisements on TV stations in Chicago, and another for Milwaukee. A Local Telephone Carrier terminates all the lines in one group to your IVR ACD because it is cheaper and more efficient to run one group of incoming lines. The calls are all answered by IVR applications, and the DNIS for each call directs the call to the appropriate application, which is invoked as a subroutine within an application.

The DNIS is stored in the system buffers of an application. These system buffers are DIGITS and ANI. At the beginning of a call, before the first cell in the application is executed, Meridian Mail can receive DNIS information regarding the called and calling numbers. Meridian IVR automatically takes and stores the called number information in the DIGITS buffer and stores the calling number information in the ANI buffer.

Agent Whisper

Agent Whisper allows the system to play a prerecorded phrase and/or convey call-related information to an agent.

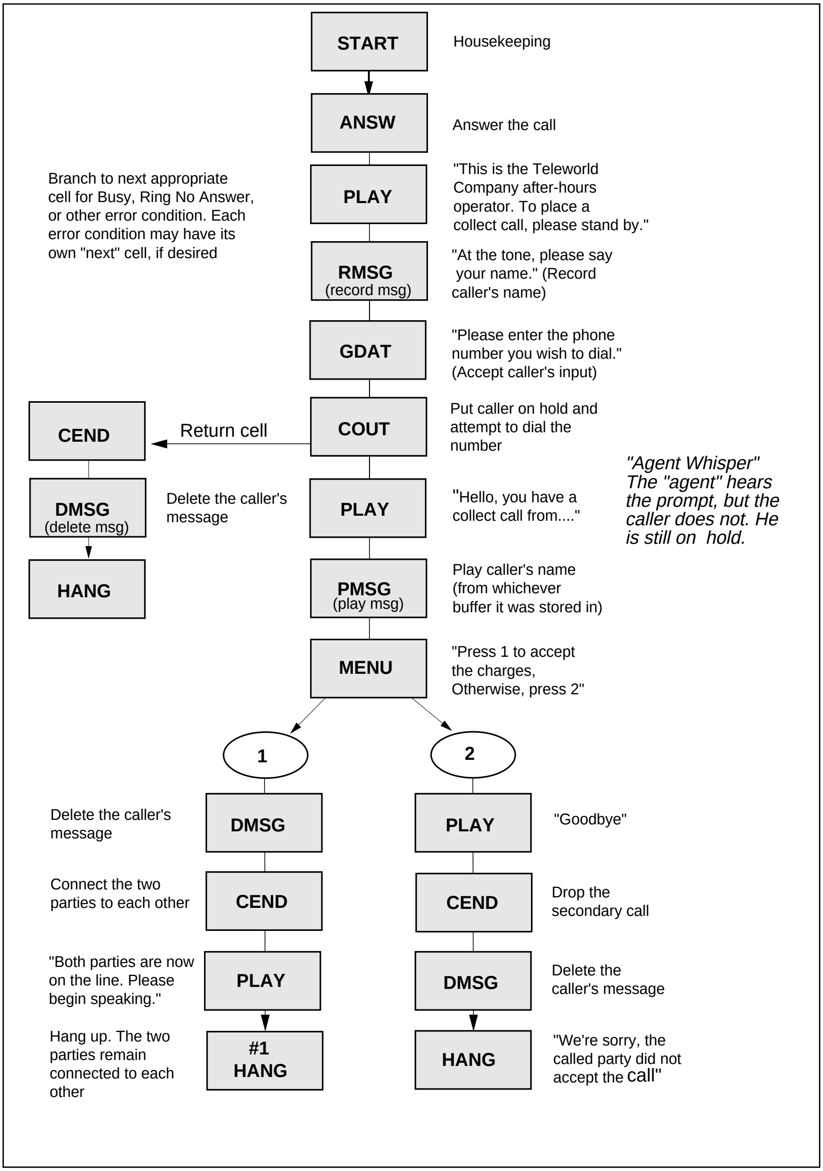
Agent Whisper is most commonly used in collect calling. A person placing a collect call is prompted to say their name into the phone where it is recorded. Then the call is processed and rings the designated number. When the phone is answered, a recorded voice informs whoever answers the phone that there is a collect call from “Name”, and then asks them to accept the collect charge. When the computer plays back the recording of the caller’s name, the caller does not hear this. This is why it is called Agent Whisper.

Agent Whisper is also used to relay information about a call to a call attendant or agent. This could be information about a series of menu choices that a caller has selected so that the Agent is prepared for the call when it is transferred. The information that is conveyed is application dependent. This information can be the DNIS number, the nature of the call, or the length of time that a caller has been waiting.

Prompts and information are relayed to the agent using a PLAY cell between the Continuation Outdial (COUT) and Continuation Outdial End (CEND) cells. Only the agent hears what the PLAY cell plays, not the caller.

Figure 1-5 shows an application that uses Agent Whisper.

Figure 1-5
Using Agent Whisper in an application


START
Housekeeping
ANSW
Answer the call
Branch to next appropriate cell for Busy, Ring No Answer, or other error condition. Each error condition may have its own "next" cell, if desired
PLAY
"This is the Teleworld Company after-hours operator. To place a collect call, please stand by."
RMSG
(record msg)
"At the tone, please say your name." (Record caller's name)
GDAT
"Please enter the phone number you wish to dial." (Accept caller's input)
CEND
Return cell
COUT
Put caller on hold and attempt to dial the number
DMSG
(delete msg)
Delete the caller's message
PLAY
"Hello, you have a collect call from...."
"Agent Whisper" The "agent" hears the prompt, but the caller does not. He is still on hold.
HANG
PMSG
(play msg)
Play caller's name (from whichever buffer it was stored in)
MENU
"Press 1 to accept the charges, Otherwise, press 2"
1
2
Delete the caller's message
DMSG
PLAY
"Goodbye"
Connect the two parties to each other
CEND
CEND
Drop the secondary call
"Both parties are now on the line. Please begin speaking."
PLAY
DMSG
Delete the caller's message
Hang up. The two parties remain connected to each other
#1
HANG
HANG
"We're sorry, the called party did not accept the call"

Identification numbers for prompts and messages

All voice applications must manage two kinds of voice recordings: prompts and messages. Prompts are played to callers to guide them through applications created in advance using the Voice Prompt Editor (VPE). Messages are created by callers and typically have a short life-span.

When a message is recorded, Meridian IVR automatically assigns a number to the message. You never need to know the number of a message because Meridian IVR takes care of message numbers automatically.

When you want to use prompts, you need to know what prompt numbers to use. Refer to the VPE to view the available prompt numbers.

Chapter 2: Understanding cells

This chapter explains how you can use cells to create your application. It presents an overview of the basic functions of the cell. (A complete description of each cell type appears in Chapter 7.)

There are two things about cells you need to remember:

- All cells have a common structure.
- Meridian IVR has many cells from which you can choose when you build an application.

Components of a cell

As mentioned before, cells are the basic building blocks of an application. There are six components of a cell:

- cell type
- cell name
- cell number
- parameters
- next cells
- tables

Cell type

Each cell has a cell type that determines the kind of activity the cell performs, such as answering a phone call or playing a message. The cell type determines the basic framework for building a cell. When you want to create a cell, you choose a cell type to establish its basic characteristics, then customize it as necessary. You can select from a variety of cells to create your application.

Cell name

When you create a new cell for your application, Meridian IVR automatically supplies the default cell name, *Untitled*, that you can and should change. It is important to choose a meaningful name for each cell so that you remember what each cell does.

Cell number

The cell number is used for identification and error tracking. Each new cell is assigned a cell number identifying the order in which it was created. The first cell is #1, the next is #2, and so forth. You can see the cell numbers when you print your application, or when you access the “dapple” utility, which is executed through the UNIX shell window, or the application editor. Refer to the *Meridian IVR Maintenance and Diagnostics Guide* (555-9001-500) for more information on this utility.

Parameters

For each cell type, there is a set of parameters to determine exactly how the cell performs its functions. The PLAY cell type is used to play one or more prompts to a caller. The following shows the list of parameters for the PLAY Prompts cell type.

Parameters for the PLAY cell type

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment Field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call auditing Process logs the content of this buffer to the audit_stat.d file.
Number of Prompts (max 32)	1	This value is updated automatically once the cell is saved with the count of the number of prompts entered in the Prompt Table. ***Note:*** If a buffer instead of a value is entered, the value in the buffer will indicate the number of prompts to play when the cell is executed in an application.

When you create a cell, it is supplied with predetermined initial values for its parameters. For example, when you first create a PLAY cell, it has a value of 1 for its Number of Prompts parameter. The initial values provided for each parameter are reasonable values which you are likely to use; they help you save time when building an application.

If a parameter’s initial value is not what you want, you can easily change it. Making changes to parameter values is one of the steps you take when customizing the cell.

Next cells

For each branch in a cell, you select a next cell which is the cell that follows if the application takes that particular branch. For example, the PLAY cell has two next cell branches called "Error" and "Success." The application takes the Error branch if the PLAY cell cannot play the designated prompts. It takes the Success branch if the PLAY cell plays the prompts.

When you create a cell, it has initial values for each of its next cells, just as it has initial values for each parameter. The initial value for each next cell, however, is always Cell 1. In most cases, you select some other value because every cell cannot branch to Cell 1. For this reason, we recommend that you use the HANG Cell as the first cell when you create your application to avoid unusual results, in case you forget to change the next cell default somewhere in your call flow. Changing the initial next cell values is another step in customizing the cell.

Tables

Some cells require tables of information. For example, the PLAY cell can play up to 32 prompts to a caller. Because you must indicate which prompts you want the PLAY cell to play, it has the Play Prompt Table, where you can list the prompts. Like parameters and next cells, tables have initial values. As a further step in customizing the cell, you can change the initial table values, if necessary

Cell categories

Cells are categorized according to their main function. The categories include: prompt, outdial, buffer manipulation, call management, application scheduling, user function, execute application, subroutine, message manipulation, and database cells.

Audiotext cells

Audiotext cells contain prompts that guide callers through an application. These first two cells play prompts and solicit input from the caller.

Play Prompts and Get Data (GDAT)

Plays one or more prompts and collects information from the caller. For example, a GDAT cell can play the prompt (for example, "Please enter your password") when the caller supplies the password by pressing telephone keys. GDAT automatically collects the digits and stores them in a buffer.

Play Menu and Get Data (MENU)

Plays a menu to the caller (for example, “Press 1 to save this message. Press 2 to review it. Press 3 to delete it”). When the caller presses a digit, MENU branches to a cell that performs the activity that the caller chooses.

The next two cells only play prompts.

Play Prompts with Data (PDAT)

Takes a number from a buffer and reads it to the caller. PDAT can read the number in any of the following formats: a date, a time, a date and time, a number, or a series of digits. For example, if the format for “1234” is number, PDAT plays a series of system prompts that say “One thousand, two hundred thirty-four”. If the format for “1234” is digits, PDAT reads this number as “one two three four.”

Play Prompts (PLAY)

Plays one or more prompts to the caller (for example, “Hello. Thank you for calling XYZ Corporation”).

Call management cells

These cells help to manage phone calls and are generally used to begin or end a call.

Answer (ANSW)

Answers a telephone call.

Hang Up (HANG)

Ends a call by hanging up the channel.

Get Call Information (INFO)

Gets detailed information on either incoming or outgoing calls.

Call Progress Detection (PRGS)

Gets the termination state of the previous outdial attempts by COUT and DOUT cells.

Data manipulation cells

Buffers store information and pass it from one cell to another. Many cells use the information that is stored in named buffers, but only data manipulation cells are designed specifically to allow you to access buffers directly.

User defined matching criteria (CASE)

Compares the contents of a buffer to a selection of predefined criteria, and passes the control to the branch that meets the criteria.

Convert data (CDAT)

Converts data from one format to another by allowing you to specify the input format, output format, source buffer, and destination buffer.

Compare (COMP)

Makes a numerical or string comparison of either the contents of one buffer or a constant value against the contents of another buffer. It branches according to the results of the comparison: if the contents are equal, the buffer branches to an equal cell, or else it branches to the greater than or less than cell. COMP also provides a mechanism for looping within an application.

Concatenate Buffers (CONC)

Joins the contents of one buffer to the contents of another buffer, and puts the result into the second buffer.

Log Application Event (EVENT)

Logs application-specific information at arbitrary points within a call flow as directed by the application writer.

Perform Mathematical Operation (MATH)

Performs a mathematical operation (for example, add, subtract, multiply, divide, or remainder) on two operands that are either buffers or constants.

Store (STOR)

Puts either a value or contents of a buffer into another buffer.

Substring Buffers (SUBS)

Copies a portion of a string and puts the copy into a buffer.

Access user function cells

You can use these cells to obtain call information, address and deposit messages in Meridian Mail mailboxes, and detect the state or progress of a call.

Address Message (ADDR)

Addresses messages by the Meridian mailbox numbers.

Send Message (SMSG)

Sends previously recorded and addressed messages.

Execute application cells

An EXEC cell in an application makes another application begin running in place of the current one. EXEC makes it possible to break a large application into several smaller ones. When a call is received, it can be answered by an application that determines which of the several other applications should take the call. An EXEC cell can then turn the call over to the appropriate application.

Execute Application (EXEC)

Begins the execution of another Meridian IVR application on the current channel. Once the new application begins, it takes control of the current call and the old application stops running.

Call a Subroutine (GSUB)

Redirects execution to a new cell in either the current application or another application. The GSUB cell works similarly to the EXEC cell except that you can return to the calling application by using the RETN cell.

Return from a Subroutine Call (RETN)

Returns control after processing a GSUB cell to the GSUB cell’s next cell.

Fax response cells

These cells allow an IVR application to send and receive faxes.

Call back fax (CFAX)

Queues a Call Back fax to be sent to a specified phone number.

Data file create/concatenate (DCAT)

Performs data manipulation on files and buffers.

Data file delete (FDEL)

Deletes one or more files.

Receive fax (FRCV)

Transfers control to a fax modem to perform a Same Call receive from the caller.

Fax modem release (FRLS)

Attempts to release a fax modem that was previously reserved by FRSV. Depending on its success at releasing the fax modem, the cell is followed by either a success branch, or the error branches “bad id” or “fax down.”

Fax modem reserve (FRSV)

Attempts to reserve a fax modem for use in either a Same Call send or a Same Call receive. Depending on its success at reserving a modem, the FRSV cell is followed by either a success branch, or the error branches “no modems” or “fax down.”

Send fax (FSND)

Transfers control to a fax modem to perform a Same Call send to a caller.

Host communication cells

These cells add host communication support.

Host Communication Abort (COMA)

Aborts a transaction that is currently in progress from a host computer connected to Meridian IVR through the application developer.

Input to host (COMI)

Passes data to a host computer connected to Meridian IVR and initiates a transaction through the application developer.

Output to host (COMO)

Retrieves data from a host computer connected to Meridian IVR and initiates a transaction through the application developer.

Outdialing cells

So far, we have discussed only applications that receive telephone calls. Meridian IVR can also originate calls through the following outdial cells:

Continuation Outdial End (CEND)

Ends a secondary call originated by a Continuation Outdial (COUT) cell. CEND can end the call in one of these ways: CEND can hang up the secondary call and resume the original phone call, or join the two callers together in a conference call using the Meridian voice channel.

Continuation Outdial (COUT)

Places the caller on hold, then makes a secondary call on the same channel. This secondary call is ended by a Continuation Outdial End (CEND) cell.

Dial Digits (DIAL)

Dials DTMF digits during a call that is already in progress. This is useful for sending digits to an external device such as a paging terminal.

Directed Outdial (DOUT)

Allows an application to originate a phone call. After it dials, it checks to see if the call is answered and branches accordingly. You can use this cell in your application to deliver faxes and make wake-up or reminder calls.

Supervised Call Transfer (SXFR)

Makes a phone call for a caller who is already connected to Meridian IVR. SXFR dials the second party's phone number, transfers the original caller to the new call, then releases both parties. This cell differs from the XFER cell because if the transfer is unsuccessful, the application takes an error branch and continues with the current caller.

Transfer Outdial (XFER)

Makes a phone call for a caller who is already connected to Meridian IVR. XFER dials the second party’s phone number, transfers the original caller to the new call, then releases both parties.

Application scheduling cells

You can use these cells to schedule applications to run at a specific time. While you can schedule any kind of application, these cells are particularly useful for wake-up and reminder calls.

Schedule Application Delivery (DELV)

Schedules an application to run at a specified time. You can schedule the application to run several times.

Get Schedule Event List (LDLV)

Retrieves information about a scheduled event.

Get Meridian Mail Date and Time (MDTE)

Gets the date and time as they are set in Meridian Mail.

Get Current Date and Time (TIME)

Gets the current date, time, and day of the week as maintained by the UNIX operating system on the application module.

Unscheduled Delivery (UDLV)

Cancels a scheduled event.

User function cells

User functions, which are customized code written in the “C” programming language, can be used to extend the range of activities performed by Meridian IVR. This cell type interfaces a user function to an application.

Connect User Function (USER)

Acts as a bridge between a user function and an application.

Message cells

These cells enable you to record and manipulate messages from callers. The message itself is not in the CURRENT MESSAGE buffer. The buffer contains a reference to the message (for example, the message ID).

Delete Message (DMSG)

Deletes the message referenced in the CURRENT MESSAGE buffer.

Play a Message (PMSG)

Plays the message referenced in the CURRENT MESSAGE buffer.

Record Message (RMSG)

Lets the caller record a message. The message identification number is stored in the CURRENT MESSAGE buffer.

Local database cells

You can use the database cells to access data stored in a Meridian IVR information database.

Check Database (CHEK)

Checks an information database for a record associated with a particular record number and retrieves the corresponding data.

Delete Records from an Information Database (DDEL)

Allows you to delete a record from an information database.

Insert record into an Information Database (DINS)

Allows you to insert a record in an information database.

Update Information Database (UPDT)

Allows you to insert data in an information database. You can insert up to ten buffers of data in a record.

Multilingual cell

The following cell allows Meridian IVR to support multiple languages.

Set Language (LANG)

Sets the offset for system- and application-specific prompts, and associates the offset with a particular language. You can use this cell to support multilingual applications.

SQL server cells

The SQL server cells allow you to manipulate data stored in an SQL database.

SQL Select Count (QCNT)

Tells you how many records exist in a database table.

SQL Delete (QDEL)

Removes one or more records from a specified database table.

SQL Insert (QINS)

Adds a record to a specific database table.

SQL Select (QSEL)

Retrieves a record of data from a database table.

SQL Update (QUPD)

Updates the values in one or more records in a database table.

Chapter 3: Building applications

This chapter explains the rules for designing applications with the Application Editor. Before you begin to build applications, we recommend that you review Chapters 1 and 2 to learn the basics of applications and cells.

The Application Editor is based on the Motif and the X Window System. By using this editor, you can significantly shorten the time required to develop effective applications. While you are designing your application, the Application Editor simultaneously builds the underlying components and structure. After planning the design of your application on paper, you can lay it out on the drawing board. All of the design tools you need are right at your fingertips.

You can design call flows quickly and efficiently with the mouse. Documenting your design is as simple as saving it to disk or printing it. On-line help is available for immediate answers to your questions about Meridian IVR. In addition, as you update your design, the Application Editor automatically builds an executable file.

Note: This chapter highlights the features of the Application Editor. For further information about the Meridian IVR interface and the Motif user environment, refer to the *Meridian IVR Getting Started Guide* (NTP 555-9001-302). The following sections will help you get started:

- “Designing an application” on page 4-3
- “Selecting prompts” on page 4-5
- “Application Editor” on page 4-5
- “Understanding the basic components” on page 4-8
- “Opening an application” on page 4-26
- “Default values for cell parameters” on page 4-29
- “Arranging cells on the drawing board” on page 4-32
- “Connecting cells” on page 4-38
- “Moving cells on the drawing board” on page 4-44
- “Removing cells from your application” on page 4-47
- “Saving your application” on page 4-50
- “Printing your application” on page 4-51
- “Executing your application” on page 4-56
- “Deleting an application” on page 4-59

Designing an application

Before you begin to build an application, you should sketch the general call flow on paper. The following guides you through the process of creating a call-flow diagram that shows all of the cells in the application.

- Turn to Chapter 4 and review the section “Creating and naming cells” on page 4-11 for an overview of the cell types you can use for your application.
- Turn to Chapter 7 for a complete description of the cell types that you have chosen, and make sure that they are appropriate.
- Sketch a call-flow diagram that shows the cells and the branches from one cell to another. Chapter 7 describes the branches that each cell type can take.
- Give each cell a meaningful name.
- Show the buffers that will be used or updated. Chapter 7 explains how each cell type uses and updates buffers.

Now that you know how your application looks, you can select the prompts you want to use.

Note: For performance reasons, we recommend that an application not exceed 300 cells. If more than 300 cells are required, you should use multiple application files. Refer to GSUB, EXEC, and RETN cells for information on connecting application files.

ATTENTION!

The following is a list of restrictions and limitations regarding the cleanup branch that you must consider when designing your application:

You should avoid using telephony cells, except DMSG, in the cleanup branch.

Do not use the DMSG cell in the cleanup branch when you use shared channels on a Messaging Service Module (MSM).

You should consider timing issues very carefully when developing applications for IVR systems connected to MSMs. The application should take less than 15 seconds to finish after a hang-up occurs. If this condition is not met, calls can be lost and sent to a Revert DN, and, in the case of dedicated channels, channel acquisition is lost.

This means that an application on an MSM that has any type of cell that takes more than 15 seconds to execute will cause calls to be lost (telephony cells excepted).

Selecting prompts

Certain prompts such as “Hello” are used often in applications; therefore we recorded and supplied a set of them for you. These prompts, which we call *standard prompts* (or system prompts), can be all you need for your application. If not, you can compose and record any additional prompts you need.

- Turn to Appendix A and examine the list of standard prompts. Select the ones you need, and write their prompt numbers on your call-flow diagram.
- If additional prompts are required, compose and record them. See the *Meridian IVR System Administration Guide* (NTP 555-9001-300) for instructions on using the Voice Prompt Editor interface, or consult your system administrator.

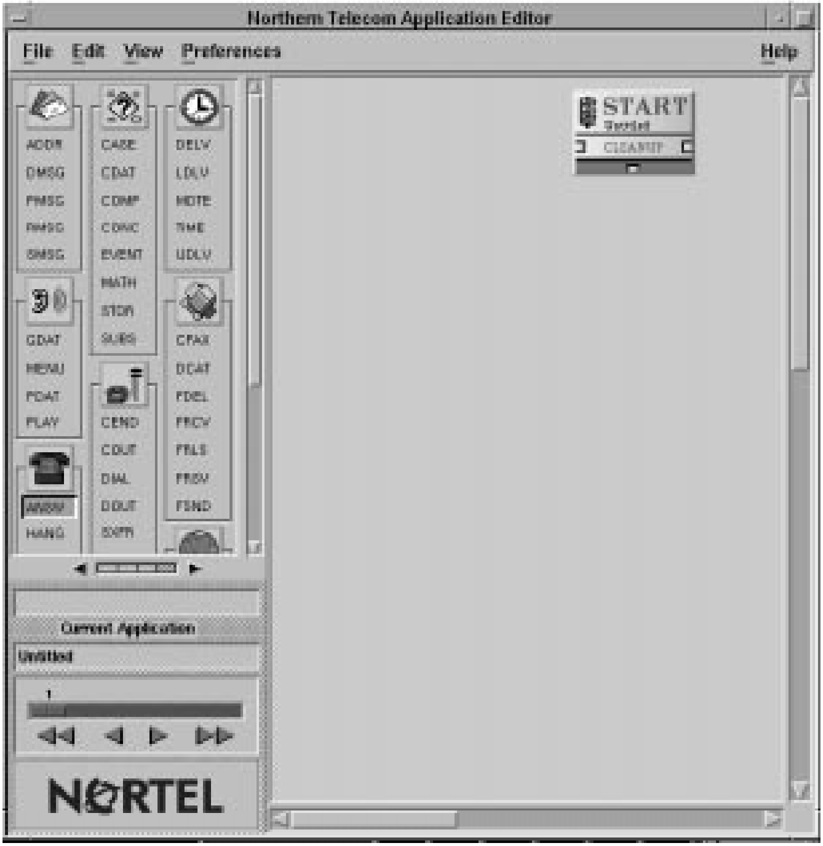
Application Editor

Procedure 3-1
Accessing the Application Editor

1 Click on the Application Editor “call-flow icon” with the left mouse button.

The Application Editor opens on the desktop as shown in Figure 3-1.

Figure 3-1
Application Editor



The layout of this window consists of four primary components: the Cell palette, Drawing board, Control area, and Menu bar.

The cell palette is the “toolbox” of available cells, organized by function. The drawing board is the area for laying out the design of your application. The control area includes push buttons and a slider for moving to different pages of the drawing board. Once you open an application, the status window displays which application is on the drawing board and which cell has been selected with the mouse. This chapter describes each component in detail.

Using the mouse in the Application Editor

You can use the mouse to perform several functions in the Application Editor (see Table 3-1).

Table 3-1
Mouse functions for the Application Editor

Button	Action	Application Editor Function
Left mouse button	Click	Selects a cell or icon. Connects the cells. Deselects a cell or icon by clicking anywhere on the Application Editor. To deselect the pencil, point to the “START” (default) cell and click on it with the left mouse button.
	Double-click	Opens the parameter window for the selected cell. Jumps to the page on which a destination cell resides.
	Drag	Moves a slider. Expands/reduces a window. Marks text in a text entry box.
	Hold	(On a scroll bar) scrolls through a list.
Middle mouse button	Click	Places a selected cell on the drawing board. If you do not select a cell on the Drawing Board (that is, the Status Window is blank), you can use the middle mouse button to highlight the cell in the cell palette.
	Drag	Moves a selected cell to another location on the drawing board.
Right mouse button		None.

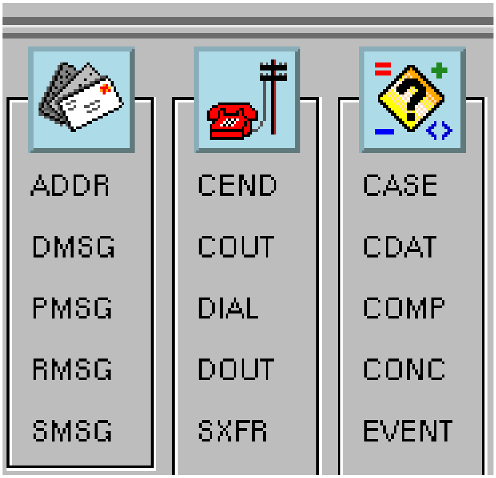
Understanding the basic components

Before you begin using the Application Editor, you should become familiar with how you can use the window components and options to design your application. This section introduces you to the basic features of the Application Editor and explains what you need to begin designing your applications.

Using the cell palette

The cell palette contains all of the cells that you can use to develop a voice application. As you can see in the section of the cells palette shown in Figure 3-2, the cells are organized by function. Each cell function is explained in the following pages.

Figure 3-2
Cell palette



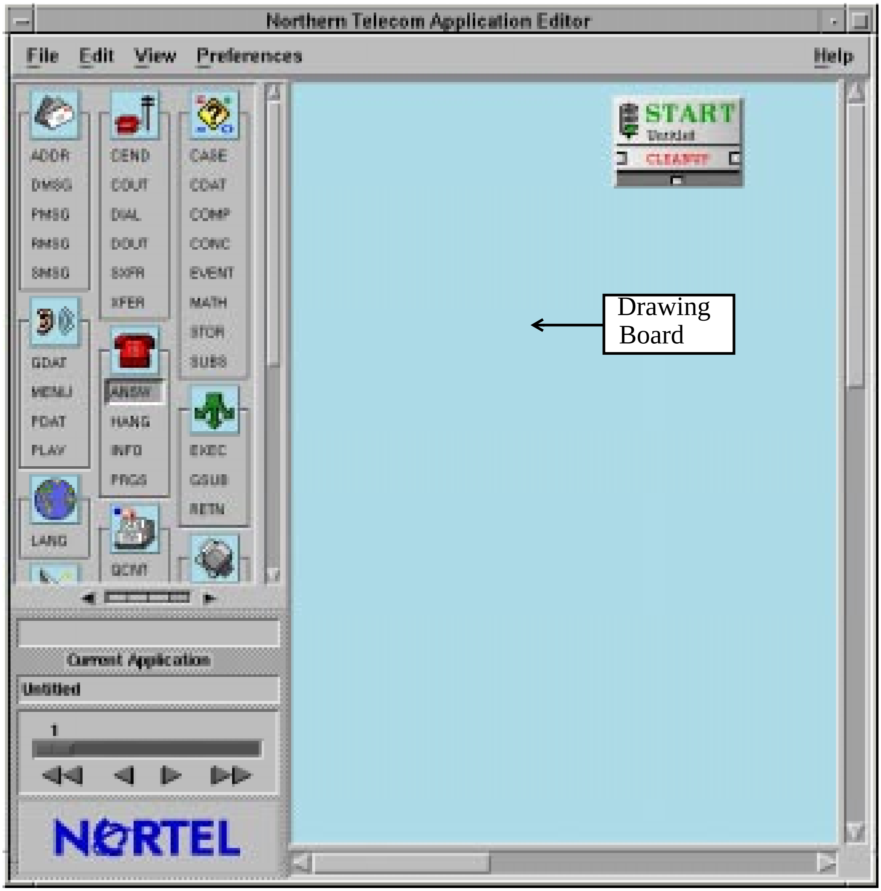
Using the drawing board

The drawing board is the area on the screen where you can design your call flow. As you draw your design, Meridian IVR simultaneously constructs the underlying structure of cells and branches.

As shown in Figure 3-3, the drawing board always includes a START cell which provides an anchor point from which to begin your design, and which sets all application defaults. You can use the scroll bars on the right side and bottom of the window to display other portions of the current page of the drawing board. The drawing board can have one or more pages.

> ***Note:*** When you print an application (see “Printing your application” in this chapter), each drawing-board page prints as one 8.5 x 11 in. page.

Figure 3-3
The drawing board



Current Application
Untitled

Hold down the left or middle mouse button, and drag the scroll bar to move either up or down the drawing board.

> ***Note:*** To scroll up or down in UNIX shell windows, click on the middle mouse button and drag the mouse on the scroll areas. To scroll one page at a time, place the pointer in the white scroll area and click the left mouse button.

Using the control area

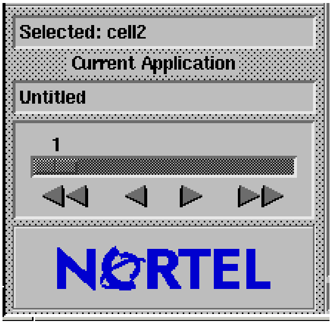
The control area includes a status window that identifies the cell you select, a selection box that lists the name of the current application, and controls you can use to move to another page of the drawing board. In Figure 3-3, the status window, located in the bottom left corner, indicates that no cell on the drawing board has been selected. The application displayed on the drawing board is a new application with the default name “Untitled,” and the Application Editor is displaying page 1 of the drawing board.

The status window

The status window, as shown in Figure 3-4, provides you with immediate feedback on what is happening with the Application Editor.

When this window is blank, the Application Editor is essentially in a neutral state waiting for you to initiate an activity. If you click on a cell with the left mouse button, the Status Window indicates that the cell has been selected. This window also displays a message when you load an application, remove a cell from the drawing board, save the application, and print the application. As soon as you complete your task, the status window is blank again.

Figure 3-4
Monitoring activity with the status window



Selected: cell2

Untitled

You will find the status window useful when you are working with cells on different pages. For example, if you select a cell on page 1 then move to page 3 of the drawing board, the Control Area still displays the name of the cell you select until you complete the task. This information helps you to always know which cell you selected.

The current application

The current application selection box displays the name of the application on the drawing board. If this is a new application, this field displays the name “Untitled.” If you save the new application, the selection box displays the new name.

Page navigation

While you are designing your application, your design can expand over one or more pages of the drawing board. The Control Area clearly displays the current page number. You can manipulate the slider or the push buttons to move to another page, as shown in Table 3-2.

> ***Note:*** Each page of the drawing board is separate from other pages of the drawing board. This feature provides you with the opportunity to modularize your application. You can place different application functions on separate pages to improve the readability and organization of your design. You can create a limitless number of application pages.

Table 3-2
Page navigation

Action	Procedure

Table 3-2
Page navigation (continued)

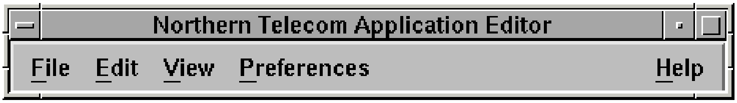
Move to FIRST page	Click with the left mouse button on << push button.
Move to LAST page	Click with the left mouse button on >> push button.
Move to PREVIOUS page	Click with the left mouse button on < push button.
Move to NEXT page	Click with the left mouse button on > push button.
Create a NEW page	Click with the left mouse button on > push button.
Move to any page BEFORE the current page	Drag the page number icon on the slider to the left.
Move to any page AFTER the current page	Drag the page number icon on the slider to the right.

Note: * To create a new page, the last page of the application must be currently displayed. This is the only way to create a new page.

Using the pull-down menus

The menu bar, shown in Figure 3-5, displays options for managing your application files, and the work you are currently doing on the drawing board.

Figure 3-5
Application Editor menu bar



File

Displays a pull-down menu of options for manipulating application files.

Edit

Displays a pull-down menu of options for manipulating data on the drawing board.

View

Displays a pull-down menu of options for managing default parameters.

Help

Displays context-sensitive help for the various options.

Preferences

Displays a pull-down menu of options you can use as default settings.

As shown in Figure 3-6, each option on the menu includes a mnemonic (that is, underscored character) and an accelerator (that is, Ctrl sequence) to provide you with greater flexibility in selecting the option. For example, instead of using the mouse to select the Open option, you can type the letter “O” with the pointer within the menu, or press <Ctrl+o> with the pointer located anywhere in the window. Table 3-3 lists the accelerator keys you can use for the file pull-down menu.

Figure 3-6
File pull-down menu

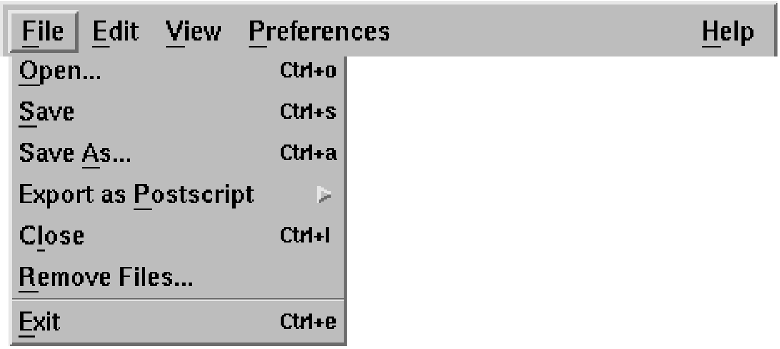


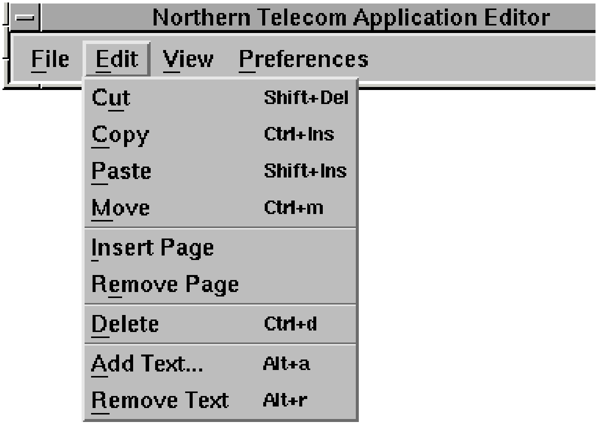
Table 3-3
Accelerator keys: file pull-down menu

Open	<Ctrl+o> Displays the File Browser window to select an existing application.
Save	<Ctrl+s> Saves the current application to the current filename. If the current application is new, it is saved as “Untitled.vpf”.
Save As	<Ctrl+a> Displays the File Browser to save the current application under a new name.
Export as Postscript	<Ctrl+p> Creates a postscript printer file (with “.ps” extension) with the same name as the application which you can later print using a system command (lp file.ps from the */u/ivr/apps* directory).
Close	<Ctrl+l> Closes the current application file and redisplays the initial drawing board. If you have made changes to the current application but have not saved them, the Application Editor prompts you to indicate whether or not you want to close without saving your changes. To save your changes, you must explicitly execute a SAVE or SAVE AS function.
Remove Files	Removes a file.

Edit pull-down menu

You can select the Edit pull-down menu displayed in Figure 3-7 to manipulate cells, modify text, and navigate pages in the application.

Figure 3-7
Edit pull-down menu



Manipulating cells

Procedure 3-2
Cutting and pasting cells

1 Place the pointer at the top left-hand corner of the cell.

2 Hold down the <Control> and <Shift> keys while you drag the mouse.

A thin outline forms around the cell as you drag the mouse.

Once you release the mouse button, the outline becomes thicker indicating that you selected the cell.

3 Click on Edit with the left mouse button, then select Cut.

To save time, press <Shift+Del>.

The application removes the selected cells from the drawing board and places them on the clipboard.

4 Click on Edit with the left mouse button, then select Paste.

To save time, press <Shift+Ins>.

A window frame appears.

5 Drag the mouse to where you want to paste the cell, then click on the left or middle mouse button.

Procedure 3-3
Copying cells

1 Place pointer at the top left hand corner of the cell.

2 Hold down the <Control> and <Shift> keys while you drag the mouse.

A thin outline forms around the cell as you drag the mouse.

Once you release the mouse button, the outline becomes thicker indicating that you selected the cell.

3 Click on Edit with the left mouse button, then select Copy.

To save time, press <Ctrl+Ins>.

4 Click on Edit with the left mouse button, then select Paste.

To save time, press <Shift+Ins>.

A window frame appears.

5 Drag the mouse to where you want to copy the cells, then click on the left or middle mouse button.

Procedure 3-4
Moving cells

1 Place the pointer at the top left hand corner of the cell.

2 Hold down the <Control> and <Shift> keys while you drag the mouse.

A thin outline forms around the cell as you drag the mouse.

Once you release the mouse button, the outline becomes thicker indicating that you selected the cell.

3 Click on Edit with the left mouse button, then select Move.

To save time, press <Ctrl+m>.

A window frame appears.

4 Drag the mouse to where you want to move the cells, then click on the left or middle mouse button.

As a short cut, you can simply select the cell by holding down the middle mouse button. When the window frame appears, drag the mouse to where you want the cells to appear, then release the mouse button.

Procedure 3-5 Deleting a cell

1 Click on the cell with the left mouse button.

2 Click on Edit with the left mouse button, then select Delete.

To save time, press <Ctrl+d>.

Manipulating pages

Procedure 3-6 Inserting a page

1 Using the slider or arrows in the status window, move to where you want to insert a page.

2 Click on Edit with the left mouse button, then select Insert page.

The application inserts the new page following the current page.

Procedure 3-7 Removing a page

1 Click on Edit with the left mouse button, then select Remove page.

The application removes the last page.

Modifying text

Procedure 3-8 Adding text

1 Click on Edit with the left mouse button, then select Add text.

The Add text window appears.

To save time, press <Alt+a>.

2 Type in the text you want to add and select the Font.

3 Click on Apply with the left mouse button.

A window frame appears.

4 Drag the mouse to where you want the text to appears, then click on the left or middle mouse button.

Procedure 3-9
Moving text

1 Select the text you want to move by holding down the middle mouse button.

A window frame appears.

2 Drag the window to where you want to move the text, then release the mouse button.

Procedure 3-10
Changing text

1 Double-click on the text you want to change with the left mouse button.

The change text box appears.

2 Type in the changes, then click on Apply with the left mouse button.

Procedure 3-11
Removing text

1 Select the text you want to remove by holding down the middle mouse button.

A window frame appears.

2 Click on Edit with the left mouse button, then select Edit.

The text becomes unbolded.

3 Click on the text with the left mouse button to remove it.

View pull-down menu

The View menu allows you to manage the appearance of the drawing board, as well as to define the default cell as shown in Figure 3-8. Table 3-4 lists the accelerator keys you can use for the view pull-down menu.

Figure 3-8
View pull-down menu

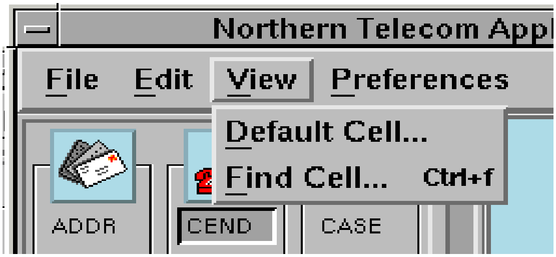


Table 3-4
Accelerator keys: view pull-down menu

Default Cell	Displays the Application Default pop-up window to modify the default cell parameters.
Find Cell	<Ctrl+f> Searches for the cell you specify.

Preferences menu

The Preferences pull-down menu as shown in Figure 3-9 allows you to set defaults on your application.

Figure 3-9
Preferences pull-down menu

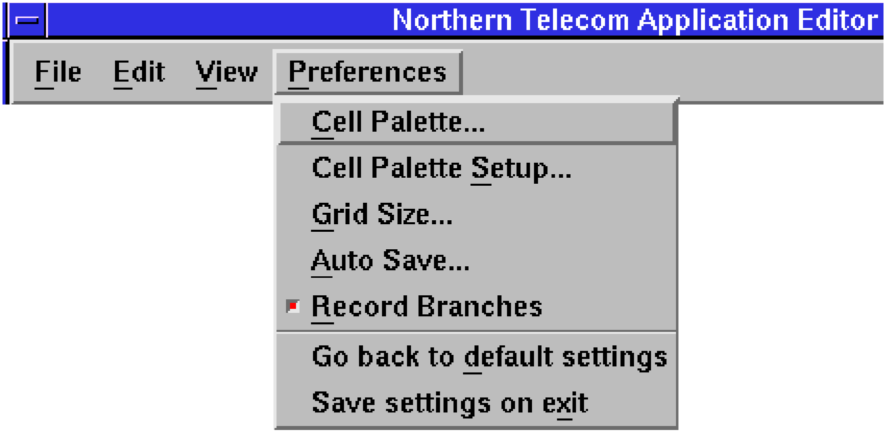


Table 3-5 lists the accelerator keys you can use with the Preferences pull-down menu.

Table 3-5
Accelerator keys: preferences pull-down menu

Cell Palette	Allows you to arrange the order of items on the cell palette.
Cell Palette Setup	Allows you to specify what items should appear on the desktop.
Grid Size	Allows you to modify the coordinate resolution (pixels) of cells placed on the drawing board. This option aligns cells according to the grid size you select.
Auto Save	Allows you to specify autosave intervals and number of backup files.
Record Branches	The red square indicates that this option is enabled.
Go back to default settings	Allows you to go back to the default settings that take effect in the next session.
Save settings on exit	Allows you to save changes when you exit the file. The red square indicates that this option is enabled.

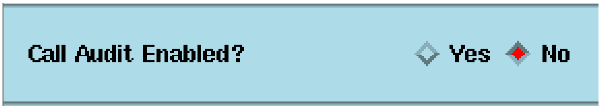
Selecting options with text entry boxes, buttons, and menus

While you are working with the Application Editor, you will notice that options are presented in a variety of ways to provide convenience and flexibility. This section briefly explains how the Application Editor presents options that you can use.

Radio buttons

Think of a radio button as a switch, just like a radio button on your car. It indicates whether or not an option is ON or OFF. To select a radio button, simply click on the button with the left mouse button (see Figure 3-10).

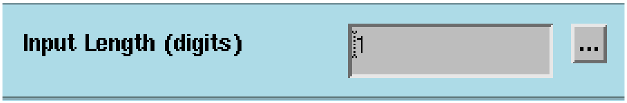
Figure 3-10
Radio buttons



Text entry boxes

A text entry box, for example Figure 3-11, displays the current value for an object (for example, a cell parameter) and is accompanied by an ellipsis push button. To access additional valid values, just click on the push button.

Figure 3-11
Text entry box with push button



There are two ways you can select a new value. You can position the pointer in the text entry box and type a new value, or in some text boxes you can select a value from the Browser.

Table 3-6 lists the ways you can manipulate text entry boxes.

Table 3-6
Manipulating text entry boxes

Action	Process
Select and type over text	Point, click, and drag the left mouse button; type text.
Position insertion point	Click with the left mouse button.
Select and type over nearest word	Point and double-click the left mouse button; type text.
Replace all text	Point and triple-click the left mouse button; type text.

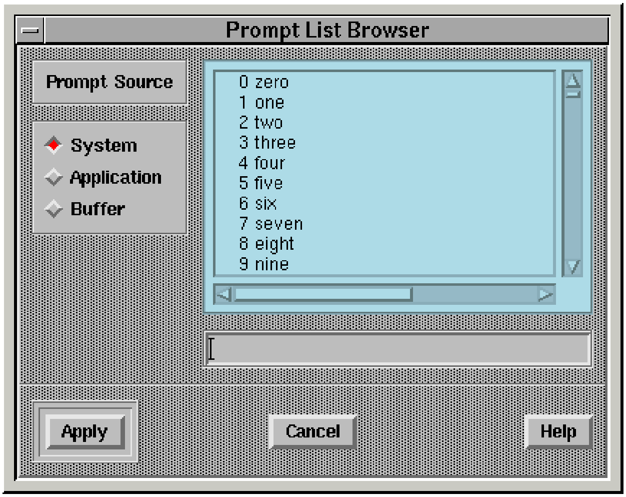
Procedure 3-12
Manipulating text entry boxes

1 Click on the ellipsis push button with the left mouse button to display the Browser window. This window lists valid values for the parameter as shown in Figure 3-12.

2 Click on the scroll bar with the left mouse button to scroll through the values to the one you want.

The Prompt Source radio buttons on the left side of the following figure indicate with which source the listed values are associated.

Figure 3-12
A browser with a list of valid values



Prompt List Browser

Prompt Source

● System
○ Application
○ Buffer

0 zero
1 one
2 two
3 three
4 four
5 five
6 six
7 seven
8 eight
9 nine

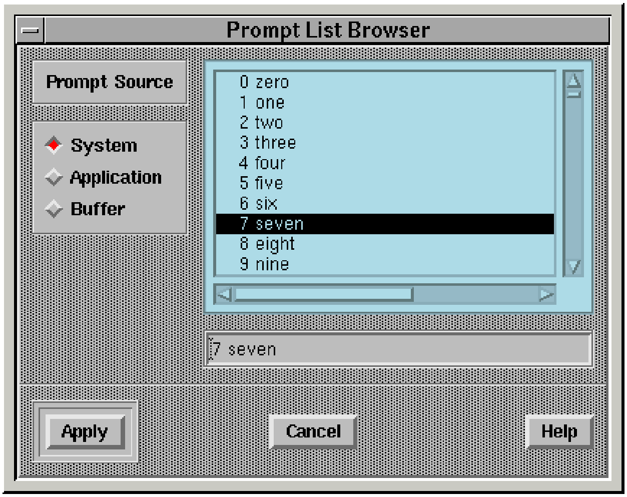
The system radio button is used for the standard prompts that are included with Meridian IVR.

The application radio button is used for the prompts that have been defined specifically for this application.

The buffer radio button lists the available buffers which you can preload with prompt IDs.

3 Click on your choice with the left mouse button to highlight it as illustrated in Figure 3-13.

Figure 3-13
Selecting a value from the browser



7 seven

4 To accept the selected value, click on the Apply push button with the left mouse button.

To return to the parameter menu without modifying the value, select the Cancel button.

