IP Gateway (Global Call) Demo Guide

for Linux and Windows

05-1662-005

COPYRIGHT NOTICE

Publication Date: July, 2005

Intel Converged Communications, Inc.
1515 Route 10
Parsippany NJ 07054

For **Technical Support**, visit the Intel Telecom Support Resources website:
http://developer.intel.com/design/telecom/support/

For **Products and Services Information**, visit the Intel Telecom Products website:
http://www.intel.com/design/network/products/telecom

For **Sales Offices**, visit the Where to Buy Intel Telecom Products page:
http://www.intel.com/buy/networking/telecom.htm

Table of Contents

List of Tables

List of Figures

1. About This Guide

This section describes the purpose of this guide, the intended audience, and references to other documents that may be useful to the user.

1.1. Purpose

This guide describes the operation of the IP Gateway (Global Call) demo.

1.2. Intended Audience

This guide is intended for application developers who will be developing a PSTN-IP gateway application using the Global Call API.

Developers should be familiar with the C programming language and either the Windows or Linux programming environment.

1.3. Related Documents

See the following for more information:

- The *Release Update* for your system release for information on problems fixed, known problems, workarounds, compatibility issues, and last minute updates not documented in the published information.
- The appropriate *Configuration Guide* for your hardware (Intel NetStructure IPT Series or DM/IP Series board) and operating system
- *Global Call API Software Reference Guide* and the *Global Call Application Developer's Guide*
- *Global Call IP Technology User's Guide*
- *http://developer.intel.com/design/telecom/support/* (for technical support)
- *http://www.intel.com/design/network/products/telecom* (for product information)

2. Demo Description

2.1. About the Demo

The IP Gateway (Global Call) demo is a host-based application that demonstrates using the Global Call API to build a PSTN–IP gateway. The demo source code can be used as sample code for those who want to begin developing an application from a working application. The demo is not designed to implement a complete gateway, lacking features such as least-cost routing, etc.

The IP Gateway (Global Call) demo is a cross-OS demo, running under the Windows* or Linux* environments. Most of the differences in the environments are handled directly by the programming interface and are transparent to the user. Other differences, due to inherent differences in the operating systems, are handled by the Platform Dependency Library (PDL).

For more information about the PDL refer to the source code in the *pdl_win* or *pdl_linux* directories.

2.2. Choosing Channels

When a call comes from the PSTN, the call is answered by a PSTN line device. During initialization, the PSTN channel was associated with a specific IP line device, so the call is connected to the IP line device that is associated with this PSTN line device.

When a call arrives from the IP network, there is no direct association of a channel, since there are no individual physical connections for the IP channels. The call is answered by a line device. During initialization, the line device was associated with a specific PSTN line device. The Global Call API tells the IP Gateway (Global Call) demo which PSTN channel is associated with this IP channel. The application then connects the IP call to the appropriate PSTN channel.

3. System Requirements

This chapter discusses the system requirement for running the IP Gateway (Global Call) demo. It contains the following topics:

- Hardware Requirements
- Software Requirements

3.1. Hardware Requirements

To run the IP Gateway (Global Call) demo, you need:

- One of the following:
  - Intel NetStructure® DM/IP Series board
  - Intel NetStructure® IPT Series board
    - an IPT Series board also requires an Intel NetStructure® DM/V-A series board for PSTN connection
- IP Network cable

For other hardware requirements, such as memory requirements, see the *Release Guide* for your system release.

3.2. Software Requirements

To run the IP Gateway (Global Call) demo, you need the Intel® Dialogic® System Release 6.x for the Linux* or Windows* Operating Systems on Intel Architecture. For a list of operating system requirements and supported compilers see the *Release Guide* for your system release.

4. Preparing to Run the Demo

This chapter discusses how to prepare to run the IP Gateway (Global Call) demo. It provides information about the following topics:

- Connecting to External Equipment
- Editing Configuration Files

4.1. Connecting to External Equipment

There are two possible hardware configurations for the IP Gateway (Global Call) demo:

- Intel NetStructure® DM/IP series board(s) with on-board NIC and a PSTN connection on the front end
- Intel NetStructure® IPT series board(s) with on-board NIC connected to an Intel NetStructure® DM/V-A series board as the PSTN interface

The following diagrams illustrate the possible hardware configurations.

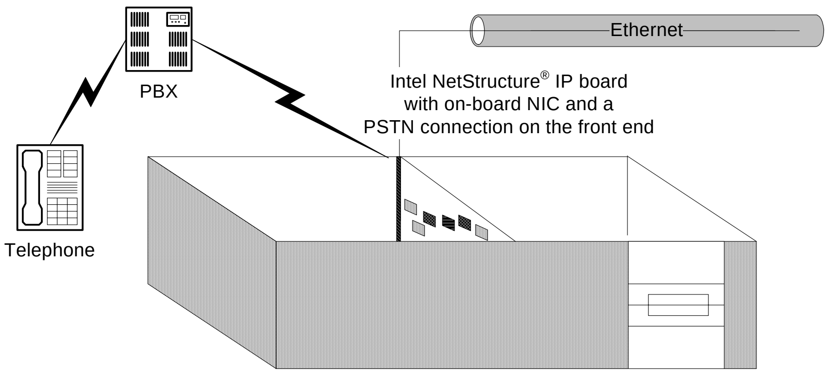
PBX
Ethernet
Intel NetStructure® IP board
with on-board NIC and a
PSTN connection on the front end
Telephone

