IPTMail_R4 Demo User's Guide

for Windows

05-1663-001

COPYRIGHT NOTICE

Publication Date: February, 2002

Part Number: 05-1663-001

Intel Corporation
Network Processing Group
1515 Route 10
Parsippany NJ 07054
U.S.A.

For **Technical Support**, visit the Dialogic support website at:
http://support.dialogic.com

For **Sales Offices** and other contact information, visit the main Dialogic website at:
http://www.dialogic.com

OPERATING SYSTEM SUPPORT

The term *Windows* refers to both the Windows NT® and Windows® 2000 operating systems. For a complete list of supported Windows operating systems, refer to the Release Guide that came with your Dialogic System Release for Windows, or to the Dialogic support site at *http://support.dialogic.com/releases*.

Table of Contents

List of Tables

List of Figures

1. About This Information

The following topics provide information about this guide:

- Purpose
- Intended Audience
- How to Use This Information
- Related Information

1.1. Purpose

This guide provides information on the IPTMail_R4 demo that is available with the IPLink software. This guide describes the demo, its requirements, and details on how it works.

1.2. Intended Audience

This information is intended for:

- Distributors
- Toolkit Developers
- Independent Software Vendors (ISVs)
- Value Added Resellers (VARs)
- Original Equipment Manufacturers (OEMs)

1.3. How to Use This Information

The information in this guide is organized as follows:

- **Demo Description** introduces you to the demo and its features
- **Demo Requirements** outlines the hardware and software required to run the demo
- **IPTMail_R4 Demo** describes the preparations required before running the demo, how to run the demo, and details on how the demo works.

1.4. Related Information

See the following for more information:

- *System Release 5.1.1 for Windows Release Update* for information on problems fixed, known problems and workarounds, compatibility issues and last minute updates not documented in the published information.
- Dialogic FirstCall InfoServer support web site: http://support.dialogic.com.

2. Demo Description

The IPTMail_R4 demo illustrates how to build a simple Internet telephony voice-mail application using the R4/GlobalCall API.

NOTE: The IPTMail_R4 demo does not function as a gateway. Therefore, it can answer calls only from the IP network. Gateway functionality can be added by connecting a gateway to interface with the PSTN.

2.1. Demo Features

The IPTMail_R4 demo allows the user to:

- play pre-recorded announcements depending on system state
- record new messages to a mailbox
- listen to waiting messages

2.2. What Does the IPTMail_R4 Demo Do?

The IPTMail demo implements a simple Internet telephony voice-mail application. It receives a call from the IP network and re-routes the call to play a series of voice menus. The demo recognizes the DTMF tones sent in response to the menu prompts, and acts in accordance with the specific tone received.

2.2.1. What the IPTMail_R4 Demo Does Not Do?

1. The demo does not allow setting the number of mailboxes on the fly. The number of mailboxes that are supported by the application is defined in the file *maildefs.h* (MAX_NUM_OF_MAILBOXS). The default is set to 200. Each mailbox can save only one message.
2. The following features are not supported by this demo application:
    - UII message
    - NonStdCmd message
    - NonStdParm data
    - Q.931Facility message

2.3. How Does the IPTMail_R4 Demo Work?

The application answers the incoming call from the IP. When the H.323 connection has been established, the application uses the DM3 Player component to play the voice-mail menu. The DM3 Signal Buffer sub-component is used to recognize the digits that were pressed by the user at the remote site. According to the digit pressed, the application performs the appropriate operation:

- record new messages in a mailbox
- listen to waiting messages

3. System Requirements

To run the IPTMail_R4 demo, you must have a system that meets the following requirements.

3.1. Hardware Requirements

The following hardware is required:

- An Intel-based Pentium 200 MHz or better PC
- At least 32 Mbytes of RAM
- Additional 85 MB of available hard disk space
- DM/IPLink-T1(E1)_NIC board
  - The voice resource may be loaded on the IPLink board or on a D/xx board, connected via an SCbus cable.
- CD-ROM drive
- VGA or higher-resolution display adapter
- Microsoft Mouse or compatible pointing device
- IP network cable

Refer to the *System Release 5.1.1 for Windows Release Guide* for the most up-to-date information on board support for this demo.

3.2. Software Requirements

The following software is required:

- Windows NT 4.0 with service pack 3 or Windows 2000
- Visual C++ 6.0
- Dialogic System Release 5.1.1 for Windows

4. The IPTMail_R4 Demo

4.1. Preparing to Run the Demo

4.1.1. Connecting to External Equipment

This section describes how to connect the IPLink board to external equipment.

Connect a gateway to connect via the PSTN network, or use an Internet telephone or NetMeeting to connect directly via the IP.

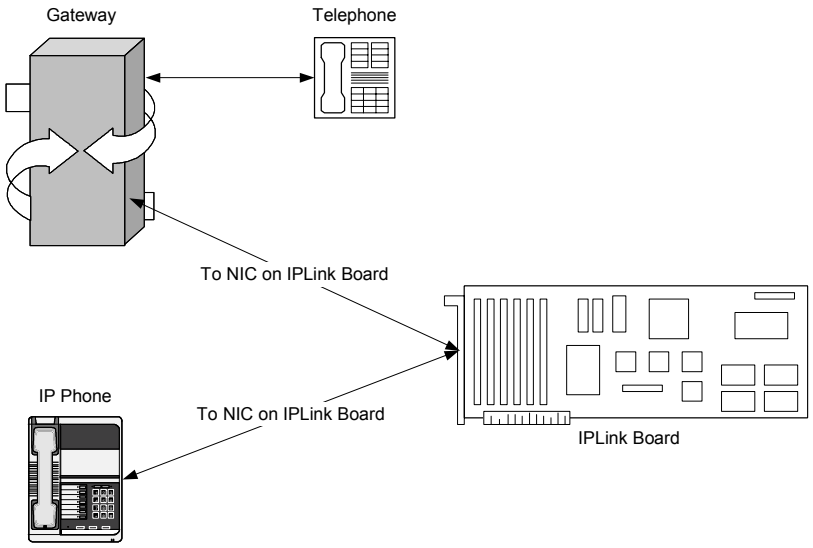

Figure 1. Connecting to External Equipment

4.1.2. Downloading Firmware

This section describes how to download the IPTMail firmware. It specifies special parameters that must be set before download.

Follow the directions for installing and configuring IPLink in the *System Release Installation and Configuration Guide for Windows*. The following parameters should be set in the Dialogic Configuration Manager (DCM) for the IPTMail demo to work properly:

- PCD/FCD Files

  Identify a PCD and corresponding FCD file that matches your configuration. Make sure to select a file that starts **ipt_evr**, in order to support the voice resources needed for the voice mail application. The PCD files supplied with the release are described in the *System Release 5.1.1 for Windows Release Guide*.

- IP Address

  1. Select the Network tab.
  2. Enter the IP address of the NIC on the IPLink board in the IPAddress field.
  3. Enter the Gateway IP address in the Gateway IPAddress field. This address should match the segment in the IPAddress field, ending with 250 (xxx.xxx.xxx.250).
  4. Click on the Start Service button to start the download.

4.1.3. Edit the Iptmail_R4.cfg File

This section describes the configuration file used with the demo. It explains the contents and presents a sample file.

Before running the demo, modify the *IPTMai_R4l.cfg* file to reflect your system environment. Use a text editor and open the file from *Dialogic\Samples\gc_demos\IPDemos\iptmail_R4\debug* or *\release*.

Sample Configuration File

Below is an example of the *IPTMail_R4.cfg* file. Update the information about the coder that should be used when originating a call toward the IP network, according to your configuration.

```
Channel = 1-30
```

```
{
   ipProtocolName = H323_NTSC
   Capability
   {
      Type = g723_5_3k
      FramesPerPkt = 1
      VAD = 0
   }
}
```

4.2. Starting the Demo

This section describes how to start the demo, what must be done before launching the demo, and launch options.