If you select Cancel, the Application Editor displays the old value. The Browser disappears and the Application Editor displays the new value.

Note: A shortcut to selecting and then applying a value is to simply double-click on a value in the list with the left mouse button.

The digit button

One of the push buttons that you encounter while using the Application Editor is the *digit* button. This button identifies the current value for a parameter which corresponds to one of the digits on a touch-tone pad as shown in Figure 3-14.

Procedure 3-13
Selecting a different digit Application Editor

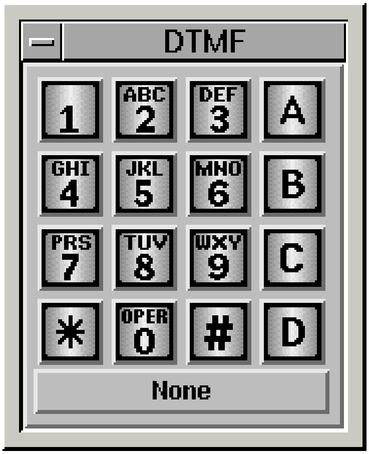
1 Click on the digit button with the left mouse button.

The Application Editor displays a touch-tone pad.

2 Click on one of the buttons on the touch-tone pad with the left mouse button.

The Application Editor re-displays the digit button with the new value.

Figure 3-14
Selecting digits with a digit push button

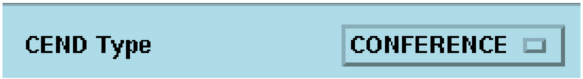


Note: If you select the Default key on the touch-tone pad, the value appears as DFLT. This is the value that is set in the application defaults. For more information, see "Default values for cell parameters" on page 3-29.

Option menus

Another method used by the Application Editor to display a current value with easy access to additional options is to display a value box with an underlying *option menu*. To illustrate this, consider Figure 3-15, which displays CONFERENCE as the current CEND Type.

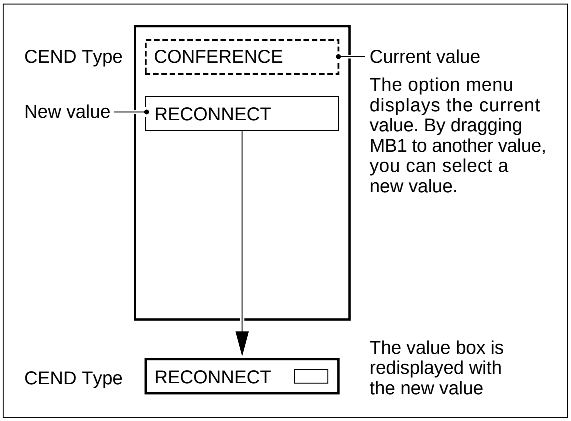
Figure 3-15
Value box



CEND Type CONFERENCE

If you click on the value box (CONFERENCE) with the left mouse button, the Application Editor displays a drop-down menu as illustrated in Figure 3-16. Select the option you want with the left mouse button. If you no longer want to change the value, click anywhere on the drawing board with the left mouse button.

Figure 3-16
Option menu


CEND Type
CONFERENCE
Current value
The option menu displays the current value. By dragging MB1 to another value, you can select a new value.
New value
RECONNECT
The value box is redisplayed with the new value
CEND Type
RECONNECT

Opening an application

When you invoke the Application Editor, the drawing board is empty except for the START cell. You can either create a new application on the drawing board or open an existing application.

Procedure 3-14
Opening an application

1 Select Open from the File menu with the left mouse button.

The Application Browser similar to that shown in Figure 3-17 appears.

2 Select the directory you are using for developing applications from the directories scroll area on the Application Browser by clicking in the directory scroll area with the left mouse button (the default directory is */u/ivr/apps*).

3 If the desired file is not in the default directory, click on the Filter button of the Application Browser with the left mouse button.

The list of files for this directory appears in the Files scroll area (see Figure 3-17).

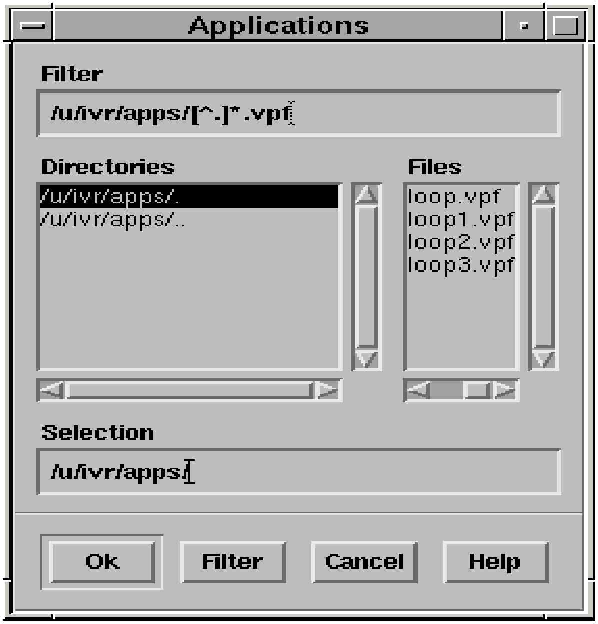
4 Click on the file you want to open with the left mouse button.

The application name appears in the Selection field of the Application Browser.

5 Click on OK with the left mouse button.

The application appears in the drawing board. The application name appears under the Current Application.

Figure 3-17
Application selection window



Filter
/u/ivr/apps/[^.]*.vpf

Directories
/u/ivr/apps/.
/u/ivr/apps/..

Files
loop.vpf
loop1.vpf
loop2.vpf
loop3.vpf

Selection
/u/ivr/apps/

Fields

File filter

The Application Editor displays only those file names that contain the character-string pattern that you specify in this field. The default filter is */u/ivr/apps/[^.]*.vpf* which displays all file names in the *apps* directory containing the pattern.vpf. To use a different filter, enter a valid UNIX pathname extension in the selection filter box, then click on the filter push button with the left mouse button to execute the query.

Directories

This field is a list of directories that shows the current directory, the parent directory, and any subordinate directories. To select a different directory, click on one of these entries with the left mouse button. The Application Editor displays a new list.

Files

A list of applications in the current directory. You can scroll the list and select a specific file.

Selection

The application file to be selected. When you select a file from the Files list, the Application Editor displays the filename in this box. Otherwise, you can enter a filename directly into this selection box.

Push buttons

OK

This button opens the application displayed in the Selection box.

Filter

This button executes the File Filter query and displays a list of files that match the filter. Pressing <Enter> automatically selects the Filter push button.

Cancel

This button returns control to the Application Editor without opening an application.

Help

This button displays Help information about this window.

You can retrieve an existing application to your drawing board simply by typing the application name in the Selection text entry box and then clicking with the left mouse button on the OK push button.

> ***Note:*** A shortcut to selecting an application displayed in the Browser list is to double-click on the desired application name with the left mouse button. The Application Editor redisplays the drawing board with the selected application.

To select a file from the Browser list with the filter, follow these steps:

Procedure 3-15
Selecting a file from the Browser list

1 Type a character-string pattern in the Filter text entry box.

Use the default pattern */u/ivr/apps/[^.]*.vpf* as shown in Figure 3-17.

2 Click on the Filter push button with the left mouse button.

A list of file names containing the string you specified appears. Notice that all of the files listed in Figure 3-17 include the .vpf string corresponding to the filter.

3 Scroll through the list of files.

4 Click on the application that you want to open with the left mouse button.

The Application Editor displays the selected application file name in the Selection text entry box.

5 Click on the OK push button with the left mouse button to open the selected application file and display it on the editor drawing board.

Default values for cell parameters

The Application Editor provides default values for a variety of parameters used by the cells in your application. You can use or modify the Application Editor defaults.

> ***Note:*** You can modify these parameters to establish values for parameters that are used universally throughout the application.

To display and modify the default cell parameters, double-click on the START cell with the left mouse button.

CAUTION!

Risk of freezing up the system

Do not double-click on the window corner to close the window, otherwise your system will freeze.

To close the window, click on the Cancel button.

If you are on another page of the drawing board, follow these steps to display the Application Defaults pop-up window as shown in Figure 3-18:

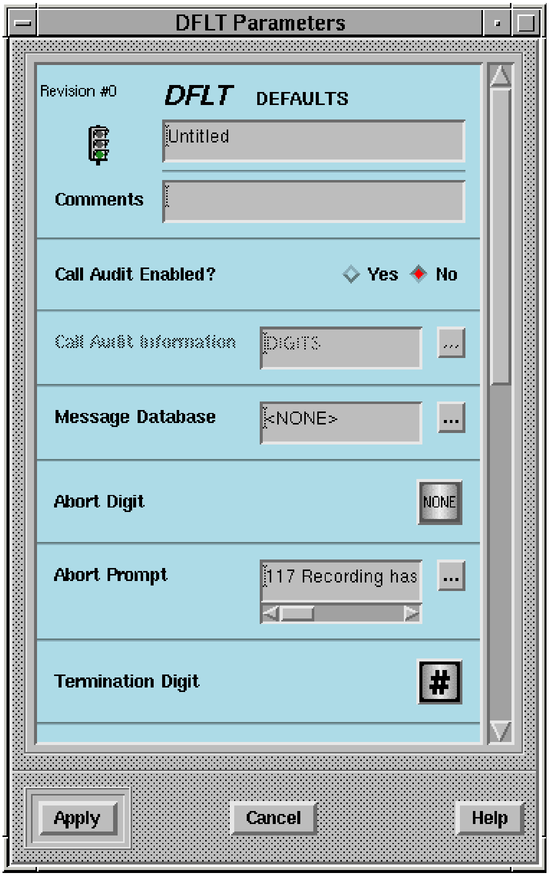
Procedure 3-16
Displaying or updating default cell parameters

1 Select the View option on the window menu bar.

The Application Editor displays the View pull-down menu.

2 Select the Default Cell option.

Figure 3-18
Default cell parameters



DFLT Parameters

Revision #0

DFLT DEFAULTS

Untitled

Comments

Call Audit Enabled? ◇ Yes ◆ No

Call Audit Information DIGITS

Message Database <NONE>

Abort Digit NONE

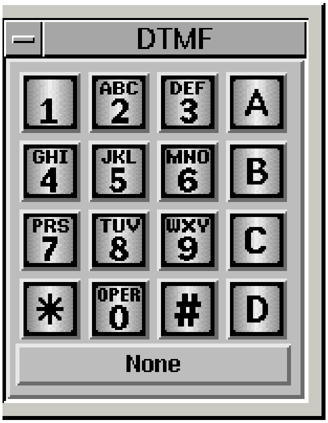
Abort Prompt 117 Recording has

Termination Digit #

You can change any of these parameters to apply to the current application. Many of the parameter values appear in text entry boxes. For these parameters, you can modify the value directly by selecting the text entry box and typing a new value in the box, or clicking on the associated ellipsis button with the left mouse button to select a value from the Browser.

Some of the parameters appear as digit buttons. When you click with the left mouse button on a digit button, the Application Editor displays the touch-tone pad pop-up window illustrated in Figure 3-19. You can click with the left mouse button on any digit to designate a new default value including “None” to indicate that the default is no digit. Once you select a digit, the touch-tone pad window disappears, and the original digit button is redisplayed with the new value.

Figure 3-19
Default cell parameters touch-tone pad window



> ***Note:*** The touch-tone pad associated with the Application Defaults window is unique. Other touch-tone pad windows displayed by the Application Editor (for example, when you display the parameters for a specific cell) show a “Default” key in place of “None.” In that case, selecting the Default digit displays the value of DFLT which indicates that the parameter uses the value defined by the application default.

The value for other parameters on the Application Defaults window are indicated by a slider. You can change the current value by dragging the mouse to a new value.

Applying the new default parameters

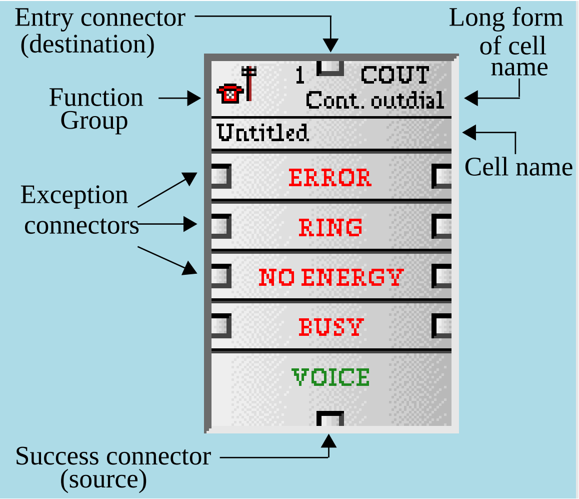
Once you have selected the default values for your application, you can save them and return to the drawing board by clicking on the Apply push button with the left mouse button. If you redisplay the default parameter values, you will see the new values (also, any values that you typed in the entry boxes are included in the Browser list of valid values). You can change the default values at any time while designing your application.

Note: The START cell parameters are the same as the default cell parameters. Modifying the START cell parameters also changes the default cell parameters accessed from the View menu bar option.

Arranging cells on the drawing board

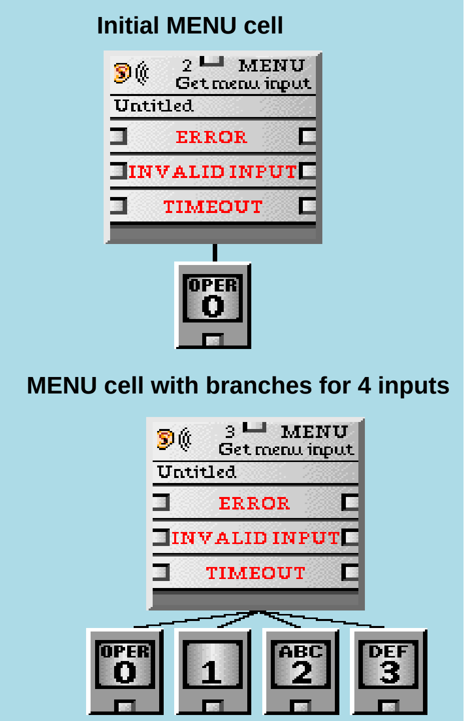
When you place a cell on the drawing board, it appears as an icon (see Figure 3-20). There are a few things you need to remember. The START cell does not have an entry point. A cell can have one or more branch connectors (success and exceptions), except for the EXEC and HANG cells which have no success branches.

Figure 3-20
COUT cell icon



One cell that deserves comment is the MENU cell. Figure 3-21 illustrates a MENU cell as it appears when you create it, and another MENU cell for which four inputs have been defined.

Figure 3-21
MENU cell icon



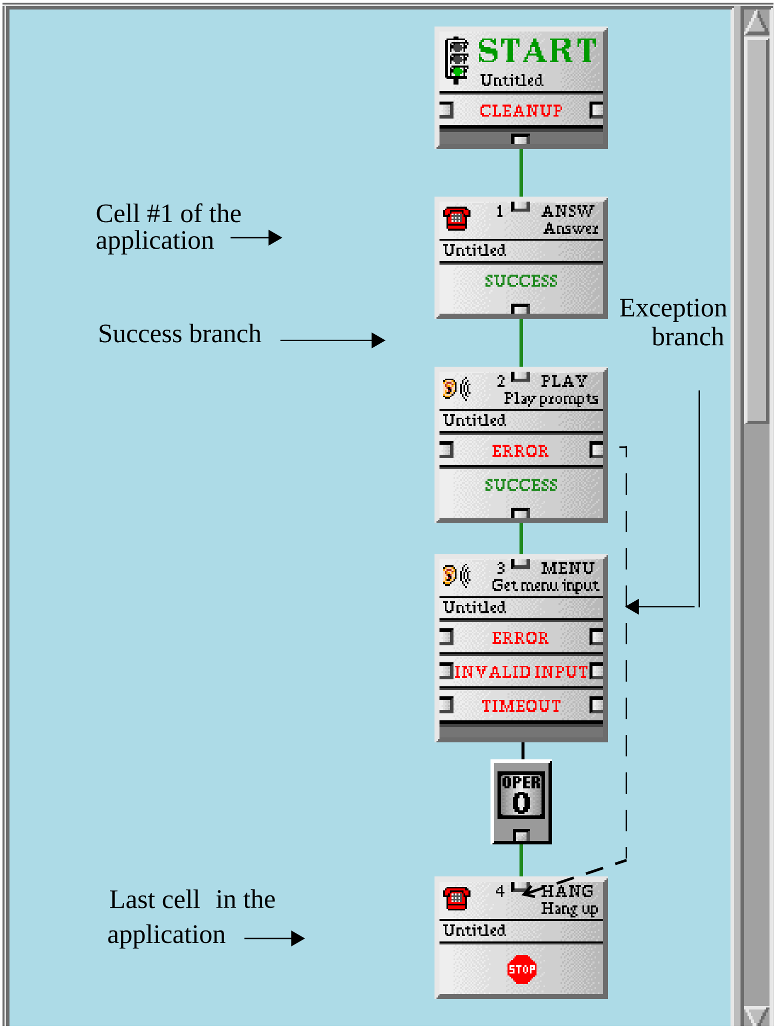
Initial MENU cell

MENU cell with branches for 4 inputs

Normal call flow is top to bottom as shown in Figure 3-22. You select cells from the palette and place them on the drawing board. You can move them anywhere and interconnect them by drawing lines between the branch connectors. You can enter cells only at the top (that is, at the entry connector), whereas you can designate branches from the bottom (success) or sides (exception).

Success branches appear as solid (green) lines, and exception branches appear as dashed (red) lines. Whether you connect an exception branch from the left or right side of the cell, the Application Editor automatically selects the branch connector that minimizes the length of the connection between the two cells.

Figure 3-22
Top-to-bottom call flow



Procedure 3-17
Placing a new cell on the drawing board

1 On the palette, click on the name of the cell you want to select with the left mouse button.

The name of the cell is pushed in showing that you have selected this cell.

2 Drag the mouse to where you want to place the cell, then click on the middle mouse button.

The Application Editor displays the cell icon for the cell.

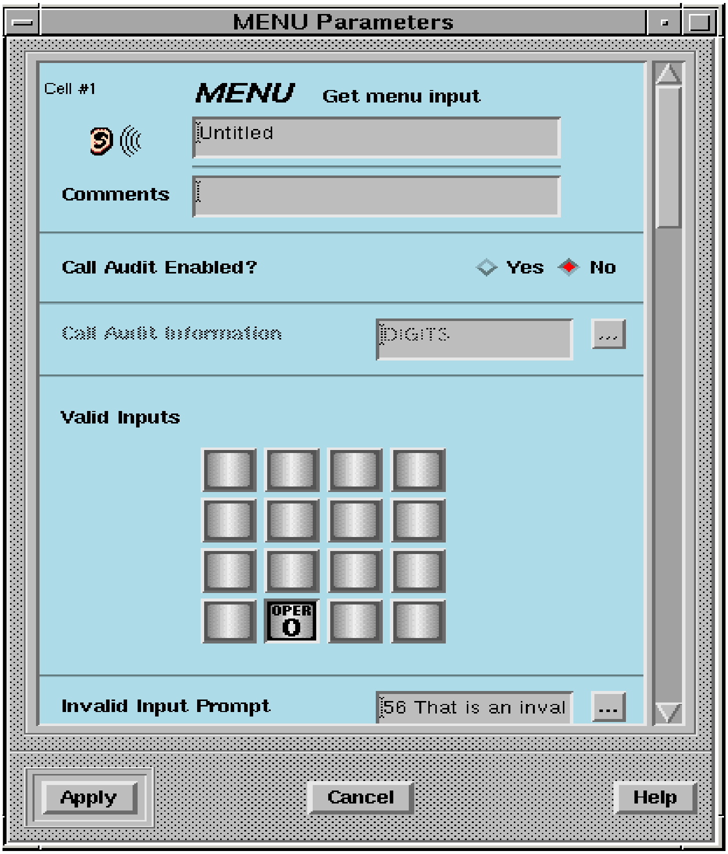
Note: Do not click on a cell that is already on the drawing board because the Application Editor interprets this action as a move-cell operation (see “Moving cells on the drawing board,” in this chapter).

As an alternative to step 2, you can press on the middle mouse button until an outline of the cell appears, drag the mouse to where you want to place the cell, and release the button.

Modifying cell parameters

To display and modify the parameters for a specific cell on the drawing board, double-click on the cell with the left mouse button. The Application Editor displays the current parameters. Figure 3-23 illustrates a parameter pop-up window for a MENU cell.

Figure 3-23
MENU cell parameter pop-up window



Title: Untitled

Comments:

Call Audit Enabled? ◇ Yes ◆ No

Call Audit Information: DIGITS ...

Invalid Input Prompt: 56 That is an inval ...

Enter a unique title in the title text entry box. This title appears on the cell icon and is useful for documenting the purpose of the cell in your application. If you do not specify a title, the cell is listed as “Untitled.” You can also include comments about the cell in the “Comments” text entry box. These comments only appear here.

The current value for each parameter is also listed. Initially, these values are set to the default cell parameters values (see “Displaying or updating default cell parameters,” in this chapter). However, you can change the parameters for each cell. For many cells, you can view all of the parameters together in the pop-up window. However, for other cells, such as the MENU cell, there are numerous parameters. For these cells, you need to scroll through the various parameters in the window. Each parameter pop-up window includes these push buttons:

Apply

Saves your changes and returns to the drawing board.

Cancel

Returns to the drawing board without saving your changes.

Help

Displays on-line documentation regarding the contents of the pop-up window.

Connecting cells

You can connect any two cells using an exception or success branch. To help you connect cells in your application, consider the following:

- There is no preference in the order that you designate cells to be connected.
- A cell can be entered only through the entry connector at the top of the cell.
- A branch must include a source (either a success or exception connector) and a destination (an entry connector).
- While the branch connect tool is active (that is, the pencil cursor appears), you can select multiple cell connectors. The branch is completed and the branch connect tool is deactivated when two sequentially selected cell connectors correspond to a source and a destination connector.
- A branch that exits a cell from one of the sides of the cell icon is an exception branch and it appears as a dashed (red) line. A branch exiting from the bottom of the cell is a success branch and it appears as a solid (green) line.
- When you are creating an exception branch, you can select either side of the cell. However, the Application Editor automatically selects the side that results in the shortest connection.
- Each source connector must have a destination; otherwise, when you save your application, the Application Editor automatically connects it to the first cell inserted in your application.

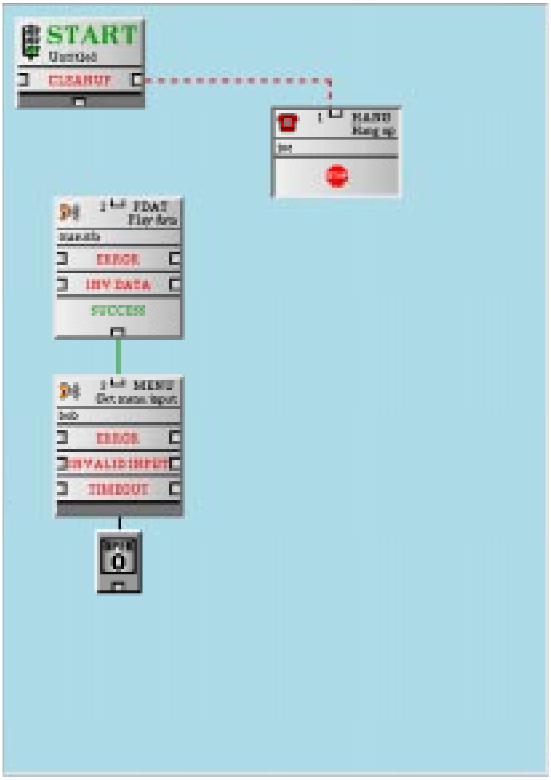
Procedure 3-18
Connecting cells

1 Click on a cell connector with the left mouse button.

The pointer changes to a pencil cursor.

You can use the pencil cursor to connect cells as shown in Figure 3-24.

Figure 3-24
Using the pencil cursor to connect cells



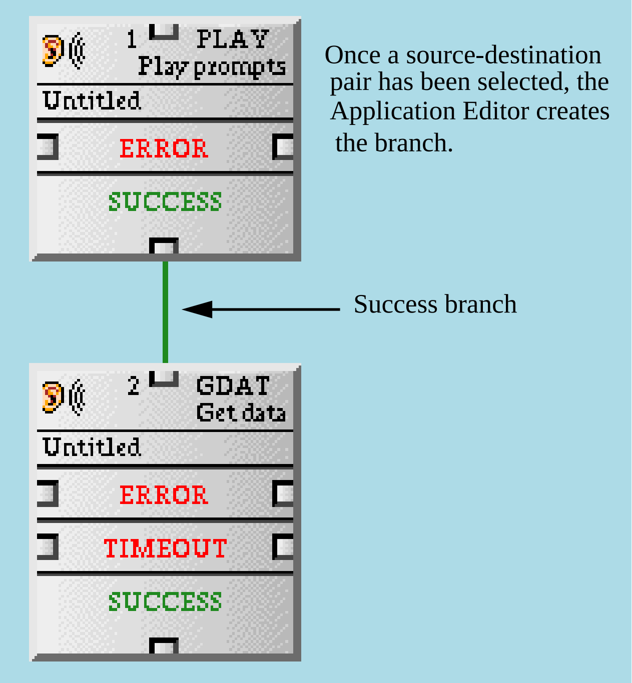
2 Point (the pencil) and click on a connector of the other cell with the left mouse button to form a source-destination pair.

As illustrated in Figure 3-25, the Application Editor creates the connection as a success branch.

Figure 3-26 illustrates both success and exception branches.

You can deselect the pencil by pointing to the “START” (default) cell and clicking on it with the left mouse button.

Figure 3-25
Success branch

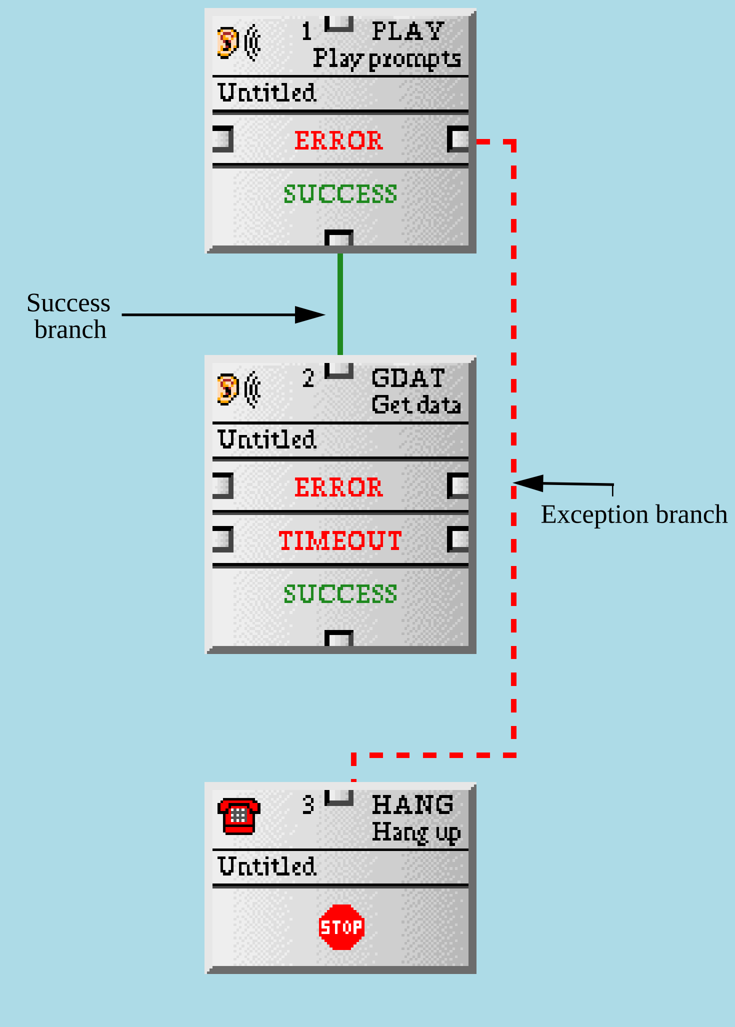


Once a source-destination pair has been selected, the Application Editor creates the branch.

Success branch

Note: A shortcut to connecting cells is to create the first cell, click on the cell's branch connector with the left mouse button, then create the next cell on the drawing board. Because you activate the branch connect tool before creating the second cell, the Application Editor automatically generates the branch when you create the cell. This technique is particularly useful when you are creating/connecting cells between pages.

Figure 3-26
Interconnecting cells



Off-page connectors

If you connect cells on the same page of the drawing board, you can easily see that the cells are interconnected. However, in some instances your call flow can have several drawing board pages. To create a branch between Cell A on the current page and Cell B on another page follow these steps:

Procedure 3-19
Connecting a cell to a cell on another page

1 Click on the branch connector of Cell A on the current page with the left mouse button.

The pointer changes to the pencil cursor.

2 Using the mouse, move the pointer to the Control Area and display the page containing Cell B.

3 Point (the pencil cursor) and click on the branch connector of Cell B with the left mouse button to complete the source-destination pair.

The Application Editor displays TO and FROM off-page connectors to indicate where the branch is. Figure 3-27 illustrates a TO off-page connector that branches from the GSUB cell to a PDAT cell on page 2 of the drawing board. It also illustrates another TO off-page connector that branches to the HANG cell on page 2 if an ERROR occurs during processing of the GSUB cell. Figure 3-28 shows the corresponding FROM off-page connectors.

Figure 3-27
Branching to a cell on another page

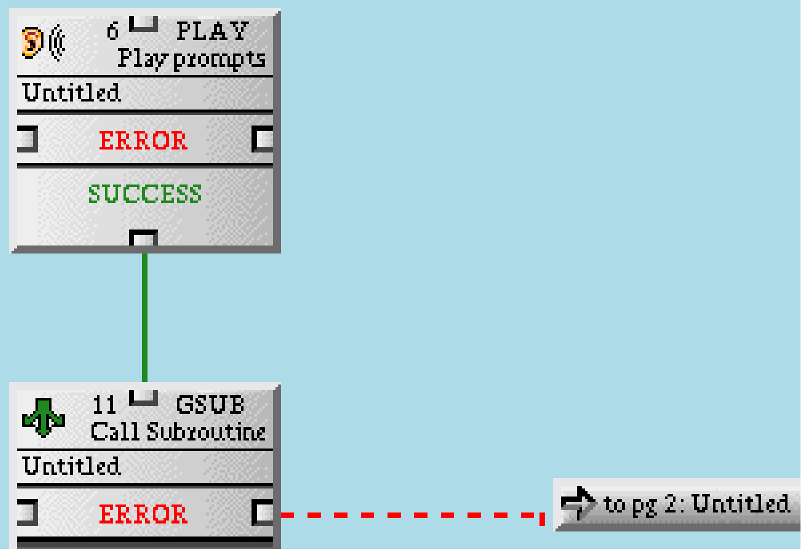
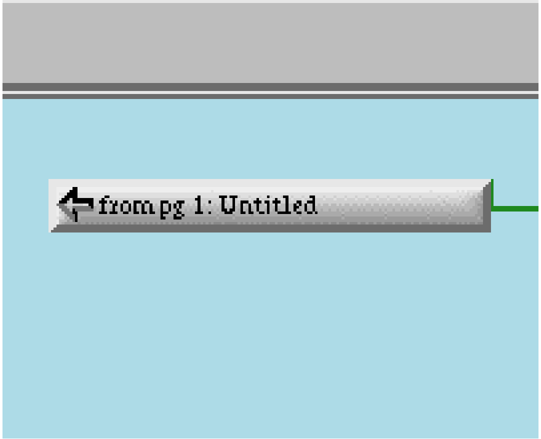


Figure 3-28
Identifying a branch from a cell on another page



If you double-click on an off-page connector with the left mouse button, the Application Editor goes directly to the indicated page and selects the destination cell to help you find it. For example, double-clicking on the off-page connector in Figure 3-28 redisplays page 1 and automatically selects the GSUB cell in Figure 3-27.

Moving cells on the drawing board

You can move a cell on the drawing board to somewhere else either on the same page or to another page. The Application Editor maintains all of the cell branches. This section describes how to move a cell and illustrates how the various branches reappear.

> ***Note:*** You should move HANG cells to a separate page because as you design your application, you will find it easier to route different cells to the same HANG cell. However, the clutter of connecting lines makes the layout difficult to understand. By moving HANG cells to a separate page, you improve the readability of the layout since off-page connectors replace the connecting lines.

Procedure 3-20
Moving a cell on the same page

1 Click on the cell that you want to move with the left mouse button.

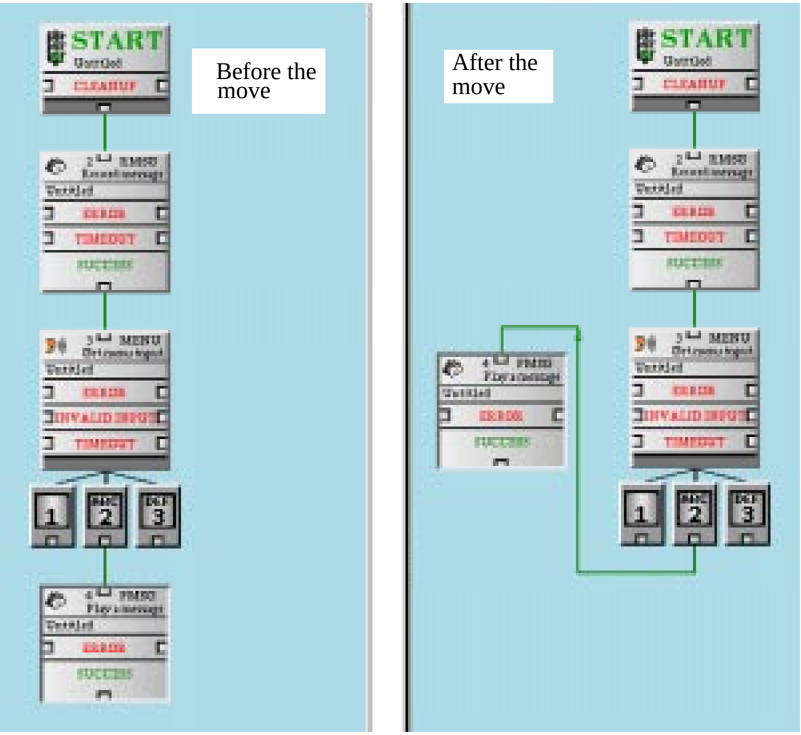
2 Drag the middle mouse button to the new location.

Any branches associated with the cell are automatically resized to maintain the connection.

Figure 3-29 illustrates the before and after screens seen when moving the Play a Message (PMSG) cell.

Note: If you select a cell and move the pointer to another cell before pressing the middle mouse button, the selected cell disappears from the display after you press the middle mouse button. The cell is still present but not visible. To redisplay the cell, move to another page, and then return to the page where the cell resides. The “disappearing” cell will be redisplayed.

Figure 3-29
Sample move of a cell



Procedure 3-21
Moving a cell to another page

1 Click on the cell you want to move with the left mouse button.

The Status Window indicates that the cell has been selected.

2 Display the page to which the cell is moving by using the page selector push buttons or slider.

Notice that the Status Window indicates that the cell is still selected.

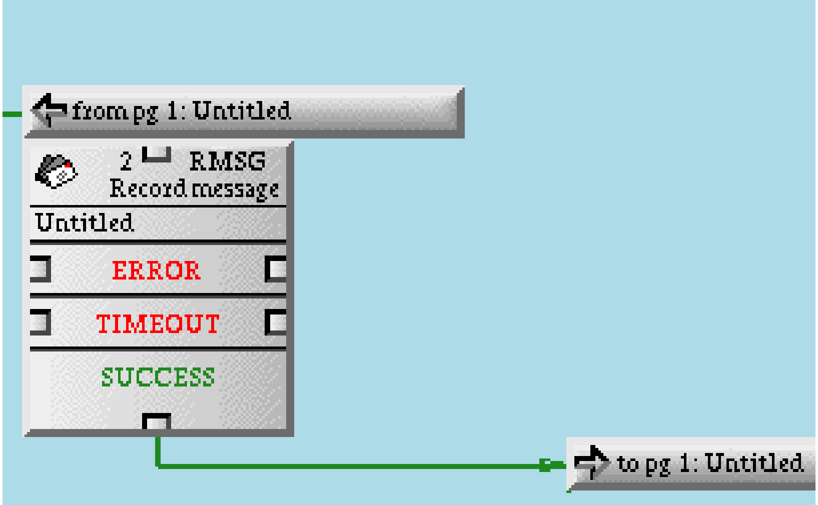
3 Press on the middle mouse button and drag the mouse to the location on the drawing board where you want to place the cell.

The Application Editor displays the cell you have moved with all branches intact.

Those branches that are now off-page are identified by off-page connectors. (At this point the Status Window is empty indicating that the cell is no longer selected.)

Figure 3-30 illustrates the RMSG cell shown in Figure 3-29 moved to page 2 of the drawing board.

Figure 3-30
Cell moved to another page



Removing cells from your application

While you are laying out your design, you may need to remove one or more cells from your application. If so, you need to remember that the Application Editor removes a cell that has branches to and from other cells.

> **ATTENTION!**
>
> You should *never* remove cell #1 (the HANG cell), the first cell that you create on the drawing board. This cell is required because all the error branches that are not defined in an application, go to that first cell, and the HANG cell allows you to hang up the call.

Procedure 3-22
Removing cells

1 Redirect each branch entering the cell to a different destination. To do this, click on the source connector with the left mouse button to activate the pencil cursor.

2 Point the pencil cursor on a connector of a different cell, then click on the left mouse button.

Repeat this process until all incoming branches are redirected.

3 Redirect each of the cell’s outgoing branches (success and exceptions) to the cell's entry connector. To do this, click on the branch connector with the left mouse button to activate the pencil cursor.

4 Click on the cell’s entry connector with the left mouse button.

Repeat this process until all outgoing branches have been redirected.

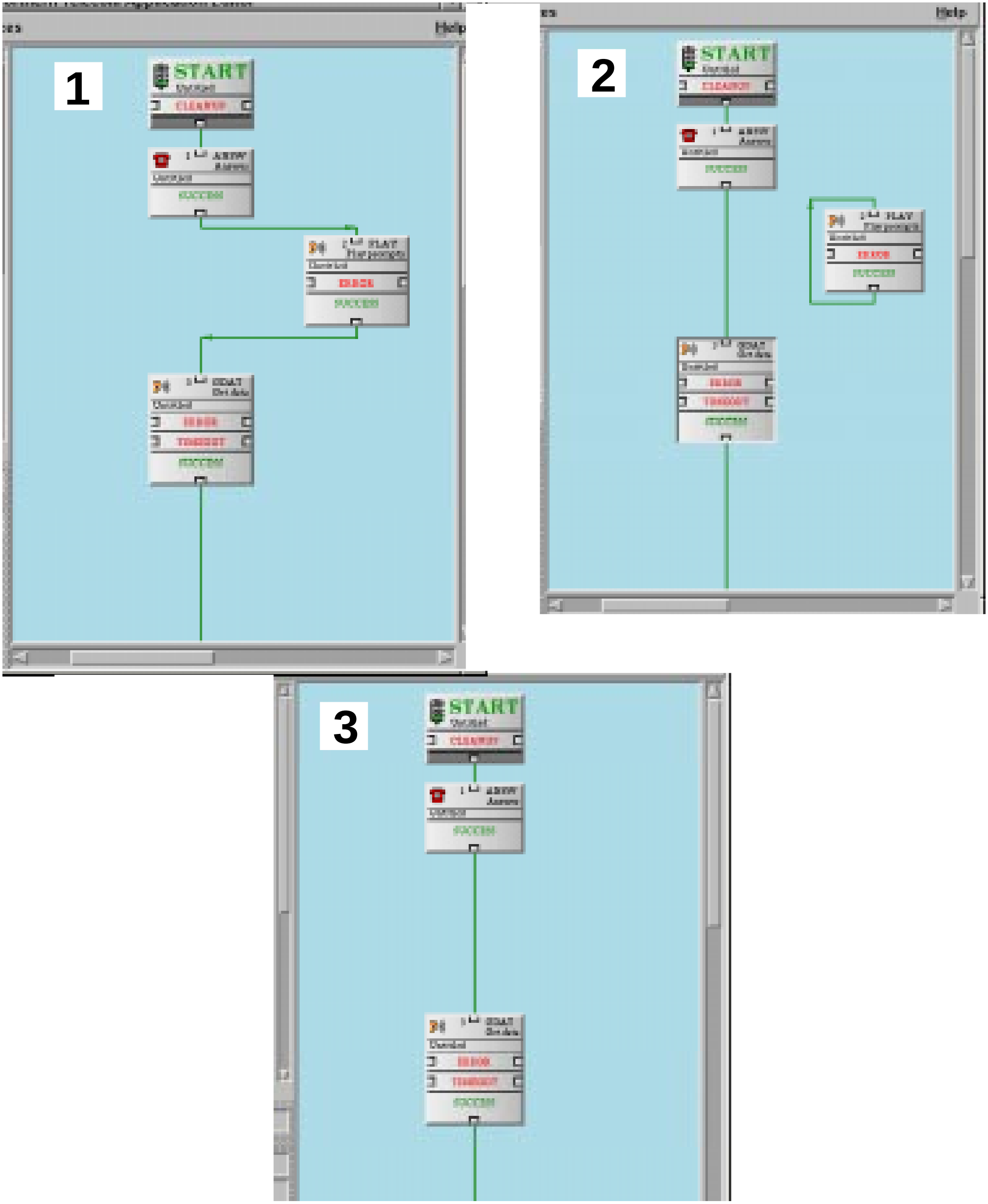
5 Click on the cell with the left mouse button to select it.

6 Select the Delete option from the Edit pull-down menu (or use the <Ctrl+d> keyboard accelerator).

The Application Editor removes the cell from the drawing board.

Figure 3-31 illustrates (1) a PLAY cell in an application, (2) the PLAY cell with all of its branches redirected, and (3) the application with the PLAY cell removed.

Figure 3-31
Removing a PLAY cell from an application



Saving your application

As a matter of prudent design technique, you should save the application on the drawing board often. You have the option of saving the application under its current name (for example, the name displayed in the “Current Application” box in the Control Area), using the Save option. You also have the option of saving an application under a new name using the Save As option.

Procedure 3-23
Saving the application under the current name

1 Click on the File with the left mouse button.

The Application Editor displays the File pull-down menu.

2 Click on the Save option with the left mouse button.

The Application Editor saves the application.

(As an alternative to steps 1 and 2, you can press <Ctrl+s>).

Procedure 3-24
Saving the application under a new name

1 Click on the File with the left mouse button.

The Application Editor displays the File pull-down menu.

2 Click on the Save As option with the left mouse button.

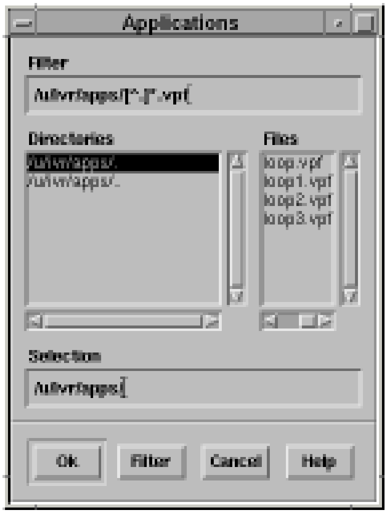
The Application Editor displays the File Browser.

(As an alternative to steps 1 and 2, you can press <Ctrl+a> to display the File Browser).

3 Enter a new name in the text entry box, and click on the OK push button with the left mouse button.

The Application Editor saves the application on the drawing board to the new file name. See Figure 3-32.

Figure 3-32
Saving an application under a new name



Filter: /u/ivr/apps/[^.]*.vpf

Directories:
/u/ivr/apps/.
/u/ivr/apps/..

Files:
loop.vpf
loop1.vpf
loop2.vpf
loop3.vpf

Selection: /u/ivr/apps/

Printing your application

The Application Editor provides a facility for postscripting your application with a listing of cell parameters, databases, user functions, prompts, and buffers used by the application. Each page of the drawing board prints as a single page.

> ***Note:*** You need a Postscript printer to print your application.

Procedure 3-25
Printing your application

1 Click on File with the left mouse button.

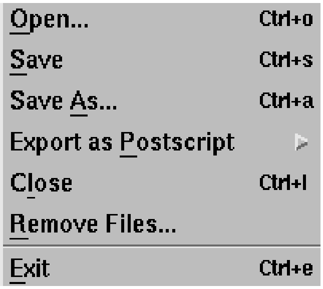
The Application Editor displays a pull-down menu.

2 Click on the Export as Postscript option with the left mouse button.

The Application Editor displays the following pull-down menu.

(As an alternative to steps 1 and 2, you can press the <Ctrl+p> keyboard accelerator which automatically creates a Postscript file of your application.)

Figure 3-33
File pull-down menu with Postscript option



3 Click on All in the sub-menu with the left mouse button.

The Application Editor generates a Postscript file (in the /u/ivr/apps directory) with the same name as your application (that is with the .ps extension).

To postscript only the call flow, click on the Call Flow option with the left mouse button.

To postscript only the contents of the selected cell, click on the cell content option with the left mouse button.

4 Move the pointer to another window running the system shell.

5 Enter the appropriate print command for printing .ps files (for example, **lp /u/ivr/apps/LeaveMsg.ps**) at the command prompt.

Developing and testing applications on a live system

Meridian IVR systems include both a development and run-time capability so that you can develop and test IVR applications on the same system. Therefore, you can make minor modifications to an existing application while the system is running. This is valuable for customers who may need to quickly add another selection to a menu (for example, an airline that has just announced a seat sale).

If you are using the Meridian IVR system for new application development or application modification in a “live” customer environment, you should ensure that the new application development or testing does not impact existing running applications.

ATTENTION!

Do not change the live application itself. Rather, you should copy the application, then modify and test the copy on a dedicated test channel that is isolated from normal customer traffic. After you have done this, replace the application with the new modified version.

Response times can be impacted

On-line development using the Application Editor consumes CPU cycles and thus impacts the response times of running applications. To minimize the risk, we recommend that you make such changes during a relatively low traffic period.

CAUTION!

Risk of damaging system

Do not test applications that use custom User Functions (“C” code) on a live system as a poorly written untested custom “C” program can potentially cause severe damage on a live system. In addition, the compilation of “C” code on a live system uses a lot of CPU cycles which can impact the response time of live applications

Ensure that the test environment is isolated

While testing new applications, ensure that the application test environment classes such as Meridian Mail and IVR channels, ACD routing, VSDN, and ACCESS are fully isolated from the live applications and, as such, do not impact live applications. We recommend that you following the following guidelines:

1 Preliminary testing by the VAD on a captive development Meridian IVR system, including Meridian Mail and Meridian 1 PBX with trunks, is necessary but not sufficient. Controlled testing to prove the sanity of the application on a live system is still needed even after the best efforts of testing on a captive development system.

2 To minimize the risk of service interruptions, you should test new or modified call-flow applications in three phases:

Phase 1
Test channels isolated from receiving customer traffic; tester-generated traffic only.

Phase 2
A single monitored test channel receiving customer traffic.

Phase 3
A few monitored test channels receiving customer traffic.

3 When testing applications, you must understand the distinctions between Meridian IVR logical channel acquisition type (dedicated versus shared), Meridian Mail voice service channel service type (ACCESS versus ALL), and Meridian 1 PBX ACD traffic routing (dedicated versus shared). When speaking of dedicated test channels, we are referring to some combined configuration which gives you the degree of control necessary for the particular application and phase of testing.