Figure 1. Hardware Configuration with Onboard NIC and PSTN Front End

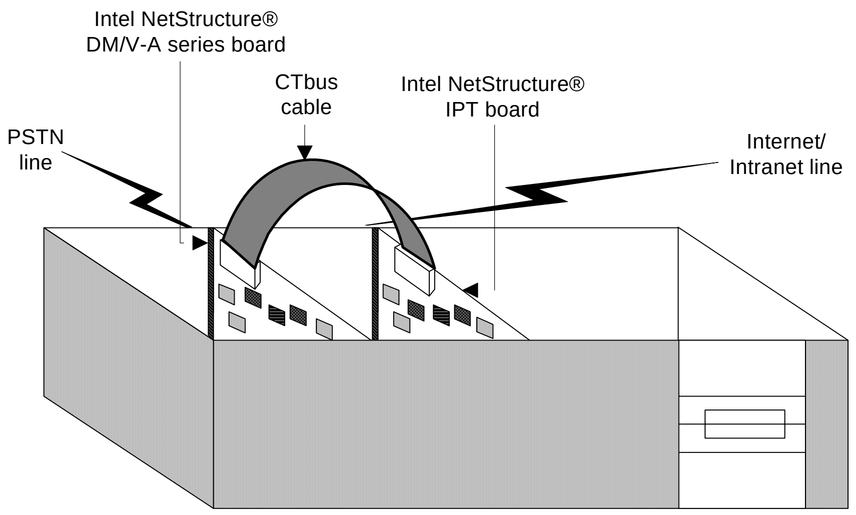

Figure 2. Hardware Configuration with Onboard NIC and separate PSTN board

The IP Gateway (Global Call) demo allows you to connect to gateways on an IP network and establish voice calls via the IP network. It also allows you to connect to H.323 terminals on the IP network and connect a call from the terminal to a telephone via one of the gateways. Figure 3 shows a typical topology for demonstrating the capabilities of the IP Gateway (Global Call) demo. Note that the two PBXs that are shown can be a single PBX. Also note that more than one PSTN line can be connected to a single gateway.

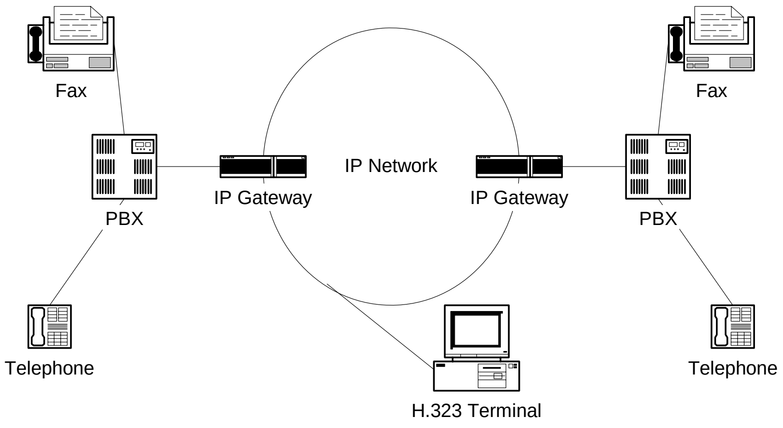

Figure 3. Typical Topology

4.2. Editing Configuration Files

This section discusses how to configure the demo for your system. It contains the following topics:

- Configuration file location
- Editing the *gateway_r4.cfg* File

4.2.1. Configuration File Location

Before running the IP Gateway (Global Call) demo, modify the *gateway_r4.cfg* file to reflect your system environment. Use a text editor and open the file from:

- Windows:
  $(INTEL_DIALOGIC_DIR)\samples\ipt_demos\gateway_r4\Release
- Linux:
  $(INTEL_DIALOGIC_DIR)/ipt_demos/gateway_r4/Release

where *$(INTEL_DIALOGIC_DIR)* identifies the base directory where the Intel Dialogic System Release was installed.

4.2.2. Editing the *gateway_r4.cfg* File

Below is an example of the *gateway_r4.cfg* file. Update the following information:

ipProtocol
: The IP Protocol used for opening the IP Line devices, values: H323, SIP, both

Channel
: Channels defined by this section of the file - may be individual channel or a range of channels

Source
: Source address

Destination
: Destination address

RemotePhoneNumber
: Destination phone number to call. Transferred during call establishment to target gateway.

LocalPhoneNumber
: The number used for PSTN calls

pstnProtocol
: PSTN protocol to use

DTMFmode
: One of the following: OutOfBand, inband, rfc2833

AudioRxCodecs
: Capability for receive audio codecs. The following capabilities are defined:

- CoderType - preferred coder. Recognized coders are:
    - g711Alaw
    - g711Mulaw
    - gsm
    - gsmEFR
    - g723_5_3k
    - g723_6_3k
    - g729a
    - g729ab

- CoderFramesPerPkt - frames per packet for the selected coder
- CoderVAD - Voice Activity Detection on/off

AudioTxCodecs

Capability for transmit audio codecs. See AudioRxCodecs for a complete description.

DataCodecs

Capability for fax codecs. The demo currently support T38 only.

MediaAlarmLostPackets

Indicates that the percentage of packets lost during a call exceeded its threshold value

- Threshold - defines when a Quality of Service (QoS) parameter is in a fault condition. A fault occurs when the result of a measurement of a QoS parameter crossed the Threshold value. Default = 20.
- DebounceOn - the time during which faults are measured (in msec., must be a multiple of Interval). Default = 10000.
- DebounceOff - the dime during which successes are measured (in mesc., must be a multiple of Interval). Default = 10000.
- Interval - the amount of time between two QoS parameter measurements (in multiples of 100 msec). Default = 1000.
- PercentSuccess - the threshold of successes during the DebounceOn time (expressed as a percentage of successes). Default = 60.
- PercentFail - the threshold of failures during the DebounceOn time (expressed as a percentage of failures). Default = 40.

MediaAlarmJitter

Indicates that the jitter (as defined in RFC 1889) exceeded its threshold value

- Threshold - Default = 60.
- DebounceOn - Default = 20000.
- DebounceOff - Default = 60000.
- Interval - Default = 5000.
- PercentSuccess - Default = 60.
- PercentFail - Default = 40.

Display

Display information passed to destination gateway during call establishment

IPT_UUI

User to User Information string. Information sent before Connected state.

UII
: User Input Indication string to send

NonStdParm
: Non-standard parameter data to send

NonStdCmd
: Non-standard command string to send

ObjId
: Object ID

Q931Facility
: Facility data to send on the Q.931 channel

DTMF
: DTMF mode. Possible options: OutOfBand, inband, rfc2833

enableRegestration
: Register with gatekeeper

TTL
: Time-to-live parameter (in seconds)

Protocol
: Call control protocol. Possible values: h323, SIP, both

max_hops
: Maximum number of router hops

regServerAddress
: Gatekeeper IP address. Use 0.0.0.0 as the default address for discovering the GK

NonStdRasCmd
: Non-standard RAS command string to send

RasObjId
: RAS object ID

Alias
: Possible alias types: 1 = string, 2 = IP address, 3 = H323 ID, 4 = phone, 5 = URL, 6 = EMail

