CT-Connect for Aspect

Programming Guide

Order Number: 05-0803-001

Revision/Update Information: This is a new manual.

Software and Version: CT-Connect™ Server for Aspect
Version 1.1, and
CT-Connect Application Programming
Interface for Aspect Version 1.1.

Date: November 18, 1996

Contents

2 Routine Specifications

3 Error and Condition Values Returned

Index

Figures

Tables

About This Manual

Dialogic®'s CT-Connect™ (CTC) for Aspect software enables a telephony application to access the features of an Aspect CallCenter® configured with the Aspect Application Bridge® software. Throughout this manual, the abbreviation CTC/AB (where AB stands for Application Bridge) is used to refer to the CTC for Aspect software.

This manual describes how to write telephony applications using the CTC Application Programming Interface (API) for Aspect software, and includes detailed descriptions of all CTC/AB programming routines.

Aspect Application Bridge Software

To support CTC/AB features described in this manual, you require one of the following:

- Aspect Application Bridge Release 5.0

  Note that not all CTC/AB features are supported with Application Bridge Release 5.0. Refer to Table 1 for details.

- Aspect Application Bridge Release 6.0. The following software options are required to fully support CTC/AB features:

  – Event Bridge™

    This software enables CTC/AB to use Event Bridge messages to communicate changes in agent states and call states for calls involving trunks or TeleSets.

  – Resource Bridge™

    This software enables your application to use the ctcAbReassignResource routine to change the ACD group, supervisor team, or Class of Service (COS) for an agent, and the ctcAbMakePredictiveCall routine to initiate calls on behalf of an ACD group. Chapter 2 describes these routines in detail.

CTC/AB Features and Application Bridge Release 5.0

Table 1 describes the restrictions that apply when you use CTC/AB with Application Bridge Release 5.0.

Table 1 CTC/AB Features and Application Bridge Release 5.0

CTC/AB Function	Restriction
ctcAbAddMonitor	Application Bridge Release 5.0 does not support monitoring ACD groups or trunk groups on a monitor channel.
ctcAbAnswerCall	This routine is not supported with Application Bridge Release 5.0.
ctcAbAssign	Application Bridge Release 5.0 does not support channels assigned to ACD groups or trunk groups.
ctcAbGetEvent and ctcAbWinGetEvent	The following restrictions apply: • Some events are not supported with Application Bridge Release 5.0. Refer to Tables 2–2, 2–3, and 2–4. • The following ctcAbEventData structure fields do not return information: aniDigits dnisDigits agentId agentGroup agentMode
ctcAbHangupCall	Application Bridge Release 5.0 does not supply a call reference identifier for the call. This routine hangs up the current call.
ctcAbMakePredictiveCall	This routine is not supported with Application Bridge Release 5.0.
ctcAbReassignResource	This routine is not supported with Application Bridge Release 5.0.
ctcAbRetrieveHeld	Application Bridge Release 5.0 does not support a call reference identifier for the call. This routine retrieves the call currently on hold.
ctcAbSetAgentStatus	Application Bridge Release 5.0 does not support agent log in or log out.

For more information, refer to the routine descriptions in Chapter 2.

Audience

This manual is for programmers writing applications that use a link between a CTC/AB server (a Windows NT™ PC running the CTC Server for Aspect software) and an Aspect CallCenter to provide users at client systems with computer-integrated telephony facilities.

This manual assumes that readers are familiar with:

- Writing programs in C
- Compiling and linking programs on the operating system(s) used by your CTC/AB clients:

  Windows NT
  Windows™ 95
  Windows 3.1/3.11 (or Windows for Workgroups 3.11)
  Digital™ UNIX® (formerly DEC™ OSF/1™)
  HP-UX®
  SCO OpenServer™
  OpenVMS™
  OS/2 WARP™ Connect

You must also have an understanding of Aspect CallCenter programming (refer to your Aspect CallCenter and Application Bridge documentation) and CTC/AB networks. Chapter 1 provides an overview of CTC/AB systems and how they are linked to the Aspect CallCenter.

Associated Documentation

CTC for Aspect Documentation

The following manuals are included in the CTC for Aspect documentation set:

- *CT-Connect for Aspect Installation Guide*

  This manual describes how to install and configure the CTC Server for Aspect and the CTC/AB API software.

- *CT-Connect for Aspect Management Guide*

  This manual describes how to manage your CTC/AB network and provides guidelines for finding and fixing problems.

- *CT-Connect for Aspect DDE Interface*

  This online help describes how CTC/AB provides support for Dynamic Data Exchange (DDE). For details, refer to the *CT-Connect for Aspect Installation Guide*.

- *CT-Connect for Aspect Release Notes*

  These online notes provide information about changes to the CTC/AB software and/or documentation at the time of release. Read the release notes before using the software.

Aspect Documentation

Refer to the Aspect CallCenter and Aspect Application Bridge documentation for details of features and any limitations that any affect the operation of the CTC/AB software.

Terms and Definitions

The following terms are used throughout this manual:

Term	Definition
OpenVMS	Refers to the OpenVMS VAX™ and OpenVMS Alpha operating systems.
Windows 3.1/3.11	Refers to Microsoft Windows 3.1, Windows 3.11, and Windows for Workgroups 3.11.
OS/2®	Refers to OS/2 WARP Connect.
CTC/AB	Refers to CTC for Aspect.
CTC/AB client	A system running the CTC/AB API software.
CTC/AB server	A Windows NT personal computer running the CTC Server for Aspect software.
Communications link	The logical link between the CTC/AB server and the Aspect CallCenter.

Conventions

The following conventions are used throughout this manual:

Convention	Meaning
`courier`	This typeface is used for code examples or interactive examples to indicate system input/output.
`drive:`	Italic (slanted) typeface indicates variable values, placeholders and function arguments.
`C:\>`	The MS–DOS® and OS/2 command prompt. The actual prompt may vary depending on your current drive and default directory.
`#`	The Digital UNIX, HP-UX, and SCO OpenServer command prompt.
`$`	The OpenVMS command prompt.

1
Programming Overview

This chapter provides a general introduction to the functions provided by the CTC/AB software and describes how to call CTC/AB routines.

1.1 Overview of CTC for Aspect

CTC/AB is a software toolkit for developing and running telephony applications. These applications create logical channels to telephony devices, for example, Aspect TeleSets, so that incoming and outgoing calls at the device can be controlled and call data monitored.

The logical channels are created over a CTC/AB communications link to an Aspect CallCenter running Application Bridge software.

1.1.1 CTC/AB Communications Link

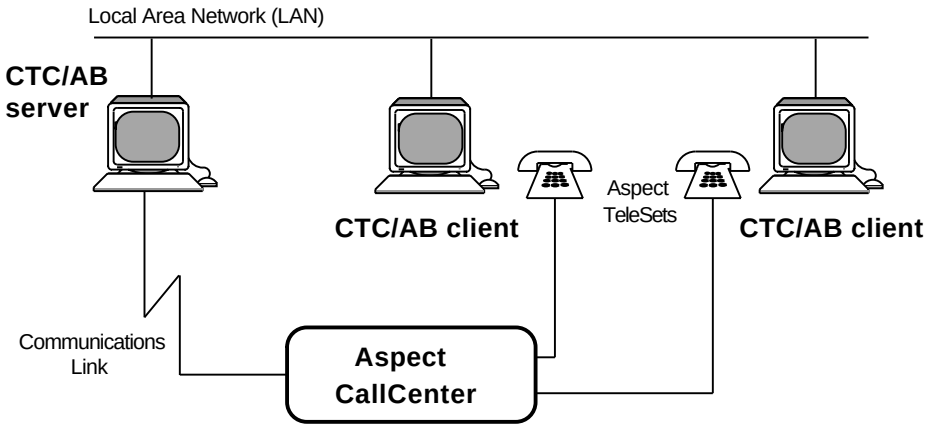
Figure 1–1 shows an example hardware and software configuration of a CTC/AB network and the CTC/AB communications link.

A CTC/AB application must run on a CTC/AB client, one of the following systems running the CTC/AB API software:

Windows NT
Windows 95
Windows 3.1/3.11
Digital UNIX
HP-UX
SCO OpenServer
OpenVMS
OS/2

CTC/AB clients pass requests to, and receive information from, the CTC/AB server (a Windows NT PC running the CTC Server for Aspect software). The CTC/AB server acts as an intermediary, passing messages between the CTC/AB clients and the Aspect CallCenter.

Figure 1–1 Example CTC/AB Network



1.1.2 Using Logical Channels

A logical channel from the CTC/AB application to a device enables the application to perform telephony functions and make use of call data and party information:

- A channel to a station or TeleSet enables the application to perform basic call processing and telephony functions usually accessed through the station or TeleSet, and receive call data and party information.
- A channel to a trunk enables the application to perform call processing (for example, deflect calls to another destination) and receive call data and party information for the trunk.
- A channel to an ACD group enables the application to make predictive calls on behalf of the group. It also enables your application to receive additional call data on channels assigned to TeleSets in the ACD group.
- A channel to a trunk group enables the application to receive additional call data on channels assigned to trunks in the trunk group.
- A channel to an InterQueue point enables the application to monitor calls that the CallCenter receives from other Aspect CallCenters in the same network.
- A channel to a monitor channel enables the application to receive information for a number of devices on one channel. Section 1.1.3 describes monitor channels.

1.1.3 Monitor Channels

A monitor channel is a single channel that your application can use to monitor multiple devices. By assigning to a monitor channel, your application can receive on one channel call data, status, and party information for a number of devices. Your application can use this information, for example, to record statistics for a specific group of TeleSet users.

Note that your application cannot use a monitor channel to control calls to devices or to set device features. You must assign logical channels to individual devices for call and feature control.

1.2 Introduction to CTC/AB Routines

CTC/AB routines can be grouped according to the following functions:

- Control of the communications channel
- Telephony functions

1.2.1 Controlling the Communications Channel

This group of routines gives you control of the communications channel between the user's process and a specified telephony device, allowing you to:

- Assign and deassign logical communications channels to and from a device
- Set and query certain characteristics for a device
- Monitor calls associated with a device
- Monitor telephony events on a channel

1.2.1.1 Routines That Control the Communications Channel

Table 1–1 lists the functions available to give you control of the communications channel, and the routines that provide those functions.

Table 1–1 Controlling the Communications Channel

Function	Routine
Assign a communications channel to a device and identify the channel uniquely to the application, or assign a monitor channel so that the application monitors a number of devices on a single channel.	ctcAbAssign
Set monitoring on for a device so that information about that device is returned on a monitor channel.	ctcAbAddMonitor
Stop monitoring a device on a monitor channel.	ctcAbRemoveMonitor
Deassign a channel from its associated device, and release resources associated with the channel.	ctcAbDeassign
Change the group, team, or Class Of Service (COS) for an agent	ctcAbReassignResource
Return information about the communications channel and the device to which the channel is assigned.	ctcAbGetChannelInformation
Declare the status of an agent by simulating the function keys on an Aspect TeleSet.	ctcAbSetAgentStatus
Set the monitoring state of the assigned device on or off. Use this routine with the ctcAbGetEvent routine to receive information on the state of calls associated with a device.	ctcAbSetMonitor
Return the monitoring state of the assigned device.	ctcAbGetMonitor
Return information on telephone calls associated with the assigned device. Depending on the type of call, you can receive information on: • Call events • Other parties involved in the telephone call	ctcAbGetEvent and ctcAbWinGetEvent
Return information about condition values.	ctcAbErrMsg

1.2.1.2 Sequence for Calling the Routines

Call the routines in the following sequence:

1. ctcAbAssign to assign the channel to a device.
2. ctcAbSetMonitor to set monitoring on for that channel.
3. ctcAbGetEvent or ctcAbWinGetEvent to monitor the channel and device while the application is making and receiving calls.

1.2.2 Telephony Functions

Table 1–2 lists the telephony functions provided by the CTC/AB API on a channel assigned to a device, and the routines that perform those functions.

Table 1–2 Telephony Functions

Telephony Function	Routine
Make a telephone call from the device to which the channel is assigned.	ctcAbMakeCall
Answer an incoming call on a hands-free feature telephone.	ctcAbAnswerCall
Clear the active call on the assigned device.	ctcAbHangupCall
Place a call on the assigned device on hold.	ctcAbHoldCall
Transfer an active call to a third party, disconnecting the assigned device.	ctcAbSingleStepTransfer
Make a call to a third party to whom you intend to transfer the current call on the assigned device, or to include all parties in a conference call.	ctcAbConsultationCall
Complete a transfer call initiated by ctcAbConsultation-Call, and disconnect the assigned device.	ctcAbTransferCall
Merge two or more calls into a single conference call.	ctcAbConferenceJoin
Cancel a consultation call and retrieve the held call.	ctcAbRetrieveHeld
Provide call path information for call routing.	ctcAbDeflectCall
Allow a virtual party on a switch to initiate calls on behalf of a user. Only when the called device answers, (or, for example, the telephone rings a preconfigured number of times) does the call get put through to the user.	ctcMakePredictiveCall

For more information about these routines, refer to Chapter 2.

1.3 Format of Routines

Each routine description in Chapter 2 shows the format of the routine written in C. The descriptions also provide details of any message returned by the Application Bridge, and include a summary of the arguments passed to the routine.

Arguments passed to a routine must be listed in your program in the same order as that shown in the format section.