4 If possible, you should test new or modified applications on test channels of dedicated acquisition type to avoid impacting live applications when setting up or taking down the isolated test environment.

Note that some applications require that Meridian IVR logical channels be configured for shared acquisition type.

5 Dedicated acquisition type can tie a Meridian IVR logical channel to an individual Meridian Mail voice channel. You can route test traffic to the application that is being proved by assigning the ACD Agent Position for the test channel to a Meridian Mail MC-DN that can be dedicated to routing test calls only. This is achieved most conveniently by using ACD Automatic Overflow. It changes the Overflow DNs to selectively route live customer traffic into the test channel during Phase 2 and Phase 3 testing, and whenever it is not being used for testing. During Phase 1 testing, you can prevent customer traffic from overflowing into the test channel.

6 Shared acquisition type depends on the ACCESS class defined for the logical channel and for the dialed DN in the Meridian Mail VSDN table. The Meridian IVR logical channel, therefore, is tied to the dialed DN (and not to any individual Meridian Mail voice channel). Establishing an isolated test environment with shared acquisition type involves configuring a Meridian IVR logical channel with an ACCESS class that is dedicated to the test application and the dialed DN.

7 With shared acquisition type, it is impossible to switch a logical channel between customer traffic and tester-generated traffic without interrupting service on all channels. This is because the switch is accomplished by changing the class of the logical channel through Meridian IVR system configuration. Reconfiguring a single logical channel requires that all Meridian IVR channels be stopped and restarted. This momentarily interrupts service on all live applications.

8 You must control ACD traffic routing on the Meridian 1 PBX so that the number of Meridian Mail voice channels that can present calls to ACCESS VSDNs of a given class does not exceed the number of active Meridian IVR logical channels configured for the given class. Otherwise, some calls will be answered only to be transferred to the Revert DN. Again, ACD Automatic Overflow provides the most convenient means of controlling call routing in the Meridian 1 PBX. You can change the Overflow DNs to prevent excess calls from being presented to logical channels of shared acquisition type.

9 You cannot set outbound applications to run exclusively against a particular Meridian IVR logical channel. Therefore, it is impossible to establish an isolated test environment for outbound applications on a live system.

10 Meridian IVR Application Management provides the capability to stop an application gracefully or forcefully. STOP FORCEFULLY idles logical channels that are hung due to User Function Cell failure to return or other application misbehavior. It does not recover logical channels that are out of service due to Meridian Mail Link or ACCESS problems.

11 To minimize service interruption when using STOP FORCEFULLY to recover hung Meridian IVR logical channels, load a separate instance of a given application (copied and suitably renamed) for each logical channel or small range of logical channel. Use STOP GRACEFULLY, monitor the channel board lights until only the hung channel is still busy, then use STOP FORCEFULLY to recover the hung channel.

12 Until you start the application for the range of channels again, customer calls are still presented to the stop channels, and each is answered and transferred back to the dialed DN which puts the caller to the back of the queue. You can replace the original CLID by the Agent Position ID of the transferring Meridian Mail agent.

13 When testing applications which use new or modified voice prompts, it is critical that you back up existing voice segment files by copying them to tape with the Meridian Mail ACCESS Prompt Transfer Tool.

14 You must create a dedicated test mailbox, and configure the Meridian IVR test channels to use the dedicated test mailbox. If you need to add or modify voice segments, you must copy the backup of the voice segment files currently in use into the test mailbox. You should test all changes thoroughly before copying the modified voice segment files into the mailbox that provides the voice prompts for customer traffic.

Note: Changing the mailbox for a single logical channel in Meridian IVR system configuration requires that you start all Meridian IVR channels so that the reconfiguration takes effect.

Executing your application

Once you save your application, you can load it then run it on a voice channel. You do not need to compile the application. However, you should discuss with your system administrator the resources and the testing configuration that your application needs.

Note: Any application you build with the Application Editor executes, no matter how erroneous the logic is. Consequently, you should test your application to ensure that your objectives are achieved during runtime prior to accepting live calls.

Closing and exiting your application

If you want to clear the drawing board without exiting the Application Editor, you can close the current application.

Procedure 3-26
Closing an application

1 Click on File with the left mouse button.

The Application Editor displays the File pull-down menu.

2 Click on Close with the left mouse button.

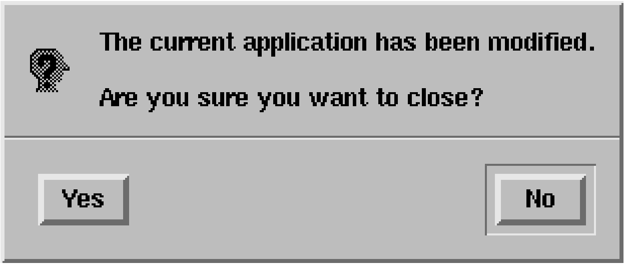
(As an alternative to steps 1 and 2, you can press <Ctrl+L>).

Unless you have made changes since last saving the application, the Application Editor clears the drawing board and displays its initial setup with only the START cell. If this is the case, the Application Editor displays the dialog box illustrated in Figure 3-34.

3 Select YES to close without saving the changes.

Or, select NO to cancel the close operation, and return to the drawing board.

Figure 3-34
Close or exit without save warning



The current application has been modified.

Are you sure you want to close?

Procedure 3-27
Exiting the Application Editor

1 Click on File with the left mouse button.

The Application Editor displays the File pull-down menu.

2 Click on Exit with the left mouse button.

(As an alternative to steps 1 and 2, you can press <Ctrl+E>).

The Application Editor does not perform a Save/Exit operation. Consequently, if you made changes to the application since the last save, the Application Editor displays the same dialog box illustrated in Figure 3-34.

3 Select YES to exit the Application Editor without saving the changes.

Or, select NO to cancel the exit operation, and return to the drawing board.

Deleting an application

Procedure 3-28
Deleting an application

1 Go to the UNIX shell window.

1 Change to the correct directory by typing **cd /u/ivr/apps**, and press <Enter>.

2 Type **rm filename.vpf,** and press <Enter>.

Note: If you created a Postscript version (that is, the application was printed), type **rm filename.ps,** and press <Enter>.

CAUTION!
Risk of losing application

Once you delete an application you cannot retrieve it.

Chapter 4: Creating a sample application

This chapter provides you with step-by-step instructions to create a sample application that you can load and run. In working through this chapter, you will acquire knowledge and experience that will help you to design and build your own applications. The chapter takes you through the following steps:

- Designing the application call-flow
- Selecting prompts
- Opening the Application Editor
- Creating the application file
- Creating and naming the cells
- Customizing the cells
- Saving and naming the application
- Printing the application
- Reviewing the application
- Testing the application
- Executing the application
- Editing the application

Chapter 3 presents all the rules for building applications; however, this chapter explains only the rules needed for constructing the sample application presented here.

> ***Note:*** This chapter shows you how to create applications using only the Application Editor.

Designing the application

The sample application has already been designed for you. This application, called “Leavemsg”, allows a caller to leave a message that is saved in a desired mailbox.

This example covers only the most basic functions needed by a message-gathering application. It does not provide for responding to unexpected results. Once you have followed the directions in this chapter to gain some experience with creating applications, you can use this sample application as a starting point to create other, more complex applications. You can also enhance this application so that it offers options on error conditions such as when a message cannot be recorded or sent, rather than the current state where the application hangs up the call when an error occurs.

Application specifics

The application is designed to do the following things:

1 Answer the call.

2 Prompt the caller to leave a message at the tone, and terminate the recording by pressing the # (pound) key.

3 Record the caller’s message.

4 Prompt the caller to choose one of three options: save, review, or delete the message.

5 If the caller chooses the save option, the application does the following:

   a Deposits the caller’s message in the destination mailbox.

   b Plays a prompt to inform the caller that the message has been delivered.

   c Removes the message from the application mailbox.

   d Terminates the call once the prompt (“Thank you. Good-bye”) is played.

6 If the caller chooses the review option, the application does the following:

a Plays back the recorded message to the caller.

b Allows the caller to re-record the message or continue, after playing back the message.

c Returns the caller to step 3 if the caller decides to re-record the message.

d Returns the caller to step 4, if the caller decides to continue, giving them the option to save, review, or delete the message.

7 If the caller chooses the delete option, the application does the following:

a Deletes the recorded message.

b Allows the caller to re-record the message or exit after deleting the message.

c Returns the caller to step 3 if they decide to re-record the message.

d Terminates the call after it plays the prompt (“Thank you. Good-bye”) if the caller decides to exit.

Figure 4-1 illustrates the basic call-flow for the Leavemsg application.

Figure 4-1
Basic call-flow for Leavemsg application

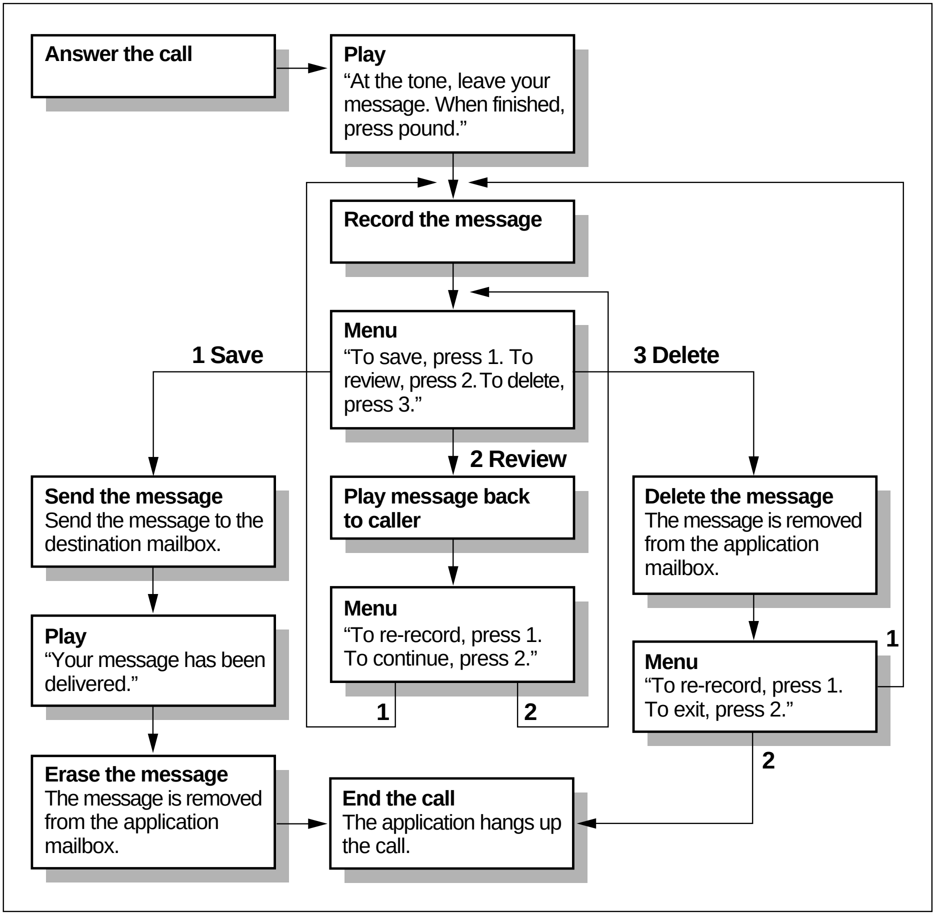
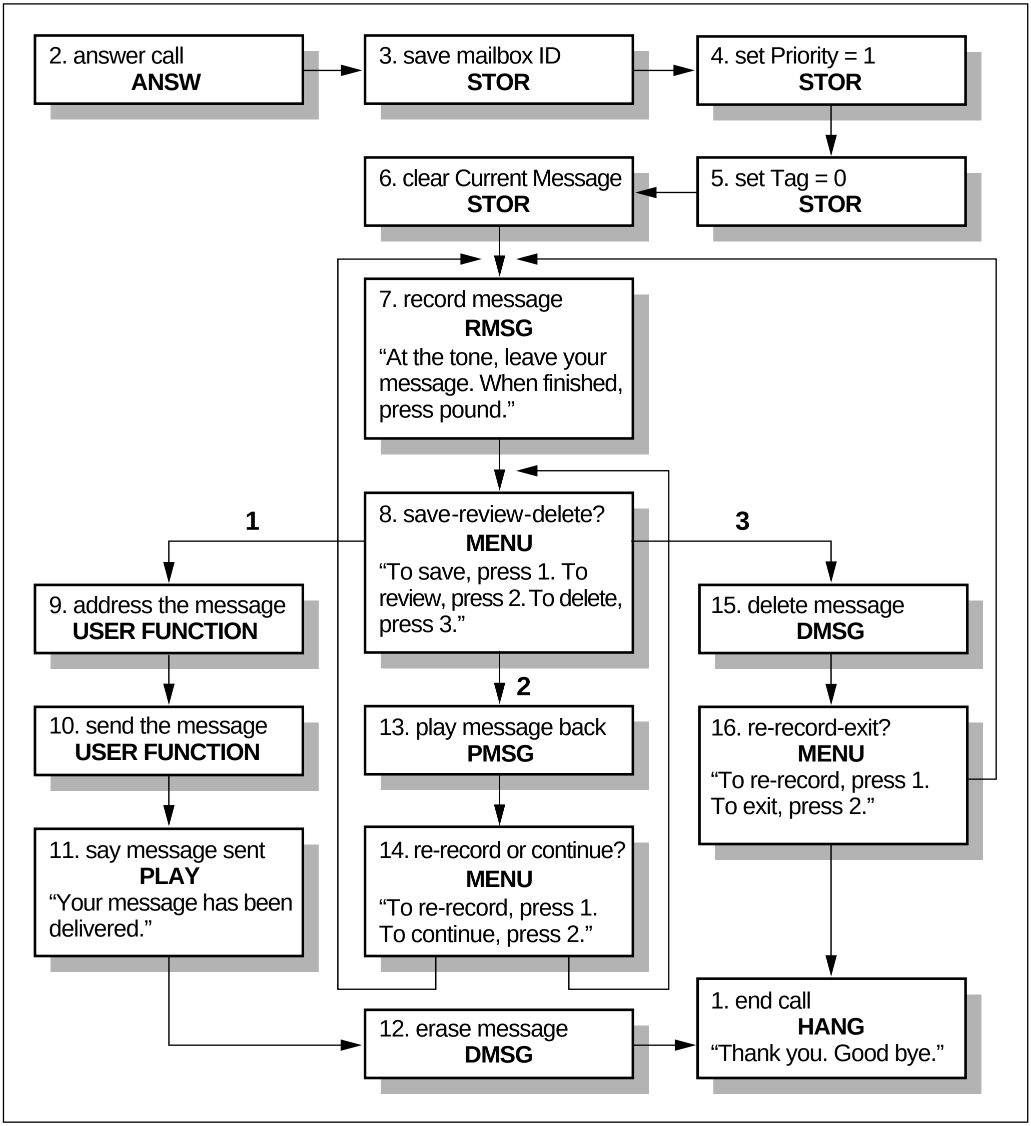

Answer the call
Play
"At the tone, leave your message. When finished, press pound."
Record the message
Menu
"To save, press 1. To review, press 2. To delete, press 3."
1 Save
3 Delete
2 Review
Send the message
Send the message to the destination mailbox.
Play message back to caller
Delete the message
The message is removed from the application mailbox.
Play
"Your message has been delivered."
Menu
"To re-record, press 1. To continue, press 2."
1
2
Menu
"To re-record, press 1. To exit, press 2."
1
2
Erase the message
The message is removed from the application mailbox.
End the call
The application hangs up the call.

Figure 4-2 converts the basic call-flow into a detailed Meridian IVR call-flow by adding all the main cells and all the prerequisite cells (that is, the STOR cells).

Figure 4-2
Meridian IVR call-flow for Leavemsg application



Call flow for Leavemsg

Figure 4-2 shows the call-flow for the Leavemsg application. You can also go to Figure 4-9 to see what the application looks like when it is finished. Follow this cell-by-cell explanation. Before the first cell in the application is executed, the dialed numbers Direct Inward Dial (DID) digits are received and stored in the DIGITS buffer.

Cell 1

The application plays “Thank you. Good-bye,” then hangs up the call.

> ***Note:*** Cell 1 is the HANG cell; however, it appears at the end of the call-flow rather than at the beginning so that any unconnected cell branches terminate here when you save the application, rather than at the ANSW cell.

Cell 2

The application answers the call.

Cell 3

The application takes the digits that are in the DIGITS buffer (that is, the number that was called) and copies the digits to the MAILBOX ID buffer. These digits are used to identify the mailbox that stores the message.

Cell 4

The application prioritizes the message for the ACCESS API in the User function. For this application, the priority specified as one means that the message has normal delivery.

Cell 5

The application tags the message for the ACCESS API in the User function. For this application, the tag is set to zero which means no special tags accompany the message.

Cell 6

The application clears the current message stored in the buffer.

Cell 7

The caller hears a prompt that says, “At the tone, leave your message. When finished, press pound.” The caller records a message, then presses the # (pound) key on the telephone to end the recording.

Cell 8

The caller hears the menu: “To save, press 1. To review, press 2. To delete, press 3.” The caller presses a digit to make a menu selection. The application branches to Cell 9, Cell 13, or Cell 15 depending on the menu selection.

Option 1: Save the message

Cell 9

The application addresses the message to the designated mailbox.

Cell 10

The application sends the message to the appropriate mailbox.

Cell 11

The caller hears a prompt that says, “Your message has been delivered.”

Cell 12

The application erases the current message after it has been sent. The application then ends the call.

Option 2: Review the message

Cell 13

The application plays the message back to the caller.

Cell 14

The application plays another menu. The choices are, “To re-record, press 1. To continue, press 2.” For the re-record option, the application returns to cell 7. For the continue option, the application returns to cell 8.

Option 3: Delete the message

Cell 15

The application deletes the message.

Cell 16

The caller is presented with a menu: “To re-record, press 1. To exit, press 2.” The application branches to Cell 7 to re-record, or to Cell 1 to exit.

Selecting prompts

Prompts are integral to your applications. They are the means by which you provide your callers with information and guide them through the steps they should perform, such as making a selection from a menu or leaving a message in a mailbox. Meridian IVR has standard prompts, listed in Appendix A, that are commonly used in applications. You do not have to record any prompts because Leavemsg uses only the following standard prompts:

• 32	"pound"
• 55	"Thank you. Good-bye."
• 59	"At the tone, leave your message. When finished, press..."
• 60	"Your message has been delivered."
• 66	"To save, press 1. To review, press 2. To delete, press 3."
• 67	"To re-record, press 1. To continue, press 2."

Application Editor

You are now ready to open the Application Editor and either create a new application or work on an existing application.

> ***Note:*** You will be developing your applications in the default directory */u/ivr/apps*. Although it is not necessary, you can create a directory specifically for developing applications. Refer to the *Meridian IVR System Administration Guide* (NTP 555-9001-300) for instructions. Keep in mind that if you create a development directory, you must always either move or copy your application to the default directory (you may consider this unnecessary work).

Procedure 4-1
Opening an existing application

1 Click on File with the left mouse button.

The File pull-down menu appears.

2 Click on Open with the left mouse button.

The Application Browser similar to that shown in Figure 4-3 appears.

3 Select the directory you are using for developing applications from the directories scroll area on the Application Browser by clicking in the directory scroll area with the left mouse button (the default directory is */u/ivr/apps*).

4 If the desired file is not in the default directory, click on the Filter button of the Application Browser with the left mouse button.

The list of files for this directory appears in the Files scroll area shown in Figure 4-3.

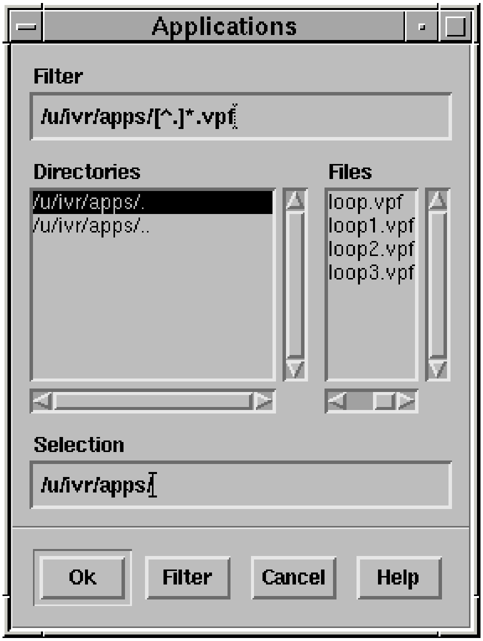
5 Select the desired file.

The application name appears in the Selection field of the Application Browser.

6 Click on OK with the left mouse button.

The application appears in the drawing board under the Current Application.

Figure 4-3
Select an application directory screen



Applications

Filter

/u/ivr/apps/[^.]*.vpf

Directories

/u/ivr/apps/.
/u/ivr/apps/..

Files

loop.vpf
loop1.vpf
loop2.vpf
loop3.vpf

Selection

/u/ivr/apps/

Creating new application files

The application is stored on the application processor as a file. You can name the application which you are going to develop now or when you are finished.

Procedure 4-2
Creating a new application file

1 Click on the File menu with the left mouse button.

The File pull-down menu appears.

2 Click on “Save As” with the left mouse button.

The Application Browser appears.

3 Enter the new file name, “Leavemsg,” in the Selection field of the Application Browser.

4 Make sure you are in the right directory and click on the name with the left mouse button.

The name appears in the Selection field.

5 Click on OK with the left mouse button.

6 If you want to overwrite the existing application file name, press YES.

If you want to give the application a new file name, press NO and type in a new application file name.

The new file name appears in the Files scroll area of the Application Browser and the Current Application field of the Application Editor window.

Editing the default cell

Once you have created a file, you can edit the values on the Default Cell screen which, in general, is used for the following purposes:

- to establish default values for some of the parameters in cells throughout the application

  Note: The Default Cell screen is not actually a cell; it performs functions that affect the cells of the application.

- to make changes to the Default Cell parameters (double-click on the START cell)

For our sample application, you do not need to change the default values for parameters.

Creating and naming cells

The next step is to create all of the cells in the application and assign a name to each one. The cells used in the Leavemsg application are listed in Table 4-1. The next three procedures show you how to create and rename the first three cells in your application.

> ***Note:*** When you are naming cells, make sure that each cell has a unique name. The Application Editor assigns a number for each cell you create. You can see the cell number in the Application Editor, when you print the application, or through the “dappl” utility. Refer to the *Meridian IVR Maintenance and Diagnostics Guide* (555-9001-500) for more information on this utility. The reports and debugging tools reference the cell number rather than the name.

Table 4-1
Cells in the leavemsg application

Cell number	Cell type	Cell description	Name
Cell 1	HANG	Hang up the call.	end call
Cell 2	ANSW	Answer call.	answer call
Cell 3	STOR	Store.	save MAILBOX ID
Cell 4	STOR	Store priority for first User function.	set PRIORITY=1
Cell 5	STOR	Set tag for second User function.	set TAG= 0
Cell 6	STOR	Clears the current message.	clear CURRENT MESSAGE
Cell 7	RMSG	Record message.	record message
Cell 8	MENU	Play menu and get data.	save-review-delete?
Cell 9	USER (Predefined “access”user function)	User function which addresses message to appropriate mailbox.	address the message

Table 4-1
Cells in the leavemsg application (continued)

Cell 10	USER (Predefined “access”user function)	User function which sends the message to the appropriate mailbox.	send the message
Cell 11	PLAY	Play prompts.	say message sent
Cell 12	DMSG	Delete message.	erase message
Cell 13	PMSG	Play message.	play message back
Cell 14	MENU	Play menu and get data.	re-record or continue?
Cell 15	DMSG	Delete message.	delete message
Cell 16	MENU	Play menu and get data.	re-record-exit?

Procedure 4-3
Creating and naming the HANG cell

1 Click on the HANG cell in the cell palette with the left mouse button.

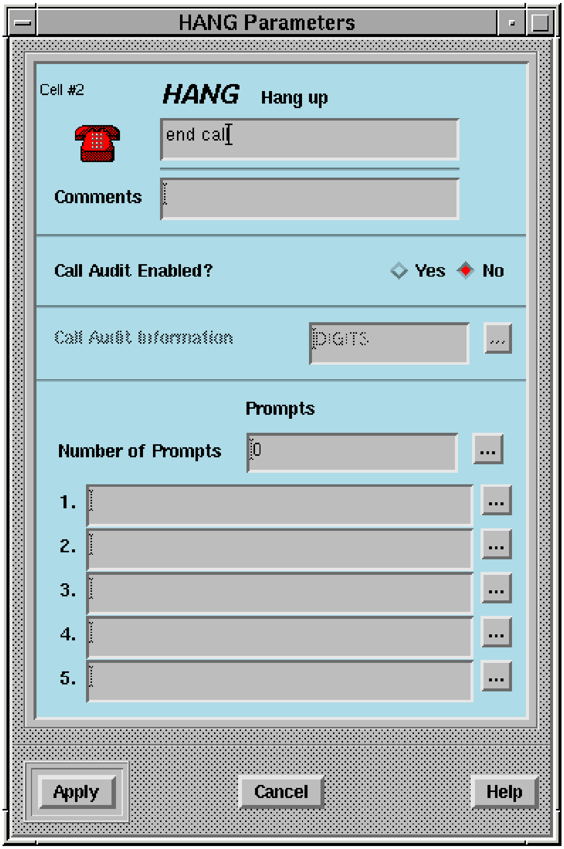
2 Click beside the START cell in the drawing board with the middle mouse button.

The HANG cell appears in the drawing board of the Application Editor window .

Note: Place the HANG cell beside the START cell, rather than within the direct path of the call-flow. Although it is the first cell, you move it to the end to follow the natural progression of the application. (All calls end by hanging up; therefore, the application is tidier if the HANG cell appears as the last cell in the call-flow). Double-click on the HANG cell with the left mouse button.

3 Double-click on the HANG cell with the left mouse button to access the parameters window as shown in Figure 4-4.

Figure 4-4
HANG cell parameters



Cell #2

HANG Hang up

end call

Comments:

Call Audit Enabled? ◇ Yes ◆ No

Call Audit Information: DIGITS

Prompts

Number of Prompts: 0

1.
2.
3.
4.
5.

4 Enter the words “end call” in the first field in the HANG cell parameter window to name the cell.

5 Click on Apply with the left mouse button to confirm your entry.

The cell name “end call” appears in the first field.

6 Once you have created all of the other cells, move the HANG cell to the end of the application before connecting the cells.

Procedure 4-4
Creating and naming the ANSW cell

1 Click on the ANSW cell with the left mouse button.

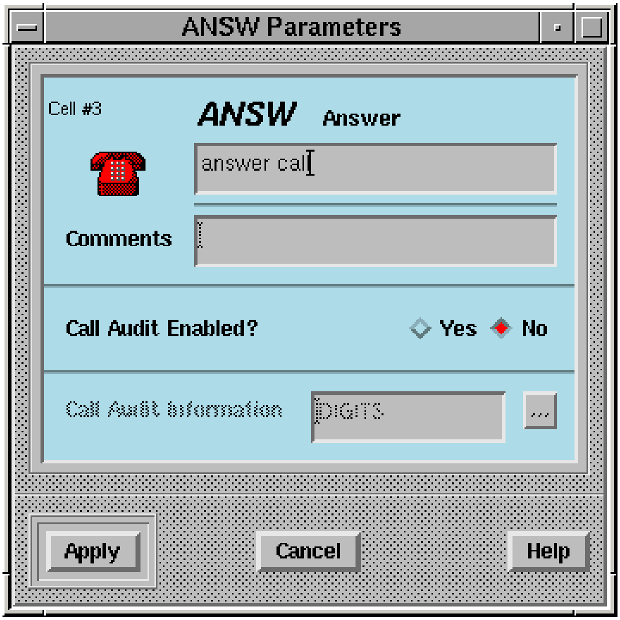
2 Click beneath the START cell in the drawing board with the middle mouse button.

The ANSW cell appears on the Application Editor window.

3 Double-click on the ANSW cell with the left mouse button.

The ANSW parameters window appears as shown in Figure 4-5.

Figure 4-5
ANSW cell and parameter



ANSW Parameters

Cell #3

ANSW Answer

answer call

Comments

Call Audit Enabled? ◇ Yes ◆ No

Call Audit Information DIGITS

4 Enter **answer call** in the first field in the ANSW cell parameter window to name the cell.

5 Click on Apply with the left mouse button to confirm your entry.

The cell name “answer call” appears in the first field.

Procedure 4-5
Creating and naming the STOR cell

1 Click on the STOR cell in the cell palette with the left mouse button.

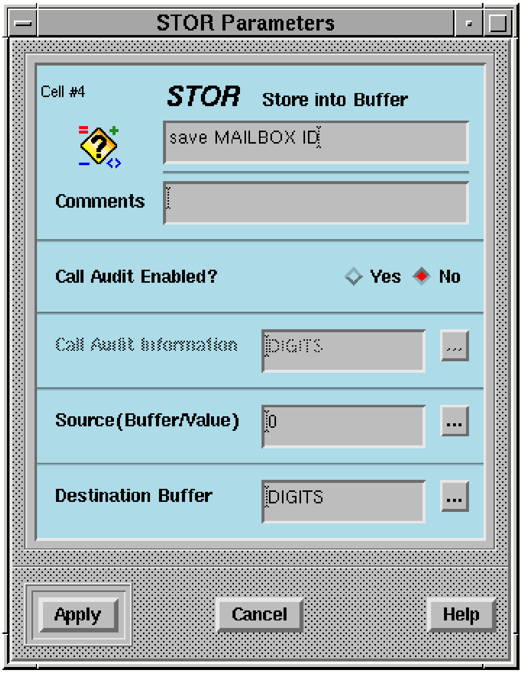
2 Click beneath the ANSW cell in the drawing board with the middle mouse button.

The STOR cell appears on the drawing board of the Application Editor window.

3 Double-click on the STOR cell with the left mouse button.

The STOR parameters window appears as shown in Figure 4-6.

Figure 4-6
STOR cell and parameter



STOR Parameters

Cell #4 **STOR** Store into Buffer

save MAILBOX ID

Comments

Call Audit Enabled? ○ Yes ● No

Call Audit Information DIGITS ...

Source(Buffer/Value) 0 ...

Destination Buffer DIGITS ...

4 Enter **save MAILBOX ID** in the first field in the STOR cell parameter window to name the cell.

5 Click on Apply with the left mouse button to confirm your entry.

The cell name “save MAILBOX ID” appears in the first field.

Creating and naming the remaining cells

Repeat this process of choosing cell types and naming cells until you have done all of the cells in this application. Refer to Table 4-1 for a list of the cell types and the names for each. You can also refer to Figure 4-9 as it provides you with a clearer idea of how to place the cells.

Customizing cells

You now can customize all the cells which you have created. You change the values of the parameters and tables, and connect cells to meet the specific needs of this application.

Customizing the HANG cell (end call)

Parameters

This cell plays one hang-up prompt which you must list listed in the Hang-up Prompt Table. Once you add a prompt, the Number of Prompts gets updated automatically once the cell is saved.

> ***Note:*** This HANG cell signals successful completion of the call. To better detect errors, we recommend that you add additional HANG cells to terminate the extra branches which various cells have (for example, Error, Invalid Input, Time-out). Each HANG cell should play a unique prompt (for example, a number), thereby providing an indication of where an error may have occurred. In this sample application, four additional HANG cells have been included. See Figure 4-9 to see where to place these additional HANG cells and where to make the appropriate connections.

Tables

The Hang-up Prompt Table has one entry: 55. (This prompt says, “Thank you. Good-bye.”)

> ***Note:*** The connections to this cell occur last. Proceed with the customization of all the other cells.

Customizing the ANSW cell (answer call)

The ANSW cell type has no parameters and no tables. If you refer back to the call-flow diagram in Figure 4-2, you will see that the next cell for ANSW cell should be STOR. To connect the ANSW cell to the START cell, and the STOR cell to the ANSW cell follow these steps:

Procedure 4-6
Connecting the ANSW cell to the START cell

1 Click on the connector at the bottom of the START cell with the left mouse button.

The pointer changes to a pencil cursor.

2 Point the pencil cursor to the connector at the top of the ANSW cell and click on the left mouse button to create the source-destination pair.

The Application Editor displays the connection lines between these two cells.

Procedure 4-7
Connecting the STOR cell to the ANSW cell

1 Click on the connector at the bottom of the ANSW cell with the left mouse button.

The pointer changes to a pencil cursor.

2 Point the pencil cursor to the top of the STOR cell, then click on the left mouse button to create the source-destination pair.

The Application Editor displays the connection lines between these two cells.

Customizing the first STOR cell (save MAILBOX ID)

There are no tables for this cell, but you can modify the parameters, if necessary. Consider the STOR cell’s two parameters:

- Source (Buffer or Value)
- Destination Buffer

The purpose of this STOR cell is to store digits from the DIGITS buffer into the MAILBOX ID buffer. The DIGITS buffer stores the number called to start the application. Therefore, the Source Buffer is the DIGITS buffer and the Destination Buffer is the MAILBOX ID buffer.

As you can see from the call-flow diagrams (see Figure 4-2 and Figure 4-9), the next cell is another STOR cell.

Procedure 4-8
Customizing the first STOR cell

1 Double-click on the STOR cell with the left mouse button to open its parameter window.

2 Select the field that says “Source (Buffer or Value).”

3 Enter “DIGITS” into this field, which changes the setting from the initial value “0”.

You can also click on the push button to the right of this field to select a value.

The “Existing Buffers” browser, with a list of valid system buffers, appears.

4 Select “DIGITS” and click on Apply with the left mouse button.

The browser disappears and “DIGITS” replaces the initial value.

5 Select the field that says “Destination Buffer”.

6 You need to change the setting from the initial value “DIGITS” to “MAILBOX ID”. Enter “MAILBOX ID” into this field.

You can also select a value by clicking on the push button to the right of this field to get the “Existing Buffers” browser.

7 Move the cursor to “MAILBOX ID.” Press Apply.

The browser disappears and “MAILBOX ID” replaces the initial value.

Note: You may prefer to use the browser because if you misspell a buffer name, you may have problems later on.

8 Click on Apply with the left mouse button.

The Application Editor saves your changes and this window disappears.

9 The next cell for the STOR cell is the second STOR cell (set Priority=1). Connect these two cells the same way you did for the ANSW cell.

Customizing the second STOR cell (set PRIORITY=1)

Parameters

The Source (Buffer/Value) should be 1.
The Destination Buffer should be PRIORITY, which you must type in because it is a user-defined buffer, not a system buffer. When you enter the name ensure that the name, spelling, spacing, and capitalization are accurate.

> ***Note:*** The new user-defined buffer name appears in the “Existing Buffers” browser once you have saved the cell.

Next Cell

The next cell is the STOR cell (set TAG=0).

Customizing the third STOR cell (set TAG=0)

Parameters

The Source (Buffer/Value) should be 0.
The Destination Buffer should be TAG. You must type this in because it is a user-defined buffer, not a system buffer.

Next Cell

The next cell is the STOR cell (clear CURRENT MESSAGE).

Customizing the fourth STOR cell (clear CURRENT MESSAGE)

Parameters

The Source (Buffer/Value) should be 0.
The Destination Buffer should be CURRENT MESSAGE. Since this is a system buffer, you can select it from the list under “Existing Buffers”. You can also type it in.

Next Cell

The next cell is the record message (RMSG) cell.

Customizing the RMSG cell

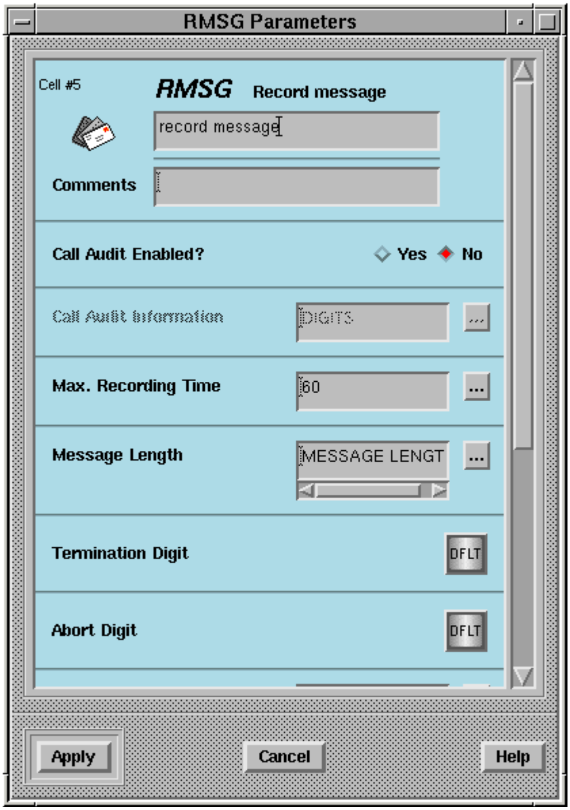
There are a few more features of the editor that you should try, for example modifying the RMSG cell.

Procedure 4-9
Modifying the RMSG cell

1 Double-click on the RMSG cell with the left mouse button.

The record message parameters window appears as shown in Figure 4-7:

Figure 4-7
RMSG cell



RMSG Parameters

Cell #5

RMSG Record message

record message

Comments

Call Audit Enabled? ○ Yes ● No

Call Audit Information DIGITS

Max. Recording Time 60

Message Length MESSAGE LENGT

Termination Digit DFLT

Abort Digit DFLT

As you can see on the window, this cell has many parameters. If you want, you can turn to Chapter 7, “Cell catalog,” and read the descriptions of the parameters, next cells, and tables.

2 Click on the down arrow key to access the Prompts section.

3 Move the cursor to field 1 and type in 59. This is the standard prompt number which represents the phrase, “At the tone, leave your message. When finished, press...”

4 Click on Apply with the left mouse button.

The text “At the tone, leave your message. When finished, press...” appears automatically.

5 Move the cursor to field 2 and type in 32. This is the standard prompt number which represents “pound.”

This next prompt tells the caller to press the termination digit. In this cell, the # key is the termination digit. No other changes are required.

Note: The Maximum Recording Time is 60 seconds and the Abort Digit, Abort Prompt, and Termination Digit are set to DFLT which indicates that they use the values set in the Default Cell.

6 Click on Apply with the left mouse button.

The Application Editor saves your changes and this window disappears.

7 Connect the Success branch from the RMSG cell to the MENU cell in the same way that you did above.

Note: Do not connect the other branches (for example, Error, Time-out, and so on) until you add the additional HANG cells.

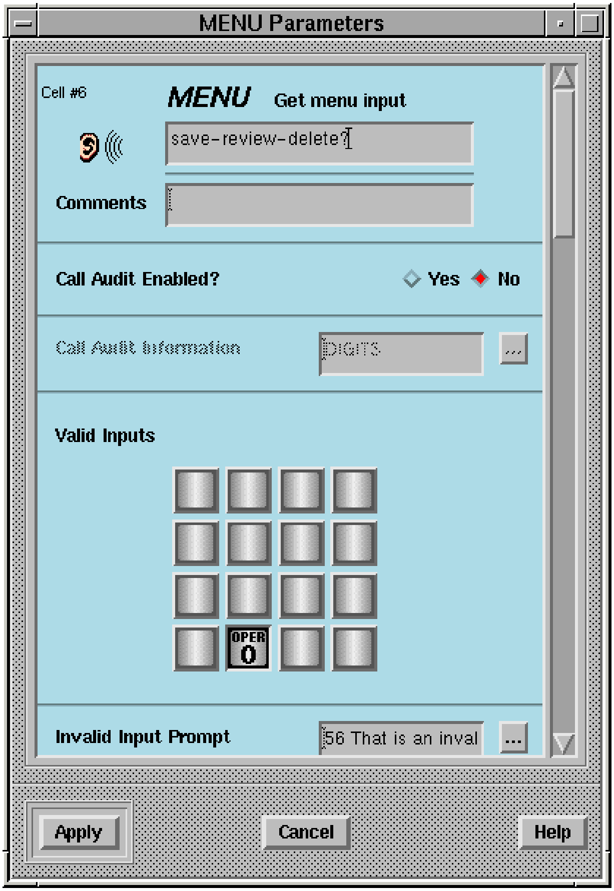
Customizing the remaining cells

Now that you have customized three different cell types, you know how to do the same for the remaining cells.

Customizing the first MENU cell

Figure 4-8 shows the MENU cell and its parameters.

Figure 4-8
MENU cell and parameters



MENU Parameters

Cell #6 *MENU* Get menu input

save-review-delete?

Comments

Call Audit Enabled? ◇ Yes [✓ No]

Call Audit Information DIGITS

Valid Inputs

Invalid Input Prompt 56 That is an inval

Parameters

You should change the Valid Inputs to 1, 2, and 3. You can select these values by clicking on the keypad. You also need to remove the 0, which you can do by clicking on it.

Tables

There is one table, the Play Prompt Table.

- The Play Prompt Table has one entry: 66. (This prompt says, “To save, press 1. To review, press 2. To delete, press 3.”)

You must replace the prompt 0 with 66 which you can do by using the delete key or by highlighting the prompt number and text, and typing over it. Remember that the Number of Prompts is updated automatically, and the text appears beside the prompt number automatically once you click on Apply.

You do not need to change any other parameters.

Next Cells

The next cell for every error branch is the HANG cell (end call).
The next cell for menu choice 1 is the USER cell (address the message).
The next cell for menu choice 2 is the PMSG cell (play message back).
The next cell for menu choice 3 is the DMSG cell (delete message).

Customizing the first USER cell (address the message)

The purpose of this cell is to address the message.

Parameters

Type in **access** for the User Function Name.
Set the Function Code to 3.
Select Reply Codes 0 and 1.
The Input Buffers should be CURRENT MESSAGE and MAILBOX ID (in that order). These can be selected or typed in.
There are no Output Buffers.

Next Cells

The next cell for Reply Code 0 that indicates user function success is the second USER function (send the message).
The next cell for Reply Code 1 that indicates user function failure condition is the DMSG cell (erase message).

> ***Note:*** For more information on the ACCESS user function, refer to Appendix B.

Customizing the second USER cell (send the message)

The purpose of this cell is to send the message to the appropriate mailbox.

Parameters

Type in **access** for the User Function Name.
Set the Function Code to 4.
Select Reply Codes 0 and 1.
The Input Buffers should be CURRENT MESSAGE, PRIORITY, and TAG (in that order). These can be selected or typed in.
There are no Output Buffers.

Next Cells

The next cell for Reply Code 0 is the PLAY cell (say message sent).
The next cell for Reply Code 1 is the DMSG cell (erase message).

> ***Note:*** For more information on the ACCESS user function, refer to Appendix B.

Customizing the PLAY cell (say message sent)

Parameters

You do not need to change any of the initial values.

Tables

There is one table: the Play Prompt Table. It has one entry: 60. (This prompt says, “Your message has been delivered.”)

Next Cell

The next cell for the Success branch is the DMSG cell (erase message).
The next cell for the Error branch is a HANG cell.

Customizing the first DMSG cell (erase message)

Parameters

There are no parameters.

Next Cell

The next cell for both branches is a HANG cell.

Customizing the PMSG cell (play message back)

Parameters

You do not need to change any of the initial values.

Next Cell

The next cell for the Success branch is the second MENU cell (re-record or continue?).
The next cell for the Error branch is a HANG cell.

Customizing the second MENU cell (re-record or continue?)

Parameters

You should change the Valid Inputs to 1 and 2. There are two valid inputs (menu choices) for this cell. You can select these values by clicking on the keypad. You also need to remove the 0 which you can do by clicking on it. You do not need to change any of the other parameter values.

Tables

There is one table, the Play Prompt Table.

- The Play Prompt Table has one entry: 67. (This prompt says, “To re-record, press 1. To continue, press 2.”)

You do not need to change any other parameters.

Next Cell

The next cell for the Error, Invalid Input, and Time-out branches is a HANG cell.
The next cell for menu choice 1 is the RMSG cell (record message).
The next cell for menu choice 2 is the MENU cell (save-review-delete?).

Customizing the second DMSG cell (delete message)

Parameters

There are no parameters.

Next Cells

The next cell for the Success branch is the MENU cell (re-record-exit?).
The next cell for the No Delete branch is a HANG cell.

Customizing the third MENU cell (re-record-exit?)

Parameters

You should change the Valid Inputs to 1 and 2 because there are two valid inputs (menu choices). You can select these values by clicking on the keypad. You also need to remove the 0 by clicking on it. You do not need to change any of the other parameter values.

Tables

You need to customize the Play Prompt Table.

- The Play Prompt Table has one entry: 67. (This prompt says, “To re-record, press 1. To continue, press 2.”)

  Note: Ideally, the prompt should say, “To re-record, press 1. To exit, press 2”; however, recording and storing prompts is not part of this exercise. Just use standard prompt 67.

Next Cells

The next cell for the Error, Invalid Input, and Time-out branches is a HANG cell.
The next cell for menu choice 1 is the RMSG cell (record message).
The next cell for menu choice 2 is the HANG cell (end call).

Saving, printing, and reviewing the application

Now that you have customized all the cells, you can save, print, and review your application.

> ***Note:*** You need a Postscript printer to print your application.

Procedure 4-10
Saving the application

1 If you have already named your application, click on File with the left mouse button.

The File pull-down menu appears.

2 Click on Save with the left mouse button.

The Application Editor saves your application.

If you are naming your application, select “Save as” from the File menu.

The Application Browser appears.

3 Make sure you are in the desired directory. Type **Leavemsg** at the end of the path in the Selection field and press OK.

4 If you want to overwrite the existing application, click on YES with the left mouse button.

If you want to give the application a new name, click on NO with the left mouse button and type in a new application name.

The Application Editor saves your application under the name you give it.

Procedure 4-11
Printing the application

1 Click on File with the left mouse button.

The File pull-down menu appears.

2 Click on the Export to Postscript option with the left mouse button.

The Application Editor displays a pull-down menu with options to postscript either the entire application, the call-flow, or the cell contents.

(As an alternative to steps 1 and 2, you can press the <Ctrl+P> keyboard accelerator.)

3 Click on one of the menu options with the left mouse button .

The Application Editor generates a Postscript file (in the /u/ivr/apps directory) with the same name as your application (that is, with the .ps extension).

4 Move the pointer to another window running the system shell.

5 Change to the correct directory by typing **cd /u/ivr/apps** and press <Enter>.

6 At the command prompt, type **lp /u/ivr/apps/LeaveMsg.ps.**

Procedure 4-12
Reviewing the application

1 Click on File with the left mouse button.

The File pull-down menu appears.

2 Click on Open with the left mouse button.

The application directory appears.