The following is an example of a configuration file.

```
########################################################################################
# Telephony Protocol :
#   For ANAPI(Analog Front End) use the root file name of the analog protocol file for
your country or telephone network)
#   For ICAPI (Digital Front End) use the root file name of the country dependent
parameter <.cdp> file
# IP Protocol :
#    The IP Protocol used for  opening the IP Line devices, values: H323, SIP, both
#
# DTMFmode
#    possible options:
#         OutOfBand, inband, rfc2833
#
#
# Capability for audio codecs:
#    g711Alaw
#    g711Mulaw
#    gsm
#    gsmEFR
#    g723_5_3k
#    g723_6_3k
#    g729a
#    g729ab
#
#  Capability for data codecs:
#    t38
#
#    Note: if you want to run the demo with coder g729 use:
#    g729a for running with VAD disable
#    and 729ab for running with VAD enable
#
# Caution:
#     If capability is g711Alaw /Mulaw  ==> FramesPerPkt = 10,20,30.
#                                           G711 frame per packet defines the packet
size in milliseconds
#     If capability is g723_5_3k / 6_3k ==> FramesPerPkt = 1, 2, 3 .
#                                           FrameSize isn't needed, default= 30ms.
#     If capability is gsm              ==> FramesPerPkt = 1, 2, 3 .
#                                           FrameSize isn't needed, default= 20ms.
#     If capability is gsmEFR           ==> FramesPerPkt = 1, 2, 3 .
#                                           FrameSize isn't needed, default= 20ms.
#     If capability is g729a            ==> FramesPerPkt = 3, 4 .
#                                           FrameSize isn't needed, default= 10ms.
#                                           VAD disable, the VAD parameter is ignored
#     If capability is g729ab            ==>FramesPerPkt = 3, 4 .
#                                           FrameSize isn't needed, default= 10ms.
#                                           VAD enable, the VAD parameter is ignored
#
########################################################################################
ipProtocol = H323
Channel = 1-120
{
   Source = NAME:Intel Corp.
   Destination = 0.0.0.0
   RemotePhoneNumber = 23
   LocalPhoneNumber = 26
   pstnProtocol = isdn
   DTMFmode = OutOfBand

   AudioRxCodecs
   {
          CoderType = g711Mulaw
          CoderFramesPerPkt = 30
          CoderVAD = 0
   }

   AudioTxCodecs
   {
          CoderType = g711Mulaw
          CoderFramesPerPkt = 30
          CoderVAD = 0
   }

   DataCodecs
   {
          CoderType = t38
   }

   MediaAlarmLostPackets
   {
        Threshold       = 20    # Threshold value
       DebounceOn      = 10000 # Threshold debounce ON
       DebounceOff     = 10000 # Threshold debounce OFF
       Interval        = 1000  # Threshold Time Interval (ms)
       PercentSuccess  = 60    # Threshold Success Percent
       PercentFail     = 40    # Threshold Fail Percent
   }

   MediaAlarmJitter
   {
       Threshold            = 60    # Threshold value
       DebounceOn           = 20000 # Threshold debounce ON
       DebounceOff          = 60000 # Threshold debounce OFF
       Interval             = 5000  # Threshold Time Interval (ms)
       PercentSuccess       = 60    # Threshold Success Percent
       PercentFail          = 40    # Threshold Fail Percent
   }

# MediaAlarmResetAlarmState               = 0

   Display = GATEWAY_Chan1
   IPT_UUI = User_to_User_1
   UII = 12345
   NonStdParm = NSP_Chan1
   NonStdCmd = NSC_Chan1
   ObjId = 2 16 840 1 113741
   Q931Facility = facility 01
   DTMF = 1
}

#values - 1 -to enable board regestration , 0 not enabling board regestration
enableRegestration = 0
board = 1-1
{
     # time to live in seconds
     TTL = 60
     # possible values: h323, SIP, both
     Protocol  = h323
     max_hops = 20
     # use 0.0.0.0 as the default address for descovering the GK
     regServerAddress    = 10.242.214.45
     NonStdRasCmd = NSC_Chan1
     RasObjId = Intel
     # possible alias types: 1 = string, 2 = IP address, 3 = H323 ID, 4 = phone, 5 =
URL, 6 = EMail
     Alias = 1
     {
          AliasType = 3
          AliasName = intel
     }
     Alias = 2
     {
          AliasType = 4
          AliasName = 1111
     }
     Prefix = 1
     {
          PrefixType = 3
          PrefixName = pmac
     }
     Alias = 3
     {
          AliasType = 4
          AliasName = 2222
     }
}
```

5. Running the Demo

This chapter discusses how to run the IP Gateway (Global Call) demo. It contains the following topics:

- Starting the Demo
- Demo Options
- Using the Demo
- Stopping the Demo

5.1. Starting the Demo

Windows

From a command prompt, change directories to:
$(INTEL_DIALOGIC_DIR)\samples\\ipt_demos\gateway_r4\Release

Type `gateway_r4` at the command prompt to run the IP Gateway (Global Call) demo using the default settings.

Linux

Change directory to:
$(INTEL_DIALOGIC_DIR)\ipt_demos\gateway_r4/Release

Type `gateway_r4` to run the IP Gateway (Global Call) demo using the default settings.

5.2. Demo Options

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

Table 1. Command Line Switches

Switch	**Action**	**Default**
-c <*filename*>	Configuration file name	*gateway_r4.cfg*
-d	Sets Debug Level (0-4): 0-FATAL: used when one or more channels are deadlocked. 1-ERROR: used when the application receives a failure which doesn't cause the channel to be deadlocked. 2-WARNING used when some problem or failure occurred without affecting the channel's usual action. 3-TRACE used at the start of the application entrance or the start of any function. 4-INFO prints data related to a specific action. **NOTE:** Debug level is inclusive; higher levels include all lower levels.	0 - FATAL
-f	Identifies the front end: 0 = analog 1 = digital T-1 2 = digital E-1	0
-h or ?	Prints the command syntax to the screen	Off

Switch	**Action**	**Default**
-l <n,…>	Printouts will be printed into channel log files. If ‘all’ follows the –l, log files will be created for all available channels. If a list of channels in the following format: C1-C2, C3-C4, C5 (e.g., 1-10,112-150,314) follows the –l, log files are created for the channel ranges or specific channels specified in the list. If “–l” option is not used all prints go to the stdout, for the first 2 channels only (to keep from overloading the CPU, and more convenient for viewing printouts).	Disabled
-n	Sets the number of channels	The lesser of **PSTN Devices** and **IP Devices**
-p	0-Disable dialing 1-Enable dialing Used for testing purposes, or if running the demo on a machine that does not have all of the necessary external connections	-p1
-q	Enables the Quality of Service feature	Disabled
-r	Sets the number of rings before answering the call on the PSTN	2
-s	0-Disable DNIS 1-Enable DNIS Used for testing purposes, or if running the demo on a machine that does not have all of the necessary external connections	0

5.3. Using the Demo

The demo always waits for input from the keyboard. While the demo is running, you can enter any of the following commands:

Table 2. Runtime Keyboard Commands

Command	**Function**
c or C	Print channel information
d<n> or D<n>	Change debug level during runtime
f or F	Send Q.931 facility information
n or N	Send non-standard command
q or Q	Terminates the application
r or R	Sends non-standard RAS
s or S	Unregister with a Gatekeeper
t or T	Sends DTMF
u or U	Sends UII (User Input Indication)

5.4. Stopping the Demo

The IP Gateway (Global Call) demo runs until it is terminated. Press “q” or “Q” to terminate the demo application.