Select Run from the Start Menu. The demo executable file can be found in *C:\Program Files\Dialogic\Samples\gc_demos\IPDemos\Iptmail_R4\Debug\Iptmail_R4.exe*. Click OK to run the IPTMail demo using the default settings.

4.3. Demo Options

To specify certain options at run-time, launch the demo from a DOS command line, using any of the switches listed in *Table 1. IPTMail Command Line Switches*:

Table 1. IPTMail Command Line Switches

Switch	Action	Default
-n	Sets the number of channels	The lesser of Voice Devices or IP devices
-c <*filename*>	Configuration file name	IPTMail_R4.cfg

Switch	Action	Default
-d	Sets Debug Level: 0 = Fatal - used when there is a non-recoverable error 1 = Error - same as Fatal; used when there is a non-recoverable error 2 = Warning - used when some problem or failure has occurred, that doesn't affect the channel's normal action. 3 = Trace - used at the start of any function, or upon the successful completion of a function. 4 = Info - prints data related to a specific action	0

Switch	Action	Default
-l	Printouts will be printed into channel log files in the demo directory. -lall = log files will be created for all available channels -l<channel list> = log files will be created only for the channel ranges or channels specified in the list. **Note:** If the "-l" option is not used, all prints will be to the stdout for channels 1 and 2 only.	Disabled
-h or ?	Prints the command syntax to the screen	Off

Each channel is initialized and the log for channels one and two is displayed in a DOS window. See *Appendix A* for the log file from one channel of a typical session receiving a call from the IP.

4.4. Using the Demo

4.4.1. Using the Voicemail

The IPTMail_R4 demo allows the caller to interact with a series of voice menus, using the telephone keypad to enter an option. Basic operations are play a pre-recorded message and record a new message. Each menu prompts the caller to select an action by pushing a key.

Main Menu [Main_Menu]
1 - Send Message

2 - Listen to Message
* - Quit

Send Message Prompt [Send_Message]
Enter Mailbox Number - between 101 - 299
* - Quit

Start Record Prompt [Start_Record]
2 - Start/Stop Record (at end Stop Record Prompt is played)
* - Quit

Stop Record Prompt [Stop_Record]
2 - Discard Message and re-record message to same mailbox
3 - Confirm Message (and Return to Main Menu) [Save_Confirm]
4 - Replay Message (and replay Stop Record Prompt)
* - Quit

Listen to Message Prompt [Listen_Menu]
- Enter Mailbox Number - between 101 - 299 (Recorded message is played)
* - Quit

Stop Listening Prompt [Stop_Listen]
2 - Discard Message and quit (and Return to Main Menu)
* - Quit

4.5. Keyboard Commands

While the demo is running, the administrator can send the following keyboard commands to the demo:

- q , ctrl+c - end application
- d - set debug level
- c - print channel information

4.7. Demo Details

This chapter provides a detailed description of the demo.

4.7.1. Files Used by the Demo

This section presents a closer examination of the IPTMail_R4 demo, by looking into its structure and source code. The IPTMail_R4 demo source code is located in the *Dialogic\Samples\gc_demos\IPDemos\iptmail_R4* directory and is included in the following files:

Table 2. IPTMail Source Files

Samples\gc_demos\IP Demos\IPTMail_R4\	APPDEFS.H	Application definitions
Samples\gc_demos\IP Demos\IPTMail_R4\	APPINIT.C	Application initialization
Samples\gc_demos\IP Demos\IPTMail_R4\	APPINIT.H	Header file for application initialization
Samples\gc_demos\IP Demos\IPTMail_R4\	APPMAIN.C	The main application file: contains the thread initialization, route ports, etc.
Samples\gc_demos\IP Demos\IPTMail_R4\	APPMAIN.H	Header file for main application
Samples\gc_demos\IP Demos\IPTMail_R4\	APPPARS.C	Read configuration file functions
Samples\gc_demos\IP Demos\IPTMail_R4\	APPPARS.H	Header file for reading configuration file functions
Samples\gc_demos\IP Demos\IPTMail_R4\	APPSTAT.C	Application state machine functions
Samples\gc_demos\IP Demos\IPTMail_R4\	APPSTAT.H	Header file for the application state machine functions
Samples\gc_demos\IP Demos\IPTMail_R4\	APPSTRC.H	Definitions of application structures
Samples\gc_demos\IP Demos\IPTMail_R4\	APPVARS.H	Definition of application variables

Samples\gc_demos\IP Demos\IPTMail_R4\	GATEIP.C	Initialize IP, Get IP available channels, handle IP events
Samples\gc_demos\IP Demos\IPTMail_R4\	GATEIP.H	Header file for the IP functions
Samples\gc_demos\IP Demos\IPTMail_R4\	MAILDEFS.H	Mailbox definitions
Samples\gc_demos\IP Demos\IPTMail_R4\	MAILUTIL.C	Mailbox utility functions
Samples\gc_demos\IP Demos\IPTMail_R4\	MAILUTIL.H	Header file for Mailbox utility functions
Samples\gc_demos\IP Demos\IPTMail_R4\	VOICE.C	Initialize voice resources, get voice resources available channels, handle voice resources events
Samples\gc_demos\IP Demos\IPTMail_R4\	VOICE.H	Header file for voice resources functions.
Samples\gc_demos\IP Demos\IPTMail_R4\	Iptmail_r4.cfg	IPTMail demo configuration file
Samples\gc_demos\IP Demos\IPTMail_R4\	Iptmail_r4.exe	IPTMail demo executable file
Samples\gc_demos\IP Demos\IPTMail_R4\	Stoplisten.vox	Voice file
Samples\gc_demos\IP Demos\IPTMail_R4\	ErrorInput.vox	Voice file
Samples\gc_demos\IP Demos\IPTMail_R4\	ListenMenu.vox	Voice file
Samples\gc_demos\IP Demos\IPTMail_R4\	MainMenu.vox	Voice file
Samples\gc_demos\IP Demos\IPTMail_R4\	SaveConfirm.vox	Voice file

Samples\gc_demos\IP Demos\IPTMail_R4\	SendMsg.vox	Voice file
Samples\gc_demos\IP Demos\IPTMail_R4\	StartRec.vox	Voice file
Samples\gc_demos\IP Demos\IPTMail_R4\	StopRec.vox	Voice File
Samples\gc_demos\IP Demos\IPTMail_R4\	ThankYou.vox	Voice File
Samples\gc_demos\IP Demos\IPTMail_R4\	UnavMenu.vox	Voice File
Samples\gc_demos\UTIL\	Libdbg.c, Libdbg.h, Libdefs.h	Project files that include all debug definitions and functions (static debug library)

NOTE: If you move the files for the IPTMail demo, be sure to move the *.vox* files as well, otherwise the IPTMail demo will not work.

4.7.2. Programming Model

The IPTMail_R4 demo is designed to operate in a single thread mode. (There is a secondary thread operating alongside the main thread. This secondary thread consists of keyboard events, and does not change the overall structure of the demo.). Because of the nature of a single threaded application, the state machines do not, and should not, block any other operation, but wait for the SRL. *Figure 2. Thread Diagram* illustrates the demo's thread structure.

NOTE: The application programming framework also allows multi-thread operation. It is not demonstrated in the IPTMail_R4 demo.