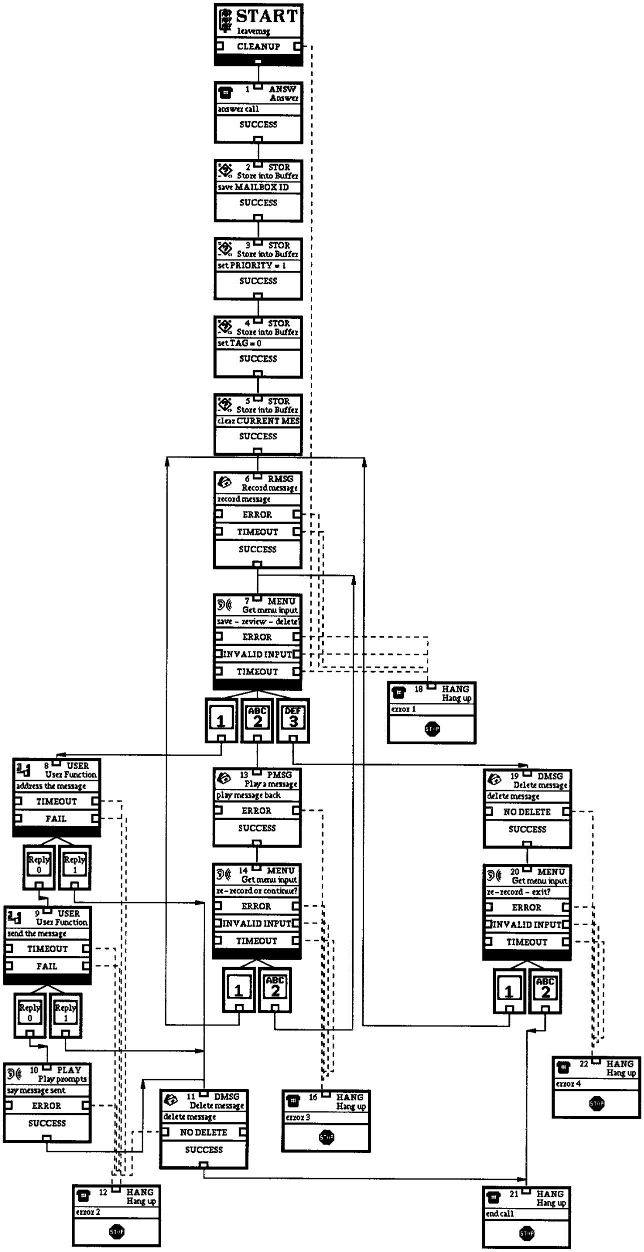
3 Select Leavemsg from the Files scroll area.

The name Leavemsg appears at the end of the path in the Selection field.

4 Click on OK with the left mouse button.

The application appears on the drawing board of the Application Editor's window, as shown in Figure 4-9.

Figure 4-9
Leavemsg application



Testing the application

Once you have reviewed the application, you are ready to load and test it. If your application is not in the default directory */u/ivr/apps*, you must place it here by copying or moving it because this is the only directory from which Meridian IVR can load applications.

> ***Note:*** Leave a copy of the application in your development directory so that you can use it as the basis for building the next version of the application.

If you have developed your application in a separate directory, you must copy it to the *apps* directory to run it.

Procedure 4-13
Copying the application

1 Click on File with the left mouse button.

The File pull-down menu appears.

2 Click on “Save As” with the left mouse button.

3 To change directories, select the appropriate directory path in the Directories field and click on Filter with the left mouse button. You may have to go through a few directory levels to get to */u/ivr/apps*.

4 Once you get to */u/ivr/apps*, you may have to select the correct filename from the Files list if more than one filename is listed, or you can type the name of your file in the Selection box.

5 Click on OK with the left mouse button.

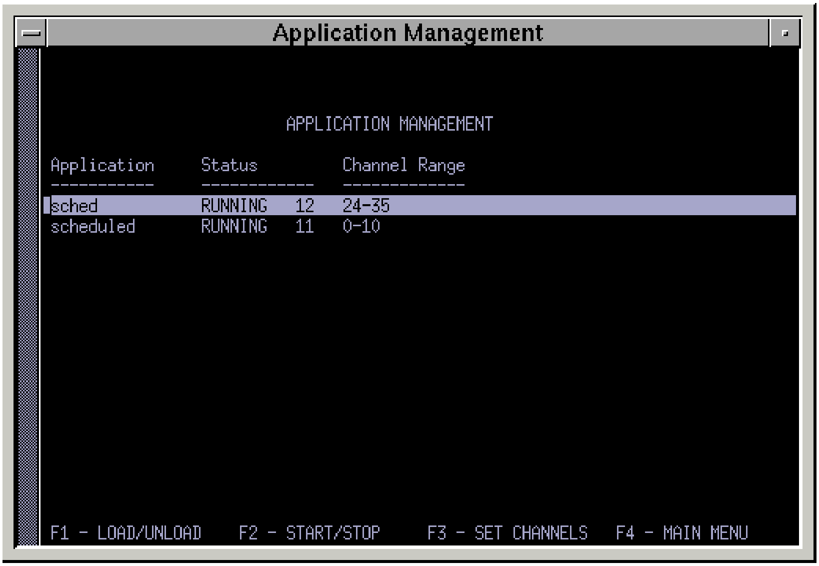
The file is now in the directory that allows you to run the application.

Procedure 4-14
Loading the application

1 From the Meridian IVR Application Management Menu, press <F3> for “Application”, or move the cursor to “Application Management” and press <Enter>.

The Application Management screen appears as shown in Figure 4-10.

Figure 4-10
Application management screen

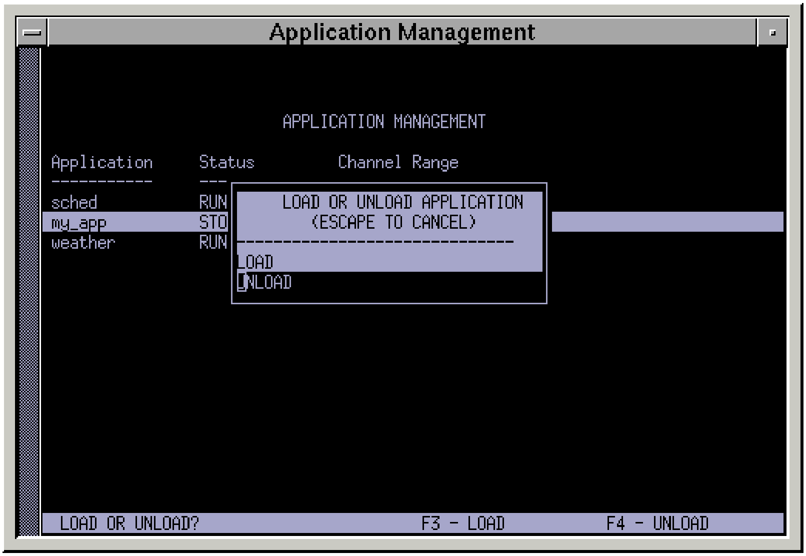


```
                         APPLICATION MANAGEMENT

Application     Status          Channel Range
-----------     ------------    -------------
sched           RUNNING   12    24-35
scheduled       RUNNING   11    0-10









F1 - LOAD/UNLOAD    F2 - START/STOP     F3 - SET CHANNELS   F4 - MAIN MENU
```

2 Press <F1> for “Load/Unload.”

The pop-up menu appears as shown in Figure 4-11.

Figure 4-11
Load or unload an application pop-up menu

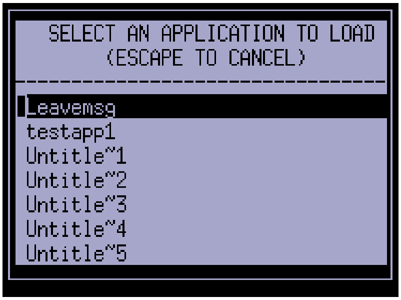


```
                    APPLICATION MANAGEMENT

Application      Status         Channel Range
-----------      ---
sched            RUN  LOAD OR UNLOAD APPLICATION
my_app           STO     (ESCAPE TO CANCEL)
weather          RUN  ------------------------------
                      LOAD
                      UNLOAD

 LOAD OR UNLOAD?                    F3 - LOAD          F4 - UNLOAD
```

3 Press <F3>, or move the cursor to “Load” and press <Enter>.

The system displays the pop-up menu that displays a list of all applications in the apps subdirectory (see Figure 4-12).

Figure 4-12
Select an application to load pop-up menu



```
  SELECT AN APPLICATION TO LOAD
        (ESCAPE TO CANCEL)
---------------------------------
 Leavemsg
 testapp1
 Untitle~1
 Untitle~2
 Untitle~3
 Untitle~4
 Untitle~5
```

4 Move the cursor so that it highlights the name of the application which you want to load (for example, Leavemsg), then press <Enter>.

The Application Management screen reappears as shown in Figure 4-13.

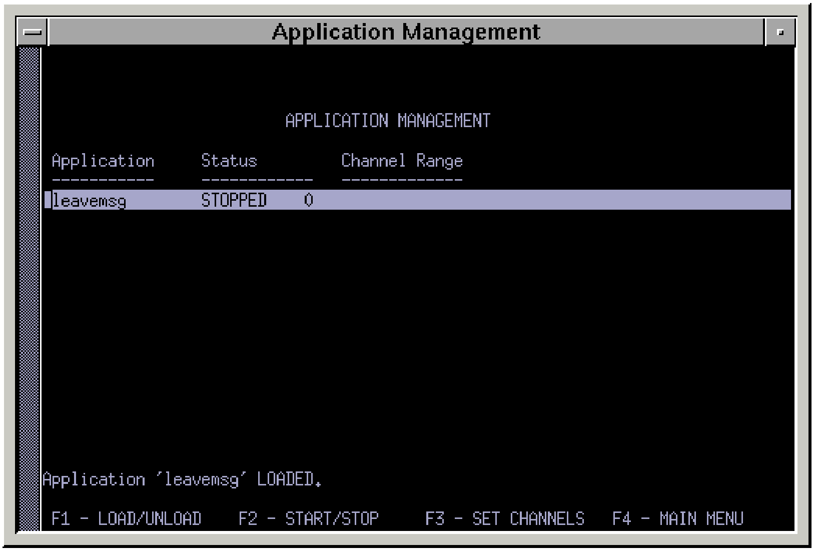
The name of the application appears in the "Applications" column to show that the application has been loaded.

The word STOPPED appears in the "Status" column to signify that the application is stopped.

After the word STOPPED, the number 0 appears indicating that the application has not been loaded on any channels.

If you select your Leavemsg application, you will see the Application Management screen as shown in Figure 4-13.

Figure 4-13
Loaded application



Application Management

```
                    APPLICATION MANAGEMENT

Application     Status         Channel Range
-----------     ------------   -------------
leavemsg        STOPPED    0

Application 'leavemsg' LOADED.

F1 - LOAD/UNLOAD    F2 - START/STOP     F3 - SET CHANNELS   F4 - MAIN MENU
```

The following message appears:

Application "Leavemsg" LOADED.

Procedure 4-15
Assigning channels

1 Press <F3> for "Set Channels."

You see the following prompt:

```
Enter The Channel Range:
```

2 Type a channel number or a range of channel numbers and press <Enter>.

To type a range, type the starting number, a hyphen, and the ending number. Use commas to separate ranges. For example, to assign the application to channels 10, 11, 12, 13, 17, 18, and 19, type the following without spaces:

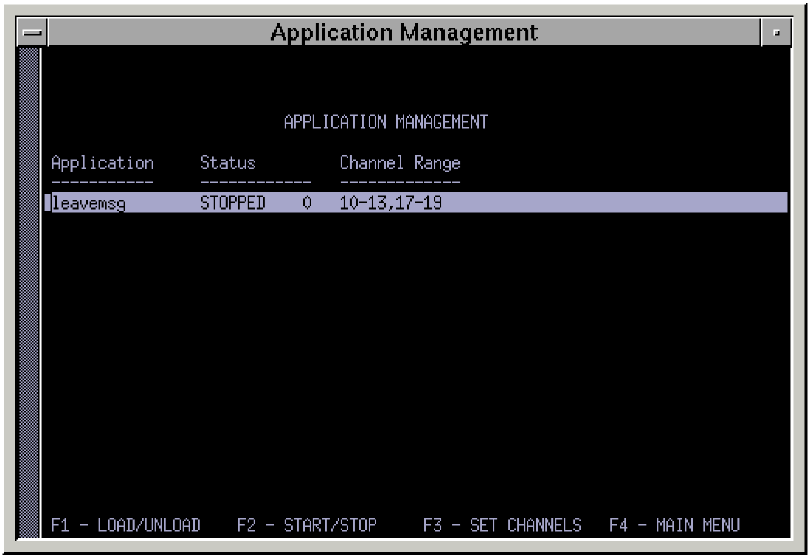
10-13,17-19

The range you choose appears in the "Channel Range" column of the Application Management screen as shown in Figure 4-14.

If you choose a channel where another application has already been loaded, you will see an error message instead.

Notice that the Status column still displays "STOPPED" indicating that the application is not running on any channels yet.

Figure 4-14
Application assigned to channels



```
                         APPLICATION MANAGEMENT

Application      Status          Channel Range
-----------      ------------    -------------
leavemsg         STOPPED    0    10-13,17-19






F1 - LOAD/UNLOAD    F2 - START/STOP     F3 - SET CHANNELS   F4 - MAIN MENU
```

Starting an application

Once you have assigned the channels, you can start your application.

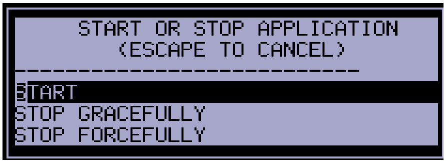
Procedure 4-16
Starting an application

1 If necessary, use the arrow keys to move the cursor until you highlight the name of the application that you want to start.

2 Press <F2> for “Start/Stop.”

You then see the start or stop application pop-up menu, as shown in Figure 4-15.

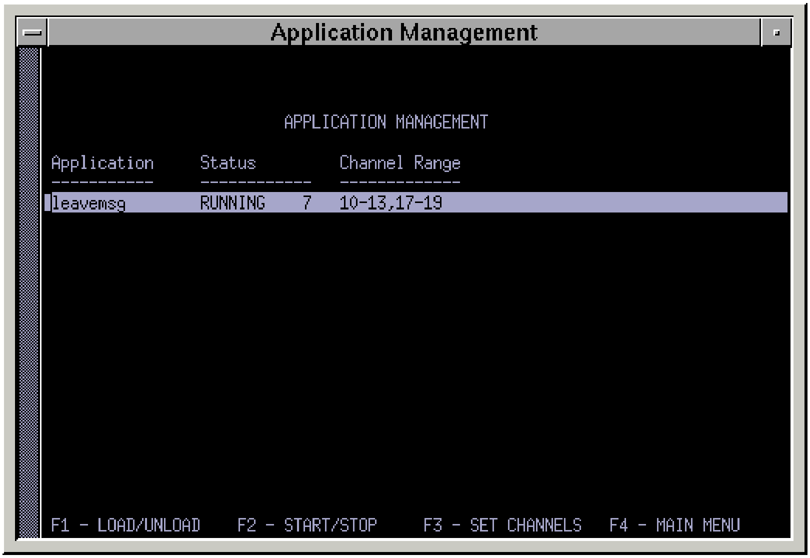
Figure 4-15
Start or stop application pop-up menu



```
     START OR STOP APPLICATION
        (ESCAPE TO CANCEL)
---------------------------
START
STOP GRACEFULLY
STOP FORCEFULLY
```

The Status column of the application’s status changes from STOPPED to RUNNING (see Figure 4-16), and also shows the number of channels on which the application is running.

Figure 4-16
An application running on seven channels



```
                    APPLICATION MANAGEMENT

Application     Status         Channel Range
-----------     ------------   -------------
leavemsg        RUNNING    7   10-13,17-19




F1 - LOAD/UNLOAD    F2 - START/STOP     F3 - SET CHANNELS   F4 - MAIN MENU
```

Once you have tested your application and sent a message to the designated mailbox, you can call into Meridian Mail, log into the system with the appropriate mailbox ID and password, and listen to the new messages.

Executing the application

Place a call to a destination DN forwarded to the application ACD-DN. The caller should hear “At the tone, leave a message...” When a message has been recorded and sent to the destination mailbox, and the application ends, log in to the destination DN’s Meridian Mailbox with the appropriate mailbox ID and password. The new message should have been deposited and can be played back.

Configuring a mailbox

Before you can test your newly created sample application, you must configure the switch and set up a mailbox for each DN. Consult your Meridian Mail guides on how to configure the switch and set up a mailbox.

> ***Note:*** A Meridian Mail mailbox must exist for each destination to be dialed.

Configuring Destination DNs

You must forward all possible destinations to the ACD DN on which you run the application. Consult your Meridian 1 guides on how to configure the Destination DN.

Editing the application

After you have tested an application, you can make changes. When you modify an application that has already been installed, remember to edit the copy that is in your development directory, *not* the running copy in the *apps* directory. You can edit an application in the same way that you create a new application. When you are editing, you can create new cells. You can also modify any existing cell by changing its name, type, parameters, tables, and next cells.

Chapter 5: Creating user functions

User functions, an advanced feature of Meridian IVR, allow you to interface "C" language code to a Meridian IVR application to perform activities which the ordinary cell types cannot do.

Up to 10 input buffers, along with one function code, can be passed into user functions. There are 99 different function codes; therefore, each user function can perform up to 99 different functions. Up to 10 output buffers, along with 10 different reply codes, can be returned from each user function. These reply codes allow different branches to be taken depending on the response received from the user function.

To utilize user functions effectively, you should be familiar with "C" programming language and the operating system running on the Meridian IVR application module. You should understand the procedures presented earlier in this manual for designing and building applications.

This chapter covers the following topics:

CAUTION!
Risk of data loss
Do not develop “C” language code on a live system as it is very powerful, and can be potentially dangerous if poorly written.

Understanding the user function process

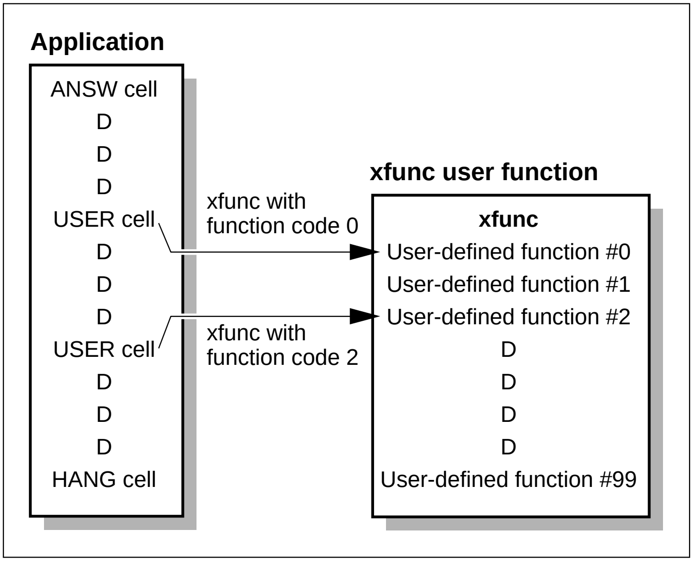
Before you code your own user functions, you should understand how an application interacts with a user function and how the process that is running (herein called “the user function process”) is shared with the other channels on the system.

Accessing the user function from the call flow

When you code a user function, you also define specific activities which the user function can perform. Each activity, or user-defined function, is identified by a two-digit function code. Your user function is referenced in the call flow of an application by using the USER cell. The USER cell references the user function, and identifies the activity the user function performs.

Figure 5-1 illustrates an application that passes control to the user function “xfunc” through the USER cell. In the first instance, xfunc executes user-defined function #0. In the next instance, xfunc executes user-defined function #2.

Figure 5-1
Application passing control to user function



The user function process

Only one user function process operates at a time regardless of how many running applications require it. If an application references a user-defined function located within the xfunc user function, the xfunc process starts automatically when you load the application.

If an application must execute a user-defined function from a user function process that is already running, that application uses the existing user function process rather than starting up a new one. This process is shared with all the available channels and does not perform any tasks until it receives a request in the message queue.

When you load and execute an application that references a user function, the following sequence of events occur:

1 Your application starts.

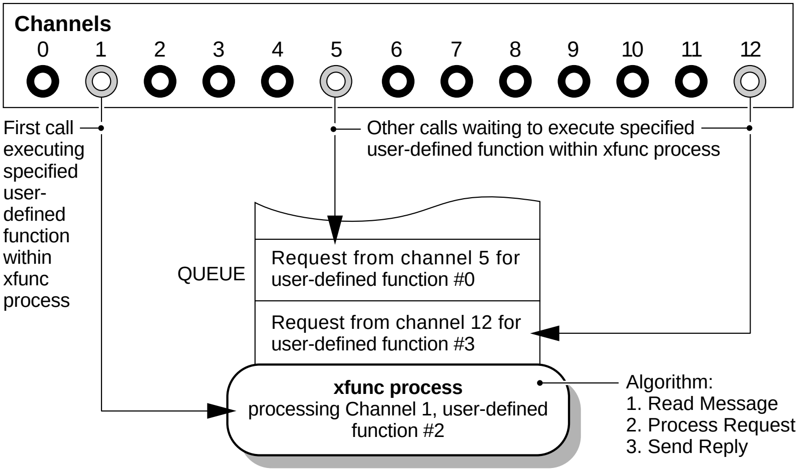
2 When the application executes a USER cell, it passes the corresponding user function process a message that includes a set of *input buffers* and a *function code* to indicate which user-defined functions should be executed.

3 The USER cell then transfers control of the call flow to the user function itself, and the application on the associated channel is suspended.

4 The user function process begins executing the specified user-defined function that is identified by a function code.

5 Meridian IVR continues processing calls on the other channels (see Figure 5-2). When the user function process has processed your task, it passes the application a set of *output buffers* and a *reply code*, then returns control to the application.

Figure 5-2
How the user function process works


Channels
0
1
2
3
4
5
6
7
8
9
10
11
12
First call executing specified user-defined function within xfunc process
Other calls waiting to execute specified user-defined function within xfunc process
QUEUE
Request from channel 5 for user-defined function #0
Request from channel 12 for user-defined function #3
xfunc process
processing Channel 1, user-defined function #2
Algorithm:
1. Read Message
2. Process Request
3. Send Reply

● The calls on these channels do not require execution of a user-defined function. Calls continue to be processed on these channels.

◎ The calls on these channels require execution of a user-defined function located within the xfunc process to be completed. They will block until the xfunc process sends a reply to their request.

Note: Application requests to the same user function process from different channels will not be read until the current request has been completed. A lengthy task could delay processing of other requests in the message queue. Consequently, design your user-defined functions to execute quickly to prevent delays.

6 The application branches to the next cell in your call flow application. The next cell is determined by the value of the reply code.

7 When you unload all applications that reference the same user function, the user function process is killed.

User-defined functions

Specific user functions can be referenced by applications running on many channels; however, only one channel can access the corresponding user function process at any given time. This means that the user function must finish whatever it is doing quickly each time it is called to avoid delays.

Input and output buffers

Input and output buffers provide a flexible way to transmit data between the application and the user function. An application can send the user function up to ten input buffers which can be any of the Meridian IVR system buffers or user-defined buffers. Similarly, the user function can return up to 10 output buffers to the application. As with all buffers in Meridian IVR, each of the input or output buffers can store a null-terminated string of up to 31 ASCII characters.

Reply code

When a user function finishes its current task, it returns a reply code which can be a value from zero to nine. When the application resumes, the USER cell checks the reply code to determine which cell the application should execute next. The USER cell has up to ten branches, and it uses the reply codes, found in the usr.c template file, to determine which of these branches to take.

If you want to force an error code, you need to define the codes in the user function by setting the reply code variable before returning to your application. Your application can check this reply code value and take a branch based upon this value. This way, the user cell passes the reply code set into the call flow and allows your application to choose which next cell to execute.

Source file for creating user functions

A source file, called “usr.c”, is included in the Meridian IVR software. You can use this file as a template to create user functions. This template contains the code you need to initialize and operate a user function process. Copy this file and insert your own code at a few specified points to create a user function. For a complete listing of the usr.c file, read the section “Using the usr.c file as a template” on page 5-13.

While your application is executing, your user function process reads the message sent from the USER cell, determines which user-defined function has been selected, executes the user-defined function, updates output buffers as necessary, and sends a message back to the USER cell with a reply code.

Error handling

Whenever a user-defined function encounters an error, it writes an error message to the event log. A user-defined function can send error and informational messages to the transaction log through calls to “post_log_msg()”, which takes a message string as its only argument. You can run the Transaction Log Report to view the error messages.

Understanding the importance of time-out

Calls can sometimes become stuck within a user function. This can prevent the call from progressing and result in hung calls and frustrated callers. However, you can code all Meridian IVR user functions with a time-out parameter to eliminate this inconvenience.

The time-out input parameter allows the application to take a time-out branch. In Meridian IVR Release 1.2, the treatment of time-outs is left completely up to custom code which the application developers must write.

For Meridian IVR Release 2.0/I, the user function template for the customized user functions is enhanced to include a time-out parameter. If you specify an input time-out value, the application is delayed in the user function for the time-out value you specify. The application then takes the time-out branch.

For any complex user functions, especially those which interface to external hosts, we strongly recommend that the application developers alter time-out treatment to meet their specific needs.

ATTENTION!

Be sure that you develop user functions properly.

User functions that crash or hang a channel cause the application to not take the time-out branch (or any other branch) from the user function cell.

Procedure 5-1
Designing a user function

1 Decide what you want the user function to do. Break its activities into separate functions that can be written as individual user-defined functions.

2 Choose a name for the user function.

The name can be up to five characters long. Do not use any of the following names since they conflict with existing components in Meridian IVR:

cli
dbs
dcm
ust
vrtd
csc
sam
qds
sde
lh
sad
log
vtk
vid
veh
psm
vbm
trs
access
msg
mm
vrm
xai
tmr
larx
csnap
sae
scm
smi
snap
bpe
xae
sri
uel
sai
vtf

3 Decide what buffer information should be passed to each user-defined function.

4 Decide what buffer information should be returned to the application.

A maximum of ten input buffers and ten output buffers can be used each time a USER cell invokes the user function.

5 Decide which reply code each user-defined function returns under various circumstances.

The user function returns a reply code which the USER cell utilizes to go to one of ten possible branches.

6 Decide how each user-defined function can determine that an error has occurred, and how it can return an error reply code.

7 Plan error messages and informational messages to cover all the possibilities.

8 Decide how each user-defined function will make calls to post_log_msg() to pass error strings.

> **ATTENTION!**
>
> We highly recommend that your user functions do not call printf(). Instead, use the error log to avoid interfering with the normal operation of the Meridian IVR system administration.

Once you have completed these steps, you are ready to code a user function.

As you read these instructions, refer to the usr.c source code in the section “Using the usr.c file as a template” on page 5-13.

Procedure 5-2
Coding a user function

1 Change the working directory to */user/ivr/gen/usr.*

2 Copy and rename the usr.c file.

The new name identifies the name of your user function. It can be up to five characters long. For example, to create a user function called “xfunc”, copy usr.c to xfunc.c. Copying this file, instead of editing it directly, ensures that you have an unaltered source file to use when you want to create another user function.

Edit the switch statement in your *filename*.c file as follows:

3 Specify a “case” statement for each user-defined function that the user function can execute.

4 For each case statement, assign a value to the reply_code variable, matching one of the codes in the Reply Code table for your application. This value can also correspond to an error condition; refer to the default case statement in the usr.c template file.

5 If you want to return information to the application from the user-defined function, assign a value (1–10) to the *p_number_of_output_buffers variable corresponding to the number of output buffers to be passed back to the application.

6 Assign values to the output buffers in the output_buffer_array structure. These values are passed back to the application during execution.

7 Save your edits.

Procedure 5-3
Building a user function

1 In the */u/ivr/usr* directory, type the following command:

make -e -f user.make NAME=usr_func <Enter>

For example, to build an executable file for xfunc.c, enter

make -e -f user.make NAME=xfunc <Enter>

Note: Since you have entered the name of your file without the .c extension, the resultant executable user function is xfunc, created in the current directory.

Testing your user function

Before you actually use your user function in an application, you should test the user function to ensure that it executes properly (for example, each case statement matches a specific user-defined function, or the appropriate number of buffers are passed). You can do this by using the user function stand-alone mode.

Procedure 5-4
Testing user function with stand-alone mode

1 Enter the following command at the shell prompt, where *user_func_name* is the name of the user function that you just created:

user_func_name -i <Enter>

2 Respond to the following prompts:

Function code The number (0–99) identifying the user-defined function and the corresponding switch-case statement in the user function you want to test. To exit the user function, type **-1.**

Current channel Enter any number from 0–63.

Number of input buffers The number of data buffers (0–10) to be passed to the user function.

Input buffers Values to be entered into each of the input buffers being used.

The user function executes, using your responses to these prompts, and displays the following information:

User function reply A code returned to the calling application. It identifies the status of the execution.

Number of output buffers The number of data buffers (0–10) that are returned to the calling application.

Output buffers The contents of the output buffers.

The following example illustrates the execution of the usr.c template in stand-alone mode. Type the following prompt at the shell prompt to start the “usr” user function:

usr -i

```
Enter Function Code (-1 to exit): 1
Enter Current Channel: 0
Enter Number Of Input Buffers: 0
==== about to call user function
==== returned from user function
User Function Reply: 1
Number Of Output Buffers: 0

Enter Function Code (-1 to exit): 2
Enter Current Channel: 0
Enter Number Of Input Buffers: 5
         Input Buffer #1: 1
         Input Buffer #2: 2
         Input Buffer #3: 3
         Input Buffer #4: 4
         Input Buffer #5: 5
==== about to call user function
==== returned from user function
User Function Reply: 2
Number Of Output Buffers: 5
         Output Buffer #1:'2'
         Output Buffer #2:'4'
         Output Buffer #3:'6'
         Output Buffer #4:'8'
         Output Buffer #5:'10'
```

```
Enter Function Code (-1 to exit):    3
Enter Current Channel: 0
Enter Number Of Input Buffers: 2
          Input Buffer #1: 1
          Input Buffer #2: 2
==== about to call user function
==== returned from user function
User Function Reply: 3
Number Of Output Buffers: 1
          Output Buffer #1:'3'

Enter Function Code (-1 to exit): 4
Enter Current Channel: 0
Enter Number Of Input Buffers: 0
==== about to call user function
==== returned from user function
User Function Reply: 4
Number Of Output Buffers: 0

Enter Function Code (-1 to exit): 5
Enter Current Channel: 0
Enter Number Of Input Buffers: 0
==== about to call user function
>>>> call to post_log_msg(’Unknown function code: 5')
==== returned from user function
User Function Reply: 0
Number Of Output Buffers: 0

Enter Function Code (-1 to exit): -1
```

To promote user function to the “exe directory”, enter this command:

make -e -f user.make NAME = user_func_name promote <Enter>

Meridian IVR cannot execute your user function unless you have previously entered this command.

Including the user function in an application

Now that you have built a user function, you can design and build an application to call the user function. To do this, you should read the previous chapters and become familiar with the procedures for designing and building applications. For more information on each user cell in your application, read the section on user cells in Chapter 7, “Cell catalog”.

Procedure 5-5
Including a user function in your application

1 Enter the name of the user function in the User Function Name parameter.

2 Enter the function code in the Function Code parameter.

3 Enter the number of reply codes in the # of Reply Codes parameter.

4 Enter the number of input buffers in the # of Input Buffers parameter.

5 Enter the number of output buffers in the # of Output Buffers parameter.

6 List the possible reply codes and their corresponding next cells in the Reply Code table.

7 Go to the Input Buffer table and list the input buffers.

8 Go to the Output Buffer table and list the output buffers.

Once you have built the application, you can load and run it. See the *Meridian IVR System Administration Guide* (NTP 555-9001-300) for procedures, or consult your system administrator. When the application is running, you should run the Transaction Log Report occasionally to see if the user function has written any errors or informational messages. The *Meridian IVR System Administration Guide* also provides procedures for running this report.

Using the usr.c file as a template

Here is a listing of the Meridian IVR usr.c source file, which you can use as a template for creating your User Functions. The usr.c file already contains sample User Function code in the switch statement (i.e., cases 1, 2, 3, and 4, highlighted in bold). Simply remove this code and enter your own case statements. *Before you modify the code, be sure you copy usr.c to another file and work on the copy*.

```
static char usr_c[]=”@(#) $Id: usr.c,vpf_main 1.4 1994/10/11
17:25:09 kwu Exp $”;
/************************************************************
*******************
 * usr.c
 *
 * This file contains the routine user_function().
 *
 *
 * WARNING:
*
 * Any include files added to this file will NOT be
 * reflected in the stated dependencies of usr.makefile.
 ***************************************************************************
*/
#include <stdio.h>

#include “vtk_std.h”

#include “vpf.h”
#include “usr_prototypes.h”

#define MAX_BUFS_PASSED


/************************************************************
*******************

 * extern global variables
 */
extern  FILE    *fp_verbose;            /* file pointer for
verbose mode    */
extern  int     timer_expired;          /* time out ? set if
TRUE*/

/************************************************************
*******************
*  For multi-threaded user functions, the number of child
processes
 *  should be specified here.
 *
 *  Note that zero children is the default, and means that the
user process
 *  will operate in a single threaded mode as it has always
done in the past.
```

```
*************************************************************
******************
 */
int     number_of_children  = 0;    /* number of children to
spawn, if any    */

/************************************************************
*******************
 *-h-   int     init_user_function(max_chans)
 *      int     max_chans;        - maximum possible channels
 *
 *  DESCRIPTION
*      User defined user_function initialization
 *
 *     This initialization routine is for the user to perform
certain
 *      tasks before the process begins taking requests from
applications.
*
 *      Note that in a multi-threaded user function. This
routine is executed
 *     in each child process, but not in the parent (manager)
process
 *
*  RETURNS:    TRUE    if successful

*              FALSE   otherwise

*************************************************************
******************
 */
int     init_user_function(max_chans)
int     max_chans;
{
    verbose_msg (“init_user_function”);

return(TRUE);

}
/************************************************************
*******************
 *-h-   void    end_user_function(max_chans)
 *      int     max_chans;        - maximum possible channels
 *
 *  DESCRIPTION
 *      User defined user_function cleanup
 *
 *      This termination routine is for the user to perform
certain
 *      tasks after the process has been notified to terminate.
 *
 *      Note that in a multi-threaded user function. This
routine is executed
 *     in each child process, but not in the parent (manager)
process
 *

*************************************************************
*****************
 */
void    end_user_function(max_chans)
int     max_chans;
{
    verbose_msg (“end_user_function”);

  return;

}
/*———+—————+—————+—————+—————+—————+—————+—————*/
/*-h-   int     user_function (function_code, current_channel,
**                              number_of_input_buffers,
input_buffer_array,
**                              p_number_of_output_buffers,
output_buffer_array)
**      int     function_code;
**      int     current_channel;
**      int     number_of_input_buffers;
**      char    input_buffer_array[MAX_BUFS][MAX_BUF_SIZE+1];
**      int     *p_number_of_output_buffers;
**      char    output_buffer_array[MAX_BUFS][MAX_BUF_SIZE+1];
**      u_long  tout;
```

```
**
**  DESCRIPTION:
**      Perform the actions of each user function.
**      Each CASE in the SWITCH statement acts as a different
user function
**      within this process
**
**  RETURNS:
**      The reply code (0 - 9) to the user function cell.
**      The cell will then branch based on this value.
*************************************************************
*******************
*/
int     user_function ( function_code, current_channel,
                        number_of_input_buffers,
input_buffer_array,
                      p_number_of_output_buffers,
output_buffer_array,

                         tout)
int     function_code;
int     current_channel;
int     number_of_input_buffers;
char    input_buffer_array[MAX_BUFS][MAX_BUF_SIZE+1];
int     *p_number_of_output_buffers;
char    output_buffer_array[MAX_BUFS][MAX_BUF_SIZE+1];
u_long  tout;
{
    /* ———— Variables for use by ALL user functions. —————————
*/
    int     reply_code = 0;
    char    error_message[80];
    /* ———— Variables used by THIS user function. —————————— */
   int     count;
  long    sum;

    /*
    * Assign a default value for the number of output buffers.
     * (Because p_number_of_output_buffers is a pointer to an
integer the
     * indirection operator (‘*’) must be used to assign an
integer value.)
    */
   *p_number_of_output_buffers = number_of_input_buffers;
   switch (function_code)
    {
         case 1:
            /* Do nothing */
            *p_number_of_output_buffers = 0;
            reply_code = 1;
            break;

       case 2:
            /* Numerically double buffer values */
            for (count=0; count<number_of_input_buffers;
count++)
            {
                sprintf(output_buffer_array[count], “%ld”,
                2*atol(input_buffer_array[count]));
            }
            *p_number_of_output_buffers =
number_of_input_buffers;
             reply_code = 2;
            break;

      case 3:
            /* Numerically sum buffer values */
            sum = 0L;
            for (count=0; count<number_of_input_buffers;
count++)
            {
                sum = sum + atol(input_buffer_array[count]);
            }
            sprintf(output_buffer_array[0], “%ld”, sum);
            *p_number_of_output_buffers = 1;
             reply_code = 3;
            break;

         case 4:
```

```
/* Any instruction (except those listed below) will     be
interrupted when
    a timeout occurs.  The  timer_expired variable will be set
to TRUE.  The
   User function programmer needs to check timer_expired after
every
   instruction that can  block the process as the example below
shows. */

/* Note: The UNIX system() call or any child of this user
function will not
     be able to receive the timeout signal*/

start_timer(tout);      /* tout is timeout duration */
sleep(180);             /* sleep for 3 minutes */

/* A suspected timeout may occur.Check global variable
timer_expired and call timeout_function if timeout   occurs */
      if (timer_expired)
     {
       timeout_function(function_code,
current_channel,
                                   number_of_input_buffers,
                                   input_buffer_array,
                                   p_number_of_output_buffers,
                                      output_buffer_array);

    p_number_of_output_buffers,
      output_buffer_array);  return(-1);
                            }
                         *p_number_of_output_buffers = 0;
                       stop_timer();
          reply_code = 4;
       break;

    default:
           /* ———— Print Error if in verbose mode —————————— */
            if (fp_verbose)
                fprintf(fp_verbose, “\tfunction_code %d not
implemented\n”,
                        function_code);
            /* ———— Report Error to error logger ——————————— */
            sprintf(error_message, “Unknown function code: %d”,
function_code);
            post_log_msg(error_message);
            *p_number_of_output_buffers = 0;
            break;
    }

    return (reply_code);
}

/************************************************************
*******************
 *-h-   int     init_user_manager(max_chans)
 *      int     max_chans;        - maximum possible channels
 *
 *  DESCRIPTION
 *      User defined user_manager initialization
 *
 *     This initialization routine is for multi-threaded user
functions
 *      where the user needs to perform certain tasks within
the manager
 *      process before it begins taking requests from
applications or
 *      communicating with its children user functions.
 *
 *      Note that this routine only executes if
number_of_children > 0
 *
 *  RETURNS:    TRUE    if successful
 *              FALSE   otherwise

*************************************************************
******************
 */
int     init_user_manager(max_chans)
int     max_chans;
{
    verbose_msg (“init_user_manager”);

    return(TRUE);
}
```

```
/*************************************************************
******************
 *-h-   void    end_user_manager(max_chans)
 *      int     max_chans;        - maximum possible channels
 *
 *  DESCRIPTION
 *      User defined user_manager cleanup
 *
 *      This termination routine is for multi-threaded user
functions
 *      where the user to perform certain tasks within the
manager
 *      process after it has been notified to terminate.
 *
 *      Note that this routine only executes if
number_of_children > 0

*************************************************************
*****************
 */
void    end_user_manager(max_chans)
int     max_chans;
{
    verbose_msg (“end_user_manager”);

    return;
}

/*************************************************************
*****************
  * -h- void timeout_function(function_code, current_channel,
  *                      number_of_input_buffers,
input_buffer_array,
  *                      p_number_of_output_buffers,
output_buffer_array)
  * DESCRIPTION
  *     Function to execute when timeout occurs.
  *
  * RETURN: nothing

*************************************************************
****************
  */
void timeout_function(function_code, current_channel,

                        number_of_input_buffers,
input_buffer_array,
                      p_number_of_output_buffers,
output_buffer_array)
int     function_code;
int     current_channel;
int     number_of_input_buffers;
char    input_buffer_array[MAX_BUFS][MAX_BUF_SIZE+1];
int     *p_number_of_output_buffers;
char    output_buffer_array[MAX_BUFS][MAX_BUF_SIZE+1];
{
 printf("Timeout function executed!\n");
}
```

Advanced user function techniques

If you are familiar with writing user functions, you may find the techniques described in the remainder of this chapter useful.

Task Manager

If you find the performance of your user function unacceptable, you can distribute its processing load. The Task Manager feature distributes incoming user function requests to one of several concurrent processes (children), instead of acting on each request sequentially.

This greatly reduces the potential bottleneck associated with a user function that interacts with a slow resource, such as a device or large database. Note that child processes can not exchange data with each other, only with IVR Generator 2.0/S.

The maximum number of children that you can use to defer load is ten. While the system does not limit the number of applications using a user function, only ten channels can use a specific user function simultaneously. Each of the ten child processes handles a single channel request in this situation. All others will wait in a queue.

Note: Degradation can occur when the user function uses more than five concurrent processes.

To designate the number of user_function children, edit the variable number_of_children at the top of the usr.c file. The default is 0, which creates a single-threaded environment (that is, one that spawns no children and processes all requests itself).

You can also include two optional functions in your “C” code:

- init_user_manager() performs initialization one time at the creation of the user function task manager (when the first application that uses the user function is loaded).
- end_user_manager() performs shutdown tasks in the user function task manager one time (when the last application that uses the user function is unloaded).

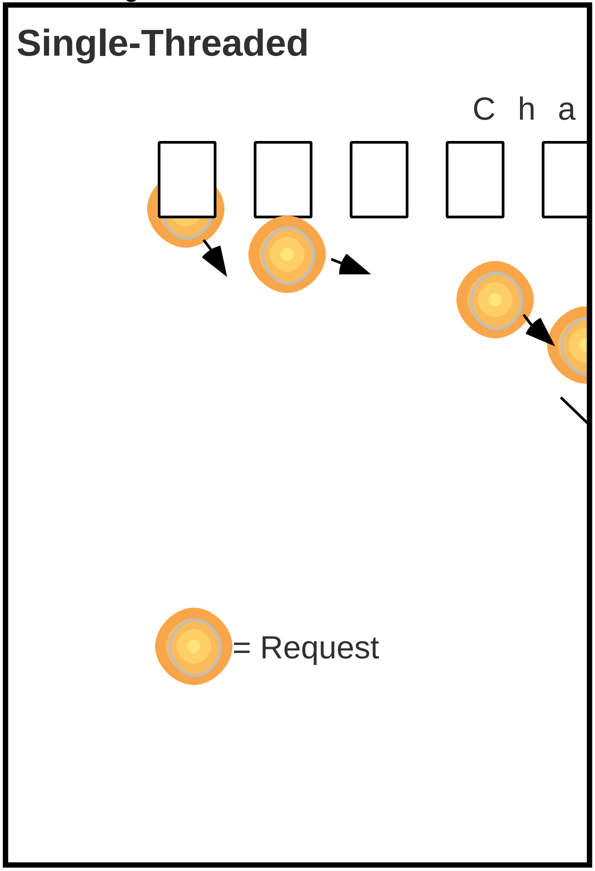
You can use the following two functions in the single-threaded environment, but when used in conjunction with a task manager with concurrent processes, they take on the following specialized purpose:

- init_user_function() performs initialization one time at the creation of each user function child process, when the first application is loaded.
- end_user_function() performs shutdown tasks in the user function child process one time, when the last application is unloaded.

The dispatching algorithm used by the task manager guarantees that, while a child concurrent process can serve more than one channel, it will consistently serve the same set of channels.

Since a child may need to maintain information about calls associated with each channel, it only needs to dynamically allocate memory for those channels with which it is associated. Therefore, when using multiple child processes (five, for example), each child is responsible only for its part of the memory allocation (in this case, 20 percent)—see Figure 5-3.

Figure 5-3
Task Manager



Time-out

You can set a parameter in your user function to follow a time-out branch if its execution time exceeds the time-out value specified on the parameter page of the USER cell.

To implement this functionality, you must issue a start_timer (duration) function in your user_function at the beginning of the code you want to bound with a time limit. If the code you are limiting exceeds the duration specified, the system will execute a handler function which will set the global variable timer_expired to TRUE. The instruction following the code that could exceed the time limit should contain a test for this variable. Be sure to include stop_timer in the user_function to shut off the timer that was previously requested.

If the timer_expired variable is TRUE, you can call the function timeout_function (which contains code you have specified) to cleanup from a time-out condition before returning a reply to the application.

For example, to open a connection to an external device and cancel it if it exceeds a specified time limit, include the following lines in your user_function:

```
start_timer(tout); /* tout is timeout duration */
open_device(....);  /* this represents any operation you want
to limit */
if (timer_expired)
/* Check global variable. TRUE means a timeout occurred. */
{
       /* to clean up */

timeout_function(function_code,
current_channel,
number_of_input_buffers,
input_buffer_array,
p_number_of_output_buffers,
output_buffer_array);
      return   usr_timeout_code();
}
stop_timer();  */ cancel the timer if connection is made within
the time allowed*/
```

Note: The timeout signal will not interrupt Unix System() calls or any child processes of the user function.

Use the timeout_function() to set up return values to the application. The timeout reply message is sent elsewhere.

Chapter 6: Using information databases

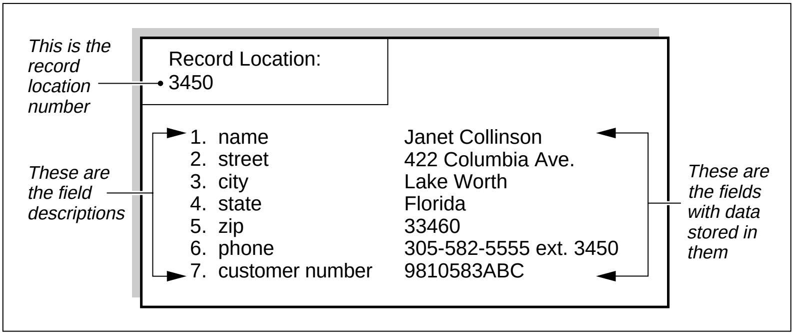
Meridian IVR has a built-in database facility that can be very useful in building applications. This database can store data in the form of character strings that you can use as part of your application.

You can create information databases with the Meridian IVR System Database Editor, one of the five interfaces of Meridian IVR. This chapter provides procedures for creating a new information database and for editing an existing information database:

Understanding information databases

Each information database consists of a set of records. Figure 6-1 illustrates a single record. A record consists of up to ten fields, each of which can store up to 31 characters including letters, numbers, spaces, or any of the other characters that you can type on a keyboard, except for commas.

Figure 6-1
An example of a record in a database



This is the record location number

Record Location:
3450

These are the field descriptions

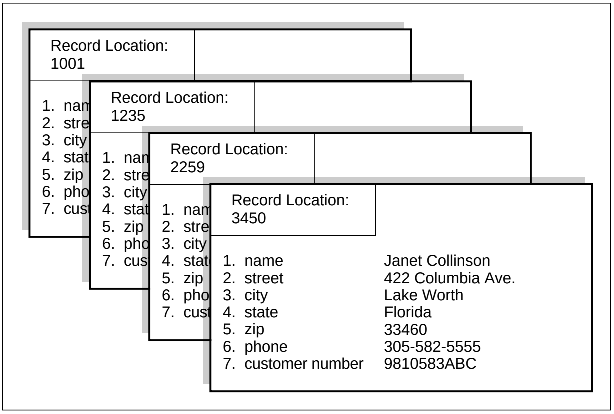
1. name	Janet Collinson
2. street	422 Columbia Ave.
3. city	Lake Worth
4. state	Florida
5. zip	33460
6. phone	305-582-5555 ext. 3450
7. customer number	9810583ABC

These are the fields with data stored in them

Each field can have a field description to keep track of the types of data that the information database stores. Every record in a particular information database has the same set of field descriptions.

Because an information database can have many records, each record has an identification number called a record location number (see Figure 6-2.) The information database is designed to associate the information in a record with this number. Although the information database can use any number as the record location number, it generally uses a telephone number.