6. Demo Details

This chapter discusses the IP Gateway (Global Call) demo in more detail. It contains the following topics:

- Files Used by the Demo
- Handling an Incoming Call
- Programming Model
- Initializations
- Event Handling
- Demo State Machine

6.1. Files Used by the Demo

6.1.1. Demo Source Files

In Windows the following files are located in *$(INTEL_DIALOGIC_DIR)\samples\ipt_demos\gateway_r4*.

In Linux the following files are located in *$(INTEL_DIALOGIC_DIR)/ipt_demos/gateway_r4.*

Table 3. Source Files Used by the IP Gateway (Global Call) Demo

Filename	**Description**	**OS**
gatedefs.h	Gateway definitions	Both
gateip.c	IP communication functions	Both
gateip.h	Function prototype for gateip.c	Both
gatemain.c	Main file (including MAIN loop)	Both

Filename	Description	OS
gatepars.c	The demo configuration file parsing functions	Both
gatepars.h	Function prototype for gatepars.c	Both
gatepstn.c	PSTN-specific functions	Both
gatepstn.h	Function prototype for gatepstn.c	Both
gatestate.c	State machine functions	Both
gatestate.h	Function prototype for gatestat.c	Both
gatestrc.h	Demo structure (including Main Structure Session)	Both
gatevars.h	Global variables	Both
gateway_r4	Linux executable	Linux
gateway_r4.cfg	Config file	Linux
gateway_r4.dsp	Visual C++ project file	Windows
gateway_r4.dsw	Visual C++ project workspace	Windows
gateway_r4_ version.c	Demo version information	Both
incfile.h	Function prototype for Global Call and R4 functions.	Both
main.h	Function prototype for gatemain.c	Both
makefile	Linux compilation file	Linux
mediaalarms.c	QoS functions	Both
mediaalarms.h	Function prototype for mediaalarms.c	Both
register.c	RAS functions	Both

Filename	**Description**	**OS**
register.h	Function prototype for register.c	Both
Release\ gateway_r4.cfg	Demo configuration file	Windows
Release/ gateway_r4.cfg	Demo configuration file	Linux
Release\ gateway_r4.exe	Executable	Windows
Release/ gateway_r4.exe	Executable	Linux

6.1.2. Utility Files

In Windows the following files are located *$(INTEL_DIALOGIC_DIR)\samples\ipt_demos\Shared*

In Linux the following files are located in *$(INTEL_DIALOGIC_DIR)/ipt_demosShared*.

Filename	**Description**	**OS**
libdbg.c	Debugging functions	Both
libdbg.h	Function prototype for libdbg.c	Both
libdefs.h	#DEFINE inclusions	Both
Release/libutil.a	Compiled Utility library	Linux
Makefile	Compilation file	Linux
util.dsp	Utility library Visual C project file	Windows
util.dsw	Utility library Visual C workspace	Windows
util_version.c	Utility library version information	Both

Filename	Description	OS
Release\util.lib	Compiled Utility library	Windows

6.1.3. PDL Files

In Windows the following files are located in
$(INTEL_DIALOGIC_DIR)\samples\ipt_demosShared

In Linux the following files are located in
$(INTEL_DIALOGIC_DIR)/ipt_demos/Shared

Filename	Description	OS
pdl_win\iptransport.cpp	PDL IP transport functions	Windows
pdl_win\iptransport.h	Function prototype for iptransport.cpp	Windows
pdl_win\pdl.c	Platform dependency functions	Windows
pdl_win\pdl.h	Function prototype for pdl.c	Windows
pdl_win\pdl_version.c	PDL version information	Windows
pdl_win\pdl_win.dsp	PDL Visual C project file	Windows
pdl_win\pdl_win.dsw	PDL Visual C workspace	Windows
pdl_win\Release\ pdl_win.lib	Compiled PDL library	Windows
/pdl_linux/ iptransport.cpp	PDL IP transport functions	Linux
/pdl_linux/iptransport.h	Function prototype for iptransport.cpp	Linux
/pdl_linux/libpdl.a	Compiled PDL library	Linux
/pdl_linux/makefile.pdl	Compilation file	Linux

Filename	Description	OS
/pdl_linux/pdl.c	Platform dependency functions	Linux
/pdl_linux/pdl.h	Function prototype for pdl.c	Linux
/pdl_linux/ pdl_linux_version.c	PDL version information	Linux

6.2. Handling an Incoming Call

This section discusses how the demo application handles incoming calls. It contains the following topics:

- Receiving a Call
- Handling a PSTN Call
- Handling an IP Call

6.2.1. Receiving a Call

The demo can receive calls from either the PSTN or the IP network. The demo uses a configuration file (*gateway_r4.cfg*) to determine parameters that are associated with a particular call. The configuration file allows you to configure different channels with different properties. See *Section 4.2. Editing Configuration* File for a more detailed description of the *gateway_r4.cfg* file, as well as a description of the different configuration properties.

6.2.2. Handling a PSTN Call

A call that arrives from the PSTN needs to be routed to either a destination PSTN number (via another gateway) or to an H.323 terminal. The demo uses the *gateway_r4.cfg* file to determine the destination IP address as well as the (optional) destination PSTN number (remote phone number). The IP Gateway (Global Call) demo initiates an IP (H.323) call to the destination IP address. If the configuration file indicates a PSTN destination number then that number is passed to the destination gateway during the call establishment procedure.

Once the destination gateway has answered the H.323 call, the IP Gateway (Global Call) demo connects the PSTN call to the IP call. An audio path is now established between the PSTN call and the destination IP station. For more details see section 6.5. Event Mechanism .

