Dialogic
Making Innovation Thrive

Dialogic® NaturalAccess™ NaturalConference™ API Developer's Manual

December 2009 64-0512-02

www.dialogic.com

Copyright and legal notices

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation or its subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Corporation at the address indicated below or on the web at www.dialogic.com.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.

Any use case(s) shown and/or described herein represent one or more examples of the various ways, scenarios or environments in which Dialogic® products can be used. Such use case(s) are non-limiting and do not represent recommendations of Dialogic as to whether or how to use Dialogic products.

Dialogic, Dialogic Pro, Brooktrout, Diva, Cantata, SnowShore, Eicon, Eicon Networks, NMS Communications, NMS (stylized), Eiconcard, SIPcontrol, Diva ISDN, TruFax, Exnet, EXS, SwitchKit, N20, Making Innovation Thrive, Connecting to Growth, Video is the New Voice, Fusion, Vision, PacketMedia, NaturalAccess, NaturalCallControl, NaturalConference, NaturalFax and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Corporation or its subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries. The names of actual companies and product mentioned herein are the trademarks of their respective owners.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

Revision history

Revision	Release date	Notes
9000-6749-10	November, 1999	SJC, NaturalConference 1.0 Beta
9000-6749-11	February, 2000	MCM, NaturalConference 1.0
9000-6749-12	July, 2000	MCM, PSF 4.0
9000-6749-13	September, 2000	MCM, CT Access 3.0 and 4.0
9000-6749-14	October, 2000	MCM, NaturalConference 1.1 Beta
9000-6749-15	January, 2001	MCM/LBG, NaturalConference 1.1
9000-6749-16	March, 2001	LBG, NACD 2000-2 , NaturalConference 1.11
9000-6749-17	April, 2001	LBG, NaturalConference 2.0, NACD 2001-1 Beta
9000-6749-18	August, 2001	LBG, NACD 2001-1
9000-6749-19	November, 2001	LBG, NaturalConference 2.1, NACD 2002-1 Beta
9000-6749-20	May, 2002	SRG, NACD 2002-1
9000-6749-21	April, 2003	LBG, Natural Access 2003-1
9000-6749-22	April, 2004	LBG/MCM, Natural Access 2004-1
9000-6749-23	November, 2004	MCM, Natural Access 2005-1
9000-6749-24	July, 2006	SRG, Natural Access 2005-1 SP2
64-0512-01	October, 2009	LBG, NaturalAccess R9.0
64-0512-02	December, 2009	LBG, NaturalAccess R9.0.1
Last modified: December 10, 2009		

Refer to www.dialogic.com for product updates and for information about support policies, warranty information, and service offerings.

Table Of Contents

1 Introduction

The *Dialogic® NaturalAccess™ NaturalConference™ API Developer's Manual* describes how to use the NaturalConference API to write applications that create and manage conferences. The manual provides detailed descriptions of the NaturalConference capabilities and functions. Use this manual with the *Dialogic® NaturalAccess™ Software Developer's Manual*.

This manual is targeted to developers of telephony and voice applications who are using NaturalAccess. This document defines telephony terms where applicable, but assumes that you are familiar with telephony concepts and switching. It also assumes that you are familiar with the C programming language.

Revision history Notices

2 Terminology

Note: The product to which this document pertains is part of the NMS Communications Platforms business that was sold by NMS Communications Corporation ("NMS") to Dialogic Corporation ("Dialogic") on December 8, 2008. Accordingly, certain terminology relating to the product has been changed. Below is a table indicating both terminology that was formerly associated with the product, as well as the new terminology by which the product is now known. This document is being published during a transition period; therefore, it may be that some of the former terminology will appear within the document, in which case the former terminology should be equated to the new terminology, and vice versa.

Former terminology	Dialogic terminology
CG 6060 Board	Dialogic® CG 6060 PCI Media Board
CG 6060C Board	Dialogic® CG 6060C CompactPCI Media Board
CG 6565 Board	Dialogic® CG 6565 PCI Media Board
CG 6565C Board	Dialogic® CG 6565C CompactPCI Media Board
CG 6565e Board	Dialogic® CG 6565E PCI Express Media Board
CX 2000 Board	Dialogic® CX 2000 PCI Station Interface Board
CX 2000C Board	Dialogic® CX 2000C CompactPCI Station Interface Board
AG 2000 Board	Dialogic® AG 2000 PCI Media Board
AG 2000C Board	Dialogic® AG 2000C CompactPCI Media Board
AG 2000-BRI Board	Dialogic® AG 2000-BRI Media Board
NMS OAM Service	Dialogic® NaturalAccess™ OAM API
NMS OAM System	Dialogic® NaturalAccess™ OAM System
NMS SNMP	Dialogic® NaturalAccess™ SNMP API
Natural Access	Dialogic® NaturalAccess™ Software
Natural Access Service	Dialogic® NaturalAccess™ Service
Fusion	Dialogic® NaturalAccess™ Fusion™ VoIP API
ADI Service	Dialogic® NaturalAccess™ Alliance Device Interface API
CDI Service	Dialogic® NaturalAccess™ CX Device Interface API
Digital Trunk Monitor Service	Dialogic® NaturalAccess™ Digital Trunk Monitoring API
MSPP Service	Dialogic® NaturalAccess™ Media Stream Protocol Processing API
Natural Call Control Service	Dialogic® NaturalAccess™ NaturalCallControl™ API
NMS GR303 and V5 Libraries	Dialogic® NaturalAccess™ GR303 and V5 Libraries

Former terminology	Dialogic terminology
Point-to-Point Switching Service	Dialogic® NaturalAccess™ Point-to-Point Switching API
Switching Service	Dialogic® NaturalAccess™ Switching Interface API
Voice Message Service	Dialogic® NaturalAccess™ Voice Control Element API
NMS CAS for Natural Call Control	Dialogic® NaturalAccess™ CAS API
NMS ISDN	Dialogic® NaturalAccess™ ISDN API
NMS ISDN for Natural Call Control	Dialogic® NaturalAccess™ ISDN API
NMS ISDN Messaging API	Dialogic® NaturalAccess™ ISDN Messaging API
NMS ISDN Supplementary Services	Dialogic® NaturalAccess™ ISDN API Supplementary Services
NMS ISDN Management API	Dialogic® NaturalAccess™ ISDN Management API
NaturalConference Service	Dialogic® NaturalAccess™ NaturalConference™ API
NaturalFax	Dialogic® NaturalAccess™ NaturalFax™ API
SAI Service	Dialogic® NaturalAccess™ Universal Speech Access API
NMS SIP for Natural Call Control	Dialogic® NaturalAccess™ SIP API
NMS RJ-45 interface	Dialogic® MD1 RJ-45 interface
NMS RJ-21 interface	Dialogic® MD1 RJ-21 interface
NMS Mini RJ-21 interface	Dialogic® MD1 Mini RJ-21 interface
NMS Mini RJ-21 to NMS RJ-21 cable	Dialogic® MD1 Mini RJ-21 to MD1 RJ-21 cable
NMS RJ-45 to two 75 ohm BNC splitter cable	Dialogic® MD1 RJ-45 to two 75 ohm BNC splitter cable
NMS signal entry panel	Dialogic® Signal Entry Panel

3 Overview of the NaturalConference service

NaturalConference service definition

The NaturalConference service is a C function library component of NaturalAccess that provides conferencing capabilities on AG and CG boards.

NaturalConference provides the following types of functions:

- Functions for opening conference resources, creating conferences, and adding members to conferences.
- Functions for retrieving information about:
  - Conference resources
  - Conferences
  - Members
- Functions for dynamically changing conference features and member capabilities.

NaturalConference objects

A conference is created and managed through three interdependent entities:

Object	Description
Conference resource	A single digital signal processor (DSP) or a set of DSPs working together. When a conference resource is created, a set of DSPs is declared, providing uniform capabilities and features. A conference resource can be likened to a building, with the number of DSPs assigned to the resource representing the scale of the building.
Conference	A space created by the application on a specified conference resource. In the building analogy, the conference is a meeting room in the building. Initially, the room is empty. The number of participants expected is determined by the application creating the conference. An application can also add seats to the meeting after the meeting begins.
Member	An object created by the application on a specified conference. Adding a member to a conference is like a participant sitting in a seat in the meeting room/building analogy.

Conference capabilities

NaturalConference capabilities can be enabled for a given conference or member. NaturalConference capabilities include:

- Automatic gain control (AGC) on input and output.

  Automatic gain control is an algorithm applied to incoming speech before compression and storage so that the amplitude of the stored speech is kept at a target level.

- Dual-tone-multi-frequency (DTMF) clamping.

  DTMF clamping enables the removal of any DTMF that is detected on an input signal.

- Tone clamping.

  Tone clamping enables the removal of single frequency tones used for signaling an analog interface.

- Echo cancellation.

  Echo cancellation is an algorithm for removing the portion of the received signal that was determined to be echo by comparison with the output signal.

- Talker privilege.

  Privileged talkers are always active talkers even when they are not speaking.

- Active talker detection.

  Active talker detection provides the ability to determine which participants are talking at a given time.

- Coaching.

  Coaching provides the ability to selectively control which conference members can hear chosen participants. This feature is typically used to enable a coach to listen to a conference and provide input to one or more colleagues without the knowledge of other conference members.

The board being used determines the capabilities available to an application. To regulate system resources, the application can choose the capabilities to use. NaturalConference enables an application to query a conference object about its capabilities.

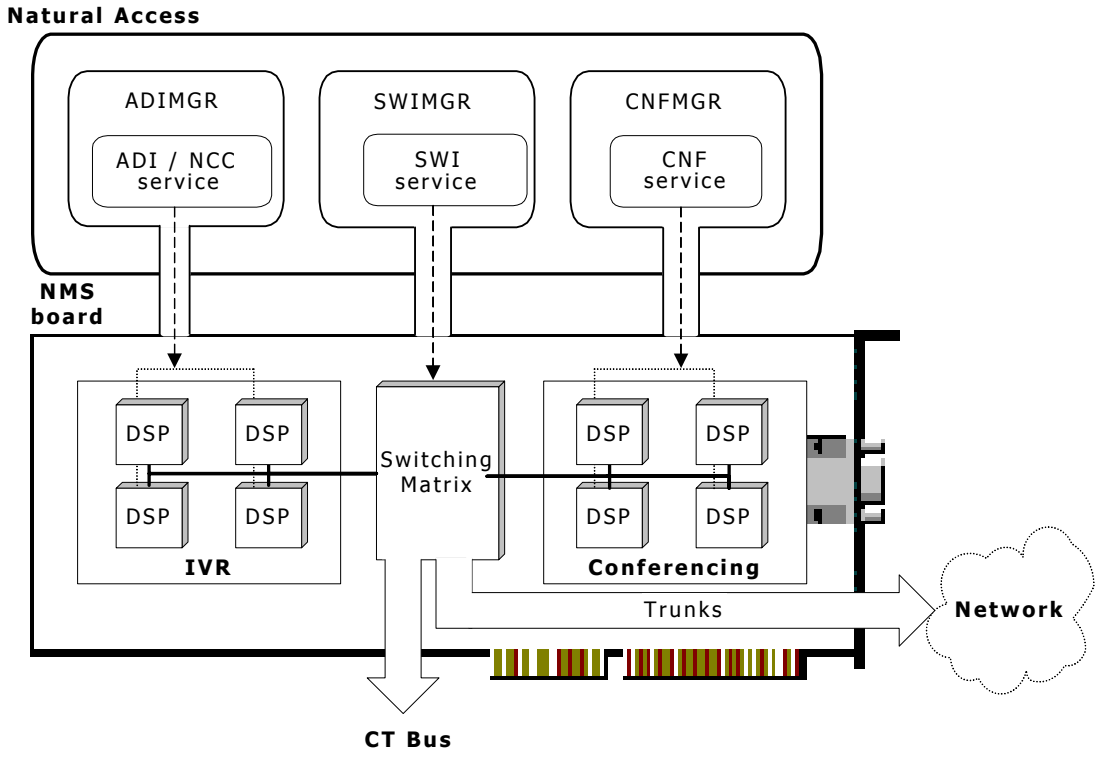
NaturalConference development environment

This topic presents the following information:

- NaturalConference system architecture
- NaturalConference data flow
- Resource handles and conference and member identifiers
- Regulatory constraints

NaturalConference system architecture

On NMS Communications AG and CG boards, NaturalConference functions communicate with enhanced voice processing modules running on digital signal processors (DSPs):



DSPs or DSP pools located on AG and CG boards can be programmed to support IVR, fax, IP telephony, or NaturalConference. NaturalConference requires dedicated DSPs. DSPs programmed for NaturalConference cannot be used for any other feature.

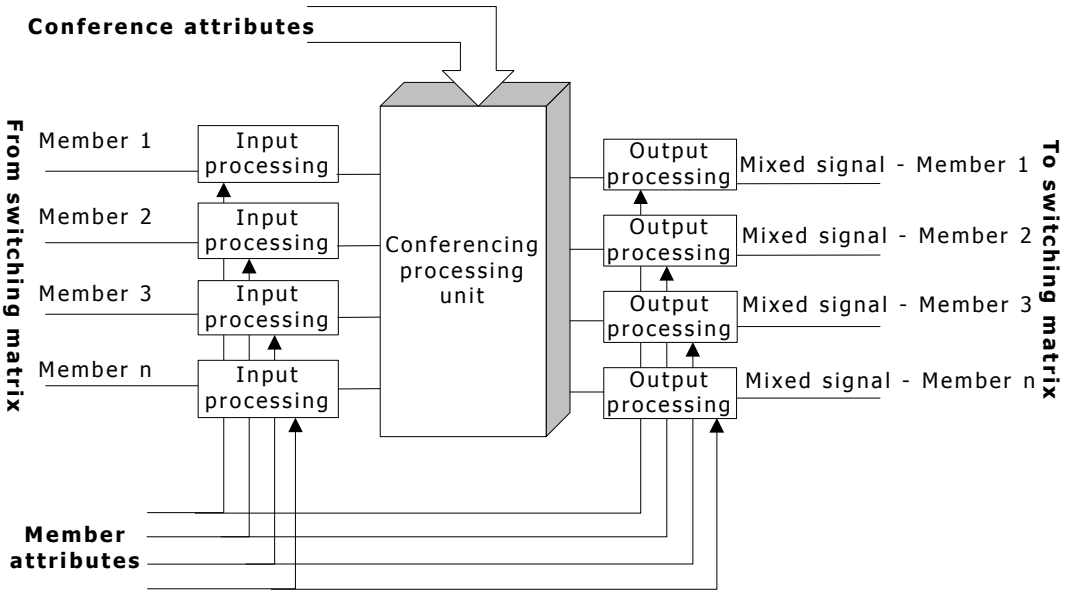
For information about specific board configurations, refer to the board-specific installation and developer's manual.

Although the voice channel associated with a particular member joining a conference does not have to be on the same board as the conference, the application must perform the appropriate switching to connect the new member's line to the conference. If the member comes from the same board, make a local-stream-to-local-stream connection; otherwise, make a bus-stream-to-local-stream connection.

For more information about switching, refer to the *Dialogic® NaturalAccess™ Switching Interface API Developer's Manual* and the *Dialogic® NaturalAccess™ Point-to-Point Switching API Developer's Manual*.

NaturalConference data flow

NaturalConference performs processing for each member's voice input individually, mixes according to the conference attributes, and generates the output signal for each member. The following illustration summarizes the processing chain performed by the conferencing code:



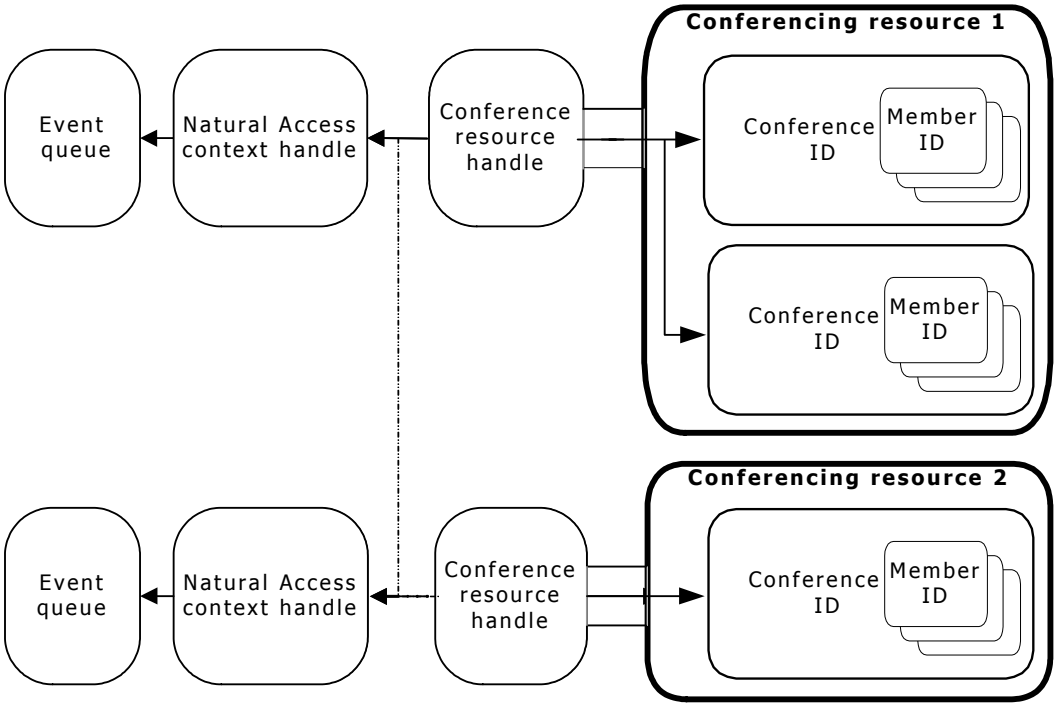
You specify conference capabilities such as DTMF clamping and echo cancellation when you declare the conference resource or create the conference. Other settings, such as the number of loudest speakers, can be changed dynamically. You specify settings (or attributes) such as talking and listening activity and gain control on a member by member basis.

Resource handles and conference and member identifiers

Any call to NaturalConference requires a conference resource handle (***cnfresourcehd***). A conference resource handle links a context and a conference resource. An application can create as many of these paths as necessary among contexts. To obtain a conference resource handle, open a conference resource with **cnfOpenResource**. Once you have a ***cnfresourcehd***, you can create conferences and add members to them.

When you create a new conference, a unique conference identifier is generated and returned to the application. Similarly, when you add a new member to a conference, a unique member identifier is generated and returned to the application. NaturalConference also provides enumeration functions to retrieve existing conference identifiers on a given resource and existing members on a given conference.

By opening a resource and creating a resource handle on a given context, the application implicitly designates which queue returns the events for the conferences created by that resource handle. The following illustration shows the relationships between Natural Access objects, resource handles, and conference identifiers:



The actual link between Natural Access and a NaturalConference object is performed at the resource handle level. Create resource handles using ***ctahd*** in the same way that instances of ***ctahd*** can be created on a queue.

Refer to the Function reference section for a complete description of NaturalConference functions.

Regulatory constraints

Telecom regulatory authorities regulate the use of NaturalConference in some countries. These regulations address two aspects of conferencing:

- The number of outside lines involved in a conference
- The transmission plan

Since these aspects vary with the applications developed using NaturalConference and the countries where NaturalConference is installed, NMS Communications does not obtain type approval for NaturalConference. Contact the regulatory authorities in the country where you plan to install NaturalConference to learn the regulations for applications developed with this product.

Setting up the Natural Access environment

You must set up the Natural Access environment before using NaturalConference. Perform the following steps to set up the Natural Access environment:

- Initialize Natural Access
- Create event queues and contexts

Initializing Natural Access for NaturalConference

To use NaturalConference, specify the service name CNF in one of the following ways:

- **ctaInitialize** invocation arguments
- [ctasys] section of the Natural Access configuration file (*cta.cfg*)

If specified during **ctaInitialize**, the function's parameters could look like the following:

```
CTA_SERVICE_NAME service_names[] = {{"cnf", "cnfmgr"},
                                    {"swi", "swimgr"}
};
ret = ctaInitialize(service_names,sizeof(service_names)
                        /sizeof(CTA_SERVICE_NAME),NULL);
```

The content of the [ctasys] section is used only when the **ctaInitialize** parameter specifies the list of manager/service pairs as NULL. The following lines illustrate how that part of the [ctasys] section should look:

```
[ctasys]
Service = cnf, cnfmgr
Service = swi, swimgr
```

For more information, see *Modifying the Natural Access configuration file* on page 22.

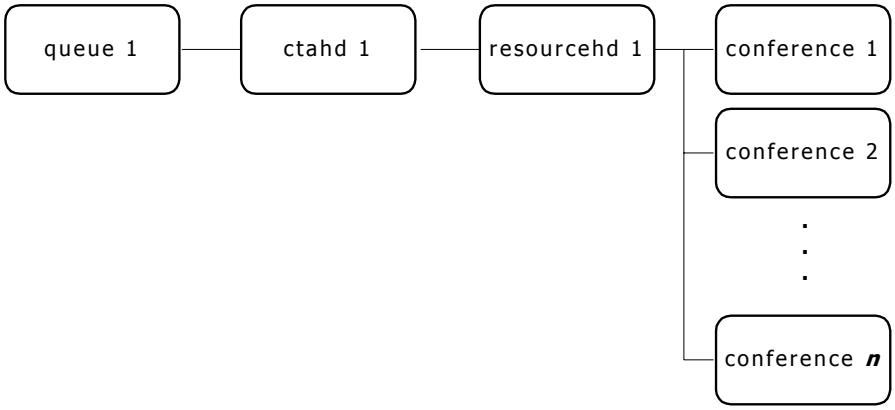
Creating event queues and Natural Access contexts

After initializing Natural Access, the application must create a queue, specify a manager to be used with that queue, and create contexts. Use the following functions:

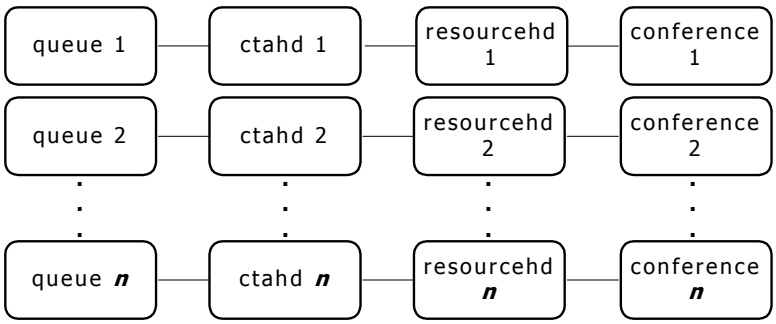
To...	Use this function...	Description
Create an event queue	**ctaCreateQueue**	Accepts manager lists or uses the default manager list from the [ctasys] section.
Create contexts (***ctahd***)	**ctaCreateContext**	Specifies the previously created queue as the queue for receiving the event related to the context.

You can create either one queue per context or one queue for all contexts. A conference is attached to a queue by means of a resource handle (***cnfresourcehd***) opened by ***ctahd***. The application must choose whether to receive events from conferences on the same queue or on different queues. To receive events from conferences on different queues, the application must create one ***ctahd*** and one ***cnfresourcehd*** for each conference. The following illustrations show different programming models of queue/conference attachment.

The following illustration shows the basic model where one ***ctahd*** is created for all conferences in the system. This model does not allow the management of conferences on separate threads.



This illustration shows the most flexible model, where a unique ***ctahd*** is created on a separate queue for each resource handle on which a conference is created. This model allows for the management of conferences on separate threads.



A context obtains access rights for NaturalConference (not for creating conferences) when calling **ctaOpenServices** on a specified ***ctahd***. The code that performs this should look like the following:

```
CTA_SERVICE_DESC service_descs[] =
{
{
{"cnf", "cnfmgr"}, /* name of the NaturalConference service     */
{0},               /* svcaddr ->  reserved                     */
{0},               /* svcargs ->  not used by NaturalConference */
{0}                /* mvipaddr -> not used by NaturalConference */
}
};
ret = ctaOpenServices( ctahd, service_descs,
                       sizeof(service_descs)/sizeof(CTA_SERVICE_DESC) );
```

To open NaturalConference on a context, initialize a service_descs structure with cnf and cnfmgr in the name field.

Linking with NaturalConference

When building a new Natural Access application that uses NaturalConference, link to *cnfapi.lib* (under UNIX, *libcnfapi.so*). *cnfmgr.dll* (under UNIX, *libcnfmgr.so*) is dynamically loaded at runtime.

Refer to the *Dialogic® NaturalAccess™ Service Writer's Manual* for more information about service implementation.

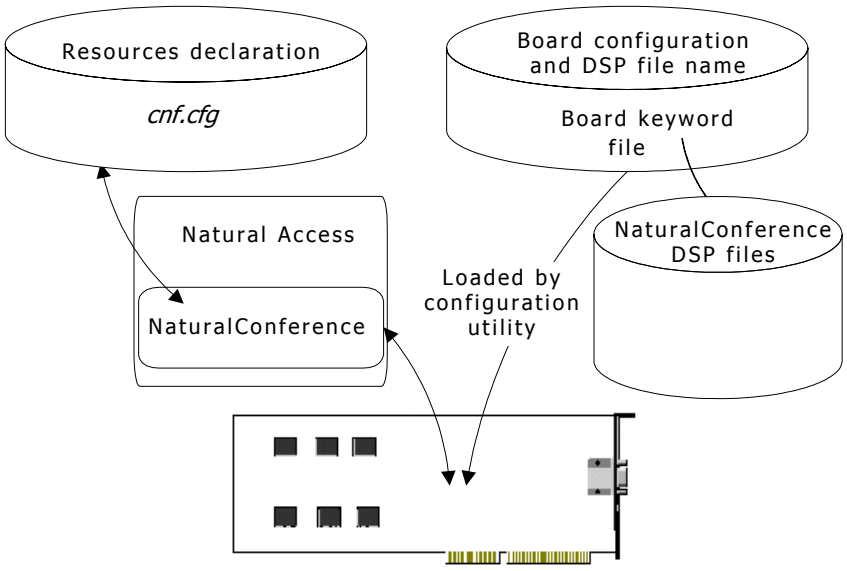
4 Configuring NaturalConference

Overview of configuring NaturalConference

Before you develop applications using NaturalConference, configure NaturalConference for your system. NaturalConference requires appropriate declarations in three configuration files:

Configuration file	Description
Natural Access configuration file (*cta.cfg*)	Defines all Natural Access services used by the application.
Board keyword file	Contains sections for declaring DSP functionality and associated DSP files to be loaded on AG and CG boards. The NaturalConference DSP file must be declared in this file. NaturalConference provides sample board keyword files.
NaturalConference configuration file (*cnf.cfg*)	Contains the conference resource declaration. NaturalConference uses this information to create and manage conference resources. If the NaturalConference configuration file does not exist, or if it does not contain a valid declaration consistent with the board configuration file content, NaturalConference cannot open any conference resources. NaturalConference provides sample NaturalConference configuration files.

The following illustration shows how NaturalConference uses the configuration files:



Modifying the Natural Access configuration file

If you use Natural Access Server (*ctdaemon*) as part of your application environment, define all Natural Access services that the application uses by specifying the service and service manager names in the Natural Access configuration file (*cta.cfg*). You must modify *cta.cfg* to use tracing for any of the Natural Access services, including the NaturalConference (CNF) service. If you do not use the Natural Access Server (*ctdaemon*), make sure that the application's Natural Access services and service managers are explicitly defined in your application code.

For more information about *ctdaemon* and the Natural Access development environment, refer to the *Dialogic® NaturalAccess™ Software Developer's Manual*.

A NaturalConference application typically uses the following services:

This service...	With this service manager...	Provides the...
CNF	CNFMGR	NaturalConference API
NCC	ADIMGR for AG and CG boards	Natural Access call control API
SWI	SWIMGR	Switching API
OAM	OAMMGR	Functions for configuring, monitoring, and testing boards. Natural Access Server (*ctdaemon*) must be running.

The default *cta.cfg* file includes the NCC service (refer to the sample *cta.cfg* file).

cta.cfg is located in the following directories:

Operating system	Directory
Windows	*\nms\ctaccess\cfg*
UNIX	*/opt/nms/ctaccess/cfg*

If the CNF service is not listed in *cta.cfg*, perform the following steps to edit the file:

Step	Action
1	Stop *ctdaemon* and all programs running Natural Access.
2	Add the following line for the CNF service: `Service = cnf, cnfmgr`
3	Save and close *cta.cfg*.
4	Restart *ctdaemon*.