Figure 6-2
A series of records in an information database



Using an information database

You can use the Check Database (CHEK) cell to manipulate database records. When you create a CHEK cell, you identify the information database from where you extract data, and the buffers where you place the data. You can create applications that use the CHEK cell to extract data and make it available to other cells in the application.

In addition, you can use the CHEK cell to determine whether a record exists for the record location number in a specific information database. If the record does not exist, the CHEK cell branches to the “not found” next cell. If the CHEK cell finds the record you specify, it automatically copies information from the record to the buffers within the application. Once this information becomes available to the application, it can be used by the application.

The CHEK cell type has three parameters:

Database name The name of the database that CHEK verifies.

Record location number The record location number of the specific record in the database.

Number of output buffers The number of buffers that the CHEK cell needs to copy data from the record you specify. For example, if the record has three fields, the CHEK cell needs three output buffers to retrieve and copy the information.

Parameters

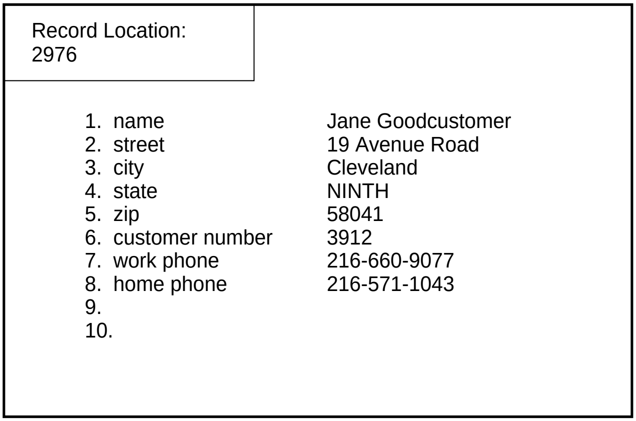
Parameter	Assigned value
Database name	customers
Record location number	customer number
Number of output buffers in buffer table	7

The CHEK cell lists the output buffers in the Output Buffer Table:

BUFFERS	
NAME	
ADDRESS	
CITY	
STATE	
ZIP CODE	
DATA1	
DATA2	

You have an information database called “customers”, and record location number 2976 stores the data shown in Figure 6-3.

Figure 6-3
Sample database record



Record Location:
2976

1.	name	Jane Goodcustomer
2.	street	19 Avenue Road
3.	city	Cleveland
4.	state	NINTH
5.	zip	58041
6.	customer number	3912
7.	work phone	216-660-9077
8.	home phone	216-571-1043
9.		
10.		

When you run this application, the CHEK cell verifies that the customer's database exists, looks for record number 2976, and copies data from the record into its output buffers. Copying begins with the first field and ends when all of the output buffers have been used. When the CHEK cell finishes copying the data, the output buffers contain the following data:

Buffer Contents

Buffer	Contents
NAME	Jane Goodcustomer
ADDRESS	19 Avenue Road
CITY	Cleveland
STATE	Ohio
ZIP CODE	58041
DATA1	3912
DATA2	216-660-9077

Notice that the data from the “home phone” field is not copied from the record to the buffers. Since you have specified only seven output buffers, the CHEK cell copies data from only the first seven fields. The copying process does not alter the information database itself.

Once the CHEK cell has copied the data into the buffers, the application can use that data in any of the ways that applications use buffer data. For example, there can be a Play Prompts with Data (PDAT) cell that reads the phone number stored in the DATA2 buffer to the caller.

Building information databases

Before you begin working with the Database Editor, you should first build an information database.

> ***Note:*** You can create more than one information database depending on how much disk space you have. However, Meridian IVR applications can use only up to 20 databases at the same time.

The following are guidelines you need to consider when creating a database:

Decide what information you want to store. As already explained, each record has ten fields. Each field can store up to 31 characters. Characters can be letters, numbers, or characters that you can type on a keyboard, except for commas.

Although you do not need field descriptions for your records, we recommend that you use them because they remind you about the types of information stored in the database.

Each record has the same set of field descriptions.

If you want to have an application, use the stored information and plan the application. You can build the application either before or after you create the database.

Choose the number of significant digits which is the maximum number of digits in the record location number. (If you would like to review record location numbers, return to Figure 6-2.)

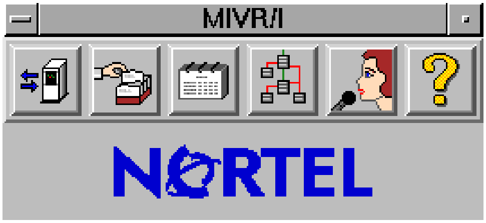
The name of the database can be up to eight characters long and can include any characters except for spaces, asterisks, pound signs, and question marks.

System Database Editor

Procedure 6-1
Opening the Database Editor

1 On the Meridian IVR main menu, click on the database icon (second from the left) with the left mouse button (see Figure 6-4).

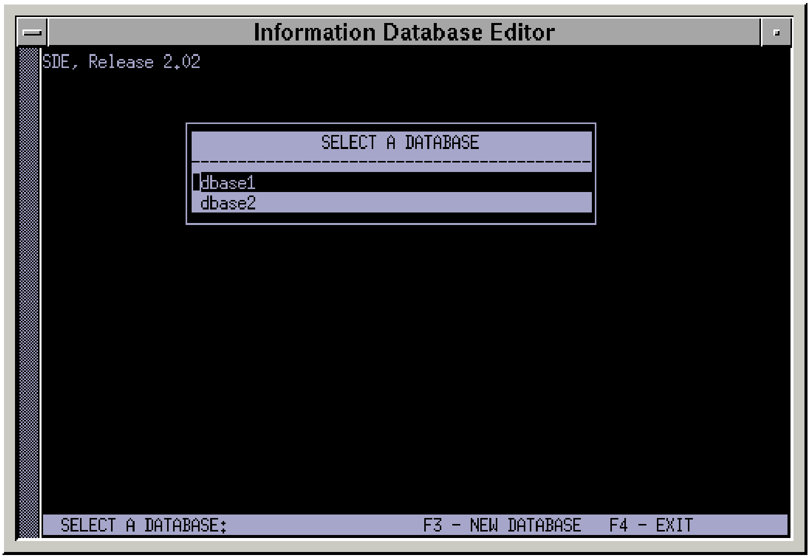
Figure 6-4
Meridian IVR GUI main menu



2 Select Information Database Editor from the pull-down menu.

Meridian IVR displays the Database Editor main menu as illustrated in Figure 6-5.

Figure 6-5
Database Editor



```
SDE, Release 2.02

SELECT A DATABASE
-------------------------------------------
dbase1
dbase2

SELECT A DATABASE:                F3 - NEW DATABASE   F4 - EXIT
```

In this and all other Database Editor windows, the arrow keys work as follows:

<Up Arrow> Moves the cursor up one line.

<Down Arrow> Moves the cursor down one line.

<Left Arrow> Moves the cursor to the top of the page.

<Right Arrow> Moves the cursor to the bottom of the page.

When you press <Enter>, you select the item at the cursor.

Procedure 6-2
Creating the database

1 From the Database Editor main menu, press <F3> for New Database.

You see the following prompt:

```
Enter The Name Of The Database:
```

2 Type the database name and press <Enter>.

Note: A message or information database name can be up to eight characters long and can include any characters except for spaces.

You see the following prompt:

```
Enter The # Of Digits (between 4 and 11, inclusive): 4
```

3 Type any value from 4 to 11 and press <Enter> to enter the new value, or press <Enter> to accept the default value 4.

In this procedure, you select the *number of significant digits* for this database. For an information database, the number of significant digits is *the maximum number of digits in the record location number*.

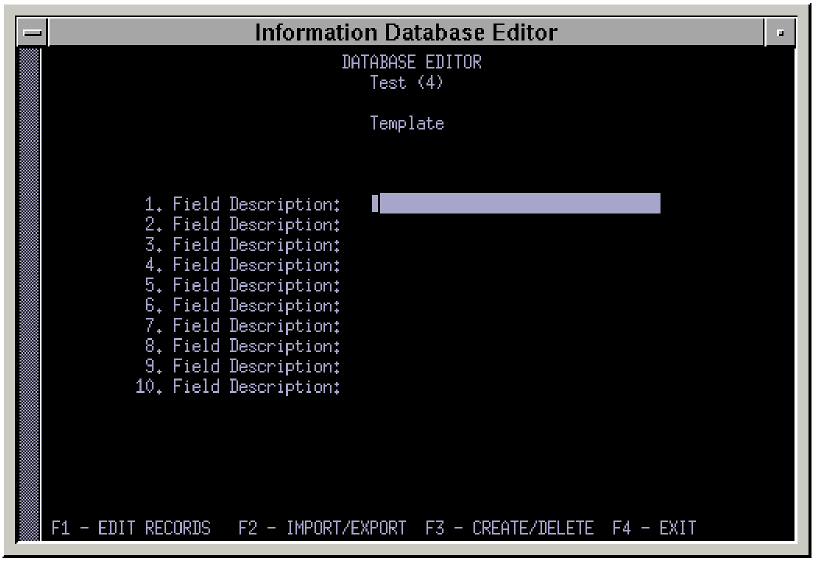
Usually, record locations are associated with DID digits, and DID channels are usually configured to receive four digits, so the number of significant digits is very often four.

If you need information on channel types, see the *Meridian IVR System Administration Guide* (NTP 555-9001-300), or consult your system administrator.

Note: Be careful when choosing the number of significant digits because once the number has been chosen, it cannot be changed.

The Database Editor displays the Template screen illustrated in Figure 6-6.

Figure 6-6
Template screen



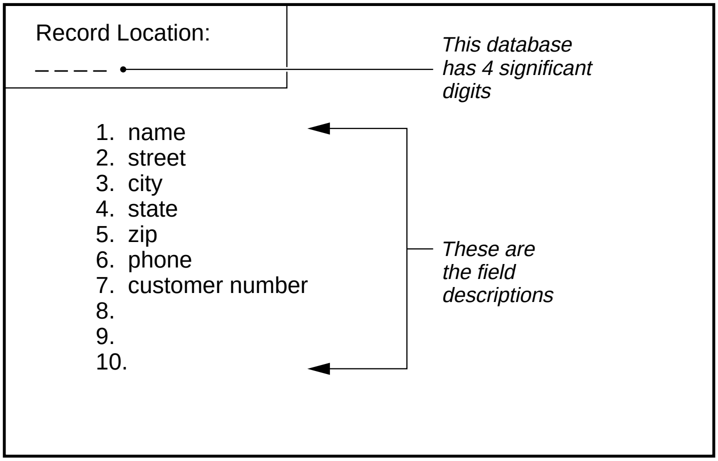
```
                                DATABASE EDITOR
                                   Test (4)

                                   Template


          1. Field Description:
          2. Field Description:
          3. Field Description:
          4. Field Description:
          5. Field Description:
          6. Field Description:
          7. Field Description:
          8. Field Description:
          9. Field Description:
         10. Field Description:

F1 - EDIT RECORDS   F2 - IMPORT/EXPORT  F3 - CREATE/DELETE  F4 - EXIT
```

You can use this screen to go to an already existing record or create a template as explained in the next section. To go to an existing record, press <F1>.

Creating the template

After you name the database and choose the number of significant digits, you can create a template for the database as illustrated in Figure 6-7. The template is the set of field descriptions for the records in the database.

Figure 6-7
Example of a template for a database



Record Location:

_ _ _ _

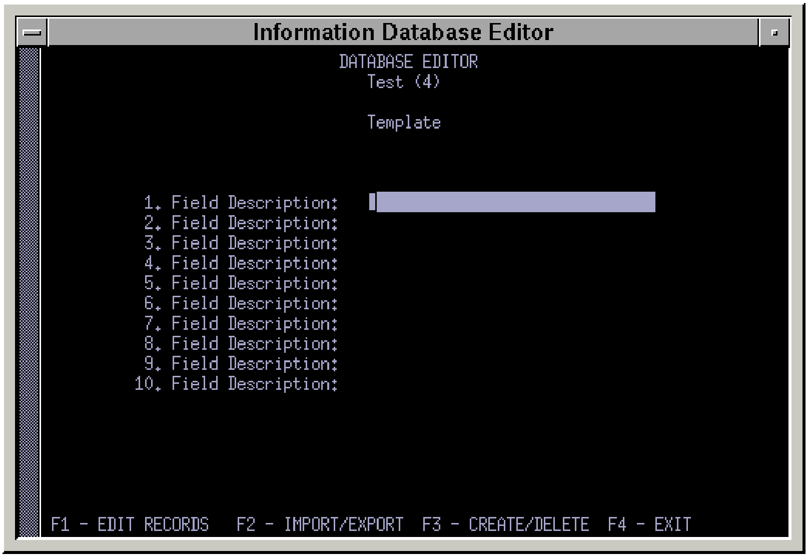
1. name
2. street
3. city
4. state
5. zip
6. phone
7. customer number
8.
9.
10.

This database has 4 significant digits

These are the field descriptions

The name of the database appears on the template screen. The number of significant digits appears in parentheses next to the database name. In the template window shown in Figure 6-8, the name of the database is “TEST” and the number of significant digits is 4.

Figure 6-8
Template for database "TEST"



Information Database Editor

```
                    DATABASE EDITOR
                      Test (4)

                      Template


      1. Field Description:
      2. Field Description:
      3. Field Description:
      4. Field Description:
      5. Field Description:
      6. Field Description:
      7. Field Description:
      8. Field Description:
      9. Field Description:
     10. Field Description:



F1 - EDIT RECORDS   F2 - IMPORT/EXPORT   F3 - CREATE/DELETE   F4 - EXIT
```

Procedure 6-3
Creating the template

1 Move the cursor to where you want a field description, then type the description.

Field descriptions can have up to 31 characters and include any characters you type on the keyboard, except for commas.

If you begin or end a field description with blanks, those blanks are removed.

If you make a mistake, press the <Backspace> key to delete the characters.

2 After you type a field description, press <Enter>, <Tab>, or the <Down> arrow key.

The cursor advances to the next field.

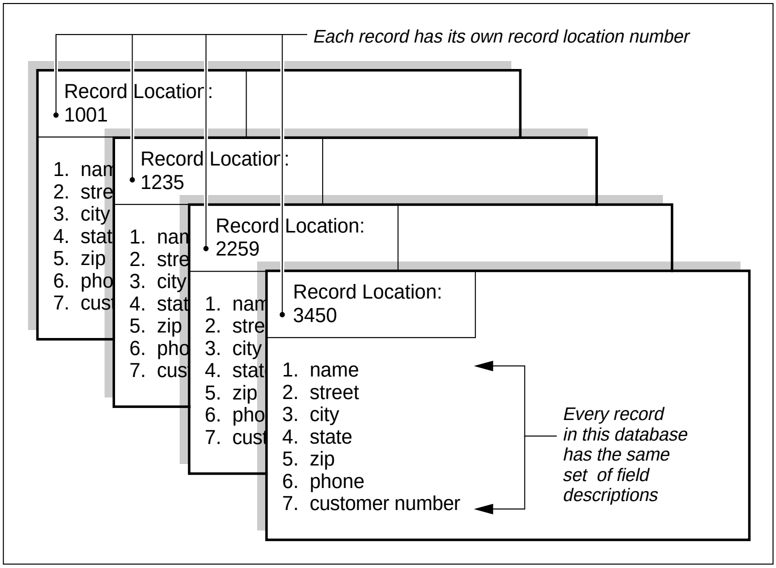
3 Repeat steps 1 and 2 until you have entered all of your field descriptions.

Now you are ready to create record locations.

Creating record locations

After you have created the template, you can create record locations. For every record location you create, you have one record as illustrated by Figure 6-9. When you create the database, you select a value for the number of significant digits that determines the length of the record location number. For example, if the number of significant digits is 4, then the record location numbers can be any numbers from 0 to 9999.

Figure 6-9
Record location numbers

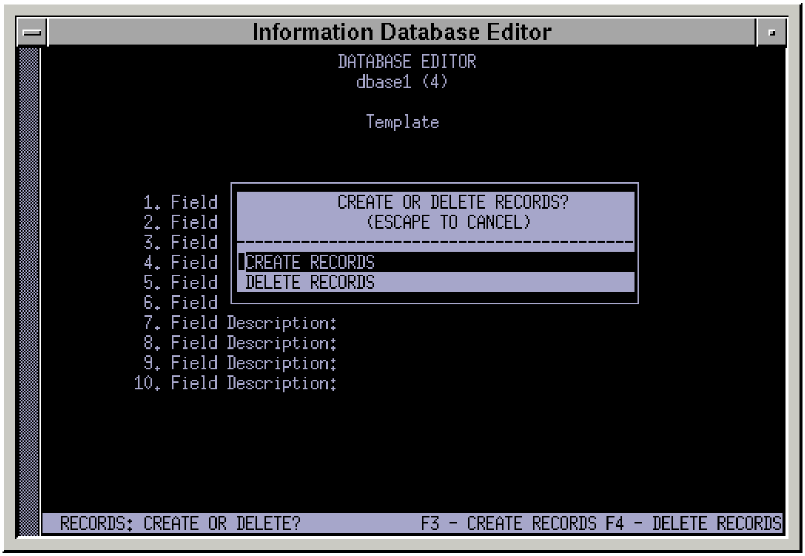


Procedure 6-4
Creating record locations

1 Press <F3>.

The Database Editor displays a pop-up window that asks you to indicate whether you want to create or delete a record (See Figure 6-10).

Figure 6-10
Create/delete record pop-up menu



```
                 DATABASE EDITOR
                   dbase1 (4)

                    Template


  1. Field    CREATE OR DELETE RECORDS?
  2. Field       (ESCAPE TO CANCEL)
  3. Field  ------------------------------
  4. Field   CREATE RECORDS
  5. Field   DELETE RECORDS
  6. Field
  7. Field Description:
  8. Field Description:
  9. Field Description:
 10. Field Description:

RECORDS: CREATE OR DELETE?      F3 - CREATE RECORDS F4 - DELETE RECORDS
```

2 Select “Create a Record”.

You see the following prompt:

```
Enter Record To Create:
```

3 Type a number for a single record location, or type a range of numbers.

To enter a range, type the beginning number, a hyphen, and an ending number.

For example, to enter the numbers from 1 to 2000, type the following without spaces:

1-2000

4 After you have typed the number or numbers, press <Enter>.

The Database Editor creates the record location(s) and the message disappears.

5 Repeat steps 1–3 to create as many record locations as you need.

Deleting record locations

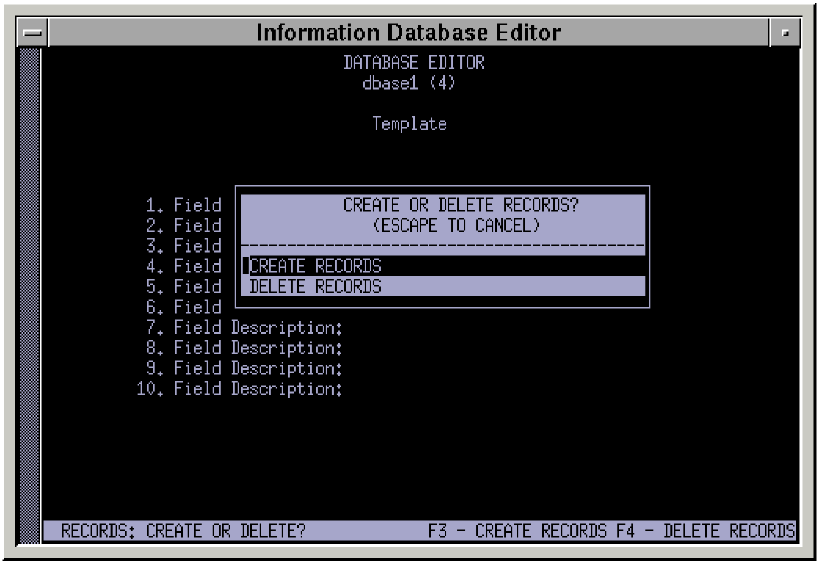
When you delete the record location, the system erases the data stored at that location.

Procedure 6-5
Deleting record locations

1 From the Template screen, press <F3> for CREATE/DELETE.

A pop-up menu appears as shown in Figure 6-11.

Figure 6-11
Create/delete record prompt menu



```
DATABASE EDITOR
dbase1 (4)

Template

 1. Field    CREATE OR DELETE RECORDS?
 2. Field       (ESCAPE TO CANCEL)
 3. Field    ----------------------------------------
 4. Field    CREATE RECORDS
 5. Field    DELETE RECORDS
 6. Field
 7. Field Description:
 8. Field Description:
 9. Field Description:
10. Field Description:

RECORDS: CREATE OR DELETE?          F3 - CREATE RECORDS F4 - DELETE RECORDS
```

2 Select DELETE RECORDS.

The system displays the following prompt:

```
Enter Record to Delete:
```

3 Type the number of the record location you want to delete, or a range of numbers for more than one record, then press <Enter>.

The system deletes the record location(s).

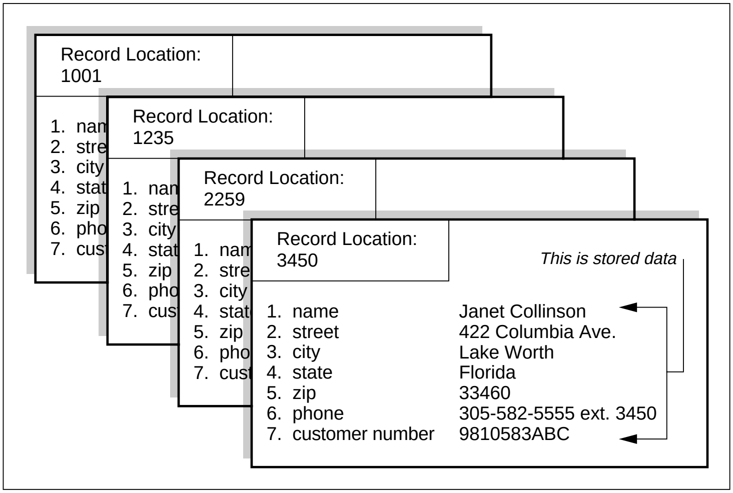
If the system cannot delete the record(s), the system will give you an error message.

If the system successfully deletes the record(s), the template screen for the database re-appears (see Figure 6-8).

Adding or updating data

Once you have created record locations, you can add data in the locations. You can update the data at any time, even when other applications are using the database. Figure 6-12 illustrates a series of records with stored data.

Figure 6-12
Records with stored data



CAUTION!
Risk of lost data

If you try to update one field in a record from a blank template, the system updates that one field but the other fields are blanked out. This causes you to lose the entire record.

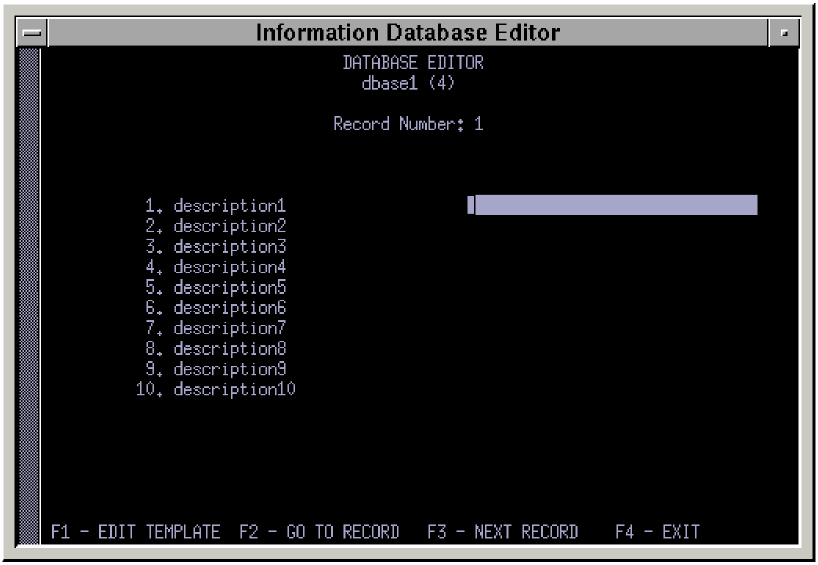
To avoid this, load the record in the database, modify the field, then save the record.

Procedure 6-6
Adding or updating data

1 From the Template screen, press <F1> for Edit Record.

You see the Record Location screen that corresponds to the lowest existing record location number (see Figure 6-13).

Figure 6-13
Adding and changing data in the record location



```
                    DATABASE EDITOR
                      dbase1 (4)

                   Record Number: 1


         1. description1
         2. description2
         3. description3
         4. description4
         5. description5
         6. description6
         7. description7
         8. description8
         9. description9
        10. description10

F1 - EDIT TEMPLATE  F2 - GO TO RECORD   F3 - NEXT RECORD    F4 - EXIT
```

2 If necessary, press <F2> for GO TO RECORD to display the screen corresponding to another record. (This procedure is discussed in the next section.)

3 Use the arrow keys to move the cursor to any field, then type in the information you want to store.

If there was already information in that field, it disappears when you begin to type.

If you make a mistake, press <Backspace> to delete characters.

4 When you have finished typing the information, press <Enter>.

Note: You cannot export a database with more significant digits to a database with less significant digits.

Changing record locations

You can change record locations at any time but only when the Record Location screen appears. Figure 6-13 shows the Record Location screen with the current record location listed on the third line.

You can change the record location in two ways:

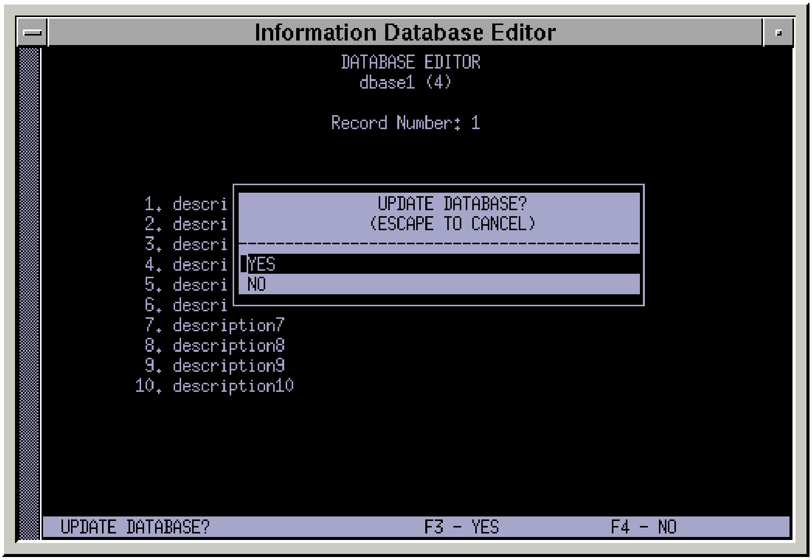
- going to the next consecutive location
- going to a specific location

Procedure 6-7
Going to the next record location

1 From the Record Location screen, press <F3> for Next Record.

If you change the data at the current record location, the Update Database pop-up menu appears, as shown in Figure 6-14.

Figure 6-14
Update Database pop-up menu



```
Information Database Editor
                 DATABASE EDITOR
                   dbase1 (4)

                Record Number: 1

 1. descri        UPDATE DATABASE?
 2. descri       (ESCAPE TO CANCEL)
 3. descri  ------------------------------------
 4. descri  YES
 5. descri  NO
 6. descri
 7. description7
 8. description8
 9. description9
10. description10

UPDATE DATABASE?              F3 - YES          F4 - NO
```

2 If you want to save your current changes, move the cursor to YES.

If you do not want to save your changes, move the cursor to NO.

3 Press <Enter>.

You go to the next consecutive record location.

If you were on the last location, you go to the first location.

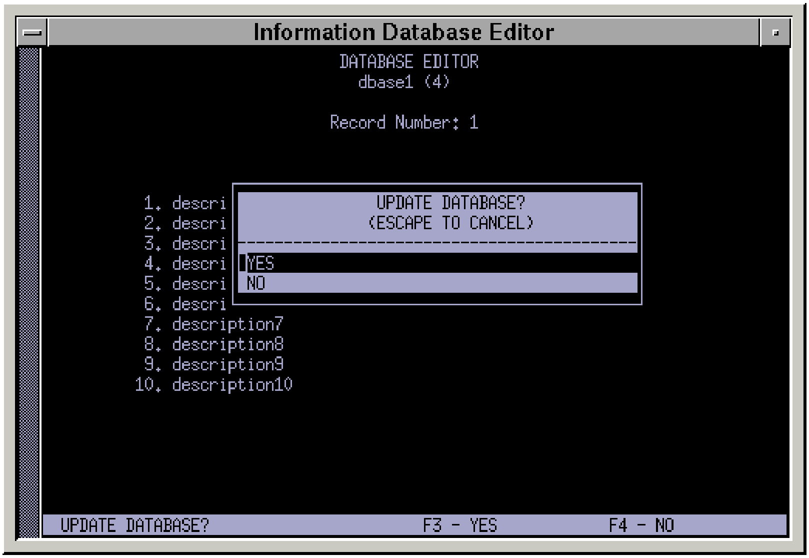
The record location number appears at the top of the screen.

Procedure 6-8
Going to any other record number

1 From the Record Location screen, press <F2> for Go To Record.

If you change the data at the current record location, the Update Database pop-up menu appears as shown in Figure 6-15.

Figure 6-15
Update Database pop-up menu (continued)



Information Database Editor

```
DATABASE EDITOR
dbase1 (4)

Record Number: 1

 1. descri   UPDATE DATABASE?
 2. descri   (ESCAPE TO CANCEL)
 3. descri  -----------------------------------------
 4. descri  YES
 5. descri  NO
 6. descri
 7. description7
 8. description8
 9. description9
10. description10

UPDATE DATABASE?                  F3 - YES          F4 - NO
```

2 If you want to save your current changes, move the cursor to YES.

If you do not want to save your changes, move the cursor to NO.

3 Press <Enter>.

You see the following prompt:

```
Enter Record:
```

4 Type a number for a single record location, or type a range of numbers and press <Enter>.

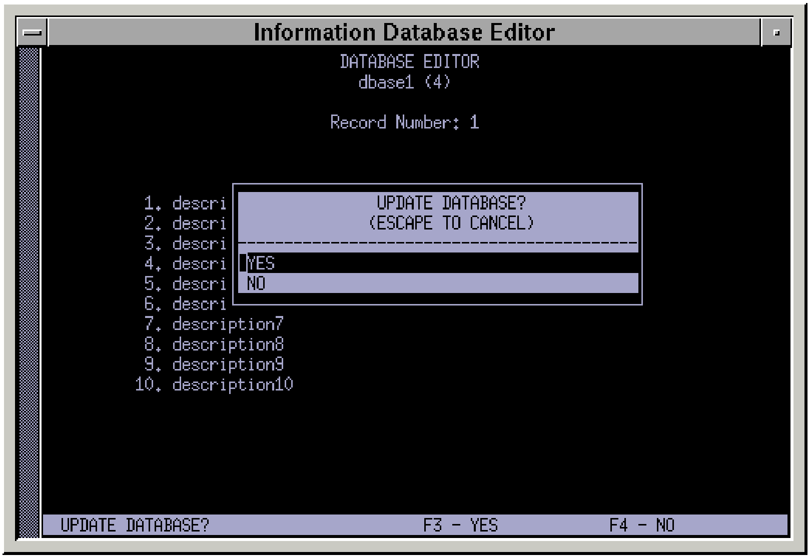
The record location(s) appears at the top of the screen.

Procedure 6-9
Exiting the database

1 From the Record Location screen, press <F4>.

If you change the data at the current record location, the Update Database pop-up menu appears again as shown in Figure 6-16.

Figure 6-16
Update Database pop-up menu (end)



```
Information Database Editor
                         DATABASE EDITOR
                          dbase1 (4)

                        Record Number: 1


          1. descri      UPDATE DATABASE?
          2. descri     (ESCAPE TO CANCEL)
          3. descri -------------------------------
          4. descri  YES
          5. descri  NO
          6. descri
          7. description7
          8. description8
          9. description9
         10. description10

UPDATE DATABASE?                F3 - YES          F4 - NO
```

2 If you want to save your most recent changes to the database, move the cursor to YES and press <Enter>.

The system saves your changes then takes you back to the Database Editor main menu.

If you want to exit without saving the most recent changes, move the cursor to NO and press <Enter>.

The system takes you back to the Database Editor main menu.

Opening a database

You can open a database and edit it at any time, even when it is being used by an application.

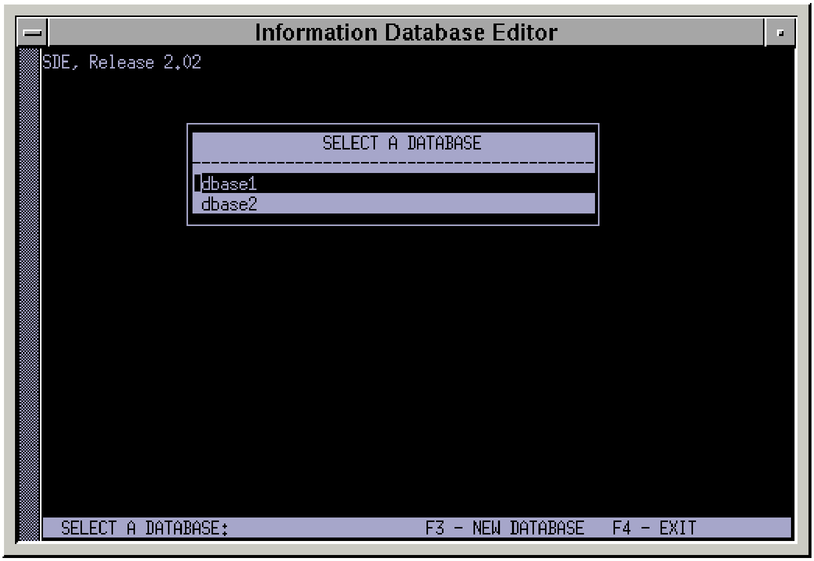
> ***Note:*** You cannot load a database, you can only select an existing database or create a new one.

Procedure 6-10
Opening a database

1 From the Database Editor main menu, as shown in Figure 6-17, move the cursor with the arrow keys to the name of the database you want to open, then press <Enter>.

The system opens the database and displays the Edit Record screen for the lowest-numbered record location.

Figure 6-17
Selecting a database from the Database Editor



Information Database Editor

SDE, Release 2.02

```
SELECT A DATABASE
-----------------------------------------
dbase1
dbase2
```

```
SELECT A DATABASE:                    F3 - NEW DATABASE   F4 - EXIT
```

Editing the template

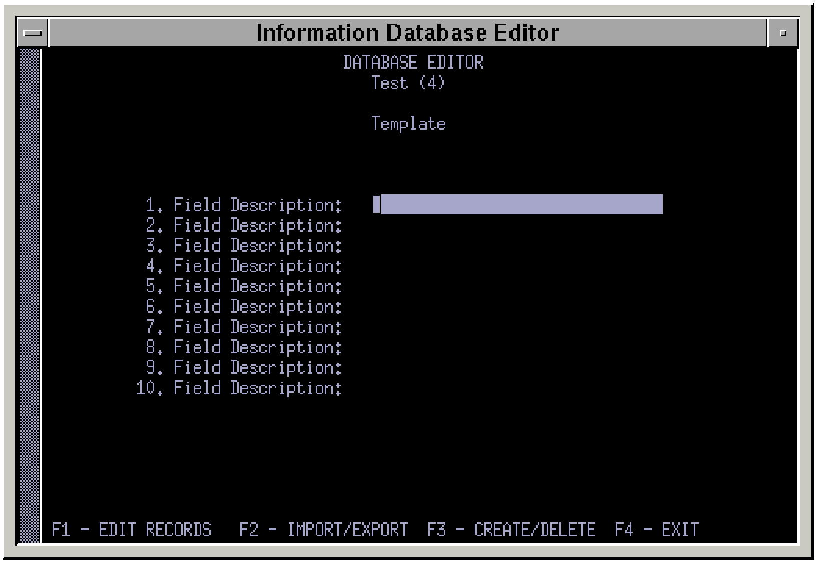
You can edit the template by changing one or more field descriptions.

Procedure 6-11
Changing a field description

1 From the Record Location screen, press <F1> for Edit Record.

The Template screen appears as shown in Figure 6-18.

Figure 6-18
Editing the template



Information Database Editor

```
                      DATABASE EDITOR
                         Test (4)

                         Template


         1. Field Description:
         2. Field Description:
         3. Field Description:
         4. Field Description:
         5. Field Description:
         6. Field Description:
         7. Field Description:
         8. Field Description:
         9. Field Description:
        10. Field Description:



F1 - EDIT RECORDS   F2 - IMPORT/EXPORT  F3 - CREATE/DELETE  F4 - EXIT
```

2 Move the cursor with the arrow keys to the field you want to change, then type in a new description.

The field description changes.

Deleting a database

You must be in the UNIX shell to delete a database.

Procedure 6-12
Deleting a database from the UNIX shell

1 Change to the directory */u/ivr/sys_files.*

2 Enter the following command: **rm dbname**, and press <Enter>.

CAUTION!
Risk of losing system files

If you have two databases named *db* and *db1.ext,* you can type **rm db*.*** to delete both files at the same time.

The first asterisk represents any character(s) that follow the file name. The second asterisk represents the extension.

We recommend that you do not perform this procedure. However, if it is necessary, do not type a space between the database name and *.* otherwise, not only will you erase the database named db; you will also erase system files that have extensions.

Chapter 7: Cell catalog

This chapter catalogs the Meridian IVR cells and describes how you can use them in your application. In addition, this chapter lists the buffers used and updated, the cell parameters, next cells, and the tables. The following lists the abbreviated and full names of the cells as they appear in this chapter.

- "ADDR - address message" on page 7-4
- "ANSW - answer" on page 7-6
- "CASE - branch on value or range" on page 7-8
- "CDAT - convert data" on page 7-11
- "CEND - continuation outdial end" on page 7-14
- "CFAX - call back fax" (See the *Fax Application Guide* [NTP 555-9001-350]).
- "CHEK - check database" on page 7-15
- "COMA - host communications abort" on page 7-17
- "COMI - host communications, input to host" on page 7-19
- "COMO - host communications, output from host" on page 7-21
- "COMP - compare" on page 7-23
- "CONC - concatenate buffers" on page 7-28
- "COUT - continuation outdial" on page 7-30
- "DCAT - data file create/concatenate" (See the *Fax Application Guide*.)
- "DDEL - delete information database records" on page 7-33
- "DELV - schedule a delivery application" on page 7-34

- “DIAL - dial digits” on page 7-41
- “DINS - insert records into an information database” on page 7-42
- “DMSG - delete message” on page 7-43
- “DOUT - directed outdial” on page 7-45
- “EVENT - log application event” on page 7-47
- “EXEC - execute application” on page 7-50
- “FDEL - data file delete” (See the *Fax Application Guide*.)
- “FRCV - fax modem receive” (See the *Fax Application Guide*.)
- “FRLS - fax modem release” (See the *Fax Application Guide*.)
- “FRSV - fax modem reserve” (See the *Fax Application Guide*.)
- “FSND - fax modem send” (See the *Fax Application Guide*.)
- “GDAT - play prompts and get data” on page 7-52
- “GSUB - call a subapplication” on page 7-57
- “HANG - hang up” on page 7-59
- “INFO - get call information” on page 7-61
- “LANG - set language” on page 7-63
- “LDLV - get schedule event list” on page 7-68
- “MATH - perform mathematical operation” on page 7-71
- “MDTE - get Meridian Mail date and time” on page 7-73
- “MENU - play menu and get data” on page 7-75
- “PDAT - play prompts with data” on page 7-78
- “PLAY - play prompts” on page 7-83
- “PMSG - play a message” on page 7-85
- “PRGS - call progress detection” on page 7-86
- “QCNT - SQL select count cell” on page 7-88
- “QDEL - SQL delete cell” on page 7-90
- “QINS - SQL insert cell” on page 7-93

Cell descriptions

The following items are explained in the cell descriptions. However, keep in mind that not every cell has parameters, next cells, and tables.

Name and abbreviation The name of the cell and its abbreviation.

Description What the cell does and how to use it.

Buffers used System buffers that supply information to the cell.

Buffers updated System buffers that the cell type updates with new data. Buffers pass information from one cell to another. You do not always have to tell the application which buffers you want to use because an application handles most buffers automatically, but you do need to consider what is happening to the buffers when you create an application. This is why each cell description tells you about the buffers that the application uses or updates.

> ***Note:*** Buffer names are not case sensitive.

Parameters A list of all the parameters that you work with when you create a cell. For each parameter, the name, initial value, and a brief explanation of what it does are provided. Notice that the list of parameter names and initial values are shown exactly as you see them on the computer screen when you are creating or editing an application.

Each cell has a comment box parameter where you can add notes that explain or clarify what you are doing. Each comment box can hold up to 79 alphanumeric characters.

Next cells The branches that determine which cell comes after the current cell. Most cell types have more than one branch to the next cell. For each branch, an explanation of why the application would take that branch is given. The list of next cells is shown exactly as you see it on the screen.

Tables A list of tables that you fill with information for the cell to use. These same tables also appear on the screen. Each table has its own page on the computer screen.

ADDR - address message

Description

The ADDR cell is used to enable a message to be addressed by mailbox numbers. The address’s full name is returned. The ADDR cell should be called after an RMSG has been successfully completed and before an SMSG cell call has been issued.

Cell type

Messages.

Buffers used

Buffer	Explanation
Current message	The number of the message to be addressed

Buffers updated

None.

Parameters

Parameter	Initial Value	Explanation
Call Audit Enabled?	No	Determines if this cell will log the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment Field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Destination mailbox	Mailbox ID	First mailbox
Additional mailbox	None	Up to 9 optional mailboxes

Next cell

Next cell	Explanation
Error	The message was not addressed.
Success	The message was addressed successfully.

Tables

None.

ANSW - answer

Description

ANSW accepts the phone call on the current channel.

It is common to accept a call immediately at the start of the call, so the ANSW cell is often the second cell in an application (remember that we recommend always putting the HANG cell first). However, in some situations, you can postpone accepting the call until some sort of verification (for example, validating the calling or called number) has been performed. It is possible to build applications where a database look up or some other kind of verification is carried out before the ANSW cell is executed.

> ***Note:*** Meridian IVR automatically answers the call, but the call must be answered through an ANSW cell at the earliest possible time.

The application is not allowed to play prompts before the ANSW cell is executed. This feature safeguards against fraudulent use of the telephone network.

Buffers used

None.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell will log the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment Field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.

Next cell

Next cell	Explanation
Success	This is the only branch.

Tables

None.

CASE - branch on value or range

Description

CASE compares the contents of a buffer to a selection of predefined criteria and passes the control to the branch that corresponds with the criteria.

You can set the selection criteria to either match exactly one of a set of values or to fall within a range of values. In that case, the Match Criteria parameter is set to RANGE, and the values in the Valid Cases table must be ordered from lowest to highest.

CASE is useful in applications for

- altering functionality and prompts based on data value such as time, date, day, or other codes set by your application
- validating input and evaluating return codes from a GSUB
- creating a subroutine library which is a palette that contains multiple related subroutine sections where the desired functionality is specified by one of the parameters

Buffers used

None.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell will log the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Selection Criteria	DIGITS	A buffer that contains the selection criteria for the CASE cell.
Comparison Type	NUMERIC	The type of comparison the cell performs. The valid choices are: NUMERIC STRING

Match Criteria	EXACT	The criteria the application uses to determine which branch to take. The valid choices are: EXACT RANGE If you select the EXACT match option in the Match Criteria parameter, the application either finds the branch matching the SELECTION CRITERIA buffer contents or takes the NOT FOUND branch. If you select the RANGE match option, the application takes the appropriate case boundary branch for data either less than or equal to the SELECTION CRITERIA buffer contents. Otherwise, the applicator takes the NOT FOUND branch. ***Note:*** If you select the RANGE match option, list the cases in the Cases Table in ascending order.
Number of Cases	1	Number of valid cases in the CASE table.

Next cell

Next cell	Explanation
Not Found	The input selection could not be found in the Valid Case table.

Tables

Tables	Explanation
Valid Cases	A list of all valid cases to match the selection criteria, and the next cell corresponding to each case. Valid cases can be any numerical or string constant, or any buffer containing a value.

CDAT - convert data

Description

CDAT converts data from one format to another by allowing you to specify the input format, output format, source buffer, and the destination buffer. Under the Conversion Type header in the pop-up menu, the following options are available:

- DATE
- TIME
- DATE & TIME
- CURRENCY

CDAT is especially useful for converting data gathered from other sources into a format compatible with other cells. For example, if your database has a date field in the format mm/dd/yy, it must be converted to mmddyy for use by the PDAT cell.

CDAT also provides the facility for extracting a single piece of data for use such as the month from the complete date.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell will log the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Input format		The format in which the data is provided.
Output format		The desired format.
Source Buffer		The name of the buffer where the data is located.
Destination Buffer		The name of the buffer where the converted data should be placed.

Next cell

Next cell	Explanation
Inv Format	The format is invalid for the type.
Error	The value is out of range for the format specified.
Success	The data has been converted to the format specified.

If the source buffer contains the number 0102, the input format is mmdd, the output format is mmm-dd, the conversion type is Date, and the number is converted to jan-02.

If the input format or output format contains characters other than those specified under the following Valid Month Format section, CDAT takes the INVALID FORMAT branch. You should note the following valid input formats:

Valid month formats

m	1–12 output only, unless there are separators
mm	01–12 input, output
mmm	jan, feb, mar, apr, can, jun, jul, aug, sep, oct, nov, dec
mmmm	january, february, march, april, can, june, july, august, september, october, november, december
Mmm	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
MMM	JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC
Mmmm	January, February, March, April, May, June, July, August, September, October, November, December
MMMM	JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER

CEND - continuation outdial end

Description

CEND is used in conjunction with a Continuation Outdial (COUT) cell which initiates a secondary telephone call on the current channel. At some point in an application, after COUT initiates a secondary phone call, CEND ends the secondary call and returns to the primary call. It ends the secondary call in one of two ways as determined by the CEND type:

- CEND can simply hang up the secondary call and then return the application to the original caller. This is done if the CEND type is RECONNECT.
- CEND can end the secondary call by connecting the two parties together in a conference call. Note that the two callers remain connected to Meridian IVR. This is done if the CEND is CONFERENCE.

Buffers used

None.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell will log the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer

Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
CEND Type (Conf/Recon)	CONFERENCE	How should the secondary call be ended? The choices are CONFERENCE and RECONNECT.

Next cell

Next cell	Explanation
Error	The secondary phone call does not exist or could not be ended.
Success	The secondary call has been ended, and the application has returned to the original caller.

Tables

None.

CHEK - check database

Description

The CHEK cell checks the information database for the existence of a record associated with a particular record number. If the record is found and output buffers are supplied, then the output buffers are filled with data that is stored in the record. (Refer to Chapter 6, “Using information databases,” for further information about information databases.)

Buffers used

The CHEK cell uses the buffer specified by the Record Number parameter.

Buffers updated

The CHEK cell updates the buffers listed in the Output Buffer Table.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell will log the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Database Name	<NONE>	The name of the information database to be checked. Initially, the database table is empty. It only contains the names of the databases used in the current application. The complete list of available databases is not displayed.
Record Number	DIGITS	The buffer that contains the record location number to be checked in an information database. The buffer can be a system buffer or a user-defined buffer.
Buffer Count (max 10)	0	A count of the number of buffers to hold output information from an information database. This value is updated automatically with the count of buffers once the cell is saved.

Next cell

Next cell	Explanation
Not Found	There was no record in the information database for the specified record number.