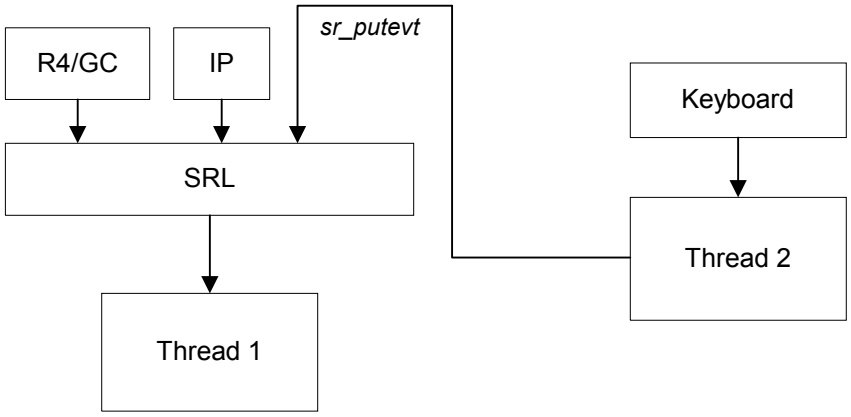

Figure 2. Thread Diagram

4.7.3. Demo-Related Initialization (Appinit.c)

1. Get any arguments from the command-line.
2. Reset the demo data structures. Initialize all channels' states to INIT.
3. Call **gc_Start():** starts all configured , call control libraries.
4. Set SRL modeltype to SR_STASYNC.
5. Set-up the call-back handler, **callback_hdlr**.
6. Call **getVoiceChannels( )** which checks the number of available voice devices :
   - Get number of voice boards, by calling **sr_getboardcnt( )**
   - For each board that was found:
     - Open the board, by calling **dx_open()**
     - Find number of channels per board, by calling **ATDV_SUBDEVS()**
     - Calculate the logical board and channel and save them into Session.VoiceParams
     - Close the board, by calling **dx_close( )**
7. Call **getIPChannels( )** which checks the number of available IP channels:
   - Get number of voice boards, by calling **sr_getboardcnt( )**

- For each board that was found:
  - Open the board, by calling **gc_open()**
  - Find number of IP channels per board, by calling **TDV_SUBDEVS()**
  - Calculate the logical board and channel and save them into Session.IPParams
  - Close the board, by calling **gc_close( )**

8. Call **getmailChannels( )** to find the demo MAX available channels (the smaller of available IP or Voice Devices and the number of channels specified with the –n command line option, if used).

   NOTE: The **printf()** function must be used when printing an error/ trace/warning during the above initialization operations one through six, because some actions are not related to a specific channel and log files are not yet opened.

9. Open the channels' log files if –l command-line option was used.
10. Read information from the configuration file (IPTMail_R4.cfg or other CFG file determined by the user) and update the ConfigFileParm in the Session data structure.
11. Print information from the configuration file.
12. Call the **IPInit()** function. See *Section 4.7.4. IPLink-Related Initialization* for a description of the **IPInit()** function.
13. Call the **VoiceInit()** function. See *Section 4.7.5. Voice-Related Initialization* for a description of the **VoiceInit()** function..
14. Call the **InitMailBoxs()** function, which initializes the MailBox structure.
15. Create waitForKey thread to receive keyboard input.

4.7.4. IPLink-Related Initialization

The IPLink initialization procedure described in this section is defined in the **IPInit()** function in the *GATEIP.C* file, which performs the following:

For all channels:

- Call **gc_OpenEx()**

Open all IP devices, and returns LineDevH.

- Save the channel number in the global array (HandleToChannel[ ]) according to the LineDevH handle.
- Call **gc_GetXmitSlot(VoiceH)**
  Gets the transmit timeslot(Xmitslot) for the IP devices and saves it in the session.IPParams structure.

4.7.5. Voice-Related Initialization

Call **VoiceInit()**, located in the *VOICE.C* file, which does the following:

For all channels:

- Call **dx_open()** to get the VoiceH (voice device handle).
- Save the channel number in the global array (HandleToChannel[ ]) according to the VoiceH handle.
- Call **dx_getxmitslot(VoiceH).** Returns the Xmitslot (the SCbus time slot connected to transmit of voice channel) and saves it in the session.VoiceParams structure.

4.7.6. Data Structures

This section describes the main data structures:

- Session
- IPParams
- VoiceParams
- MailBox

Session:

The Session data structure (defined in *Appstrc.h*) contains the following fields:

- SessionState
  - The state machine current state

- State machine function
  - The current state machine function according to SessionState

- Session Number
  - The channel number

- ConfigFileParm
  - Information from the CFG file

- LogFile
  - A pointer to a session log file

- lpMailBox
  - Pointer to the MailBox that is used

- IPparams
  - IPPARAMS structure

- VoiceParams
  - VOICEPARAMS structure

- enteredExtNum
  - the extension number entered by the user

There is a session structure for every channel. Each channel session contains pointer to a MailBox structure.

```
typedef struct {

    USHORT            sessionState;     /* Session state */
    appStateFxn       stateFxn;         /* Next state machine function to be called */
    UINT              sessionNumber;    /* Index session number */
    CallParameters    ConfigFileParm;   /* Information from cfg file */
    FILE              *LogFile;
    LPMAILBOX         lpMailBox;        /* Pointer to the MailBox that is used */
    VOICEPARAMS       VoiceParams;      /* The voice device parameters */
    IPPARAMS           ipParams;        /* The IP device parameters */
    char              enteredExtNum[10]; /* the Ext. number entered by the user*/
    BOOL              IsRouted; /* If Route done then set flag to TRUE else FALSE */
  } AppSession;
```

IPPARAMS

The IPParams data structure (defined in *Appstrc.h*) contains the following fields:

- LineDevH

  - The Line Device Handle, to identify the physical device(s) that carries the call

- VoiceDevH
  - The voice device handle

- Xmitslot
  - The timeslot number for the IP

- logBoard
  - The logical board number

- logChan
  - The logical channel number

- crn
  - The Call Reference Number, generated after getting IP Offered

```
typedef struct {
     LINEDEV   linedev;   /* line device handle to identify the IP physical */
                          /* device that carries the call */
     int       VoiceDevH; /* the voice device handle */
     long      XmitSlot;  /* the transmit timeslot number assigned for the Voice */
                          /* Resourses */
     int       logBoard;  /* The logical board number for the session */
     int       logChan;   /* The logical channel number for the session */
     CRN       crn;       /* The current call's CRN */
} IPPARAMS;
```

VOICEPARAMS

The VoiceParams data structure (defined in *Appstrc.h*) contains the following fields:

- VoiceDevH
  - The voice device handle

- Xmitslot
  - The timeslot number for the voice resources

- logBoard
  - The logical board number

- logChan
  - The logical channel number

- digp -
  - A structure(DV_DIGIT) for the digits when getting TDX_GETDIG event

- iott
  - A structure (DX_IOTT) for the file handle, played or recorded file

- fPlayerStopped
  - A flag which indicates the Player status

- fRecorderStopped
  - A flag which indicates the Recorder status

```
typedef struct {
     int   VoiceDevH;     /* the voice device handle */
     long  XmitSlot; /*the  transmit timeslot number assigned for the Voice Resourses */
     int   logBoard;      /* The logical board number for the session */
     int   logChan;       /* The logical channel number for the session */
     DV_DIGIT digp[256];  /* A structure for the digits when getting TDX_GETDIG event */
     DX_IOTT  iott;       /* data structure contains parameters for the Input/Output */
                          /* Transfer Table. */
     BOOL  fPlayerStopped; /* A flag which indicates the Player status */
     BOOL  fRecorderStopped;  /* A flag which indicates the Recorder status */
} VOICEPARAMS;
```

MailBox

The MailBox data structure (defined in *Maildefs.h*) contains the following fields

- ExtNum
  - an array that contains the digits of the mailbox number

- msgFileStatus
  - 0 = no message
  - 1 = one saved message

- isBusy
  - 1 = mailbox is currently answering a call
  - 0 = mailbox is ready to answer an incoming call

```
typedef struct _MAILBOX {
   // mail-box number
   char ExtNum[MAX_EXTNUM_LENGTH+1];

   // file status: 0 for no message,1 for saved message
   MB_STATUS  msgFileStatus;

   // Mailbox busy: 0 for ready to answer an incoming call, 1 for currently answering a
call
   BOOL          isBusy;

} MAILBOX, *LPMAILBOX;
```

4.7.7. Event Mechanism

The IPTMail_R4 demo uses the SRL mechanism to retrieve the IP/R4/Keyboard events. The application waits for events using the **sr_waitevt(-1)** function. When an event occurs, SRL calls event handlers automatically.