6.2.3. Handling an IP Call

A call that arrives from the IP network needs to be routed to a PSTN number. That number may arrive as part of the call establishment procedure (if the call was originated by another IP Gateway for example). If the destination number arrived during call establishment, then the IP Gateway (Global Call) demo uses that number to call the PSTN. If no destination number was included in the call establishment procedure, then the IP Gateway (Global Call) demo uses the *gateway_r4.cfg* file to determine the destination number to call (local phone number). Once the IP Gateway (Global Call) demo answers and connects the call on the IP network, it initiates (dial out) a call on the PSTN line and connects the two calls. This allows the calling party to hear the call progress tones on the local PSTN. For more details see section 6.5. Event Mechanism .

6.3. Programming Model

The IP Gateway (Global Call) Object Oriented demo operates with two threads, as shown in Figure 4.

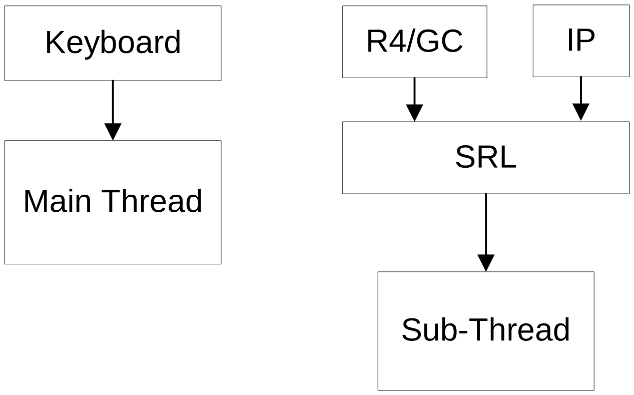

Figure 4. Programming Model

The threads are created as follows:

- The first (main) thread is created by the demo application to get the keyboard input.
- The second thread is an SRL thread, created as a result of the demo application calling **sr_enblhdlr**( ) in Windows. In Linux, the thread must be explicitly created. All Global Call events are received through the SRL.

6.4. Initializations

The application **main()** function calls **gateInitialize()**, which does the following:

1. Calls **checkArg( )** to check for command line parameters and handle them accordingly.
2. Calls **IPTResetSession()** to reset the demo data structures and initialize all channels' states to INIT.
3. Calls **ClearAllBoards()** to reset the board structures to default values.
4. Calls **gateConfiguration()** to read information from the configuration file (*gateway_R4.cfg* or other CFG file determined by the user) and update the ConfigFileParm in the Session data structure.
5. Calls **gc_Start( )** to open all configured, call control libraries.
6. Calls **printAllLibs()** to print library status (open or failed).
7. Sets-up the call-back handler, **PDLsr_enbhdlr( )**. The callback handler handles events that it receives from the SRL library. For more details see *Section 6.5.2. Handling SRL Events*.
8. Calls **pstnGetVOXChannels( )** which checks how many available PSTN voice channels there are by doing the following:
   - Gets number of PSTN boards, by calling **PDLsr_getboardcnt( )**.
   - For each board that was found:
     - Calls **dx_Open( )** to open an analog board, or **dt_Open( )** to open a digital board.
     - Calls **ATDV_SUBDEVS( )** to get the number of channels on the board.

    - Calculates the logical board and channel and saves them into Session.pstnParams
    - Closes the board, by calling **dx_Close( )** or **dt_Close( )**.

9. Call **ipGetChannels( )** which checks how many available IP channels there are by doing the following:

    - Gets number of IP boards from #define MAX_IP_BOARDS in *gatedefs.h*

    - For each board that was found:
        - Calls **gc_OpenEx( )** to open the board
        - Calls **ATDV_SUBDEVS( )** to get the number of channels on the board
        - Calculates the logical board and channel and save them in Session.ipParams
        - Registers the board with the Gatekeeper by calling **boardRegistration( )**

10. Calls **getGateChannels( )** 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).

11. Calls **pstnOpenFrontEnd( )** which opens the PSTN channels by doing the following. For each channel:

    - Calls **gc_OpenEx ( )**, which returns the PSTN **LineDevH**, and saves it in Session.pstnParams

    - If the PSTN board is an analog board:
        - Calls **gc_LoadDxParm( )**
        - Calls **gc_GetVoiceH( )**, which returns the PSTN **VoiceH**, and saves it in Session.pstnParams

    - If the PSTN board is a digital board:
        - Calls **gc_OpenEx( )**, which returns the PSTN **LineDevH**, and saves it in the Session.pstnParams structure
        - Calls **gc_GetNetworkH( )**, which returns the PSTN **NetwH**, and saves it in Session.pstnParams

12. Calls **ipOpenDevices( )** which opens the IP channels by doing the following:

- Calls **gc_OpenEx( )** which opens all IP devices, returns the IP **LineDevH**, and saves it in Session.ipParams
- Saves the channel number in the global array HandleToChannel[ ] according to the **LineDevH** handle

13. The application **main()** function calls **waitForKey( )**, to receive keyboard input.

6.5. Event Mechanism

The IP Gateway (Global Call) demo uses the SRL mechanism to retrieve events. When an event occurs, SRL calls event handlers automatically. All events are received by the SRL and then passed to the **callback_hdlr()** function for handling.

In the initialization phase of the demo the **gateInitialize()** function sets up the call-back handler, by calling **PDLsr_enbhdlr()**.

6.5.1. Handling Keyboard Input Events

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

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

6.5.2. Handling SRL Events

When the R4/Global Call event is received, the **callback_hdlr( )** function performs the following:

1. Calls **gc_GetMetaEvent( )** to get the event
2. If the event is for a board, the application calls **rasProcessEvent( )** to process it.

3. Otherwise, the application calls **gc_GetUsrAttr( )** and then calls either **ipGetEvent( )** to process the IP event, or **pstnGetEvent( )** to process the PSTN event.

6.5.3. Handling Application Exit Events

Normal application exit events don't enter the SRL. The **main()** function calls **PDLSetApplicationExitPath()** before initialization. In Linux, this function sets the signals (SIGINT, SIGTERM, SIGABRT) for making the appropriate exit from the application. In Windows, this function enables the detection of CTRL_CLOSE_EVENT (closing the window).

6.6. Demo State Machine

The application waits for a *GCEV_UNBLOCKED* event in the GATE_INIT state. Upon receiving this event, the application calls **ag_getxmitslot()** for an analog PSTN board or **dt_getxmitslot()** for a digital PSTN board to get the transmit time slot (Xmitslot) for the PSTN device and saves it in the session.PSTNParams structure. The application then calls **gc_GetXmitSlot(VoiceH)** to get the transmit time slot (Xmitslot) for the IP device and saves it in the session.IPParams structure.