1.3.1 Unsigned Integers and Windows 3.1/3.11

With the exception of ctcErrMsg, the format section for each routine in Chapter 2 shows the routine status return as a 32-bit unsigned integer. On Windows 3.1/3.11 this is the equivalent of an unsigned longword, but, for simplicity, Chapter 2 shows status returns and arguments as unsigned integers only.

If you are writing a Windows 3.1/3.11 program, use unsigned longwords wherever the format section or argument for a routine requires a 32-bit unsigned integer.

1.4 Application Bridge Messages

For some routines, CTC/AB passes message requests to the Application Bridge. Each routine description includes a section that describes any messages that the Application Bridge returns in response to a request.

For example, when you use ctcAbAssign to assign a channel to a TeleSet, CTC/AB sends an Equipment Status Request to the Application Bridge to check that the TeleSet equipment number is valid and logged in. The Application Bridge returns an Equipment Status Request Response message to verify that the TeleSet is valid. Details of this message are included in the description of ctcAbAssign.

For more information about Application Bridge messages, refer to your Aspect Application Bridge documentation.

1.5 Use of Arguments

The Arguments section of a routine description describes the use of each argument. Each argument has three characteristics: data type, access type, and passing mechanism.

1.5.1 Data Type

When a calling program passes an argument to a CTC/AB routine, the routine expects the argument to be of a particular data type. The *type* entry indicates the type of data used for an argument. This can be:

- Byte (unsigned)—8 bits
- Word (unsigned)—16 bits
- Integer (unsigned)—32 bits
- Character string (unsigned)
- Structure

- ctcChanId

The structure and ctcChanId data types are described in the following subsections.

Data Structures

Some CTC/AB arguments are addresses of data structures. A data structure is a block of memory that contains a series of fields of predefined offsets. Each of these structures has a fixed format defined in a CTC/AB definitions file installed on your system (see Section 1.6 for more information about definitions files).

There are two types of structure:

- An input structure requires the application to pass information to the CTC/AB API for one or more of the defined fields. For example, to create a channel to a TeleSet, the ctcAbAssign routine requires the application to provide the equipment number for the TeleSet. CTC/AB has read-only access to the content of an input structure.
- Output structures are used to provide the application with information. The application program passes to CTC/AB the address of a block of memory for the structure. CTC/AB writes information into the structure for the application to read. CTC/AB has write-only access to the content of an output structure.

ctcChanId Data

The ctcChanId datatype is defined in one of the CTC/AB definitions files installed on your system (see Section 1.6). It contains the channel identifier returned by the ctcAbAssign routine for the device in use.

1.5.2 Access to Data

The *access* entry indicates whether the CTC/AB routine:

- Reads data passed to it by the application (read only)
- Returns data to the application (write only)
- Reads data from the application and returns data to the application (read and write)

1.5.3 Passing Mechanism

The *mechanism* entry indicates whether the application passes data to the CTC/AB routine by value or by reference:

- **By Value**

  When your program passes an argument by value, the argument entry contains the actual, uninterpreted value of the argument. The by value mechanism is usually used to pass constants. For example, to pass the constant 100 by value, the calling program puts 100 directly into the argument list.

- **By Reference**

  When your program passes an argument by reference, the argument entry contains the address of the location that contains the value of the argument. For example, if variable *x* is allocated at location 2000, which currently contains the value 100, the argument entry will contain 2000.

1.5.4 Optional Arguments

Some routine arguments are "optional". This means that you still include the argument in your program but, depending on the passing mechanism, you can specify the value zero (0) or the address of a zero-length character string with the argument instead of data.

The way that you use an optional argument depends on the passing mechanism for the argument:

- If the argument is passed by value, specify zero (0) instead of the described value.
- If the argument is passed by reference and provides input to CTC/AB, specify the address of a null data type, for example, a zero-length character string.
- If the argument is passed by reference and obtains output from CTC/AB, supply enough memory to accommodate that argument's output.

To find out if an argument is optional, refer to the routine descriptions in Chapter 2.

1.6 Definitions

During the CTC/AB API installation procedure, CTC/AB copies a definitions file, ctabdef.h, to your system for you to include in your C program. This file includes the following definitions files for condition values, constants, and data structures:

Definitions	Files	
Condition values for status returns	ctaberr.h, ctc_err.h	
Constants	ctabcod.h, ctc_code.h	
Data Structures	ctabrpc.h, ctc_rpc.h	(Windows NT, Windows 95, Digital UNIX, HP-UX, SCO OpenServer, OpenVMS, OS/2)
	ctabw16.h	(Windows 3.1/3.11)

The file ctabdef.h and the definition files that it includes are copied to the directory specified during installation. For more information, refer to the *CT-Connect for Aspect Installation Guide*.

1.7 Constants

CTC/AB constants have one of the following prefixes:

This prefix...	Is used for...
ctcK_	Literals. For example, the value ctcK_AgentReady can be specified with the agentMode argument for the ctcAbSetAgentStatus routine to indicate that the agent is ready to take calls.
ctcM_	Masks. These are used to indicate whether an option (such as a function or event) is supported or used. For example, the value ctcM_Assign can be returned in the procedureSupport field of the ctcAbChanData structure. (See the description of the ctcAbGetChannelInformation routine for more information.)

1.8 Errors and Condition Values

Each routine returns a condition value (32-bit unsigned integer) as a completion code to indicate whether the call to the routine has been successful or whether an error has occurred.

Dialogic recommends that you always check the return status to determine success or failure of calls to CTC/AB routines, and choose a suitable recovery path if there is an error.

For an explanatory list of the condition values and errors that can be returned by CTC/AB routines, refer to Chapter 3.

1.8.1 Link Problems

If the link between the Aspect Aspect CallCenter and CTC/AB server fails, the CTC/AB server:

- Returns a ctcLinkDown or ctcLinkReset condition value.
- Clears all monitors and cancels any outstanding ctcAbGetEvent or ctcAbWinGetEvent requests.
- Deassigns all channels to devices.

1.9 Exception Handling

Any severe network problem that affects communication between the CTC/AB client and CTC/AB server may result in a software exception. If you want to handle this type of exception, refer to the Remote Procedure Call (RPC) programming documentation for your operating system.

1.10 Calling CTC/AB Routines

All CTC/AB routines operate synchronously. This means that they return to the caller only when the operation is complete.

Waiting for each operation to complete may be inappropriate for your application. For example, your application can use the ctcAbGetEvent routine to return information on telephone calls associated with the assigned device. This routine does not complete until there is activity on the assigned device. For your application to continue with operations, you must call the routines in a multithreaded program.

Multithreaded programming enables routines to be processed concurrently rather than in sequence. This means that applications are not blocked as they wait for operations to complete; operations that are asynchronous in nature can be performed in parallel with operations that are synchronous.

1.11 CTC/AB and Multithreaded Programming

This section provides an overview of threads and multithreaded programs for applications that require both synchronous and asynchronous operations.

Note that you do not need to create a multithreaded program if:

- Your application uses only synchronous operations.
- You are writing a CTC/AB application on a system running Windows 3.1/3.11 or Windows for Workgroups.

  These CTC/AB clients use a Windows Socket interface for CTC/AB API calls (all other CTC/AB client systems use Distributed Computing Environment (DCE) RPC services). Refer to Section 1.12 for more information.

1.11.1 Threads

A thread is a separate, sequential flow of control within a program. It is the movement of a processor through a program's instructions.

1.11.2 Multithreaded Programming

Most traditional programs consist of a single thread. In a multithreaded program, multiple threads are created to execute different parts of a program. This enables a program to overlap activities.

Threads in a multithreaded program share the address space, memory (except for stacks and register contents), and other resources provided by a single process. When the process is created, a single thread is created and used by the program. This is the main thread. From this thread, the program can create another thread, for example, for an operation that needs to wait for input from another device. It continues to perform more immediate work using the main thread.

If the program has a number of slow operations to perform, it can create additional threads from the main thread as they are required.

1.11.3 Thread Execution

A processor executes a thread until the thread has to wait for a resource to become available, for example, or for synchronization with another thread. At this point, the processor starts to run another thread. The processor continues in this way, executing one thread and then another.

No complicated data-passing mechanisms are required for one thread to communicate with another thread. A thread writes its output to memory and another thread can read it as input. When one thread has completed a task, it uses an indication mechanism (for example, a conditional variable) to let the other thread know that the input data is ready.

1.11.4 Using Multithreaded Programming with CTC/AB

Using multithreaded programming, a CTC/AB application can complete both of the following activities:

- It can use the main thread (the thread created at the same time as the process) for all synchronous operations. For example, calling the ctcAbMakeCall routine.
- It can create another thread for monitoring the device. For example, for calling the ctcAbGetEvent routine which returns information only when there is call activity. Dialogic recommends that you create a separate thread for this routine.

An online programming example (CTABCP.EXP) is provided as part of the CTC/AB API kit. This example shows how multithreaded programming is used. For details of its location, refer to the *CT-Connect for Aspect Installation Guide*.

1.11.5 Creating a Multithreaded Program

The procedure for creating threads in your program depends on the operating system you are using. For some operating systems, you may need to obtain a threads package.

For information about creating and using threads, refer to the application development documentation for your operating system:

- On a Windows NT or Windows 95 system, refer to the development documentation provided with your system. Threads are provided as part of the operating system for these platforms.
- On Digital UNIX and OpenVMS systems, you can use the DCE Thread Library routines. These routines are described in the *Digital DCE Application Development Reference* manual.
- On HP-UX systems, you can use DCE Thread Library routines. For more information, refer to the documentation provided with the HP® DCE Runtime Services software.
- On SCO OpenServer systems, you can use DCE Thread Library routines. For more information, refer to the documentation provided with the SCO® DCE Executive software.

- On OS/2 systems, you can use DCE Thread Library routines. For more information, refer to the documentation provided with the IBM® DCE for OS/2 WARP software.

1.12 Using the CTC/AB Windows Socket Interface

The CTC/AB Windows Socket interface is installed with the CTC/AB API software on systems running Windows 3.1/3.11. It enables CTC/AB applications running on these systems to use ctcAbWinGetEvent, a non-blocking routine that returns information for the assigned device. See the description of ctcAbWinGetEvent for more information.

Note that the ctcAbWinGetEvent routine is available for systems running Windows 3.1/3.11 or Windows for Workgroups only. If you are writing a Windows NT or Windows 95 application, call ctcAbGetEvent in a multithreaded program.

1.13 Compiling and Linking Your Program

Sections 1.13.1 to 1.13.5 contain platform-specific information about compiling and linking your program.

1.13.1 Windows NT and Windows 95 Clients

Before you compile and link your program on a Windows NT or Windows 95 system, note the information in the following sections.

Multithreaded Programs and Thread Stack Size

If you create threads for the ctcAbGetEvent routine in your program, note the following:

- On Windows NT systems, Dialogic recommends that you use a thread stack size of no more than 64 Kbytes when you link your program. The default thread stack size on Windows NT systems is 1 Mbyte.
- On Windows 95 systems, if you encounter problems with virtual memory, try reducing the thread stack size when you link your program. For more information, refer to your Windows 95 documentation.

Paths

During the CTC/AB API installation, the following paths are added to the AUTOEXEC.BAT file on your PC:

drive:*directory*\lib
drive:*directory*\include
drive:*directory*\bin

where *drive*:*directory* is the drive and directory used for the CTC/AB API installation. By default, this is C:\DIALOGIC\CTCAB.

These paths define the location of the CTC/AB API library and definitions files.

1.13.2 Windows 3.1/3.11 Client

Dialogic recommends that when you compile a CTC/AB program on a system running Windows 3.1/3.11, you:

- Check that the following header files are copied to your INCLUDE directory:

  CTABDEF.H
  CTABCOD.H
  CTABW16.H
  CTABERR.H
  CTC_CODE.H
  CTC_ERR.H

- Use the large memory model.

You can link your program using one of the following methods:

- Implicit Import—This gives you access to all the CTC/AB routines by including the import library in the linker command.
- Dynamic Run-Time Import—This allows access to only the routines you specify within your application code.

Implicit Import

To link your application with the CTC/AB API copy the import library CTABAPI.LIB to your library directory and include it in the linker command file. For example:

```
link /NOD/CO ctcapp.obj,,ctcapp.map/map,libw llibcew
ctabapi.lib,ctcapp.def
```

Dynamic Run-Time Import

Dynamic Run-Time Import eliminates the need for you to link your program with the CTC/AB import library. Your application first loads the import library, and then retrieves the address of the CTC/AB functions you specify.

For example, to call the ctcAbHangupCall routine:

```
HANDLE hLibrary;
FARPROC lpFunc;

hLibrary = LoadLibrary ("CTABAPI.DLL");
if (hLibrary >= 32)
        {
         lpFunc = GetProcAddress (hLibrary, "ctcAbHangupCall");
         if (lpFunc !=(FARPROC)NULL)
                 (*lpFunc) (hChan);
         FreeLibrary (hLibrary);
         };
```

1.13.3 Digital UNIX Client

On a CTC/AB client running Digital UNIX, the CTC/AB API is provided as the shareable object, /usr/shlib/libctabapi.so. You include this shareable object as input when you link your program to create an executable image.

To compile your program, you use the cc -c command. For example:

```
#cc -c ctabprog.c
```

where *ctabprog.c* is source code written in C.

To link your program, you use the cc command and -l to specify the shareable object, dce, pthreads, c_r, and mach objects. For example:

```
# cc -o ctabprog ctabprog.o -lctabapi -ldce -lpthreads -lc_r -lmach
```

where *ctabprog* is the executable image and *ctabprog.o* is the compiled program.