Tables

Table	Explanation
Output Buffer Table	The names of the buffers that hold output from the information database.

COMA - host communications abort

Description

The COMA cell aborts a transaction in progress from a host computer that is connected to Meridian IVR through the Application Processor (AP).

The host computer runs a terminal-based application through the host connectivity products. The Meridian IVR AP replaces the human operator and provides information to the running host application.

The COMA cell places the host application back to a known starting point, and frees up all the memory and buffer tables associated with application data retrieval.

Buffers used

None.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell will log the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment Field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.

Next cell

Next cell	Explanation
Success	The transaction completed successfully.
Error	An error has occurred, or there is no transaction in progress.

Tables

None.

COMI - host communications, input to host

Description

The COMI cell passes data to a host computer connected to the Meridian IVR Application Processor (AP) and initiates a transaction through the host connectivity products.

The host computer runs a terminal-based application through the Meridian IVR AP. The AP replaces the human operator and provides information for the running host application.

The COMI cell handles all the issues related to sending data to the host computer. It also provides the interface for data from the IVR application.

The COMI cell uses a script that determines the flow of the host application's screens. The script consists of screen and action templates. You must create this script separately with a text editor. The COMI cell also uses a set of buffers to provide input to the screens.You may need to use the COMI cell more than once to provide all the data necessary to complete one transaction. The "More input" flag is used to postpone the execution of the transaction until all input data has been gathered.

> ***Note:*** Refer to Chapter 4, "Creating a sample application", for information on how to write a script.

Upon receipt of all input buffers, the execution of a COMI cell consists of a host transaction. A transaction is a script that, if followed from beginning to end, defines actions that must be performed. Several screens can be traversed to obtain the desired information. The last step in the transaction is to bring the host program back to a known starting point.

If the script does not specify reset and logout actions, and the mode is set to manual, the communications session is left on the last screen of the virtual terminal session. The next use of a COMI cell in this application starts its script as if it were resuming from that point in the last transaction. This feature allows the IVR application to make decisions affecting the host transaction flow and eliminates the need to start each transaction at the login screen.

Buffers used

None.

Buffers updated

None.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Action Template	None	Name of the action template.
More Input?	No	Will more input buffers be required before the transaction should begin?
Time-out	75	Time-out for host response: value 0–75 seconds.
# Input Buffers	0	Value 0–10

Next cell

Next cell	Explanation
Success	Input data received and/or the transaction began successfully.
Error	An error has occurred; possible cause: unexpected screen found or field not found.

Tables

Table	Explanation
Input Buffer Table	Lists the name of each input buffer.

COMO - host communications, output from host

Description

The COMO cell retrieves data from a host computer connected to the Meridian IVR AP through host connectivity products.

The host computer runs a terminal-based application through the host connectivity products. The Meridian IVR AP replaces a human operator, provides information to the running host program, takes information contained on the screens, and stores this information for future retrieval through the COMO cell.

The COMO cell provides the interface to the IVR application for the host data retrieved by the script. The script is initiated by the COMI cell when it provides a set of buffers to hold output data retrieved from the screens.

If the blocking node is set to Y, then the COMO cell waits for the transaction initiated by the COMI cell to complete. It then returns up to 10 buffers of information to the IVR application, along with a status code indicating the success of the transaction. If the blocking mode is set to N and the transaction is not compete, the COMO cell gets a status code indicating “Not Ready” and executes the Next cell “Not Ready” branch without retrieving any data. If the transaction has completed, up to 10 buffers plus a status code indicating the success of the transaction are then returned to the application. The MIVR AP retains all the information from the last transaction. In future occurrences, the COMO cell will retrieve the next logical sequential data.

The COMO cell retrieves information from a transaction previously initiated through a COMI cell and only waits for completion, if necessary. You can use the asynchronous capabilities provided by this pair of cells to your advantage because they allow interaction with a caller through other cells between these two cells while the transaction is in progress.

Buffers used

None.

Buffers updated

None.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Blocking (y/n)	Yes	Indicates whether or not the COMO cell should wait for a transaction to complete before it transitions to the next cell.
Output Buffers	0	Value 0–10.

Next cell

Next cell	Explanation
Success (End of Data)	The transaction completed successfully.
Success (More Data)	The transaction completed successfully, but there is more data to be retrieved through another COMO cell.
Not Ready	The transaction has not completed yet. It is only valid if blocking is set to n.
Time-out	The time-out value has been exceeded.
Error	An error has occurred; possible cause: unexpected screen found or field not found.

Tables

Table	Explanation
Output Buffer Table	Lists the name of each output buffer.

COMP - compare

Description

The COMP cell compares the contents of a buffer against either the contents of another buffer or a constant value, and uses the results of the comparison to determine the next cell. Two types of comparison are possible: numerical (for example, comparing two numbers), and string-based.

By using the “Step Flag” parameter, the COMP cell allows you to increment or decrement (that is, increase or decrease by one) the contents of Buffer A after the comparison. This feature is only useful with numeric comparisons. For a string comparison, this feature is meaningless; consequently, the COMP cell ignores the Step Flag parameter for string comparisons.

Buffers used

COMP uses Buffer A and Buffer B.

Buffers updated

Buffer A is updated if the COMP cell uses the Step Flag parameter to decrement or increment its contents, and you set the Type of Comparison parameter to Numeric.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Buffer A	DIGITS	A buffer name which can be a system buffer or a user-defined buffer.
Buffer B	0	A buffer (either a system buffer or a user-defined buffer) or a constant value (maximum size is 29). It is compared with Buffer A. If you specify a constant value in a string comparison, enclose it in quotation marks.

Step Flag	NONE	In a numeric comparison, the flag identifies whether Buffer A should be increased by one, decreased by one, or left unchanged after the comparison. This is ignored in a string comparison.
Type of Comparison	Numeric	The type of comparison to be performed: numeric or string.

Next cell

Next cell	Explanation
A < B	A is less than B.
A > B	A is greater than B.
A = B	A is equal to B.

Tables

None.

Examples

In a string comparison, the result is based on alphanumeric order: 1–10, A–Z. Table 7-1 illustrates string comparisons the COMP cell performs.

Table 7-1
String compare examples

Contents of Buffer A	Contents of Buffer B	Next cell
"A"	"C"	A < B
"1111"	"21"	B > A
"1ZZ"	"ZZ"	A < B
"MAINE"	"MAINE"	A = B
"STATUS"	"ORDER"	A > B
"ABBOT"	"ABBO"	A > B

Suppose that you want to perform a numeric or string comparison of the contents of the DIGITS buffer against the contents of the NUMBER OF RECORDS buffer. Table 7-2 shows the results.

Table 7-2
Comparison of two buffers

Buffer A = Digits **Buffer B = Number of records buffer**	
Conditional	**Next cell**
IF DIGITS < NUMBER OF RECORDS BUFFER	A < B
IF DIGITS = NUMBER OF RECORDS BUFFER	A = B
IF DIGITS > NUMBER OF RECORDS BUFFER	A > B

You can also compare the contents of a buffer against a constant value as illustrated in Table 7-3.

Table 7-3
Comparison of a buffer to a constant

Buffer A = Digits **Buffer B = 1**	
Conditional	**Next cell**
If Digits < 1	A < B
If Digits = 1	A = B
If Digits > 1	A > B

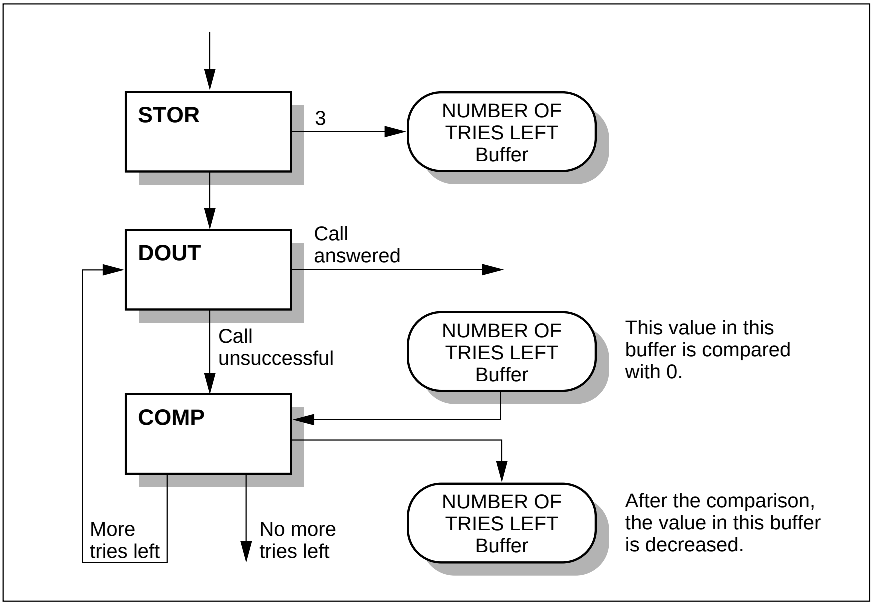
By using the Step Flag parameter in numeric comparisons, you can include a COMP cell in your application to perform a looping function. To use this feature, set the value of the Step Flag parameter to INCREMENT or DECREMENT.

Figure 7-1 illustrates a portion of an application that uses COMP for looping. The application has a DOUT cell that makes an outgoing phone call. It also has a buffer called NUMBER OF TRIES LEFT to indicate how many additional times the application should try to make the call if the first attempt fails. The NUMBER OF TRIES LEFT buffer starts out with a value of 3. The Step Flag parameter in the COMP cell is set to decrement the buffer.

If the DOUT cell is unsuccessful in making the call, the application branches to the COMP cell which compares the NUMBER OF TRIES LEFT buffer with the constant value of 0. If the buffer value is greater than 0, then DOUT tries to make the call again.The COMP cell decrements the value of the buffer, and the application returns to the DOUT cell. Because the application follows a set of branches forming a circular pathway, it is said to "loop" back to the DOUT cell.

The application keeps looping through the DOUT cell to try the call again until the call is answered or until there are no more tries left. If the call is not answered on the first try, the application can make up to three additional attempts because the original value of the NUMBER OF TRIES LEFT buffer is 3.

Figure 7-1
Using COMP for looping in an application


STOR
3
NUMBER OF TRIES LEFT Buffer
DOUT
Call answered
Call unsuccessful
NUMBER OF TRIES LEFT Buffer
This value in this buffer is compared with 0.
COMP
NUMBER OF TRIES LEFT Buffer
After the comparison, the value in this buffer is decreased.
More tries left
No more tries left

CONC - concatenate buffers

Description

The CONC cell concatenates (joins) two character strings. After concatenation, the destination buffer contains its original contents followed by the source string. The buffer identified by the Destination String Length parameter contains the number of characters in the destination buffer following concatenation.

Buffers used

The CONC cell uses the buffers specified by the Source, Destination Buffer, and Destination String Length parameters.

Buffers updated

The CONC cell updates the buffers specified by the Destination Buffer and Destination String Length parameters.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.

Source (Buffer/Value)	DIGITS	The source string or the buffer that contains the source string. If the source is a string rather than a buffer, you must enclosed it in quotation marks.
Destination Buffer	DIGITS	The buffer to which the source string is appended.
Destination String Length	LENGTH	The length of the string in the Destination Buffer after concatenation. If the error branch is taken, the Destination String Length buffer contains a “0”.

Next cell

Next cell	Explanation
Fail	The combined length of the two strings is too long for the Destination Buffer. Each buffer in Meridian IVR can store up to 31 characters.
Success	Concatenation has been successful.

Tables

None.

Examples

Suppose the source is a buffer containing the string 6678437, and the destination is a buffer containing the string 800. After the CONC cell has concatenated the two strings, the destination buffer contains the string 8006678437. The Length buffer contains the value 10. The contents of the source buffer are unchanged.

As another example, suppose the source is the string 2509393 and the destination is a buffer containing the string 508. After the CONC cell has concatenated the two strings, the destination buffer contains the string 5082509393.

COUT - continuation outdial

Description

The COUT cell puts a caller on hold and makes a secondary call by performing an add-on call to the phone number specified by the Phone Number parameter.

After checking for the dial tone, an attempt is made to dial the number. If the number cannot be dialed, the application branches to the Error next cell. If the dialing is successful, it allows a specified maximum number of rings to occur. If this maximum occurs before the called party answers, the call is considered not to have been answered, and the application branches to the Ring next cell. If a busy signal is detected, the application branches to the Busy next cell. Notice that if the application branches to the Error, Ring, or Busy cell, it takes the original caller off hold and continues normally.

If the secondary call is answered, then the application branches to the Answer next cell. The application continues; the original caller remains on hold, and the called party presses telephone keys or performs any of the other activities that the application can require.

> ***Note:*** CEND must be performed before any prompts or messages.

The application can end the secondary call in one of three ways:

- A CEND cell can hang up or disconnect the secondary call. In this case, the original caller is taken off hold.
- A CEND cell can join the two callers together in a conference call.
- The called party (that is, the person who received the secondary call) can hang up before the application ends the secondary call. In this case, the application takes the error branch of the COUT cell.

COUT is the only cell type in which a return cell can be specified. An error branch is quite different from a next cell. The application goes directly to the return cell any time the recipient of the secondary call hangs up on the call, no matter which cell is being executed at the moment. If you want to play anything to the caller from the error branch, insert a CEND cell with a reconnect option before the PLAY cell.

Buffers used

The COUT cell uses the buffer specified by the Phone Number parameter.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Ring Count (0–31)	0	How many rings to allow before branching to the Ring next cell. ***Note:*** If you set this parameter to 0, the application uses the default value of 31 rings.
Phone Number	DIGITS	The buffer containing the phone number to be dialed.

Next cell

Next cell	Explanation
Busy	The called number is busy.
Ring	The ring count has been exceeded.
Error	An error was encountered during the process of dialing the phone number.
No Energy Detect	The secondary call was answered, but no voice energy has been detected. This branch is not supported in this release.
Voice	The secondary call was answered and voice energy has been detected. The secondary call is in progress.

Return cell

Return cell	Explanation
Hang up	The secondary call was answered, and the application branched to the Success next cell of the COUT cell with the secondary call in progress. However, the called party hung up before the application could end the secondary call. The application stops whatever it is doing and goes directly to this Return Cell.

Tables

None.

DDEL - delete information database records

Description

The DDEL cell allows you to delete a record from an information database. The record you specify is deleted from the information database if it is found.

Buffers used

The DDEL cell uses the buffer specified by the record number parameter.

Buffers updated

None.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell will log the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Database Name	<NONE>	The name of the information database from which records have to be deleted.
Record Number	DIGITS	The buffer that contains the storage location number that has to be deleted from the information database. The buffer can be a system buffer or a user-defined buffer.

Next cell

Next cell	Explanation
Error	The record number could not be deleted.
Not Found	There was no record in the information database for the specified record number.
Success	The record was successfully deleted.

Tables

None.

DELV - schedule a delivery application

Description

One application uses the DELV cell to schedule another application to start automatically, immediately, or at some time in the future.

The DELV cell originates outgoing phone calls to provide services such as message delivery, pager access, telemarketing, wake-up calls, and reminder calls. The scheduled application is called an event.

Applications scheduled by the DELV cell run on outbound channels only. These applications typically have a DOUT cell early in the call flow to make an outgoing phone call. When the call has been answered, the scheduled application can play prompts to the called party, collect digits from the called party, or perform any of the other standard application activities.

The DELV cell simply makes a request for the application to start at some time in the future. If the request is accepted, the DELV cell takes the success branch to the next cell. Other than the buffers that are passed from the scheduling application to the scheduled application, there is no link between the two applications. The scheduled application begins at the appropriate time, often long after the scheduling application has finished.

The DELV cell contains several options and features to increase the flexibility of this very useful cell. Since a typical use of the DELV cell is to have a call placed at a specific time to play a message, there are parameters for specifying the outdial number, delivery time, and message number.

You can schedule an application more than once. There are parameters that help you reschedule the application. There are two basic reasons for rescheduling:

- Some applications, such as those that deliver wake-up calls, are intended to be run daily at a specific time. For this reason, you select an automatic rescheduling feature with the Time Type parameter.
- An outdial can be unsuccessful because the line is busy or the call is not answered. You can want it to try again. For this reason, the DOUT cell in the delivery application can have error branches leading to a DELV cell so that the application can reschedule itself. This procedure is called manual rescheduling. There are parameters called Delivery Interval (for the length of time until the next scheduled delivery) and Delivery Attempts that the delivered application can use in rescheduling itself. The application typically decrements the Delivery Attempts parameter each time you run the application. When the number of delivery attempts reaches zero, the application no longer reschedules itself.

Since Meridian IVR can have a long schedule of future events, it returns a unique identifier to the DELV cell whenever an application is scheduled, and stores it in the buffer specified by the Delivery Event ID parameter.

It is often useful to pass additional pieces of information to a scheduled application—for example, another message or a second phone number. For this purpose, the DELV cell passes the system buffers—DATA EXCHANGE numbers 1–5, to the scheduled application. What they contain and how their contents are to be interpreted are entirely up to you. The DELV cell itself does not store any information in these buffers, so the application must put information into them before the DELV cell is executed.

A more detailed discussion of the individual parameters of the DELV cell follows.

Application Name

Specifies either the *name* of the application (without the .vpf extension) to be scheduled or a buffer containing the name. You must load and start the scheduled application by the delivery time. See the *Meridian IVR System Administration Guide* (NTP 555-9001-300) for instructions on loading applications and for information on outbound channels. Since the application is automatically assigned an available outbound channel by Meridian IVR at a scheduled time, the system administrator does not need to assign any channels to the application. However, you must start the application.

Outdial Number

Specifies a buffer containing the phone number that the scheduled application calls. This buffer is passed to the scheduled application.

Time Type

Determines the manner in which the schedule is calculated.

- If the Time Type is ABSOLUTE, the application runs at a specific date and time. The DELV cell uses the Delivery Time parameter to calculate delivery time.
- If the Time Type is RELATIVE, the application runs a certain length of time from the moment the DELV cell is executed. The DELV cell uses the Delivery Interval parameter to calculate delivery time.
- If the Time Type is RESCHEDULE, the application automatically runs at intervals specified by the Reschedule Interval parameter, starting with the date and time specified by the Delivery Time parameter. The DELV cell uses the Delivery Time and Reschedule Interval parameters to calculate delivery time.

Delivery Time

Selects the buffer that specifies the time when the scheduled application should run. The format for the delivery time is as follows: *mmddyyyyhhmm* (2 digits for month, 2 digits for day, 4 digits for year, 2 digits for hour, and 2 digits for minutes—12 digits altogether in a 24-hour format.)

Delivery Interval

Specifies the length of time in minutes until the next scheduled delivery of the application. This is used by the DELV cell only when the Delivery Time parameter is set to RELATIVE. For the other Delivery Time types, the DELV cell itself does not use this parameter; however, if the Delivery Interval parameter specifies a buffer, other cells in the application can use that buffer.

Delivery Attempts

Is passed to the scheduled application and placed in the Delivery Attempts buffer. This parameter specifies information that the scheduled application can use to determine whether it should reschedule itself, in case the call is not successful. It is important to note that this field is not used by the DELV cell itself, nor does it automatically cause the application to be rescheduled. Typically, the scheduled application has cells that use the Delivery Interval and Delivery Attempts parameters to determine whether it should reschedule itself. For example, a scheduled application could use a COMP cell to determine whether it should take a branch to a DELV cell.

Delivery Handle

Can specify a message number to be passed to the scheduled application and placed in the Current Message buffer. The scheduled application could use a Play Message (PMSG) cell to play the message to the called party.

Delivery Event ID

Receives the identifier for the scheduled event. If you later decide to unschedule this event with an unschedule delivery (UDLV) cell, you must use this ID number. (See the description of the UDLV cell type.) It can be helpful for the application to use a play prompt with the data (PDAT) cell to play the delivery event ID to the caller so the caller can make a note of it.

Reschedule Interval

Used by the DELV cell only when the Time Type is RESCHEDULE. It tells Meridian IVR how many days to allow between automatically rescheduled deliveries. Automatically rescheduled events occur at the Delivery Time every Reschedule Interval number of days. It is possible to reschedule events up to seven days apart.

Buffers used

DELV uses the buffers specified by the following parameters:

Buffer	Explanation
Application Name	The application to be scheduled.
Outdial Number	Passed to the scheduled application and stored in its DIGITS buffer.
Delivery Time	Passed to the scheduled application and stored in its DELIVERY TIME buffer.
Delivery Interval	Passed to the scheduled application and stored in its DELIVERY INTERVAL buffer.
Delivery Attempts	Passed to the scheduled application and stored in its DELIVERY ATTEMPTS buffer.
Delivery Handle	Passed to the scheduled application and stored in its CURRENT MESSAGE buffer.

Buffers updated

The DELV cell updates the buffer specified by the Delivery Event ID parameter.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell will log the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer

Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Application Name (Buffer or Value)	“”	The name of the application to be scheduled. You can specify the actual name of the application between quotation marks (do not include the .vpf extension), or a buffer that contains the name of the application.
Outdial Number	DIGITS	The buffer containing the number to be outdialed by the scheduled application. This buffer is passed to the scheduled application as the DIGITS buffer.
Time Type (ABS/REL/ RESCHEDULE)	ABSOLUTE	Should the application be scheduled for one delivery at a specific date and time (ABSOLUTE), should it be scheduled for one delivery a certain length of time from the present (RELATIVE), or should it be scheduled for repeated delivery at specified intervals (RESCHEDULE)?
Delivery Time	DELIVERY TIME	The buffer that stores the delivery date and time. This buffer is passed to the scheduled application as the DELIVERY TIME buffer.
Delivery Interval (Buffer or Value)	DELIVERY INTERVAL	The length of time in minutes between execution of the DELV cell and execution of the application. It is required only if the Time Type is RELATIVE. This parameter is passed to the scheduled application and stored in the DELIVERY INTERVAL buffer.

Reschedule Interval (Value)	1	The time in days between each scheduled delivery. The value of this parameter can be 1–7. It is required only if the Time Type is RESCHEDULE.
Delivery Attempts (Buffer or Value)	DELIVERY ATTEMPTS	The maximum number of delivery attempts to make. The DELV cell does not use this information directly; it is passed to the scheduled application and stored in the DELIVERY ATTEMPTS buffer.
Delivery Handle	CURRENT MESSAGE	A buffer containing a message number to be passed to the scheduled application and stored in the CURRENT MESSAGE buffer.
Delivery Event ID	DELIVERY EVENT ID	The buffer that is updated with the Delivery Event ID number.

Next cell

Next cell	Explanation
Fail	The application could not be scheduled.
Success	The application has been scheduled.

Tables

None.

DIAL - dial digits

Description

The DIAL cell dials a series of DTMF digits on the current channel while a call is already in progress. This cell type does not outdial, but it can be used to dial digits to an external device such as a paging terminal.

Buffers used

The DIAL cell uses the buffer specified by the Phone Number parameter.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell will log the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Phone Number	DIGITS	The buffer containing the digits to be dialed. Can be a system buffer or a user-defined buffer.

Next cell

Next cell	Explanation
Error	The digits could not be dialed.
Success	The digits were dialed.

Tables

None.

DINS - insert records into an information database

Description

The DINS cell allows you to insert a record into an information database.

Buffers used

The DINS cell uses the buffer specified by the Record Number parameter.

Buffers updated

None.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.

Parameters	**Initial value**	**Explanation**
Database Name	<NONE>	The name of the information database from which records have to be inserted.
Record Number	DIGITS	The buffer that contains the storage location number that has to be inserted from the information database. The buffer can be a system buffer or a user-defined buffer.

Next cell

Next cell	**Explanation**
Error	The record number could not be inserted.
Record Exists	The record number already exists.
Success	The insertion of the record was successful.

Tables

None.

DMSG - delete message

Description

The DMSG cell removes messages from the Meridian IVR mailboxes where they are stored. The DMSG cell operates on the current message. If the message has just been recorded with the RMSG cell, the message is already in the CURRENT MESSAGE buffer.

Buffers used

Buffer	**Explanation**
CURRENT MESSAGE	The number of the message to be deleted.

Buffers updated

None.

Parameters

Parameter	Initial Value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.

Next cell

Next cell	Explanation
No Delete	The message cannot be deleted or the message does not exist.
Success	The message has been deleted.

Tables

None.

DOUT - directed outdial

Description

DOUT makes an outgoing telephone call on the current channel. Since DOUT places a call on the current channel, you cannot use this cell in applications that answer incoming calls. DOUT typically performs message delivery applications.

> ***Note:*** DOUT requires the use of an output channel. See the *Meridian IVR System Administration Guide* (NTP 555-9001-300) for a description of outbound channels.

After checking for a dial tone, an attempt is made to dial the number. If the number cannot be dialed, the application branches to the Error next cell. If the dialing is successful, it allows a specified maximum number of rings to occur. If this maximum occurs before the called party answers, the call is considered not to have been answered, and the application branches to the Ring next cell. If a busy signal is detected, the application branches to the Busy next cell.

If the call is answered, the application branches to the Success next cell.

> ***Note:*** DOUT has a parameter called “Ring Count.” For customers in the United States and Canada, “Ring Count” specifies the maximum number of rings. For customers outside the United States and Canada, “Ring Count” specifies the maximum number of tone bursts which is twice the number of rings.

Buffers used

The DOUT cell uses the buffer specified by the Phone Number parameter.

Buffers updated

STATUS.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Ring Count (0–31)	0	The number of rings allowed before branching to the Ring next cell. ***Note:*** If you set this parameter to 0, the application uses the default value of 31 rings.
Phone Number	DIGITS	The buffer containing the digits to be dialed. Can be a system buffer or a user-defined buffer.

Next cell

Next cell	Explanation
Error	Any other error has occurred.
Busy	The called number is busy.
Ring	The ring count has been reached without an error.

No Energy Detect	The secondary call was answered, but no voice energy has been detected. This branch is not supported for this release.
Voice	The call has been answered, and voice energy has been detected.

Tables

None.

EVENT - log application event

Description

The EVENT cell logs application-specific information at arbitrary points within a callflow as directed by the application writer. The EVENT cell automatically logs information such as date and time of event, application name, and a set of user-specified buffers containing application-specific information.

For each cell that uses an EVENT cell, two types of information are logged:

- data specific to the call
- data specific to each EVENT cell encountered

You can turn off the auditing capabilities of the EVENT cell without removing the cell by means of the “sconfig tool”.

To invoke sconfig, enter:

sconfig -audit [on off]

You must reset Meridian IVR 2.0/I for changes made by sconfig to take effect.

The information collected for the call includes the following:

- application name
- ANI
- channel number
- AP number
- trunk number
- start date and time
- stop date and time
- number of EVENT cells logged (up to 2 kbytes of data)

The information collected for each execution of an EVENT cell includes the following:

- application name (for GSUBs and EXECs)
- cell name
- cell number
- contents of comment field
- number of buffers to log (up to five)
- contents of these buffers (logs the values placed in these buffers by previous cells)
- date and time the cell execution completed

Note: Meridian IVR 2.0/I can log only 2048 bytes of call statistics for a single call (including call statistics logged by EVENT cells, and cells with CALL AUDITING enabled) to the audit_stat.d statistics file according to the following formula:

[(13 + (string length of buffers)) * the number of cells audited] 2048.

If this occurs, the system logs an error into the event log stating that the 2048-byte limit was exceeded and any further call statistics will be lost.

For each cell that is audited (using either the EVENT cell of Call Audit Enabled parameter or any other cell), the amount of data collected per audited cell is the following:

13 bytes + [number of characters +1 for each buffer specified (up to 5 buffers in the EVENT cell, or 1 representing the Audit Information parameter in any other cell)].

Buffers used

None.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Number of Event Buffers (0–5)	0	Number of buffers that contain event-specific information.

Next cell

Next cell	Explanation
Success	The event was successfully logged.

Tables

Tables	Explanation
Event Buffer Table	The buffers containing event specific information.

> ***Note:*** The data collected by the EVENT cell is also collected by every cell in the application call flow which has the parameter “Call audit enabled?” set to Yes.

Data collected by the EVENT cell is not sent to the audit_stat.d statistics file if there are no cells connected to the cleanup branch of the START cell.

All data is written to the file stat.d/audit_stat.d. You can convert this data to ASCII by means of the “exp” tool.

EXEC - execute application

Description

The EXEC cell tells Meridian IVR to begin executing another application on the current channel. This cell makes it possible to break a large application into several smaller ones. When a call is received, it can be handled by an application that determines which of several other applications should take the call. An EXEC cell can then turn the call over to the appropriate application.

> ***Note:*** When Application A does an EXEC of Application B, Application B should not use the ANSW cell because Application B is not answering the call. Application B only continues where Application A left off.

If the EXEC cell fails, the cleanup handler for a particular application is not executed.

If Meridian IVR is able to start the other application, control of the call is turned over to that application and the current application stops running. If Meridian IVR is unable to start the other application, the current application branches to the Error next cell.

The name of the application to be executed is taken from a buffer. You can use a STOR cell to put the filename into a buffer that makes the filename available to the EXEC cell. The filename must be placed in quotation marks. In addition, the .vpf extension, which is added to a filename by Meridian IVR, must be removed. For example, if you want to start the application named filename.vpf, you can use STOR to place the file “filename” into the buffer.

When the new application starts, the current application passes the values of all its system buffers to the new application. The values of its user-defined buffers are not passed to the new application; therefore, you should use only system buffers to store values which the new application will use.

> ***Note:*** EXEC cannot start the new application unless you load it. See the *Meridian IVR System Administration Guide* (NTP 555-9001-300) for information on loading and starting applications.

Buffers used

The EXEC cell uses the buffer specified by the Application Name Buffer parameter.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer

Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Application Name	DIGITS	The buffer containing the name of the application to be executed. It can be a system buffer or a user-defined buffer.

Next cell

Next cell	Explanation
Error	The other application could not be started. This is the only branch.

Tables

None.

GDAT - play prompts and get data

Description

The GDAT cell plays one or more prompts and collects up to 16 digits of input from the caller. Input is in the form of Dual Tone Multiple Frequency (DTMF) digits that the caller enters by pressing telephone keys.

The caller can enter digits during or after playback of the prompts. The Input Digit Buffer parameter designates the buffer that stores the caller’s input.

The GDAT cell has two parameters to limit the time the caller has for entering input:

- **Input Time-out** Sets the time limit for entering all of the digits.
- **Interdigit Time-out** Limits the time between entering one digit and the next. It also limits the time for entering the first digit.

If either of these time-out values is exceeded, the input is considered invalid and a time-out occurs. The caller hears the time-out prompt.

Digit entry ends when the caller has either entered the number of digits indicated by the Input Length parameter, or pressed the termination digit (usually the # symbol; see the Default Cell screen). If the caller does either of these things within the time limits, the input is considered valid.

If the caller hangs up instead of entering digits, the GDAT cell does not recognize the hang up and the clean-up branch is not taken. Only cells that play a prompt, such as the MENU cell and the PLAY cell, recognize the hang up and take the clean-up branch. Therefore, if you use a GDAT cell, include cells that check the number of digits in your application. This way, if a caller hangs up, these cells recognize the hang-up and route it to the HANG cell.

The GDAT cell has a retry feature that allows the caller to try again after a time-out. You choose the maximum number of time-out retries. If the caller exceeds this maximum, the application branches to the Time-out next cell.

The Abort Play on Input parameter controls what happens if the caller tries to enter input while a prompt is playing. If this parameter is set to Yes, a caller who is familiar with the application can skip through the prompts by pressing digits that the application is going to request.

> ***Note:*** The prompts stop playing once digit entry ends.

The Allow Overdialing parameter determines whether input is buffered from one cell to another. For example, if there are two cells that require a single digit of input, and if the Allow Overdialing parameter is set to Yes, the caller can press these digits in quick succession, and the second digit is saved until the second cell needs it. If the Abort Play on Input parameter is also set to Yes, the caller does not even hear the prompt for the second cell.

The GDAT cell includes the Term. Digit Buffer parameter for capturing the digit (if any) that the caller enters to terminate the string of digits they enter. The default buffer that is updated is the Termination Digit Buffer. This feature of the GDAT cell can be useful with successive PDAT or COMP cells. A PDAT cell with the Data Type parameter set to ALPHANUMERIC can play back the contents of the buffer specified in the GDAT cell's Termination Digit Buffer parameter. A COMP cell can determine whether the terminating digit captured by the GDAT cell is a “#” or “*”.

> ***Note:*** The # and * characters are always termination digits.

Buffers used

None.

Buffers updated

The GDAT cell updates the buffers specified by the Input Digit Buffer, # of Digits Input, and Termination Digit Buffer parameters.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Input Length (max 16)	1	The number of digits to accept from the caller.
Input time-out (0–75 secs)	15	The time in seconds in which the caller must enter all digits. If this value is exceeded, a time-out occurs. Setting this parameter to 0 sets the time allowed for entering digits to the time it takes to play all of the prompts specified in the Play Prompt Table.

Interdigit time-out (0–15 secs)	5	The maximum time that can elapse between entry of one digit and the next. It is also the maximum time for entry of the first digit. If this time is exceeded, than a time-out occurs. A value of 0 means that there is no limit.
Time-out Prompt	56	The number of the prompt that the caller will hear after a time-out occurs. The initial value, prompt 56, says “That is an invalid response.”
Number of Time-out Retries	3	The number of times the caller can try again after a time-out.
Abort Play on Input? (Y/N)	YES	Should the playback be aborted if the caller completes input entry during playback?
Allow Overdialing? (Y/N)	NO	Are digits allowed to accumulate from one cell to the next?
Input Digit	DIGITS	The buffer that accepts the caller's input. It can be a system buffer or a user-defined buffer.
# of Digits Input	NUMBER OF DIGITS	The buffer that stores the number of digits that have been accepted from the caller. It can be a system buffer or a user-defined buffer.

Term. Digit Buffer	TERMINATION DIGITS	The buffer that captures the # or * termination digit (if any) that terminates the string of digits entered by the caller. ***Note:*** If you set the termination to NONE in the Default cell, the * and # keys still act as termination digits. For example, if you enter *76 or #76 in the GDAT cell, the cell terminates on the * key, and no digits are collected.
Number of Prompts (max 32)	1	This value is updated automatically once the cell is saved with the count of the number of prompts entered in the Prompt Table. ***Note:*** If a buffer instead of a value is entered, then the value in the buffer indicates the number of prompts to play when the cell is executed in an application.

Next cell

Next cell	Explanation
Error	One or more of the prompts listed in the Play Prompt Table does not exist.
Time-out	The Number of time-out Retries has been exceeded.
Success	The prompts were played and input was collected from the caller.

Tables

Table	Explanation
Play Prompt Table	The prompts played to the caller. This can be specified as prompt numbers or as buffer names.

GSUB - call a subapplication

Description

The GSUB cell tells Meridian IVR to redirect execution to a new cell in either the current application or another application. If the GSUB cell calls a different application, execution starts with the first cell of the new application. When a RETN cell is executed, Meridian IVR transfers control to the Next cell - Success branch of the GSUB cell.

When Meridian IVR branches to the subroutine, the system buffers retain their current contents and are available in the called application. When the subroutine exists within the current application, the contents of any user buffer can be changed by the called application when control transfers back to the Next cell - Success of the GSUB cell from the calling application.

When the subroutine exists within another application, only the system buffers are passed. Any changes made to the system buffers in the called application affect the ones in the calling application. The called application's user buffers are completely internal and cannot be read or written by the calling application. The calling application's user buffers are not affected.

Notes:

1 You must load and start the application called by GSUB. You do not need to assign channels. Only system buffers are passed.

2 Do not use more than 12 levels of GSUB nesting. Exceeding this limit may cause unexpected results when you run your application.

Buffers used

The GSUB cell uses the buffer specified in the Application Name parameter. System buffers can change during processing of the subapplication (for example, between the GSUB and RETN cells).

Buffers updated

None. System buffers retain their contents at the start of the subapplication. However, buffers that are changed between the GSUB and the RETN cells retain these changes after the RETN cell is executed. The value inside the system buffer reflects the outcome of whatever operation occurs during the subapplication. Any user buffer's contents in the application containing the GSUB cell are saved prior to the start of the subapplication and later restored after the RETN cell is executed. The value inside the user buffer remains the same and is unaffected by the subapplication. If the GSUB branches within the current application, user-defined buffer values are treated as system buffers, and their values can be altered upon RETN.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Application Name (Buffer or Value)	“”	The name of the application to begin executing. If a value, the application name should be in quotation marks. If the subroutine is within the current application, you simply specify a blank string “”.

Next cell

Next cell	Explanation
Error	The operation was not successful. No such application is currently loaded.
Success	The operation was successful. This is where execution resumes after the RETN cell is executed.
GSUB	This appears if there is an internal subroutine with no application name specified.

Tables

None.

HANG - hang up

Description

The HANG cell the call on the current channel. There is an option to play prompts to the caller before hanging up the channel.

Buffers used

None.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Number of Prompts (max 5)	0	This value is updated automatically once the cell is saved with the count of the number of prompts entered in the Prompt Table. ***Note:*** If a buffer instead of a value is entered, then the value in the buffer indicates the number of prompts to play when the cell is executed in an application.

Next cell

None.

Tables

Table	Explanation
Play Prompt Table	The prompts played to the caller. The table can be specified as prompt numbers or as buffer names.

INFO - get call information

Description

You can use the INFO cell for an application that requires detailed information on either incoming or outgoing calls. For example, an application that successfully executes a COUT cell can call an INFO cell to determine the address of the agent that answered the call in a call center environment.

Cell type

Call Management.

Buffers used

None.

Buffers updated

Upon a call to the INFO cell, the Calling DN (DN of caller) ID returned in the ANI_DIGITS buffer and the Called DN (DN dialed by caller) is returned in the DIGITS buffer.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Call State	DATA EXCHANGE #1	Last Call Progress State change
Call Info	DATA EXCHANGE #2	Last Call Info State change
Called TN	DATA EXCHANGE #3	Called TN in Packed Format. The Called TN is 4 bytes of numeric data which are the actual SL–1 physical address of the set. This field is only set for internal calls that have been answered. m_GetCallInfo returns the TN information in the Meridian 1 internal format. This format can be converted to the TN of the agent's set using the following algorithm: TN (packed format) = {TN[0] * 256^3 + TN[1] * 256^2 + TN[2] * 256 + TN[3]}
Call ID	DATA EXCHANGE #4	Unique Call Identified

Call DNIS	DATA EXCHANGE #5	Directory Number Identification System
Channel ID	DATA EXCHANGE #6	Voice Channel Identifier

Next cell

Next cell	Explanation
Error	The Call Information was not returned.
Success	The Call Information was returned successfully.

Tables

None.

LANG - set language

Description

The LANG cell sets the offset for system and application-specific prompts, and associates that offset with a particular language. You can use this cell to avoid the occurrence of conflicting prompts from different applications.

The LANG cell allows you to specify a base prompt number for system prompts and a separate base prompt number for application prompts. As prompt IDs are used throughout the application, the appropriate offset is applied.

The PDAT cell uses the Language parameter. The value for this parameter determines the syntax used for speaking numbers, dates and so on.

The Vocabulary Group parameter lists those groups with common vocabularies. You can enter this value directly into the text field. Meridian IVR validates this list against the list specified by the field for the Vocabulary Group browser and warns of invalid values.

Predefined system prompts range from 0 to 999. If you want to support multiple languages, the system prompt range is enlarged by 1000 prompts for every language supported. For example, if you support English, French, Spanish and German you need (4 x 1000) = 4000 prompts. You can lay out the prompt on the disk in the following manner:

0–999	English system prompts
1000–1999	French system prompts
2000–2999	Spanish system prompts
3000-3999	German system prompts

You can also have multilingual application prompts. For example, you can map these prompts in the following manner:

4000–4999	English application prompts
5000–5999	French application prompts
6000–6999	Spanish application prompts
7000-7999	German application prompts

Once you place the prompts on Meridian Mail, you can program the application to support these languages. A MENU cell allows the caller to execute the application in whatever language they choose. Each branch of the MENU cell goes to the LANG cell that loads the value of the offset on the disk where the system prompts start (English=0, French=1000, Spanish=2000, German=3000), where the application prompts start (English=4000, French=5000, Spanish=6000, German=7000), and the language ID which is used to format data played through the PDAT cell.

You record the prompts needed to support the additional character sets using the procedure for recording application prompts. The offset to the first such prompt is specified in the field “ISO-8859/1 Offset” in the LANG cell. For example, if the application prompts start at 500 and ISO-8859/1 prompts are stored in the range 564 to 660, set the field “ISO-8859/1 Offset” to 64. Set this value only once in the application, regardless of the number of PDAT cells. Using the LANG cell enables multilingual systems to connect to the host. Functionality for the multiple-character set is incorporated into the PDAT cell by the data type STRING.

To support the ISO-8859/1 character set, you must record 96 additional prompts in the sequence defined by the ISO-8859/1 character set. Once recorded, store these prompts in the range of application prompts (that is, not in the range of system prompts). This procedure enables you to record and store the prompts only when the multiple-character set is needed. If you store these prompts in the range of system prompts, then you cannot use the 96 corresponding prompt numbers when communicating with a non-ASCII character host.

International Host Connectivity

Meridian IVR currently supports connectivity to hosts that use ASCII character sets. To support international host connectivity, the MIVR 2.06 release extends connectivity to hosts that use non-ASCII 8-bit character sets: namely, EBCDIC and ISO-8859/1. This feature enables applications to send and receive data from the host. In addition, the data retrieved from the host at run time is represented in ISO-8859/1 (that is, ASCII 8-bit character set) regardless of the character set of the host. You can use the buffer data with any other cell.

Database Support

Local databases supported by Meridian IVR store information in ISO-8859/1 character representation (that is, non-ASCII character sets) regardless of the type of character set of the host. As a result, data from both within the database and data in the runtime components is represented in one character set only. Since runtime data uses ISO-8859/1 character representation, applications can store and update the local database using the same character set. You must ensure that data in the local database is in ISO-8859/1 format.

Procedure 7-1
Relocating application prompts

1 With the MVPMT tool, copy the prompts to the new location.

Refer to the “Tools” chapter in the *System Administration Guide* (NTP 555-9001-300) for more information about the MVPMT tool.

2 Add the LANG cell to your application anywhere before the first cell that references your application prompts.

3 Specify the amount that the application prompts have been moved in the field application offset, that is, the application prompt was moved to prompt location 300. The application offset field of the LANG cell contains the difference between the location specified in the application and the new location which is 300 – 200 = 100.

To use the LANG cell for multilingual purposes, do the following:

- If your application supports multiple languages, the system prompts field of the LANG cell is updated with the offset where the system prompts of a particular language are stored on Meridian Mail. For example, if the French prompts on Meridian Mail started at prompt 500, enter 500 into the System Prompts field of the LANG cell.
- If your application supports application prompts for multiple languages, the application prompts field of the LANG cell is updated with the offset where the application prompts of a particular language are stored on Meridian Mail. For example, if the French application prompts on Meridian Mail started at prompt 1500, enter 1500 into the application prompts field of the LANG cell.
- To provide multilingual formatting of data in the PDAT cell, the language field of the LANG cell is updated. For example, if you want the formatting of data to be in French, specify FRENCH in the language field of the LANG cell.

  Note: All applications should contain a LANG cell in case you need to relocate prompts. This way, only the LANG cell is affected. The LANG cell values are preserved in any application that calls subroutines or executes with the EXEC cell.

Buffers used

None.

Buffers updated

None.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
System Prompts	0	Buffer or value: Offset to be added to the application prompts to determine the actual prompt to play.
Application Prompts	0	Buffer or value: Offset to be added to the application prompts to determine the actual prompt to play.
ISO-8859/1 Prompts	0	This value specifies the offset of the first ISO-8859/1 prompt. For example, if the application prompts start at 500, and the ISO-8859/1 prompts are stored in the range 564 to 660, the offset value for the first ISO-8859/1 prompt is 64.

Parameters	Initial value	Explanation
Language	English	Buffer or value to be used to determine proper syntax for PDAT data.
Vocabulary Group Name	“NONE”	Name associated with a group of vocabularies. The list of possible choices is generated from the vocabulary system file.

Next cell

Next cell	Explanation
Success	Language properly defined with valid offsets.
Error	An error has occurred; possible cause is that offset is not within the allowed range.

Tables

None.

LDLV - get schedule event list

Description

The LDLV cell retrieves information associated with a delivery application that has been scheduled by a Schedule a Delivery Application (DELV) cell. Given the application name and the outdial number, LDLV retrieves the following items:

- the delivery event ID
  (LDLV puts this into the specified buffer.)
- the message number associated with the scheduled application
  (LDLV puts this into the specified buffer.)
- the buffers DATA EXCHANGE #1, DATA EXCHANGE #2, DATA EXCHANGE #3, DATA EXCHANGE #4, and DATA EXCHANGE #5

You can use one LDLV cell repeatedly in an application to retrieve information about every scheduled event for the specified application and out-dial number. Because there can be several such events, you may need to check each one by having the application use a PMSG cell to play the retrieved message and five PDAT cells to play the contents of the DATA EXCHANGE buffers.

Buffers used

LDLV uses the buffers specified by the Application Name and Outdial Number Buffer parameters.

The DATA EXCHANGE buffers (1–5) are used in the DELV cell to pass data to the application being scheduled in the DELV cell. The LDLV cell only retrieves the contents of those buffers.

Buffers updated

LDLV updates the buffers specified by the Delivery Handle and Schedule Event ID parameters. It also retrieves the DATA EXCHANGE #1, DATA EXCHANGE #2, DATA EXCHANGE #3, DATA EXCHANGE #4, and DATA EXCHANGE #5 buffers.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.

Application Name (Buffer or Value)	“”	The scheduled application. You can specify the actual name (without the extension) of the application in quotes, or in a buffer that contains the name of the application. This is a mandatory field.
Outdial Number	DIGITS	The phone number that the scheduled application is set to call. This is a mandatory field.
Delivery Handle	CURRENT MESSAGE	The buffer containing the message number to be passed to the scheduled application.
Schedule Event ID	DELIVERY EVENT ID	The buffer that stores the ID of the scheduled application. If you decide to unschedule the application with a UDLV cell, you have to use this ID. A STOR cell should be used immediately prior to the LDLV cell to set the Event ID to 0. Any other value will cause the cell to retrieve the wrong Event ID associated with that scheduled application. ***Note:*** The Schedule Event ID must be set to zero for the whole list of events to be searched.

Next cell

Next cell	Explanation
Fail	An error has occurred.
No Events	There are no more scheduled events with the specified application name and out-dial number.
Success	Information about the scheduled event has been retrieved.

Tables

None.

MATH - perform mathematical operation

Description

The MATH cell performs the following mathematical operations: addition, subtraction, multiplication, division, and producing a remainder. The math operation identified by the Operation Flag is performed on two operands that are buffers or constant values. The result of the operation is stored in the Result Buffer.

The MATH cell indicates what its highest and lowest possible values are. This cell warns you of inconsistent values greater than the maximum value or lower than the minimum value. The MATH cell only takes an error branch when a division by zero is performed.

You can use the MATH cell only with numerical value results or those represented as buffers or constants. If one of the operands is a character string, the MATH cell takes the error branch. Only integer results are possible; consequently, when the operation is division, only the whole number part is stored in the Result Buffer. If the result of the operation is greater than 2,147,483, 647 or less than –2,147,483,648, inconsistent results are returned.

Buffers used

MATH uses Buffer A and Buffer B.