In the initialization phase of the demo the **mailInitialization()** function will perform the following:

1. Set SRL modeltype to SR_STASYNC, by calling **sr_setparm( ).**
2. Set-up the call-back handler, by calling **sr_enbhdlr( ).**

The SRL will be notified of application exit events using a special key CTRL_CLOSE_EVENT, when calling **sr_putevt()**.

Retrieving Events From the Queue

There is an endless loop **{while(1) }** in the **main()** function in the *APPMAIN.C* file. In that loop, the application waits forever for an event by calling the **sr_waitevt( )** function. The event must be handled immediately and event-specific information should be retrieved before the next call to **sr_waitevt().**

When the next event occurs or when a timeout is reached, the **sr_waitevt()** returns and the call-back handler function is called automatically.

Handling SRL Events

When the R4/GC/Keyboard event is received, the application performs the following:

1. Get the event device handle, by calling **sr_getevtdev().**

2. Get the channel number related to the event, from the global array (HandleToChannel[ ]).
3. Update the METAEVENT structure by calling **gc_GetMetaEvent()**.
4. Get the event type, by calling **sr_getevttype().**
   If type==CTRL_CLOSE_EVENT,
   then end application,
   else run the state machine function.

5. Application Flow

5.1. Using State Machines

There is an endless loop (while(1)) in the **main()** function in the *APPMAIN.C* file. In that loop, the application waits for an event by calling the **sr_waitevt()** function.

All channels are initialized to the INIT state.

As soon as an event is received, the event type, the channel number, and the reason (reason for the event, if there is one) are analyzed, and the appropriate state machine function is called.

After all the operations are performed within the channel's event state, the next state machine function is updated according to the event received and the SessionState.

5.2. Inbound IP Call

The following state diagram describes the call states for an inbound call from the IP to the demo. Each state is represented by an ellipse containing the state name, the event(s) associated with that state, and the actions performed by the application. The states are connected with arrows indicating the valid state changes.

NOTE: Due to space restrictions, the state name in the diagram is presented in shortened format. Each state name should be prefixed by IPTMAIL_.

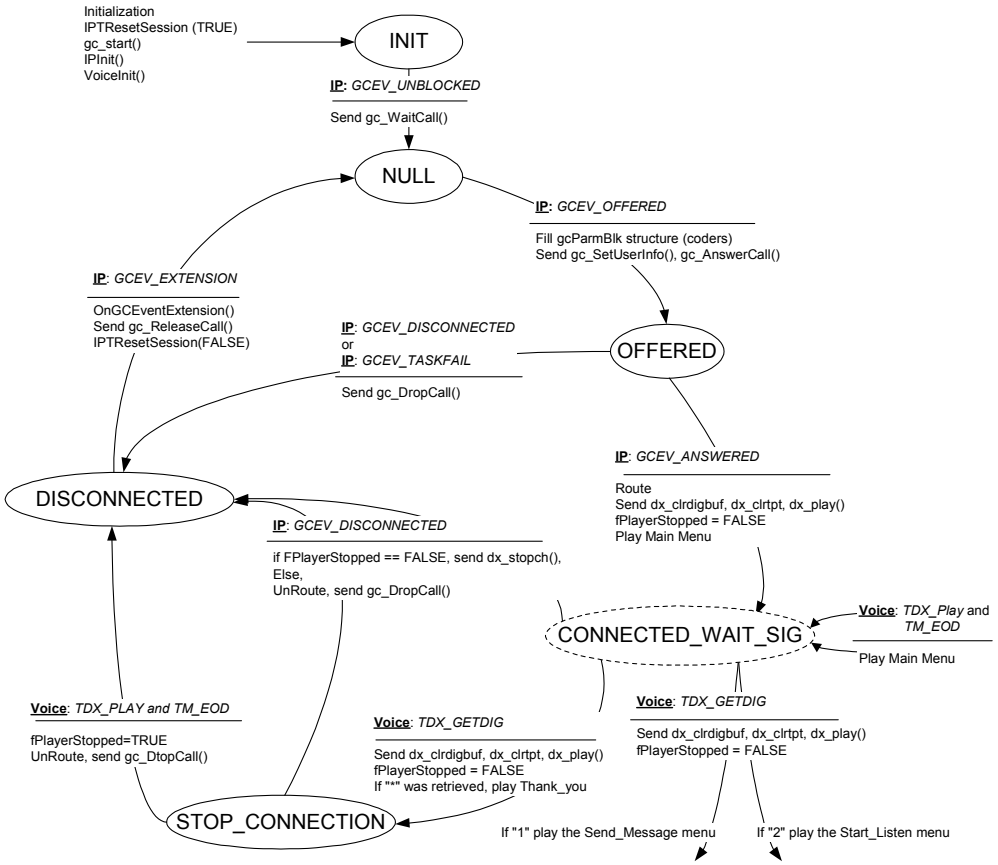

Figure 3. State Diagram — Inbound IP Call

5.2.1. Null State

The application waits for a call event in the IPTMAIL_ NULL state.

When the application receives the event *GCEV_OFFERED*, it calls an internal function **ipAnswerCall()** which fills the gcParmBlk structure with coder

information and sends **gc_SetUserInfo()** and **gc_AnswerCall( )** to the firmware. The call state transitions to IPTMAIL_ OFFERED.

If, for any reason, the call should fail, the application receives a *GCEV_TASKFAIL* event. The application is ended and all channels are moved to the IPTMAIL_QUIT state.

5.2.2. Offered State

When the application receives a *GCEV_ANSWERED* event, it routes the IP and Voice devices to listen to each other, flushes the DTMF buffer, enables the DTMF tones used by the Main menu, and plays the Main menu. The call state transitions to IPTMAIL_ CONNECTED_WAIT_SIG.

If, for any reason, the function should fail, the application receives a *GCEV_TASKFAIL* event. It calls the internal function **DisconnectCall** which, in turn, stops the I/O operation if in process, calls **gc_DropCall(),** and **unRoutes** the call, if needed. The call state transitions to IPTMAIL_ DISCONNECTED.

If, for any reason, the remote side should disconnect, the application receives a *GCEV_DISCONNECTED* event. It calls the internal function **DisconnectCall** which, in turn, stops the I/O operation if in process, calls **gc_DropCall**(), and **unRoutes** the call, if needed. The call state transitions to IPTMAIL_ DISCONNECTED.

5.2.3. Connected_Wait_Sig State

The application waits for the user to make a selection from the Main menu:

- record a message (“1”)
- listen to a message (“2”)
- exit (“*”)

When the application receives a *TDX_PLAY* event, it determines if the Player was stopped because it reached the end of data or a DTMF digit was detected. In the case of end of data (*TM_EOD*), the Player replays the menu. In the case of a digit (*TM_MAXDTMF*), the application sends **dx_getdig()** to retrieve the digit.

The application waits in the IPTMAIL_CONNECTED_WAIT_SIG_STATE until it receives a *TDX_GETDIG* event.

If the retrieved digit is "1", the application plays the Send_Message menu. The call state transitions to IPTMAIL_CONNECTED_REC_AND_SEND.

If the retrieved digit is "2", the application plays the Start_Listen menu. The call state transitions to IPTMAIL_CONNECTED_START_LISTEN.