1.13.4 HP-UX Client

On a CTC/AB client running HP-UX, the CTC/AB API is provided as the shareable object, libctabapi.sl. You include this shareable object as input when you link your program to create an executable image.

For example, to compile a program written in C, you use:

```
#cc -c -o ctabprog.o -Ae +O4 -I/usr/include/reentrant \
-D_REENTRANT ctabprog.c
```

where *ctabprog.o* is the name you give to the output (the compiled program) and *ctcabprog.c* is source code written in C.

To link a program written in C, you use:

```
# ld ctcabprog.o /lib/crt0.o -o ctabapp -s -Bimmediate -Bnonfatal -lctabapi \
-ldce -lm -lc_r
```

where *ctabprog.o* is the compiled program and *ctabapp* is the executable image.

1.13.5 SCO OpenServer Client

To compile your program on a CTC/AB client running SCO OpenServer, you use:

```
cc -c -o ctabprog.o -belf ctabprog.c
```

where *ctabprog.o* is the name you give to the output (the compiled program) and *ctabprog.c* is source code written in C.

To link your program, you use:

```
#ld ctabprog.o /lib/crt0.o -o ctabapp -s -lctabapi -ldce -lcma -lm \
-lsocket -lc
```

where *ctabprog.o* is the compiled program and *ctabapp* is the executable image.

1.13.6 OpenVMS Client

On a CTC/AB client running OpenVMS, the CTC/AB API is provided as the shareable image, SYS$SHARE:CTABAPI.EXE. You include this shareable image as input to the linker.

Compile your program in the usual way and then complete the following procedure to link your image:

1. Create an options file that contains the following:

   ```
   SYS$SHARE:CTABAPI.EXE/SHAREABLE
   ```

   Identify the Run-Time Library shareable image in the options file. For example:

   ```
   SYS$SHARE:CTABAPI.EXE/SHAREABLE
   SYS$SHARE:VAXCRTL.EXE/SHAREABLE
   ```

   where VAXCRTL.EXE is the shareable image for the VAX C Run-Time Library.

   For more information, refer to your programming utilities documentation.

2. Use the LINK command to link your image:

   ```
   $ LINK ctab_program, filename.OPT/OPTION, DCE:DCE.OPT/OPTION
   ```

   where *ctab_program* is the name of your compiled program and *filename*.OPT is the name of your options file. DCE.OPT is the DCE options file.

1.13.7 OS/2 Client

The following is an example command used to compile a program on a CTC/AB client running OS/2 WARP Connect:

```
icc ctabprog.c /Gm+ /Su4 /Ms /C+ /Sem -D_CMA_PROTO_ -D_CMA_NOWRAPPERS_
-DCMA_UNIPROCESSOR -DINTEL80x86 -DIBMOS2
```

where *ctabprog.c* is source code written in C. This produces a compiled program ctabprog.obj.

The following example shows how to link the CTC/CMP program:

```
ilink ctabprog.obj /E /NOI /NOE /ST:100000 /O:ctabprog ctabapi.lib
dceos2.lib os2386.lib
```

where *ctabprog.obj* is the compiled program and *ctabprog* is the executable image.

2
Routine Specifications

This chapter provides detailed specifications of the CTC/AB routines in alphabetical order.

2.1 Aspect Application Bridge Messages

The descriptions of the routines in this chapter indicate how to invoke the Aspect Application Bridge messages and responses through CTC/AB. For full details of the Aspect Application Bridge and the Aspect Call Control Table (CCT) facility, refer to your Aspect Application Bridge documentation.

2.2 Application Bridge Software Releases

CTC/AB routines can be used with Aspect Application Bridge Release 5.0 and Release 6.0 **unless otherwise stated**. Any differences between releases that affect CTC/AB functionality are noted in this chapter.

ctcAbAddMonitor
Adds a Device to a Monitor Channel

Format in C

unsigned int ***ctcAbAddMonitor*** *(ctcChanId channel,*
*struct ctcAbAssignData *assignData)*

Description

The ctcAbAddMonitor routine sets monitoring on for a device and associates it with a monitor channel. A monitor channel is a single channel used to monitor multiple devices so that all event information is returned on the monitor channel.

You can monitor the following devices on a monitor channel:

- Station
- TeleSet
- ACD group
- Trunk
- Trunk group
- InterQueue point
- Monitor channel

Monitoring a Device on a Monitor Channel

To set up a monitor channel, you use the following routines:

1. ctcAbAssign to create a monitor channel
2. ctcAbAddMonitor for each device you want to monitor on the monitor channel
3. ctcAbGetEvent to return information on the monitor channel

Monitoring Another Monitor Channel

To monitor another monitor channel, use the ctcAbGetChannelInformation to obtain a device number for the monitor channel (returned in the setDN field of the ctcAbChanData structure) and specify this as the deviceDN with the assignData argument.

Note that:

- You can only use one level of nested monitoring for monitor channels. This means that you cannot monitor a monitor channel if that channel is alreading monitoring another monitor channel.
- A monitor channel cannot monitor itself.

Removing Monitoring for a Device

To stop monitoring a device on the monitor channel, you use ctcAbRemove-Monitor. For more information, see the description of the ctcAbRemoveMonitor routine.

Restrictions

The following restrictions apply for ctcAbAddMonitor:

- This routine is supported for channels assigned to monitor channels only.
- To monitor ACD groups and trunk groups on a monitor channel, Aspect Application Bridge Release 6.0 is required.
- ctcAbAddMonitor is not supported on CTC/AB clients running Windows 3.1/3.11. CTC/AB applications running on Windows 3.1/3.11 cannot assign to monitor channels.
- If you are monitoring a number of devices on the monitor channel and activity on each device is high, a number of events may occur at the same time. In this situation, it is possible for the CTC/AB server to lose an event message and return a ctcEventDataLost error.

  To avoid losing event data when monitoring high-activity devices, Dialogic recommends you create more than one monitor channel.

Aspect Application Bridge Message

None.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the monitor channel.

You use this argument to identify the monitor channel that will be used for monitoring the device.

The ctcChanId datatype is defined in a CTC/AB definitions file (see Section 1.6).

assignData

type: **structure**
access: **read only**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type ctcAbAssignData. The structure is defined in a CTC/AB definitions file (see Section 1.6) and is formatted as follows:

```
    struct ctcAbAssignData  {
       unsigned short      deviceType;
       unsigned char       APIversion;
       unsigned char       spare;
       unsigned char       deviceDN [ctcMaxDnLen];
  }
```

The following sections describe the ctcAbAssignData fields.

deviceType

This 16-bit field identifies the type of device you are assigning to the monitor channel. Specify one of the values in the following table:

Specify this value...	To monitor...
ctcK_Station	A station
ctcK_Teleset	A TeleSet
ctcK_AcdGroup	An ACD group†
ctcK_Trunk	A trunk
ctcK_TrunkGroup	A trunk group†
ctcK_InterQueue	An InterQueue point
ctcK_MonitorChannel	A monitor channel

†Aspect Application Bridge Release 6.0 is required for this device type.

APIversion

This 8-bit field identifies the version of CTC/AB API software used. ctcAbAddMonitor does not use the information in this field. Specify the value zero.

deviceDN

This 24-byte field identifies the device to be monitored on the monitor channel. Specify one of the following:

For this type of device...	Specify...
Station	Equipment number
TeleSet	Equipment number
Trunk	Trunk number
ACD group	Group number
Trunk group	Group number
InterQueue point	InterQueue request number
Monitor channel	The setDN value returned by the routine ctcAbGetChannelInformation. See the description of this routine for more information.

This ASCII string that can contain any combination of numbers 0 through 9 and the characters * and #. The maximum length for deviceDN is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

ctcAbAnswerCall
Answer a Call

Format in C

unsigned int ***ctcAbAnswerCall*** *(ctcChanId channel,*
unsigned int callRefId)

Description

When CTC/AB notifies your application that there is an incoming call on the assigned TeleSet, you can use the ctcAbAnswerCall routine to answer that call.

CTC/AB notifies you of an incoming call only if both of the following conditions apply:

1. You have set monitoring on, using ctcAbSetMonitor.
2. You are using the ctcAbGetEvent or ctcAbWinGetEvent routine, which indicates a change in state to receive (ringing).

The call reference identifier for the incoming call is returned by ctcAbGetEvent or ctcAbWinGetEvent.

When the call is put through, the agent can speak to the caller using hands-free operation on a speakerphone TeleSet. ctcAbAnswerCall cannot be used with standard telephones.

Restrictions

The following restrictions apply:

- This routine is supported for channels assigned to TeleSets only.
- Aspect Application Bridge Release 6.0 is required to support this routine.

Aspect Application Bridge Message

Answer Call Request (ACR)

CTC/AB sends an ACR to request that the Aspect CallCenter answers the call associated with the specified TeleSet. For full details of this message, refer to the Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

The ctcChanId datatype is defined in a CTC/AB definitions file (see Section 1.6).

callRefId

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This argument is a 32-bit integer that contains the call reference identifier for the incoming call. Specify the call reference identifier returned by ctcAbGetEvent or ctcAbWinGetEvent for the incoming call.

ctcAbAssign
Assign a Channel

Format in C

```
unsigned int ctcAbAssign
                    (ctcChanId                *channel,
                    struct  ctcAbAssignData   *assignData,
                    unsigned char             serverName [ctcNodeNameLen],
                    unsigned char             logicalIdentifier [ctcLogIdLen],
                    unsigned char             networkType [ctcNetLen])
```

Description

Before a device (for example, a TeleSet next to a user's PC) can be linked to a CTC/AB network, the device and the communications channel must be uniquely identified to CTC/AB by your application.

The ctcAbAssign routine assigns a logical communications channel between the application, the CTC/AB server, and the specified device, and then returns an identifier (ID) for that channel.

Use ctcAbAssign to assign a channel to the following:

- Station (administrative telephone, or auxiliary device such as an Interactive Voice Response (IVR) unit)
- TeleSet
- ACD group (also known as agent group or queue)
- Trunk
- Trunk group
- InterQueue point
- Monitor channel

When to Use ctcAbAssign

You must use the ctcAbAssign routine before any of the other CTC/AB routines so that you know the channel ID associated with a device. All subsequent routines that you invoke for the device require you to specify that channel ID.

You need assign a channel only once for each user session; that is, you do not have to assign and deassign the channel for each telephone call a user makes from a particular TeleSet.

Assigning Channels to ACD Groups or Trunk Groups

Application Bridge Release 6.0 provides additional event information for TeleSets and trunks (see Tables 2–2, 2–3, and 2–4). Assigning a channel to an ACD group or trunk group enables your application to receive this information.

For example, to receive Application Bridge Release 6.0 events for the TeleSets in an ACD group, you:

1. Assign a channel to the ACD group
2. Assign a channel to each TeleSet in the ACD group

You need assign to the ACD group only once for each group of TeleSets. The additional events are returned on the channels to the TeleSets.

InterQueue Points

If your Aspect CallCenter is part of a network of CallCenters, you can monitor calls that it receives from another CallCenter by assigning a channel to an InterQueue.

A Network InterQueue is a virtual channel used to notify a CallCenter that it will receive a call from another CallCenter in the network. When the target CallCenter receives this notification, it processes it as if the call had already been presented. For example, it can respond with an Application Bridge Call Information Message (CIM) or Call Track Information Message (CTIM). At the same time, it negotiates receipt of the call over a real trunk.

When negotiation is complete, the call is sent over the trunk and, depending on the Call Control Table (CCT) used to process the call, another CIM or CTIM may be generated. If not, it may appear that only a Call Connect Message (CCM) has been generated for the call on that trunk.

Monitor Channels

If you need to monitor a number of devices, you can use ctcAbAssign to create a single monitor channel that receives all events for the devices. For example, your application can create one channel to receive event information for all TeleSets in an office, building, or for a particular group.

To set up a monitor channel, you use the following sequence of routines:

1. ctcAbAssign to assign a monitor channel
2. ctcAbAddMonitor for each device you want to monitor on the monitor channel
3. ctcAbGetEvent to return information for all devices associated with the monitor channel

Note that you do not need to use ctcAbSetMonitor in this sequence to enable monitoring for the device; when you use ctcAbAddMonitor, monitoring is automatically enabled.

Supported Devices and Routines

Table 2–1 shows which routines are supported by each type of device.

Note that a user process is not given exclusive access to a device; it is possible for more than one channel to be assigned to the same device.

Restrictions

To assign a channel to a trunk group or ACD group, Aspect Application Bridge Release 6.0 is required.

Aspect Application Bridge Messages

Equipment Status Request (ESR)
Event Monitor Request (EMR)

- If you assign to a station, TeleSet, or trunk, CTC/AB sends an ESR to the Application Bridge. An Equipment Status Request Response (ESRR) is returned by the Application Bridge to verify that the specified number is valid. The application is then free to call other routines.
- If you assign to an ACD group or trunk group, CTC/AB sends an EMR to the Application Bridge.
- If you assign to an InterQueue point or monitor channel, no Application Bridge message is returned.

For full details of the ESR and EMR messages, refer to the Aspect Application Bridge documentation.