The application then calls **gc_WaitCall()** to set the conditions for processing an inbound call.

If the application receives *GCEV_TASKFAIL*, *GCEV_BLOCKED*, or *GCEV_OPENEX_FAIL*, it calls **endApplication()** to gracefully shut down the application.

If the application receives *GCEV_OPENEX*, it does nothing to avoid causing an error.

The state transitions to GATE_NULL.

6.6.1. Call Establishment from IP

This section describes what happens when a call is initiated from the IP network.

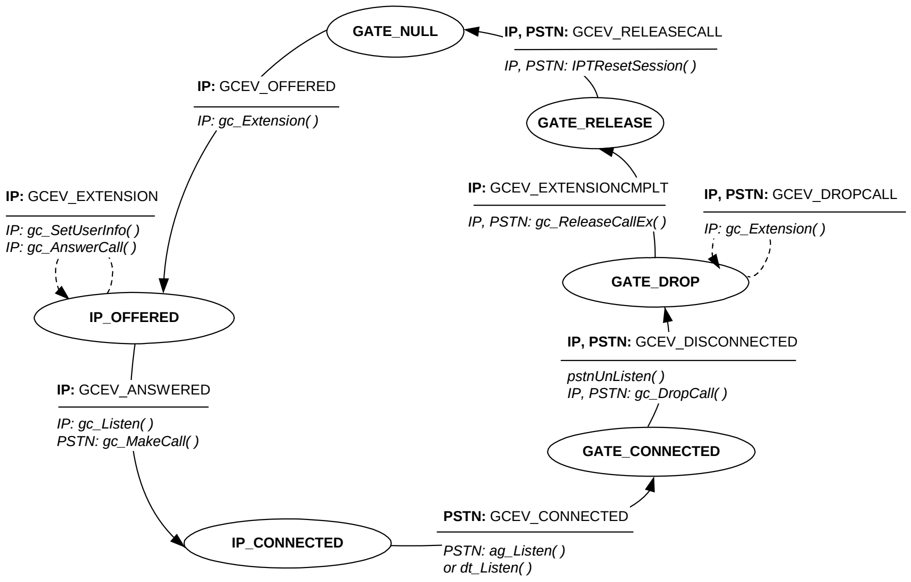

Figure 5. Call Establishment from IP

1. In GATE_NULL, the application receives *GCEV_OFFERED* from the IP side.

   The application checks if there is a conflict with PSTN side. If there is no conflict, the application calls **gc_Extension( )** to get coder and telephone number information from the IP side.

   The state transitions to IP_OFFERED.

2. In IP_OFFERED, the application waits for *GCEV_EXTENSION* which contains the coder and telephone number information.

   The application then calls **gc_SetUserInfo( )** and **gc_AnswerCall( )**.

   When the application receives *GCEV_ANSWERED* from the IP side, the application calls **gc_Listen( )**, to tell the IP line device to listen to the PSTN time slot. The application calls **gc_MakeCall( )** for the PSTN side to set up the call on the PSTN side.

   The state transitions to IP_CONNECTED.

3. In IP_CONNECTED, when the application receives *GCEV_CONNECTED* from the PSTN side, the application calls **pstnListen( )**, which in turn calls **ag_Listen( )** or **dt_Listen( )** (**ag** for analog; **dt** for digital) to tell the PSTN line device to listen to the IP time slot

   The state transitions to GATE_CONNECTED

6.6.2. Call Establishment from PSTN

This section describes what happens when a call is initiated from the PSTN network.

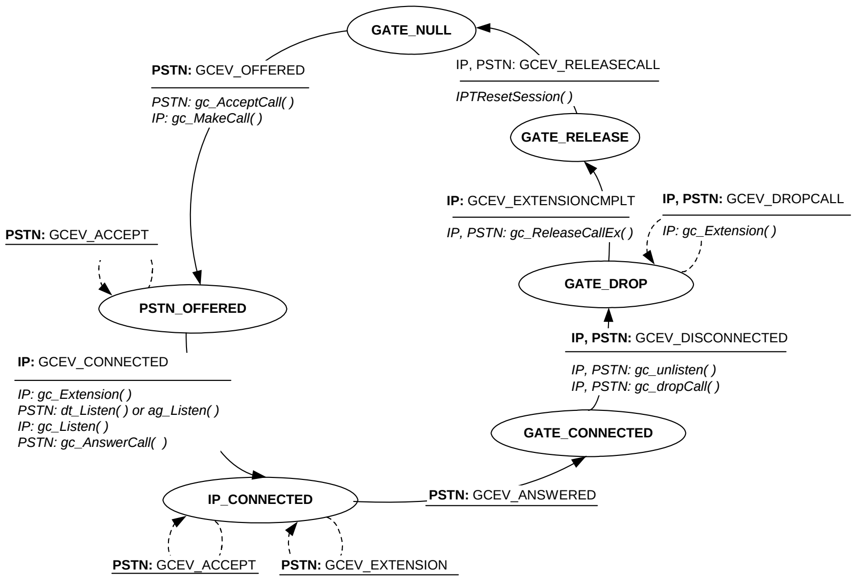

Figure 6. Call Establishment from PSTN

1. In GATE_NULL, when the application receives *GCEV_OFFERED* from the PSTN side, the application calls **gc_AcceptCall( )** for the PSTN and **gc_MakeCall( )** for the IP side.

   The state transitions to PSTN_OFFERED

2. In PSTN_OFFERED the application waits for *GCEV_CONNECTED* from the IP side.

   When the application receives *GCEV_CONNECTED* it calls:

- **gc_Extension( )** to get the call information from the IP side
- **gc_Listen( )** to tell the IP line device to listen to the PSTN time slot
- **pstnListen( )** which calls **ag_Listen( )** or **dt_Listen( )** (**ag** for analog; **dt** for digital) to tell the PSTN line device to listen to the IP time slot
- **gc_AnswerCall( )** to answer the call on the PSTN

  The state transitions to IP_CONNECTED.

3. In IP_CONNECTED, when the application receives *GCEV_ANSWERED* from the PSTN the state transitions to GATE_CONNECTED.

6.6.3. Call Teardown

1. When either side (PSTN or IP) sends a *GCEV_DISCONNECTED* event in any state except for IP_OFFERED, the application calls **gc_Unlisten( )** for the IP side and **ag_Unlisten( )** or **dt_Unlisten( )** for the PSTN side. The application also calls **gc_DropCall( )** for both sides of the call to disconnect the call.