If the retrieved digit is "*", the application plays the Thank_You menu and the call state transitions to IPTMAIL_STOP_CONNECTION.

If no digit was retrieved (due to timeout or a non-enabled digit), the application re-plays the Main menu.

The application can also receive a *GCEV_DISCONNECTED* event. It calls **dx_stopch()** to prevent further playing, unroutes the clusters, and calls **gc_DropCall()**.

Once the call state transitions to IPTMAIL_CONNECTED_WAIT_SIG, the demo application begins to operate in voice-mail mode. See *Chapter 6. Using the Voice Mail* for a complete description of this part of the demo.

5.2.4. Disconnected State

The application waits for a *GCEV_DROPCALL* event. When it receives this event, it sends the **ExtensionGetCallInfo()** function to get the call duration time and RTCP information. This information is returned in the *GCEV_EXTENSION* event. When it receives this event, the application calls **gc_ReleaseCall( )** and then calls **IPTResetSession()**. The call state transitions to IPTMAIL_NULL and the channel may be reused.

If the Player or Recorder haven't yet been closed, the application waits for a *TDX_PLAY* or *TDX_RECORD,* closes the file and calls **DisconnectCall()** which in turn calls **gc_DropCall()**.

5.2.5. Stop_Connection State

The application waits for a *TDX_PLAY* event. It requests the termination reason, closes the played file, unroutes the devices and calls **gc_DropCall()**. The call state transitions to IPTMAIL_DISCONNECTED.

The application may receive a *GCEV_DISCONNECTED* event. It checks to see if the Player has stopped. If PlayerStopped = FALSE, it sends **dx_stopch()** and waits for *TDX_PLAY* and *TM_EOD* events. Upon receipt of these events, the application unroutes the clusters and calls **gc_DropCall()**. The call state transitions to IPTMAIL_DISCONNECTED.

6. Using the Voice Mail

Each state function within the voice mail part of the application performs the same basic activities:

1. Wait for an IP and Voice Resource event.
2. Use **dx_play()** to play the appropriate menu. *TDX_PLAY* termination events will be generated to indicate completion.

   After getting the *TDX_PLAY* event the application calls **ATDX_TERMMSK(chdev)** to check the termination reason. It may either *TM_EOD* (end of data/file) or *TM_MAXDTMF* (user pressed digit/s as determined by the TPT – Termination Parameter Table).
   - If *TM_EOD*, then play the relevant menu again.
   - If *TM_MAXDTMF*, then collect the DTMF from the digit buffer.
3. Use **dx_rec()** to record a file. *TDX_RECORD* termination events will be generated to indicate completion.
4. Call **dx_getdig()** to **c**ollect the DTMF tone(s) from the digit buffer. *TDX_GETDIG* termination event will be generated to indicate completion.
5. Call **dx_clrdigbuf()** to clear all digits in the channel's firmware digit buffer.
6. Call **dx_clrtpt()** to clear the Table Parameter Termination (TPT) structure.

6.1. Recording a Message

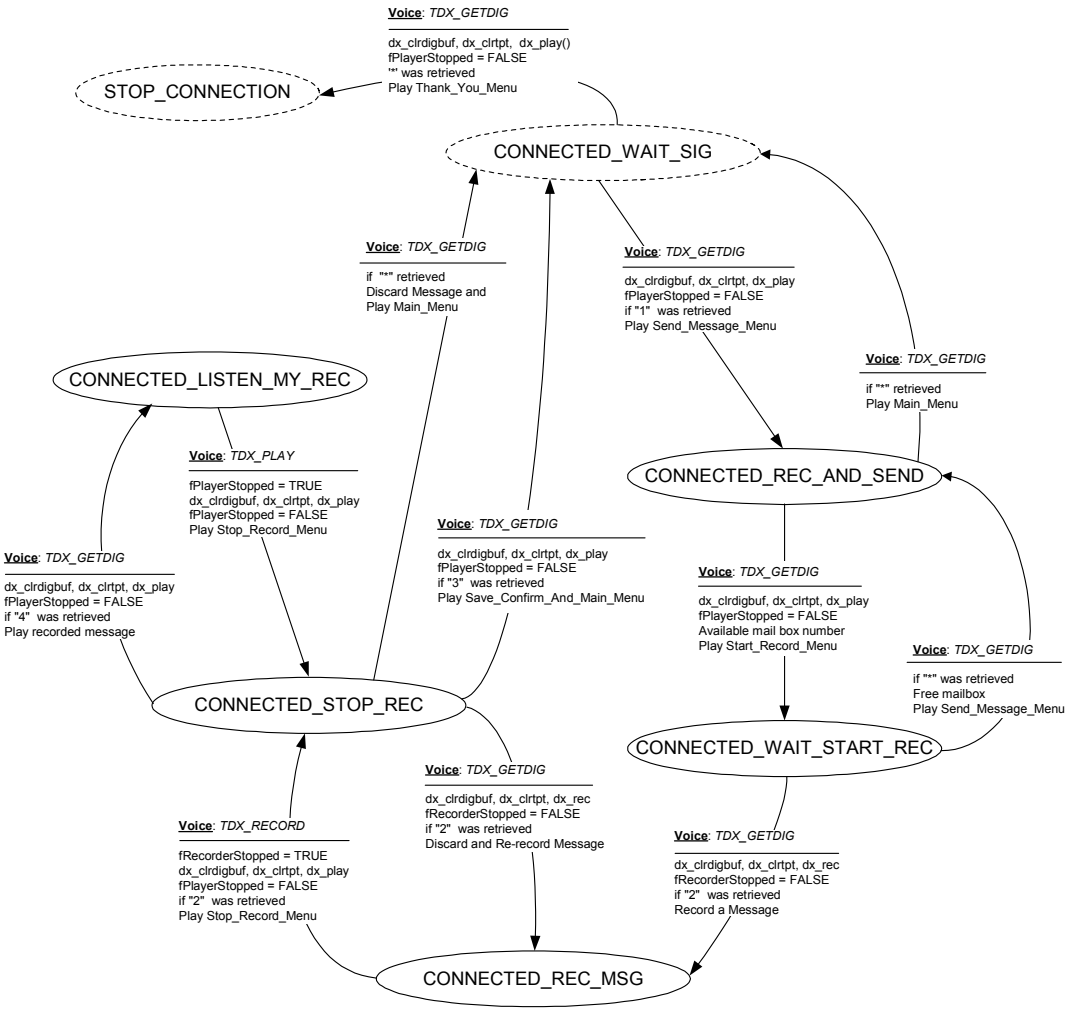

Figure 4. Record Message State Diagram

6.1.1. Connected_Rec_And_Send State

The application waits for a *TDX_GETDIG* event. The retrieved digit may be either an asterisk "*" or a 3-digit mailbox number. Any other digit or combination of digits generates an error.

If the digit is "*", the application replays the Main menu and the call state transitions to IPTMAIL_CONNECTED_WAIT_SIG.

If three digits are retrieved, the application checks if it is a legal mailbox number. If it is not, the application plays the ERROR_MENU and waits for another *TDX_GETDIG* event.

If the mailbox number is legal, the application checks if the requested mailbox is available. If it is unavailable, the application plays the UNAVAILABLE_TRY_AGAIN_MENU and waits for another *TDX_GETDIG* event.

If the mailbox number is available, the application plays the STARTREC_MENU and the call state transitions to IPTMAIL_CONNECTED_WAIT_START_REC.

If the application receives a *TDX_PLAY* event, it requests the termination reason. If it receives *TM_EOD*, the STOPREC_MENU is played. Otherwise it calls **dx_getdig()** to collect the digit/s from the digit buffer,. The application then waits for a *TDX_GETDIG* event. The call state remains in IPTMAIL_CONNECTED_REC_AND_SEND.