Table 2–1 Routines Supported by Device Type

	Station	TeleSet	ACD Group	Trunk	Trunk Group	InterQueue	Monitor Channel
ctcAbAddMonitor							X
ctcAbAnswerCall		X					
ctcAbAssign	X	X	X	X	X	X	X
ctcAbConferenceJoin		X					
ctcAbConsultationCall		X					
ctcAbDeassign	X	X	X	X	X	X	X
ctcAbDeflectCall		X		X			
ctcAbErrMsg	X	X	X	X	X	X	X
ctcAbGetChannelInformation	X	X	X	X	X	X	X
ctcAbGetEvent	X	X	X	X	X	X	X
ctcAbGetMonitor	X	X	X	X	X	X	
ctcAbHangupCall	X	X		X			
ctcAbHoldCall		X					
ctcAbMakeCall	X	X					
ctcAbMakePredictiveCall			X				
ctcAbReassignResource		X					
ctcAbRemoveMonitor							X
ctcAbRetrieveHeld		X					
ctcAbSetAgentStatus		X					
ctcAbSetMonitor	X	X	X	X	X	X	
ctcAbSingleStepTransfer	X	X		X			
ctcAbTransferCall		X					
ctcAbWinGetEvent	X	X	X	X	X	X	

Arguments

channel

type: **ctcChanId**
access: **write only**
mechanism: **by reference**

This argument is a pointer that receives the address of a ctcChanId datatype. The ctcChanId datatype is defined in a CTC/AB definitions file (see Section 1.6).

The channel ID is the value used by the other CTC/AB routines to identify the device they are using.

assignData

type: **structure**
access: **read only**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type ctcAbAssignData. The structure is defined in a CTC/AB definitions file (see Section 1.6) and is formatted as follows:

```
   struct ctcAbAssignData  {
      unsigned short      deviceType;
      unsigned char       APIversion;
      unsigned char       spare;
      unsigned char       deviceDN [ctcMaxDnLen];
 }
```

The following sections describe the ctcAbAssignData fields.

deviceType

This 16-bit field identifies the type of device to which the channel is assigned. Specify one of the values in the following table:

Specify this value...	To assign a channel to...
ctcK_Station	A station
ctcK_Teleset	A TeleSet
ctcK_AcdGroup	An ACD group†
ctcK_Trunk	A trunk
ctcK_TrunkGroup	A trunk group†
ctcK_InterQueue	An InterQueue point
ctcK_MonitorChannel	A monitor channel

†Aspect Application Bridge Release 6.0 is required to support this device type.

APIversion

This 8-bit field identifies the version of the CTC/AB API software used. This ensures compatibility with previous and new versions of the CTC/AB software when you compile your application. Specify one of the following values in the APIversion field:

Value	Description
ctcK_AbCtcV11	Specify this value if you are writing a CTC/AB application for use only with Version 1.1 of the CTC /AB API software.
ctcK_AbCurrentVersion	Specify this value and your application will be compatible with the current version of the CTC/AB API installed on your CTC/AB client system. When you upgrade to a future version of the CTC/AB API, your application will automatically gain access to any new events provided as part of that version.

To ensure future compatibility, Dialogic recommends you use ctcK_AbCurrentVersion.

ctcAbAssign

deviceDN

Use this 24-byte field to specify the telephone number for the device.

For this type of device...	Specify...
Station	Equipment number
TeleSet	Equipment number
Trunk	Trunk number
ACD group	Group number
Trunk group	Trunk number
InterQueue point	InterQueue request number
Monitor channel	The address of a zero-length character string.

This ASCII string that can contain any combination of numbers 0 through 9 and the characters * and #. The maximum length for deviceDN is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6).

Note that if the number is longer than ctcMaxDnLen (excluding the null termination character (NUL)), CTC/AB returns a ctcInvDevice condition value.

serverName

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

This argument is the address of a character string that identifies the CTC/AB server. The string can contain the name or address for the CTC/AB server.

For example, on a CTC/AB client running Windows 3.1/3.11, this string can contain the TCP/IP host name or IP address for the CTC/AB server. On an OpenVMS client, you can specify the TCP/IP host name, node name, IP address, or DECnet address for the CTC/AB server.

The maximum length for serverName is specified by the literal ctcNodeName-Len in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

logicalIdentifier

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

This argument is the address of a character string that contains an identifier for the link. The identifier is assigned to the link at the CTC/AB server and is defined during the installation of the CTC Server for Aspect software, or after the installation with the Configuration Program or Control Program. For details of these programs, see the *CT-Connect for Aspect Management Guide*.

You must supply the logical identifier exactly as it appears on the CTC/AB server, including the same combination of uppercase and/or lowercase letters.

The maximum length for logicalIdentifier is specified by the literal ctcLogIdLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

networkType

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

This argument is the address of a character string value that identifies the network protocol used between the CTC/AB client and the CTC/AB server.

Check with the system manager of your CTC/AB network for details of the network protocol, and specify one of the values in the following table:

Network Protocol	Value
NetBIOS™ over NetBEUI	ncacn_nb_nb
TCP/IP	ncacn_ip_tcp
DECnet	ncacn_dnet_nsp
NetBIOS over TCP/IP	ncacn_nb_tcp
Named pipes	ncacn_np
Novell® SPX	ncacn_spx
Local RPC†	ncalrpc

†This protocol can be used only if the CTC/AB server and CTC/AB client are the same Windows NT PC.

The maximum length for the networkType value is specified by the literal ctcNetLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

ctcAbConferenceJoin
Merge Calls into a Conference

Format in C

unsigned int ***ctcAbConferenceJoin*** *(ctcChanId channel)*

Description

The ctcAbConferenceJoin routine merges two or more calls into a single conference call.

For example, for A to include B and C in a conference call:

1. A calls B, using ctcAbMakeCall. B answers the call.
2. A places B on hold, using ctcAbHoldCall.
3. A calls C, using ctcAbConsultationCall.
4. When connected and talking to C, A creates the conference call using ctcAbConferenceJoin. A, B, and C are included in the conference call.

A maximum of three parties can be included in a conference call.

Restrictions

This routine is supported for channels assigned to TeleSets only.

Aspect Application Bridge Message

Process Key Request (PKR)

CTC/AB sends a Process Key Request to the Application Bridge to simulate the user pressing the TeleSet conference key to complete the conference call. The Application Bridge returns a Process Key Request Response (PKRR) message to verify that the parties are merged in a conference call. The application is then free to call other routines.

For full details of this message, refer to the Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the assigned device in use.

ctcAbConsultationCall
Make a Consultation Call

Format in C

```
unsigned int ctcAbConsultationCall
                    (ctcChanId              channel,
                    unsigned char           calledNumber [ctcMaxDnLen],
                    unsigned int            *callRefId,
                    struct   ctcAbVarData   *varData)
```

Description

The ctcAbConsultationCall routine makes a call to a third party when there is a call on hold at the assigned TeleSet. You can then use one of the following routines:

- ctcAbTransferCall to transfer the call and disconnect the assigned TeleSet
- ctcAbConferenceJoin to join the held call and the call to the third party into a conference call

Transferring a Call

To transfer a call, use ctcAbConsultationCall followed by ctcAbTransferCall.

For example, for A to transfer to C an incoming call from B (where A's current call is the call from B):

1. B calls A, using ctcAbMakeCall, and A answers.
2. A places B on hold using ctcAbHoldCall.
3. A calls C, using ctcAbConsultationCall.
4. A invokes ctcAbTransferCall when connected to C. B and C are now connected, and A is automatically disconnected.

Making a Conference Call

For a conference call, you use ctcAbConsultationCall only for the second call, or subsequent calls; you make the initial call using ctcAbMakeCall.

For example, for A to include B and C in a conference call:

1. A calls B, using ctcAbMakeCall.
2. A places B on hold with ctcAbHoldCall.
3. A calls C, using ctcAbConsultationCall.
4. A invokes ctcAbConferenceJoin when connected and talking to C. A, B, and C are then in a conference call.

A maximum of three parties can be included in a conference call.

Restrictions

This routine is supported for channels assigned to TeleSets only.

Aspect Application Bridge Message

Place Call Request

CTC/AB sends a Place Call Request (PCR) to the Aspect Application Bridge to make a consultation call. The Application Bridge returns a Place Call Request Response (PCRR) message to verify that the consultation call can be placed.

For full details of this message, refer to the Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

ctcAbConsultationCall

calledNumber

type: **character string (unsigned)**
access: **read only**
mechanism: **by value**

This character string contains the number of the device you are calling:

- To make a consultation call to another TeleSet or telephone, specify its telephone number. The number must be specified as an ASCII string which can contain any combination of numbers 0 through 9 and the characters * and #.
- To make a consultation call to a group associated with a CCT, specify an ASCII string containing #8 followed by the number for the CCT. For example, #8111 (where 111 is the number for the CCT).
- To make a consultation call to an agent through a CCT, specify an ASCII string containing #8 followed by the number for the CCT. Then, specify the number for the agent in one of the variable data fields in the ctcAbVarData structure. See the description of the varData argument.

  Note that the CCT must be set up so that it checks the correct variable data field for an agent number. See your Aspect CallCenter administrator for more information.

The maximum length for calledNumber is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

callRefId

type: **integer (unsigned)**
access: **write only**
mechanism: **by reference**

This argument is the address of a 32-bit integer that receives the call reference value for the call to the third party.

varData

type: **structure**
access: **read only**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type ctcAbVarData. The structure is defined in a CTC/AB definitions file (see Section 1.6) and is formatted as follows:

```
struct ctcAbVarData {
  unsigned char    varDataA [ctcMaxDataALen];
  unsigned char    varDataB [ctcMaxDataBLen];
  unsigned char    varDataC [ctcMaxDataCLen];
  unsigned char    varDataD [ctcMaxDataDLen];
  unsigned char    varDataE [ctcMaxDataELen];
}
```

The structure contains the following fields:

- **varDataA**

  The address of a character string that contains information corresponding to the Application Bridge variable field A. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataA is specified by the literal ctcMax-DataALen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataB**

  The address of a character string that contains information corresponding to the Application Bridge variable field B. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataB is specified by the literal ctcMax-DataBLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataC**

  The address of a character string that contains information corresponding to the Application Bridge variable field C. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataC is specified by the literal ctcMax-DataCLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataD**

  The address of a character string that contains information corresponding to the Application Bridge variable field D. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataD is specified by the literal ctcMaxDataDLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataE**

  The address of a character string that contains information corresponding to the Application Bridge variable field E. This ASCII string can contain any combination of alphanumeric characters.

  The maximum length for varDataE is specified by the literal ctcMaxDataE-Len in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

ctcAbDeassign
Deassign a Channel

Format in C

unsigned int ***ctcAbDeassign*** *(ctcChanId channel)*

Description

The ctcAbDeassign routine deassigns the channel from the device and frees all resources associated with it, both locally and on the CTC/AB server.

Use this routine at the end of a user session; that is, when the user has finished using a CTC/AB application and the application no longer needs to make use of the device to which the channel was assigned.

Monitoring is switched off before ctcAbDeassign completes. If you call ctcAbDeassign and there are outstanding ctcAbGetEvent or ctcAbWinGetEvent requests, a ctcMonitorOff condition value is returned.

Aspect Application Bridge Message

Event Monitor Request (EMR)

- If the channel is assigned to an ACD group or trunk group, CTC/AB sends an EMR to the Application Bridge.
- If the channel is assigned to a station, TeleSet, trunk, InterQueue point, or monitor channel, no Application Bridge message is returned.

For full details of the EMR message, refer to the Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

ctcAbDeflectCall
Deflect a Ringing Call

Format in C

```
unsigned int ctcAbDeflectCall
                (ctcChanId              channel,
                unsigned int            callRefId,
                unsigned char           cct [ctcMaxDnLen],
                unsigned int            aspectAck,
                struct  ctcAbVarData    *varData)
```

Description

The ctcAbDeflectCall routine responds to an Aspect Application Bridge Call Information Message (CIM) or Call Track Information Message (CTIM) received as an event. ctcAbDeflectCall causes the CTC/AB server to send a Call Track Information Message Response (CTIMR) to the Aspect Call Control Table (CCT) processing the call. Providing call path information in this way is often referred to as call routing.

The CCT currently processing the call must contain a RECEIVE DATA step to receive the CTIMR from the CTC/AB server. The CCT can then route the call another CCT, which is identified by the cct argument.

For a detailed description of the CCT mechanism, refer to the Aspect Application Bridge documentation.

Restrictions

This routine is supported for channels assigned to TeleSets and trunks only.

Aspect Application Bridge Message

Call Track Information Message Response (CTIMR)

The CTC/AB server receives a Call Information Messaged (CIM) or Call Track Information Message (CTIM) as an event from the Aspect Application Bridge. Note that if a CTIM is sent to the CTC/AB server, the REQUEST field must contain the value 0 (no request).

The CTC/AB server responds by sending a CTIMR. The CCT must contain a RECEIVE DATA step to receive this response.

For full details of these messages, refer to the Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

callRefId

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This argument is a 32-bit integer that contains the call identifier value for the ringing call. The call identifier value is returned by the ctcAbGetEvent routine.

cct

type: **character string (unsigned)**
access: **read only**
mechanism: **by value**

This character string contains the number of the CCT to which the call will be deflected. The ASCII string can contain any combination of numbers 000 through 999 and the characters * and #. Specify 000 if you want to use a default CCT as defined by the Aspect CallCenter administrator.