   The state transitions to GATE_DROP.

2. When the application receives *GCEV_DROPCALL* from both sides the application calls **gc_Extension( )** to get RTCP information for the call.

   When the application receives *GCEV_EXTENSION* with the RTCP information it calls **gc_ReleaseCall( )** to release the call.

   The state then transitions to GATE_RELEASE.

3. When the application receives a *GCEV_RELEASECALL* event it sends **IPTResetSession()** and the call state transitions to GATE_NULL.

If a *GCEV_DISCONNECTED* event is received from the IP side when the state is IP_OFFERED:

1. The application calls **gc_DropCall( )** for the IP side and the state transitions to IP_DROP.

2. When the application receives *GCEV_DROPCALL* from the IP side, it calls **gc_Extension( )** to get the RTCP information.

   When the application receives *GCEV_EXTENSION* the application calls **gc_ReleaseCall( )**and the state transitions to GATE_NULL.

6.6.4. Glare Conditions

Glare conditions occur when a call is being initiated from both sides at the same time. If such a condition is discovered, the state transitions directly to GATE_DROP and proceeds with call teardown.

Appendix A

Log File of IP Call Establishment

```
DATE: 08/16/01 TIME: 10:49:20
TRACE: File: gatepstn.c Line: 189
        End of pstnOpenFrontEnd function on channel 14


DATE: 08/16/01 TIME: 10:49:21
TRACE: File: gateip.c Line: 99
        Start ipOpenDevices function on channel 14


DATE: 08/16/01 TIME: 10:49:21
TRACE: File: gateip.c Line: 116
        End of ipOpenDevices function on channel 14


DATE: 08/16/01 TIME: 10:49:21
TRACE: File: gatepstn.c Line: 369
        In pstnGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:49:21
TRACE: File: gatestate.c Line: 57
In GATE_INIT State on channel 14
      got Event GCEV_UNBLOCKED (0x833) from PSTN


DATE: 08/16/01 TIME: 10:49:21
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:49:21
TRACE: File: gatestate.c Line: 57
In GATE_INIT State on channel 14
      got Event GCEV_UNBLOCKED (0x833) from IP


DATE: 08/16/01 TIME: 10:49:21
TRACE: File: gateip.c Line: 466
        End of ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:53:00
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:53:00
TRACE: File: gatestate.c Line: 129
In GATE_NULL State on channel (0xe)
      got event GCEV_OFFERED (0x824) from IP


DATE: 08/16/01 TIME: 10:53:00
```

```
TRACE: File: gateip.c Line: 466
        End of ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:53:00
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:53:00
TRACE: File: gateip.c Line: 227
        Start OnExtension function on channel 14


DATE: 08/16/01 TIME: 10:53:00
INFO: File: gateip.c Line: 250
        Got extension data display:


DATE: 08/16/01 TIME: 10:53:00
INFO: File: gateip.c Line: 262
        Got extension data phone list:


DATE: 08/16/01 TIME: 10:53:00
INFO: File: gateip.c Line: 336
        Got extension data H221NONSTANDARD: country_code 181,extension 11,
manufacturer_code 11


DATE: 08/16/01 TIME: 10:53:00
INFO: File: gateip.c Line: 342
        Got extension data  IPPARM_VENDOR_PRODUCT_ID IPLink


DATE: 08/16/01 TIME: 10:53:00
INFO: File: gateip.c Line: 348
        Got extension data  IPPARM_VENDOR_VERSION_ID Dialogic Corp.


DATE: 08/16/01 TIME: 10:53:00
INFO: File: gateip.c Line: 291
        Got extension data  IPPARM_CONFERENCE_GOAL


DATE: 08/16/01 TIME: 10:53:00
INFO: File: gateip.c Line: 298
        Got extension data  IP_CONFERENCEGOAL_ID ┐┴A&ö═◘▯ V4444ï


DATE: 08/16/01 TIME: 10:53:00
TRACE: File: gateip.c Line: 362
        End of OnExtension function on channel 14


DATE: 08/16/01 TIME: 10:53:00
TRACE: File: gatestate.c Line: 357
In IP_OFFERED State on channel (0xe)
      got event GCEV_EXTENSION (0x868) from IP


DATE: 08/16/01 TIME: 10:53:00
```

```
TRACE: File: gateip.c Line: 466
        End of ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:53:01
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:53:01
TRACE: File: gatestate.c Line: 357
In IP_OFFERED State on channel (0xe)
      got event GCEV_ANSWERED (0x802) from IP


DATE: 08/16/01 TIME: 10:53:01
TRACE: File: gatepstn.c Line: 369
        In pstnGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:53:01
TRACE: File: gatestate.c Line: 654
In IP_CONNECTED State on channel 14
      got event GCEV_CONNECTED (0x822) from PSTN


DATE: 08/16/01 TIME: 10:53:01
TRACE: File: gatepstn.c Line: 232
        In pstnListen function on channel 14


DATE: 08/16/01 TIME: 10:53:01
TRACE: File: gatepstn.c Line: 275
        End of pstnListen function on channel 14


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gateip.c Line: 436
Got GCEV_DISCONNECTED. Reason: Remote Termination

DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gatestate.c Line: 797
In GATE_CONNECTED State on channel 14
      got event GCEV_DISCONNECTED (0x826) from IP


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gatepstn.c Line: 296
        In pstnUnListen function on channel 14


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gatepstn.c Line: 348
        End of pstnUnListen function on channel 14


DATE: 08/16/01 TIME: 10:54:24
```

```
TRACE: File: gatestate.c Line: 841
Drop call on channel 14

DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gateip.c Line: 466
        End of ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gatepstn.c Line: 369
        In pstnGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gatestate.c Line: 880
In GATE_DROP State on channel 14
      got event GCEV_DROPCALL (0x805) from PSTN


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gatestate.c Line: 880
In GATE_DROP State on channel 14
      got event GCEV_DROPCALL (0x805) from IP


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gateip.c Line: 466
        End of ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 14


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gateip.c Line: 227
        Start OnExtension function on channel 14


DATE: 08/16/01 TIME: 10:54:24
INFO: File: gateip.c Line: 272
        Got extension data RTCP info:timestamp 644440,tx_packets 1948,tx_octets 490896
send_indication 1


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gateip.c Line: 362
        End of OnExtension function on channel 14


DATE: 08/16/01 TIME: 10:54:24
TRACE: File: gatestate.c Line: 880
In GATE_DROP State on channel 14
      got event GCEV_EXTENSION (0x868) from IP
```