6.1.2. Connected_Wait_Start_Rec State

The application waits for a *TDX_GETDIG* event. The retrieved digit may be either an asterisk "*" or "2". Any other digit generates an error.

If the digit is "*", the application plays the SENDMSG_MENU. The call state transitions to IPTMAIL_CONNECTED_RECORD_AND_SEND.

If the digit is "2", the application:

- Calls **dx_clrtpt()** to clear the TPT (Termination Parameter Table) structures
- Sets the DV_TPT structure to terminate on "2"

- Set up DX_IOTT (data structure that contains parameters for the Input/Output Transfer Table)
- Calls **dx_clrdigbuf()** to clear the digit from the buffer
- Calls **dx_rec()** to begin recording on the channel

The call state transitions to IPTMAIL_CONNECTED_RECORD_MSG.

6.1.3. Connected_Rec_Msg State

The application waits for a *TDX_RECORD* event. It gets the termination reason and calls **dx_fileclose()** to close the recorded file.

The application plays the STOPREC_MENU and the call state transitions to IPTMAIL_CONNECTED_STOP_REC.

6.1.4. Connected_Stop_Rec State

The application waits for a *TDX_GETDIG* event. The retrieved digit may be either an asterisk "*", "2", "3", or "4". Any other digit generates an error.

If the returned digit is "2", the application discards the recorded message and plays the beep prompt to begin recording again. The call state transitions to IPTMAIL_CONNECTED_RECORD_MSG.

If the returned digit is "3", the application marks the mailbox as full and plays the SAVE_CONFIRM_AND_ MAIN_MENU. The call state transitions to IPTMAIL_CONNECTED_WAIT_SIG.

If the returned digit is "4", the application plays the message back and the call state transitions to IPTMAIL_CONNECTED_LISTEN_MY_REC.

If the returned digit it "*", the application frees the mailbox and doesn't store the message. The application plays the MAIN_MENU and the call state transitions to IPTMAIL_CONNECTED_WAIT_SIG.

6.1.5. Connected_Listen_My_Rec State

The application waits for a *TDX_PLAY* event. When it receives the event, it closes the played file, and plays the STOPREC_MENU. The call state transitions to IPTMAIL_CONNECTED_STOP_REC.

6.2. Listening to a Message

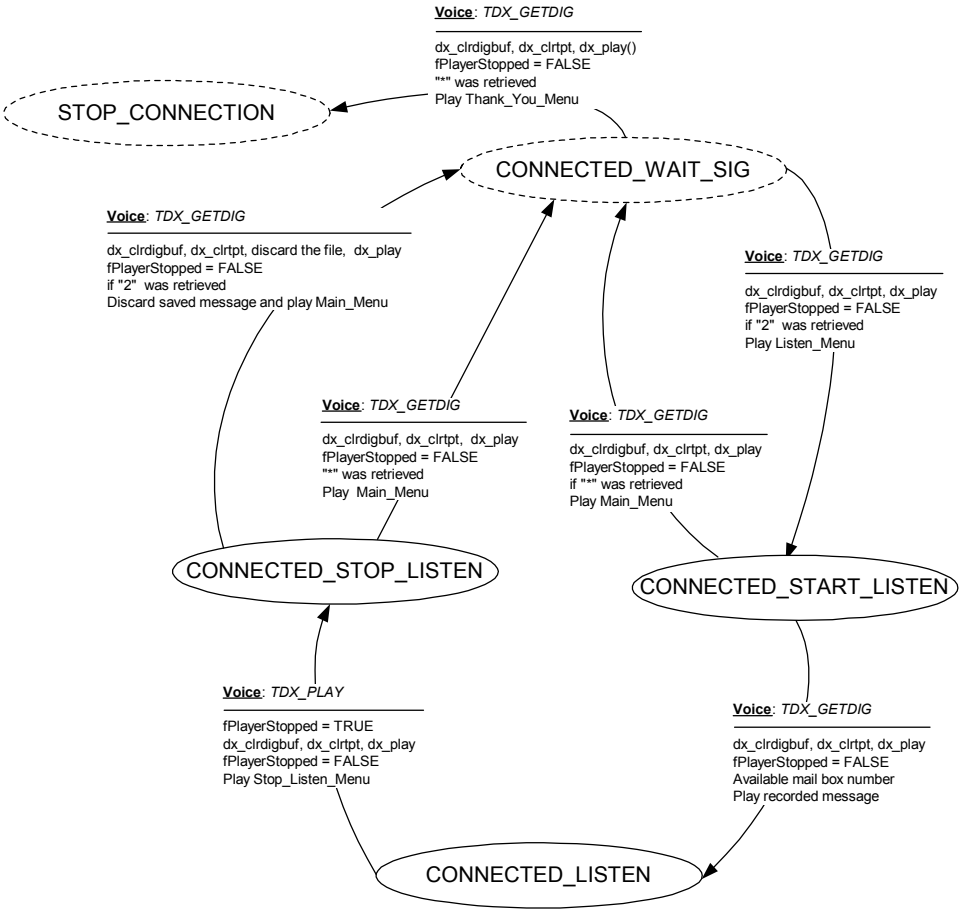

Figure 5. Listen to Message State Diagram

6.2.1. Connected_Start_Listen State

The application waits for a *TDX_GETDIG* event. The retrieved digit may be either an asterisk "*" or a 3-digit mailbox number. Any other digit or combination of digits generates an error.

If the digit is "*", the application replays the Main menu and the call state transitions to IPTMAIL_CONNECTED_WAIT_SIG.

If three digits are retrieved, the application checks if it is a legal mailbox number. If it is not, the application plays the ERROR_MENU and waits for another *TDX_GETDIG* event.

If the mailbox number is legal, the application checks if the requested mailbox is available. If it is unavailable, the application plays the UNAVAILABLE_TRY_AGAIN_MENU and waits for another *TDX_GETDIG* event.

If the mailbox number is available, the application plays the file and the call state transitions to IPTMAIL_CONNECTED_LISTEN.

6.2.2. Connected_Listen State

The application waits for a *TDX_PLAY* event. When it receives the event, it gets the termination reason, closes the played file, and plays the STOPLISTEN_MENU. The call state transitions to IPTMAIL_CONNECTED_STOP_LISTEN.

6.2.3. Connected_Stop_Listen State

The application waits for a *TDX_GETDIG* event. The retrieved digit may be either an asterisk "*" or "2". Any other digit or combination of digits generates an error.

If the digit is "*", the application preserves the message in the mailbox and plays the MAIN menu. The call state transitions to IPTMAIL_CONNECTED_WAIT_SIG.

If the digit is "2" the application discards the message, frees the mailbox, and plays the MAIN menu. The call state transitions to IPTMAIL_CONNECTED_WAIT_SIG.

6.3. Error Handling

Figure 6. Error Handling—GCEV_DISCONNECTED

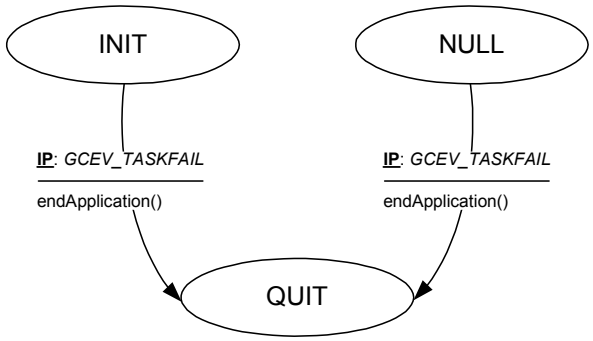

Figure 7. Error Handling—GCEV_TASKFAIL

6.3.1. IPTMAIL_QUIT State

The application waits for either a *TDX_PLAY* or a *TDX_RECORD* event.