Buffers updated

Result Buffer is updated with the result of the math operation.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Buffer A (or Value)	DIGITS	The first operand. This can be a buffer (either a system buffer or a user-defined buffer) or a constant value (maximum number of 2,147,483,647 and minimum number of –2,147,483,648).
Operation Flag (+,–,/,*,%)	+	The operation to be performed.
Buffer B (Or Value)	0	The second operand. This can be a buffer (either a system buffer or a user-defined buffer) or a constant value (maximum number of 2,147,483,647 and minimum number of –2,147,483,648).
Result Buffer	DIGITS	A buffer in which the result of the mathematical operation is stored. This can be a system buffer or a user-defined buffer.

Next cell

Next cell	Explanation
Error	The operation was not successful.
Success	The operation was successful.

Tables

None.

MDTE - get Meridian Mail date and time

Description

The MDTE cell obtains the date and time as set in the Meridian Mail machine (through the use of the Meridian ACCESS API m_GetSysDate). The returned date and time are those of the Meridian Mail machine's general system clock. The returned date is used as the time stamp on all files and the reference point for delayed message delivery. The Current Date parameter contains the date in the format MMDDYYYY.

Cell type

Application Scheduling.

Buffers used

None.

Buffers updated

The MDTE cell updates the buffers specified by the Current Time and Current Date parameters.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Current Date	DATE	The buffer containing the current MM date.
Current Time	TIME	The buffer containing the current MM time.

Next cell

Next cell	Explanation.
Error	The MM date and time were not returned.
Success	The MM date and time were returned successfully.

Tables

None.

MENU - play menu and get data

Description

The MENU cell plays a menu to the caller, accepts the caller’s menu selection, checks the Valid Input Table to find the next cell corresponding to the selection, and then branches to the next cell. The menu selection must be a single digit that the caller enters by pressing a telephone key. If the menu selection cannot be found in the Valid Input Table, the application branches to the Invalid Input next cell. If the caller does not enter a digit within the time limit set by the Input Time-out parameter, a time-out occurs and the caller hears the invalid input prompt.

The MENU cell has a feature that allows the caller to try again after entering invalid input. The # of Invalid Input Retries parameter determines the maximum number of times that the caller can enter invalid input. If the caller exceeds the maximum number of invalid input retries, the application branches to the Invalid Input next cell.

The Abort Play On Input parameter controls what happens if the caller tries to enter input while a prompt is playing. If this parameter is set to Yes, a caller who knows the application can skip through the prompts by pressing digits that the application is going to request. In addition, the caller does not hear the prompt for the second cell.

The Allow Overdialing parameter determines whether input is buffered from one cell to another. For example, if there are two cells that both require a single digit of input, and if the Allow Overdialing parameter is set to Yes, the caller can press these digits in quick succession and the second digit is saved until the second cell needs it.

The Interdigit timeout value parameter specified in the START cell is used to limit the time to press the first digit in the input to be entered, and the time between each digit in the input. In the START cell, you should set this parameter to five seconds. If this time is exceeded without a resource becoming available, the error branch is taken. Note that the Interdigit timeout value in the START cell cannot exceed the Input timeout value specified in the MENU cell.

Buffers used

None.

Buffers updated

Buffer	Explanation
DIGITS	The digit obtained from the caller.
NUMBER OF DIGITS	The number of digits obtained from the caller. For MENU, the number of digits is always 1.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Valid Inputs	0	Valid inputs include the following: Single digits 0–9 Special characters * and # Letters A, B, C, D.
Invalid Input Prompt	56	The prompt the caller hears after entering invalid input. The initial value, prompt 56, says, “That is an invalid response.”

Number of Invalid Input Retries	3	The maximum number of times the caller can enter invalid input and incur a time-out before the application branches to the time-out next cell.
Time-out Prompt	56	The number of the prompt that the caller hears after a time-out occurs. The initial value, prompt 56, says, “That is an invalid response.”
Input Time-out (0–75 secs)	15	The time in seconds in which the caller must enter all digits. If this value is exceeded, a time-out occurs. Setting this parameter to 0 sets the time allowed for entering digits to the time it takes to play all of the prompts specified in the Play Prompt Table.
Number of Time-out Retries	3	The number of times the caller can try again after exceeding the Input Timeout value or the Interdigit Timeout value.
Abort Play on Input? (Y/N)	YES	Should the playback be aborted if the caller completes input entry during playback?
Allow Overdialing? (Y/N)	NO	Are digits allowed to accumulate from one cell to the next?
Number of Prompts (max 32)	1	This value is updated automatically once the cell is saved with the count of the number of prompts entered in the Prompt Table.

Next cell

Next cell	Explanation
Error	One or more of the prompts listed in the Play Prompt Table does not exist.
Invalid Input	The caller's menu selection could not be found in the Valid Input Table.
Time-out	The Number of time-out Retries has been exceeded.

Note: If the caller's input or menu selection is found in the Valid Input Table, the next cell is taken from the Valid Input Table.

Tables

Table	Explanation
Play Prompt Table	The prompts that are played to the caller. They can be specified as prompt numbers or as buffer names.
Valid Input Table	A list of all valid inputs (menu selections) and the next cell corresponding to each. Valid inputs must be single digits.

PDAT - play prompts with data

Description

The PDAT cell plays prompts to announce the contents of a buffer. The buffer content is read according to the specified type. For example, if the buffer contains the number 0102 and the type is *date*, the number is read as "January second"; if the type is *number*, the number is read as "one hundred two"; and if the type is *digits*, the number is read as "zero one zero two."

If the type is date and the digits are not valid for dates, the PDAT cell takes the INVALID DATA branch. Notice that the date and time have specific formats:

- The format for date is either mmdd (two digits for the month and two digits for the day — four digits altogether) or mmddyyyy (two digits for the month, two digits for the day, and four digits for the year — eight digits altogether). Only the first four digits, which are the digits for the month and day, are read.
- The format for time is hhmm (two digits for the hour and two digits for the minute— four digits altogether). A 24-hour clock is used. The PDAT cell includes “a.m” or “p.m.” when it reads the time.
- The format for date and time is mmddyyyyhhmm (two digits for the month, two digits for the day, four digits for the year, two digits for the hour, and two digits for the minute). There are 12 digits altogether. A 24-hour clock is used. The year is not read. The word “at” is inserted between the day and the hour. For example, if the buffer contains the value “010119900130", and the type is *date and time*, PDAT reads the number as “January 1 at 1:30 a.m.” Notice that the year, 1990, is not read even though it is in the buffer.

Introductory prompts can be played before the number is read.

> ***Note:*** The PDAT cell uses system prompts. They must already have been installed on Meridian Mail before PDAT can function. See Appendix A for a list of standard prompts.

Buffers used

The PDAT cell uses the buffer specified by the Data Buffer parameter.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Abort Play on Input	No	This parameter controls what happens if the caller enters input while a prompt is playing. If this parameter is set to Yes, then the caller who is familiar with the application can skip through the prompts by pressing digits that the application is going to request.

Allow Overdialing	No	This parameter determines whether input is buffered from one cell to another. If you set this parameter to No, then the input buffer is cleared before the next GDAT or MENU cell requests input. However, if you set this parameter to Yes, the input buffer is not cleared. The buffer saves the digits until the next request for input. For example, if there are two cells that need a single digit of input and this parameter is set to Yes on the second cell, then the caller can press two digits in quick succession at the first cell, and the second digit is saved until the second cell needs it.
Data Type	TIME	The format for the number. Valid data types are DIGITS NUMBER DATE TIME DATE AND TIME DATE WITH YEAR MILITARY TIME DAY OF WEEK DOLLARS DOLLARS AND CENTS FRACTIONS FLOATING POINT ALPHANUMERIC DIGITS NUMBER

Data Buffer	DIGITS	The buffer containing the number to be read. The number must be valid for the data type. The buffer can be a system buffer or a user-defined buffer.
Number of Prompts (max 5)	0	This value is updated automatically once the cell is saved with the count of the number of prompts entered in the Prompt Table. ***Note:*** If a buffer instead of a value is entered, then the value in the buffer indicates the number of prompts to play when the cell is executed in an application.

Next cell

Next cell	Explanation
Error	The number is invalid for the type, or one or more of the introductory prompts do not exist or could not be played.
Inv Data	The data input is not valid for the format specified.
Success	The number has been read.

Tables

Table	Explanation
Play Prompt Table	The introductory prompts that are played to the caller. They can be specified as prompt numbers or as buffer names.

PLAY - play prompts

Description

The PLAY cell plays prompts to the caller. This cell can play a maximum of 32 prompts.

Buffers used

None.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer

Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Number of Prompts (max 32)	1	This value is updated automatically once the cell is saved with the count of the number of prompts entered in the Prompt Table. ***Note:*** If a buffer instead of a value is entered, then the value in the buffer indicates the number of prompts to play when the cell is executed in an application.

Next cell

Next cell	Explanation
Error	One or more of the prompts listed in the Play Prompt Table does not exist.
Success	The prompts were played back successfully.

Tables

Table	Explanation
Play Prompt Table	The introductory prompts that are played to the caller. They can be specified as prompt numbers or as buffer names.

PMSG - play a message

Description

The PMSG cell plays a message to the caller.

The PMSG cell plays the message located in the CURRENT MESSAGE buffer. If the RMSG cell just recorded the message, the message is already in the CURRENT MESSAGE buffer. The caller can stop playing the message by pressing the Termination Digit.

> ***Note:*** To retrieve a message that has been forwarded in a Meridian mailbox, you should use Meridian Mail.

Buffers used

CURRENT MESSAGE is the number of the message to be played.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer

Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Termination Digit	DEFAULT	The key that the caller can press to terminate playback. DEFAULT is the termination digit from the Default Cell screen.

Next cell

Next cell	Explanation
Error	The message could not be played or does not exist.
Success	The message was played back successfully.

Tables

None.

PRGS - call progress detection

Description

The PRGS cell obtains the termination state of the previous out-dial attempted by COUT or DOUT cells. This cell contains multiple branches depending on the call’s progress.

Cell type

Call Management.

Buffers used

None.

Buffers updated

None.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Call State	TBD	Last call progress state change.
Call Info	TBD	Last Call Info State change.

Next cell

Next cell	Explanation
No Answer	The call was not answered.
Misc. Answer	The call was answered (Miscellaneous).
Short	The call was answered by Short Silence.
Long	The call was answered by Long Silence.
Pager	The call was answered by Pager.
Error	Failure to establish connection.
Voice	The call was answered by Voice.

Tables

None.

QCNT - SQL select count cell

Description

The QCNT cell interfaces with a relational DBMS to execute an SQL Select Count statement that returns the number of rows in a database table or views matching specific selection criteria. Upon successful completion of the cell, the SQL ROWS system buffer contains the number of rows that were found.

Buffers used

Buffers can be used in a “WHERE” clause.

Buffers updated

Buffer	Explanation
SQL ROWS	QCNT updates this buffer with the number of rows matching the selection criteria.
SQL ERRORS	If an error occurs, QCNT updates this buffer.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
SQL Table Name	“ ”	The name of the table, view, or a buffer containing the name for which the Select count is being requested.

Next cell

Next cell	Explanation
Fail	An error occurred while processing this cell transaction. The system buffer SQL ERROR contains the error code returned by the DBMS.
No Rows	No rows in the table or view meet the WHERE clause criteria. The system buffer, SQL ROWS, is set to zero.
Success	The function has been successfully performed. The system buffer, SQL ROWS, contains the number of rows that were found.

Tables

Tables	Explanation
WHERE Clause	The WHERE clause table identifies the conditions against which the transaction is matched. The WHERE clause table consists of the following fields: TYPE: Value fields data type (STRING, NUMERIC, MONEY, DATE, FLOAT) LOCATION: Logic operator (AND, OR, NOT) (‘s: Left parenthesis type COLUMN: SQL column name or SQL expression OPERATOR: Operator (<,=,>) VALUE: Value or buffer)’s: Right parenthesis type

QDEL - SQL delete cell

The QDEL cell interfaces with a relational DBMS to execute an SQL delete statement. This transaction removes one or more rows from a specified database table. The rows to be deleted are selected according to the criteria you specify in the WHERE Clause table. Upon successful completion of the cell, the SQL ROWS system buffer contains the number of rows deleted.

Buffers used

Buffers can be used in the WHERE clause.

Buffers updated

Buffer	Explanation
SQL ROWS	QDEL updates this buffer.
SQL ERRORS	If an error occurs, QDEL updates this buffer.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment Field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
“SQL Table Name”	“ ”	The name of the table or view, or a buffer containing the name for which the Select Count is being requested.

Next cell

Next cell	Explanation
Next cell - Fail	An error has occurred. The system buffer SQL ERROR contains the error code returned by the DBMS.
Next cell - No Row	No rows in the specified table or view meet the WHERE clause criteria. The system buffer SQL ROWS is set to zero.
Next cell - Success	The delete rows transaction was successful. The system buffer SQL ROWS contains the number of rows that have been deleted.

Tables

Tables	Explanation
WHERE Clause	The WHERE clause table identifies the conditions against which the transaction is matched. The WHERE clause table consists of the following fields: TYPE: Value fields data type (STRING, NUMERIC, MONEY, DATE, FLOAT) LOGICAL: Logic operator (AND, OR, NOT) (‘s: Left parenthesis type COLUMN: SQL column name or SQL expression OPERATOR: Operator (<,=,>) VALUE: Value or buffer)’s: Right parenthesis type

QINS - SQL insert cell

Description

The QINS cell interfaces with a relational DBMS to execute an SQL Insert statement. QINS adds a row to a specific database table according to the Column and Value table you specify.

Buffers used

Any of the values you list in the Column and Value table can be Application Editor buffer names.

Buffers updated

Buffers	Explanation
SQL ERROR	If an error occurs, QINS updates this buffer.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
“SQL Table Name” (Buffer Value)	“ ”	The name of the table to view from which a row of data is selected.

Next cell

Next cell	Explanation
Fail	An error has occurred while processing this transaction. The system buffer SQL ERROR contains the error code returned by the DBMS.
Success	A row has been added to the specified database table with the values listed in the Column and Value table.

Tables

Tables	Explanation
Column and value	A list of column names and the corresponding values for the row being inserted in the database table.
TYPE:	Value field data type (STRING, NUMERIC, MONEY, DATE, FLOAT)
WHERE Clause	The WHERE clause table identifies the conditions against which the transaction is matched. The WHERE clause table consists of the following fields: TYPE: Value fields data type (STRING, NUMERIC, MONEY, DATE, FLOAT) LOCATION: Logic operator (AND, OR, NOT) (‘S: Left parenthesis type COLUMN: SQL column name or SQL expression OPERATOR: Operator (<,=,>) VALUE: Value or buffer)’s: Right parenthesis type

QSEL - SQL select cell

Description

The QSEL cell interfaces with a relational database DBMS to execute an SQL Select statement that retrieves a row of data from a database table or view. The columns to be selected and the buffers into which the values should be returned are identified in the Column and Buffer table. You specify the selection criteria for the retrieval in the WHERE Clause table. Although there is no limit to the number of rows that an SQL Select statement can return, the QSEL cell returns the values for only one row. To select multiple rows, you should include one QSEL per row.

Buffers used

Buffers can be used in a “WHERE” clause.

Buffers updated

Buffers	Explanation
Buffers listed in the Column and Buffer table	Column values for the row matching the WHERE clause selection criteria are returned into the specified buffers.
SQL ERROR	If an error occurs, this buffer is updated with the error code returned by the DBMS.
SQL ROWS	This buffer contains the number of rows returned by the transaction. Since QSEL returns a maximum of one row, this buffer IS set to 0 or 1.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
SQL Table Name (Buffer Value)	“ ”	The name of the table to view from which a row of data is selected.

Next cell

Next cell	Explanation
Fail	An error occurred while processing this transaction. The system buffer SQL ERROR contains the error code returned by the DBMS.
No rows	No rows in the table or view meet the WHERE clause criteria. The system buffer SQL ROWS is set to zero.
Success	One or more rows matched the WHERE clause criteria. The system buffer SQL ROWS is set to 1.

Tables

Tables	Explanation
Column and value	A list of columns and the Application Editor Buffers into which the corresponding values are returned for the row retrieved from the database table.
WHERE Clause	The WHERE clause table identifies the conditions against which the transaction will be matched. The WHERE clause table consists of the following fields: TYPE: Value fields data type (STRING, NUMERIC, MONEY, DATE, FLOAT) LOCATION: Logic operator (AND, OR, NOT) (‘S: Left parenthesis type COLUMN: SQL column name or SQL expression OPERATOR: Operator (<,=,>) VALUE: Value or buffer)’s: Right parenthesis type

QUPD - SQL update cell

Description

The QUPD cell interfaces to a relational DBMS to execute an SQL Update statement that modifies the values in one or more rows of a database table. Identify the list of columns to be changed and the corresponding new values in the Column and Value table. Specify the criteria for selecting the rows to be modified in the WHERE Clause table. When the cell completes successfully, the system buffer SQL ROWS contains the number of rows that were updated.

Buffer used

Buffers can be used in a “WHERE” clause.

Buffers updated

Buffers	Explanation
SQL ERROR	If an error occurs, QUPD updates this buffer to the error code returned by the DBMS.
SQL ROWS	QUPD updates this buffer to reflect the number of rows that are updated.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
SQL Table Name (Buffer Value)	“ ”	The name of the table or view from which a row of data is updated.

Next cell

Next cell	Explanation
Fail	An error occurred while processing this transaction. The system buffer SQL ERROR contains the error code returned by the DBMS.
No rows	No rows in the table or view meet the WHERE clause criteria. The system buffer SQL ROWS is set to zero.
Success	One or more rows were updated in the specified database table. The system buffer SQL ROWS contains the number of rows that were modified.

Tables

Tables	Explanation
Column and Value	A list of column names and their values for the rows being updated in the database table.
WHERE Clause	The WHERE clause table identifies the conditions against which the transaction is matched. The WHERE clause table consists of the following fields: TYPE: Value fields data type (STRING, NUMERIC, MONEY, DATE, FLOAT) LOCATION: Logic operator (AND, OR, NOT) (‘S: Left parenthesis type COLUMN: SQL column name or SQL expression OPERATOR: Operator (<,=,>) VALUE: Value or buffer)’s: Right parenthesis type

RETN - return from a subroutine call

Description

The RETN cell is used in conjunction with the GSUB cell. When the RETN cell is executed, Meridian IVR returns control to the next cell specified by the GSUB cell. The GSUB cell can be in the same application or a different application than the RETN cell. If control cannot be transferred, Meridian IVR executes the error branch of the next cell specified for the GSUB cell.

Buffers used

None.

Buffers updated

None. Any system buffers that were changed between the GSUB and the RETN cell remains changed after the RETN cell is executed. This also applies to user-defined buffers if the RETN cell is within the same application as the GSUB cell. If the RETN cell is in a separate application, the user-defined buffers are restored to their values before the GSUB cell is executed.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.

Next cell

Next cell	Explanation
Error	The operation was not successful.

Tables

None.

RMSG - record message

Description

The RMSG cell allows the caller to record a message. The number of the newly recorded message is then stored in the CURRENT MESSAGE buffer.

To end recording and save the message, the caller can press the Termination Digit. Alternatively, the caller can press the Abort Digit to end recording, but the message is discarded and is not added to a Meridian mailbox.

The Maximum Recording Time sets a limit. If the caller does not press either the Termination Digit or the Abort Digit within this limit, recording ends automatically. The application can then branch to the Time-out next cell.

The Introduction Prompt Table lists prompts to be played for the caller before recording begins. The prompts can tell the caller about the Termination and Abort digits as well as the maximum recording time.

After the RMSG cell records a message, the application can either

- delete the message from the CURRENT MESSAGE buffer using the DMSG cell
- put the message into a Meridian mailbox using the predefined access user function

If the message is not deleted or added to the mailbox, the contents of the message are no longer available to Meridian IVR.

Buffers used

None.

Buffers updated

Buffer	Explanation
CURRENT MESSAGE	The number of the newly recorded message.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Maximum Recording Time (1–840 secs)	60	The maximum recording time in seconds. If this value is exceeded, recording ends automatically.
Message length	MESSAGE LENGTH	A buffer which captures the returned value of the length of the current recorded message, in seconds.
Termination Digit	DEFAULT	The key the caller can press to end recording and keep the message. DEFAULT is the termination digit from the Default Cell screen.

Abort Digit	DEFAULT	The key the caller can press to end recording and erase the message. DEFAULT is the abort digit from the Default Cell screen.
Abort Prompt	DEFAULT	The prompt the caller hears after pressing the abort digit. DISABLE means that no prompt is played. DEFAULT is the abort prompt from the Default Cell screen.
Number of Prompts (max 5)	0	This value is updated automatically once the cell is saved with the count of the number of prompts entered in the Prompt Table.

Next cell

Next cell	Explanation
Error	The message has not been recorded.
Time-out	The maximum recording time as specified (from 1 to 840 seconds) has been exceeded.
Success	Recording has been successful. If you want to add the recorded message to the Meridian IVR mailbox, the next cell should be a USER cell that puts the message in the mailbox. Using this approach ensures that the message is saved even if the caller hangs up.

Tables

Table	Explanation
Introduction Prompt Table	The prompts played to the caller before recording begins. They can be specified as prompt numbers or as buffer names.

SMSG - send message

Description

The SMSG cell sends a message that has been previously recorded and addressed.

Cell type

Messages.

Buffers used

The SMSG cell uses the buffer specified by the Current Message parameter.

Buffers updated

None.

Parameters

Parameter	Valid values	Default
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Current Message	CURRENT MESSAGE	The number of the message to be sent.

Priority	Normal, Urgent, Economy	Normal
Tags	None, Private	None

Next cells

Next cell	Explanation
Error	The message was not sent.
Success	The message was sent successfully.

Tables

None.

START - default cell

The START or Default cell is usually the first cell in your application. This cell allows you to do the following:

- Select a message database for the application to use.
- Establish the Minimum Voice Duration which relates to a special option for MENU cells.
- Choose any cell in the application as the first to be executed when you run the application.
- Designate the first cell in a sub-application for this application. Subapplications are explained later.

Another important function of the START cell is that it allows you to establish default values for some of the cell parameters in the application. For example, an application can have an RMSG cell that allows the caller to record messages. One of the RMSG parameters is the Termination Digit.The value you select for this parameter determines the key that the caller should press to finish recording a prompt or a message. To select a value for this parameter, you can enter any value from 0 to 9, * or #, A, B, C, or D as the Termination Digit.

However, instead of entering a specific value, you can enter DEFAULT as the value of the Termination Digit parameter in the RMSG cell. If you do, the actual value of this cell's Termination Digit is taken from the Termination Digit on the Default cell parameter.

If your application has many RMSG cells, this feature can be very useful. If you change the value on the Default cell screen, you automatically change the value of the Termination Digit in any RMSG that has been assigned the value DEFAULT. This way, you can use a single change on the Default cell parameter to change the whole application.

Other cell types can have a Termination Digit. If you use the value DEFAULT for all the Termination Digits, the application will have a consistent "feel". You should use the value DEFAULT wherever possible to ensure that key actions and time-out values remain the same from cell to cell.

There are several restrictions you need to remember when you use the cleanup branch in your application.These restrictions are listed in the attention box found in the section "Designing an application" on page 3-3 in Chapter 3.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer

Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Abort Digit	NONE	The caller can press this digit to stop recording a prompt or a message. Once the caller presses this key, the application does not save the recording or add it to the message database.
Abort Prompt	117 Recording has been aborted. The message will not be saved.	The caller hears this prompt if playback or recording is aborted. This can happen if the caller exceeds the Maximum Paused Time, and your application allows you to select ABORT as the Pause time-out Action.
Termination Digit	#	The caller presses this digit to finish recording a prompt or a message, or to finish playing back a message. When the caller uses this key to finish recording, the application saves the recording and adds it to the message database.
Interdigit Time-out (seconds)	5	The maximum amount of time to press the first digit in the input, and the time between each digit in the input.
Minimum Voice Duration (ms)	200	Sets the minimum valid duration of speech that can be accepted for positive confirmation.

SQL DBMS Type	NONE	The type of the DBMS package, for example, Ingres.
SQL Database Name	NONE	The name of the SQL database from which you want to access data.
SQL Maximum Server Count	1	The number of servers allowed.
User Name		The name of the user to access the SQL database.
Password		The password of the user that accesses the SQL database.
Host Name	NONE	The name of the machine on which the SQL database resides.

STOR - store

Description

The STOR cell puts a copy of a character string into a buffer. The source can be either a buffer or a string of up to 31 letters, numbers, or any of the other characters that you type on a keyboard, except commas. For example, if the source is the DIGITS buffer containing the value 23 and the destination is the MAILBOX ID buffer, the STOR cell puts the number 23 into the MAILBOX ID buffer. The value in the DIGITS buffer is not changed by this procedure.

Note: If the source is a numerical value, enclose it in quotation marks.

Buffers used

The STOR cell uses the buffer specified by the Source parameter.

Buffers updated

The STOR cell updates the buffer specified by the Destination Buffer parameter.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Source (Buffer or Value)	0	A buffer name or a string of characters. The characters or the contents of the buffer are copied to the destination buffer.
Destination Buffer	DIGITS	A buffer name.

Next cell

Next cell	Explanation
Success	The data has been copied from the source to the destination.

Tables

None.

SUBS - substring buffer

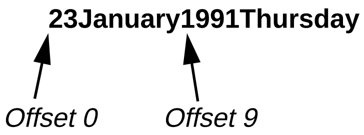
Description

The SUBS cell copies a specified portion of a string of characters and places it in a destination buffer. The source can be either a string or a buffer that contains a string. If the source is a string, enclose it in quotation marks.

The Offset parameter specifies the starting point in the source string beginning with 0. The Length parameter states the number of characters to copy. If the value of the Length parameter is 0, the range begins at the offset and continues to the end of the source string.

Example

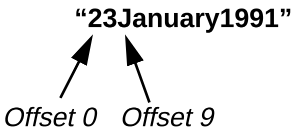
Suppose the source is a buffer containing the following string:



If the value of the Offset parameter is 9 and the value of the Length parameter is 4, the SUBS cell copies **1991** from the source buffer and places it in the destination buffer. Any value that was originally in the destination buffer is erased before the new substring is placed in it. The value in the source buffer is unchanged after the SUBS cell is executed.

Example

As another example, suppose the source is the following string:



If the value of the Offset parameter is 2 and the value of the Length parameter is 0, the SUBS cell copies **January1991** from the source buffer and places it in the destination buffer. Notice that the entire substring from the offset to the end of the string is copied because the length is 0.

Buffers used

The SUBS cell uses the buffer specified by the Offset parameter.

Buffers updated

The SUBS cell updates the buffer specified by the Destination parameter.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Source (Buffer/Value)	DIGITS	Either a string which must be enclosed in quotation marks or a buffer containing a string from which the substring is to be copied.
Destination Buffer	DIGITS	The buffer in which the copied substring is to be placed.

Offset (Buffer/Value)	0	The position of the first character to be copied. Characters in the source string are counted from left to right, beginning with zero.
Length (Buffer/Value)	0	The number of characters to be copied. This can be specified as a value or as a buffer containing the value. If the length is 0, the string is copied to the end.

Next cell

Next cell	Explanation
Fail	An error has occurred (for example, there is nothing in the source buffer).
Success	The substring function has been performed successfully.

Tables

None.

SXFR - supervised call transfer

Description

The SXFR cell is similar to the XFER (Call Transfer) cell, except that SXFR does not disconnect the caller if the transfer is unsuccessful; instead, the application takes the error branch and continues with the current caller.

Cell type

Call Management.

Buffer used

The SXFR cell uses the buffer specified by the Phone Number parameter.

Buffer updated

Buffer	Explanation
Status Code	The ACCESS return code if the transfer failed. The return codes for this cell are Code #9: MME_BUSY_DN DN is busy. Code #12: MME_CALL_FAILURE Call connection attempt has failed.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.

Phone Number	DIGITS	The buffer containing the phone number to be dialed.The phone number is passed to ACCESS directly, without any translation for asterisks or other special characters.
Status Code	DATA EXCHANGE #1	THE ACCESS return code if the transfer failed.

Next cell

Next cell	Explanation
Error	The call was not transferred successfully; the current call is still active. The status code is stored in the specified buffer.

> ***Note:*** If the transfer outdial is successful, there is no next cell. The application ends and executes the cleanup handler.

TIME - get current date and time

Description

The TIME cell gets the current date, time, and day of the week from the UNIX clock. The Current Time parameter contains the time as 24-hour clock time in the format HHMM. The Current Date parameter contains the date in the format MMDDYYYY. The Day of Week parameter contains the day of the week in numeric format where 1 is Monday and 7 is Sunday.

> ***Note:*** To get the current date from Meridian Mail, use the predefined access user function.

Buffers used

None.

Buffers updated

The TIME cell updates the buffers specified by the Current Time, Current Date, and Day of Week parameters.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Current Time	TIME	The buffer that contains the current time.
Current Date	DATE	The buffer that contains the current date.
Day of Week	DAY	The buffer that contains the day of the week.

Next cell

Next cell	Explanation
Error	The current date and time were not returned.
Success	The current date and time were returned successfully.

Tables

None.

UDLV - unschedule delivery

Description

The UDLV cell unschedules an event that was previously scheduled by a DELV cell.

When the event is scheduled, an event ID number is returned to the DELV cell and placed in a buffer. This event ID number must be entered as the Schedule Event ID parameter of the UDLV cell to indicate which event to unschedule. If necessary, you can use a Get Schedule Event List (LDLV) cell to get the event ID number again.

Buffers used

The UDLV cell uses the buffer specified by the Schedule Event ID parameter.

Buffers updated

None.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer

Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Schedule Event ID	DELIVERY EVENT ID	The buffer containing the ID number of the event to be unscheduled.

Next cell

Next cell	Explanation
Error	The delivery event ID is invalid.
Success	The event has been unscheduled.

Tables

None.

UPDT - update information database

Description

The UPDT cell allows data to be inserted into an informational database. If the record specified is found in the specified information database, up to ten buffers of data can be inserted into the record.

You must specify a buffer with valid contents for each buffer held in the record being updated.

Buffers used

Record Number.

Buffers updated

None.

Parameters

Parameters	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Database Name	<None>	The name of the information data base to be updated.
Record Number	DIGITS	The buffer that contains the storage location number to be updated in an information database. This can be a system buffer or a user-defined buffer.
Buffer count	0	How many buffers of information to insert into a database record. The prompt numbers must be listed in the Input Buffer Table.

Next cell

Next cell	Explanation
Not found	There was no record in the information database for the specified record number.
Found	The data was inserted into the specified record number.

Tables

Tables	Explanation
Input Buffer Table	The name of the buffers that contain the data to be inserted into the information database.

USER - connect user function

Description

The USER cell type is designed to be used by experienced computer programmers. This cell interfaces the application with a user function which is customized code that has been written in the “C” programming language. Before you use this cell type, refer to Chapter 5, “Creating user functions”.

A user function is a collection of individual user-defined functions bundled together. You access a specific user-defined function when calling the user function. For this reason, the Connect User Function cell type has a parameter for a Function Code. The user function calls only the user-defined function that is indicated by the Function Code parameter.

When you create a cell with the Connect User Function cell type, you can designate a set of input buffers to pass the user function any information it needs. You can also designate a set of output buffers so that the user function can return information to the application once the function has finished. As with any Meridian IVR buffer, each of these input and output buffers can be used to store a string of up to 31 characters which can be any ASCII character, except commas.

- If the status code is Success, the next cell depends on the reply code. The USER cell checks the Reply Code Table to find the next cell corresponding to the reply code.
- If the status code is Failure, the application branches to the Fail next cell.
- If the status code is time-out, the application branches to the time-out next cell.

  Note: Do not use more than 13 USER cells in an application. Exceeding this limit may cause unexpected results when you run the application.

Buffers used

The USER cell uses the input buffers listed in the Input Buffer Table.

Parameters

Parameter	Initial value	Explanation
Call Audit Enabled?	No	Determines if this cell logs the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment field Contents of the Call Audit Information buffer

Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
User Function Name	<NONE>	The name of the user function to be interfaced with the application.
Function Code (0–99)	1	The individual user-defined function to be executed this time. To create your own user functions, refer to Chapter 3, “Creating user functions.”
Time-out (0–75 secs)	15	Maximum time in seconds for the user function to perform its task. If this value is exceeded, a time-out occurs and a time-out status is returned.
Reply Codes	None	Number of valid reply codes to be returned after execution of the user function. You must list the reply codes in the Reply Code Table.
Input Buffers (max 10)	0	Number of buffers the user function passes back to the application. This value is updated automatically once the cell is saved with the count of the number of buffers.
Output Buffers (max 10)	0	Number of buffers the user function passes back to the application. This value is updated automatically once the cell is saved with the count of the number of buffers entered in the Buffer Count Table.

Next cell

> ***Note:*** If the user function returns a Success status code, the next cell comes from the Reply Code Table.

Next cell	Explanation
Time-out	A time-out status code has been returned.
Fail	A failure status code has been returned.

Tables

Table	Explanation
Input Buffer Table	Lists the name of each input buffer.
Output Buffer Table	Lists the name of each output buffer.

USER (access) - get system date

Description

This predefined user function called “access” is included with the Meridian IVR system. Get System Date is invoked through the USER cell type by specifying “access” as the user function name and 1 as the function code.

Get System Date, referencing ACCESS API m_GetSysDate (), returns the date and time as set in the Meridian Mail machine. The returned date and time are those of the general system clock of the Meridian Mail machine. The returned date is the time stamp on all files, and the reference point for delayed message delivery.

> ***Notes:***
>
> ***1*** You can also use the MDTE cell to get the system date and time.
>
> ***2*** Do not use more than 13 USER cells in an application. Exceeding this limit may cause unexpected results when you run the application.

Inputs

None.

Outputs

Output	Content	Examples
Buffer 1	ACCESS return code	
Buffer 2	Year	1994
Buffer 3	Month	Range from 1 to 12
Buffer 4	Day	Range from 1 to 31
Buffer 5	Hour	Range from 0 to 23
Buffer 6	Minute	Range from 0 to 59
Buffer 7	Second	Range from 0 to 59
Buffer 8	Date	Format mmddyyyy
Buffer 9	Time	Format hhmm

Reply codes

Reply Codes	Meaning
Reply code 0	Success
Reply code 1	Failure

Notes

Buffers 8 and 9 are in the proper format required for playing date and time in the PDAT cell.

USER (access) - get call info

Description

This predefined user function called “access” is included with the Meridian IVR system. Get Call Info is invoked through the USER cell type by specifying “access” as the user function name and 2 as the function code.

Get Call Info, referencing ACCESS API m_GetCallInfo (), is for an application that requires detailed information on either incoming or outgoing calls. For example, an application that successfully executes a COUT cell can call this function to determine the address of the agent that answered the call in a call center environment.

Notes:

1 You can also use the INFO cell to get call information.

2 Do not use more than 13 USER cells in an application. Exceeding this limit may cause unexpected results when you run the application.

Inputs

None.

Outputs

Output	Content	Examples	Type of output
Buffer 1	ACCESS return code		Character string
Buffer 2	Call State	Last CallProgress State Change	Short integer
Buffer 3	Call InfoLast	CallInfo State Change	Short integer
Buffer 4	Calling DN	DN of caller	Character string
Buffer 5	Calling DN Type	DN Type of caller	Character string
Buffer 6	Called DN	DN dialed by caller	Character string

Buffer 7	Called DN Type	DN Type of called DN	Character string
Buffer 8	Called TN	Packed TN	Character string
Buffer 9	Call Type	Type of call	Character string
Buffer 10	Device Type	Device type	Character string

Note: The values that go into buffers 2 and 3 are listed in tables under the section “USER (access) - call progress detection” on page 7-131.

Reply codes

Reply codes	Meaning
Reply code 0	Success
Reply code 1	Failure

Notes

The m_GetCallInfo() API has been implemented primarily to support Coordinated Screen Transfer.

Following is a scenario for how you can implement this and steps you can follow to implement Coordinated Screen Transfer.

- Use the COUT cell to call an agent.
- Use the USER cell with the function name “access” and a function code of 2.
- Use a CHEK cell using the packed TN returned in buffer 8 as the record number to look up the agent to which this packed DN refers. Retrieve agent data from the database.
- Pass the agent data onto the host (through a customized user function or the predefined TRS user function) which causes the information screen to be transferred to the agent who was just called.
- Use CEND to conference the agent with the caller.
- Use the HANG cell to take the IVR system out of the call.

This scenario requires you to set up an information database that matches packed TNs to a video display terminal so that the host computer can identify the terminal to receive the screen.

This scenario also assumes that the IVR application has already collected information from the caller and is currently accessing the host screen which must be sent to the agent. Your approach can vary.

USER (access) - add box to address

Description

This predefined user function called "access" is included with the Meridian IVR system. Add Box To Address is invoked through the USER cell type by specifying "access" as the user function name and 3 as the function code.

Add Box To Address, referencing ACCESS API m_AddBoxToAddr (), enables a message to be addressed by mailbox numbers. The address's full name is returned. This function should be called after an RMSG has been successfully completed and before a Send Message has been issued.

Notes:

1 You can use the ADDR message cell to address a message by a mailbox number.

2 Do not use more than 13 USER cells in an application. Exceeding this limit may cause unexpected results when you run the application.

Inputs

Input	Content
Buffer 1	Number of message to be addressed
Buffer 2	Destination Mailbox
Buffer 3	Destination Mailbox (optional)
Buffer 4	Destination Mailbox (optional)
Buffer 5	Destination Mailbox (optional)
Buffer 6	Destination Mailbox (optional)
Buffer 7	Destination Mailbox (optional)
Buffer 8	Destination Mailbox (optional)
Buffer 9	Destination Mailbox (optional)
Buffer 10	Destination Mailbox (optional)

Outputs

Output	Content
Buffer 1	ACCESS return code
Buffer 2	Full name of owner of mailbox (or access error code)
Buffer 3	Full name of owner of mailbox (or access error code)
Buffer 4	Full name of owner of mailbox (or access error code)
Buffer 5	Full name of owner of mailbox (or access error code)
Buffer 6	Full name of owner of mailbox (or access error code)
Buffer 7	Full name of owner of mailbox (or access error code)

Output	Content
Buffer 8	Full name of owner of mailbox (or access error code)
Buffer 9	Full name of owner of mailbox (or access error code)
Buffer 10	Full name of owner of mailbox (or access error code)

Reply codes

Reply codes	Meaning
Reply code 0	Success
Reply code 1	Failure

Notes

A call to this user function with this function code addresses the message specified in buffer 1 to all the mailboxes specified in buffers 2 to 10. You can specify anywhere from one to nine destination mailboxes.

Upon completion of the user function, the first 31 characters of the full name of the addressee are contained if the addressing option for the particular mailbox was successful; otherwise, the access error code can be found in its place.

The user function attempts to address the message to all destination mailboxes independent of how many of the mailboxes cannot be found.

USER (access) - send message

Description

This predefined user function called “access” is included with the Meridian IVR system. Send Message is invoked through the USER cell type by specifying “access” as the user function name and 4 as the function code.

Send Message, referencing ACCESS API m_SendMsg (), sends a message that has been previously recorded and addressed.

Notes:

1 You can also use the SMSG cell to send a message.

2 Do not use more than 13 USER cells in an application. Exceeding this limit may cause unexpected results when you run the application.

Inputs

Input	Content
Buffer 1	Number of message to be sent
Buffer 2	Priority

Priority*	Description
1	Normal delivery: specified if neither Urgent or Economy apply.
2	Urgent (high priority) delivery across network.
8	Messages held for delivery across a network.

Buffer 3 Tags	Refer to the following table for the description of the tags.

Tag*	Description
0	No special tags.

4	Sender receives an acknowledgment message when the message is first opened by each recipient.
16	Indicates that the contents are confidential; the message cannot be forwarded.

Note: * Since these ACCESS APIs are called from within a USER cell, no input error checking is performed. No error is detected if Priority or Tag contain invalid names.

Outputs

Output	**Content**
Buffer 1	ACCESS return code

Reply codes

Reply codes	**Meaning**
Reply code 0	Success
Reply code 1	Failure

Note: Once a message is sent, it can not be resent or addressed differently. It can only be played and deleted.

USER (access) - call progress detection

This predefined user function called “access” is included with the Meridian IVR system. Call Progress Detection is invoked through the USER cell type by specifying “access” as the user function name and 5 as the function code.

Call Progress Detection, referencing ACCESS API Call Progress Detection, obtains the termination state of the previous out-dial attempted by COUT or DOUT cells. This function, unlike the other function returns, has more reply codes.

Notes:

1 You can use the PRGS cell to obtain the termination state of the previous out-dial attempts.

2 Do not use more than 13 USER cells in an application. Exceeding this limit may cause unexpected results when you run the application.

Inputs

None.

Outputs

Output	Content
Buffer 1	ACCESS return code
Buffer 2	Call State (last Call Progress State Change)

Note: The numerical values listed in the tables below are the values that go into the buffer.

Call State (Buffer 2)	Description
1	Call is established or answered.
3	Call party has rung.
8	Called party is busy.
9	Call is rejected.

11	Call connection attempt failed.
12	Call resulted in collision.
13	Call/Transfer/Conference/Re-connect is successful.
999	Set has gone on-hook.
998	Called/calling DN changed.

Buffer 3	Call Info (last CallInfo State Change)

Call Info (Buffer 3)	Description
0	No additional information is available.

Call state information

Call state=3

Call Info values	Description
1	Waiting for ACD queue.
2	Waiting in the attendant queue.
5	Waiting in the ESN queue.
6	Idle ACD agent found, being rung.
3	A trunk has been seized.
4	The call is parked.

Call state=9

Call Info values	Description
8	Call blocked due to no resources.
9	Access restricted.

6	Idle ACD agent found, being rung.
3	A trunk has been seized.
4	The call is parked.

Call state=11

Call Info values	Description
10	Bad originating DN.
11	Bad called DN.
13	Incomplete originating DN.
14	Incomplete called DN.
12	Internal switch error.
15	Originating party is busy.
16	Originating party is maintenance busy.
19	Another user is using the DN.
17	500/2500 set is on hook.

Call state=11

Call Info values	Description
18	Originating party disconnected.
20	Invalid TN.
21	Incorrect customer number.
22	Initialization of transfer failed.
25	Completion of transfer failed.
24	Conference failed.
63	Internal error.

Call State=12

Call Info values	Description
61	Collision with the same service.
62	Collision with a different service.

Reply codes

Reply code 1	Answered
Reply code 0	Not Answered

XFER - transfer outdial

Description

The XFER cell dials out a phone call on behalf of a caller who already has a call in progress on a channel. Meridian IVR dials a second party's phone number, transfers the original caller to the new call, then releases the call. If the transfer is successful, the application ends. If the transfer cannot be performed, the application branches to the Error next cell and continues to process the original call with no special recovery required.

The XFER cell, in contrast to Continuation Outdial (COUT), makes no attempt to ensure that the second party accepts the call. The XFER cell releases the call as soon as it dials the digits, and before it detects ringing on the second party's phone number.

Buffers used

The XFER cell uses the buffer specified by the Phone Number parameter.

Buffers updated

None.

Parameters

Parameter	Initial Value	Explanation
Call Audit Enabled?	No	Determines if this cell will log the following information to the call audit statistics file (audit_stat.d): Application Name Cell Name Cell Number Date and Time of Cell Execution Contents of the Cell Comment Field Contents of the Call Audit Information buffer
Call Audit Information	DIGITS	When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file.
Phone #	DIGITS	The buffer containing the phone number to be dialed.

Next cell

Next cell	Explanation
Error	The call was not transferred successfully; the current call is still active. The status code is stored in the specified buffer.

Note: If the transfer outdial is successful, there is no next cell and the application ends.

Tables

None.

Appendix A: Standard prompts

The following are two lists of standard Meridian IVR prompts. These prompts are likely to be useful in any kind of application. Recordings of these prompts are included with Meridian IVR.