The maximum length for cct is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

aspectAck

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This argument specifies the acknowledgement to be used in the RESP field of the Aspect Call Track Information Message Response. Specify one of the following values:

Value	Acknowledgement
ctcK_AbAck	Conveys a positive acknowledgement.
ctcK_AbNack	Conveys a negative acknowledgement.

varData

type: **structure**
access: **read and write**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type ctcAbVarData. The structure is defined in a CTC/AB definitions file (see Section 1.6) and is formatted as follows:

```
struct ctcAbVarData {
  unsigned char    varDataA [ctcMaxDataALen];
  unsigned char    varDataB [ctcMaxDataBLen];
  unsigned char    varDataC [ctcMaxDataCLen];
  unsigned char    varDataD [ctcMaxDataDLen];
  unsigned char    varDataE [ctcMaxDataELen];
}
```

The structure contains the following fields:

- **varDataA**

  This field contains information corresponding to the Application Bridge variable field A. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataA is specified by the literal ctcMax-DataALen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataB**

  This field contains information corresponding to the Application Bridge variable field B. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataB is specified by the literal ctcMax-DataBLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataC**

  This field contains information corresponding to the Application Bridge variable field C. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataC is specified by the literal ctcMax-DataCLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataD**

  This field contains information corresponding to the Application Bridge variable field D. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataD is specified by the literal ctcMaxDataDLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataE**

  This field contains information corresponding to the Application Bridge variable field E. This ASCII string can contain any combination of alphanumeric characters.

  The maximum length for varDataE is specified by the literal ctcMaxDataE-Len in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

ctcAbErrMsg
Get the Defined Name for a Condition Value

Format in C

*char ****ctcAbErrMsg*** *(unsigned int errorCode)*

Description

The ctcAbErrMsg routine returns the address of a null-terminated character string that contains the defined name for a condition value.

Each condition value is associated with a name in a CTC/AB definitions file (ctaberr.h or ctc_err.h). You use the name:

- To establish the nature of the condition. For example, ctcInvAgentMode indicates that agentMode argument for ctcAbSetAgentStatus contains an invalid value.
- To refer to Chapter 3 which lists CTC/AB conditions alphabetically and describes possible causes for the error.

For example, if you specify the value 1014 with the errorCode argument, CTC/AB returns the address of a null-terminated character string that contains the name ctcInvLogId. You can then refer to Chapter 3 for a description of ctcInvLogId.

Aspect Application Bridge Message

None.

Arguments

errorCode

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This argument is a 32-bit integer that contains the condition value returned by a CTC/AB routine.

CTC/AB returns the address of a null-terminated character string that contains the name associated with the condition value that you specify.

Note that:

- If the routine cannot map a name to the condition value, it returns the address of a character string containing the decimal value of the input.
- If you specify a condition value for an RPC error, the character string contains:
  - The CTC/AB-defined name associated with the condition value
  - The RPC name associated with the condition value (with the prefix rpc_)

  For example, the routine can return the address of the character string ctcRpcConnecFail/rpc_s_ss_in_null_context. In this example, ctcRpc-ConnecFail is the CTC/AB-defined name and rpc_s_ss_in_null_context is the RPC name associated with the condition.

ctcAbGetChannelInformation
Get Information About a Channel

Format in C

unsigned int ***ctcAbGetChannelInformation***
(ctcChanId channel,
*struct ctcAbChanData *channelData)*

Description

The ctcAbGetChannelInformation routine returns information about the communications channel and the device to which the channel is assigned. The routine provides the following information:

- The line type (station, TeleSet, ACD group, trunk, trunk group, InterQueue point, or monitor channel).
- The CTC/AB procedures supported.
- For a station, its telephone number. When you assign to a station, you specify its equipment number.
- If the assigned device is a TeleSet, and, at the time of the assign, an agent was logged in to the TeleSet, its telephone number. When you assign to a TeleSet, you specify its equipment number.
- For a trunk, its trunk number.
- For an ACD group or trunk group, its group number.
- For an InterQueue point, its InterQueue number.
- If the assigned device is a monitor channel, a device number for the monitor channel. To receive events for the monitor channel on another monitor channel, you specify this device number with the ctcAbAddMonitor routine. See the description of ctcAbAddMonitor for more information.

You need to use ctcAbGetChannelInformation only once, each time you assign a channel to a device; it provides static information about the channel and the device to which it is assigned.

Aspect Application Bridge Message

None.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

channelData

type: **structure**
access: **write only**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type ctcAbChanData. The structure is defined in a CTC/AB definitions file (see Section 1.6) and is formatted as follows:

```
struct ctcAbChanData {
  unsigned int       lineType;
  unsigned int       procedureSupport;
  unsigned int       attributeSupport;
  unsigned char      setDN [ctcMaxDnLen];
  }
```

The following sections describe the fields in the ctcAbChanData structure.

ctcAbGetChannelInformation

lineType

This 32-bit integer contains a value that identifies the type of device associated with the line. The following table shows the values that can be returned.

This value...	Indicates that the channel is assigned to...
ctcK_Station	A station
ctcK_TeleSet	A TeleSet
ctcK_AcdGroup	An ACD group
ctcK_Trunk	A trunk
ctcK_TrunkGroup	A trunk group
ctcK_InterQueue	An InterQueue point
ctcK_MonitorChannel	A monitor channel

procedureSupport

This 32-bit integer identifies the procedure routines supported by the Aspect CallCenter. The following values can be returned:

ctcM_AddMonitor
ctcM_AnswerCall
ctcM_Assign
ctcM_ConferenceJoin
ctcM_ConsultationCall
ctcM_Deassign
ctcM_DeflectCall
ctcM_GetChannelInformation
ctcM_GetEvent
ctcM_HangupCall
ctcM_HoldCall
ctcM_MakeCall
ctcM_MakePredictiveCall
ctcM_ReassignResource
ctcM_RemoveMonitor
ctcM_RetrieveHeld
ctcM_SingleStepTransfer
ctcM_TransferCall

Note that:

- If ctcM_GetEvent is returned, both ctcAbGetEvent and ctcAbWinGetEvent are supported.
- A function-supported mask is not returned for ctcAbErrMsg. This routine is supported but, because its function is CTC/AB client-based and requires no interaction with the Aspect CallCenter, no value is returned.

attributeSupport

This 32-bit integer identifies the attribute routines supported by the Aspect CallCenter. Attribute routines are routines that set operating modes for the assigned TeleSet.

The following values can be returned:

ctcM_GetMonitor
ctcM_SetAgentStatus
ctcM_SetMonitor

setDN

This field can return the following:

- If the assigned device is a station, its telephone number.
- If the assigned device is a TeleSet, and, at the time of the assign, an agent was logged in at the TeleSet, its telephone number.
- If the assigned device is a trunk, its trunk number.
- If the channel is assigned to an ACD group or trunk group, the group number.
- If the channel is assigned to an InterQueue point, the InterQueue number.
- If the channel is assigned to a monitor channel, a device number. You can use this number to set up monitoring the monitor channel. See the description of ctcAbAddMonitor for more information.

The maximum length for setDN is specified by the literal ctcMaxDnLen, in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

ctcAbGetEvent
Get Information About Event and State Changes

Format in C

```
unsigned int ctcAbGetEvent (ctcChanId                channel,
                            struct  ctcAbEventData   *eventData,
                            unsigned int             dontWait)
```

Description

The ctcAbGetEvent routine returns details of telephone activity on the assigned device, or devices associated with a monitor channel.

The ctcAbGetEvent routine returns:

- Call references
- Call-tracking data corresponding to information in the track number and track node fields of Application Bridge messages
- Call events
- Agent events
- The other party involved in the telephone call
- Data corresponding to information in the variable data fields of Application Bridge messages
- Subtype data corresponding to information in the SUBTYPE field of Application Bridge messages
- Transferred call identifiers
- Statistics data corresponding to the RTIME, QTIME, and TTIME fields of Application Bridge messages

The amount of information that CTC/AB returns depends on the information provided by the Aspect CallCenter. This may be different for a call that is internal to the Aspect CallCenter and for an outside call, depending on the type of trunks connected to the Aspect CallCenter.

Calling ctcAbGetEvent

For all assigned devices **except monitor channels**, you must set monitoring on with the ctcAbSetMonitor routine before you use this routine.

Note that if you post a ctcAbGetEvent request and the previous ctcAbGetEvent request has not yet completed, a ctcEventInProgress error is returned.

Using ctcAbGetEvent With TeleSets and Trunks

For channels assigned to TeleSets or trunks, some event information is available with Application Bridge Release 6.0 only (see Tables 2–2, 2–3, and 2–4). To receive this information, a channel must be assigned to the ACD group or trunk group associated with the device before you assign to the device.

For example, to receive Application Bridge Release 6.0 events for the TeleSets in an ACD group, you:

1. Assign a channel to the ACD group
2. Assign a channel to each TeleSet in the ACD group

You do not need to assign more than one channel to the ACD group. CTC/AB returns the additional events on each channel assigned to a TeleSet in the group.

Using ctcAbGetEvent With Monitor Channels

When an event occurs for a device monitored on a monitor channel, the device can be identified by the DN returned in the monitorParty field of the ctcAbEventData structure. All other event information returned in the ctcAbEventData structure is associated with that device.

If you are monitoring another monitor channel, you can identify it by the number returned in the nestedMonitorChannel field of the ctcAbEventData structure.

After each event received on a monitor channel, you must repost the ctcAbGetEvent routine. If you are monitoring a number of high-activity devices, Dialogic recommends that you use more than one monitor channel to monitor these devices. Although the CTC/AB server stores up to 20 events, it is possible for it to lose event information when a number of events occur at the same time. If event data is lost, the CTC/AB server returns the condition value ctcEventDataLost.

Using ctcAbGetEvent With InterQueue Points

If your Aspect CallCenter is part of a network of CallCenters, you can monitor calls that it receives from another CallCenter by assigning a channel to an InterQueue.

A Network InterQueue is a virtual channel used to notify a CallCenter that it will receive a call from another CallCenter in the network. When the target CallCenter receives this notification, it processes it as if the call had already been presented. For example, it can respond with an Application Bridge Call Information Message (CIM) or Call Track Information Message (CTIM). At the same time, it negotiates receipt of the call over a real trunk.

When CTC/AB receives an Application Bridge message for a call associated with the InterQueue, it generates an event shown in Table 2–5. As soon as negotiation for a trunk is complete, the call is no longer associated with the InterQueue. Instead, CTC/AB receives Application Bridge messages for the call on the trunk and generates corresponding events. Table 2–3 shows CTC/AB events that are returned for trunks.

This means that to track the progress of the call, a channel must be assigned to both the InterQueue point and the trunk used for the call.

Restrictions

The following restrictions apply:

- ctcAbGetEvent returns information only when there is call activity. For this reason, Dialogic recommends you use a multithreaded program to call this routine so that your application can continue (see Section 1.11).

  This does not apply to applications on CTC/AB clients running Windows 3.1/3.11. To return telephone activity and call other routines, these applications must use ctcAbWinGetEvent. See the description of ctcAbWinGetEvent for more information.

- Not all events described are supported with Application Bridge Release 5.0; some events require Application Bridge Release 6.0 with the Event Bridge software option.

  Tables 2–2 to 2–5 indicate which events require Release 6.0 and Event Bridge software.

- Aspect Application Bridge Release 5.0 does not return information for the following ctcAbEventData structure fields:

  aniDigits
  dnisDigits
  agentId
  agentGroup
  agentMode

Aspect Application Bridge Messages

Refer to Tables 2–2 to 2–5 for details of the Application Bridge message associated with each CTC/AB event.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

eventData

type: **structure**
access: **write only**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type ctcAbEventData. The structure is defined in a CTC/AB definitions file (see Section 1.6) and is formatted as follows:

```
struct ctcAbEventData{
  unsigned int     refId;
  unsigned int     trackNumber;
  unsigned int     trackNode;
  unsigned int     oldRefId;
  unsigned int     oldTrackNumber;
  unsigned int     oldTrackNode;
  unsigned int     event;
  unsigned int     otherPartyType;
  unsigned char    otherParty [ctcMaxDnLen];
  unsigned int     otherPartyTrunk;
  unsigned char    aniDigits [ctcMaxDnLen];
  unsigned char    dnisDigits [ctcMaxDnLen];
  unsigned char    lineId [ctcMaxDnLen];
  unsigned char    agentId [ctcMaxDnLen];
  unsigned char    agentGroup [ctcMaxDnLen];
  unsigned int     agentMode;
  unsigned char    varDataA [ctcMaxDataALen];
  unsigned char    varDataB [ctcMaxDataBLen];
  unsigned char    varDataC [ctcMaxDataCLen];
  unsigned char    varDataD [ctcMaxDataDLen];
  unsigned char    varDataE [ctcMaxDataELen];
  unsigned char    subType [ctcMaxSubTypeLen];
  unsigned int     rTime;
  unsigned int     qTime;
  unsigned int     tTime;
  unsigned char    monitorParty [ctcMaxDnLen];
  unsigned char    nestedMonitorChannel [ctcMaxDnLen];
}
```

The strings in the ctcAbEventData structure are all null-terminated. The following sections describe the ctcAbEventData fields.

refId

This 32-bit field contains the call reference for a particular call. Use this call reference when you use CTC/AB routines that affect existing calls. For example, ctcAbTransferCall.

Note that after a call has been transferred, a new call reference for the call is returned in this field.

trackNumber

This is a 32-bit field that contains one of the following:

- If the CTC/AB server receives the Application Bridge Call Track Information Message (CTIM), the contents of the Application Bridge TRACKNUM field.
- If the CTC/AB server receives the Application Bridge Call Track Transfer Message (CTTM), the contents of the Application Bridge NEW_TRACKNUM field.