Sample Natural Access configuration file

For details about the Natural Access configuration file, refer to the *Dialogic® NaturalAccess™ Software Developer's Manual*. The following code excerpt from a Natural Access configuration file includes the CNF service:

```
#==========================================================================
# cta.cfg
#
# This is an example of a file that specifies Natural Access configuration.
# It allows you to:
#   - specify generic operational settings that apply to Natural Access
#     Server, ctdaemon,and all Natural Access applications.
#     Note: these settings can be overwritten by Natural Access applications
#     via ctaInitialize.
#   - specify application specific settings
#   - specify Natural Access Server and ctdaemon specific settings
#   - redefine service specific parameter defaults
#
#==========================================================================



#==========================================================================
# Natural Access System Configuration (ctasys)
#
# Valid options are:
#   Service = name, dll        - tells the daemon about available "services"
#                              - tells the Natural Access server what
#                                "services" to export
#
# Note: NCC should always precede ADI when both services are listed.
#==========================================================================
[ctasys]
Service = ncc, adimgr
Service = adi, adimgr
Service = dtm, adimgr
Service = ppx, ppxmgr
Service = swi, swimgr
Service = vce, vcemgr
Service = oam, oammgr
Service = cnf, cnfmgr


#==========================================================================
```

DSPs and conferencing resources

This topic presents conference resource information for:

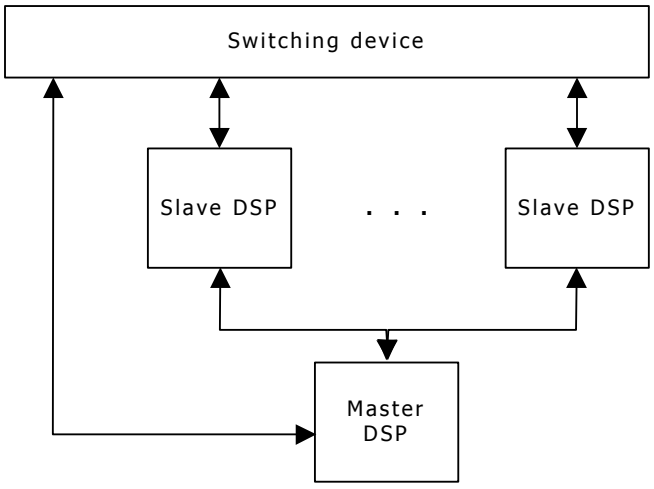
- AG boards
- CG boards

NaturalConference enables several DSPs to be merged into a single conference resource. This concept is called spanning. Spanning provides a single space to create conferences with more participants. Spanning can also be used to optimize the management of the resource.

To provide this feature, dedicate DSPs to two different tasks:

Task	Description
Signal processing (echo cancellation, tone clamping, gain control)	The DSP dedicated to this task is called a slave. These DSPs perform MIPS-consuming tasks on individual conference channels, but do not manage the conference (mixing) process.
Mixer or master	A DSP configured as the mixer or master primarily manages the conference processing (the mixing) for any members in a conference. As the master, the DSP subcontracts MIPS-consuming tasks to slave DSPs. It keeps enough power to perform the conference processing of a larger number of conference participants.

A conference resource implementing spanning is composed of a single master and one or more slaves. The following illustration provides a logical view of the NaturalConference master/slave architecture:



As shown in this illustration, the master DSP manages the signal processing for a small number of conferees. The number of members with locally managed signal processing on the master depends on the number of slave DSPs that are connected to the master. Refer to *Managing resources* on page 45 for more information about precise conference resource capabilities.

To set up a configuration that implements spanning, indicate precisely which DSPs belong to which resources. Master and slave firmware are different and have different names. Both the firmware name and the resource for each DSP must be specified in the appropriate configuration files. Refer to *Modifying the board keyword file* on page 27 for more information about board and conference configuration files.

You do not have to use spanning. You have the option to make a single DSP a separate conference resource. To do so, load the appropriate firmware and modify *cnf.cfg* accordingly.

The spanned conference resource architecture relies on the board's architecture to establish connections between master and slaves. Different types of boards provide different means and capacities for interconnecting DSP and voice channels. Read the configuration related sections of this manual to understand and modify the appropriate configuration file to create the desired conference configurations.

Conferencing resources on AG boards

On AG boards, you can create many conferencing resources that implement spanning. There are two requirements to properly declaring conferencing resources:

- The DSP being configured must be located on the same DSP bank as the conferencing resource.
- The maximum number of members managed by a master DSP cannot be greater than 64, regardless of the number of DSPs allocated to a conferencing resource.

The following table lists the existing DSP banks and the corresponding DSP numbers. This information is also valid for AG 2000C and AG 2000-BRI boards.

Board model	1st bank DSP number	2nd bank DSP number	3rd bank DSP number
AG 2000/200	0..1	Not applicable.	Not applicable.
AG 2000/400	0..3	Not applicable.	Not applicable.

The first DSP of each resource must be loaded with the master firmware. The other DSP of the conferencing resource must be loaded with the slave firmware.

For example, a single conferencing resource with five DSPs is configured as conferencing resource #0 spanned on five DSPs:

- DSP #1: Master
- DSP #2: Slave
- DSP #3: Slave
- DSP #4: Slave
- DSP #5: Slave

Two conferencing resources with four and three DSPs respectively are configured as:

- Conferencing resource #0 spanned on four DSPs:
  - DSP #1: Master
  - DSP #2: Slave
  - DSP #3: Slave
  - DSP #4: Slave
- Conferencing resource #1 spanned on three DSPs:
  - DSP #5: Master
  - DSP #6: Slave
  - DSP #7: Slave

Conferencing resources on CG boards

On CG boards, DSPs can be configured as a conferencing resource implementing spanning. According to the CG board model, the number of groups of DSPs, as well as the DSP numbers that can be used with spanned conferencing, are different. The existence of spanning does not prevent a DSP on the board from being used as a single DSP conferencing resource when the appropriate (standalone) firmware is loaded.

Note: For specific information about firmware names, capacities, and possible configurations implementing spanning according to CG board type, refer to the NaturalConference *readme* file.

The following examples refer to CG boards.

- Master DSP# = 2
- Slave DSP# = 3 to 9

For example, a single conferencing resource with eight DSPs is configured as conferencing resource #0 spanned on eight DSPs:

- DSP #2: Master
- DSP #3: Slave
- DSP #4: Slave
- DSP #5: Slave
- DSP #6: Slave
- DSP #7: Slave
- DSP #8: Slave
- DSP #9: Slave

A configuration with a conferencing resource including four DSPs is configured as:

- DSP #2: Master
- DSP #3: Slave
- DSP #4: Slave
- DSP #5: Slave

By default, the number of timeslots (or voice channels) that can be managed by a single DSP on a CG board is 16. However, in certain circumstances this number can be set to 32 (depending on the conference capabilities being used).

When echo cancellation is not used, each DSP that is a part of a conferencing resource can manage up to 32 members and corresponding timeslots. Specify the parameters in the NMS OAM configuration file by setting the three keywords as shown for a four DSP conferencing resource:

```
DSP.C5x[2..5].NumTxTimeSlots = 32
DSP.C5x[2..5].NumRxTimeSlots = 32
DSP.C5x[2..5].CmdQSize = 0x110
ConferencingStream.Enable = YES
```

When a DSP is programmed to manage 32 timeslots, the value for the CmdQSize keyword must be set to 0x110. By adding ConferencingStream.Enable = YES to the configuration file, logical stream switching of members into conferences is enabled.

For example, the NMS OAM configuration file should contain the following statements to set up one or more single DSP conferencing resources with a maximum of 16 members per conferencing resource:

```
DSP.c5x[any dsp number].Files = <standalone_firmware_name>
DSP.c5x[any dsp number].Files = <standalone_firmware_name>
.
.
.
ConferencingStream.Enable = YES
```

To set-up a four DSP conferencing resource implementing spanning, without echo cancellation that can manage up to 96 members, the NMS OAM configuration file should contain the following statements:

```
DSP.C5x[2].Files = <master_firmware_name>
DSP.C5x[3..5].Files = <slave_firmware_name>
DSP.C5x[2..5].Libs = cg6klibu cg6mslib
DSP.C5x[2..5].NumTxTimeSlots = 32
DSP.C5x[2..5].NumRxTimeSlots = 32
DSP.C5x[2..5].CmdQSize = 0x110
ConferencingStream.Enable = YES
```

The *cg6mslib* file is used when implementing spanning independent of the number of timeslots used by each DSP. A configuration with one master and seven slaves requires the following keywords in the NMS OAM configuration file:

```
DSP.C5x[2].Files = <master_firmware_name>
DSP.C5x[3..9].Files = <slave_firmware_name>
DSP.C5x[2..9].Libs = cg6klibu cg6mslib
```

Modifying the board keyword file

On AG and CG boards, NaturalConference uses dedicated DSPs that cannot be used with other features such as IVR, fax, or NMS Fusion. Dedicating a DSP to NaturalConference can reduce the number of ports available for use with IVR. The number of IVR and NaturalConference ports simultaneously available depends on the board model installed and the configuration of the computer.

To declare DSPs as conferencing resources, modify the board keyword file to associate DSPs with a NaturalConference DSP file. Use one of the sample board keyword files provided with NaturalConference as a model. Choose a board keyword file that corresponds to the board installed on your system.

Follow this procedure to declare DSPs for use by NaturalConference:

Step	Action
1	Open the board keyword file.
2	For each board you want NaturalConference to use, add the following keyword and parameter: • AG boards: DSP.C5x[***n***].Image = ag*.c54 • CG boards: DSP.C5x[***n***].Files = cg* where ***n*** is the DSP number and ***** represents the NaturalConference DSP files. You can also specify a range of DSPs. For example: `DSP.C5x[1..3].Image = ag2conf.c54` Refer to *DSPs and conferencing resources* on page 24 for information about modifying the NMS OAM configuration file when implementing conferencing resource spanning.
3	Save the modifications.
4	Create a NaturalConference configuration file, as described in *Setting up the NaturalConference configuration file* on page 37.
5	Run the board configuration utility to load or reload the board. If the configuration utility succeeds without error messages, NaturalConference works according to the resources declared in the NaturalConference configuration file.

Refer to the board-specific installation and developer's manual for a complete description of the keywords.

The board keyword files are located in the following directories:

Board type	Operating system	Directory
AG boards	Windows	*\nms\ag\cfg*
	UNIX	*/opt/nms/ag/cfg*
CG boards	Windows	*\nms\cg\cfg*
	UNIX	*/opt/nms/cg/cfg*

The following table lists sample keyword files for some of the AG and CG boards that support NaturalConference:

Board	Sample keyword file	Description
AG 2000/200 AG 2000C/200 AG 2000-BRI/200	*agpi2cnf_200.cfg*	Eight analog ports in mu-law with one conferencing resource containing a single DSP.
AG 2000/200 AG 2000C/200 AG 2000-BRI/200	*agpi2cnfa_200.cfg*	Eight analog ports in A-law with one conferencing resource containing a single DSP.
AG 2000/400 AG 2000C/400 AG 2000-BRI/400	*agpi2cnf_400.cfg*	Eight analog ports in mu-law with one conferencing resource containing three DSPs.
AG 2000/400 AG 2000C/400 AG 2000-BRI/400	*agpi2cnfa_400.cfg*	Eight analog ports in A-law with one conferencing resource containing three DSPs.
CG 6060 CG 6060C	*c6060cnf_ivr_nonspan.cfg*	IVR configuration with one conferencing resource.
CG 6060 CG 6060C	*c6060cnf_ivr_span.cfg*	IVR configuration with one conferencing resource spanning two DSPs.
CG 6565 CG 6565C CG 6565E	*c6565cnf_ivr_nonspan.cfg*	IVR configuration with one conferencing resource.
CG 6565 CG 6565C CG 6565E	*c6565cnf_ivr_span.cfg*	IVR configuration with one conferencing resource spanning two DSPs.

Sample board keyword files

This topic presents sample board keyword files for running NaturalConference on the following boards:

- AG 2000/400
- CG 6060/C
- CG 6565/C

AG 2000/400 sample board keyword file

The following sample board keyword file sets up an AG 2000/400 board running eight loopstart protocols for an analog interface in mu-law, capable of handling 40 to 80 conferencing ports in a single conferencing resource with three DSPs.

```
##                       agpi2cnf_400.cfg
#
#  This model of AG configuration file has to be used with
#  an AG2000/400 with 8 LoopStart interfaces configured in mu-Law.
#  One DSP is used for IVR and three DSPs are loaded with NaturalConference
#  to form one conferencing resource containing three DSPs.
#
##

Clocking.HBus.ClockSource = OSC
Clocking.HBus.ClockMode = STANDALONE

# Protocols definition
TCPFiles[0] = nocc.tcp    # "no trunk control" protocol
TCPFiles[1] = lps0.tcp    # Loopstart protocol

# Slac Files Definition
#NetworkInterface.Analog[0..7].ConfigFile = a2usasl6.slc
#NetworkInterface.Analog[0..7].ConfigFile = a2eurlsc.slc
#NetworkInterface.Analog[0..7].ConfigFile = a2jpnsl6.slc

# Default Modules activated
DSP.C5x.DSPFiles[0] = callp
DSP.C5x.DSPFiles[1] = dtmf
DSP.C5x.DSPFiles[2] = mf
DSP.C5x.DSPFiles[3] = ptf
DSP.C5x.DSPFiles[4] = signal
DSP.C5x.DSPFiles[5] = tone
DSP.C5x.DSPFiles[6] = voice
DSP.C5x.DSPFiles[7] = rvoice
#DSP.C5x.DSPFiles[8] = wave
#DSP.C5x.DSPFiles[9] = oki

DLMFiles[0] = gtp.leo
DLMFiles[1] = voice.leo
DLMFiles[2] = svc.leo

XLaw = MU-LAW


#############################################################################
#                                                                           #
# one group of three DSPs is loaded with NaturalConference to               #
# form one conferencing resource containing three DSPs.                     #
#                                                                           #
#############################################################################
DSP.C5x[1].Image                  = agcnfm.c54
DSP.C5x[2..3].Image               = agcnfs.c54

ConferencingStream.Enable         = YES
```

CG 6060/C sample board keyword files

The following sample board keyword file sets up an IVR configuration with one conferencing resource on a CG 6060/C board:

```
#
#     c6060cnf_ivr_nonspan.cfg
#     CG 6060 configuration file
#
#     This file configures the board to run NaturalConference in standalone
#     mode (non-spanning).
#

Clocking.HBus.ClockMode                          = STANDALONE
Clocking.HBus.ClockSource                        = OSC

DSPStream.VoiceIdleCode[0..15]                    = 0x7F
DSPStream.SignalIdleCode[0..15]                   = 0x00

#---------------------------
#  NOTE: T1 configuration
#---------------------------
NetworkInterface.T1E1[0..15].Type                 = T1
NetworkInterface.T1E1[0..15].Impedance            = DSX1
NetworkInterface.T1E1[0..15].LineCode             = B8ZS
NetworkInterface.T1E1[0..15].FrameType            = ESF
NetworkInterface.T1E1[0..15].SignalingType        = RAW
DSP.C5x[0..47].Libs                               = cg6klibu
DSP.C5x[0..47].XLaw                               = MU_LAW

#---------------------------
#  NOTE: E1 configuration
#---------------------------
#NetworkInterface.T1E1[0..15].Type                 = E1
#NetworkInterface.T1E1[0..15].Impedance            = G703_120_OHM
#NetworkInterface.T1E1[0..15].LineCode             = HDB3
#NetworkInterface.T1E1[0..15].FrameType            = CEPT
#NetworkInterface.T1E1[0..15].SignalingType        = RAW
#DSP.C5x[0..47].Libs                               = cg6kliba
#DSP.C5x[0..47].XLaw                               = A_LAW

#---------------------------
# Hardware Echo Cancellation
# NOTE: it is in by pass by default
# NOTE: uncomment the following two keyword lines to enable and set the XLaw
# accordingly
#---------------------------
# HardwareEcho.EchoChipEnabled = YES
# HardwareEcho.XLaw = A_LAW

# NumTxTimeSlots, NumRxTimeSlots may be 16 or 32
DSP.C5x[0].NumTxTimeSlots  =   16
DSP.C5x[0].NumRxTimeSlots  =   16

DSP.C5x[0].CmdQStart    = 0xE800
DSP.C5x[0].DataInQStart = 0xF800
DSP.C5x[0].DspOutQStart = 0xFB00

# The following is required when #timeslots = 32 (but not for 16)
#DSP.C5x[0].CmdQSize     = 0x110

DSP.C5x[0].Files           =   cg6conf.f41
SwitchConnectMode          =   AllConstantDelay
ConferencingStream.Enable  =   YES

#---------------------------
# Resource management
#---------------------------
Resource[0].Name                                 = RSC1
Resource[0].Size                                 = 120
```

```
Resource[0].TCPs                                = nocc
Resource[0].StartTimeSlot                       = 0

################################################################
# Before modifying this resource definition string refer to the CG 6060
# Installation and Developers Manual.
# NOTE: echo.ln20_apt25 - echo running on DSP has been removed
#       from resource definitions.  We recommend user to use
#       the hardware echo chip for echo cancellation instead.
################################################################
Resource[0].Definitions         = ( dtmf.det_all & ptf.det_2f & tone.gen & \
  callp.gnc & ptf.det_4f & \
  ( (rvoice.rec_mulaw & rvoice.play_mulaw) | \
  (rvoice.rec_alaw & rvoice.play_alaw) | \
  (rvoice.rec_lin & rvoice.play_lin) | \
  (voice.rec_16 & (voice.play_16_100 | voice.play_16_150 | voice.play_16_200)) | \
  (voice.rec_24 & (voice.play_24_100 | voice.play_24_150 | voice.play_24_200)) | \
  (voice.rec_32 & (voice.play_32_100 | voice.play_32_150 | voice.play_32_200)) | \
  (voice.rec_64 & (voice.play_64_100 | voice.play_64_150 | voice.play_64_200)) | \
  (wave.rec_11_16b & wave.play_11_16b) | \
  (wave.rec_11_8b & wave.play_11_8b) | \
  (oki.rec_24 & (oki.play_24_100 | oki.play_24_150 | oki.play_24_200)) | \
  (oki.rec_32 & (oki.play_32_100 | oki.play_32_150 | oki.play_32_200)) | \
  (ima.rec_24 & ima.play_24) | \
  (ima.rec_32 & ima.play_32) | \
  (gsm_ms.frgsm_rec & gsm_ms.frgsm_play) | \
  g726.rec_32 | g726.play_32) )

# NOTE: If the DSP cores listed below do not exist on the board, the DSP cores
# will be ignored and will not be booted or used.
Resource[0].Dsps = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 \
                   24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 \
                   44 45 46 47

DebugMask                                       = 0x0
```

The following sample board keyword file sets up an IVR configuration with one conferencing resource spanning two DSPs on a CG 6060/C board:

```
#
#     c6060cnf_ivr_span.cfg
#     CG6060 configuration file
#
#     This file configures the board to run Natural Conferencing in spanning mode.
#

Clocking.HBus.ClockMode                         = STANDALONE
Clocking.HBus.ClockSource                       = OSC

DSPStream.VoiceIdleCode[0..15]                   = 0x7F
DSPStream.SignalIdleCode[0..15]                  = 0x00

#---------------------------
#  NOTE: T1 configuration
#---------------------------
NetworkInterface.T1E1[0..15].Type                = T1
NetworkInterface.T1E1[0..15].Impedance           = DSX1
NetworkInterface.T1E1[0..15].LineCode            = B8ZS
NetworkInterface.T1E1[0..15].FrameType           = ESF
NetworkInterface.T1E1[0..15].SignalingType       = RAW
DSP.C5x[0..47].Libs                              = cg6klibu
DSP.C5x[0..47].XLaw                              = MU_LAW

#---------------------------
#  NOTE: E1 configuration
#---------------------------
#NetworkInterface.T1E1[0..15].Type                 = E1
#NetworkInterface.T1E1[0..15].Impedance            = G703_120_OHM
#NetworkInterface.T1E1[0..15].LineCode             = HDB3
#NetworkInterface.T1E1[0..15].FrameType            = CEPT
#NetworkInterface.T1E1[0..15].SignalingType        = RAW
```

```
#DSP.C5x[0..47].Libs                                = cg6kliba
#DSP.C5x[0..47].XLaw                                = A_LAW

#---------------------------
# Hardware Echo Cancellation
# NOTE: it is in by pass by default
# NOTE: uncomment the following two keyword lines to enable and set the XLaw accordingly
#---------------------------
# HardwareEcho.EchoChipEnabled = YES
# HardwareEcho.XLaw = A_LAW

# NumTxTimeSlots, NumRxTimeSlots may be 16 or 32
DSP.C5x[0..1].NumTxTimeSlots  =   16
DSP.C5x[0..1].NumRxTimeSlots  =   16

DSP.C5x[0..1].CmdQStart    = 0xE800
DSP.C5x[0..1].DataInQStart = 0xF800
DSP.C5x[0..1].DspOutQStart = 0xFB00

# The following is required when #timeslots = 32 (but not for 16)
#DSP.C5x[0..1].CmdQSize     = 0x110

DSP.C5x[0..1].Libs[1]   = cg6mslib
DSP.C5x[0].Files        = cgcnfm
DSP.C5x[1].Files        = cgcnfs

SwitchConnectMode          =   AllConstantDelay
ConferencingStream.Enable  =   YES

#---------------------------
# Resource management
#---------------------------
Resource[0].Name                                = RSC1
Resource[0].Size                                = 120
Resource[0].TCPs                                = nocc
Resource[0].StartTimeSlot                       = 0

################################################################
# Before modifying this resource definition string refer to the CG 6060
# Installation and Developers Manual.
# NOTE: echo.ln20_apt25 - echo running on DSP has been removed
#       from resource definitions.  We recommend user to use
#       the hardware echo chip for echo cancellation instead
################################################################
Resource[0].Definitions         = ( dtmf.det_all & ptf.det_2f & tone.gen & \
    callp.gnc & ptf.det_4f & \
    ( (rvoice.rec_mulaw & rvoice.play_mulaw) | \
    (rvoice.rec_alaw & rvoice.play_alaw) | \
    (rvoice.rec_lin & rvoice.play_lin) | \
    (voice.rec_16 & (voice.play_16_100 | voice.play_16_150 | voice.play_16_200)) | \
    (voice.rec_24 & (voice.play_24_100 | voice.play_24_150 | voice.play_24_200)) | \
    (voice.rec_32 & (voice.play_32_100 | voice.play_32_150 | voice.play_32_200)) | \
    (voice.rec_64 & (voice.play_64_100 | voice.play_64_150 | voice.play_64_200)) | \
    (wave.rec_11_16b & wave.play_11_16b) | \
    (wave.rec_11_8b & wave.play_11_8b) | \
    (oki.rec_24 & (oki.play_24_100 | oki.play_24_150 | oki.play_24_200)) | \
    (oki.rec_32 & (oki.play_32_100 | oki.play_32_150 | oki.play_32_200)) | \
    (ima.rec_24 & ima.play_24) | \
    (ima.rec_32 & ima.play_32) | \
    (gsm_ms.frgsm_rec & gsm_ms.frgsm_play) | \
    g726.rec_32 | g726.play_32) )

# NOTE: If the DSP cores listed below do not exist on the board, the DSP cores will
# be ignored and will not be booted or used
Resource[0].Dsps = 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 \
                   24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42  \
                43 44 45 46 47

DebugMask                                       = 0x0
```

CG 6565/C sample board keyword files

The following sample board keyword file sets up an IVR configuration with one conferencing resource on a CG 6565/C board:

```
#
#     c565cnf_ivr_nonspan.cfg
#     CG 6565 configuration file
#
#     This file configures the board to run NautralConference in standalone
#     mode (non-spanning).
#
Clocking.HBus.ClockMode                          = STANDALONE
Clocking.HBus.ClockSource                        = OSC

DSPStream.VoiceIdleCode[0..15]                   = 0x7F
DSPStream.SignalIdleCode[0..15]                  = 0x00

#---------------------------
#  NOTE: T1 configuration
#---------------------------
NetworkInterface.T1E1[0..15].Type                = T1
NetworkInterface.T1E1[0..15].Impedance           = DSX1
NetworkInterface.T1E1[0..15].LineCode            = B8ZS
NetworkInterface.T1E1[0..15].FrameType           = ESF
NetworkInterface.T1E1[0..15].SignalingType       = RAW
DSP.C5x[0..95].Libs                              = cg6klibu
DSP.C5x[0..95].XLaw                              = MU_LAW

#---------------------------
#  NOTE: E1 configuration
#---------------------------
#NetworkInterface.T1E1[0..15].Type               = E1
#NetworkInterface.T1E1[0..15].Impedance          = G703_120_OHM
#NetworkInterface.T1E1[0..15].LineCode           = HDB3
#NetworkInterface.T1E1[0..15].FrameType          = CEPT
#NetworkInterface.T1E1[0..15].SignalingType      = RAW
#DSP.C5x[0..95].Libs                             = cg6kliba
#DSP.C5x[0..95].XLaw                             = A_LAW

#---------------------------
# Hardware Echo Cancellation
# NOTE: it is in by pass by default
# NOTE: uncomment the following two keyword lines to enable and set the XLaw
# accordingly
#---------------------------
# HardwareEcho.EchoChipEnabled = YES
# HardwareEcho.XLaw = A_LAW

# NumTxTimeSlots, NumRxTimeSlots may be 16 or 32
DSP.C5x[0].NumTxTimeSlots  =   16
DSP.C5x[0].NumRxTimeSlots  =   16

DSP.C5x[0].CmdQStart    = 0xE800
DSP.C5x[0].DataInQStart = 0xF800
DSP.C5x[0].DspOutQStart = 0xFB00

# The following is required when #timeslots = 32 (but not for 16)
#DSP.C5x[0].CmdQSize     = 0x110

DSP.C5x[0].Files        =   cg6conf.f41
SwitchConnectMode          =   AllConstantDelay
ConferencingStream.Enable  =   YES

#---------------------------
# Resource management
#---------------------------
Resource[0].Name                                 = RSC1
Resource[0].Size                                 = 120
Resource[0].TCPs                                 = nocc
Resource[0].StartTimeSlot                        = 0
```

```
#################################################################
# Before modifying this resource definition string refer to the CG 6565
# Installation and Developers Manual.
# NOTE: echo.ln20_apt25 - echo running on DSP has been removed
#       from resource definitions.  We recommend user to use
#       the hardware echo chip for echo cancellation instead.
#################################################################
Resource[0].Definitions         = ( dtmf.det_all & ptf.det_2f & tone.gen & \
  callp.gnc & ptf.det_4f & \
  ( (rvoice.rec_mulaw & rvoice.play_mulaw) | \
  (rvoice.rec_alaw & rvoice.play_alaw) | \
  (rvoice.rec_lin & rvoice.play_lin) | \
  (voice.rec_16 & (voice.play_16_100 | voice.play_16_150 | voice.play_16_200)) | \
  (voice.rec_24 & (voice.play_24_100 | voice.play_24_150 | voice.play_24_200)) | \
  (voice.rec_32 & (voice.play_32_100 | voice.play_32_150 | voice.play_32_200)) | \
  (voice.rec_64 & (voice.play_64_100 | voice.play_64_150 | voice.play_64_200)) | \
  (wave.rec_11_16b & wave.play_11_16b) | \
  (wave.rec_11_8b & wave.play_11_8b) | \
  (oki.rec_24 & (oki.play_24_100 | oki.play_24_150 | oki.play_24_200)) | \
  (oki.rec_32 & (oki.play_32_100 | oki.play_32_150 | oki.play_32_200)) | \
  (ima.rec_24 & ima.play_24) | \
  (ima.rec_32 & ima.play_32) | \
  (gsm_ms.frgsm_rec & gsm_ms.frgsm_play) | \
  g726.rec_32 | g726.play_32) )

# NOTE: If the DSP cores listed below do not exist on the board, the DSP cores
# will be ignored and will not be booted or used
Resource[0].Dsps = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 \
                   24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 \
                   44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 \
                   64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 \
                   84 85 86 87 88 89 90 91 92 93 94 95

DebugMask                                       = 0x0
```

The following sample board keyword file sets up an IVR configuration with one conferencing resource spanning two DSPs on a CG 6565/C board:

```
#
#     c565cnf_ivr_span.cfg
#     CG 6565 configuration file
#
#     This file configures the board to run Natural Conferencing in spanning mode.
#

Clocking.HBus.ClockMode                         = STANDALONE
Clocking.HBus.ClockSource                       = OSC

DSPStream.VoiceIdleCode[0..15]                   = 0x7F
DSPStream.SignalIdleCode[0..15]                  = 0x00

#---------------------------
#  NOTE: T1 configuration
#---------------------------
NetworkInterface.T1E1[0..15].Type                = T1
NetworkInterface.T1E1[0..15].Impedance           = DSX1
NetworkInterface.T1E1[0..15].LineCode            = B8ZS
NetworkInterface.T1E1[0..15].FrameType           = ESF
NetworkInterface.T1E1[0..15].SignalingType       = RAW
DSP.C5x[0..95].Libs                              = cg6klibu
DSP.C5x[0..95].XLaw                              = MU_LAW

#---------------------------
#  NOTE: E1 configuration
#---------------------------
#NetworkInterface.T1E1[0..15].Type                 = E1
#NetworkInterface.T1E1[0..15].Impedance            = G703_120_OHM
#NetworkInterface.T1E1[0..15].LineCode             = HDB3
#NetworkInterface.T1E1[0..15].FrameType            = CEPT
#NetworkInterface.T1E1[0..15].SignalingType        = RAW
```

```
#DSP.C5x[0..95].Libs                                = cg6kliba
#DSP.C5x[0..95].XLaw                                = A_LAW

#---------------------------
# Hardware Echo Cancellation
# NOTE: it is in by pass by default
# NOTE: uncomment the following two keyword lines to enable and set the XLaw accordingly
#---------------------------
# HardwareEcho.EchoChipEnabled = YES
# HardwareEcho.XLaw = A_LAW

# NumTxTimeSlots, NumRxTimeSlots may be 16 or 32
DSP.C5x[0..1].NumTxTimeSlots  =   16
DSP.C5x[0..1].NumRxTimeSlots  =   16

DSP.C5x[0..1].CmdQStart    = 0xE800
DSP.C5x[0..1].DataInQStart = 0xF800
DSP.C5x[0..1].DspOutQStart = 0xFB00

# The following is required when #timeslots = 32 (but not for 16)
#DSP.C5x[0..1].CmdQSize     = 0x110

DSP.C5x[0..1].Libs[1]   = cg6mslib
DSP.C5x[0].Files        = cgcnfm
DSP.C5x[1].Files        = cgcnfs

SwitchConnectMode          =   AllConstantDelay
ConferencingStream.Enable  =   YES

#---------------------------
# Resource management
#---------------------------
Resource[0].Name                                = RSC1
Resource[0].Size                                = 120
Resource[0].TCPs                                = nocc
Resource[0].StartTimeSlot                       = 0

################################################################
# Before modifying this resource definition string refer to the CG 6565
# Installation and Developers Manual.
# NOTE: echo.ln20_apt25 - echo running on DSP has been removed
#       from resource definitions.  We recommend user to use
#       the hardware echo chip for echo cancellation instead
################################################################
Resource[0].Definitions         = ( dtmf.det_all & ptf.det_2f & tone.gen & \
    callp.gnc & ptf.det_4f & \
    ( (rvoice.rec_mulaw & rvoice.play_mulaw) | \
    (rvoice.rec_alaw & rvoice.play_alaw) | \
    (rvoice.rec_lin & rvoice.play_lin) | \
    (voice.rec_16 & (voice.play_16_100 | voice.play_16_150 | voice.play_16_200)) | \
    (voice.rec_24 & (voice.play_24_100 | voice.play_24_150 | voice.play_24_200)) | \
    (voice.rec_32 & (voice.play_32_100 | voice.play_32_150 | voice.play_32_200)) | \
    (voice.rec_64 & (voice.play_64_100 | voice.play_64_150 | voice.play_64_200)) | \
    (wave.rec_11_16b & wave.play_11_16b) | \
    (wave.rec_11_8b & wave.play_11_8b) | \
    (oki.rec_24 & (oki.play_24_100 | oki.play_24_150 | oki.play_24_200)) | \
    (oki.rec_32 & (oki.play_32_100 | oki.play_32_150 | oki.play_32_200)) | \
    (ima.rec_24 & ima.play_24) | \
    (ima.rec_32 & ima.play_32) | \
    (gsm_ms.frgsm_rec & gsm_ms.frgsm_play) | \
    g726.rec_32 | g726.play_32) )

# NOTE: If the DSP cores listed below do not exist on the board, the DSP cores will
# be ignored and will not be booted or used
Resource[0].Dsps = 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 \
                   24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42  \
                43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61  \
                62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80  \
                81 82 83 84 85 86 87 88 89 90 91 92 93 94 95

DebugMask                                       = 0x0
```

Setting up the NaturalConference configuration file

After modifying the board keyword file, create a NaturalConference configuration file (*cnf.cfg)* to declare conferencing resources.