Appendix B

Log File of PSTN Call Establishment

```
DATE: 08/16/01 TIME: 10:57:55
TRACE: File: gatepstn.c Line: 189
        End of pstnOpenFrontEnd function on channel 10


DATE: 08/16/01 TIME: 10:57:56
TRACE: File: gateip.c Line: 99
        Start ipOpenDevices function on channel 10


DATE: 08/16/01 TIME: 10:57:56
TRACE: File: gateip.c Line: 116
        End of ipOpenDevices function on channel 10


DATE: 08/16/01 TIME: 10:57:57
TRACE: File: gatepstn.c Line: 369
        In pstnGetEvent function on channel 10


DATE: 08/16/01 TIME: 10:57:57
TRACE: File: gatestate.c Line: 57
In GATE_INIT State on channel 10
      got Event GCEV_UNBLOCKED (0x833) from PSTN


DATE: 08/16/01 TIME: 10:57:57
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 10


DATE: 08/16/01 TIME: 10:57:57
TRACE: File: gatestate.c Line: 57
In GATE_INIT State on channel 10
      got Event GCEV_UNBLOCKED (0x833) from IP


DATE: 08/16/01 TIME: 10:57:57
TRACE: File: gateip.c Line: 466
        End of ipGetEvent function on channel 10


DATE: 08/16/01 TIME: 10:58:37
TRACE: File: gatepstn.c Line: 369
        In pstnGetEvent function on channel 10


DATE: 08/16/01 TIME: 10:58:37
TRACE: File: gatestate.c Line: 129
In GATE_NULL State on channel (0xa)
      got event GCEV_OFFERED (0x824) from PSTN


DATE: 08/16/01 TIME: 10:58:37
```

```
TRACE: File: gateip.c Line: 140
        Start ipMakeCall function on channel 10


DATE: 08/16/01 TIME: 10:58:37
TRACE: File: gateip.c Line: 205
        End of ipMakeCall function on channel 10


DATE: 08/16/01 TIME: 10:58:37
TRACE: File: gatepstn.c Line: 369
        In pstnGetEvent function on channel 10


DATE: 08/16/01 TIME: 10:58:37
TRACE: File: gatestate.c Line: 511
In PSTN_OFFERED State on channel (0xa)
      got event GCEV_ACCEPT (0x804) from PSTN


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 10


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gatestate.c Line: 511
In PSTN_OFFERED State on channel (0xa)
      got event GCEV_PROCEEDING (0x827) from IP


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gateip.c Line: 466
        End of ipGetEvent function on channel 10


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 10


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gatestate.c Line: 511
In PSTN_OFFERED State on channel (0xa)
      got event GCEV_CONNECTED (0x822) from IP


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gatepstn.c Line: 232
        In pstnListen function on channel 10


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gatepstn.c Line: 275
        End of pstnListen function on channel 10


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gateip.c Line: 466
        End of ipGetEvent function on channel 10
```

```
DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 10


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gateip.c Line: 227
        Start OnExtension function on channel 10


DATE: 08/16/01 TIME: 10:58:38
INFO: File: gateip.c Line: 250
        Got extension data display: target


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gateip.c Line: 362
        End of OnExtension function on channel 10


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gatestate.c Line: 654
In IP_CONNECTED State on channel 10
      got event GCEV_EXTENSION (0x868) from IP


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gateip.c Line: 466
        End of ipGetEvent function on channel 10


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gatepstn.c Line: 369
        In pstnGetEvent function on channel 10


DATE: 08/16/01 TIME: 10:58:38
TRACE: File: gatestate.c Line: 654
In IP_CONNECTED State on channel 10
      got event GCEV_ANSWERED (0x802) from PSTN


DATE: 08/16/01 TIME: 11:00:03
TRACE: File: gatepstn.c Line: 369
        In pstnGetEvent function on channel 10


DATE: 08/16/01 TIME: 11:00:03
TRACE: File: gatepstn.c Line: 416
Got GCEV_DISCONNECTED. Reason: Normal clearing

DATE: 08/16/01 TIME: 11:00:03
TRACE: File: gatestate.c Line: 797
In GATE_CONNECTED State on channel 10
      got event GCEV_DISCONNECTED (0x826) from PSTN


DATE: 08/16/01 TIME: 11:00:03
TRACE: File: gatepstn.c Line: 296
        In pstnUnListen function on channel 10
```

```
DATE: 08/16/01 TIME: 11:00:03
TRACE: File: gatepstn.c Line: 348
        End of pstnUnListen function on channel 10


DATE: 08/16/01 TIME: 11:00:03
TRACE: File: gatestate.c Line: 841
Drop call on channel 10

DATE: 08/16/01 TIME: 11:00:03
TRACE: File: gatepstn.c Line: 369
        In pstnGetEvent function on channel 10


DATE: 08/16/01 TIME: 11:00:03
TRACE: File: gatestate.c Line: 880
In GATE_DROP State on channel 10
      got event GCEV_DROPCALL (0x805) from PSTN


DATE: 08/16/01 TIME: 11:00:05
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 10


DATE: 08/16/01 TIME: 11:00:05
TRACE: File: gatestate.c Line: 880
In GATE_DROP State on channel 10
      got event GCEV_DROPCALL (0x805) from IP


DATE: 08/16/01 TIME: 11:00:05
TRACE: File: gateip.c Line: 466
        End of ipGetEvent function on channel 10


DATE: 08/16/01 TIME: 11:00:05
TRACE: File: gateip.c Line: 385
        In ipGetEvent function on channel 10


DATE: 08/16/01 TIME: 11:00:05
TRACE: File: gateip.c Line: 227
        Start OnExtension function on channel 10


DATE: 08/16/01 TIME: 11:00:05
INFO: File: gateip.c Line: 272
        Got extension data RTCP info:timestamp 649480,tx_packets 7971,tx_octets 733332
send_indication 1


DATE: 08/16/01 TIME: 11:00:05
TRACE: File: gateip.c Line: 362
        End of OnExtension function on channel 10


DATE: 08/16/01 TIME: 11:00:05
TRACE: File: gatestate.c Line: 880
In GATE_DROP State on channel 10
      got event GCEV_EXTENSION (0x868) from IP
```

```
DATE: 08/16/01 TIME: 11:00:05
TRACE: File: gateip.c Line: 466
       End of ipGetEvent function on channel 10
```

Index