The track number identifies a call path that originates from a CallCenter node, and is not necessarily the same as the call reference.

trackNode

This is a 32-bit field that contains one of the following:

- If the CTC/AB server receives the Application Bridge Call Track Information Message (CTIM), the contents of the Application Bridge TRACKNODE field.
- If the CTC/AB server receives the Application Bridge Call Track Transfer Message (CTTM), the contents of the Application Bridge NEW_ TRACKNODE field.

The track node is the number of the CallCenter node from which the call track originated. For example, the number of the CallCenter receiving an incoming call.

oldRefId

If the reference identifier for a call changes, this 32-bit field contains the previous call reference.

oldTrackNumber

If the CTC/AB server receives the Application Bridge Call Track Transfer Message (CTTM), the contents of the Application Bridge ASS_TRACKNUM field are copied to this 32-bit field.

oldTrackNode

If the CTC/AB server receives the Application Bridge Call Track Information Message (CTIM), the contents of the Application Bridge ASS_TRACKNODE field are copied to this 32-bit field.

event

This 32-bit integer identifies the call or agent state event. For possible event values returned in this field:

- For stations and TeleSets, refer to Table 2–2.
- For trunks, refer to Table 2–3.
- For trunk groups and ACD groups, refer to Table 2–4.
- For InterQueue points, refer to Table 2–5.

These tables also show:

- How the values map to Aspect Application Bridge messages. For full details of the Application Bridge messages, refer to the Aspect Application Bridge documentation.
- Which of the other fields in the ctcAbEventData structure contain information when an event occurs. The amount of information that CTC/AB returns is dependent on the information provided by the Aspect CallCenter.

When ctcAbGetEvent returns the information, compare the values returned in the integer with the call event literals supplied in a CTC/AB definitions file (see Section 1.6).

Table 2–2 Events Returned by ctcAbGetEvent for Stations and TeleSets

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_AgentLoggedOn† (TeleSets only)			
	Agent State Event Message (ASEM)	The agent has logged on.	agentId agentGroup
ctcK_AgentLoggedOff† (TeleSets only)			
	Agent State Event Message (ASEM)	The agent has logged off.	agentId agentGroup
ctcK_AgentModeChange† (TeleSets only)			
	Agent State Event Message (ASEM)	An ASEM is sent whenever the state of an agent changes. For example, when the agent changes from Ready (Available) to Busy (Reserved).	agentMode agentId agentGroup
ctcK_CallConnect			
	Call Connect Message (CCM)	A SEND CONNECT step has been encountered during CCT processing, generating a Call Connect message. This message notifies the application that a call is ringing at the assigned station or TeleSet.	refId subType otherPartyType otherParty otherPartyTrunk Variable data fields‡

†Requires Aspect Application Bridge Release 6.0 and Event Bridge software.

‡One or more of the varDataA, varDataB, varDataC, varDataD, and varDataE fields

(continued on next page)

Table 2–2 (Cont.) Events Returned by ctcAbGetEvent for Stations and TeleSets

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_CallDisconnect			
	Call Disconnect Message (CDM)	A previously identified call has been disconnected.	refId subType otherPartyType otherParty otherPartyTrunk rTime qTime tTime Variable data fields‡
ctcK_CallQueued† (TeleSets only)			
	Call Queued Event Message (CQEM)	During CCT processing, one of the following steps was encountered: SELECT AGENT GROUP SELECT AGENT SUPERGROUP SELECT TRUNK GROUP	refId otherPartyType otherParty otherPartyTrunk
ctcK_CallTrackInformation			
	Call Track Information Message (CTIM)	A SEND TRACKDATA step has been encountered during CCT processing, generating a CTIM.	refId trackNumber trackNode subType otherPartyType otherParty otherPartyTrunk rTime qTime tTime Variable data fields‡

†Requires Aspect Application Bridge Release 6.0 and Event Bridge software.

‡One or more of the varDataA, varDataB, varDataC, varDataD, and varDataE fields

(continued on next page)

Table 2–2 (Cont.) Events Returned by ctcAbGetEvent for Stations and TeleSets

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_CallTrackTransfer			
	Call Track Transfer Message (CTTM)	Indication that a particular call has been transferred to a different destination. Track information is associated with the call.	refId trackNumber trackNode oldTrackNumber oldTrackNode subType otherPartyType otherParty otherPartyTrunk rTime qTime tTime Variable data fields‡
ctcK_CallTransfer			
	Call Transfer Message (CTM)	Indication that a particular call has been transferred to a different destination.	refId subType otherPartyType otherParty otherPartyTrunk rTime qTime tTime Variable data fields‡
ctcK_DestSeized† (TeleSets only)			
	Call Offered Event Message (COEM)	A call has been successfully dialed. If this call is external to the Aspect CallCenter, the network number has been verified and the outbound trunk seized. This does not indicate that the other end is actually ringing or answered.	refId otherPartyType otherParty otherPartyTrunk Variable data fields‡

†Requires Aspect Application Bridge Release 6.0 and Event Bridge software.

‡One or more of the varDataA, varDataB, varDataC, varDataD, and varDataE fields

(continued on next page)

Table 2–2 (Cont.) Events Returned by ctcAbGetEvent for Stations and TeleSets

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_InboundCall† (TeleSets only)			
	Call Offered Event Message (COEM)	An inbound call is ringing at the assigned station or TeleSet.	refId otherPartyType otherParty otherPartyTrunk Variable data fields‡
ctcK_Offhook† (TeleSets only)			
	Call Noticed Event Message (CNEM)	One of the following line keys has been pressed on the TeleSet: OUTSIDE LINE 1 OUTSIDE LINE 2 INSIDE LINE SUPERVISOR MESSAGE HELP	aniDigits
ctcK_OpAnswered† (TeleSets only)			
	Call Connected Event Message (CCEM)	The other party answered the call from the assigned station or TeleSet.	refId otherPartyType otherParty otherPartyTrunk Variable data fields‡ lineId
ctcK_OpConferenced† (TeleSets only)			
	Call Conferenced Event Message (CCFEM)	Another party on the call has created a conference call	refId oldrefId

†Requires Aspect Application Bridge Release 6.0 and Event Bridge software.

‡One or more of the varDataA, varDataB, varDataC, varDataD, and varDataE fields

(continued on next page)

Table 2–2 (Cont.) Events Returned by ctcAbGetEvent for Stations and TeleSets

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_OpDisconnected† (TeleSets only)			
	Call Disconnected Event Message (CDEM)	The other party on the call has been released and the call has been terminated.	refId
ctcK_TpAnswered† (TeleSets only)			
	Call Connected Event Message (CCEM)	This party has answered the call made to the assigned station or TeleSet.	refId otherPartyType otherParty otherPartyTrunk lineId
ctcK_TpConferenced† (TeleSets only)			
	Call Conferenced Event Message (CCFEM)	This party has included another party in a conference call.	refId oldrefId otherPartyType otherParty otherPartyTrunk
ctcK_TpDisconnected† (TeleSets only)			
	Call Disconnected Event Message (CDEM)	The last party on the call has been released and the call has been terminated.	refId
ctcK_TpRetrieved† (TeleSets only)			
	Call Retrieved Event Message (CREM)	A held call has been retrieved at the assigned station or TeleSet.	refId

†Requires Aspect Application Bridge Release 6.0 and Event Bridge software.

(continued on next page)

Table 2–2 (Cont.) Events Returned by ctcAbGetEvent for Stations and TeleSets

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_TpSuspended† (TeleSets only)			
	Call Held Event Message (CHEM)	This party has placed a call on hold.	refId
ctcK_Transferred†			
	Call Transfer Event Message (CTEM)	The call has been transferred.	refId oldrefId otherPartyType otherParty otherPartyTrunk

†Requires Aspect Application Bridge Release 6.0 and Event Bridge software.

Table 2–3 Events Returned by ctcAbGetEvent for Trunks

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_CallConnect			
	Call Connect Message (CCM)	A SEND CONNECT step has been encountered during CCT processing, generating a Call Connect message. This message notifies the application that a call is ringing on the assigned trunk.	refId subType otherPartyType otherParty otherPartyTrunk Variable data fields‡
ctcK_CallDisconnect			
	Call Disconnect Message (CDM)	A previously identified call has been disconnected (or abandoned) before the call was answered.	refId subType rTime qTime tTime Variable data fields‡

‡One or more of the varDataA, varDataB, varDataC, varDataD, and varDataE fields

(continued on next page)

Table 2–3 (Cont.) Events Returned by ctcAbGetEvent for Trunks

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_CallInformation			
	Call Information Message (CIM)	A SEND DATA step has been encountered during CCT processing, generating a CIM.	refId subType rTime qTime tTime Variable data fields‡
ctcK_CallQueued†			
	Call Queued Event Message (CQEM)	During CCT processing, one of the following steps was encountered: SELECT AGENT GROUP SELECT AGENT SUPERGROUP SELECT TRUNK GROUP	refId otherPartyType otherParty otherPartyTrunk
ctcK_CallTrackInformation			
	Call Track Information Message (CTIM)	A SEND TRACKDATA step has been encountered during CCT processing, generating a CTIM.	refId trackNumber trackNode subType otherPartyType otherParty otherPartyTrunk rTime qTime tTime Variable data fields‡

†Requires Aspect Application Bridge Release 6.0 and Event Bridge software.

‡One or more of the varDataA, varDataB, varDataC, varDataD, and varDataE fields

(continued on next page)

Table 2–3 (Cont.) Events Returned by ctcAbGetEvent for Trunks

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_CallTrackTransfer			
	Call Track Transfer Message (CTTM)	Indication that a particular call has been transferred to a different destination. Track information is associated with the call.	refId trackNumber trackNode oldTrackNumber oldTrackNode subType otherPartyType otherParty otherPartyTrunk rTime qTime tTime Variable data fields†
ctcK_CallTransfer			
	Call Transfer Message (CTM)	Indication that a particular call has been transferred to a different destination.	refId subType otherPartyType otherParty otherPartyTrunk rTime qTime tTime Variable data fields‡
ctcK_InboundCall			
	Call Noticed Event Message (CNEM)	A call has come in to the ACD for the trunk.	refId dnisDigits aniDigits

†Requires Aspect Application Bridge Release 6.0 and Event Bridge software.
‡One or more of the varDataA, varDataB, varDataC, varDataD, and varDataE fields

Table 2–4 Event Returned by ctcAbGetEvent for ACD Groups and Trunk Groups

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_CallQueued			
	Call Queued Event Message (CQEM)	During CCT processing, one of the following steps was encountered: SELECT AGENT GROUP SELECT AGENT SUPERGROUP SELECT TRUNK GROUP	refId otherPartyType otherParty otherPartyTrunk

Table 2–5 Events Returned by ctcAbGetEvent for InterQueues

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_CallDisconnect			
	Call Disconnect Message (CDM)	A previously identified call has been disconnected (or abandoned) before a trunk for the call could be negotiated.	refId subType rTime qTime tTime Variable data fields‡
ctcK_CallInformation			
	Call Information Message (CIM)	A SEND DATA step has been encountered during CCT processing, generating a CIM.	refId subType rTime qTime tTime Variable data fields‡
ctcK_CallQueued†			
	Call Queued Event Message (CQEM)	During CCT processing, one of the following steps was encountered: SELECT AGENT GROUP SELECT AGENT SUPERGROUP SELECT TRUNK GROUP	refId otherPartyType otherParty otherPartyTrunk

†Requires Aspect Application Bridge Release 6.0 and Event Bridge software.

‡One or more of the varDataA, varDataB, varDataC, varDataD, and varDataE fields

(continued on next page)

Table 2–5 (Cont.) Events Returned by ctcAbGetEvent for InterQueues

Event-Value	Aspect Message	Description	ctcAbEventData Fields Used
ctcK_CallTrackInformation			
	Call Track Information Message (CTIM)	A SEND TRACKDATA step has been encountered during CCT processing, generating a CTIM.	refId trackNumber trackNode subType otherPartyType otherParty otherPartyTrunk rTime qTime tTime Variable data fields‡
ctcK_InboundCall			
	Call Noticed Event Message (CNEM)	A call has come in to the ACD for the trunk.	refId dnisDigits aniDigits

‡One or more of the varDataA, varDataB, varDataC, varDataD, and varDataE fields

otherPartyType

This 32-bit field identifies the other party. It can contain one of the following values:

ctcK_TeleSet
ctcK_Station
ctcK_Trunk

otherParty

This field contains a character string that specifies the number for the other party.

The maximum length for otherParty is specified by the literal ctcMaxDnLen in a CTC/AB definitions file. (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

otherPartyTrunk

If the otherPartyType field contains the value ctcK_Trunk, this 32-bit field contains the trunk line number for the other party. The trunk line number is provided by the Aspect CallCenter.

aniDigits

This character string contains Automatic Number Identification (ANI) digits.

The maximum length for aniDigits is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

dnisDigits

This character string contains Dialed Number Identification Service (DNIS) digits.

The maximum length for dnisDigits is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

lineId

This character string contains the Terminating Line Identifier (TLI) for the call. The TLI identifies the actual destination or extension that answered the call.

The maximum length for lineId is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

agentId

This character string contains the extension number used by the agent when logging on.

The maximum length for agentId is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

agentGroup

This character string contains the agent's group number.