Use the NaturalConference configuration file to assign DSPs from a given board to a numbered conferencing resource. A NaturalConference application can open this resource using **cnfOpenResource**.

You can also use *cnf.cfg* to

- Disable the resource capabilities that you do not need, potentially increasing the number of members supported on a given resource.
- Add a new conferencing resource without having to modify the application. You can take advantage of the new resource by using **cnfGetResourceList** to retrieve a list of available resources.
- Assign a conferencing resource to a new DSP without having to recompile the application.

This topic presents the following information:

- Customizing *cnf.cfg* for the system
- File structure
- File statements

Customizing cnf.cfg for the system

To create a *cnf.cfg* for the system, modify one of the sample files installed with NaturalConference. The sample files are installed in the following directories:

Operating system	Directory
Windows	*\nms\ctaccess\cfg*
UNIX	*/opt/nms/ctaccess/cfg*

The following table lists the NaturalConference sample configuration files:

Board	NaturalConference configuration file	Board keyword file	Description
AG 2000/200 AG 2000C/200 AG 2000-BRI/200	*cnf_ag2_200.cfg*	*agpi2cnf_200.cfg* *agpi2cnfa_200.cfg*	One conferencing resource with one DSP
AG 2000/400 AG 2000C/400 AG 2000-BRI/400	*cnf_ag2_400.cfg*	*agpi2cnf_400.cfg* *agpi2cnfa_400.cfg*	One conferencing resource with three DSPs
CG 6060/C	*cnf_c6060cnf_ivr_nonspan.cfg*	*c6060cnf_ivr_nonspan.cfg*	One conferencing resource on one DSP
CG 6060/C	*cnf_c6060cnf_ivr_span.cfg*	*c6060cnf_ivr_span.cfg*	One conferencing resource spanning two DSPs
CG 6565/C	*cnf_c6565cnf_ivr_nonspan.cfg*	*c6565cnf_ivr_nonspan.cfg*	One conferencing resource on one DSP
CG 6565/C	*cnf_c6565cnf_ivr_span.cfg*	*c6565cnf_ivr_span.cfg*	One conferencing resource spanning two DSPs

NaturalConference configuration file structure

The NaturalConference configuration file begins with a [CONFERENCING] section. Statements in this section are global statements that apply to all resources defined in the file.

Each conferencing resource is defined in a [Resource ***n***] section. Statements in this section are local statements that apply only to the designated resource. The numbering of a resource is user-defined and may be defined in an unordered fashion. Valid resource numbers are from 0 through 4095.

A statement has the general syntax keyword = ***value***. Not all keywords can be used in global statements.

White space, such as indentations or a space around an equals sign, is ignored but is useful for clarity. The number sign (#) and semicolon (;) are both comment delimiters. NaturalConference ignores any text that follows a comment delimiter character on the same line. Case is ignored for Windows but is important for file names for UNIX-based operating systems.

The following example is an excerpt from a typical NaturalConference configuration file for an AG or a CG board:

```
[CONFERENCING] # Start of active region

Flags = NO_ECHO_CANCEL | NO_COACHING # Common statements

[Resource 0] # Start of a resource-specific section for Resource 0

    Board = 0   # Resource-specific statements ...
    DSP = 3
    Flags = NO_DTMF_CLAMPING


[Resource 1] # Start of a resource-specific section for Resource 1

    Board = 1               # Resource-specific statements ...
    DSP = 10


[Resource 2] # Start of a resource-specific section for Resource 2

    Board = 2               # Resource-specific statements ...
    DSP = 10


# EOF (End of active regions)
```

NaturalConference configuration file statements

The following table summarizes the NaturalConference configuration file statements. Brackets ([ ]) signify an optional value. A vertical bar (|) signifies a choice of more than one value.

Keyword	Global	Local	Allowed values	Specifies...	Required?
Board	x	x	0..***n***	The board number as defined in the board keyword file. Every board in the system must be assigned a unique board number.	Yes
DSP		x	[dsp1[..dsp***n***]]	The DSP number to be used by this resource.	Yes
Flags	x	x	EC_10_MS_WINDOW EC_20_MS_WINDOW NO_ECHO_CANCEL EC_100_MS_CV EC_200_MS_CV NO_DTMF_CLAMPING NO_TONE_CLAMPING NO_COACHING	The echo cancellation window size, echo cancellation convergence time, and the capabilities the application will not use.	No
NumTimeSlots	x	x	16, 32	The number of timeslots (translates into the number of conferees) per DSP on a CG board. Must be identical to the value of the NMS OAM keywords DSP.C5x[].NumRxTimeSlots and DSP.C5x[].NumTxTimeSlots.	No (defaults to 16). Ignored on AG boards.

Using flags

The Flags statement is optional. Use this statement to specify the capabilities the application will not use. By disabling certain capabilities, you free up system resources, making it possible to increase the number of members supported on a given resource. If you specify no flags, all capabilities currently supported by the loaded DSP module are available. Applications cannot dynamically activate a capability at the conference or member level if the capability is disabled in the configuration file.

Flags that you specify in the [CONFERENCING] section apply to every conferencing resource declared in the configuration file, unless flags are specified for an individual resource, as in the following example:

```
[CONFERENCING]
Flags = NO_DTMF_CLAMPING | NO_COACHING

[RESOURCE 0]
Flags = NO_ECHO_CANCEL | NO_TONE_CLAMPING
```

In this example, the flags specified for resource 0 override the flags set for the conference. The resulting flags for resource 0 are NO_ECHO_CANCEL | NO_TONE_CLAMPING.

Sample NaturalConference configuration files

This topics presents sample NaturalConference configuration files for:

- AG 2000/400
- CG 6060/C
- CG 6565/C

AG 2000/400 sample cnf.cfg

The following NaturalConference configuration file excerpt for an AG 2000/400 board defines one conferencing resource with three DSPs on Board 0:

```
#                        cnf_ag2_400.cfg
#
#      --- NaturalConference Configuration File --
#
# Sample configuration file for AG 2000
#
# This file is associated with the AG configuration files agpi2cnf_400.cfg
# and agpi2cnfa_400.cfg installed in the \nms\ag\cfg directory.
#

[CONFERENCING]

# Flags = NO_ECHO_CANCEL | NO_DTMF_CLAMPING
Flags = NO_COACHING

[RESOURCE 0]

Board = 0
DSP = 1..3
```

CG 6060/C sample cnf.cfg

The following NaturalConference configuration file for a CG 6060/C board defines one conferencing resource on one DSP:

```
#                        cnf_c6060cnf_ivr_nonspan.cfg
#
#      --- NaturalConference Configuration File --
#
# Sample configuration file for CG 6060
#
# This file is associated with the CG configuration file c6060cnf_ivr_nonspan.cfg.
#

[CONFERENCING]

Flags = NO_ECHO_CANCEL | NO_DTMF_CLAMPING | NO_COACHING | NO_TONE_CLAMPING


[RESOURCE 0]

Board = 0
NumTimeSlots = 16
DSP = 0
```

The following NaturalConference configuration file for a CG 6060/C board defines one conferencing resource spanning two DSPs:

```
#                       cnf_c6060cnf_ivr_span.cfg
#
#      --- NaturalConference Configuration File --
#
# Sample configuration file for CG 6060
#
# This file is associated with the CG configuration file c6060cnf_ivr_span.cfg.
#

[CONFERENCING]

Flags = NO_ECHO_CANCEL | NO_DTMF_CLAMPING | NO_COACHING | NO_TONE_CLAMPING


[RESOURCE 0]

Board = 0
NumTimeSlots = 16
DSP = 0..1
```

CG 6565/C sample cnf.cfg

The following NaturalConference configuration file for a CG 6565/C board defines one conferencing resource on one DSP:

```
#                       cnf_c6565cnf_ivr_nonspan.cfg
#
#      --- NaturalConference Configuration File --
#
# Sample configuration file for CG 6565
#
# This file is associated with the CG configuration file c6565cnf_ivr_nonspan.cfg.
#

[CONFERENCING]

Flags = NO_ECHO_CANCEL | NO_DTMF_CLAMPING | NO_COACHING | NO_TONE_CLAMPING


[RESOURCE 0]

Board = 0
NumTimeSlots = 16
DSP = 0
```

The following NaturalConference configuration file for a CG 6565/C board defines one conferencing resource spanning two DSPs:

```
#                       cnf_c6565cnf_ivr_span.cfg
#
#      --- NaturalConference Configuration File --
#
# Sample configuration file for CG 6565
#
# This file is associated with the CG configuration file c6565cnf_ivr_span.cfg.
#

[CONFERENCING]

Flags = NO_ECHO_CANCEL | NO_DTMF_CLAMPING | NO_COACHING | NO_TONE_CLAMPING


[RESOURCE 0]

Board = 0
NumTimeSlots = 16
DSP = 0..1
```

Verifying NaturalConference

After you install and configure NaturalConference, verify that the system is operational:

- Verify that the NaturalConference software is successfully loaded.
- Verify the NaturalConference configuration.
- Test your system's ability to receive calls and add them to a conference.

Verifying the NaturalConference installation

Use the Natural Access version checker utility, *ctavers*, to verify that the NaturalConference software is successfully loaded. *ctavers* verifies that all the Natural Access libraries defined in the Natural Access configuration file (*cta.cfg*) are accessible.

To run *ctavers*, enter the following command:

```
ctavers
```

ctavers displays a list of all the installed Natural Access components. Verify that NaturalConference is installed. In the following example, the NaturalConference statements are shown in bold:

```
H:\>ctavers -x cnf,cnfmgr
Natural Access Version Checker Utility  V.5 (Jun 29 2001)

Natural Access Release:             4.03
Natural Access Compatibility Level: 2
CTA: DISPATCHER      v 4.03 Jun 29 2001 compat: expdisp=1  expap
Mgr: CNFMGR          v 2.00 Jul 17 2001 compat: reqdisp=1
Svc: CNF   (CNFMGR ) v 2.00 Jul 17 2001 compat: reqdisp=1  expap
```

Verifying the NaturalConference configuration

Use the NaturalConference utility, *cnfinfo*, to verify that the NaturalConference software is properly configured. *cnfinfo* checks all the boards currently loaded on your system and opens all the resources defined in the NaturalConference configuration file (*cnf.cfg*). Refer to *Setting up the NaturalConference configuration file* on page 37 for more information.

To run *cnfinfo*, enter the following command:

```
cnfinfo
```

cnfinfo displays a list of the installed boards and the defined resources, including the maximum number of members with all capabilities, and the maximum number of members with all capabilities except echo canceller or DTMF clamping. You can evaluate how many NaturalConference ports are available when using the capabilities of each resource.

For example:

```
H:\>cnfinfo
CNFINFO: NaturalConference Utility V2.0 (Jul 25 2001)



                           |  Rs/Bd    |
                           |   0/0     |
|--------------------------|-----------|
|Features                  | 0x20ff    |
|Full                      |   24      |
|Wo (E)                    |   24      |
|Wo (D)                    |   24      |
|Wo (T)                    |   24      |
|Wo (Co)                   |   50      |
|Wo (E & D)                |   24      |
|Wo (E & T)                |   24      |
|Wo (E & Co)               |   64      |
|Wo (D & T)                |   24      |
|Wo (D & Co)               |   50      |
|Wo (T & Co)               |   50      |
|Wo (E & D & T)            |   24      |
|Wo (E & D & Co)           |   64      |
|Wo (E & T & Co)           |   64      |
|Wo (D & T & Co)           |   60      |
|Wo (E & D & T & Co)       |   64      |
|--------------------------|---------- |
--------------------------------------------------------------------------------

(Type "cnfinfo -a" to have details on the abbreviations used in this table)

Board  CNF Resources   CNF Ports
-----  -------------  ---------
   0        1            24
```

Receiving calls and adding callers to a conference

Use the NaturalConference sample program, *cnfjoin*, to test your system's ability to receive calls and add them to a conference.

cnfjoin opens the first conference resource available and opens a designated number of channels for call control. When *cnfjoin* receives a call, it immediately adds the caller to a conference. All callers are added to the same conference. When a caller disconnects, it is removed from the conference and another caller can take its place.

The *cnfjoin* program is based on Natural Call Control (NCC) and can be used either with ISDN, CAS, or analog protocols.

To use *cnfjoin*, call control and all conference resources must reside on the same board. If you are using the default configuration files provided by the installation, your system is properly configured to use *cnfjoin* since these files configure call control and conferencing on the same board.

For more information, refer to *cnfjoin overview* on page 107.

5 Optimizing performance

Managing resources

This section explains how to configure the AG and CG boards in the system to optimize the performance of a NaturalConference application.

To increase the number of members in a conference:

- Create conferencing resources with more DSPs.
- Chain conferences at the application level.
- Disable capabilities either at the resource level or at the conference level. For example, disable coaching in one of the following ways:
    - At the resource level by setting the NO_COACHING flag in *cnf.cfg*.
    - At the conference level by setting the CNF_NO_COACHING flag in **cnfCreateConference**.

This topic presents the following information:

- Conferencing resource and capacity limitations
- Managing available members
- Resource management example

Conferencing resource and capacity limitations

The following table describes the factors that can limit the number of members that a conferencing resource can manage:

Limiting factor	Description
DSP memory limitation	Can occur on an AG board when using echo cancellation with 20 ms tail. This limitation causes the lowest DSP capacity by allowing no more than nine members per DSP.
MIPS limitation	Set at 133 MIPS per DSP on CG boards. This limitation factor applies when using a MIPS consuming capability like echo cancellation and tone clamping. This limitation is reached on the master DSP when implementing a spanned conferencing resource. When using echo cancellation, the lowest DSP capacity reached is 16 members per DSP.
Number of voice channels a DSP has access to	On AG boards, a DSP has access to 64 voice channels bringing the overall limitation of members per conferencing resource to 64. This number may vary depending on the capabilities being used versus the number of DSPs working together as a single resource. On CG boards, all DSPs are connected in groups of four to streams with 128 channels. Therefore, all DSPs can support up to 32 members.
Number of members	When coaching is not used, a single conference can use the full conferencing resource. If coaching is used in a conference, the number of members per conference is limited to 32. On AG boards, there is a limit of eight active talkers per conference. On CG boards, the limit is 16.
Private link	A private link can be implemented between each DSP according to each board type. Because a private link does not exist on AG boards, the previous limitation (number of voice channels) is the actual limitation on AG boards. On CG boards, private links exist in some DSP groups. This private link enables you to bypass the previous limitation so you can implement the master/slave model (spanned resource). Because DSP organization is board type dependent, refer to the *readme* file and to the sample NaturalConference configuration files delivered with the software to understand which DSPs can be used together to build a spanned conferencing resource.
Switching matrix limitation	Members are connected to conferencing resources through the switching matrix, regardless of whether the member comes from the CT bus or from the local trunks. The primary limitation is the number of connections to and from the CT bus and can vary according to each board family, model, and revision. The limitation can be 128 duplex (256 connections), 256 duplex, 2 x 256 duplex, or more. Refer to the board-specific documentation for information about the actual switching limitations.

Obtaining actual conferencing resource capacity

The actual conferencing resource capacity is variable and sometimes dynamic. To obtain precise numbers, the configuration must be installed and the conferencing firmware loaded onto the DSP.

Run the *cnfinfo* utility delivered with NaturalConference for a detailed list of the capabilities being used and the precise capacity for each conferencing resource configured. The *readme* file delivered with NaturalConference describes the firmware, its characteristics, and information about its capacity.

Consider the following performance limitations:

AG board:

- With 20 ms echo cancellation: 12 members per DSP
- With 10 ms echo cancellation: 16 members per DSP
- Without echo cancellation, DTMF, and tone clamping: 32 members per DSP
- Maximum number of DSPs being used in a single conferencing resource (spanning): 16
- Maximum capacity per conferencing resource: 64 members

CG board:

- With 20 ms echo cancellation: 12 members per DSP
- With 10 ms echo cancellation: 16 members per DSP
- Without echo cancellation, DTMF, and tone clamping: 32 members per DSP
- Maximum number of DSPs being used in a single conferencing resource (spanning): 8
- Maximum capacity per conferencing resource: 128 members

Managing available members

You can manage the available members for a conference in the following ways:

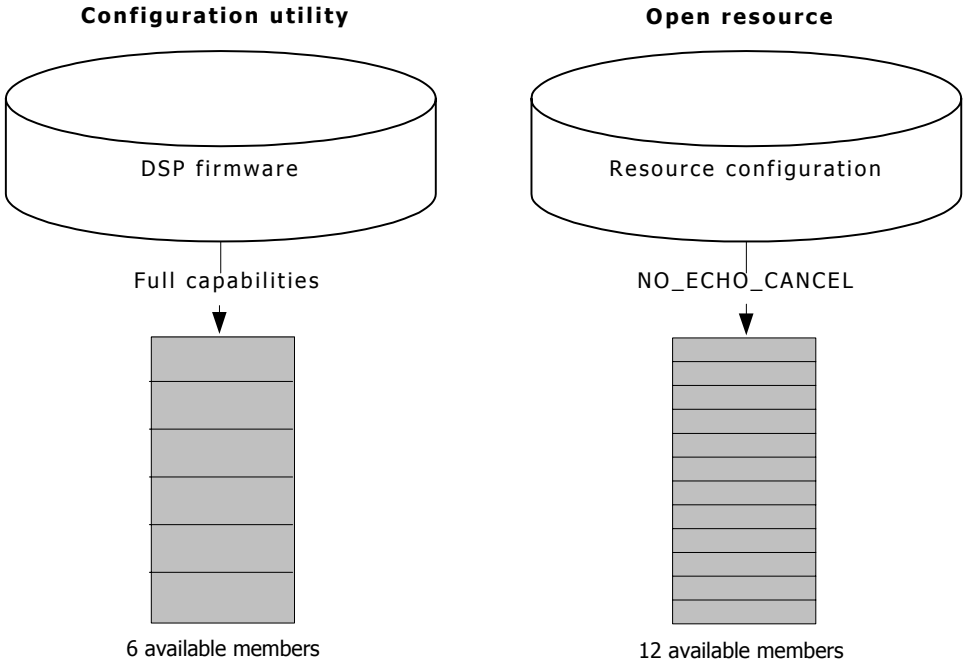
- Use **cnfGetResourceInfo** to determine how many members you can add to a conference. This function gives you the maximum number of members the resource can accept if using the full capabilities of the resource, as well as the number of members available for a new conference.
- If you know before opening the resource that you never use certain capabilities, disable the capabilities in the *cnf.cfg* by using the Flags keyword as described in *NaturalConference configuration file statements* on page 39. The information retrieved by **cnfGetResourceInfo** will be closer to your needs.
- If you intend to create a conference using fewer capabilities than the resource allows, specify the unused capabilities for this resource in the CNF_CONFERENCE_PARMS structure when you create the conference. When the conference is created, you may have more available members on this resource than you expected. Then you can invoke **cnfResizeConference** to increase the number of available members. If you resize your conference progressively (adding one member each time) you will reach the real limit of the resource.

Resource management example

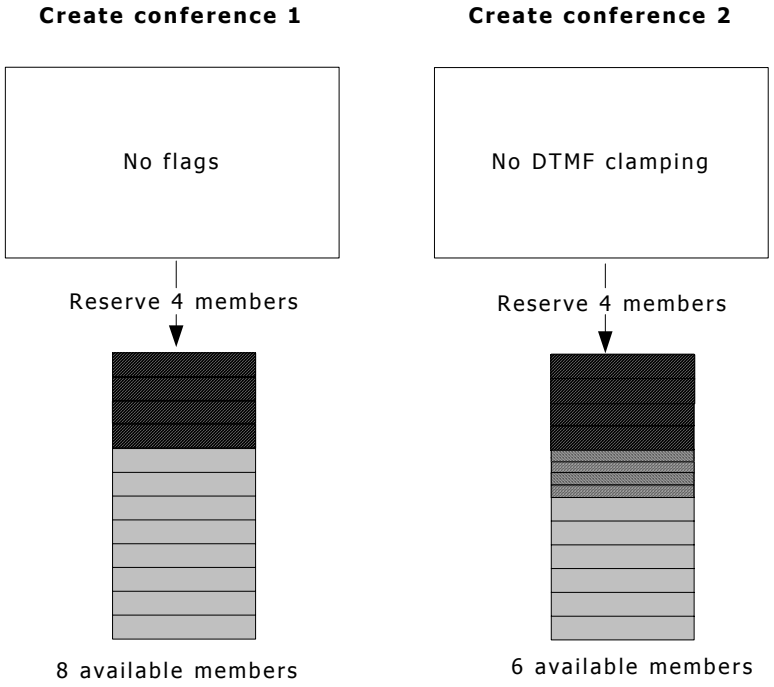
The following example illustrates resource management:

1. A call to **cnfGetResourceInfo** indicates that the resource can handle six members with full capabilities.

2. Because the application does not require echo cancellation, the keyword flags=NO_ECHO_CANCEL is specified in the *cnf.cfg*. The resource is opened and **cnfGetResourceInfo** is invoked, indicating that the number of available members has increased to 12.



3. A new conference with full capabilities and four members is created, reducing the number of available members to eight.
4. When a second conference with no DTMF clamping and four members is created, the number of available members is only reduced by two (to six available members).



Port density examples for AG boards

This topic presents examples of port densities that can be achieved when configuring AG boards for developing voice, fax, and conference applications:

The following AG board port density examples assume:

- IVR requires:

Capabilities	MIPS	DSP modules
Echo cancellation (16 ms window length, 400 ms adapt time)	3.63	*echo.m54*
DTMF and silence detector	1.94	*dtmf.m54*
Call progress	1.06 (AG 2000 only)	*callp.m54*
Precise tone filter (2 single frequency or 1 tone pair)	1.25	*ptf.m54*
NMS record 32 Kbit/s	3.38	*voice.m54*
	Total: 11.26 MIPS per channel	

  For a 100 MIPS device (90 MIPS effectively), the required number of IVR channels is eight (90 divided by 11.26 and rounded to the nearest whole number).

- Fax could be used either in reception or transmission in V17, V27, or V29, requiring eight channels per DSP.
- NaturalConference can support 16 members per DSP with all capabilities activated (10 ms echo cancellation window length).

The following table provides AG board port density examples:

Board name	Channels	NaturalConference ports
AG 2000/200-8L AG 2000C/200-8L AG 2000-BRI/200-8L	8 IVR channels	16 ports of NaturalConference with echo cancellation, DTMF clamping, and tone clamping.
AG 2000/200-8L AG 2000C/200-8L AG 2000-BRI/200-8L	None (when using a dedicated conferencing resource board.)	48 to 92 ports of NaturalConference.
AG 2000/400-8L AG 2000C/400-8L AG 2000-BRI/400-8L	8 IVR channels and 8 FAX channels	32 ports of NaturalConference with echo cancellation, DTMF clamping, and tone clamping.
AG 2000/400-8L AG 2000C/400-8L AG 2000-BRI/400-8L	None (when using a dedicated conferencing resource board.)	48 to 96 ports of NaturalConference. (Due to memory limitations of the coprocessor program of the AG 2000/C and AG 2000-BRI boards.)

Port density examples for CG boards

This topic presents the following test case scenarios for running NaturalConference on CG boards:

- PSTN based cases
- IP based cases

PSTN based conferencing test cases

The following CG board port density examples assume:

- IVR ports: call progress, DTMF detect, play and record with most encoding formats, or fax
- NaturalConference ports: echo cancellation with 10 ms echo tail, 200 ms convergence time, DTMF clamping, no coaching)

You can run eight IVR ports per DSP.

Board name	IVR ports	NaturalConference ports
CG 6060/32	120 IVR ports - 4 T1/E1 trunks	128 NaturalConference ports with echo cancellation and DTMF clamping
CG 6060C/32	120 IVR ports - 4 T1/E1 trunks	128 NaturalConference ports with echo cancellation and DTMF clamping
CG 6565/64	240 IVR ports - 8 T1/E1 trunks	256 NaturalConference ports with echo cancellation and DTMF clamping
CG 6565C/128	480 IVR ports - 16 T1/E1 trunks	512 NaturalConference ports with echo cancellation and DTMF clamping

IP based conferencing test cases

The following table presents the CG board capacities for the specified IP based conferencing test cases:

	Test case	
Board name	**Three party IP conferencing with G.711**	**Three party IP conferencing with G.711 and G.729A**
CG 6060/32	120	128
CG 6060C/32	120	128
CG 6565/64	240	128
CG 6565C/128	360	256

The test cases represent typical three party IP conferencing use scenarios. Each IP port for these test cases consists of a conferencing seat and a Fusion port:

Test case	Description
Three party IP conferencing with G.711	All parties are established in the conference as eligible talkers. **Each conferencing seat:** Full duplex G.711 (TDM) Automatic gain control (AGC) enabled **Each Fusion port:** Full duplex G.711 with 20 ms packets, detection of VAD and DTMF enabled, and silence suppression disabled to present a worst-case scenario.
Three party IP conferencing with G.711 and G.729A	All parties are established in the conference as eligible talkers. **Each conferencing seat:** Full duplex G.711 (TDM) Automatic gain control (AGC) enabled **Each Fusion port:** Full duplex G.711 or G.729A with 20 ms packets, detection of VAD and DTMF enabled, and silence suppression disabled to present a worst-case scenario.

6 Developing applications

Opening a conferencing resource

Opening the CNF service with **ctaOpenServices** enables the ***ctahd*** to access to the NaturalConference API. The next step is to open a conferencing resource and obtain a resource handle (***cnfresourcehd***) to perform actions on the resource.