Table A-1
Standard English prompts

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
0	zero	0000	0	zero
1	one	0000	1	one
2	two	0000	2	two
3	three	0000	3	three
4	four	0000	4	four
5	five	0000	5	five
6	six	0000	6	six
7	seven	0000	7	seven
8	eight	0000	8	eight
9	nine	0000	9	nine
10	ten	0000	10	ten
11	eleven	0000	11	eleven
12	twelve	0000	12	twelve
13	thirteen	0000	13	thirteen

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
14	fourteen	0000	14	fourteen
15	fifteen	0000	15	fifteen
16	sixteen	0000	16	sixteen
17	seventeen	0000	17	seventeen
18	eighteen	0000	18	eighteen
19	nineteen	0000	19	nineteen
20	twenty	0000	20	twenty
21	thirty	0000	21	thirty
22	forty	0000	22	forty
23	fifty	0000	23	fifty
24	sixty	0000	24	sixty
25	seventy	0000	25	seventy
26	eighty	0000	26	eighty
27	ninety	0000	27	ninety
28	hundred	0000	28	hundred
29	thousand	0000	29	thousand
30	oh	0000	30	oh
31	star	0000	31	star
32	pound	0000	32	pound
33	January	0000	33	January
34	February	0000	34	February
35	March	0000	35	March
36	April	0000	36	April

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
37	May	0000	37	May
38	June	0000	38	June
39	July	0000	39	July
40	August	0000	40	August
41	September	0000	41	September
42	October	0000	42	October
43	November	0000	43	November
44	December	0000	44	December
45	on	0000	45	on
46	at	0000	46	at
47	a.m.	0000	47	a.m.
48	p.m.	0000	48	p.m.
49	The system storage is nearing capacity	0000	49	Error_1
50	The system storage is at capacity	0000	50	Error_2
51	We were unable to record this message.	0000	51	Error_3
52	You cannot record any new message at this time.	0000	52	Error_4
53	Welcome to...	0000	53	Welcome
54	Hello	0000	54	Hello
55	Thank you. Good-bye.	0000	55	Thank_you_
56	That is an invalid response.	0000	56	Error_5
57	There are...	0000	57	There_are
58	The number of messages waiting is...	0000	58	Number_Msg_Wtng

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
59	At the tone, leave your message. When finished, press...	0000	59	At_Tone
60	Your message has been delivered.	0000	60	Msg_Delivered
61	We were unable to deliver your message.	0000	61	Error_6
62	No new messages may be recorded at this time.	0000	62	Error_7
63	There are no more messages waiting.	0000	63	No_new_Msgs
64	The operator will be with you momentarily.	0000	64	Operator
65	You have paused recording. To resume, press...	0000	65	Pause_Recording
66	To save, press 1. To review, press 2. To delete, press 3.	0000	66	To_Save
67	To rerecord, press 1. To continue, press 2.	0000	67	To_Re-Record
68	To review, press 1. To rerecord, press 2. To continue, press 3.	0000	68	To_Review
69	Please enter the system password.	0000	69	Enter_Sys_Paswd
70	This message was sent on	0000	70	Msg_Sent
71	Please enter your password.	0000	71	Enter_Passwd
72	Please hold.	0000	72	Hold
73	That is an invalid password.	0000	73	Error_8
74	We are temporarily unable to access this message.	0000	74	Error_9
75	You have no messages waiting.	0000	75	No_Msg
76	The newly recorded prompt is...	0000	76	New_Rec_Prompt
77	The message has been deleted.	0000	77	Msg_Deleted
78	million	0000	78	million

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
79	billion	0000	79	billion
80	dollar	0000	80	dollar
81	dollars	0000	81	dollars
82	cent	0000	82	cent
83	cents	0000	83	cents
84	area code	0000	84	area_code
85	and	0000	85	and
86	by	0000	86	by
87	(0.2 seconds of silence)	0000	87	.2 seconds_slnc
88	(0.4 seconds of silence)	0000	88	.4 seconds_slnc
89	(0.8 seconds of silence)	0000	89	.8 seconds_slnc
90	up	0000	90	up
91	down	0000	91	down
92	half	0000	92	half
93	third	0000	93	third
94	fourth	0000	94	fourth
95	fifth	0000	95	fifth
96	sixth	0000	96	sixth
97	seventh	0000	97	seventh
98	eighth	0000	98	eighth
99	ninth	0000	99	ninth
100	tenth	0000	100	tenth
101	point	0000	101	point

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
102	points	0000	102	points
103	thirds	0000	103	thirds
104	fourths	0000	104	fourths
105	fifths	0000	105	fifths
106	sixths	0000	106	sixths
107	sevenths	0000	107	sevenths
108	eighths	0000	108	eighths
109	ninths	0000	109	ninths
110	tenths	0000	110	tenths
111	sixteenths	0000	111	sixteenths
112	thirty-seconds	0000	112	thirty-seconds
113	sixty-fourths	0000	113	sixty-fourths
114	weekdays	0000	114	weekdays
115	Resume recording at the tone.	0000	115	Resume
116	At the tone, leave your message.	0000	116	At_Tone
117	Recording has been aborted. The message will not be saved.	0000	117	Error_10
118	When finished, press...	0000	118	When_Finished
119	Your account cannot be accessed at this time.	0000	119	Error_11
120	Please call again later.	0000	120	Error_12
121	A	0000	121	A
122	B	0000	122	B
123	C	0000	123	C

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
124	D	0000	124	D
125	E	0000	125	E
126	F	0000	126	F
127	G	0000	127	G
128	H	0000	128	H
129	I	0000	129	I
130	J	0000	130	J
131	K	0000	131	K
132	L	0000	132	L
133	M	0000	133	M
134	N	0000	134	N
135	O	0000	135	O
136	P	0000	136	P
137	Q	0000	137	Q
138	R	0000	138	R
139	S	0000	139	S
140	T	0000	140	T
141	U	0000	141	U
142	V	0000	142	V
143	W	0000	143	W
144	X	0000	144	X
145	Y	0000	145	Y
146	Z	0000	146	Z

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
147	Sunday	0000	147	Sunday
148	Monday	0000	148	Monday
149	Tuesday	0000	149	Tuesday
150	Wednesday	0000	150	Wednesday
151	Thursday	0000	151	Thursday
152	Friday	0000	152	Friday
153	Saturday	0000	153	Saturday
154	twenty-one	0000	154	twenty-one
155	twenty-two	0000	155	twenty-two
156	twenty-three	0000	156	twenty-three
157	twenty-four	0000	157	twenty-four
158	twenty-five	0000	158	twenty-five
159	twenty-six	0000	159	twenty-six
160	twenty-seven	0000	160	twenty-seven
161	twenty-eight	0000	161	twenty-eight
162	twenty-nine	0000	162	twenty-nine
163	thirty-one	0000	163	thirty-one
164	thirty-two	0000	164	thirty-two
165	thirty-three	0000	165	thirty-three
166	thirty-four	0000	166	thirty-four
167	thirty-five	0000	167	thirty-five
168	thirty-six	0000	168	thirty-six
169	thirty-seven	0000	169	thirty-seven

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
170	thirty-eight	0000	170	thirty-eight
171	thirty-nine	0000	171	thirty-nine
172	forty-one	0000	172	forty-one
173	forty-two	0000	173	forty-two
174	forty-three	0000	174	forty-three
175	forty-four	0000	175	forty-four
176	forty-five	0000	176	forty-five
177	forty-six	0000	177	forty-six
178	forty-seven	0000	178	forty-seven
179	forty-eight	0000	179	forty-eight
180	forty-nine	0000	180	forty-nine
181	fifty-one	0000	181	fifty-one
182	fifty-two	0000	182	fifty-two
183	fifty-three	0000	183	fifty-three
184	fifty-four	0000	184	fifty-four
185	fifty-five	0000	185	fifty-five
186	fifty-six	0000	186	fifty-six
187	fifty-seven	0000	187	fifty-seven
188	fifty-eight	0000	188	fifty-eight
189	fifty-nine	0000	189	fifty-nine
190	sixty-one	0000	190	sixty-one
191	sixty-two	0000	191	sixty-two
192	sixty-three	0000	192	sixty-three

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
193	sixty-four	0000	193	sixty-four
194	sixty-five	0000	194	sixty-five
195	sixty-six	0000	195	sixty-six
196	sixty-seven	0000	196	sixty-seven
197	sixty-eight	0000	197	sixty-eight
198	sixty-nine	0000	198	sixty-nine
199	seventy-one	0000	199	seventy-one
200	seventy-two	0000	200	seventy-two
201	seventy-three	0000	201	seventy-three
202	seventy-four	0000	202	seventy-four
203	seventy-five	0000	203	seventy-five
204	seventy-six	0000	204	seventy-six
205	seventy-seven	0000	205	seventy-seven
206	seventy-eight	0000	206	seventy-eight
207	seventy-nine	0000	207	seventy-nine
208	eighty-one	0000	208	eighty-one
209	eighty-two	0000	209	eighty-two
210	eighty-three	0000	210	eighty-three
211	eighty-four	0000	211	eighty-four
212	eighty-five	0000	212	eighty-five
213	eighty-six	0000	213	eighty-six
214	eighty-seven	0000	214	eighty-seven
215	eighty-eight	0000	215	eighty-eight

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
216	eighty-nine	0000	216	eighty-nine
217	ninety-one	0000	217	ninety-one
218	ninety-two	0000	218	ninety-two
219	ninety-three	0000	219	ninety-three
220	ninety-four	0000	220	ninety-four
221	ninety-five	0000	221	ninety-five
222	ninety-six	0000	222	ninety-six
223	ninety-seven	0000	223	ninety-seven
224	ninety-eight	0000	224	ninety-eight
225	ninety-nine	0000	225	ninety-nine
226	hundreds	0000	226	hundreds
227	five hundred	0000	227	five-hundred
228	thousands	0000	228	thousands
229	ten thousands	0000	229	ten-thousands
230	hundred thousands	0000	230	hundred-thousands
231	ones	0000	231	ones
232	year	0000	232	year
233	day	0000	233	day
234	of	0000	234	of
235	millions	0000	235	millions
236	billions	0000	236	billions
237	minus	0000	237	minus
238	negative	0000	238	negative

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
239	first	0000	239	first
240	second	0000	240	second
241	eleventh	0000	241	eleventh
242	twelfth	0000	242	twelfth
243	thirteenth	0000	243	thirteenth
244	fourteenth	0000	244	fourteenth
245	fifteenth	0000	245	fifteenth
246	sixteenth	0000	246	sixteenth
247	seventeenth	0000	247	seventeenth
248	eighteenth	0000	248	eighteenth
249	nineteenth	0000	249	nineteenth
250	twentieth	0000	250	twentieth
251	thirtieth	0000	251	thirtieth
252	percent (“%”)	0000	252	percent
253	left square bracket (“[”)	0000	253	left_sqr_bracket
254	backslash (“\”)	0000	254	backslash
255	return	0000	255	return
256	enter	0000	256	enter
257	tab	0000	257	tab
258	backspace	0000	258	backspace
259	delete	0000	259	delete
260	space (“ ”)	0000	260	space
261	exclamation mark (“!”)	0000	261	exclamation_mark

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
262	period (“.”)	0000	262	period
263	ampersand (“&”)	0000	263	ampersand
264	apostrophe (“ ’ ”)	0000	264	apostrophe
265	plus (“+”)	0000	265	plus
266	comma (“,”)	0000	266	comma
267	dash (“-”)	0000	267	dash
268	divide (“/”)	0000	268	divide
269	colon (“:”)	0000	269	colon
270	semi-colon (“;”)	0000	270	semi-colon
271	less than (“< ”)	0000	271	less_than
272	equal (=)	0000	272	equal
273	greater than (“>”)	0000	273	greater_than
274	question mark (“?”)	0000	274	question_mark
275	open parenthesis (“(”)	0000	275	open_parenthesis
276	close parenthesis (“)”)	0000	276	close_parenthesis
277	asterisk (“*”)	0000	277	asterisk
278	caret (“^”)	0000	278	caret
279	underbar (“_”)	0000	279	underbar
280	single quote (“ ’ ”)	0000	280	single_quote
281	double quote (“ ” ”)	0000	281	double_quote
282	right square bracket (“]”)	0000	282	rght_sqr_bracket
283	tilde (“~”)	0000	283	tilde
284	pipe (“|”)	0000	284	pipe

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
285	open brace (“{”)	0000	285	open_brace
286	close brace (“}”)	0000	286	close_brace
287	dollar sign (“$”)	0000	287	dollar_sign
288	at symbol (“@”)	0000	288	at_symbol
289	Celsius	0000	289	Celsius
290	Fahrenheit	0000	290	Fahrenheit
291	degrees	0000	291	degrees
292	above	0000	292	above
293	below	0000	293	below
294	centigrade	0000	294	centigrade
295	degree	0000	295	degree
296	hour	0000	296	hour
297	minute	0000	297	minute
298	one (feminine gender)	0000	298	one_fem
299	twenty-one (feminine gender)	0000	299	twentyone_fem
300	twelve-noon	0000	300	twelve-noon
301	twelve-midnight	0000	301	twelve-midnight
302	Please enter your account number.	0000	302	Plse_Enter_Acc#
303	Please enter your personal identification number.	0000	303	Plse_Enter_idnt#
304	Please enter your employee number.	0000	304	Plse_Enter_Empl#
305	The balance of your account is...	0000	305	Balance_is...
306	Please wait while we retrieve your account information.	0000	306	Plse_wait

Table A-1
Standard English prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
307	If you are calling from a touch-tone phone, press 1. Otherwise, please wait on the line and your call will be transferred.	0000	307	If_calng_touch
308	That entry could not be found.	0000	308	Error_13
309	Number-sign	0000	309	number_sign
310	Thank you.	0000	310	Thank_you
311	Good-bye.	0000	311	Good-bye
312	Thank you for calling.	0000	312	Thank_you_calling
313	Not a valid selection.	0000	313	Not_Valid_Slctn
314	Please wait for an operator to take your call.	0000	314	Please_Wait_Oprt
315	You have entered...	0000	315	You_Entered
316	Transferring to an attendant.	0000	316	Transferring_To
317	One moment please.	0000	317	One_Moment
318	If this is correct, press 1. If not, press 2.	0000	318	If_Correct_Press
319	...followed by...	0000	319	Followed_By
320	We're sorry. Your account cannot be accessed at this time.	0000	320	We_Sorry
321	Please try again later.	0000	321	Please_Try_Later

Table A-2
Standard French prompts

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
0	zéro	1000	0	zero
1	un	1000	1	one
2	deux	1000	2	two
3	trois	1000	3	three
4	quatre	1000	4	four
5	cinq	1000	5	five
6	six	1000	6	six
7	sept	1000	7	seven
8	huit	1000	8	eight
9	neuf	1000	9	nine
10	dix	1000	10	ten
11	onze	1000	11	eleven
12	douze	1000	12	twelve
13	treize	1000	13	thirteen
14	quatorze	1000	14	fourteen
15	quinze	1000	15	fifteen
16	seize	1000	16	sixteen
17	dix-sept	1000	17	seventeen
18	dix-huit	1000	18	eighteen
19	dix-neuf	1000	19	nineteen
20	vingt	1000	20	twenty
21	trente	1000	21	thirty

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
22	quarante	1000	22	forty
23	cinquante	1000	23	fifty
24	soixante	1000	24	sixty
25	soixante-dix	1000	25	seventy
26	quatre-vingt	1000	26	eighty
27	quatre-vingt-dix	1000	27	ninety
28	cent	1000	28	hundred
29	mille	1000	29	thousand
30	zéro	1000	30	oh
31	étoile	1000	31	star
32	carré	1000	32	pound
33	janvier	1000	33	January
34	février	1000	34	February
35	mars	1000	35	March
36	avril	1000	36	April
37	mai	1000	37	May
38	juin	1000	38	June
39	juillet	1000	39	July
40	août	1000	40	August
41	septembre	1000	41	September
42	octobre	1000	42	October
43	novembre	1000	43	November

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
44	décembre	1000	44	December
45	le	1000	45	on
46	à	1000	46	at
47	(24-hour clock used in French)	1000	47	a.m.
48	(24-hour clock used in French)	1000	48	p.m.
49	La mémoire du système est presque saturée.	1000	49	Error_1
50	La mémoire du système est saturée.	1000	50	Error_2
51	Il a été impossible d'enregistrer ce message.	1000	51	Error_3
52	Vous ne pouvez enregistrer de nouveau message en ce moment.	1000	52	Error_4
53	Ici …	1000	53	Welcome
54	Bonjour	1000	54	Hello
55	Merci et au revoir.	1000	55	Thank_you_
56	Cette réponse n'est pas valide.	1000	56	Error_5
57	Il y a …	1000	57	There_are
58	Nombre de messages en attente:	1000	58	Number_Msg_Wtn g
59	À la tonalité, dictez votre message. Lorsque vous aurez terminé, faites le …	1000	59	At_Tone
60	Votre message a été remis.	1000	60	Msg_Delivered
61	Il a été impossible de remettre ce message.	1000	61	Error_6
62	Il est impossible d'enregistrer de nouveaux messages en ce moment.	1000	62	Error_7

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
63	Il n’y a plus de messages en attente.	1000	63	No_new_Msgs
64	Le téléphoniste vous répondra dans un instant.	1000	64	Operator
65	Vous avez fait une pause pendant l’enregistrement. Pour continuer, faites...	1000	65	Pause_Recording
66	Pour sauvegarder le message, faites le 1. Pour en revoir le contenu, faites le 2. Pour l’effacer, faites le 3.	1000	66	To_Save
67	Pour enregistrer le message de nouveau, faites le 1. Pour continuer, faites le 2.	1000	67	To_Re-Record
68	Pour revoir le contenu du message, faites le 1. Pour l’enregistrer de nouveau, faites le 2. Pour continuer, faites le 3.	1000	68	To_Review
69	Veuillez entrer le mot de passe du système.	1000	69	Enter_Sys_Paswd
70	Ce message a été envoyé le...	1000	70	Msg_Sent
71	Veuillez entrer votre mot de passe.	1000	71	Enter_Passwd
72	Veuillez ne pas quitter.	1000	72	Hold
73	Ce mot de passe n’est pas valide.	1000	73	Error_8
74	Il est temporairement impossible d'accéder à ce message.	1000	74	Error_9
75	Vous n’avez pas de message en attente.	1000	75	No_Msg
76	Le nouveau guide vocal enregistré est…	1000	76	New_Rec_Prompt
77	Le message est effacé.	1000	77	Msg_Deleted
78	million	1000	78	million
79	milliard	1000	79	billion
80	dollar	1000	80	dollar

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
81	dollars	1000	81	dollars
82	cent	1000	82	cent
83	cents	1000	83	cents
84	indicatif régional	1000	84	area_code
85	et	1000	85	and
86	par	1000	86	by
87	(0,2 secondes de silence)	1000	87	.2 seconds_slnc
88	(0,4 secondes de silence)	1000	88	.4 seconds_slnc
89	(0,8 secondes de silence)	1000	89	.8 seconds_slnc
90	en hausse de	1000	90	up
91	en baisse de	1000	91	down
92	demi	1000	92	half
93	troisième	1000	93	third
94	quatrième	1000	94	fourth
95	cinquième	1000	95	fifth
96	sixième	1000	96	sixth
97	septième	1000	97	seventh
98	huitième	1000	98	eighth
99	neuvième	1000	99	ninth
100	dixième	1000	100	tenth
101	virgule	1000	101	point
102	virgules	1000	102	points

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
103	tiers	1000	103	thirds
104	quarts	1000	104	fourths
105	cinquièmes	1000	105	fifths
106	sixièmes	1000	106	sixths
107	septièmes	1000	107	sevenths
108	huitièmes	1000	108	eighths
109	neuvièmes	1000	109	ninths
110	dixièmes	1000	110	tenths
111	seizièmes	1000	111	sixteenths
112	trente-deuxièmes	1000	112	thirty-second
113	soixante-quatrièmes	1000	113	sixty-fourths
114	jours ouvrables	1000	114	weekdays
115	À la tonalité, continuez l'enregistrement.	1000	115	Resume
116	À la tonalité, quittez le message.	1000	116	At_Tone
117	L'enregistrement a été abandonné. Le message ne sera pas sauvegardé.	1000	117	Error_10
118	Lorsque vous avez terminé, faites…	1000	118	When_Finished
119	Il est impossible d’accéder à votre compte pour le moment.	1000	119	Error_11
120	Veuillez rappeler plus tard.	1000	120	Error_12
121	A	1000	121	A
122	B	1000	122	B
123	C	1000	123	C

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
124	D	1000	124	D
125	E	1000	125	E
126	F	1000	126	F
127	G	1000	127	G
128	H	1000	128	H
129	I	1000	129	I
130	J	1000	130	J
131	K	1000	131	K
132	L	1000	132	L
133	M	1000	133	M
134	N	1000	134	N
135	O	1000	135	O
136	P	1000	136	P
137	Q	1000	137	Q
138	R	1000	138	R
139	S	1000	139	S
140	T	1000	140	T
141	U	1000	141	U
142	V	1000	142	V
143	W	1000	143	W
144	X	1000	144	X
145	Y	1000	145	Y

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
146	Z	1000	146	Z
147	dimanche	1000	147	Sunday
148	lundi	1000	148	Monday
149	mardi	1000	149	Tuesday
150	mercredi	1000	150	Wednesday
151	jeudi	1000	151	Thursday
152	vendredi	1000	152	Friday
153	samedi	1000	153	Saturday
154	vingt et un	1000	154	twenty-one
155	vingt-deux	1000	155	twenty-two
156	vingt-trois	1000	156	twenty-three
157	vingt-quatre	1000	157	twenty-four
158	vingt-cinq	1000	158	twenty-five
159	vingt-six	1000	159	twenty-six
160	vingt-sept	1000	160	twenty-seven
161	vingt-huit	1000	161	twenty-eight
162	vingt-neuf	1000	162	twenty-nine
163	trente et un	1000	163	thirty-one
164	trente-deux	1000	164	thirty-two
165	trente-trois	1000	165	thirty-three
166	trente-quatre	1000	166	thirty-four
167	trente-cinq	1000	167	thirty-five

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
168	trente-six	1000	168	thirty-six
169	trente-sept	1000	169	thirty-seven
170	trente-huit	1000	170	thirty-eight
171	trente-neuf	1000	171	thirty-nine
172	quarante et un	1000	172	forty-one
173	quarante-deux	1000	173	forty-two
174	quarante-trois	1000	174	forty-three
175	quarante-quatre	1000	175	forty-four
176	quarante-cinq	1000	176	forty-five
177	quarante-six	1000	177	forty-six
178	quarante-sept	1000	178	forty-seven
179	quarante-huit	1000	179	forty-eight
180	quarante-neuf	1000	180	forty-nine
181	cinquante et un	1000	181	fifty-one
182	cinquante-deux	1000	182	fifty-two
183	cinquante-trois	1000	183	fifty-three
184	cinquante-quatre	1000	184	fifty-four
185	cinquante-cinq	1000	185	fifty-five
186	cinquante-six	1000	186	fifty-six
187	cinquante-sept	1000	187	fifty-seven
188	cinquante-huit	1000	188	fifty-eight
189	cinquante-neuf	1000	189	fifty-nine

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
190	soixante et un	1000	190	sixty-one
191	soixante-deux	1000	191	sixty-two
192	soixante-trois	1000	192	sixty-three
193	soixante-quatre	1000	193	sixty-four
194	soixante-cinq	1000	194	sixty-five
195	soixante-six	1000	195	sixty-six
196	soixante-sept	1000	196	sixty-seven
197	soixante-huit	1000	197	sixty-eight
198	soixante-neuf	1000	198	sixty-nine
199	soixante et onze	1000	199	seventy-one
200	soixante-douze	1000	200	seventy-two
201	soixante-treize	1000	201	seventy-three
202	soixante-quatorze	1000	202	seventy-four
203	soixante-quinze	1000	203	seventy-five
204	soixante-seize	1000	204	seventy-six
205	soixante-dix-sept	1000	205	seventy-seven
206	soixante-dix-huit	1000	206	seventy-eight
207	soixante-dix-neuf	1000	207	seventy-nine
208	quatre-vingt-un	1000	208	eighty-one
209	quatre-vingt-deux	1000	209	eighty-two
210	quatre-vingt-trois	1000	210	eighty-three
211	quatre-vingt-quatre	1000	211	eighty-four

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
212	quatre-vingt-cinq	1000	212	eighty-five
213	quatre-vingt-six	1000	213	eighty-six
214	quatre-vingt-sept	1000	214	eighty-seven
215	quatre-vingt-huit	1000	215	eighty-eight
216	quatre-vingt-neuf	1000	216	eighty-nine
217	quatre-vingt-onze	1000	217	ninety-one
218	quatre-vingt-douze	1000	218	ninety-two
219	quatre-vingt-treize	1000	219	ninety-three
220	quatre-vingt-quatorze	1000	220	ninety-four
221	quatre-vingt-quinze	1000	221	ninety-five
222	quatre-vingt-seize	1000	222	ninety-six
223	quatre-vingt-dix-sept	1000	223	ninety-seven
224	quatre-vingt-dix-huit	1000	224	ninety-eight
225	quatre-vingt-dix-neuf	1000	225	ninety-nine
226	centaines	1000	226	hundreds
227	cinq cent	1000	227	five-hundred
228	milliers	1000	228	thousands
229	dix mille	1000	229	ten-thousands
230	cent mille	1000	230	hundred-thousands
231	un	1000	231	ones
232	année	1000	232	year
233	jour	1000	233	day

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
234	de	1000	234	of
235	millions	1000	235	millions
236	milliards	1000	236	billions
237	moins	1000	237	minus
238	moins	1000	238	negative
239	premier	1000	239	first
240	deuxième	1000	240	second
241	onzième	1000	241	eleventh
242	douzième	1000	242	twelfth
243	treizième	1000	243	thirteenth
244	quatorzième	1000	244	fourteenth
245	quinzième	1000	245	fifteenth
246	seizième	1000	246	sixteenth
247	dix-septième	1000	247	seventeenth
248	dix-huitième	1000	248	eighteenth
249	dix-neuvième	1000	249	nineteenth
250	vingtième	1000	250	twentieth
251	trentième	1000	251	thirtieth
252	pour cent (“%”)	1000	252	percent
253	crochet d'ouverture (“[”)	1000	253	left_sqr_bracket
254	oblique inverse (“\”)	1000	254	backslash
255	retour	1000	255	return

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
256	entrée	1000	256	enter
257	tabulation	1000	257	tab
258	espace arrière	1000	258	backspace
259	effacement	1000	259	delete
260	espacement (“ ”)	1000	260	space
261	point d'exclamation (“!”)	1000	261	exclamation_mark
262	point (“.”)	1000	262	period
263	perluète (“&”)	1000	263	ampersand
264	apostrophe (“’”)	1000	264	apostrophe
265	plus (“+”)	1000	265	plus
266	virgule (“,”)	1000	266	comma
267	tiret (“-”)	1000	267	dash
268	séparation (“/”)	1000	268	divide
269	deux-points (“:”)	1000	269	colon
270	point-virgule (“;”)	1000	270	semi-colon
271	inférieur à (“<”)	1000	271	less_than
272	égal à (=)	1000	272	equal
273	supérieur à (“>”)	1000	273	greater_than
274	point d'interrogation (“?”)	1000	274	question_mark
275	parenthèse d'ouverture (“(”)	1000	275	open_parenthesis
276	parenthèse de fermeture (“)”)	1000	276	close_parenthesis
277	astérisque (“*”)	1000	277	asterisk

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
278	accent circonflexe (“^”)	1000	278	caret
279	caractère de soulignement (“_”)	1000	279	underbar
280	guillemet simple (“ ’ ”)	1000	280	single_quote
281	guillemet double (“ ” ”)	1000	281	double_quote
282	crochet de fermeture (“]”)	1000	282	rght_sqr_bracket
283	tilde (“~”)	1000	283	tilde
284	barre verticale (“|”)	1000	284	pipe
285	accolade d'ouverture (“{”)	1000	285	open_brace
286	accolade de fermeture(“}”)	1000	286	close_brace
287	signe de dollar (“$”)	1000	287	dollar_sign
288	a commercial (“@”)	1000	288	at_symbol
289	Celsius	1000	289	Celsius
290	Fahrenheit	1000	290	Fahrenheit
291	degrés	1000	291	degrees
292	(not used in French to express temperature)	1000	292	above
293	(not used in French to express temperature)	1000	293	below
294	Celsius	1000	294	centigrade
295	degré	1000	295	degree
296	heure	1000	296	hour
297	minute	1000	297	minute
298	une (feminine gender)	1000	298	one_fem
299	vingt et une (feminine gender)	1000	299	twentyone_fem

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
300	midi	1000	300	twelve-noon
301	minuit	1000	301	twelve-midnight
302	Veuillez entrer votre numéro de compte.	1000	302	Plse_Enter_Acc#
303	Veuillez entrer votre numéro d'identification personnel.	1000	303	Plse_Enter_idnt#
304	Veuillez entrer votre numéro matricule.	1000	304	Plse_Enter_Empl#
305	Le solde de votre compte est de…	1000	305	Balance_is...
306	Veuillez ne pas quitter: le système est en train de chercher l'information sur votre compte.	1000	306	Plse_wait
307	Si vous appelez à partir d'un poste Touch-Tone, faites le 1. Sinon, veuillez ne pas quitter: votre appel sera transféré.	1000	307	If_calng_touch
308	Il a été impossible de trouver cette inscription.	1000	308	Error_13
309	carré	1000	309	number_sign
310	Merci.	1000	310	Thank_you
311	Au revoir.	1000	311	Good-bye
312	Nous vous remercions de votre collaboration.	1000	312	Thank_you_calling

Table A-2
Standard French prompts (continued)

#	Meridian IVR Release 2 Prompt Script	File	Seg	VPE Prompt Name
313	Cette sélection n’est pas valide.	1000	313	Not_Valid_Slctn
314	Un téléphoniste vous répondra; veuillez ne pas quitter.	1000	314	Please_Wait_Oprt
315	Vous avez accédé à...	1000	315	You_Entered
316	Votre appel est transféré à un préposé.	1000	316	Transferring_To
317	Un moment s’il vous plaît.	1000	317	One_Moment
318	Si cette information est correcte, faites le 1. Sinon, faites le 2.	1000	318	If_Correct_Press
319	… suivi du	1000	319	Followed_By
320	Nous sommes désolés, mais il est impossible d’accéder à votre compte pour le moment.	1000	320	We_Sorry
321	Veuillez rappeler plus tard.	1000	321	Please_Try_Later
322	première (feminine gender)	1000	322	first_fem

Appendix B: Meridian ACCESS return codes

This Appendix lists all of the symbolic constants (return codes) returned by Meridian ACCESS API functions. Symbolic constants begin with one of the following prefixes as shown in Table B-1.

Table B-1
Prefixes

Prefix	Description
MMS	Status
MME	Error
MMW	Warning

Access return codes (see Table B-2) are error messages that the application passes on to the buffer if the call is unsuccessful. Once the buffer receives the error code, it tells the application how to handle the call. You can use the buffers to manage the call according to your specific needs.

Table B-2
Access return codes

Return code	Symbolic constant	Description
0	MMS_OKAY	Success
1	MME_BAD_PARAMETER	Bad parameter passed to function
2	MMS_NOT _READY	No result available yet
3	MME_TIMEOUT	No result - command timed out
4	MME_NO_LOCAL_MEMORY	Out of memory (local)
5	MME_INVALID_CLASS	Invalid application class
6	MME_NOT _ACQUIRED	Command invalid before “Acquire”
7	MME_NOT_REGISTERED	Calling process is not registered with the LH
8	MME_ALREADY _REGISTERED	Calling process is already registered with the LH
9	MME_BUSY_DN	DN is busy
10	MME_NOT_ANSWERED	No answer at DN
11	MME_CALL_FAILURE	Call has been rejected
12	MME_CALL_FAILURE	Call connection attempt has failed
13	MME_CALL_COLLISION	Call resulted in collision
14	MME_OPER_TIMEOUT	Time-out performing operation
15	MME_CALL_DISCONNECTED	Call has disconnected
16	MME_NO_QUEUE_SPACE	MSG send failed: no queue space
17	MME_BAD_PROCESS_TYPE	Invalid process type
18	MME_API_QUEUE_DOWN	System error accessing API queue

Table B-2
Access return codes (continued)

Return code	Symbolic constant	Description
19	MME_EVENT_QUEUE_DOWN	System error accessing Event queue
20	MME_MONITOR_EXISTS	Monitor function already installed
21	MME_NOT_MONITOR	Client is not the monitor process
22	MME_FUNCTION_NOT_AVAIL	API not usable: wrong ACCESS version
23	MME_BAD_SEM_KEY	Could not access/open a semaphore
24	MME_BAD_PATH	No file at path specified
25	MME_FORK_ERROR	Could not fork at process path
26	MMW_ALREADY_DEAD	Link Manager was already dead
27	MME_NOT_PARENT	Did not spawn LMP via m_StartLink
28	MMW_DEAD_CHILD	Caller had dead child beside LMP
29	MME_LH_DEFUNCT	LMP took too long to die
30	MME_LH_NOT_SYNCH	LH not synchronized with MM command failed
31	MMS_LH_NOT_SYNCH	LH not synchronized with MM command succeeded
32	MMS_LH_IN_SYNCH	LH is synchronized with MM
33	MME_LH_SICK	LH returned an unexpected value
34	MME_MON_RESTRICTED	API is restricted from monitor
35	MME_NO_CONFIG	No LH configuration file found
99	MME_NOT_SUPPORTED	Operation not currently supported
102	MME_BAD_PSWD	Invalid Password
103	MME_NO_TASK	No MM ACCESS Toolkit available
104	MME_FULL_SERVER	No free blocks, server is full

Table B-2
Access return codes (continued)

Return code	Symbolic constant	Description
105	MME_FULL_CABINET	No free disk space in User Cabinet
106	MME_DO_LOGON	Must be logged on to use this cmd
109	MME_ACCESS_DENIED	Access to account denied
111	MME_COMMAN_FAILED	Command failed, check SEER console
113	MME_BAD_LOGON_TYPE	Invalid account type
115	MME_ALREADY_ACQUIRED	Already acquired
117	MME_MAX_LOGONS	Too many failed m_Logon attempts
120	MME_INVALID_FUNCTION	API function not supported
122	MME_NO_MEMORY	Out of memory
126	MME_BAD_ID	Bad userid or mailbox number
128	MME_BAD_FLAG	Invalid flag (0 or 1 are valid)
129	MMW_DUP_LOGON	Warning: Logged on elsewhere
131	MME_BAD_VERSION	API library being used not supported by Meridian Mail
133	MME_INVALID_CUST	Invalid customer number specified
134	MME_ALREADY_LOGON	Command not valid while logged in
135	MME_ENS_EXISTS	An application has already acquired ENS
136	MME_NOT _ENS	Must be an ENS app to use this command
150	MME_OPTION_NOT_AVAIL	Option not available to customer
151	MME_MAX_REQUESTS	Max. # of acquire requests reached
152	MMW_ALREADY_RELEASED	Session already released by system
200	MME_NO_ACT_CHNL	No active voice channel
203	MME_BAD_POSITION	Invalid voice start position

Table B-2
Access return codes (continued)

Return code	Symbolic constant	Description
204	MME_BAD_TO_POS	Invalid play position
205	MME_BAD_RECORD_POS	Invalid recording position
208	MME_BAD_DIRECTION	Invalid direction
211	MME_CHAN_IN_USE	Voice channel already in use
212	MME_NO_ACQUIRED_CHNL	No voice channel has been acquired
213	MME_NO_INC_CALL	No incoming call to answer
214	MME_DO_ADDONCALL	Must call m_AddOnCall first
215	MME_CHANNEL_READY	m_AcceptCall (already) issued
217	MME_OTHER_TELEPHONY	Other telephony command in progress
223	MME_PLAYING	Play command already in progress
224	MME_BAD_SEQUENCE	Invalid command sequence
225	MME_RECORDING	Record command already in progress
227	MME_VOICE_FAILURE	Voice operation failure
228	MMS_NO_VOICE	No voice segment to play
229	MMS_AT_EOS	At end of voice segment
231	MME_SILENCE_TIMEOUT	Ended because too much silence
232	MME_RECORD_LIMIT	Recording limit reached
233	MME_BAD_NUM_SEGS	Bad number of segments specified
235	MME_SEG_Q_FULL	Segment play queue is full
236	MME_INVALID_DTMF	Invalid Dual Tone Multiple Frequency string
237	MME_BAD_DETECTION	Context must be SOUND/SILENCE
238	MME_BAD_DURATION	Duration must be <=5 minutes

Table B-2
Access return codes (continued)

Return code	Symbolic constant	Description
239	MME_NO_PREV_DETECT	No Previous Detect in progress
240	MME_DETECT_INPROG	Sound Detect already in progress
250	MME_INSTL_EVENT	Must install event handler first
309	MME_NO ENTRY_FOUND	No such entry found in directory
400	MME_CABINET	Unable to access user's cabinet
401	MME_INVALID_HANDLE	Invalid file handle passed to command
402	MME_BAD_HANDLE	Unassigned file handle
403	MME_BAD_COMMIT	Invalid commit flag (parameter)
405	MMS_AT_BOF	Reached the beginning of file
406	MME_READ_MODE	Cannot open Read file in Write mode
407	MMS_AT_EOF	Reached the end of file
409	MME_FILE_OPEN	File is already open
410	MMW_COMMIT_IGNORED	Read-only file: Not committed
411	MME_READ_ONLY	Cannot do command on Read-only file
415	MME_FNAME_FORMAT	Invalid filename format
416	MME_MAX_OPEN	Maximum open file limit reached
419	MME_DO_FILEPAT	Must call m_FilePattern first
420	MME_FILE_DNE	File does not exist
425	MME_BAD_NEW_FLAG	Invalid new flag passed
426	MME_BAD_MODE	Invalid file access mode used
431	MME_BAD_IMMED	Invalid delete parameter
432	MME_BAD_COMMAND	Command invalid on this file type

Table B-2
Access return codes (continued)

Return code	Symbolic constant	Description
433	MME_BAD_SEG_ID	Segment ID not found in file
434	MME_TITLE_LENGTH	Invalid length in field
436	MME_DO_SEGPAT	Must call m_SEGPattern first
437	MME_SCRIPT_LENGTH	Invalid script length
438	MME_SCRIPT_LENGTH	Invalid script length
439	MME_NO_SEGS	No voice segments in the file
441	MME_MAX_SEG_FILES	Too many open segment files for play
442	MME_MAX_SCRIPT_SIZE	Script for voice segment too long
444	MME_MAC_SEGS	Reached max # segs allowed in file
445	MME_BAD_SEG_TYPE	Bad voice segment file type
446	MME_BAD_LANGUAGE	Invalid language specified
448	MME_BAD_EDIT_POS	Invalid segment editing position
449	MME_BAD_OPERATOR	Invalid segment editing operator
450	MME_BAD_AMOUNT	Invalid amount specified
500	MME_FILE_NOT_MSG	File is not a message file
508	MME_BAD_RCVR	Invalid receiver in address list
509	MME_MAX_RCVRS	Exceeded maximum # of message recipients
511	MME_BAD_SUBJECT	Invalid subject string
512	MME_EMPTY_MSG	Cannot send an empty message
513	MME_NOT_RECEIVED	CallSender/Reply only on received messages
515	MME_DO_ADDRPAT	Must call m_AddrPattern first

Table B-2
Access return codes (continued)

Return code	Symbolic constant	Description
519	MME_EXTERNAL	Cannot reply to external messages
520	MME_FORWARDED_PRIVATE	Cannot forward a private message
522	MME_NEED_RCVR	Need 1 or more receivers to send
523	MME_MULTIMATCH	Multiple names matched, specify
524	MME_INCOMING	Cannot be used on this message type
525	MME_MAX_DELAY	Delay delivery time too long
526	MME_REMOTE	Remote site not recognized
527	MME_SYS_MSG	Operations invalid on system messages
528	MME_BROADCAST	Cannot ReplyAll to Broadcast messages
529	MME_AMIS_REPLY	Cannot reply all on AMIS message
600	MME_PDL_DNE	List number not found
601	MME_BAD_PDL_NUM	Invalid PDL list number
602	MME_MAX_PDL_ENTRIES	Exceeded number of entries in PDL
603	MME_USER_PROFILE	Unable to access user profile
622	MME_RESTRICTED	Restricted to admin access only
623	MME_BAD_BOX	Invalid box number
625	MME_BAD_BOX_SURNAME	Invalid last name
626	MME_BAD_GIVEN	Invalid first name
627	MME_BAD_LIST	Invalid list number
628	MME_PSWD_TOO_SHORT	Password too short
629	MME_BAD_GREET	Invalid personal greeting type
630	MME_DUP_OLD	Old password and logged on elsewhere

Table B-2
Access return codes (continued)

Return code	Symbolic constant	Description
631	MME_PSWD_OLD	Old passwords cannot be reused
632	MME_OPEN_PERS_VERIF	Personal verification already open
633	MME_OPEN_GREETING	Greeting already open
634	MME_NOT_NUMERIC	Non-numeric in numeric field
636	MME_NO_MATCHING_BOX	No matching box address in PDL
637	MME_DO_PDLPAT	Must call m_PDLPattern first
638	MME_NOT_PDL	Not a PDL file
639	MME_BAD_MSG_TYPE	Invalid external message type
700	MME_API_NOT_INIT	Set HiLev flag before invoking Application Programming Interface (API)
701	MME_BAD_EXIT_DIGIT	Invalid digit in ExitDigits
702	MME_INTER_KEY_TO	Inter Digit Time-out occurred
703	MME_KEY_OVERFLOW	Key buffer overflow occurred
633	MME_OPEN_GREETING	Greeting already open
634	MME_NOT_NUMERIC	Non-numeric in numeric field
636	MME_NO_MATCHING_BOX	No matching box address in PDL
637	MME_DO_PDLPAT	Must call m_PDLPattern first
638	MME_NOT_PDL	Not a PDL file
704	MME_API_INTERRUPTED	API interrupted by Meridian Mail event

Table B-2
Access return codes (continued)

Return code	Symbolic constant	Description
705	MME_BAD_ITEMTOPLAY	ItemToPlay in Invalid format
706	MME_BAD_PLAYTYPE	Invalid PlayType specified
707	MME_PLAY_TIMEOUT	PlayEnd event not received
806	MME_BAD_DN	Invalid Directory Number passed
808	MME_BAD_ANSWER	Invalid answer flag
811	MME_RESTRICTED_DN	DN has a restricted prefix
900	MME_LH_TABLE_FULL	Link Handler Register Table full
910	MME_TRANS_TABLE_FULL	Link Handler Trans Table full
1000	MME_ECHO_FAIL	Echo test failed: corrupted string
1005	MME_AUTOEVENTION	m_EventCheck with autoevention

Appendix C: Sample cell chart

The following page contains a blank cell chart. If you are creating an application and want to plan the application on paper, you will find it convenient to fill out a cell chart for each cell as shown in Table C-1.

On each cell chart, you can do the following:

- Name the cell. It is helpful to choose a name that reflects what the cell does.
- Choose a cell type.
- Write a brief description of what the cell does.
- Make a list of the prompt numbers you use.
- Find out what buffers are used or updated, and add them to your chart.
- List the parameter values you use for this cell type. In most cases, the default value is the one you want.
- Make a list of the next cell.
- List the tables, if any, for this cell and the values that go into each.

Table C-1
A blank cell chart

Cell Chart
Application Name
Cell Name
Cell Type
Description
Prompts
Buffers
PARAMETERS
NEXT CELLS
TABLES

Appendix D: Converting applications

The file transfer program (FTP) converts applications from Meridian IVR Release 1 to Meridian IVR Release 2 in the UNIX shell.

To convert applications, you need to follow three procedures:

1 Transfer the application from Meridian IVR Release 1 to a host computer.

2 Transfer the application from the host computer to Meridian IVR Release 2.

3 Convert the application on the Meridian IVR Release 2 system.

Note: Both Meridian IVR Release 1 and Release 2 systems must be accessible from the network.

Procedure D-1
Transferring the application from Release 1

1 Type the ftp command followed by the Internet Protocol (IP) address of the Release 1 system. For example

ftp 11.222.3.44

The system indicates that you are connected to the IP address of the Release 1 system.

2 Log onto the Release 1 system by typing your user ID and password.

The system displays the following messages:

```
User root logged in

Remote system type is UNIX

Using binary mode to transfer files
```

3 Change from the root directory to the application directory by typing

cd /user/ivr/gen/apps

The screen displays the following message:

```
 CWD command successful
```

4 Set file type to binary by typing

binary

The system displays the following message:

```
Type set to I
```

5 Get the application you want to convert by typing

get application_name.vpf

where .vpf is the extension for application names.

The system displays the following message:

```
PORT command successful

Opening BINARY mode data connection for...

Transfer complete.
```

The system also indicates how many bytes it retrieved in the transfer.

6 Exit the file transfer program by typing

quit

The system returns to the UNIX prompt.

Procedure D-2
Transferring the application to Release 2

1 Type the ftp command followed by the Internet Protocol (IP) address of the Release 2 system. For example

ftp 66.777.8.99

The system indicates that you are connected to the IP address of the Release 2 system.

2 Log onto the Release 2 system by typing your user ID and password.

The system displays the following messages:

```
User root logged in
Remote system type is UNIX
Using binary mode to transfer files
```

3 Change from the root directory to the application directory by typing

cd /u/ivr/apps

The screen displays the following message:

```
CWD command successful
```

4 Set file type to binary by typing

binary

The system displays the following message:

```
Type set to I
```

5 Transfer the file to the Release 2 system by typing

put *application_name*.vpf

where .vpf is the extension for application names.

The system displays the following messages:

```
PORT command successful
Opening BINARY mode data connection...
Transfer complete.
```

The system also indicates how many bytes it sent in the transfer.

6 Exit the file transfer program by typing

quit

The system returns to the UNIX prompt.

Procedure D-3
Converting the application (one-step procedure)

1 Type the following in a UNIX shell window on the Release 2 system:

/u/ivr/exe/appconvert /u/ivr/apps/*application_name*.vpf

The system returns to the UNIX prompt. You can now open the application with the Application Editor.

Appendix E: Upgrading databases

To upgrade Release 1 sites to Release 2/I, you need to integrate customized data with the Release 2 environment. This involves three procedures:

1 converting Release 1 applications

2 porting Release 1 user functions

3 converting Release 1 databases

Converting Release 1 applications

To load and run Release 1 applications on the Release 2/I system, you must first convert the applications.

The following details the procedure to convert the Release 1 application <app_name>.

Procedure E-1
Converting Release 1 applications

1 Using ftp, transfer the file in binary form from the Release 1 directory */user/ivr/gen/apps* to the Release 2 directory */u/ivr/apps*.

2 Convert the application using the conversion utility

```
$ cd /u/ivr/exe

$ ./appconvert ../apps/<app_name>.vpf
```

This completes the application conversion. Prior to being able to load and run the application, you may have to convert user functions and databases.

Porting Release 1 User Functions

User functions created on the Release 1 system will need to be recompiled for use in the SCO environment. In addition, the user function template*usr.c* has been enhanced for Release 2. As a result, code incorporated into your Release 1 user function *c file* will need to be copied into the new template.

Further, porting of your Release 1 code may involve minor design changes to observe differences in the Motorola and SCO libraries. Finally, your user function may be compiled in the same fashion as on the Release 1 system:

```
> make -e -f usr.make NAME=<name>
```

Converting the Release 1 Database

This procedure consists of exporting the database records on the Release 1 system, and importing them on the Release 2 system.

Procedure E-2
Converting Release 1 databases

1 On the Release 1 system, select the database from the database editor and open that database.

2 Export records from the Release 1 database into a file <file_name> using the “export” menu item. This file will be created in the /user/ivr/gen/exe directory.

3 Transfer the file <file_name> as ‘ascii’ from the Release 1 system to the Release 2 system by performing a File Transfer Protocol (FTP). The file should be put in the /u/ivr/sys_files directory.

4 Change permissions, group, owner of the file on the Release 2 system:

```
chmod 777 <file_name>

chown ivr <file_name>

chgrp mivr1 <file_name>
```

5 On the Release 2 system, create a suitable named database using the database editor.

6 Select import, and specify the <file name> and INTERNAL format to import the records.

Note: No data appears immediately in the database, this is normal.

7 Save the new database you created and exit the editor.

8 Re-invoke the editor and open the new database.

At this point, all imported records are present.

Glossary

application

With respect to Meridian IVR, an application is a program that controls the activity on one or more telephone trunks connected to an AP. With respect to a host computer, it is any type of program that carries out a task.

application developer

A person who creates Meridian IVR applications.

branch

A pathway between cells in a Meridian IVR application.

call flow

A diagram of an application.

call ID

Unique call identifier. This is the unique identifier assigned to a call by the Meridian 1 switch and maintained throughout the entire duration of the call.

caller

A person whose phone call is received or originated by a Meridian IVR application.

cell

The basic element of a Meridian IVR application. Each cell performs an action, such as playing a prompt to a caller. Each cell has a set of branches to other cells. After the cell performs its action, it determines which branch the application should follow to the next cell.

COMA cell

Meridian IVR cell that cancels a transaction with a host.

COMI cell

Meridian IVR cell that sends input to a host via the TRS process.

COMO cell

Meridian IVR cell that receives output from a host via the TRS process.

DELV cell

Meridian IVR cell used in application development by one application to schedule another application to be started automatically, immediately, or at some time in the future. It is used for originating outgoing phone calls to provide services such as message delivery, pager access, telemarketing, wake-up calls, and reminder calls. The scheduled application is called an event.

DN

Directory number.

DNIS

Directory number identification system. It is a service provided on a trunk. DNIS identifies to the called system the last three or four digits of the number actually dialled by the caller. The DNIS digits are sent as in-bank DTMF tones on non-ISDN trunks, or using dial pulses on dial pulse (DIP) trunks. On ISDN PRA trunks, DNIS is carried in the called party IE field of the Q.931 setup message.

DTMF

Dual tone multiple frequency tones, known as touch tones. Applications can collect information from callers by having them press telephone keys to create DTMF tones.

mailbox

A directory that callers can access through a voice channel to store and retrieve voice messages and voice prompts. Each mailbox has its own password.

Meridian IVR

A set of integrated software programs that allow you to develop and execute IVR applications.

message

A voice recording made by a caller.

prompt

A voice recording that helps lead a caller through an application.

user function

Customized “C” code compiled to create a UNIX process. A Meridian IVR application can access an external user function using the USER cell.

user-defined function

An individual “C” function within a user function. Each user function may have several user-defined functions bundled together. Which user-defined function is processed is determined by the specified function code.

usr

User function process. It is an essential Meridian IVR process that controls customer-written user functions.

VAD

A Value Added Developer who develops Meridian IVR applications.

Value Added Developer

See VAD.

xae

Application Editor. It is a non-essential Meridian IVR process.

Meridian IVR

Application Development Guide

Nortel
Customer Documentation
522 University Avenue, 14th Floor
Toronto, Ontario, Canada
M5G 1W7

Information is subject to change without notice. Northern Telecom reserves the right to make changes in design or components as progress in engineering and manufacturing may warrant.

Publication Number:	555-9001-310
Product release:	2.0/I
Document release:	Standard 1.0
Date:	February 1996

Printed in the United States of America

NORTEL
NORTHERN TELECOM