The maximum length for agentGroup is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

agentMode

This 32-bit field returns the current work mode for an agent. It can contain one of the following values:

ctcK_AgentAfterCallWork
ctcK_AgentOtherWork
ctcK_AgentReady
ctcK_AgentReserved

varDataA

This 24-byte field contains a copy of the data in the Application Bridge message variable field A.

varDataB

This 12-byte field contains a copy of the data in the Application Bridge message variable field B.

varDataC

This 8-byte field contains a copy of the data in the Application Bridge message variable field C.

varDataD

This 8-byte field contains a copy of the data in the Application Bridge message variable field D.

varDataE

This 44-byte field contains a copy of the data in the Application Bridge message variable field E.

subType

This 16-byte field contains a copy of the Application Bridge SUBTYPE message.

rTime

The contents of the Application Bridge RTIME field are copied to this 32-bit field when the CTC/AB server receives the following Application Bridge messages:

Call Transfer Message (CTM)
Call Disconnect Message (CDM)
Call Track Transfer Message (CTTM)

The RTIME field indicates the time (in seconds) that the incoming call rang before it was answered.

qTime

The contents of the Application Bridge QTIME field are copied to this 32-bit field when the CTC/AB server receives the following Application Bridge messages:

Call Transfer Message (CTM)
Call Disconnect Message (CDM)
Call Track Transfer Message (CTTM)

The QTIME message indicates the time (in seconds) that the call was in a queue before it was connected.

tTime

The contents of the Application Bridge TTIME field are copied to this 32-bit field when the CTC/AB server receives the following Application Bridge messages:

Call Transfer Message (CTM)
Call Disconnect Message (CDM)
Call Track Transfer Message (CTTM)

The TTIME message indicates the time (in seconds) that the call remained connected to its first destination before it was transferred.

monitorParty

Information is returned in this field when you call ctcAbGetEvent for a monitor channel. You assign to a monitor channel to receive events for a number of devices over a single channel (see ctcAbAssign for more information).

The device number returned in this field identifies the device for which event information is returned.

The maximum length for monitorParty is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

nestedMonitorChannel

This field returns a device number that identifies the nested monitor channel for which event information is returned. A nested monitor channel is a channel that is monitored by another monitor channel. Information is returned in this field only if you call ctcAbGetEvent for a monitor channel and that channel is monitoring another monitor channel.

The maximum length for nestedMonitorChannel is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

dontWait

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This 32-bit integer is a Boolean value which, when set, allows an application to poll for events without having to create a separate thread. If there is no new event data, the routine will not block and a ctcNoEvent condition value is returned.

ctcAbGetMonitor
Get Information About the Monitoring State

Format in C

unsigned int ***ctcAbGetMonitor*** *(ctcChanId channel,*
*unsigned int *monitorMode)*

Description

This routine returns information about the current monitoring state of the assigned device. The monitoring state can be changed with the ctcAbSetMonitor routine.

Restrictions

This routine is not supported for channels assigned to monitor channels.

Aspect Application Bridge Message

None.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

monitorMode

type: **integer (unsigned)**
access: **write only**
mechanism: **by reference**

This argument is the address of a 32-bit integer that receives one of the values in the following table:

Value	Description
ctcK_On	Indicates that monitoring on the assigned device is set on
ctcK_Off	Indicates that monitoring on the assigned device is set off

ctcAbHangupCall
Disconnect a Call

Format in C

unsigned int ***ctcAbHangupCall*** *(ctcChanId channel,*
unsigned int callRefId)

Description

The ctcAbHangupCall routine terminates an active call on the station, TeleSet, or trunk and returns it to the null state.

Restrictions

The following restrictions apply:

- This routine is supported for channels assigned to stations, TeleSets, or trunks only.
- Aspect Application Bridge Release 5.0 does not support a call reference identifier for the call you wish to hang up. See the description of the callRefId argument for details.

Aspect Application Bridge Message

Release Call Request (RCR)

CTC/AB sends an RCR to request that the Aspect CallCenter release the specified station, TeleSet, or trunk. For full details of this message, refer to the Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

callRefId

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This 32-bit integer contains the call reference identifier for the call you wish to hang up. This call reference identifier is returned by the ctcAbGetEvent routine.

Call reference identifiers are not supported by Aspect Application Bridge software Release 5.0. If you are using Release 5.0, you must specify the value zero (0) with this argument and the Aspect CallCenter hangs up the current active call on the assigned device.

ctcAbHoldCall
Put Current Call on Hold

Format in C

unsigned int ***ctcAbHoldCall*** *(ctcChanId channel)*

Description

The ctcAbHoldCall routine puts the current call on the assigned device on hold. You can then make a consultation call using ctcAbConsultationCall and either transfer the held call with ctcAbTransferCall, or create a conference call with ctcAbConferenceJoin. See the description of ctcAbConsultationCall, ctcAbTransferCall, and ctcAbConferenceJoin for details.

Retrieving a Call

If you place a call on hold, you can retrieve the call with ctcAbRetrieveHeld.

Restrictions

This routine is supported for channels assigned to TeleSets only.

Aspect Application Bridge Message

Process Key Request (PKR)

CTC/AB sends a Process Key Request to the Application Bridge to simulate the user pressing the TeleSet hold key. The Application Bridge returns a Process Key Request Response (PKRR) message to verify that the call is on hold. The application is then free to call other routines.

For full details of this message, refer to the Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

ctcAbMakeCall
Make a Call

Format in C

```
unsigned int ctcAbMakeCall
                    (ctcChanId              channel,
                    unsigned char           calledNumber [ctcMaxDnLen],
                    unsigned char           cct [ctcMaxDnLen],
                    unsigned int            callType,
                    unsigned int            *callRefId,
                    struct      ctcAbVarData *varData)
```

Description

The ctcAbMakeCall routine makes a call from the station or TeleSet to which the channel is assigned to any number that the Aspect CallCenter recognizes as valid.

You identify the device that you want to call with the calledNumber argument. This argument specifies the directory number (the telephone number) for the device.

Restrictions

This routine is supported for channels assigned to stations and TeleSets only.

Aspect Application Bridge Message

Place Call Request (PCR)

The Application Bridge returns a Place Call Request Response (PCRR) message to verify that the call is placed. The application is then free to call other routines.

For full details of this message, refer to the Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

calledNumber

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

This argument is the address of a character string that contains the number of the device you want to call. The ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

Note that if you specify the CCT to be used (see the cct argument), you do not need to include the access code in the number for the device.

The maximum length for calledNumber is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note this maximum length includes the null termination character (NUL).

cct

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

Use this argument in conjunction with the callType argument to establish the type of call and processing required. This argument is the address of a character string that contains the number of the CCT you want to use. The CCT contains instructions for processing the call within the Aspect CallCenter until it is connected.

This ASCII string contains a three-digit number. Specify the value 000 when using Least Cost Routing.

The maximum length for cct is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

callType

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This 32-bit integer contains a value that identifies the type of call you want to make. Specify one of the values in the following table:

Specify this value...	To instruct the Application Bridge...
ctcK_AbCallTypeLeastCost	To select the CCT (for least-cost routing). For this type of call, you must specify the value 000 with the cct argument.
ctcK_AbCallTypeUseCCT	To use the specified CCT to make the outbound call. For this type of call, you must specify the number of the CCT with the cct argument.
ctcK_AbCallTypeInternal	That this is an internal call to another agent. For this type of call, specify the agent's extension number with the calledNumber argument, and the address of a zero-length character string with the cct argument.

callRefId

type: **integer (unsigned)**
access: **write only**
mechanism: **by value**

This argument is the address of a 32-bit integer that receives the call reference value of the new call.

varData

type: **structure**
access: **read only**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type ctcAbVarData. The structure is defined in a CTC/AB definitions file (see Section 1.6) and is formatted as follows:

```
struct ctcAbVarData {
  unsigned char    varDataA [ctcMaxDataALen];
  unsigned char    varDataB [ctcMaxDataBLen];
  unsigned char    varDataC [ctcMaxDataCLen];
  unsigned char    varDataD [ctcMaxDataDLen];
  unsigned char    varDataE [ctcMaxDataELen];
}
```

The structure contains the following fields:

- **varDataA**

  This field contains information corresponding to the Application Bridge variable field A. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataA is specified by the literal ctcMax-DataALen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataB**

  This field contains information corresponding to the Application Bridge variable field B. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataB is specified by the literal ctcMax-DataBLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataC**

  This field contains information corresponding to the Application Bridge variable field C. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataC is specified by the literal ctcMax-DataCLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataD**

  This field contains information corresponding to the Application Bridge variable field D. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataD is specified by the literal ctcMaxDataDLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataE**

  This field contains information corresponding to the Application Bridge variable field E. This ASCII string can contain any combination of alphanumeric characters.

  The maximum length for varDataE is specified by the literal ctcMaxDataE-Len in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

ctcAbMakePredictiveCall
Make Predictive Calls

Format in C

```
unsigned int ctcAbMakePredictiveCall
                (ctcChanId              channel,
                 unsigned char          calledNumber [ctcMaxDnLen],
                 unsigned char          cct [ctcMaxDnLen],
                 unsigned int           callType,
                 struct   ctcAbPredData *predData,
                 struct   ctcAbVarData  *varData,
                 unsigned int           *callRefId)
```

Description

The ctcAbMakePredictiveCall routine allows a virtual party on the Aspect CallCenter to initiate calls on behalf of a group of users.

Depending on the configuration of your switch, at some time during the progress of the call, the call is allocated to a physical device. Only when the called device answers (or, for example, the phone rings a preconfigured number of times) does the call get put through to the user.

Restrictions

The following restrictions apply:

- This routine requires Application Bridge Release 6.0 and the Resource Bridge option.

  For more information about this software, refer to your Aspect documentation.

- This routine is supported for channels assigned to ACD groups only.

Aspect Application Bridge Message

Make Predictive Call Request (MPCR)

CTC/AB sends a Make Predictive Call Request to the Application Bridge. The Application Bridge returns a Make Predictive Call Request Response message. For more information, refer to the Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

calledNumber

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

This argument is the address of a character string that contains the number of the device you want to call. The ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

The maximum length for calledNumber is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

cct

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

Use this argument in conjunction with the callType argument to establish the type of call and processing required. This argument is the address of a character string that contains the number of the CCT you want to use. The CCT contains instructions for processing the call within the Aspect CallCenter until it is connected.

This ASCII string contains a three-digit number. Specify the value 000 when using Least Cost Routing.

callType

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This 32-bit integer contains a value that identifies the type of call you want to make. Specify one of the following:

Specify this value...	To instruct the Application Bridge...
ctcK_AbCallTypeLeastCost	To select the CCT (for least-cost routing). For this type of call, you must specify the value 000 with the cct argument.
ctcK_AbCallTypeUseCCT	To use the specified CCT to make the outbound call. For this type of call, you must specify the number of the CCT with the cct argument.
ctcK_AbCallTypeInternal	That this is an internal call to another agent. For this type of call, specify the agent's extension number with the calledNumber argument, and the address of a zero-length character string with the cct argument.

predData

type: **structure**
access: **read only**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type ctcAbPredData. The structure is defined in a CTC/AB definitions file (see Section 1.6) and is formatted as follows:

```
struct ctcAbPredData {
  unsigned char    origLineId [ctcMaxDnLen];
  unsigned int     rnaTimeout;
  unsigned int     answerMode;
  unsigned int     amsDelay;
  unsigned int     amrMode;
  unsigned int     answerMap;
  unsigned int     adparam;
  unsigned int     countryCode;
}
```

The following sections describe each ctcAbPredData field.

origLineId

This character string contains the originating line identity, used if the outgoing trunk is DPNSS. It is an ASCII string that can contain any combination of numbers 0 through 9 and the characters * and #.

The maximum length for this string is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

rnaTimeout

This 32-bit field contains the maximum time (in seconds) that the Aspect CallCenter allows before declaring a call unanswered. The time starts from the receipt of ringback tone or when an "alerting" message is received from the Aspect CallCenter network.

answerMode

This 32-bit integer contains a value that you specify to indicate when a call is successful (for example, after the call is answered or after the phone rings a set number of times). If the call is successful, the switch puts the call through to an agent.

Specify one of the following values:

Value	Description
ctcK_AbAnsDisRingback	Specifies that the call is successful if the call is answered (ringback tone ceased).
ctcK_AbAnsDisVoice	Specifies that the call is successful if answered after a specified delay. This ensures that the call is not answered by a modem.
ctcK_AbAnsScreen	Specifies that the call is successful if the call is answered before the Answer Machine Screen Delay time. This delay time determines whether a human or an answering machine answers the call. If an answering machine is detected, the call is not put through to an agent.

amsDelay

This 32-bit integer specifies the Answer Machine Screen Delay time (in seconds). This delay time is used if you specify ctcK_AbAnsScreen with the answerMode argument.

amrMode

This 32-bit field contains a value that specifies the method of reporting answer machine detection. This is used only if you specify ctcK_AbAnsScreen with the answerMode argument.