The following table lists the functions to use when opening a conference resource:

To...	Description
Retrieve a list of resource numbers defined in the system, call **cnfGetResourceList** with the appropriate parameters.	Returns the resource numbers defined in the NaturalConference configuration file (*cnf.cfg*). **Note:** By modifying *cnf.cfg* you can increase or reduce the number of resources without having to recompile the program.
Obtain a conferencing resource handle (***cnfresourcehd***), call **cnfOpenResource** using the resource number obtained with **cnfGetResourceList**.	Returns a resource handle (***cnfresourcehd***). This conferencing resource handle has a reference to the ***ctahd*** used when opening the resource. Each time you access the conference and member objects, use the ***cnfresourcehd*** instead of the ***ctahd***. Wait for the CNFEVN_OPEN_RESOURCE_DONE event before using the ***cnfresourcehd.*** This event verifies that the ***cnfresourcehd*** was allocated (especially important when using the Server mode of Natural Access).
Retrieve information associated with a conferencing resource, call **cnfGetResourceInfo** with the returned ***cnfresourcehd***.	Returns a CNF_RESOURCE_INFO structure containing information such as the capabilities of the resource, the maximum number of members the resource can accept, and the number of members available for a new conference. CNF_RESOURCE_INFO also gives you information about the AG or CG board type and board number where the resource is located as defined in *cnf.cfg*. Use this information to open a switching handle (***swihd***) associated with this resource. The ***swihd*** is invoked when connecting a member of a conference to the trunk interfaces (refer to *Establishing the connection* on page 55).

Placing a call

To place a call, invoke **nccPlaceCall**. Manage the call placement event sequence until the call is connected and NCCEVN_CALL_CONNECTED is generated. Refer to the *Dialogic® NaturalAccess™ NaturalCallControl™ API Developer's Manual.*

Creating conferences

This topic presents:

- Creating a conference
- Adding members to a conference

Creating a conference

After opening a conference resource and reviewing the resource capabilities, you can create a new conference. The following table lists the functions to use when creating a new conference:

To...	Description
Create a new conference, call **cnfCreateConference** with ***cnfresourcehd*** and the appropriate parameters.	Returns a conference identifier (***confid***) to use when accessing the conference object. To guarantee that you are able to add a certain number of members to the conference, or if you know in advance how many members will participate in this conference, you can allocate a certain number of members when calling **cnfCreateConference**. You can also open a conference and allocate no members. You may be able to add members on the fly if the conference resource has room. When calling **cnfCreateConference** you can use fewer capabilities than the resource supports, enabling you to add more members to the conference. For more information, refer to the section on Optimizing performance.
Verify changes at the resource level after creating the conference, call **cnfGetResourceInfo**.	Returns a CNF_RESOURCE_INFO structure containing information such as the capabilities of the resource, the maximum number of members the resource can accept, and the number of members available for a new conference. **Note:** After calling **cnfCreateConference**, the number of conferences increases and the number of available members decreases.
Redefine the number of members allocated to a conference, call **cnfResizeConference** with appropriate parameters.	Enables you to increase or decrease the number of members allocated to the conference.

Retrieve a list of conferences, call **cnfGetConferenceList** with appropriate parameters.	Returns the ***confidlist***.
	By performing a loop using the returned ***confidlist***, you can get information about each conference and retrieve the user value. For example:
	`for (confindex = 0; confindex < numconfid; confindex++)`
	`{`
	`error = cnfGetConferenceInfo(cnfresourcehd,`
	`confidlist[confindex], &conferenceinfo,`
	`sizeof(CNF_CONFERENCE_INFO));`
	`.`
	`.`
	`.`
	`}`

Adding members to a conference

To add members to a conference, follow these steps:

Step	Action	Function
1	Add the member to the conference.	**cnfJoinConference** with appropriate parameters. If the call succeeds, the member identifier (***memberid***) is returned and can be used by the application for any function call requiring a ***memberid***.
2	Retrieve the member's information.	**cnfGetMemberInfo** using a valid ***memberid.*** Returns a CNF_MEMBER_INFO structure containing switching related information such as the input MVIP-95 local stream (even value) and the timeslot number for the member. To get the output MVIP-95 local stream, increment the input local stream by one.
3	On AG and CG boards, establish a connection between the member's telephone line and the conference bridge.	**swiMakeConnection** For more information, see *Establishing the connection* on page 55.
4	Set up one or more member attributes and activate the member in the conference.	**cnfSetMemberAttribute** or **cnfSetMemberAttributeList** For example, the talker attribute is set to 0 by default. To enable the new member to speak and be heard: `error = cnfSetMemberAttribute(cnfresourcehd, memberid, MEMBER_ATTR_TALKER_ENABLE, 1);` For more information, see *Setting member attributes* on page 58.

Establishing the connection

After you add a member to a conference, call **swiMakeConnection** to set up the connection between the member's telephone line and the conference bridge. When the connection is established, streams are reserved for NaturalConference. Use *showcx95* to determine the number of timeslots available for conferencing.

To enable a NaturalConference stream, the following statement must be added to the board keyword file:

```
ConferencingStream.Enable = YES
```

For more information about the board keyword file, refer to *Modifying the board keyword file* on page 27.

The following table lists the number of reserved timeslots by board type. Not all timeslots are used. The timeslot range does not have to be consecutive.

Board type	Number of reserved timeslots
AG Series	256
CG Series	128 (Default value. This can be changed.)

To change the default number of timeslots allocated to a logical conferencing stream (usually 256), add the following statement to the board keyword file:

```
ConferencingStream.SlotCount = xxx
```

The following example code connects a member to the trunk interfaces (MVIP-95 local streams 0 and 1):

```
error = cnfGetMemberInfo(cnfresourcehd, memberid, &memberinfo,
                         sizeof(CNF_MEMBER_INFO));

/* Allocate SWI_TERMINUS input structure using memberinfo.stream and memberinfo.timeslot.
 */
SWI_TERMINUS input = { MVIP95_LOCAL_BUS, memberinfo.stream,
                         memberinfo.timeslot };

/* Allocate SWI_TERMINUS output structure for the trunk interface
(stream 1) */
SWI_TERMINUS output = { MVIP95_LOCAL_BUS, 1, trunk.timeslot };

/* Full duplex connection */
error = swiMakeConnection(swihd, &input, &output, 1);
input.stream += 1;
output.stream -= 1;
error = swiMakeConnection(swihd, &output, &input, 1);
```

Refer to the *Dialogic® NaturalAccess™ Switching Interface API Developer's Manual* for more information about switching.

Using the Point-to-Point Switching service

To use the Point-to-Point Switching (PPX) service to establish the connection between the trunk streams and the member's conference seat, update *ppx.cfg* with local streams 32 and 33 or local streams 64 and 65. For example:

```
Inputs
   LOCAL:0..14(2):0..23   # Trunk (0..29 for E1)
   LOCAL:16..18(2):0..127 # DSP
   LOCAL:32:0..256 # Conferencing stream
End Inputs
Outputs
   LOCAL:1..15(2):0..23   # Trunk (0..29 for E1)
   LOCAL:17..19(2):0..127 # DSP
   LOCAL:33:0..256 # Conferencing stream
End Outputs
```

The following table shows the NaturalConference logical stream assignments by board type:

Board type	NaturalConference logical stream
AG Series	32 and 33
CG Series	68 and 69

Refer to the *Dialogic® NaturalAccess™ Point-to-Point Switching API Developer's Manual* for more information.

Managing conferences

This topic presents:

- Setting conference attributes
- Setting member attributes
- Playing a tone
- NaturalConference events
- Closing a conference

Setting conference attributes

Conferences are created with both parameters and attributes. Conference parameters contain information about a given conference, and cannot change over the life of the conference. Attributes denote certain special characteristics that a given conference possesses, and change frequently during the life of the conference.

You retrieve conference parameters and other information together in a corresponding CNF_CONFERENCE_INFO structure, while you access attributes individually.

You can assign the following types of attributes to a conference:

- Loudest speaker
- Event mask
- Active talkers
- Active talkers timer

For a description of conference attributes, refer to **cnfGetConferenceAttribute**.

The following table lists the functions for accessing conference information and attributes:

To...	Description
Retrieve conference information, call **cnfGetConferenceInfo**	Returns a CNF_CONFERENCE_INFO structure containing information such as the number of members allocated, the number of members attending, and the conference capabilities.
Retrieve a conference attribute, call **cnfGetConferenceAttribute**	Returns the value of the specified conference attribute.
Set a conference attribute, call **cnfSetConferenceAttribute**	Sets the value of the specified conference attribute.

Retrieve the list of members currently attending the conference, call **cnfGetMemberList** using a valid conference identifier	Returns the ***memberidlist***.
	By performing a loop using the retrieved ***memberidlist***, you can get information about each member. For example:
	`for (memberindex = 0; memberindex < nummemberid; memberindex ++)`
	`{`
	`error = cnfGetMemberInfo(cnfresourcehd, memberidlist[memberindex],`
	`&memberinfo, sizeof(CNF_MEMBER_INFO));`
	`.`
	`.`
	`.`
	`}`

Setting member attributes

Information and attributes associated with members differ in the same way they do for conferences. Members are created with both parameters and attributes. While member parameters cannot change over the life of the conference, member attributes change frequently during the life of the conference.

You retrieve member parameters and other information together in a corresponding CNF_MEMBER_INFO structure, while you access attributes individually.

You can assign the following types of attributes to conference members:

- Automatic gain control
- Talking and listening control
- Talker privileges
- DTMF clamping
- Tone clamping
- Echo cancellation
- Self echo
- G.711 law

For a detailed description of the member attributes, refer to **cnfGetMemberAttribute**.

The following table lists the functions for accessing member information and attributes:

To...	Call this function...	Description
Retrieve member information	**cnfGetMemberInfo**	Returns a CNF_MEMBER_INFO structure containing stream and timeslot information for the given member.
Retrieve a member attribute	**cnfGetMemberAttribute**	Returns the value of the specified member attribute.
Set a member attribute	**cnfSetMemberAttribute**	Sets the value of the specified member attribute.
Retrieve several member attributes	**cnfGetMemberAttributeList**	Retrieves multiple member attributes with one function invocation.
Set several member attributes	**cnfSetMemberAttributeList**	Sets multiple member attributes with one function invocation.

Playing a tone

Call **cnfStartTone** to play a tone to all members of a conference to indicate that a new member is joining or leaving the conference. When **cnfStartTone** is invoked, members stop hearing each other and hear only the tone.

Tone attributes are stored in the CNF_TONE_PARMS structure. You can also set default parameters for the tone attributes.

The following table lists the functions to call to play a tone:

To...	Call this function...	Description
Play a tone	**cnfStartTone**	The program plays a sequence of user-defined tones. If the event_mask is set to CNF_EVNMSK_TONE_DONE before **cnfStartTone** is called, when the specified number of iterations (defined in CNF_TONE_PARMS) is complete, NaturalConference generates CNFEVN_TONE_DONE.
Abort tone generation or stop an infinite duration tone	**cnfStopTone**	The program immediately terminates active tone generation. If the event_mask is set to CNF_EVNMSK_TONE_DONE before **cnfStartTone** is called, the program generates CNFEVN_TONE_DONE.

NaturalConference events

NaturalConference uses the Natural Access event reporting mechanism, **ctaWaitEvent**. **ctaWaitEvent** returns the CTA_EVENT structure informing the application about which event occurred on which context. The structure includes information specific to the event.

Some events are generated only by setting the appropriate bit in the CNF.CONFERENCE_ATTR event_mask parameter or by modifying the CONF_ATTR_EVENT_MASK attribute using **cnfSetConferenceAttribute**.

NaturalConference generates the following events:

Event	Description
CNFEVN_ACTIVE_TALKERS_CHANGE	One or more of the active talkers changed.
CNFEVN_CLOSE_RESOURCE_DONE	***cnfresourcehd*** was released.
CNFEVN_OPEN_RESOURCE_DONE	***cnfresourcehd*** was allocated.
CNFEVN_TONE_DONE	Tone generation completed or was stopped.

When receiving CNFEVN_ACTIVE_TALKERS_CHANGE, you can retrieve the actual list of members that are active talkers by calling **cnfGetActiveTalkersList**. If the space available for retrieving the list of members is large enough, the number of ***memberid*** fields filled by the function and returned in ***nummemberids*** is the number of members speaking, as defined by the attribute CONF_ATTR_ACTIVE_TALKERS value currently set for that conference. The returned list contains the member that is the active talker when the function is called. The active talker lists are not queued in NaturalConference.

Closing a conference

To close a conference, follow these steps:

Step	Action
1	Remove all conference members by calling **cnfLeaveConference**.
2	Close the conference by calling **cnfCloseConference**.
3	Close the conference resource by calling **cnfCloseResource**.

Conference and call completion

Although a conference merges voices coming from different calls, there is no direct relationship between individual calls and conference members. The application features and scenarios drive the way calls and members are created and destroyed. A call can be added or removed from the same or different conferences without having to be released. When a call is disconnected, the application should force the corresponding members to leave the conference as soon as possible.

The following table lists the functions to call when closing a conference:

To...	Call this function...	Description
Remove a member from a conference	**cnfLeaveConference**	Removes the member from the conference and destroys the member identifier.
Close a conference	**cnfCloseConference**	Closes the conference and enables the system resources that were occupied by that conference to be used by other (new or existing) conferences located on the same resource. Closing a conference with members still joined does not have a negative impact on NaturalConference behavior. If a conference still contains members when it is closed, the members are forced to leave in an automatic internal process equivalent to invoking **cnfLeaveConference** on each member. After the conference is empty, it is destroyed.
Close a conference resource	**cnfCloseResource**	Closes the resource, enabling the system resources to be used by another application. When you close a resource, all conferences using that ***cnfresourcehd*** are closed by an automatic internal process equivalent to calling **cnfCloseConference** on each running conference. Conferences using a different ***cnfresourcehd*** are not affected. Wait for CNFEVN_CLOSE_RESOURCE_DONE before making any other calls to NaturalConference (especially important when using the Server mode of Natural Access). This event verifies that the resource was released.

Disconnecting the call

After a conference completes, the application must release the call. Call **nccDisconnectCall**, wait for the NCCEVN_CALL_DISCONNECTED event, then call **nccReleaseCall**.

Refer to the *Dialogic® NaturalAccess™ NaturalCallControl™ API Developer's Manual* for complete details on call control.

Closing Natural Access services

Natural Access supports opening and closing services on a context as needed during an application's execution. An application can free system resources it no longer needs to optimize performance.

Call **ctaCloseServices** to close the NaturalConference service. **ctaCloseServices** does not release the call, nor does it close the NCC service unless specified.

7 Function summary

Conference management functions

The following functions enable you to manage conferences:

Function	Synchronous/ Asynchronous	Description
cnfCreateConference	Synchronous	Creates a new conference on a conference resource.
cnfJoinConference	Synchronous	Adds a member to a conference.
cnfLeaveConference	Synchronous	Removes a member from a conference.
cnfCloseConference	Synchronous	Closes (destroys) a conference.
cnfGetConferenceList	Synchronous	Retrieves the list of existing conferences in a resource.
cnfGetConferenceInfo	Synchronous	Retrieves information about a conference such as the number of members allocated, the number of members attending, and the conference capabilities.
cnfGetActiveTalkersList	Synchronous	Retrieves the list of active talkers in a conference.
cnfGetMemberList	Synchronous	Retrieves the list of members in a conference.
cnfResizeConference	Synchronous	Changes the number of members allocated to a conference.
cnfGetConferenceAttribute	Synchronous	Retrieves a conference attribute.
cnfSetConferenceAttribute	Synchronous	Modifies a conference attribute.
cnfStartTone	Synchronous	Starts playing a tone to every member of the conference.
cnfStopTone	Synchronous	Stops playing the tone started by **cnfStartTone**.

Resource management functions

The following functions enable you to manage resources:

Function	Synchronous/ Asynchronous	Description
cnfOpenResource	Asynchronous	Opens a conference resource. Wait for CNFEVN_OPEN_RESOURCE_DONE before making any other calls to NaturalConference.
cnfGetResourceList	Synchronous	Retrieves the list of resources defined in the system.
cnfGetResourceInfo	Synchronous	Retrieves information about a conference resource such as the capabilities of the resource, the maximum number of members the resource can accept, and the number of members available for a new conference.
cnfCloseResource	Asynchronous	Closes a conference resource. Wait for CNFEVN_CLOSE_RESOURCE_DONE before making any other calls to NaturalConference.

Member management functions

The following functions enable you to manage members:

Function	Synchronous/ Asynchronous	Description
cnfGetCoaching	Synchronous	Retrieves a list of conference members who can or cannot hear a member.
cnfSetCoaching	Synchronous	Specifies the conference members who can and cannot hear a member.
cnfGetMemberAttribute	Synchronous	Retrieves a member attribute.
cnfSetMemberAttribute	Synchronous	Modifies a member attribute.
cnfGetMemberAttributeList	Synchronous	Retrieves the attributes of several members in one function call.
cnfGetMemberInfo	Synchronous	Returns stream and timeslot information for a conference member.
cnfSetMemberAttributeList	Synchronous	Modifies the attributes of several members in one function call.

8 Function reference

Using the function reference

This section provides an alphabetical reference to the NaturalConference functions. A typical function description includes:

Prototype	The prototype is shown followed by a list of the function arguments. NMS Communications data types include: • WORD 16-bit unsigned • DWORD 32-bit unsigned • INT16 16-bit signed • INT32 32-bit signed • BYTE 8-bit unsigned If a function argument is a data structure, the complete data structure is defined.
Return values	The return value for a function is either SUCCESS or an error code. For asynchronous functions, a return value of SUCCESS indicates the function was initiated; subsequent events indicate the status of the operation.
Example	Example functions that start with Demo are excerpts taken from demonstration function libraries shipped with the product. Example functions that start with my are excerpts taken from sample application programs shipped with the product. The notation `/* ... */` indicates additional code that is not shown.

cnfCloseConference

Closes a conference.

Prototype

DWORD **cnfCloseConference** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***confid***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
confid	Conference identifier returned by **cnfCreateConference**.

Return values

Return value	Description
SUCCESS	
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conferencing resource handle.
CNFERR_BOARD_TIMEOUT	Board timed out while waiting for a response message.
CNFERR_INVALID_IDENTIFIER	***confid*** is not a valid conference identifier.

Details

cnfCloseConference closes and deletes the conference specified by ***confid***. The conference resource being used is released and made available for creating a new conference or resizing an existing conference.

Refer to *Closing a conference* on page 59 for more information.

See also

cnfCreateConference

Example

```
extern CNFRESOURCEHD      cnfresourcehd;

DWORD myCloseConference( DWORD confid)
{
   DWORD error;

   error = cnfCloseConference (cnfresourcehd, confid);

   if (error != SUCCESS)
      {
      printf("Error when closing conference : %d", error);
      return(error);
      }
   return(SUCCESS);
}
```

cnfCloseResource

Closes a conference resource handle.

Prototype

DWORD **cnfCloseResource** (CNFRESOURCEHD ***cnfresourcehd***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.

Return values

Return value	Description
SUCCESS	
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_BOARD_TIMEOUT	Board timed out while waiting for a response message.

Events

Event	Description
CNFEVN_CLOSE_RESOURCE_DONE	Function completed successfully and the resource handle was released.

Details

cnfCloseResource closes the conference resource handle and destroys the ***cnfresourcehd***. All existing conferences created using this ***cnfresourcehd*** are closed and destroyed. Wait for CNFEVN_CLOSE_RESOURCE_DONE before making any other calls to NaturalConference (important when using the Server mode of Natural Access).

Refer to *Closing a conference* on page 59 for more information about closing a resource.

See also

cnfGetResourceList, **cnfOpenResource**

Example

```
extern CTAHD           ctahd;
extern CTAQUEUEHD      qhd;
extern CNFRESOURCEHD   cnfresourcehd;

DWORD myCloseResource()
{
       DWORD error;
       CTA_EVENT evt;
       char  textbuf[80] = "";

       error = cnfCloseResource(cnfresourcehd);

       if  (error != SUCCESS)
           {
           }
       do
       {
           error = ctaWaitEvent( qhd, &evt, CTA_WAIT_FOREVER );
           if(error != SUCCESS)
           {
            ctaGetText( ctahd, error, textbuf, sizeof( textbuf ));
             printf( "\rError when calling ctaWaitEvent => %s\n", textbuf);
            return(error);
           }
       } while ( evt.id != CNFEVN_CLOSE_RESOURCE_DONE );
```

cnfCreateConference

Creates a new conference.

Prototype

DWORD **cnfCreateConference** (CNFRESOURCEHD ***cnfresourcehd***, CNF_CONFERENCE_PARMS ***parms***, DWORD ***confid***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.

parms	Pointer to a CNF_CONFERENCE_PARMS structure:
	`typedef struct`
	`{`
	`  DWORD size;  /* Size of the structure */`
	`  DWORD flags;  /* Flags to describe capabilities not used */`
	`  DWORD allocated_members;  /* Number of members to allocate*/`
	`  DWORD user_value;  /*Application-provided value*/`
	`} CNF_CONFERENCE_PARMS;`
	Refer to *CNF.CONFERENCE* on page 142 for complete field descriptions.
	If ***parms*** is NULL, take the default parameters.

confid	Pointer for the new conference identifier.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***confid*** is a NULL pointer.
CTAERR_BOARD_ERROR	Conference cannot be created because of an HMIC limitation or an on-board memory limitation (only with AG 2000 boards).
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_BOARD_TIMEOUT	Board timed out while waiting for a response message.
CNFERR_INVALID_PARAMETER	***parms*** is inconsistent or out of range.
CNFERR_NOT_ENOUGH_RESOURCE_SPACE	Conference cannot be created because the resource is full. Either the number of reserved seats with the requested capabilities is exhausted, or the number of conference IDs for that resource is exhausted. There are only 255 conference IDs per conference resource.

Details

cnfCreateConference creates a new conference on the resource specified by ***cnfresourcehd***. If the function succeeds, the new conference identifier is returned in ***confid***.

Refer to *Creating conferences* on page 54 for more information.

See also

cnfGetConferenceInfo, **cnfGetResourceList**, **cnfResizeConference**

Example

```
extern CNFRESOURCEHD cnfresourcehd;

DWORD myCreateConference(unsigned allocated)
{
   DWORD confid;
   DWORD error;
   CNF_CONFERENCE_PARMS confparms;

   confparms.size = sizeof(CNF_CONFERENCE_PARMS);
   confparms.allocatedmembers = allocated;
   confparms.flags = CNF_NO_ECHO_CANCEL | CNF_NO_TONE_CLAMPING;
   confparms.user_value = NULL;
   error = cnfCreateConference (cnfresourcehd, &confparms,
                                &confid);
   if (error != SUCCESS)
      {
      printf("Error when creating conference : %d", error);
      return(error);
      }
   return(confid);
}
```

cnfGetActiveTalkersList

Returns the list of members actively talking in a conference.

Prototype

DWORD **cnfGetActiveTalkersList** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***confid***, unsigned ***maxmemberids***, DWORD ***memberidlist***, unsigned ***nummemberids***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
confid	Conference identifier returned by **cnfCreateConference**.
maxmemberids	Maximum number of entries in ***memberidlist*** array.
memberidlist	Pointer to an array of DWORD where the function returns the list of member identifiers. If NULL, this function just returns the number of members talking in a conference.
nummemberids	Pointer to a returned number of member identifiers.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***nummemberids*** or ***memberidlist*** pointer is NULL.
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_INVALID_IDENTIFIER	***confid*** is not a valid conference identifier.

Details

cnfGetActiveTalkersList is typically called after the application receives CNFEVN_ACTIVE_TALKERS_CHANGE. Using this function, the application can compare the actual count of members currently included in the conference with the list of the corresponding member identifiers. To change the number of members selected as active talkers, set the corresponding attribute. Refer to **cnfGetConferenceAttribute** and **cnfSetConferenceAttribute** for further details about conference related attributes.

The successive active talkers lists are not queued in NaturalConference. The application receiving the corresponding event must retrieve the list in the appropriate time delay.

For more information, refer to *NaturalConference events* on page 59.

See also

cnfCreateConference, **cnfGetMemberList**

Example

```
extern CNFRESOURCEHD cnfresourcehd;

unsigned getTalkerList(DWORD confid, DWORD *talkeridlist,
                       unsigned maxmemberids)
{
 DWORD error;
 unsigned numtalkerids = 0;

 error = cnfGetActiveTalkersList (cnfresourcehd, confid,
                                  maxmemberids, talkeridlist,
                                  &numtalkerids);
 if (error != SUCCESS)
{
printf("Error when retrieving active talker list :%d", error);
}
return (numtalkerids);
}
```

cnfGetCoaching

Retrieves a list of conference members who can or cannot hear member ***memberid***.

Prototype

DWORD **cnfGetCoaching** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***memberid***, BYTE ***flag***, unsigned ***maxlistenerids***, DWORD ****listeneridlist***, unsigned ***numlistenerids***)

Argument	Description
cnfresourcehd	Resource handle returned by **cnfOpenResource**.
memberid	Member identifier returned by **cnfJoinConference**.
flag	Value of either CNF_COACHING_HEAR or CNF_COACHING_SILENT.
maxlistenerids	Maximum number of entries in ***listeneridlist*** array.
listeneridlist	Depending on the flag, points to a list of members who can hear ***memberid*** (CNF_COACHING_HEAR) or to a list of members who cannot hear ***memberid*** (CNF_COACHING_SILENT).
numlistenerids	Number of listeners in the conference. Returned by the function.

Return values

Return value	Description
SUCCESS	
CTAERR_BOARD_ERROR	
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference handle.
CNFERR_CAPABILITY_NOT_AVAILABLE	Conference does not have coaching capabilities.
CNFERR_INVALID_IDENTIFIER	***memberid*** is not a valid member identifier.

Details

cnfGetCoaching retrieves the list of members who can or cannot (depending on the value of ***flag***) hear the member for which the function was called. This feature is typically used to enable a coach to listen to a conference and provide input to one or more colleagues without the knowledge of other conference members.

See also

cnfSetCoaching

Example

```
// Force the relationship for a member to SILENT
void SetSilent(CNFRESOURCEHD cnfresourcehd, DWORD basemember, DWORD member)
{
DWORD error;
unsigned numlistenerids;
DWORD*memberidlist = NULL;

// Retrieve the current list size
error = cnfGetCoaching(cnfresourcehd, basemember, CNF_COACHING_SILENT, 0, NULL, &numliste
nerids);
if (error != SUCCESS)
{
printf("Error %d when retrieving coaching on member %d", error , basemember);
}

// allocate buffer
memberidlist = malloc((numlistenerids + 1) * sizeof(unsigned));
if (memberidlist == NULL);
{
printf("Unable to allocate memory buffer");
}

// Retrieve the current list
error = cnfGetCoaching(cnfresourcehd, basemember, CNF_COACHING_SILENT, numlistenerids,
memberidlist, &numlistenerids);

memberidlist[numlistenerids] = member;
numlistenerids += 1;
error = cnfSetCoaching(cnfresourcehd, basemember, CNF_COACHING_SILENT, memberidlist,
numlistenerids);
if (error != SUCCESS)
{
printf("Error %d when setting coaching on member %d", error , basemember);
}
free(memberidlist);
}
```

cnfGetConferenceAttribute

Returns the current value of a conference attribute.

Prototype

DWORD **cnfGetConferenceAttribute** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***confid***, DWORD ***attribute***, INT32 ****value***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
confid	Conference identifier returned by **cnfCreateConference**.
attribute	Specifies the attribute to be retrieved. Refer to the Details section for a description of conference attributes.
value	Pointer to the returned attribute value.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***value*** is NULL.
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_INVALID_ATTRIBUTE	***attribute*** is not a valid attribute for a conference.
CNFERR_INVALID_IDENTIFIER	***confid*** is not a valid conference identifier.

Details

The following table describes the valid conference attributes:

Keyword	Allowed values	Description
CONF_ATTR_NUM_LOUDEST_SPEAKERS	1..***n*** when coaching is disabled ***1..8*** on AG boards when coaching is enabled ***1..16*** on CG boards when coaching is enabled	Number of members used for generating the conference output signal. In the interest of quality, the number of mixed member voices is minimized for output signal generation. An application can adjust this number according to its needs.
CONF_ATTR_EVENT_MASK	0 to disable all the events or CNF_EVNMSK_ACTIVE_TALKERS_CHANGE or CNF_EVNMSK_TONE_DONE	Bitmask describing the event the application is to receive for the conference.
CONF_ATTR_ACTIVE_TALKERS	1..***n***	Number of talking members factored in while generating CNFEVN_ACTIVE_TALKERS_CHANGE. This attribute enables an application to tune the way CNFEVN_ACTIVE_TALKERS_CHANGE is generated. For example, when set to one (1), NaturalConference generates an event only when the main talker is changing. If set to 6, NaturalConference generates an event when one of the 6 main talkers has changed. This has nothing to do with the selection of the loudest speaker except that if both attributes are set with the same value, the application considers the active talkers member list as the current member used for generating the conference's output signal.
CONF_ATTR_ACTIVE_TALKERS_TIMER	100 to 60000	Minimum time (in multiples of 20 ms) between the generation of two CNFEVN_ACTIVE_TALKERS_CHANGE events.

Refer to *CNF.CONFERENCE_ATTR* on page 142 for attribute default values.

See also

cnfCreateConference, **cnfGetConferenceInfo**, **cnfSetConferenceAttribute**

Example

```
extern CNFRESOURCEHD cnfresourcehd;

INT32 getActiveTalkersTimer(DWORD confid)
{
INT32 timer;
DWORD error;

error = cnfGetConferenceAttribute(cnfresourcehd, confid,
                                  CONF_ATTR_ACTIVE_TALKERS_TIMER,
                                  &timer);

return(timer);
}
```

cnfGetConferenceInfo

Returns information about a conference such as the number of members allocated, the number of members attending, and the conference capabilities.