Upon receiving either event, the application gets the termination reason, unroutes the call, and closes the voice and IP devices. If all voice devices are closed, the application exits.

6.3.2. Errors from IP

The error might occur in different phases:

- If the called function returns a <0 , first process the error [by calling: gc_ErrorValue( ), gc_ResultMsg( )] and then end the call or exit the application.
- If the function was called successfully, but **GCEV_TASKFAIL** event is received later, indicating that the function failed, the application will be terminated

6.3.3. Errors from Voice Resources:

- The main error which could be received while calling the R4 function toward the Voice Resources is represented by the **TDX_ERROR** event
- The IPTMail_R4 demo drops the call after receiving this error.

Appendix A

Log File

```
02/13/02  10:01:29
TRACE: File: gateip.c Line: 150
     Got TX timeslot(47) of the IP device on channel1

02/13/02  10:01:29
TRACE: File: voice.c Line: 200
     Got TX timeslot(96) of the voice device on channel1

02/13/02  10:01:29
TRACE: File: appstat.c Line: 62
In IPTMAIL_INIT on channel 1
      got Event GCEV_UNBLOCKED (0x833)

02/13/02  10:01:29
TRACE: File: appstat.c Line: 81
     gc_WaitCall was called on channel 1

02/13/02  10:01:32
TRACE: File: appstat.c Line: 139
In IPTMAIL_NULL  on channel 1
      got Event GCEV_OFFERED (0x824)

 02/13/02  10:01:32
TRACE: File: gateip.c Line: 311
     Answering Call on channel 1

 02/13/02  10:01:32
TRACE: File: appstat.c Line: 216
In IPTMAIL_OFFERED on channel 1
      got event GCEV_ANSWERED (0x802)

02/13/02  10:01:32
TRACE: File: appmain.c Line: 228
     Voice device succeed to listen to IP device timeslot, on channel 1.

 02/13/02  10:01:32
TRACE: File: appmain.c Line: 238
     IP device succeed to listen to Voice device timeslot, on channel 1.

02/13/02  10:01:32
TRACE: File: appstat.c Line: 251
     Route done on channel 1

 02/13/02  10:01:32
TRACE: File: voice.c Line: 288
     The VOX file: mainmenu.vox was opened on channel 1

 02/13/02  10:01:32
TRACE: File: appstat.c Line: 261
     Player started on channel 1

 02/13/02  10:01:38
TRACE: File: appstat.c Line: 313
IPTMAIL_CONNECTED_WAIT_SIG on channel 1
      got event TDX_PLAY (0x81)
```

```
 02/13/02  10:01:38
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: Receiving DTMF digits

02/13/02  10:01:38
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

02/13/02  10:01:38
TRACE: File: voice.c Line: 445
     dx_getdig function sent successfully on channel 1

 02/13/02  10:01:38
TRACE: File: appstat.c Line: 313
IPTMAIL_CONNECTED_WAIT_SIG on channel 1
      got event TDX_GETDIG (0x83)

 02/13/02  10:01:38
TRACE: File: appstat.c Line: 334
     Received digits: <1> on channel 1

 02/13/02  10:01:38
TRACE: File: voice.c Line: 288
     The VOX file: sendmsg.vox was opened on channel 1

 02/13/02  10:01:52
TRACE: File: appstat.c Line: 491
In IPMAIL_CONNECTED_RECORD_AND_SEND on channel 1
      got event TDX_PLAY (0x81)

 02/13/02  10:01:52
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: End Of File

 02/13/02  10:01:52
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:01:52
TRACE: File: voice.c Line: 288
     The VOX file: sendmsg.vox was opened on channel 1

 02/13/02  10:02:03
TRACE: File: appstat.c Line: 491
In IPMAIL_CONNECTED_RECORD_AND_SEND on channel 1
      got event TDX_PLAY (0x81)

02/13/02  10:02:03
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: Receiving DTMF digits

 02/13/02  10:02:03
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:02:03
TRACE: File: voice.c Line: 445
     dx_getdig function sent successfully on channel 1
```

```
 02/13/02  10:02:05
TRACE: File: appstat.c Line: 491
In IPMAIL_CONNECTED_RECORD_AND_SEND on channel 1
      got event TDX_GETDIG (0x83)

 02/13/02  10:02:05
TRACE: File: appstat.c Line: 511
     Received digits: <121> on channel 1

 02/13/02  10:02:05
TRACE: File: appstat.c Line: 576
     Available Mail_Box number on channel 1

 02/13/02  10:02:05
TRACE: File: voice.c Line: 288
     The VOX file: startrec.vox was opened on channel 1

 02/13/02  10:02:05
TRACE: File: appstat.c Line: 587
     Start Record Menu was played on channel 1

 02/13/02  10:02:17
TRACE: File: appstat.c Line: 685
In IPTMAIL_CONNECTED_WAIT_START_REC on channel 1
      got event TDX_PLAY (0x81)

 02/13/02  10:02:17
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: End Of File

 02/13/02  10:02:17
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:02:17
TRACE: File: voice.c Line: 288
     The VOX file: startrec.vox was opened on channel 1

 02/13/02  10:02:17
TRACE: File: appstat.c Line: 685
In IPTMAIL_CONNECTED_WAIT_START_REC on channel 1
      got event TDX_PLAY (0x81)

 02/13/02  10:02:17
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: Receiving DTMF digits

 02/13/02  10:02:17
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:02:17
TRACE: File: voice.c Line: 445
     dx_getdig function sent successfully on channel 1

02/13/02  10:02:17
TRACE: File: appstat.c Line: 685
In IPTMAIL_CONNECTED_WAIT_START_REC on channel 1
      got event TDX_GETDIG (0x83)
```

```
 02/13/02  10:02:17
TRACE: File: appstat.c Line: 706
     Received digits: <2> on channel 1

 02/13/02  10:02:17
TRACE: File: voice.c Line: 355
     The VOX file: MB121.vox was opened on channel 1

02/13/02  10:02:22
TRACE: File: appstat.c Line: 850
In IPTMAIL_CONNECTED_RECORD_MSG on channel 1
      got event TDX_RECORD (0x82)

 02/13/02  10:02:22
TRACE: File: appstat.c Line: 866
     Recorder stopped on channel 1.
     Reason: Specific digit received

 02/13/02  10:02:22
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:02:22
TRACE: File: voice.c Line: 288
     The VOX file: stoprec.vox was opened on channel 1

02/13/02  10:02:37
TRACE: File: appstat.c Line: 975
In IPTMAIL_CONNECTED_STOP_REC on channel 1
      got event TDX_PLAY (0x81)

02/13/02  10:02:37
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: Receiving DTMF digits

 02/13/02  10:02:37
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:02:37
TRACE: File: voice.c Line: 445
     dx_getdig function sent successfully on channel 1

 02/13/02  10:02:37
TRACE: File: appstat.c Line: 975
In IPTMAIL_CONNECTED_STOP_REC on channel 1
      got event TDX_GETDIG (0x83)

 02/13/02  10:02:37
TRACE: File: appstat.c Line: 994
     Received digits: <4> on channel 1

 02/13/02  10:02:37
TRACE: File: voice.c Line: 288
     The VOX file: MB121.vox was opened on channel 1

 02/13/02  10:02:37
TRACE: File: appstat.c Line: 1053
     Play Recorded file on channel 1
```

```
 02/13/02  10:02:41
TRACE: File: appstat.c Line: 1184
In IPTMAIL_CONNECTED_LISTEN_MY_REC on channel 1
      got event TDX_PLAY (0x81)

02/13/02  10:02:41
TRACE: File: appstat.c Line: 1200