Prototype

DWORD **cnfGetConferenceInfo** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***confid***, CNF_CONFERENCE_INFO ***confinfo***, unsigned ***size***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
confid	Conference identifier returned by **cnfCreateConference**.

confinfo	Pointer to a CNF_CONFERENCE_INFO structure:
	`typedef struct`
	`{`
	`  DWORD size;  /* size of the structure */`
	`  DWORD allocated_members;  /* number of members allocated */`
	`  /* for this conference */`
	`  DWORD joined_members;  /* number of members attending */`
	`  /* this conference */`
	`  DWORD flags;  /* capabilities not used flags */`
	`  DWORD capabilities;`
	`  DWORD user_value;`
	`} CNF_CONFERENCE_INFO;`
	Refer to the Details section for a description of these fields.

size	The size of the buffer pointed by ***confinfo***.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***confinfo*** pointer is NULL.
CTAERR_BAD_SIZE	***size*** is smaller than the size of CNF_CONFERENCE_INFO.
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_INVALID_IDENTIFIER	***confid*** is not a valid conference identifier.

Details

cnfGetConferenceInfo returns the information structure for the conference designated by ***confid***.

The CNF_CONFERENCE_INFO structure contains the following fields:

Field	Description
size	Number of bytes written to the buffer pointed to by ***confinfo***.
allocated_members	Number of members allocated on the conference resource for this conference.
joined_members	Number of members currently joined in this conference.
flags	Capabilities disabled at conference creation time.
capabilities	Capabilities enabled at conference creation time.
user_value	Application prompted value.

For more information, refer to *Setting conference attributes* on page 57.

See also

cnfCreateConference, **cnfGetConferenceList**, **cnfGetMemberInfo**, **cnfGetResourceInfo**

Example

```
extern CNFRESOURCEHD cnfresourcehd;

INT32 getNumberOfMember(DWORD confid)
{
   CNF_CONFERENCE_INFO confinfo;
   error = cnfGetConferenceInfo (cnfresourcehd, confid, &confinfo,
                                 sizeof(CNF_CONFERENCE_INFO));
   if (error != SUCCESS)
      {
      printf("Error when retrieving conference information :
                %d",error);
      }
   return(confinfo.joined_member);
}
```

cnfGetConferenceList

Returns the list of conferences running on a resource.

Prototype

DWORD **cnfGetConferenceList** (CNFRESOURCEHD ***cnfresourcehd***, unsigned ***maxconfids***, DWORD ****confidlist***, unsigned ****numconfids***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
maxconfids	Maximum number of entries in ***confidlist*** array.
confidlist	Pointer to an array of DWORD to receive the list of conference identifiers. If NULL, this function returns the number of conferences currently running in ***numconfids***.
numconfids	Pointer to the returned number of conference identifiers.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***numconfid*** pointer is NULL.
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.

Details

cnfGetConferenceList returns the actual list of conference identifiers running on a given resource. To determine the number of conferences currently running, set ***maxconfids*** to 0 and ***confidlist*** to NULL.

For more information, refer to *Creating conferences* on page 54.

See also

cnfCreateConference, **cnfGetConferenceInfo**, **cnfGetResourceInfo**, **cnfGetResourceList**

Example

```
extern CNFRESOURCEHD cnfresourcehd;

unsigned getConferenceList(DWORD *confidlist, unsigned maxconfids)
{
 DWORD error;
 unsigned numconfids = 0;

 error=cnfGetConferenceList(cnfresourcehd, maxconfids,
                            confidlist, &numconfids);
 if (error != SUCCESS)
{
printf("Error when retrieving conference list :%d", error);
}
return (numconfids);
}
```

cnfGetMemberAttribute

Returns the current value of a member attribute.

Prototype

DWORD **cnfGetMemberAttribute** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***memberid***, DWORD ***attribute***, INT32 ****value***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
memberid	Member identifier returned by **cnfJoinConference**.
attribute	Specifies the attribute to be retrieved. Refer to the Details section for a complete description of member attributes.
value	Pointer to the returned attribute value.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***value*** is NULL.
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_INVALID_ATTRIBUTE	***attribute*** is not a valid attribute for a member.
CNFERR_INVALID_IDENTIFIER	***memberid*** is not a valid member identifier.

Details

The following table lists the valid member attributes:

Keyword	Allowed values	Description
MEMBER_ATTR_INPUT_AGC_ENABLE	FALSE or TRUE	Input AGC state (enabled/disabled).
MEMBER_ATTR _OUTPUT_AGC_ENABLE	FALSE or TRUE	Output AGC state (enabled/disabled).
MEMBER_ATTR_INPUT_GAIN	-12 to 12	Value of the input gain currently applied.
MEMBER_ATTR_OUTPUT_GAIN	-12 to 12	Value of the output gain currently applied.
MEMBER_ATTR_INPUT_AGC_TARGETAMPL	-45 to 0	Target amplitude for AGC.
MEMBER_ATTR_INPUT_AGC_SILENCEAMPL	-45 to 0	Noise threshold amplitude for AGC.
MEMBER_ATTR_OUTPUT_AGC_TARGETAMPL	-45 to 0	Target amplitude for output AGC.
MEMBER_ATTR_OUTPUT_AGC_SILENCEAMPL	-45 to 0	Noise threshold amplitude for output AGC.

Keyword	Allowed values	Description
MEMBER_ATTR_TALKER_ENABLE	FALSE or TRUE	If the member is a talker in a conference.
MEMBER_ATTR_LISTENER_ENABLE	FALSE or TRUE	If the member can hear the others in the conference. If a member is neither talker nor listener, the member is considered as temporarily out of the conference even if the corresponding conferencing resource is still allocated for the member.
MEMBER_ATTR_TALKER_PRIVILEGE	FALSE or TRUE	If the member is considered a privileged talker. A privileged talker is always an active talker even when not speaking.
MEMBER_ATTR_DTMF_CLAMPING_ENABLE	FALSE or TRUE	If the DTMF clamping capability is currently activated for this member. **Note:** If DTMF clamping is disabled at the resource or conference level, this attribute is ignored.
MEMBER_ATTR_DTMF_CLAMPING_DELAYLINE	0 to 28 in ms	Delay line to buffer the signal received by each member before removing the DTMF with the DTMF clamping module. **Note:** If DTMF clamping is disabled, the delay line is not activated.
MEMBER_ATTR_TONE_CLAMPING_ENABLE	FALSE or TRUE	If the tone clamping capability is currently activated for this member. **Note:** If tone clamping is disabled at the resource or conference level, this attribute is ignored.
MEMBER_ATTR_EC_ENABLE	FALSE or TRUE	If the echo cancellation capability is currently activated for this member. **Note:** If echo cancellation is disabled at the resource or conference level, this attribute is ignored.
MEMBER_ATTR_EC_GAIN	-54 to +24 in db	Amplification applied to the echo input signal. Relevant only when echo cancellation is activated.
MEMBER_ATTR_EC_PREDELAY	0 to 9	Output delay. Relevant only when echo cancellation is activated.

Keyword	Allowed values	Description
MEMBER_ATTR_SELF_ECHO_ENABLE	FALSE or TRUE	If the voice received from this member on its input is mixed and returned to the member on its output. This attribute can be enabled when the output is not connected to the member, but is actually used for recording.
MEMBER_ATTR_INPUT_G711_LAW	CNF_G711_DEFAULT, CNF_G711_ALAW, or CNF_G711_MULAW	Current G.711 law applied in the member's input.
MEMBER_ATTR_OUTPUT_G711_LAW	CNF_G711_DEFAULT, CNF_G711_ALAW, or CNF_G711_MULAW	Current G.711 law applied in the member's output.

Refer to *CNF.MEMBER_ATTR* on page 143 for attribute default values.

See also

cnfGetMemberAttributeList, **cnfGetMemberInfo**, **cnfJoinConference**, **cnfSetMemberAttribute**

Example

```
extern CNFRESOURCEHD cnfresourcehd;
INT32 getMemberTalkerStatus(DWORD memberid)
{
DWORD error;
INT32 talkerActive;

error = cnfGetMemberAttribute(cnfresourcehd, memberid,
                              MEMBER_ATTR_TALKER_ENABLE,
                              &talkerActive);

return(talkerActive);
}
```

cnfGetMemberAttributeList

Returns the current value of several member attributes in a single function invocation.

Prototype

DWORD **cnfGetMemberAttributeList** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***memberid***, DWORD ***attributes***, INT32 ***values***, unsigned ***count***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
memberid	Member identifier returned by **cnfJoinConference** or **cnfGetMemberList**.
attributes	Pointer on the array of attributes to be retrieved.
values	Pointer on the returned attribute value array.
count	Number of entries in ***attributes*** and ***values*** array.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***attributes*** or ***values*** is NULL.
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_INVALID_ATTRIBUTE	One or more attributes in the array ***attributes*** are not valid attributes for a member.
CNFERR_INVALID_IDENTIFIER	***memberid*** is not a valid member identifier.

Details

Use **cnfGetMemberAttributeList** to pass an array of attributes instead of calling **cnfGetMemberAttribute** several times to get each attribute value separately.

Refer to **cnfGetMemberAttribute** and *Setting member attributes* on page 58 for more information.

Example

```
extern CNFRESOURCEHD cnfresourcehd;

typedef struct {
   INT32 enable;
   INT32 gain;
   INT32 predelay;
   } EC_CONTEXT;

DWORD getMemberECData(DWORD memberid, EC_CONTEXT *ec_context)
{
DWORD attributeList[3] = {
  MEMBER_ATTR_EC_ENABLE,
  MEMBER_ATTR_EC_GAIN,
  MEMBER_ATTR_EC_PREDELAY,
  };

INT32 values[3];
DWORD error;

error = cnfGetMemberAttibuteList( cnfresourcehd, memberid,
                                  attributeList, values, 3);

ec_context->enable = values[0];
ec_context->gain = values[1];
ec_context->predelay = values[2];
return(error);
}
```

cnfGetMemberInfo

Returns stream and timeslot information for a conference member.

Prototype

DWORD **cnfGetMemberInfo** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***memberid***, CNF_MEMBER_INFO ***memberinfo***, unsigned ***size***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
memberid	Member identifier returned by **cnfJoinConference** or **cnfGetMemberList**.

memberinfo	Pointer to a CNF_MEMBER_INFO structure:
	`typedef struct`
	`{`
	`  DWORD size;  /* size of the structure  */`
	`  DWORD stream;  /* MVIP address  */`
	`  DWORD timeslot;  /* MVIP address  */`
	`  DWORD user_value  /* application provided value*/`
	`} CNF_MEMBER_INFO;`
	Refer to the Details section for a description of these fields.

size	Maximum size of the buffer pointed by ***memberinfo.***

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***memberinfo*** pointer is NULL.
CTAERR_BAD_SIZE	***size*** is smaller than the size of CNF_MEMBER_INFO.
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_BOARD_TIMEOUT	The board timed out while waiting for a response message.
CNFERR_INVALID_IDENTIFIER	***memberid*** is not a valid member identifier.

Details

cnfGetMemberInfo returns the information structure corresponding to the member designated by ***memberid***.

The CNF_MEMBER_INFO structure contains the following fields:

Field	Description
size	Number of bytes written to the buffer pointed to by ***memberinfo***.
stream	The input MVIP-95 local stream (even value) to be used for connecting the member's line to the conference. To get the output stream, increment this stream by one.
timeslot	The MVIP timeslot to be used for connecting the member's line into the conference.
user_value	Application prompted value.

For more information, refer to *Setting member attributes* on page 58.

See also

cnfJoinConference

Example

```
extern CNFRESOURCEHD cnfresourcehd;

DWORD getMemberMVIPAddr(DWORD memberid, DWORD *stream,
                        DWORD *timeslot)
{
 CNF_MEMBER_INFO memberinfo;
 DWORD error;

 error = cnfGetMemberInfo (cnfresourcehd, memberid, &memberinfo,
                           sizeof(CNF_MEMBER_INFO));

*stream = memberinfo.stream;
*timeslot = memberinfo.timeslot;
return(error);
}
```

cnfGetMemberList

Returns the list of members currently attending a given conference.

Prototype

DWORD **cnfGetMemberList** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***confid***, unsigned ***maxmemberids***, DWORD ***memberidlist***, unsigned ***nummemberids***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
confid	Conference identifier returned by **cnfCreateConference**.
maxmemberids	Maximum number of entries in ***memberidlist*** array.
memberidlist	Pointer to an array of DWORD to receive the list of member identifiers. If NULL, this function returns the number of members attending the conference in ***nummemberids***.
nummemberids	Pointer to the returned number of member identifiers.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***nummemberids*** pointer is NULL.
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_INVALID_IDENTIFIER	***confid*** is not a valid conference identifier.

Details

cnfGetMemberList returns the actual list of members attending a given conference. To determine the current number of members without identifiers, set ***maxmemberids*** to 0 and ***memberidlist*** to NULL.

For more information, refer to *Creating conferences* on page 54.

See also

cnfGetMemberInfo, **cnfJoinConference**

Example

```
extern CNFRESOURCEHD cnfresourcehd;

unsigned getMemberList(DWORD confid, DWORD *memberidlist,
                       unsigned maxmemberids)
{
 DWORD error;
 unsigned nummemberids = 0;

 error = cnfGetMemberList(cnfresourcehd, confid, maxmemberids,
                          memberidlist, &nummemberids);
 if (error != SUCCESS)
{
printf("Error when retrieving member list :%d", error);
}
return (nummemberids);
}
```

cnfGetResourceInfo

Returns information for a conference resource such as the capabilities of the resource, the maximum number of members the resource can accept, and the number of members available for a new conference.

Prototype

DWORD **cnfGetResourceInfo** (CNFRESOURCEHD ***cnfresourcehd***, CNF_RESOURCE_INFO ***resourceinfo***, unsigned ***size***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.

resourceinfo	Pointer to a CNF_RESOURCE_INFO structure:
	`typedef struct`
	`{`
	`  DWORD size;  /* size of the structure */`
	`  DWORD board;  /* board number */`
	`  DWORD boardtype;  /* unused */`
	`  DWORD capabilities;  /* capabilities currently available */`
	`  DWORD conference;  /* number of conferences currently running*/`
	`  DWORD max_members;  /* maximum number of members */`
	`  DWORD available_members;  /* number of members currently available */`
	`} CNF_RESOURCE_INFO;`
	Refer to the Details section for a description of these fields.

size	The maximum size of the buffer pointed by ***resourceinfo***.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***resourceinfo*** pointer is NULL.
CTAERR_BAD_SIZE	***size*** is smaller than the size of DWORD.
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conferencing resource handle.

Details

cnfGetResourceInfo returns the information structure for the resource designated by ***cnfresourcehd***.

The CNF_RESOURCE_INFO structure contains the following fields:

Field	Description
size	Number of bytes written to the buffer pointed to by ***resourceinfo***.
board	Board number where the conference resource is located as declared in the board keyword file and in *cnf.cfg*.
boardtype	This field is longer used. It is kept for compatibility.
capabilities	Bitmask representing capabilities provided by the conferencing resource. The supported capabilities are defined in *cnfdef.h*. They are: CNF_RESCAP_AUTO_GAIN_CONTROL CNF_RESCAP_TALKER_PRIVILEGE CNF_RESCAP_ACTIVE_TALKER CNF_RESCAP_DTMF_CLAMPING CNF_RESCAP_TONE_CLAMPING CNF_RESCAP_ECHO_CANCELER CNF_RESCAP_EC_10_MS_WINDOW CNF_RESCAP_EC_20_MS_WINDOW CNF_RESCAP_EC_100_MS_CV CNF_RESCAP_EC_200_MS_CV
conference	Number of conferences currently running on the resource.
max_members	Maximum number of members the resource can manage if using the full resource capabilities.
available_members	Number of members available for creating a new conference using the full resource capabilities.

For more information, refer to *Creating conferences* on page 54.

See also

cnfCloseResource, **cnfGetConferenceList**, **cnfGetResourceList**, **cnfOpenResource**

Example

```
DWORD getResourceUsage(CNFRESOURCEHD cnfresourcehd,
                       INT32 *percentage)
{
   DWORD error;
   CNF_RESOURCE_INFO resourceinfo;

   error = cnfGetResourceInfo (cnfresourcehd, &resourceinfo,
                               sizeof(CNF_RESOURCE_INFO));

   *percentage = 100 - (resourceinfo.available_members /
                        resourceinfo.max_member * 100);
   return(error);
}
```

cnfGetResourceList

Returns the list of resources defined in the system.

Prototype

DWORD **cnfGetResourceList** (CTAHD ***ctahd***, unsigned ***maxresources***, unsigned ****resourcelist***, unsigned ****numresources***)

Argument	Description
ctahd	Natural Access handle returned by **ctaCreateContext**.
maxresources	Maximum number of entries in ***resourcelist*** array.
resourcelist	Pointer to an array of unsigned to receive the list of resource numbers. If NULL, this function returns the number of resources in ***numresources***.
numresources	Pointer to the returned number of resources defined in the system.

Return values

Return value	Description
SUCCESS	
CTAERR_INVALID_CTAHD	***ctahd*** is not a valid handle.

Details

cnfGetResourceList returns the actual list of resources defined in the system. To check for availability of a resource, open the resource. To determine the current number of resources without the actual number, set ***maxresources*** to 0 and ***resourcelist*** to NULL.

For more information, refer to *Creating conferences* on page 54.

See also

cnfCloseResource, **cnfOpenResource**

Example

```
unsigned printResourceList (CTAHD ctahd)
{
 DWORD error;
 unsigned resindex;
 unsigned numresources = 0;
 unsigned *resourcelist = NULL;

 // First call just retrieves the number of conferencing resource
 error= cnfGetResourceList(ctahd, 0, NULL, &numresources);
  if (error != SUCCESS)
  {
  printf("Error when retrieving resource list :%d", error);
  }

if (numresources)
{
  if (resourcelist = malloc(numresources * sizeof(unsigned)))
  {
      // This call retrieves the whole list of conferencing resource
      error= cnfGetResourceList(ctahd, numresources, resourcelist,
                                &numresources);
      if (error != SUCCESS)
      {
      printf("Error when retrieving resource list :%d", error);
      }

      for (resindex = 0; resindex < numresources; resindex++)
         printf("Conferencing resource %d exists",
                 resourcelist[resindex]);

      free(resourcelist);
  }
}
else
  printf("No Conferencing resource detected using ctahd %x",
          ctahd);

return (numresources);
}
```

cnfJoinConference

Adds a new member to a conference.

Prototype

DWORD **cnfJoinConference** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***confid***, CNF_MEMBER_PARMS ****parms***, DWORD ****memberid***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
confid	Conference identifier returned by **cnfCreateConference**.

parms	Pointer to a CNF_MEMBER_PARMS structure:
	`typedef struct`
	`{`
	`  DWORD size;  /* Size of the structure */`
	`  DWORD user_value  /* Application provided value */`
	`} CNF_MEMBER_PARMS;`
	If ***parms*** is NULL, take the default parameters.

memberid	Pointer to a returned member identifier.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***memberid*** is a NULL pointer.
CTAERR_BOARD_ERROR	New member cannot be added because of an HMIC limitation or on-board memory limitation (only with AG 2000 boards). This happens only when the member is added on-the-fly without being allocated.
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_BOARD_TIMEOUT	Board timed out while waiting for a response message.
CNFERR_INVALID_IDENTIFIER	***confid*** is not a valid conference identifier.
CNFERR_INVALID_PARAMETER	***parms*** is inconsistent or out of range.
CNFERR_NOT_ENOUGH_RESOURCE_SPACE	Member cannot be added because the conference resource is full. Either the number of reserved seats with the requested capabilities is exhausted, or the number of conference IDs for that resource is exhausted. There are only 255 conference IDs per conference resource.

Details

cnfJoinConference adds a new member to a conference specified by ***confid***. If the function succeeds, the new member identifier is returned in ***memberid***.

The default values of member attributes are taken from the corresponding Natural Access parameters. To enable the new member to talk, set the talker_enable parameter in the Natural Access parameter section of *cnf.cfg*, or set the attribute in the application by using **cnfSetMemberAttribute** or **cnfSetMemberAttributeList**.

Refer to *Adding members to a conference* on page 55 for more information.

See also

cnfGetMemberInfo, **cnfLeaveConference**

Example

```
extern CNFRESOURCEHD cnfresourcehd;

DWORD myJoinConference(DWORD confid, DWORD *memberid)
{
   DWORD error;
   CNF_MEMBER_INFO memberinfo;

   error = cnfJoinConference (cnfresourcehd, confid, NULL,
                              &memberid);
   if (error != SUCCESS)
      {
      printf("Error when joining conference : %d", error);
      return(error);
      }

   error = cnfGetMemberInfo (cnfresourcehd, *memberid,
                         &memberinfo, sizeof(CNF_MEMBER_INFO));

/* Perform switching operation by using memberinfo.stream and timeslot */

   error = cnfSetMemberAttribute (cnfresourcehd, *memberid,
                                  MEMBER_ATTR_TALKER_ENABLE, TRUE);
   return(error);
}
```

cnfLeaveConference

Removes a member from a conference and destroys the member identifier.

Prototype

DWORD **cnfLeaveConference** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***memberid***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
memberid	Member identifier returned by **cnfJoinConference** or **cnfGetMemberList**.

Return values

Return value	Description
SUCCESS	
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_BOARD_TIMEOUT	Board timed out while waiting for a response message.
CNFERR_INVALID_IDENTIFIER	***memberid*** is not a valid member identifier.

Details

cnfLeaveConference removes the member specified by ***memberid*** from ***cnfresourcehd***. If the remaining number of members in the conference is less than the allocated number, the seat resource freed by this member is available for that conference only. Otherwise, the corresponding resource is available for any conference on the resource. For more information on removing members, refer to *Closing a conference* on page 59.

See also

cnfJoinConference

Example

```
extern CNFRESOURCEHD cnfresourcehd;

DWORD leaveConference(DWORD memberid)
{
 DWORD error;
 unsigned numtalkerids = 0;

 error = cnfLeaveConference(cnfresourcehd, memberid);
 if (error != SUCCESS)
{
printf("Error when leaving the conference :%d", error);
}

return (error);
}
```

cnfOpenResource

Opens a conference resource and obtains a conference resource handle.

Prototype

DWORD **cnfOpenResource** (CTAHD ***ctahd***, unsigned ***resourcenum***, CNFRESOURCEHD ****cnfresourcehd***)

Argument	Description
ctahd	Context handle returned by **ctaCreatecontext**.
resourcenum	Conference resource number as declared in the NaturalConference configuration file (*cnf.cfg*).
cnfresourcehd	Pointer to a returned resource handle.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***cnfresourcehd*** is a NULL pointer.
CTAERR_BOARD_ERROR	Resource was defined in *cnf.cfg*, but the corresponding DSPs were not loaded by the configuration utility.
CTAERR_INVALID_CTAHD	***ctahd*** is not a valid handle.
CNFERR_BOARD_TIMEOUT	Board timed out while waiting for a response message.
CNFERR_CANNOT_CREATE_CHANNEL	No communication channels are available for use on the specified resource.
CNFERR_RESOURCE_NOT_AVAILABLE	Resource number designated by ***resourcenum*** does not exist or is not available.
CNFERR_RESOURCE_NOT_DEFINED	Resource is not defined in the NaturalConference configuration file (*cnf.cfg*).

Events

Event	Description
CNFEVN_OPEN_RESOURCE_DONE	Function completed successfully and a new resource handle was allocated.

Details

cnfOpenResource opens a new conference resource handle on the resource specified by ***resourcenum***. Many handles can be opened on the same conference resource. This handle is required for every other function in NaturalConference. Wait for CNFEVN_OPEN_RESOURCE_DONE before using the ***cnfresourcehd*** (important when using the Server mode of Natural Access).

For more information, refer to *Creating conferences* on page 54.

See also

cnfCloseResource, **cnfGetResourceInfo**

Example

```
extern CTAHD          ctahd;
extern CTAQUEUEHD     qhd;

DWORD myOpenResource(unsigned resnum)
{
      CNFRESOURCEHD newcnfresourcehd;
      DWORD error;
      CTA_EVENT evt;
      char  textbuf[80] = "";

          error = cnfOpenResource (ctahd,  resnum,  &newcnfresourcehd);
          if (error != SUCCESS)
          {
          }
          do
          {
              error = ctaWaitEvent( qhd, &evt, CTA_WAIT_FOREVER );
              if(error != SUCCESS)
              {
                  ctaGetText( ctahd, error, textbuf, sizeof( textbuf ) );
                  printf( "\rError when calling ctaWaitEvent => %s\n",
                      textbuf);
                  return(error);
              }
          } while ( evt.id != CNFEVN_OPEN_RESOURCE_DONE );
```

cnfResizeConference

Redefines the number of members allocated to a conference.

Prototype

DWORD **cnfResizeConference** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***confid***, DWORD ***mode***, unsigned ***number***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
confid	Conference identifier returned by **cnfCreateConference**.
mode	Specifies the resizing mode to be used. Refer to the Details section for a description of the allowed values for this parameter.
number	Number associated with the given resizing ***mode***.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***mode*** is not one of the possible values.
CTAERR_BOARD_ERROR	Conference cannot be resized because of an HMIC limitation or an on-board memory limitation (only with AG 2000 boards).
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_BOARD_TIMEOUT	Board timed out while waiting for a response message.
CNFERR_INVALID_IDENTIFIER	***confid*** is not a valid conference identifier.
CNFERR_NOT_ENOUGH_RESOURCE_SPACE	Conference cannot be resized because the resource is full. Either the number of reserved seats with the requested capabilities is exhausted, or the number of conference IDs for that resource is exhausted. There are only 255 conference IDs per conference resource.

Details

cnfResizeConference enables the application to change the number of members allocated on the resource for a conference. The resizing operation to perform is specified by ***mode***. This argument must be one of the following values (predefined in *cnfdef.h*):

Mode	Description
CNF_REDUCE_ALLOCATED	Reduces the number of allocated members by ***number***.
CNF_INCREASE_ALLOCATED	Increases the number of allocated members by ***number***.
CNF_RESIZE_ALLOCATED	Resizes the number of allocated members to ***number***.

Refer to *Creating conferences* on page 54 for more information.

See also

cnfCreateConference, **cnfGetConferenceInfo**

Example

```
DWORD error;

/* Add space for a new allocated member */
error = cnfResizeConference (cnfresourcehd, confid,
                             CNF_INCREASE_ALLOCATED, 1);
if (error != SUCCESS)
   {
   printf("Error when resizing conference : %d", error);
   return(error);
   }
```

cnfSetCoaching

Specifies the conference members who can and cannot hear member ***memberid***.

Prototype

DWORD **cnfSetCoaching** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***memberid***, BYTE ***flag***, DWORD ****listeneridlist,*** unsigned ***numlistenerids***)

Argument	Description
cnfresourcehd	Resource handle returned by **cnfOpenResource**.
memberid	Member identifier returned by **cnfJoinConference**.
flag	Value of either CNF_COACHING_HEAR or CNF_COACHING_SILENT.
listeneridlist	Depending on the flag, points to a list of members who can hear ***memberid*** (CNF_COACHING_HEAR) or to a list of members who cannot hear ***memberid*** (CNF_COACHING_SILENT).
numlistenerids	Number of entries in ***listeneridlist***.

Return values

Return value	Description
SUCCESS	
CTAERR_BOARD_ERROR	
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference handle.
CNFERR_CAPABILITY_NOT_AVAILABLE	The conference does not have coaching capabilities.
CNFERR_INVALID_IDENTIFIER	***memberid*** or one of the ***memberid*** in the list is not a valid member identifier or does not belong to the same conference.

Details

Use **cnfSetCoaching** to specify the members who can and cannot hear (depending on the flag) the member for which this function was called. This feature is typically used to enable a coach to listen to a conference and provide input to one or more colleagues without the knowledge of other conference members.

Note: When calling **cnfSetCoaching**, the flag for each member not listed in the ***listeneridlist*** array is set to the opposite of the value of flag (CNF_COACHING_SILENT / CNF_COACHING_HEAR) passed to **cnfSetCoaching** . This is particularly important to be aware of when calling **cnfSetCoaching** for newly added members where previous coaching relationships have already been defined.

If the flag is...	Then...
CNF_COACHING_HEAR	Members in the list can hear ***memberid***.
CNF_COACHING_SILENT	Members in the list cannot hear ***memberid***. Members not in the list hear ***memberid*** by default. If members are using a specific member attribute or calling this function, they can hear the other members.

See also

cnfGetCoaching

Example

```
// Create a sub conference between 2 members

void createSubConference(CNFRESOURCEHD cnfresourcehd, DWORD member1, DWORD member2)
{
DWORD error;

// Only member2 hears member1
error = cnfSetCoaching(cnfresourcehd, member1, CNF_COACHING_HEAR , &member2, 1);
if (error != SUCCESS)
{
printf("Error %d when setting coaching on member %d", error , member1);
}

// Only member1 hears member2
error = cnfSetCoaching(cnfresourcehd, member2, CNF_COACHING_HEAR , &member1, 1);
if (error != SUCCESS)
{
printf("Error %d when setting coaching on member %d", error , member2);
}

}
```

cnfSetConferenceAttribute

Sets the value of a conference attribute.

Prototype

DWORD **cnfSetConferenceAttribute** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***confid***, DWORD ***attribute***, INT32 ***value***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
confid	Conference identifier returned by **cnfCreateConference**.
attribute	Specifies the attribute to set. Refer to **cnfGetConferenceAttribute** for a complete list and description of the conference attributes.
value	New value for the attribute.

Return values

Return value	Description
SUCCESS	
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conferencing resource handle.
CNFERR_BOARD_TIMEOUT	Board timed out while waiting for a response message.
CNFERR_INVALID_ATTRIBUTE	***attribute*** is not a valid attribute for a conference.
CNFERR_INVALID_IDENTIFIER	***confid*** is not a valid conference identifier.
CNFERR_INVALID_PARAMETER	***value*** is inconsistent or out of range for the given attribute.

Details

Refer to **cnfGetConferenceAttribute** for a complete description of the conference attributes. Refer to *Setting conference attributes* on page 57 for information on setting conference attributes.

Refer to *CNF.CONFERENCE_ATTR* on page 142 for information about attribute default values.

See also

cnfCreateConference, **cnfGetConferenceInfo**

Example

```
// Setup 1 second between each active talker change event
error = cnfSetConferenceAttribute ( cnfresourcehd, confid,
                                    CONF_ATTR_ACTIVE_TALKERS_TIMER,
                                    1000);
```

cnfSetMemberAttribute

Sets the value of a member attribute.

Prototype

DWORD **cnfSetMemberAttribute** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***memberid***, DWORD ***attribute***, INT32 ***value***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
memberid	Member identifier returned by **cnfJoinConference**.
attribute	Specifies the attribute to set. Refer to **cnfGetMemberAttribute** for a complete list of the member attributes.
value	New value for the attribute.

Return values

Return value	Description
SUCCESS	
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_BOARD_TIMEOUT	Board timed out while waiting for a response message.
CNFERR_INVALID_ATTRIBUTE	***attribute*** is not a valid attribute for a member.
CNFERR_INVALID_IDENTIFIER	***memberid*** is not a valid member identifier.
CNFERR_INVALID_PARAMETER	***value*** is inconsistent or out of range for the given attribute.

Details

Refer to *CNF.MEMBER_ATTR* on page 143 for information about attribute default values.

See also

cnfGetMemberInfo, **cnfJoinConference**, **cnfSetMemberAttributeList**

Example

```
error = cnfSetMemberAttribute( cnfresourcehd, memberid,
                               MEMBER_ATTR_TALKER_ENABLE, TRUE);
```

cnfSetMemberAttributeList

Sets the value of several member attributes with a single function invocation.

Prototype

DWORD **cnfSetMemberAttributeList** (CNFRESOURCEHD ***cnfresourcehd***, DWORD ***memberid***, DWORD ****attributes***, INT32 ****values***, unsigned ***count***)

Argument	Description
cnfresourcehd	Handle returned by **cnfOpenResource**.
memberid	Member identifier returned by **cnfJoinConference** or **cnfGetMemberList**.
attributes	Pointer to the array of attributes to set.
values	Pointer to the values corresponding to the ***attributes***.
count	Number of entries in ***attributes*** and ***values***.

Return values

Return value	Description
SUCCESS	
CTAERR_BAD_ARGUMENT	***attributes*** or ***values*** is NULL.
CTAERR_INVALID_HANDLE	***cnfresourcehd*** is not a valid conference resource handle.
CNFERR_BOARD_TIMEOUT	Board timed out while waiting for a response message.
CNFERR_INVALID_ATTRIBUTE	One or more attributes in the array ***attributes*** is not a valid attribute for a member.
CNFERR_INVALID_PARAMETER	***value*** is inconsistent or out of range for the given attribute.

Details

Use **cnfSetMemberAttributeList** to pass an array of attributes instead of calling **cnfSetMemberAttribute** several times to set each attribute value separately.

Refer to **cnfSetMemberAttribute** and *Setting member attributes* on page 58 for information about member attributes.

Example

```
DWORD attributeList[3] = {
      MEMBER_ATTR_EC_ENABLE,
      MEMBER_ATTR_EC_GAIN,
      MEMBER_ATTR_EC_PREDELAY,
      };

INT32value[3] = { TRUE, 0, 5};

error = cnfSetMemberAttributeList( cnfresourcehd, memberid,
                                   attributeList, values, 3);
```

cnfStartTone

Plays a tone to all members of the conference.

Prototype

DWORD **cnfStartTone** (CNFRESOURCEHD ***reshd***, DWORD ***confid***, CNF_TONE_PARMS ****parms***)

Argument	Description
reshd	Resource handle returned by **cnfOpenResource**.
confid	Conference identifier returned by **cnfCreateConference**.

parms	Pointer to a CNF_TONE_PARMS structure:
	`typedef struct`
	`{`
	`  DWORD size ;  /* size of this structure */`
	`  DWORD freq1;  /* first frequency (Hz) */`
	`  INT32 ampl1;  /* level of first tone (dBm) */`
	`  DWORD freq2;  /* second frequency (Hz) */`
	`  INT32 ampl2;  /* level of second tone (dBm) */`
	`  DWORD ontime;  /* on duration of DTMF tone (ms) */`
	`  DWORD offtime;  /* off duration of DTMF tone (ms) */`
	`  INT32 iterations;/* times to repeat above; */`
	`  /* -1 = forever last offtime is */`
	`  /* trimmed if repeat>1 */`
	`} CNF_TONE_PARMS;`
	Refer to *CNF.TONE* on page 144 for a description of these fields.

Return values

Return value	Description
SUCCESS	
CTAERR_BOARD_ERROR	Tone cannot be generated because the board stopped responding.
CTAERR_INVALID_HANDLE	***reshd*** is not a valid conference handle.
CNFERR_CONFERENCE_EMPTY	Conference ***confid*** currently has no members.
CNFERR_INVALID_IDENTIFIER	***confid*** is not a valid conference identifier.
CNFERR_INVALID_PARAMETER	***parms*** is inconsistent or out of range.

Events

Event	Description
CNFEVN_TONE_DONE	Tone generation completed or was stopped. Generated only if event_mask is set to CNF_EVNMSK_TONE_DONE when the function is called and the function returns SUCCESS.

Details

cnfStartTone plays a tone to all conference members to indicate that a new member is joining or leaving the conference. When this function is invoked, members stops hearing each other and hear only the tone.

If the ***iterations*** count is one (1), the tone is not complete until the ***offtime*** has expired. If the ***iterations*** count is more than one, then the final ***offtime*** is omitted.

To generate a tone continuously (forever), set ***iterations*** to -1 and specify an ***offtime*** of 0 (zero). Use **cnfStopTone** to terminate tone generation prematurely.

For more information, refer to *Playing a tone* on page 58.

Example

```
extern CNFRESOURCEHD cnfresourcehd;

int myConferenceTone( DWORD confid )
{
 CTA_EVENT event;
 CNF_TONE_PARMS toneparms;

  // Fill the tone structure
  toneparms.size = sizeof(CNF_TONE_PARMS);
  toneparms.freq1 = 1000;
  toneparms.ampl1 = -20;
  toneparms.freq2 = 500;
  toneparms.ampl2 = -20;
  toneparms.ontime = 200;
  toneparms.offtime = 200;
  toneparms.iterations = 2;

  cnfStartTone ( cnfresourcehd, confid, &toneparms);

    do
    {
        myGetEvent( &event );        /* see ctaWaitEvent example */
    } while( event.id != CNFEVN_TONE_DONE );

    if( CTA_IS_ERROR( event.value ) )
        return MYFAILURE;            /* API error               */
    else
        return SUCCESS;              /* started successfully    */
}
```

cnfStopTone

Stops the tone generation started by **cnfStartTone**.

Prototype

DWORD **cnfStopTone** (CNFRESOURCEHD ***reshd***, DWORD ***confid***)

Argument	Description
reshd	Resource handle returned by **cnfOpenResource**.
confid	Conference identifier returned by **cnfCreateConference**.

Return values

Return value	Description
SUCCESS	
CTAERR_BOARD_ERROR	Tone cannot be stopped because the board stopped responding.
CTAERR_INVALID_HANDLE	***reshd*** is not a valid conference handle.
CNFERR_CONFERENCE_EMPTY	Conference ***confid*** currently has no members.
CNFERR_INVALID_IDENTIFIER	***confid*** is not a valid conference identifier.
CNFERR_INVALID_PARAMETER	***parms*** is inconsistent or out of range.

Events

Event	Description
CNFEVN_TONE_DONE	Tone generation completed or was stopped. Generated only if event_mask is set to CNF_EVNMSK_TONE_DONE when **cnfStartTone** is called, and if **cnfStartTone** returns SUCCESS.

Details

cnfStopTone terminates the tone generation started by **cnfStartTone**. If no tone was being played, the function returns SUCCESS.

For more information, refer to *Playing a tone* on page 58.

Example

```
extern CNFRESOURCEHD cnfresourcehd;

int stopConferenceTone( DWORD confid )
{
 CTA_EVENT event;

    if( cnfStopTone( cnfresourcehd,  confid) != SUCCESS )
        return MYFAILURE;

    do
    {
        myGetEvent( &event );        /* based on ctaWaitEvent   */

  } while( event.id != CNFEVN_TONE_DONE );

  if( CTA_IS_ERROR( event.value ) )
        return MYFAILURE;            /* API error                */
    else
        return SUCCESS;              /* started successfully     */

}
```

9 cnfjoin demonstration program

cnfjoin overview

cnfjoin demonstrates adding callers to a conference. *cnfjoin* opens a set of available conferencing resources and a designated number of channels for call control. When *cnfjoin* receives a call, it adds the caller to a conference. All callers are added to the same conference. When a caller disconnects, the program removes the caller from the conference, allowing another caller to take its place.

cnfjoin features NaturalConference functions, Voice Message (VCE) service functions, and Natural Call Control (NCC) service functions. *cnfjoin* can be used with ISDN, CAS, or analog protocols.

System requirements

cnfjoin can be used on AG boards and CG boards. To use *cnfjoin*, call control and all conferencing resources must reside on the same board. If you are using the default configuration files provided by the installation, your system is properly configured to use *cnfjoin*.

Using cnfjoin

To use *cnfjoin*, enter the following command:

```
cnfjoin [options]
```

where ***options*** are:

Option	Description	Default
-h	Displays Help	None.
-F ***filename***	Natural Access configuration file name	*nms\ctaccess\cfg\cta.cfg*
-b ***n***	Board number for the line interface and for NaturalConference	0
-p ***protocol***	Protocol to run	lps0 for AG Series boards isd0 for CG Series boards
-n ***count***	Number of voice channels to open	0
-r ***resource***	Resource number to open	The first found.
-m ***count***	Number of conferences to open. See *Chaining conferences* on page 109.	1
-v ***count***	Number of virtual members to add. See *Using virtual members* on page 108.	0
-l ***level***	Reports a low level event	0
-t ***tracemask***	Sets the Natural Access tracemask	0x0

Using virtual members

Use virtual members to play and record *pcm* files on a conference. Virtual members are connected to an ADI channel full duplex. Virtual members have no trunk connection.

Note: *pcm* files contain voice sampled at 8 kHz with 16 bit format (128 kbyte/s). You can create, edit, and play *pcm* files with the Cool Edit utility distributed with Natural Access. *pcm* files accessed by *cnfjoin* must be in the working directory.

Use the following menu options for virtual members:

Use this option...	To...
pf ***n***	Play file ***n***.*pcm* on channel ***n*** (-1 for all).
rf ***n***	Record file ***n***.*pcm* on channel ***n*** (-1 for all).
ps ***n***	Stop playing the file on channel ***n*** (-1 for all).
rs ***n***	Stop recording the file on channel ***n*** (-1 for all).
s	Display the numbers of actual and virtual members connected to the conference.
h	Access Help.
q	Quit *cnfjoin*.

Chaining conferences

Conferences are opened on separate resources. Each conference can host the maximum number of members available on a resource. When the m value is 1 (default value), only one conference is opened. If the m value is greater than 1, a master conference is opened on the first resource, and ***count*** number of child conferences are opened on the following ***count*** resources. Each child conference hosts a subset of the actual conferees plus a special member that is connected to the master conference. The master conference mixes the ***count*** special members and redistributes the mixing output to all the child conferences.

Note: Make sure that more than ***count*** + 1 resources are available on the same board to run this configuration.

cnfjoin example

The following example sets up one conferencing resource. Up to 32 members can dial into the T1 link and join the conference. This example assumes you are using a CG6060 board connected to one T1 interface.

Step	Action
1	Configure the system using *c6060cnf_ivr_nonspan.cfg* and *cnf_c6060cnf_ivr_nonspan.cfg*.
3	Start *cnfjoin* by entering the following command at the prompt: `cnfjoin -n 32`

10 JCnfDemo demonstration program

JCnfDemo overview

The *JCnfDemo* demonstration program is the multi-platform Java version that demonstrates building conferences and connecting users to conferences through a graphical user interface.

JCnfDemo includes a small module for receiving calls, prompting a vocal welcome message, and managing commands from the user (through DTMF).

JCnfDemo uses Natural Call Control (NCC) for the trunk interface, Switching (SWI) to connect a caller to a channel, and NaturalConference (CNF) for the conferencing features.

For each Natural Access service that is used, *JCnfDemo* uses the *jninmss* package that wraps the Natural Access API.

System requirements

JCnfDemo can be used on AG and CG boards and is available under the following operating systems:

Operating system	Java version
Windows	Java version 1.3.0_01 or higher
UNIX	Java 2 Runtime Environment

The Java 2 Runtime Environment can be found at *http://www.sun.com/j2se/1.3*.

The following board configurations support *JCnfDemo*:

- Two separate boards connected through an H.100 link or an MVIP link. The first board could be a trunk interface board using ADIMGR for call control. The second board could be a dedicated board for conferencing. The trunk interface board must support NCC.
- One board for both conferencing and trunk resources.

Installed files

The *JCnfDemo* program is installed in the *nms\cnf\javademo* directory. The following tables list the *JCnfDemo* components:

Component	Description
JCnfDemo.jar	Contains all of the graphics and *Jni* class files to run the *JCnfDemo*.
A configuration file: *JcnfDemo.ini* under Windows *.jcnfdemorc* under UNIX	This configuration file is located in the *home/nmss_demo_java/* directory and will be created at the first use of *JCnfDemo* if it does not already exist.
images	Directory containing all of the graphic files used by *JCnfDemo* (logo, tree icons, check boxes, member representation). This directory is placed in the current directory.
Prompt	Directory containing the *welcome.pcm* sound file.
src/cnfdemo	Directory containing the Java source for the graphical user interface.
src/jninmss	Directory containing the Java source for the Natural Access interface.

The following library is also installed:

Windows	UNIX	Description
jninmss.dll	*libjninmss.so*	Dynamic library used to interface the Natural Access services through the Java Native Interface (JNI).

Before running JCnfDemo

Before running *JCnfDemo*, ensure that:

- Natural Access is properly installed, as described in the Natural Access installation booklet.
- NaturalConference is properly configured, as described in the *Overview of configuring NaturalConference* on page 21.
- The board is properly configured, as described in the board installation manual.
- Java 2 Runtime 1.3.0_01 higher is properly installed.

Complete the following steps to verify the Java runtime installation:

Step	Action
1	Enter the following command to verify that your installed Java runtime engine is compatible with *JCnfDemo*: `java -version`

2	Under Windows, verify that the PATH environment contains the directory where the *JCnfDemo* library file (*jninmss.dll*) is installed.
	Under UNIX, verify that the LD_LIBRARY_PATH contains the directory that contains the *JCnfDemo* library file (*libjninmss.so*).
	For example, if the installed directory of the *JCnfDemo* program is *nms\cnf\javademo,* add *nms\cnf\javademo* to your PATH or LD_LIBRARY_PATH variable:
	`set PATH=%PATH%;\nms\cnf\javademo (under Windows)`
	`export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/nms/cnf/javademo`
	`(under UNIX with korn shell)`
	`setenv LD_LIBRARY_PATH"$LD_LIBRARY_PATH:/opt/nms/cnf/javademo"`
	`(under UNIX with C shell)`
	Note: The directory *home\nmss_demo_java* contains the *\images* directory and the *welcome.pcm* file. Under Windows, *home* is *c:\documents* and *c:\settings\user*.

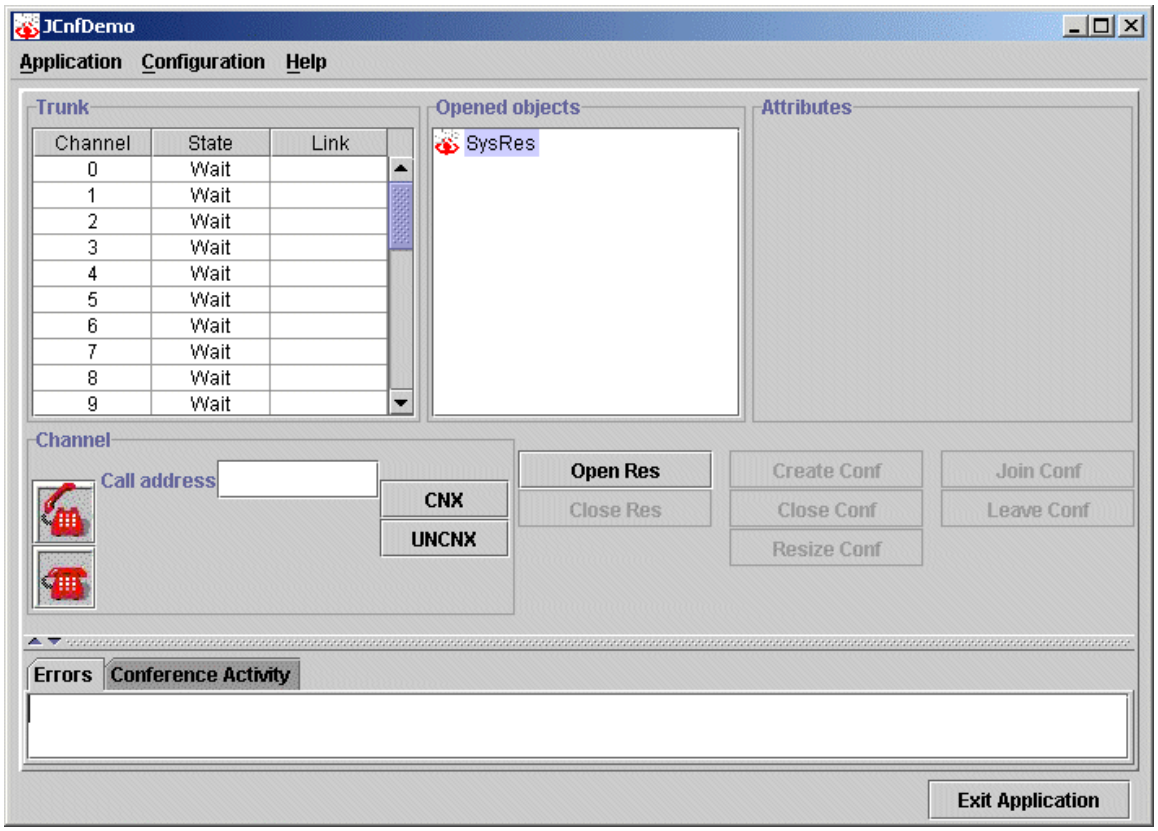
Starting and using JCnfDemo

To start *JCnfDemo*, enter the following command:

```
java -jar JCnfDemo.jar
```

The configuration file (*JCnfDemo.ini* under Windows, *.jcnfdemorc* under UNIX) is placed in the home directory (*c:\documents* and *c:\settings\user* under Windows, *~home* under UNIX).

The following image shows the *JCnfDemo* main window:



The *JCnfDemo* main window contains four main sections:

Use this section...	To...
Trunk	Manage calls.
Opened Objects	View the open resources, the conferences running on each resource, and the members attending each conference.
Attributes	View and change the capabilities of the selected conference or member. View information for the selected resource, conference, or member.
Errors/Conference Activity	View error events and the activity in a selected conference. View and set the coaching.

The following table describes the buttons on the *JCnfDemo* window:

Use this button...	To...
Exit Application	Close *JCnfDemo*.
Open Res	Open a resource.
Close Res	Close the selected resource.
Create Conf	Create a conference on the selected resource.
Close Conf	Close the selected conference.
Resize Conf	Adjust the number of members allocated to the selected conference.
Join Conf	Add members to the selected conference.
Leave Conf	Remove members from the selected conference.
CNX	Connect a channel and a member.
UNCNX	Disconnect a channel and a member.
Phone (calling)	Perform a call from a channel.
Phone (released)	Release a call from a channel.

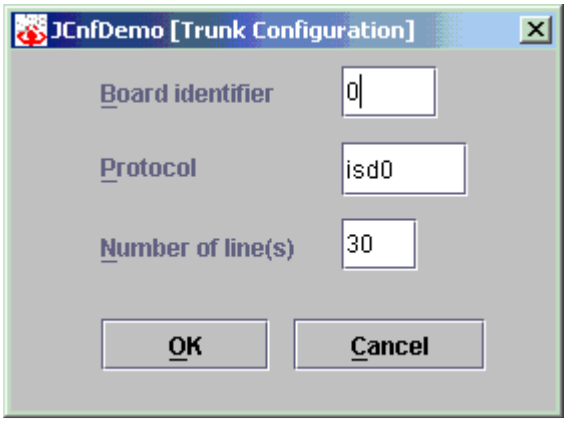
The following table describes the menu selections available on the *JCnfDemo* window:

Use this menu item...	To...
Configuration > Board Configuration	Open the Trunk Configuration dialog box to view or modify your current configuration.
Application > Exit	Exit *JCnfDemo*.
Help > About	View the current version.

Configuring JCnfDemo

The first time you use *JCnfDemo*, configure the trunk. The parameters you enter are saved in the *JCnfDemo* configuration file (*JCnfDemo.ini* or *.jcnfdemorc*).

To configure *JCnfDemo*:

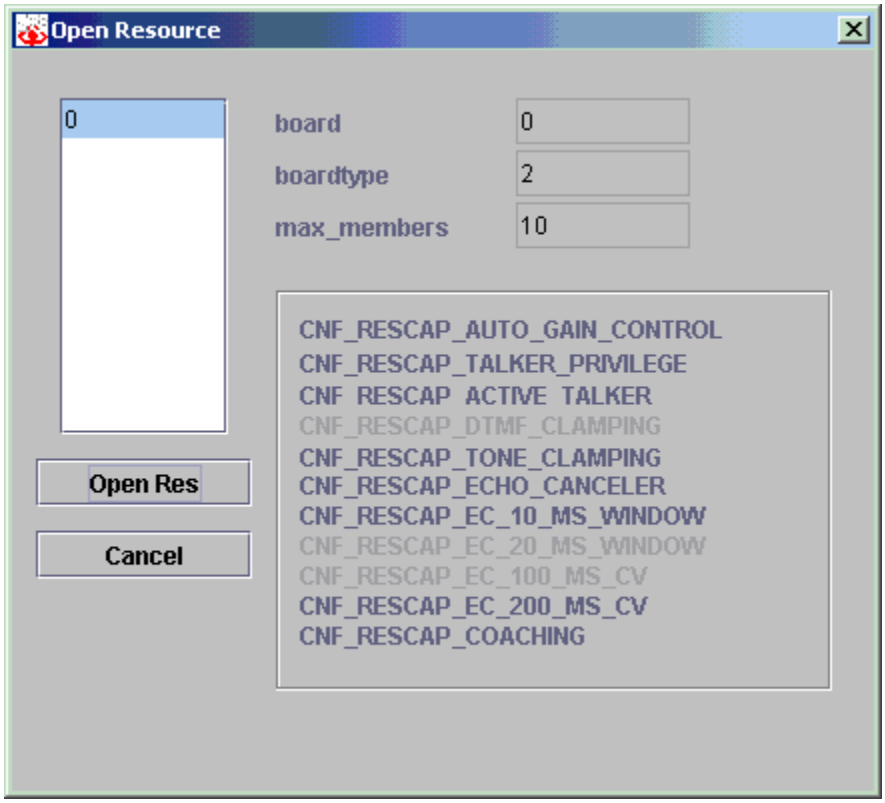
<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Click Configuration > Board Configuration. The JCnfDemo [Trunk Configuration] dialog box displays:
 
</td></tr>
<tr><td>2</td><td>Complete the fields as described in the following table:
<table>
<tr><th>Field</th><th>Enter the...</th></tr>
<tr><td>Board identifier</td><td>Trunk board identifier as defined in the board keyword file.</td></tr>
<tr><td>Protocol</td><td>Line protocol.</td></tr>
<tr><td>Number of line(s)</td><td>Number of lines to start on the trunk interface.</td></tr>
</table></td></tr>
<tr><td>3</td><td>Click OK. The configuration file is updated.</td></tr>
<tr><td>4</td><td>Close and restart JCnfDemo to apply the changes.</td></tr>
</table>

Opening resources

Perform the following steps to open a resource:

1. On the JCnfDemo window, click **Open Resource**.

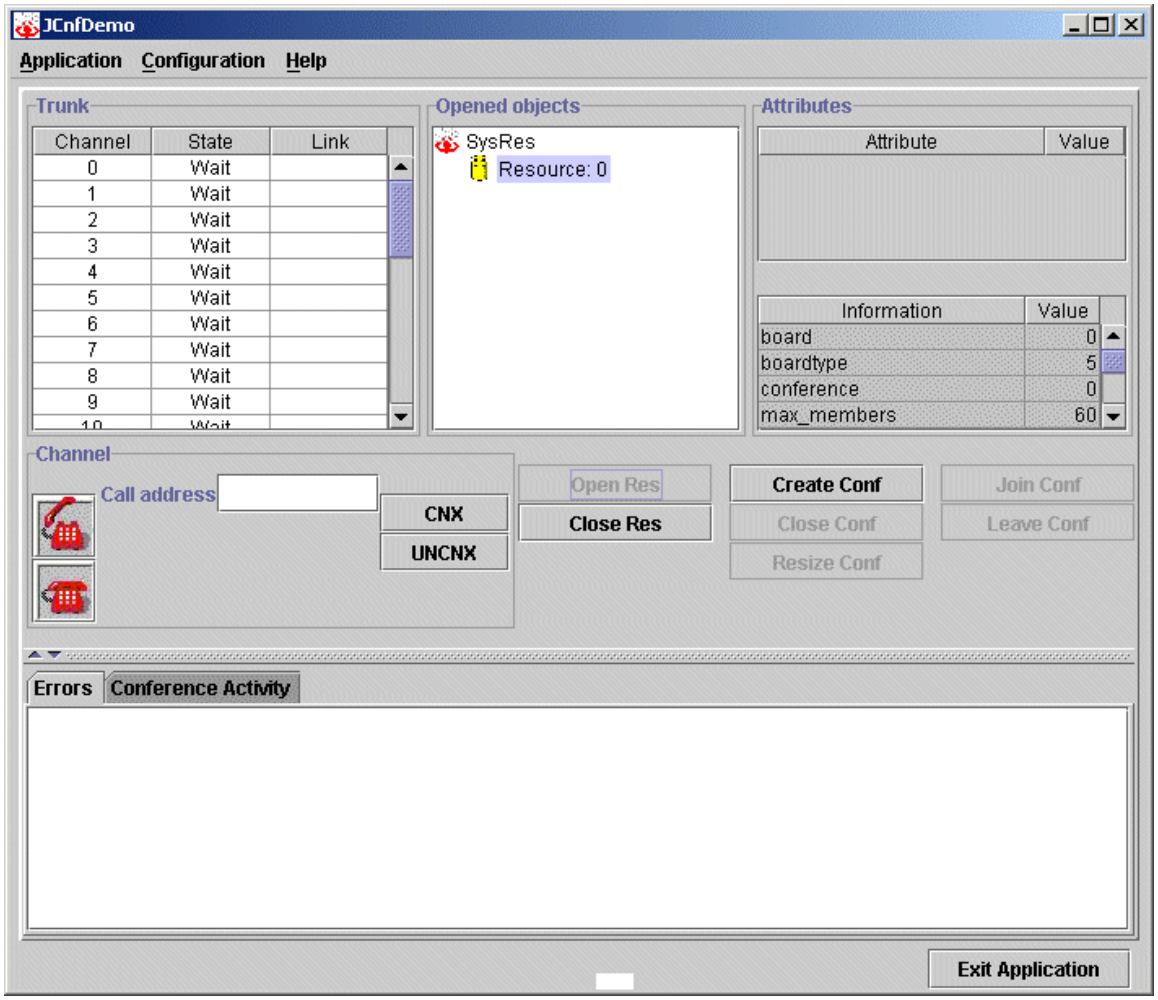
   The Open Resource dialog box displays, listing the conference resources configured in the system:



2. Select a resource from the Resources box. The following information about the resource displays:
   - Board number from the board keyword file
   - Board type from the NaturalConference configuration file (*cnf.cfg*)
   - Maximum number of members the resource supports
   - Capabilities the resource supports

Note: If the resource is already open, only the resource number is listed in the Resources box. The related fields are not filled in.