     Player stopped on channel 1.
     Reason: End Of File

 02/13/02  10:02:41
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:02:41
TRACE: File: voice.c Line: 288
     The VOX file: stoprec.vox was opened on channel 1

 02/13/02  10:02:41
TRACE: File: appstat.c Line: 1214
     Stop Record Menu was played on channel 1

 02/13/02  10:02:53
TRACE: File: appstat.c Line: 975
In IPTMAIL_CONNECTED_STOP_REC on channel 1
      got event TDX_PLAY (0x81)

 02/13/02  10:02:53
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: Receiving DTMF digits

 02/13/02  10:02:53
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:02:53
TRACE: File: voice.c Line: 445
     dx_getdig function sent successfully on channel 1

 02/13/02  10:02:53
TRACE: File: appstat.c Line: 975
In IPTMAIL_CONNECTED_STOP_REC on channel 1
      got event TDX_GETDIG (0x83)

 02/13/02  10:02:53
TRACE: File: appstat.c Line: 994
     Received digits: <3> on channel 1

 02/13/02  10:02:53
TRACE: File: voice.c Line: 288
     The VOX file: saveconfirm.vox was opened on channel 1

 02/13/02  10:02:53
TRACE: File: appstat.c Line: 1037
     Play Save_Confirm and MainMenu started on channel 1

 02/13/02  10:03:05
TRACE: File: appstat.c Line: 313
IPTMAIL_CONNECTED_WAIT_SIG on channel 1
      got event TDX_PLAY (0x81)
```

```
 02/13/02  10:03:05
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: Receiving DTMF digits

 02/13/02  10:03:05
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:03:05
TRACE: File: voice.c Line: 445
     dx_getdig function sent successfully on channel 1

 02/13/02  10:03:05
TRACE: File: appstat.c Line: 313
IPTMAIL_CONNECTED_WAIT_SIG on channel 1
      got event TDX_GETDIG (0x83)

02/13/02  10:03:05
TRACE: File: appstat.c Line: 334
     Received digits: <2> on channel 1

 02/13/02  10:03:05
TRACE: File: voice.c Line: 288
     The VOX file: listenmenu.vox was opened on channel 1

 02/13/02  10:03:17
TRACE: File: appstat.c Line: 1305
In IPMAIL_CONNECTED_START_LISTEN on channel 1
      got event TDX_PLAY (0x81)

 02/13/02  10:03:17
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: Receiving DTMF digits

 02/13/02  10:03:17
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:03:17
TRACE: File: voice.c Line: 445
     dx_getdig function sent successfully on channel 1

 02/13/02  10:03:17
TRACE: File: appstat.c Line: 1305
In IPMAIL_CONNECTED_START_LISTEN on channel 1
      got event TDX_GETDIG (0x83)

 02/13/02  10:03:17
TRACE: File: appstat.c Line: 1325
     Received digits: <121> on channel 1

 02/13/02  10:03:17
TRACE: File: appstat.c Line: 1388
     Available Mail_Box number on channel 1

 02/13/02  10:03:17
TRACE: File: voice.c Line: 288
     The VOX file: MB121.vox was opened on channel 1

 02/13/02  10:03:17
```

```
TRACE: File: appstat.c Line: 1401
     Play Recorded file on channel 1

 02/13/02  10:03:21
TRACE: File: appstat.c Line: 1495
In IPTMAIL_CONNECTED_LISTEN on channel 1
      got event TDX_PLAY (0x81)

 02/13/02  10:03:21
TRACE: File: appstat.c Line: 1511
     Player stopped on channel 1.
     Reason: End Of File

02/13/02  10:03:21
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:03:21
TRACE: File: voice.c Line: 288
     The VOX file: stoplisten.vox was opened on channel 1

 02/13/02  10:03:32
TRACE: File: appstat.c Line: 1620
In IPTMAIL_CONNECTED_STOP_LISTEN on channel 1
      got event TDX_PLAY (0x81)

 02/13/02  10:03:32
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: End Of File

 02/13/02  10:03:32
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:03:32
TRACE: File: voice.c Line: 288
     The VOX file: stoplisten.vox was opened on channel 1

 02/13/02  10:03:33
TRACE: File: appstat.c Line: 1620
In IPTMAIL_CONNECTED_STOP_LISTEN on channel 1
      got event TDX_PLAY (0x81)

 02/13/02  10:03:33
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: Receiving DTMF digits

 02/13/02  10:03:33
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:03:33
TRACE: File: voice.c Line: 445
     dx_getdig function sent successfully on channel 1

 02/13/02  10:03:33
TRACE: File: appstat.c Line: 1620
In IPTMAIL_CONNECTED_STOP_LISTEN on channel 1
      got event TDX_GETDIG (0x83)
```

```
02/13/02  10:03:33
TRACE: File: appstat.c Line: 1639
     Received digits: <*> on channel 1

 02/13/02  10:03:33
TRACE: File: voice.c Line: 288
     The VOX file: mainmenu.vox was opened on channel 1

 02/13/02  10:03:33
TRACE: File: appstat.c Line: 1693
     Main Menu was played on channel 1

 02/13/02  10:03:38
TRACE: File: appstat.c Line: 313
IPTMAIL_CONNECTED_WAIT_SIG on channel 1
      got event TDX_PLAY (0x81)

 02/13/02  10:03:38
TRACE: File: voice.c Line: 49
     Player stopped on channel 1.
     Reason: Receiving DTMF digits

 02/13/02  10:03:38
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

02/13/02  10:03:38
TRACE: File: voice.c Line: 445
     dx_getdig function sent successfully on channel 1

 02/13/02  10:03:38
TRACE: File: appstat.c Line: 313
IPTMAIL_CONNECTED_WAIT_SIG on channel 1
      got event TDX_GETDIG (0x83)

 02/13/02  10:03:38
TRACE: File: appstat.c Line: 334
     Received digits: <*> on channel 1

 02/13/02  10:03:38
TRACE: File: voice.c Line: 288
     The VOX file: thankyou.vox was opened on channel 1

 02/13/02  10:03:43
TRACE: File: appstat.c Line: 1796
In IPTMAIL_STOP_CONNECTION on channel 1
      got event TDX_PLAY (0x81)

 02/13/02  10:03:43
TRACE: File: appstat.c Line: 1812
     Player stopped on channel 1.
     Reason: End Of File

02/13/02  10:03:43
TRACE: File: voice.c Line: 508
     File was closed  on channel 1

 02/13/02  10:03:43
TRACE: File: appstat.c Line: 1819
     Player was stopped, and disconnecting the call on channel 1.

 02/13/02  10:03:43
```

```
TRACE: File: appmain.c Line: 417
     Making UnRoute on channel 1

 02/13/02  10:03:43
TRACE: File: appmain.c Line: 425
     Dropping IP call on channel 1.

 02/13/02  10:03:43
TRACE: File: appstat.c Line: 1901
In IPTMAIL_DISCONNECTED on channel 1
      got event GCEV_DROPCALL (0x805)

02/13/02  10:03:43
TRACE: File: gateip.c Line: 196
     gc_Extension(RTCPINFO,CALLDURATION) function was called on cahnnel 1

 02/13/02  10:03:43
TRACE: File: appstat.c Line: 1901
In IPTMAIL_DISCONNECTED on channel 1
      got event GCEV_EXTENSION (0x868)

 02/13/02  10:03:43
INFO: File: gateip.c Line: 244
     Got extension data RTCPINFO:
     timestamp 1092704,
     tx_packets 3653,
     tx_octets 73060
     send_indication 1

02/13/02  10:03:43
INFO: File: gateip.c Line: 233
     Got extension data CALLDURATION: 130

 02/13/02  10:03:43
TRACE: File: appstat.c Line: 1968
     Releasing Call on channel 1.

 02/13/02  10:03:46
TRACE: File: appmain.c Line: 39
     Voice Device was closed on channel 1.

 02/13/02  10:03:46
TRACE: File: appmain.c Line: 48
     IP Line Device was closed on channel 1.
```

Index