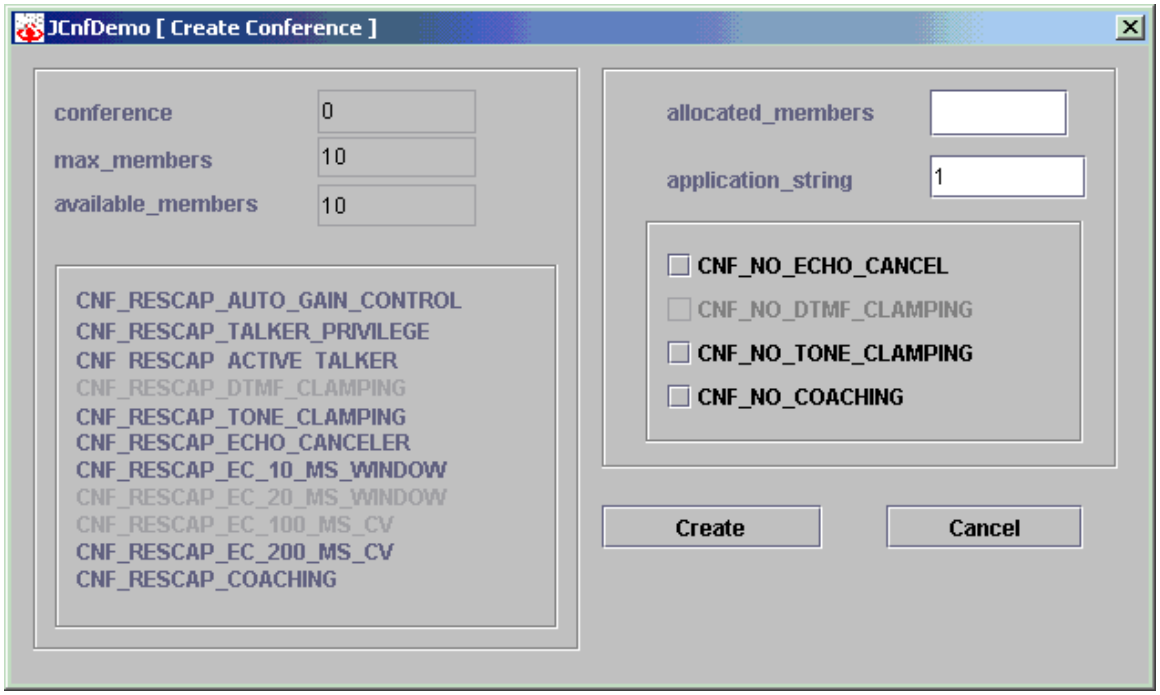
3. Click **Open** to open the resource. The resource opens and displays in the Opened Objects list on the JCnfDemo window. The resource information displays in the Information box.



Creating conferences using JCnfDemo

Perform the following steps to create a conference:

1. Complete the steps for opening a resource.
2. On the *JCnfDemo* window, select the resource from the Opened Objects list.
3. Click **Create Conf**. The Create Conference dialog box displays:

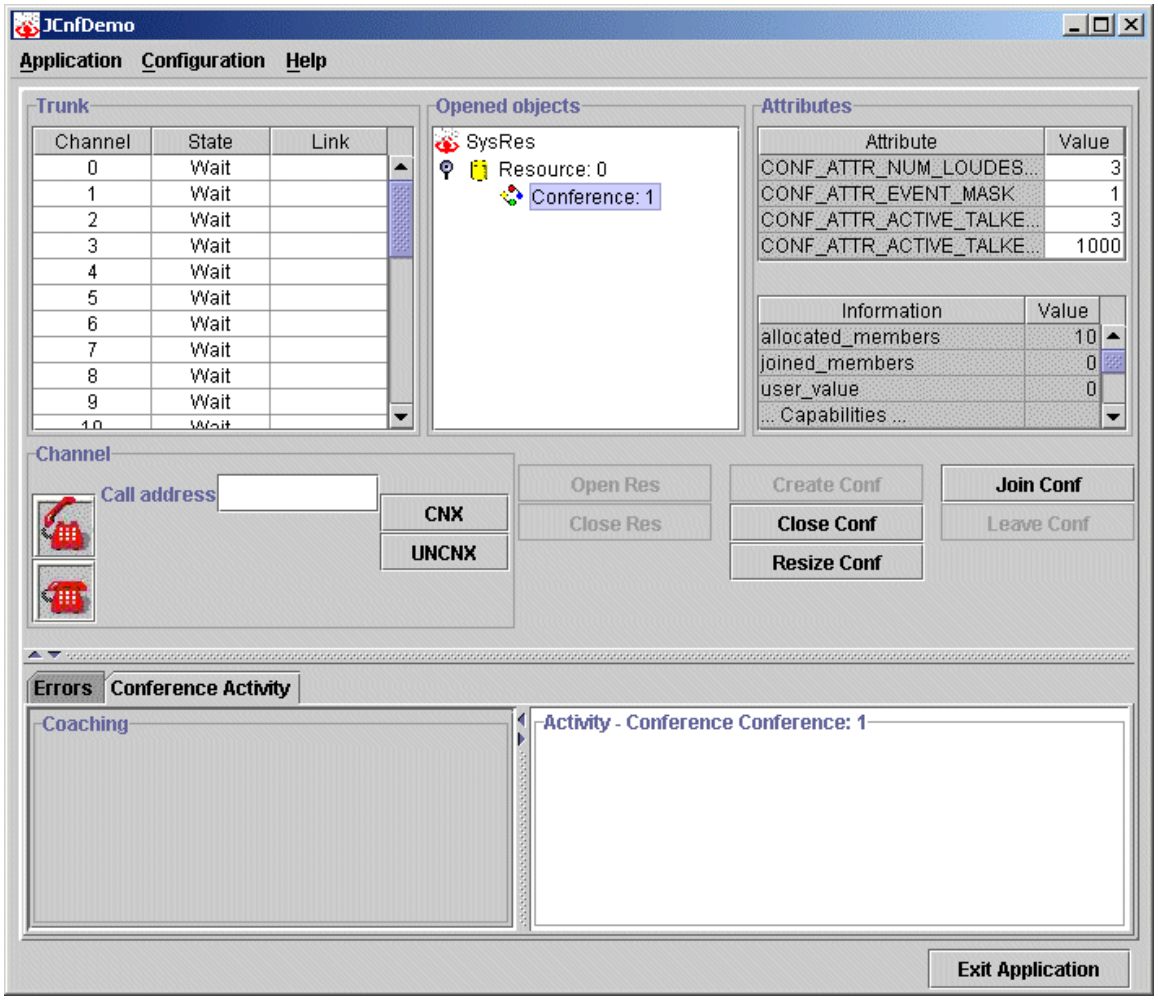


The Resource section lists the number of conferences running on the resource, the maximum number of members the resource supports, the number of available members, and the resource capabilities.

4. To enter information for the new conference, complete the fields as described in the following table:

Field	Description
allocated_members	Enter the number of members to allocate to the conference.
application_string	Enter a user-defined name for this conference; for example, Conf 1. This name displays in the Opened Objects list on the JCnfdemo main window.
flags	Lists the capabilities you can disable for this conference. Select a capability to disable it.

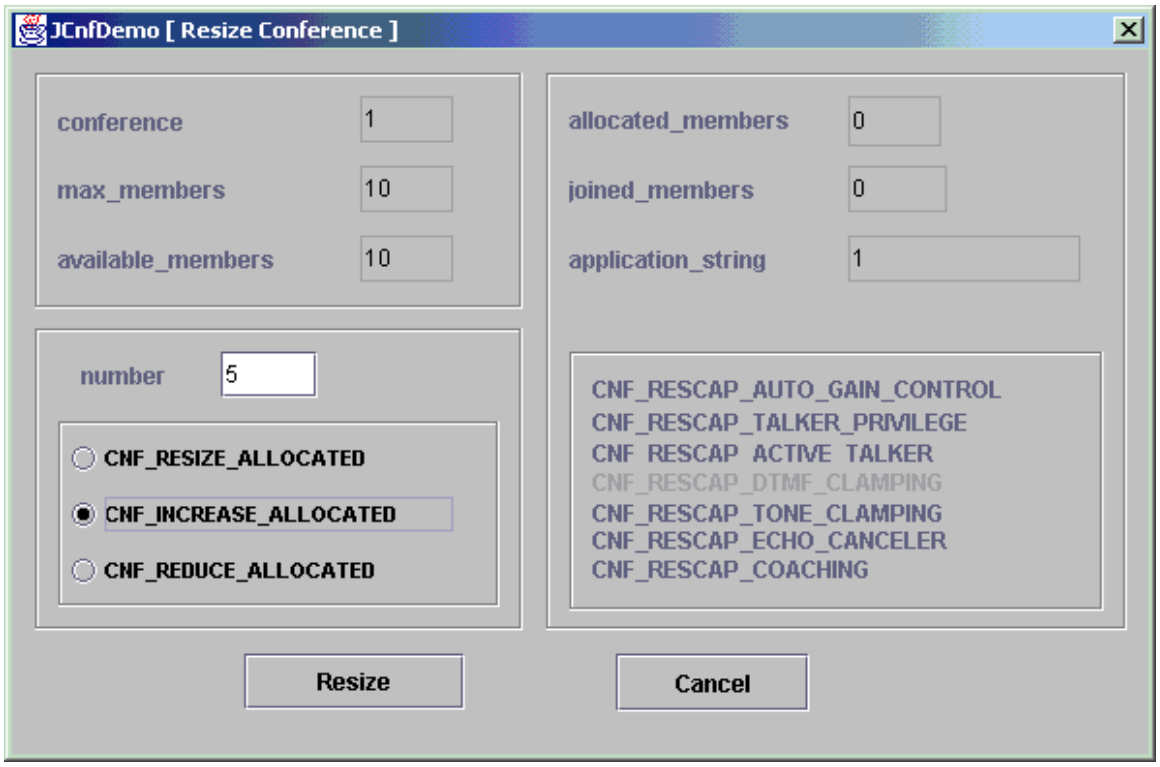
5. Click **Create** to create the new conference. The new conference displays in the Opened Objects list on the JCnfDemo window. The conference attributes and information displays in the Attribute and Information boxes.



Resizing conferences

You can adjust the number of members allocated to a conference at any time. To resize a conference:

1. On the *JCnfdemo* window, select the conference you want to resize and click **Resize Conf**. The Resize Conference dialog box displays:



The following information displays in the dialog box:

Section	Description
Resource	Displays information about the resource, such as the number of conferences running on the resource, the maximum number of members the resource supports, and the number of available members.
Conference	Displays current conference parameters, such as the number of members allocated and the number of members attending.
Resize Parameters	Enables you to resize the conference.
Capabilities	Displays the capabilities available for this conference.

2. Use the Resize Parameters section to change the number of members allocated to the conference:

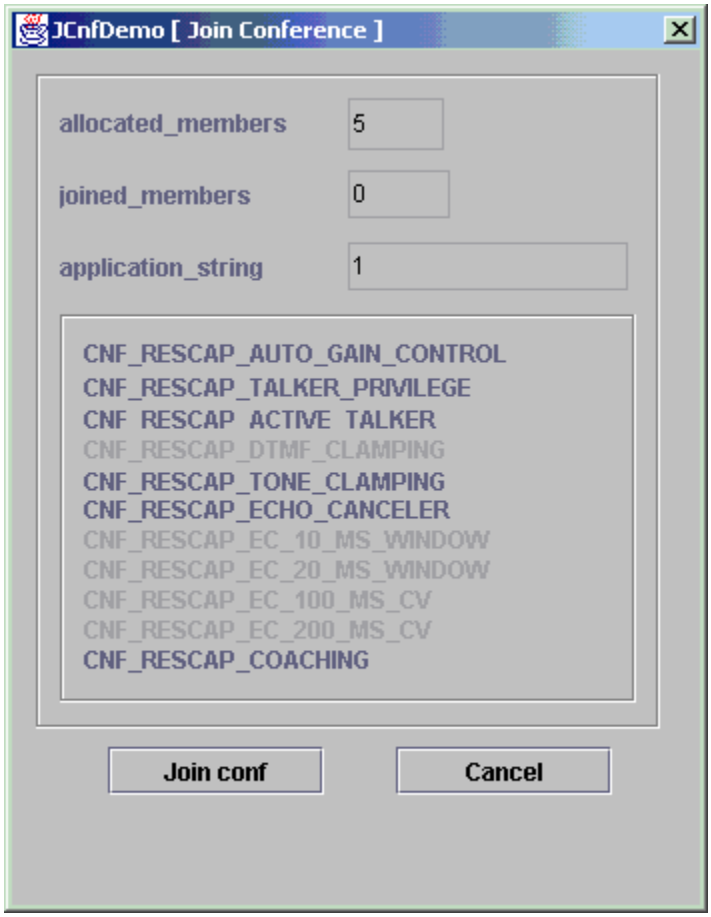
To...	Enter the number in the Number field and select...
Reduce the number of allocated members by a specific number	CNF_REDUCE_ALLOCATED
Increase the number of allocated members by a specific number	CNF_INCREASE_ALLOCATED
Resize the conference to a specific number	CNF_RESIZE_ALLOCATED

3. Click **Resize** to resize the conference.

Adding and removing members from a conference

Perform the following procedure to add and remove members from a conference:

1. On the *JCnfdemo* window, select the conference to which you want to add a member and click **Join Conf**. The Join Conference dialog box displays:

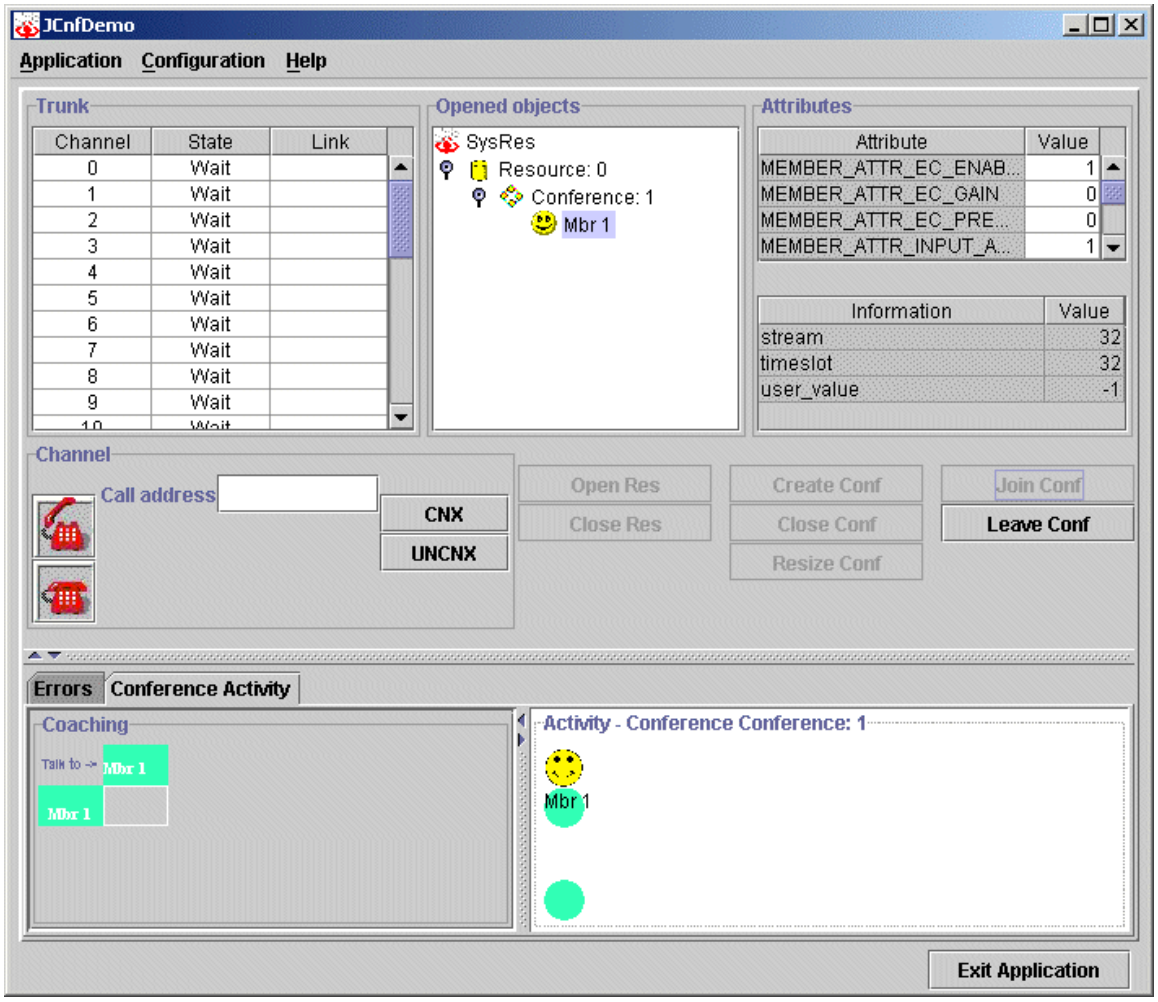


   The Conference section displays the number of members allocated to the conference, the number of members currently attending the conference, the conference identifier, and the conference capabilities.

2. In the application_string field of the Member section, enter a user-defined identifier for the member you are adding to the conference (for example, Mbr 1).

3. Click **Join conf**. The new member is added to the Opened Objects list on the main window. Attributes and information display in the Attributes/Information box.

A smiley face displays in the conference activity and the member is added in the coaching panel:

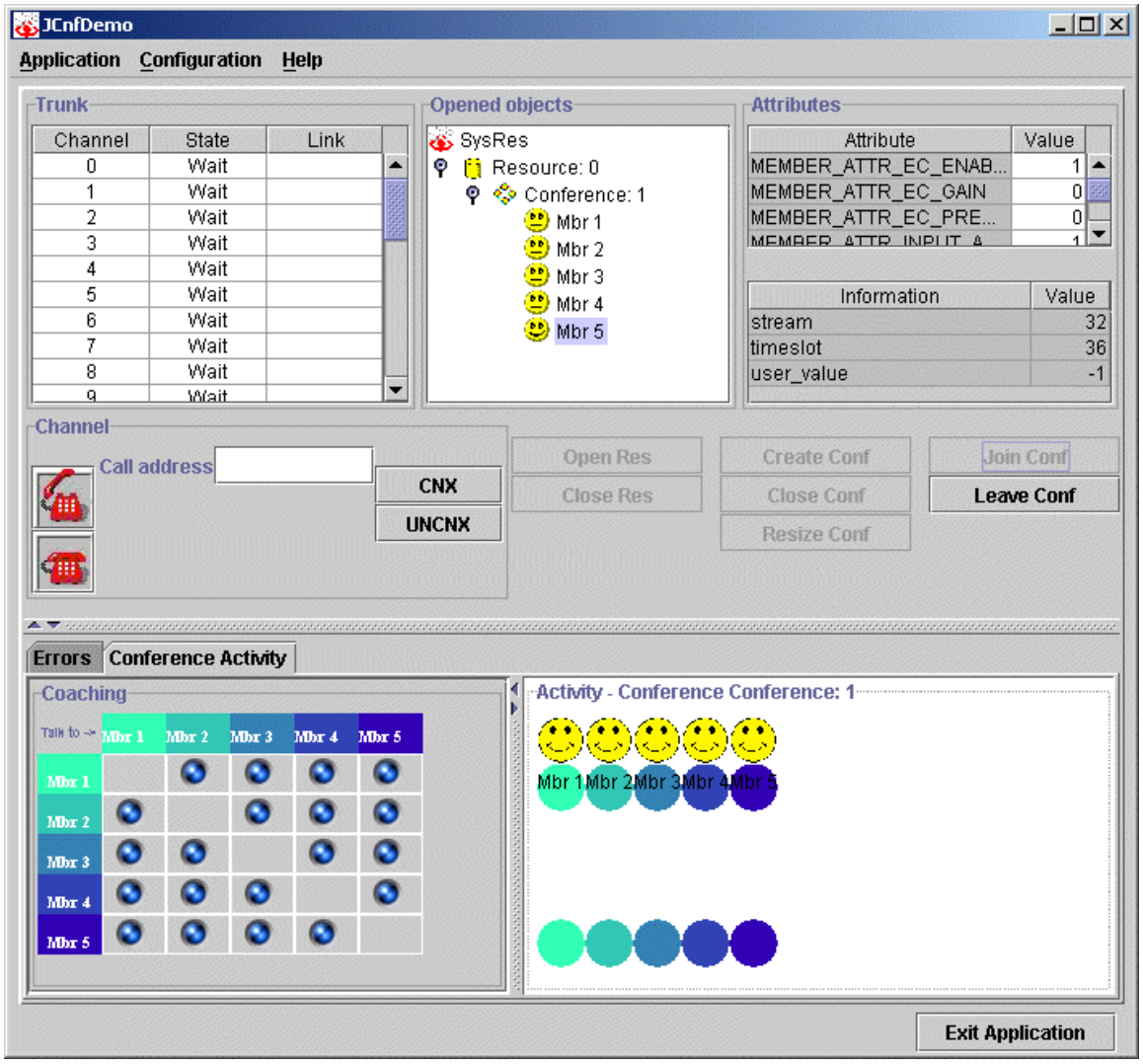


After a conference is created and members are added, the application receives the active talker event (CNFEVN_ACTIVE_TALKERS_CHANGE). *JCnfDemo* displays an animated *.gif* in the activity panel to identify the speaker. When the caller stops speaking, the animation stops.

4. To remove a member from a conference, select the member from the Opened Objects list on the main window and click **Leave Conf**.

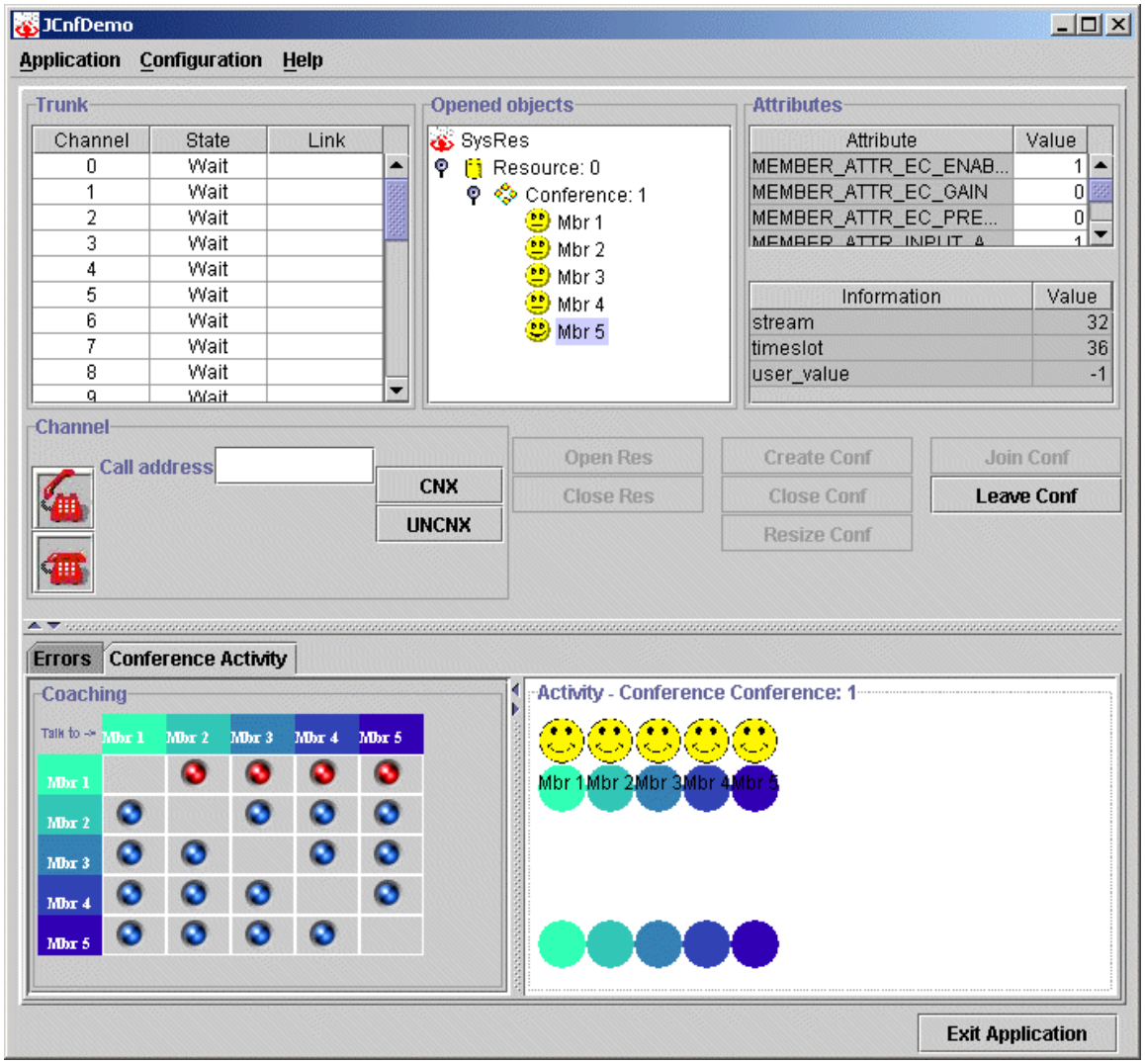
Setting the coaching between members

When adding several members in the same conference, you can view and set the coaching between the members. A panel of check boxes represents coaching, where each row and column represents a member.

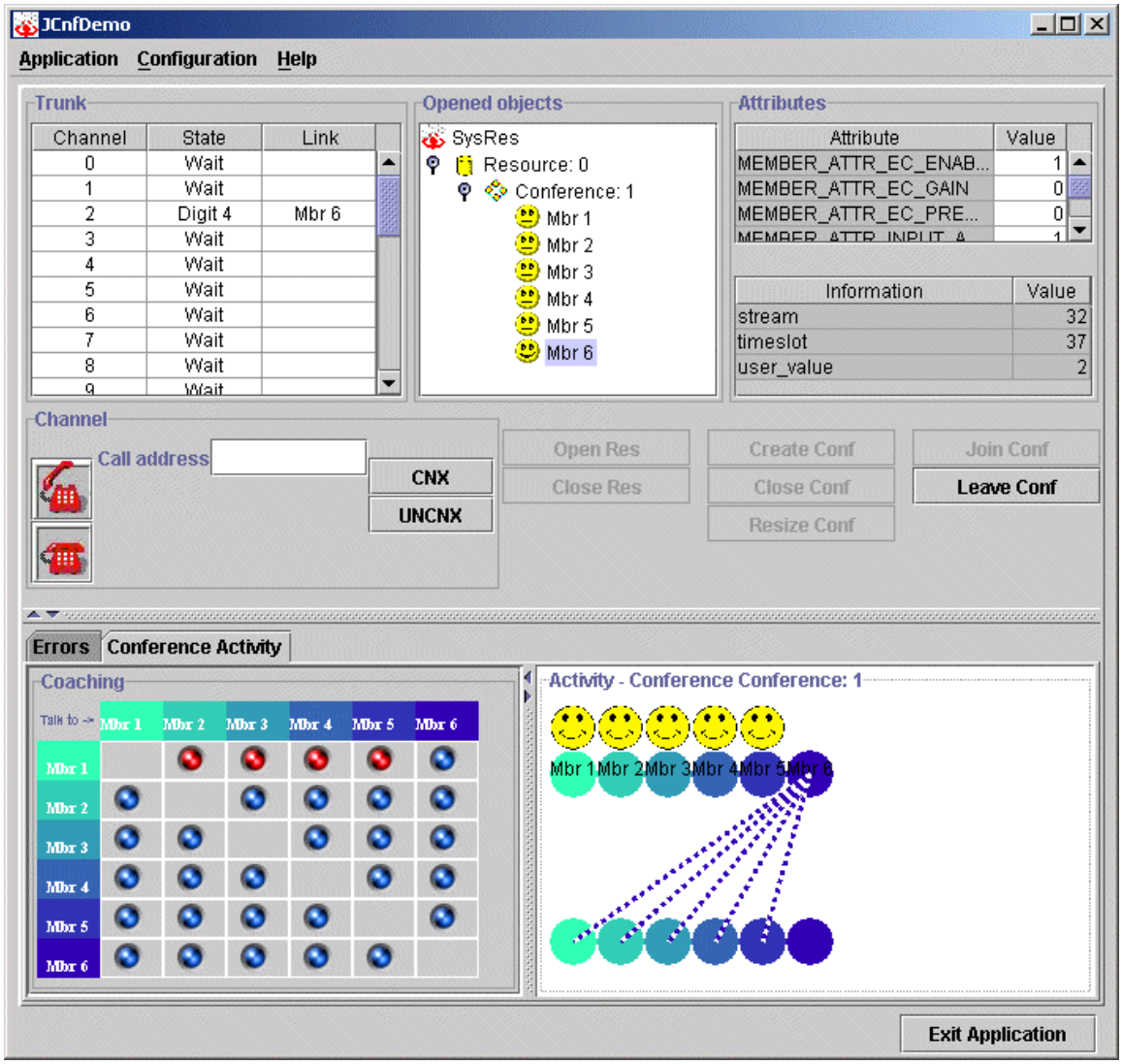


In this example, all five members can talk to each other. If a box is not selected, then those two members cannot talk to each other.

In the following example, Mbr1 cannot talk to Mbr 2, Mbr3, Mbr4, or Mbr 5.



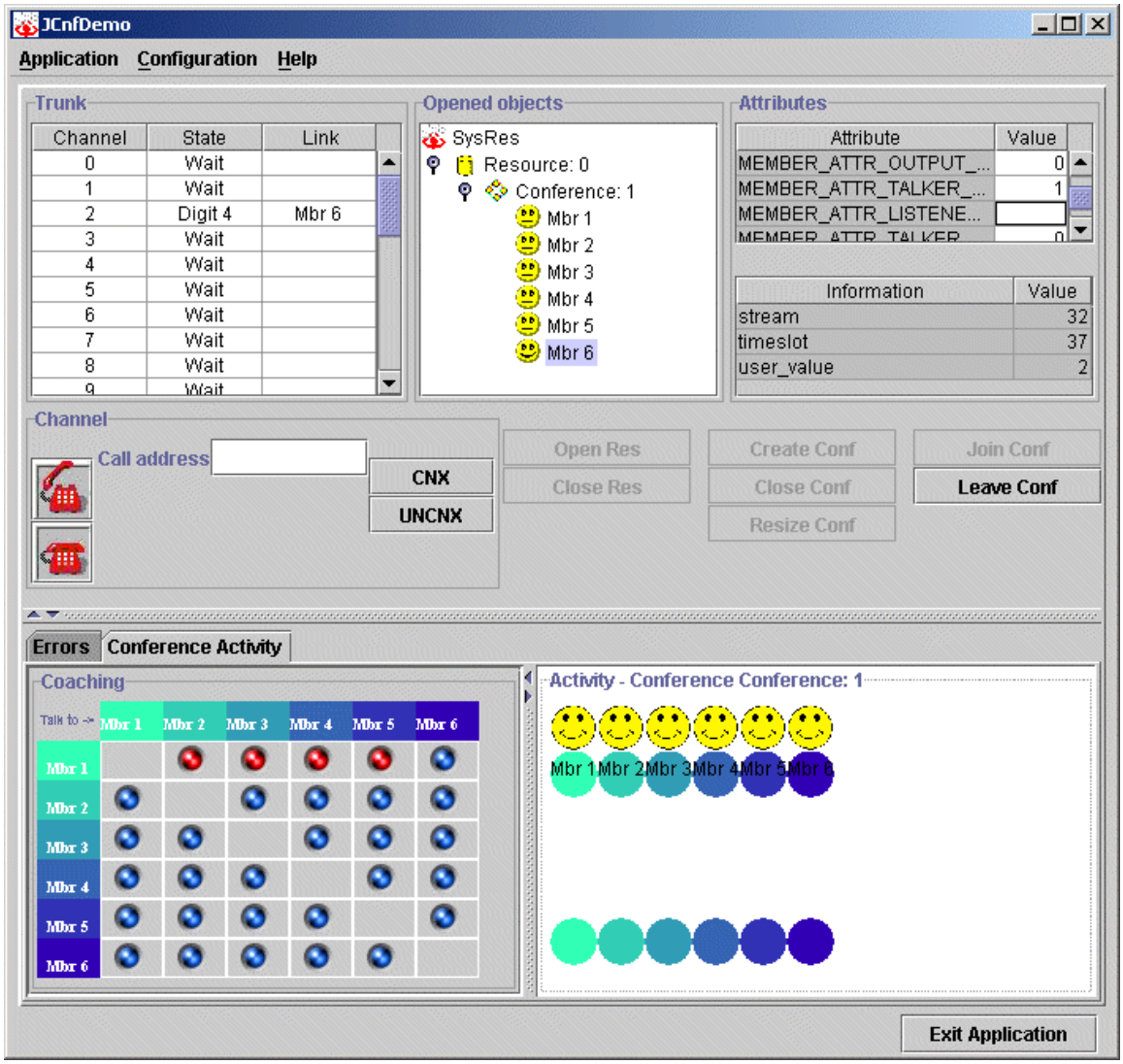
When a member is talking, a broken line is drawn between the speaker and the members who can hear him.


JCnfDemo
Application
Configuration
Help
Trunk
Channel
State
Link
0 Wait
1 Wait
2 Digit 4 Mbr 6
3 Wait
4 Wait
5 Wait
6 Wait
7 Wait
8 Wait
Opened objects
SysRes
Resource: 0
Conference: 1
Mbr 1
Mbr 2
Mbr 3
Mbr 4
Mbr 5
Mbr 6
Attributes
Attribute
Value
MEMBER_ATTR_EC_ENAB... 1
MEMBER_ATTR_EC_GAIN 0
MEMBER_ATTR_EC_PRE... 0
Information
Value
stream 32
timeslot 37
user_value 2
Call address
CNX
UNCNX
Open Res
Close Res
Create Conf
Close Conf
Resize Conf
Join Conf
Leave Conf
Errors
Conference Activity
Coaching
Talk to -->
Activity - Conference Conference: 1
Exit Application

Changing attributes

Perform the following steps to change attributes for individual members and conferences:

1. In the Opened Objects list of the JCnfDemo window, select the member or conference whose attributes you want to change.
2. In the Attributes section, double-click on the attribute value and edit it. The new value is validated, and the attribute is changed.



Receiving calls

Using *JCnfDemo* you can automatically connect an incoming call to a conference. To use this feature, create two conferences, conference 1 and conference 2. You do not need to add members to the conferences.

When a channel receives an incoming call, a welcome message asks the caller to press DTMF 1 or DTMF 2 to select a conference. *JCnfDemo* creates a new member on the selected conference, and connects the caller to the conference with an MVIP connection. After the caller hangs up, *cnfdemo* restores the MVIP connection and releases the channel.

Closing conferences and resources

Perform the following steps to close a conference:

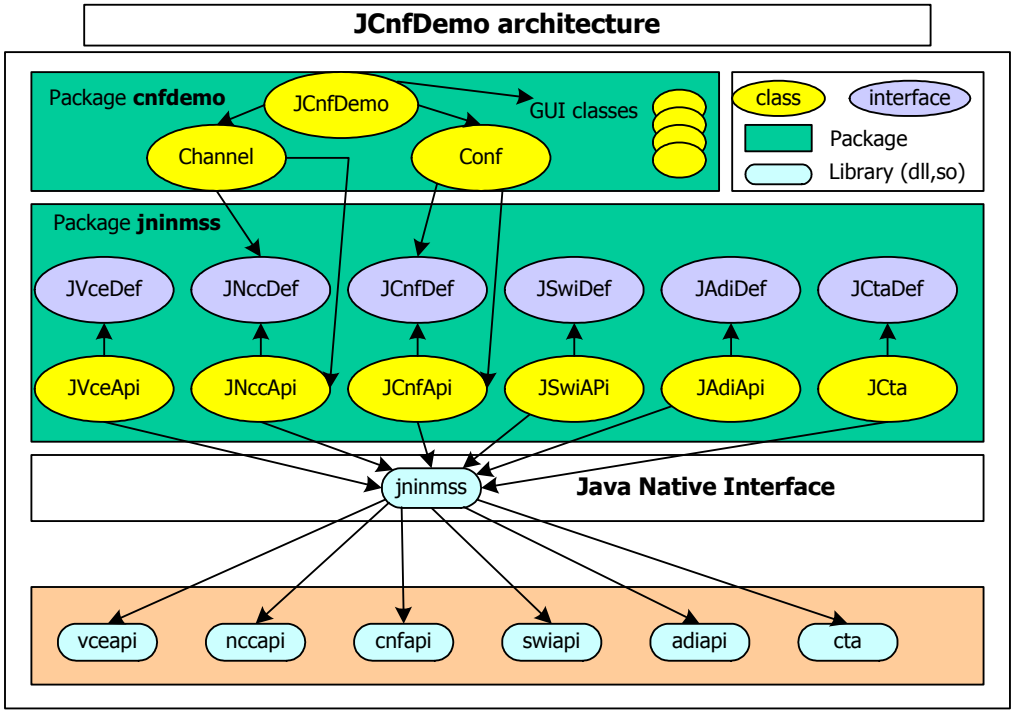
Step	Action
1	On the *JCnfDemo* window, select the conference you want to close.
2	Click **Close Conf**. The conference is closed and all members are removed.

Perform the following steps to close a resource:

Step	Action
1	On the *JCnfDemo* window, select the resource you want to close.
2	Click **Close Res**. The resource is closed, all members are removed, and all conferences are closed.

JCnfDemo architecture

The following illustration shows the *JCnfDemo* architecture:



Note: To keep the illustration clear, not all of the links are shown. *Conf*, *Channel,* and *JCnfDemo* implement most of the *J*Def* interfaces.

The *jninmss* library uses native code through Java. It provides a link between classes and C structures, and forwards the commands to the Natural Access libraries.

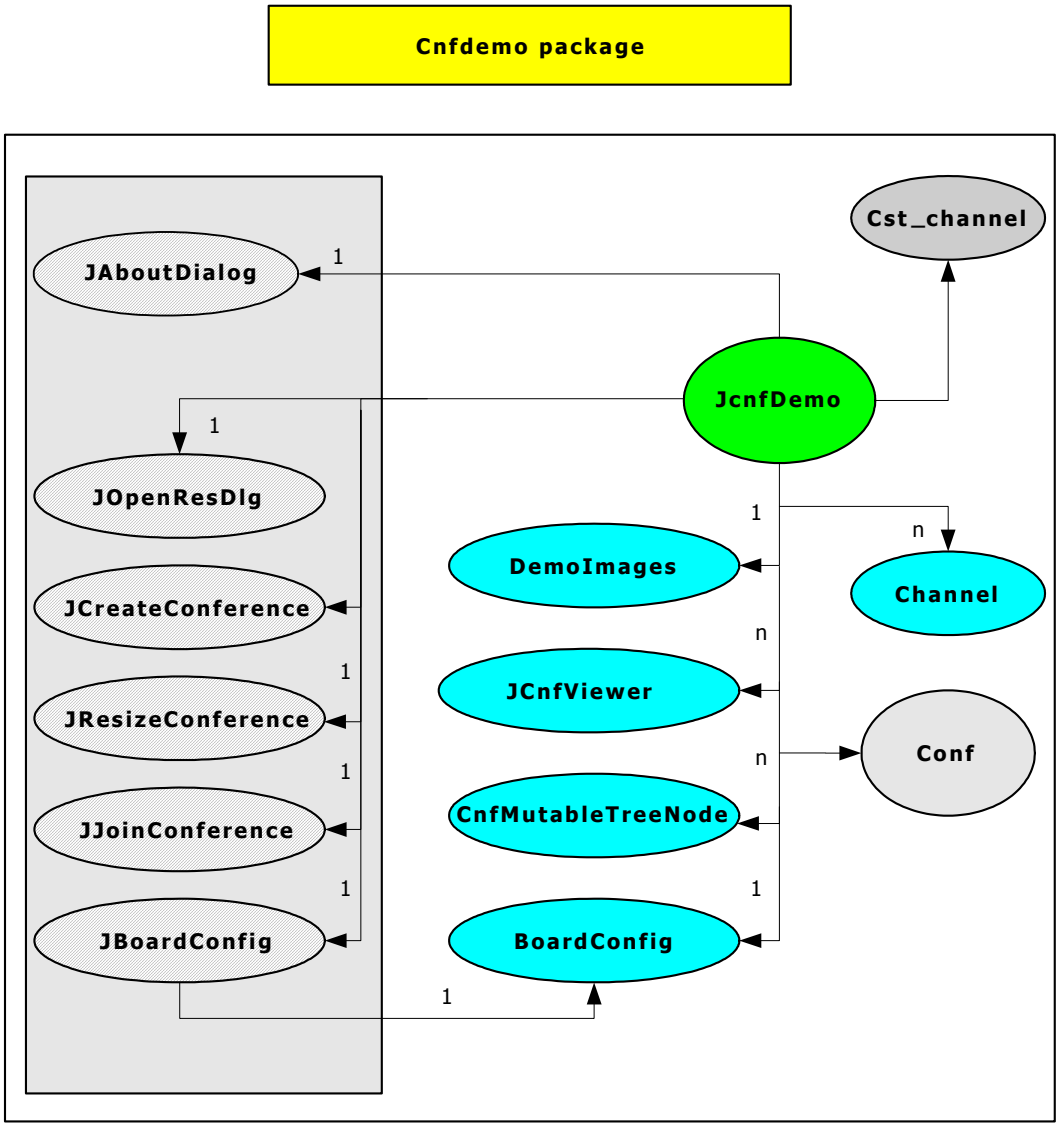
The *JCnfDemo* consists of the following packages:

Package	Files	Description
cnfdemo	GUI files	Contains the graphical user interface, *conf*, and *channel* classes.
cnfdemo	*JCnfDemo.java*	*JCnfDemo* is the main class. It provides the GUI and launches the *conf* and *channel* threads.
cnfdemo	*Conf.java*	*Conf* is a thread that listens and sends conference events.
cnfdemo	*Channel.java*	*Channel* is a thread that listens and sends channel events.
jninmss	*JAdiDef.java, JAdiApi.java, JNccDef.java, JNccApi.java, JCnfDef.java, JCnfApi.java, JVceDef.java, JVceApi.java, JSwiDef.java, JSwiApi.java, JCtaDef.java, JCta.java, JNMSSLoader.java, JNccApi.java, JNccDef.java, JSwiApi.java, JSwiDef.java, JVceApi.java, JVceDef.java, common.cpp, common.h, jadiapi.cpp, jcnfapi.cpp, jcta.cpp, jnccapi.cpp, jswiapi.cpp, jvceapi.cpp*	The *J*Def.java* files are interface files that wrap the structures defined in **def.h*. The *J*Api.java* files consist of the *J*Api* class definition that wrap the **api* library from Natural Access. *.*cpp* files are native wrappers of Natural Access libraries.

A java class, a java interface, and a native wrapper represent each Natural Access service.

The cnfdemo package

The following illustration shows the architecture of the *cnfdemo* package:



The following table describes the classes:

Class/interface	Description
JCnfDemo	Entry point. Launches the graphical user interface for NaturalConference and the channels with Natural Access services.
Conf	Thread to receive Natural Access events.
Channel	Thread corresponding to a line.
Cst_channel	Interface.
DemoImages	Loads images.
JcnfViewer	Conference activity viewer.
CnfMutableTreeNode	Node from the tree to represent resource, conference, or member.

Class/interface	Description
BoardConfig	Class to read/write configuration.
JAboutDialog	Dialog box when asking about information.
JOpenResDlg	Dialog box when opening resource.
JCreateConference	Dialog box when creating conference.
JResizeConference	Dialog box when resizing conference.
JJoinConference	Dialog box when joining a conference.
JBoardConfig	Dialog box when configuring the demo.

Cnfdemo package implements interfaces of the *jninmss* package because *Conf* and *Channel* use the *jninmss* package.

The jninmss package

Java conferencing belongs to the *jninmss* package. *jninmss* consists of two files:

- *JCnfDef.java*
- *JCnfApi.java*

Because *JCnfDef.java* implements an interface, java classes should implement *JCnfDef.java* to use the CNF API.

JCnfDef contains all the constants relative to the CNF API, and all the definitions of classes that wrap the C structures used in the C CNF API.

The following code sample is an extract from the *JCnfDef.java* file:

```
public interface JCnfDef
{
// Constants definition
/*-------------------------------------------------------------------------
CNF Service ID.
*/
public static int CNF_SVCID                          = 0x1A;

/*
* Base number for defining other keys
*/
public static int CNF_BASEID                         = 0x001A0000;

// Other constants definition
[...]
// C structures wrapped by java classes:

/* Conference parameters (used in cnfCreateConference)
*/
final public class CNF_CONFERENCE_PARMS
{
// Object size
public int      size;
// Conference creation flags
public int      flags;
// Number of allocated members
public int      allocated_members;
// User value.
public int      user_value;
};

// Other C Structures definition...
[...]
}
```

JCnfApi.java wraps the Natural Conference (CNF) API and contains the definition of the class *JCnfAPI*. It can also access *CnfApi* through *JCnfApi,* forwarding calls to Natural Access.

The following code sample is an extract from the *JcnfApi.java* file:

```
public class JCnfApi implements JCnfDef, JCtaDef
{
/**
* jcnfOpenResource native interface wrapper.
*
* @param cta_handles CTA_HANDLE object.
* @param cnf_handles CNF_HANDLES object.
* @param resourcenum Resource number.
* @return int iRet, cnfOpenResource native return value.
*/

public int jcnfOpenResource ( CTA_HANDLES cta_handles ,
                 CNF_HANDLES cnf_handles , int resourcenum )
{
int iRet;
iRet = cnfOpenResource ( cnf_handles , cta_handles.ctahd ,
                resourcenum );
return (iRet);
}

[....]

/////// PRIVATE NATIVE INTERFACE   ...
/**
* cnfOpenResource native interface.
*
* @param cnf_handles CNF_HANDLES object.
* @param ctahd CTA context handle.
* @param resourcenum Resource number.
* @return int iRet, cnfOpenResource native return value.
*/
private native int cnfOpenResource ( CNF_HANDLES cnf_handles , int ctahd ,
                        int resourcenum );

// <<native>> means that the Virtual Machine will call cnfOpenResource
// from C Library jcnfapi

[...]
}
```

The *javah* program creates a header file from the *JcnfApi.java* file, called *jninmss_JcnfApi.h*. This file describes the entry points of the *jninmss* library.

The *JCnfApi* library links with *cnfapi*. It forwards all calls to the C library after transferring data from objects to structures.

Event management

Events are retrieved using **ctaWaitEvent** in a thread. NaturalConference uses one thread for all conferences. This thread waits for an internal shutdown event, an active talker change event, or a notification event.

Each channel has its own waiting events thread. When events are received, the corresponding thread notifies the *JCnfDemo*, which then updates the GUI.

For more information, see *NaturalConference events* on page 59.

11 Errors and events

Alphabetical error summary

All Natural Access functions return a status code. If the return code is not SUCCESS (0), it is an error code indicating that the function failed and the reason for the failure.

NaturalConference error codes are defined in the *cnfdef.h* include file. The error codes are prefixed with CNFERR_.

The following table alphabetically lists the NaturalConference service errors. All errors are 32 bit.

Error name	Hexadecimal	Decimal	Description
CNFERR_BOARD_TIMEOUT	0x1A0007	1703943	Board timed out while waiting for a response message. This can happen if the system is overloaded or the board is dead.
CNFERR_CANNOT_CREATE_CHANNEL	0x1A0008	1703944	No communication channels are available for use on the specified resource. Other processes running in Library mode may be using too many channels.
CNFERR_CONFERENCE_EMPTY	0x1A0009	1703945	A tone started playing on an empty conference.
CNFERR_INVALID_ATTRIBUTE	0x1A0002	1703938	The attribute is not valid. Verify that the attribute is valid and corresponds to the identifier type (conference/member). Attribute keywords for conference and member are declared in *cnfdef.h*.

Error name	Hexadecimal	Decimal	Description
CNFERR_INVALID_IDENTIFIER	0x1A0001	1703937	Conference or member identifier is not valid. This can happen when: • A member identifier is used when a conference identifier is expected. • A conference identifier is used when a member identifier is expected. • The identifier was not created using the ***cnfresourcehd*** used for calling the function. • The corresponding object has been already destroyed.
CNFERR_INVALID_PARAMETER	0x1A0003	1703939	One of the parameters used is inconsistent or out of range.
CNFERR_NOT_ENOUGH_RESOURCE_SPACE	0x1A0006	1703942	Not enough available space exists on the conference resource to perform the desired operation. Either the number of reserved seats with the requested capabilities is exhausted, or the number of conference IDs for that resource is exhausted. There are only 255 conference IDs per conference resource.
CNFERR_RESOURCE_NOT_AVAILABLE	0x1A0005	1703941	Specified resource is currently in use by another process or does not exist.
CNFERR_RESOURCE_NOT_DEFINED	0x1A0004	1703940	Specified resource is incorrectly declared in the NaturalConference configuration file (*cnf.cfg*).

Numerical error summary

The following table numerically lists the NaturalConference errors:

Hexadecimal	Decimal	Error name
0x1A0001	1703937	CNFERR_INVALID_IDENTIFIER
0x1A0002	1703938	CNFERR_INVALID_ATTRIBUTE
0x1A0003	1703939	CNFERR_INVALID_PARAMETER
0x1A0004	1703940	CNFERR_RESOURCE_NOT_DEFINED
0x1A0005	1703941	CNFERR_RESOURCE_NOT_AVAILABLE
0x1A0006	1703942	CNFERR_NOT_ENOUGH_RESOURCE_SPACE
0x1A0007	1703943	CNFERR_BOARD_TIMEOUT
0x1A0008	1703944	CNFERR_CANNOT_CREATE_CHANNEL
0x1A0009	1703945	CNFERR_CONFERENCE_EMPTY

Events

NaturalConference uses the Natural Access event reporting mechanism, **ctaWaitEvent. ctaWaitEvent** returns the CTA_EVENT structure informing the application about the events that occurred on each context. The structure includes information specific to the event. Events are prefixed with CNFEVN_.

The event structure contains the following information:

Field	Description
id	Event code (CNFEVN_***XXX***).
ctahd	Context handle the conference was created on.
timestamp	Time the event was created.
userid	Defined by **ctaCreateContext**.
size	Size of the area pointed to by buffer. If the buffer is NULL, this field may be used to hold an event-specific value.
buffer	Points to data returned with the event.
value	***confid*** of the conference that sent the event.
objHd	***cnfresourcehd*** the conference was created on.

The following table lists the NaturalConference events:

Event name	Hexadecimal	Decimal	Description
CNFEVN_ACTIVE_TALKERS_CHANGE	0x1A2003	1712131	One or more of the active talkers changed. To generate this event, set the CNF_EVNMSK_ACTIVE_TALKERS_CHANGE bit in the conference event mask. To change the delay between sending two CNFEVN_ACTIVE_TALKERS_CHANGE events, set the attribute in the corresponding conference.
CNFEVN_CLOSE_RESOURCE_DONE	0x1A2002	1712130	***cnfresourcehd*** was released.
CNFEVN_OPEN_RESOURCE_DONE	0x1A2001	1712129	***cnfresourcehd*** was allocated.
CNFEVN_TONE_DONE	0x1A2004	1712132	Tone generation completed or was stopped. To generate this event, set the CNF_EVNMSK_TONE_DONE bit in the conference event mask.

For information about monitoring conference events, refer to *NaturalConference events* on page 59.

12 NaturalConference parameters

Overview of NaturalConference parameters

The behavior of some NaturalConference functions is defined by multiple parameters. These parameters are grouped together into structures for convenience and efficiency. Each parameter structure has a default value that is compatible with many configurations. The parameters can be modified to:

- Enable or disable function features.
- Adapt a function for exceptional configurations.

This section describes the following NaturalConference parameter structures:

- CNF.CONFERENCE
- CNF.CONFERENCE_ATTR
- CNF.MEMBER
- CNF.MEMBER_ATTR
- CNF.TONE

The dependent functions are also specified.

CNF.CONFERENCE

Dependent function: **cnfCreateConference**

Field name	Type	Default	Units	Description
flags	DWORD	0	Mask	Capabilities the conference does not plan to use. You can disable capabilities at the resource level (in *cnf.cfg* as described in *NaturalConference configuration file statements* on page 39) or at the conference level. NaturalConference optimizes the conference resource usage according to the capabilities used, and can possibly allow the use of more members on the resource if enough system resources are freed up. Use the following flags to disable capabilities at the conference level: CNF_NO_COACHING CNF_NO_DTMF_CLAMPING CNF_NO_ECHO_CANCEL CNF_NO_TONE_CLAMPING
allocated_members	DWORD	0	Member	Number of members NaturalConference reserves for the conference. Whether the allocated members are used (joined) or not, the corresponding space is reserved on the conference resource. Call **cnfResizeConference** to modify the allocated.space. When using coaching (CNF_NO_COACHING is not specified in the flags), the maximum number of members in a conference is limited to 32.
user_value	DWORD	0		Application-supplied value or pointer that can be retrieved by calling **cnfGetConferenceInfo**.

CNF.CONFERENCE_ATTR

Dependent function: **cnfSetConferenceAttribute**

Field name	Type	Default	Units	Description
num_loudest	DWORD	3	Member	Number of members selected for building the conference's output signal. Recommended values are 3 - 5.
event_mask	DWORD	0	Mask	Mask used to determine which events NaturalConference generates for the conference. Valid values are: 0 CNF_EVNMSK_ACTIVE_TALKERS_CHANGE CNF_EVNMSK_TONE_DONE
active_talkers	DWORD	3	Member	Number of members retained for building the list of active talkers and generating CNFEVN_ACTIVE_TALKERS_CHANGE if a change occurs. Recommended values are 3 - 5.
active_talkers_timer	DWORD	1000	ms	Time NaturalConference waits between sending two CNFEVN_ACTIVE_TALKERS_CHANGE events. Valid range is 100 - 60000.

CNF.MEMBER

Dependent function: **cnfJoinConference**

Field name	Type	Default	Units	Description
user_value	DWORD	0	Integer	User defined value.

CNF.MEMBER_ATTR

Dependent functions: **cnfSetMemberAttribute**, **cnfGetMemberAttributeList**

Field name	Type	Default	Units	Description
ec_enabled	DWORD	1	Boolean	Enables echo cancellation capability on the input signal. Set to 1 to enable EC and 0 to disable it.
ec_gain	INT32	0	dB	Amount of amplification applied to echo input. Valid automatic gain range is -54 - +24.
ec_predelay	DWORD	0	ms	Output sample delay. Valid automatic gain range is 0 - 9.
input_agc_enable	DWORD	1	Boolean	Enables automatic gain control on the input signal. Set to 1 to enable AGC and 0 to disable it.
input_agc_targetampl	INT32	-19	dBm	Target amplitude for AGC. Valid range is -45 - 0.
input_agc_silenceampl	INT32	-40	dBm	Noise threshold for AGC. Gain adjustment is suspended for signals below this level. Valid range is -45 - 0.
input_gain	INT32	0	dB	Gain applied to the signal before it is encoded. If AGC is enabled, this is the initial gain. Valid automatic gain range is -12 - +12.
input_law	DWORD	0	Integer	G.711 law to be applied in the member's input. Valid values are: 0 = Default 1 = A-law 2 = mu-law
output_agc_enable	DWORD	1	Boolean	Enables automatic gain control on the output signal. Set to 1 to enable AGC and 0 to disable.
output_agc_targetampl	INT32	-19	dBm	Target amplitude for output AGC. Valid range is -45 - 0.
output_agc_silenceampl	INT32	-40	dBm	Noise threshold for output AGC. Gain adjustment is suspended for signals below this level. Valid range is -45 - 0.
output_gain	INT32	0	dB	Gain applied to the encoded audio. If output AGC is enabled, this is the initial gain. Valid automatic gain range is -12 - +12.

Field name	Type	Default	Units	Description
output_law	DWORD	0	Integer	G.711 law to be applied in the member's output. Valid values are: 0 = Default 1 = A-law 2 = mu-law
self_echo_enable	DWORD	0	Boolean	Indicates if the voice received from this member on its input is mixed and returned to that member on its output. This attribute can be enabled when the output is not connected to the member, but is actually used for recording. Set to 1 to enable and 0 to disable.
talker_enable	DWORD	0	Boolean	Enables the member as talker. Set to 1 to enable and 0 to disable.
listener_enable	DWORD	1	Boolean	Enables the member as listener. Set to 1 to enable and 0 to disable.
talker_privilege	DWORD	0	Boolean	Enables the member as privileged talker. Set to 1 to enable and 0 to disable.
dtmf_clamping_enable	DWORD	1	Boolean	Enables the DTMF clamping capability. Set to 1 to enable and 0 to disable.
dtmf_clamping_delay_line	DWORD	28	ms	Delay line to buffer the signal received by each member before removing the DTMF with the DTMF clamping module. Valid range is 0 - 28 ms.
tone_clamping_enable	DWORD	1	Boolean	Enables the tone clamping capability. Set to 1 to enable and 0 to disable.

CNF.TONE

Dependent function: **cnfStartTone**

Field name	Type	Default	Units	Description
freq1	DWORD	1000	Hz	First (or only) frequency of the generated tone. Valid range is 200 - 3600.
ampl1	INT32	-20	dBm	Amplitude of the first (or only) frequency component. Valid range is -54 - 3.
freq2	DWORD	0	Hz	Second frequency of the generated tone, or 0 if the tone is a single frequency. If not 0, valid range is 200 - 3600.
ampl2	INT32	0	dBm	Amplitude of the second frequency component, if any. Valid range is -54 - 3.
ontime	DWORD	200	ms	Duration of the tone. Valid range is 1 - 65535.
offtime	DWORD	0	ms	Duration of silence between tones. Specify 0 for no off time. Valid range is 0 - 65535.
iterations	INT32	1	integer	Number of times to repeat the alternating tone and silence period. A count of -1 means repeat forever. Otherwise the valid range is 1 - 32767.

Index