Specify one of the values in the following table:

Value	Description
ctcK_AbAmrThreshold	Report as soon as the duration of the response exceeds that usually given by humans. This mode provides the best opportunity for an agent to leave a message after the beep. It also enables an agent to use the Outbound Call Management System (OCMS) to override answer machine classification.
ctcK_AbAmrImmediate	Report as soon as the recorded voice message has ended. The Aspect CallCenter assumes that this message is the initial answering machine greeting.
ctcK_AbAmrFixed	Report after a fixed period of time following the end of the recorded voice message. The Aspect CallCenter attempts to delay until after the beep tone.
ctcK_AbAmrBeepUnknown	Report when the initial recorded voice message and beep tone have ended. This mode treats immediate beep tone following answer as an unknown tone and the Aspect CallCenter does not report it as an answering machine.
ctcK_AbAmrBeepAnsMc	Report when the initial recorded message and beep tone have ended. If the Aspect CallCenter detects immediate beep tone following answer, it reports it as an answering machine.

answerMap

This 32-bit field indicates which action(s) are taken in response to Application Bridge event(s) that are generated when the application makes the call:

- To take the ANSWER branch of the WAIT ANSWER CCT step, you use this field to specify a bitmask that corresponds to the event:

Event	Bitmask
Cause 22 received in ISDN DISCONNECT message. This indicates that the called number has been changed and a recorded announcement providing a new telephone number may follow	ctcM_AbAnsMapCause22
Special Information Tone (SIT) 1 received from the network	ctcM_AbAnsMapSit1
SIT 2 received from the network	ctcM_AbAnsMapSit2
SIT 3 received from the network	ctcM_AbAnsMapSit3
SIT 4 received from the network	ctcM_AbAnsMapSit4
SIT 5 received from the network	ctcM_AbAnsMapSit5
SIT 6 received from the network	ctcM_AbAnsMapSit6
SIT 7 received from the network	ctcM_AbAnsMapSit7
Unidentifyable SIT received from the network (includes European and Australian SIT)	ctcM_AbAnsMapSitOther
Timeout occurred after call answer without receiving voice (during answering machine screening only)	ctcM_AbAnsMapTimeout
Continuous voice with duration exceeding timeout value occurred after call answer (during answering machine screening only)	ctcM_AbAnsMapVoice
Unclassifiable tone received	ctcM_AbAnsMapUknown

- To disconnect the call, you do not specify a bitmask for the event.

For example, if you specify ctcM_AbAnsMapSit2 and ctcM_AbAnsMapSit3, the ANSWER branch is taken when a SIT 2 or SIT 3 event is received.

adParam

This 32-bit field indicates whether an optional feature for Answer Detect is used for the call. Currently, the Application Bridge supports only one optional feature for Answer Detect: detection for the end of ringback.

Use this field in the following way:

- To detect the end of ringback only if the ringback OFF interval exceeds 10 seconds, use this field to specify the bitmask ctcM_AbAnsDetRingbackEnd.
- To detect the end of ringback if either the ringback OFF interval exceeds maximum or the ON interval is shorter than minimum (except for the first ON interval), do not specify a bitmask in this field.

countryCode

This 32-bit field contains a value that identifies the destination country. This information is used by the Aspect CallCenter to specify country-specific tone frequencies.

Currently, the Aspect CallCenter supports one value only:

Value	Description
ctcK_CountryUSCan	Specifies that the target country is the United States or Canada

For other destinations, specify the value zero (0) in this field.

varData

type: **structure**
access: **read only**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type ctcAbVarData. The structure is defined in a CTC/AB definitions file (see Section 1.6) and is formatted as follows:

```
struct ctcAbVarData {
  unsigned char    varDataA [ctcMaxDataALen];
  unsigned char    varDataB [ctcMaxDataBLen];
  unsigned char    varDataC [ctcMaxDataCLen];
  unsigned char    varDataD [ctcMaxDataDLen];
  unsigned char    varDataE [ctcMaxDataELen];
}
```

The structure contains the following fields:

- **varDataA**

  This field contains information corresponding to the Application Bridge variable field A. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataA is specified by the literal ctcMax-DataALen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataB**

  This field contains information corresponding to the Application Bridge variable field B. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataB is specified by the literal ctcMax-DataBLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataC**

  This field contains information corresponding to the Application Bridge variable field C. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataC is specified by the literal ctcMax-DataCLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataD**

  This field contains information corresponding to the Application Bridge variable field D. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataD is specified by the literal ctcMax-DataDLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataE**

  This field contains information corresponding to the Application Bridge variable field E. This ASCII string can contain any combination of alphanumeric characters.

  The maximum length for varDataE is specified by the literal ctcMaxDataE-Len in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

ctcAbMakePredictiveCall

callRefId

type: **integer (unsigned)**
access: **write only**
mechanism: **by reference**

This argument is the address of a 32-bit integer that receives the call reference identifier for the new call.

ctcAbReassignResource
Reassign an Agent to Another Group, Team, or Class of Service

Format in C

```
unsigned int ctcAbReassignResource (ctcChanId      channel,
                                    unsigned char  group [ctcMaxDnLen],
                                    unsigned char  team [ctcMaxDnLen],
                                    unsigned int   cos )
```

Description

The ctcAbReassignResource routine reassigns an agent to a different ACD group, supervisor team, or Class of Service (COS).

The agent's default and current settings for group, team, and COS are stored in their user record on the Aspect CallCenter. You can use ctcAbReassignResource to change the *current* settings for an agent after they have logged on to a TeleSet.

ctcAbReassignResource enables you to:

- Change the COS for an agent
- Move an agent to a new ACD group (also known as agent group)
- Move an agent to a new supervisor team
- Reassign an agent to their default group, team, or COS

Note that when the Aspect CallCenter initializes, the current settings for group, team, and COS in an agent's user record are set to match the default settings.

Restrictions

The following restrictions apply:

- This routine requires Application Bridge Release 6.0 and the Resource Bridge option.

  For more information about this software, refer to your Aspect documentation.

- ctcAbReassignResource is supported for channels assigned to TeleSets only.

ctcAbReassignResource

Aspect Application Bridge Message

Reassign Resource Request (RRR)

CTC/AB sends an RRR to the Application Bridge. The Application Bridge returns a Reassign Resource Request Response message. For more information, refer to your Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

group

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

This argument is the address of a character string that contains the number for the ACD group to which the agent will be moved.

The maximum length for group is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

To reassign an agent to their default group, specify the address of a zero-length character string.

team

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

This argument is the address of a character string that contains the number for the supervisor team to which the agent will be moved.

The maximum length for team is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

To reassign an agent to their default team, specify the address of a zero-length character string.

cos

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This 32-bit integer contains the new COS to be assigned to the agent. Specify a value that identifies the COS as defined on the CallCenter.

To reassign an agent to their default COS, specify the value zero (0).

ctcAbRemoveMonitor
Remove a Device From a Monitor Channel

Format in C

unsigned int ***ctcAbRemoveMonitor*** *(ctcChanId channel,*
unsigned char deviceDN [ctcMaxDnLen])

Description

The ctcAbRemoveMonitor routine removes monitoring for a device associated with a monitor channel. Use this routine when you no longer want to receive event information for the device on the monitor channel.

To stop monitoring **all** devices on a monitor channel, and deassign the monitor channel, use ctcAbDeassign.

Restrictions

The following restrictions apply:

- This routine is supported for channels assigned to monitor channels only.
- This routine is not supported on CTC/AB clients running Windows 3.1 /3.11. CTC/AB applications running on Windows 3.1/3.11 cannot assign to monitor channels.

Aspect Application Bridge Message

None

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

deviceDN

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

This argument is the address of a character string that contains the DN for the device you no longer want to monitor. Specify one of the following:

For this type of device...	Specify...
Station	Equipment number
TeleSet	Equipment number
Trunk	Trunk number
ACD group	Group number
Trunk group	Trunk number
InterQueue	InterQueue number
Monitor channel	The setDN value returned by the routine ctcAbGetChannelInformation. See the description of this routine for more information.

This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

The maximum length for deviceDN is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

ctcAbRetrieveHeld
Retrieve a Call on Hold

Format in C

unsigned int ***ctcAbRetrieveHeld*** *(ctcChanId channel,*
unsigned int callRefId)

Description

The ctcAbRetrieveHeld routine cancels a consultation call and retrieves the call on hold.

For example, for A to cancel a consultation call and retrieve a call on hold:

1. A calls B, and places B on hold with ctcAbHeldCall.
2. A calls C using ctcAbConsultationCall.
3. There is no answer from C, so A uses ctcAbRetrieveHeld to cancel the call to C and retrieve the call to B.

Restrictions

The following restrictions apply:

- This routine is supported for channels assigned to TeleSets only.
- Aspect Application Bridge Release 5.0 does not support a call reference identifier for the held call. See the description of the callRefId argument for more information.

Aspect Application Bridge Message

CTC/AB sends one of the following:

- Process Key Request (PKR) for Application Bridge Release 5.0
- Retrieve Call Request (RTCR) for Application Bridge Release 6.0

This simulates the user pressing a TeleSet key to retrieve the held call. The Application Bridge returns a response message to verify that the call is retrieved. The application is then free to call other routines. For full details of these messages, refer to the Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

callRefId

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This 32-bit integer contains the call identifier value for the held call you wish to retrieve. This value is returned by the ctcAbGetEvent routine.

The call identifier value for the held call is not supported by Aspect Application Bridge Release 5.0. If you are using Application Bridge Release 5.0, specify the value zero (0) with this argument.

ctcAbSetAgentStatus
Set the Status for an Agent

Format in C

unsigned int ***ctcAbSetAgentStatus***

```
(ctcChanId      channel,
unsigned int    agentMode,
unsigned int    reason,
unsigned char   agentData [ctcMaxDnLen],
unsigned char   logicalAgent [ctcMaxDnLen])
```

Description

The ctcAbSetAgentStatus routine enables agents to declare themselves to be:

- Logged in
- Ready to take calls
- Completing details after a call
- Unavailable (not ready to take calls)
- Logged out

The Aspect CallCenter will not present calls to a user unless they declare themselves as ready to take calls.

Restrictions

The following restrictions apply:

- This routine is supported for channels assigned to TeleSets only.
- Agent log in and log out are not supported by Aspect Application Bridge Release 5.0. See the descriptions of the agentMode, reason, agentData, and logicalAgent, arguments for more information.

Aspect Application Bridge Message

The CTC/AB sends one of the following:

- For Application Bridge Release 5.0, a Process Key Request (PKR).
- For Application Bridge Release 6.0, a PKR, Sign On Request (SONR), or Sign Off Request (SOFR).

The Application Bridge returns a response message to verify the agent status. The application is then free to call other routines. For more information about these messages, refer to your Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

agentMode

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This argument specifies the status for a user. It contains one of the following values:

Value	Description
ctcK_AgentLogin†	Signs on the agent to the specified TeleSet.
ctcK_AgentLogout†	Signs off the agent from the specified TeleSet.
ctcK_AgentReady	Agent is available (ready to take calls). This simulates the user pressing the Aspect TeleSet READY key.
ctcK_AgentNotReady	Agent is not ready to receive calls (idle state). This simulates the user pressing the Aspect TeleSet RELEASE key.
ctcK_AgentAfterCallWork	Agent releases a call (hangs up) and is logging wrap-up information. This simulates the user pressing the Aspect TeleSet WRAP-UP key.

†Requires Aspect Application Bridge Release 6.0.

ctcAbSetAgentStatus

reason

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This 32-bit argument specifies the reason why an agent has signed off (ctcK_AgentLogout with the agentMode argument). Specify a value from 1 through 999 to identify a reason code defined on the Aspect CallCenter. Specify the value zero (0) to use the system default value (no code specified).

The reason argument is not supported for Release 5.0 of the Aspect Application Bridge software. If you are using this release, specify the value zero (0) with this argument.

agentData

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

This 32-bit argument can be used to specify a password (if required) when the agent logs in (ctcK_AgentLogin with the agentMode argument).

The maximum length for agentData is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

Note that this argument is not supported for Release 5.0 of the Aspect Application Bridge software. If you are using this release, specify the address of a zero-length string.

logicalAgent

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

This 32-bit argument specifies the extension number for the agent when they log in or log out.

The maximum length for logicalAgent is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

Note that this argument is not supported for Release 5.0 of the Aspect Application Bridge software. If you are using this release, specify the address of a zero-length string.

ctcAbSetMonitor
Set Monitoring for a Device

Format in C

unsigned int ***ctcAbSetMonitor*** *(ctcChanId channel,*
unsigned int monitorMode)

Description

The ctcAbSetMonitor routine changes the monitoring state of the assigned device.

You can use this routine with ctcAbGetEvent to receive useful information on the state of calls associated with a single device, such as a TeleSet. Status information is returned whenever a significant event occurs; for example, when an incoming call arrives, or when an active call is disconnected.

Monitoring Devices

Monitoring a device can provide information on the other party or parties involved in a phone call. It can return:

- The extension numbers for those parties on the same Aspect CallCenter
- For an outside call, the trunk number in use on the Aspect CallCenter

Monitoring also returns a reference number for calls on the assigned device. This call reference identifies the call, and can be used by the application for call tracking. In addition, some CTC/AB routines require the call reference to be passed as an argument to the routine.

Restrictions

This routine is not supported for channels assigned to monitor channels.

Aspect Application Bridge Message

None.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

monitorMode

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This argument is a 32-bit integer that contains one of the values in the following table:

Value	Description
ctcK_On	Sets monitoring on for the device
ctcK_Off	Sets monitoring off for the device

ctcAbSingleStepTransfer
Transfer a Call

Format in C

```
unsigned int ctcAbSingleStepTransfer
                    (ctcChanId              channel,
                    unsigned int            callRefId,
                    unsigned char           cct [ctcMaxDnLen],
                    unsigned int            trunk,
                    unsigned int            *newCallRefId,
                    struct   ctcAbVarData   *varData,
                    struct   ctcAbStatsData  *stats)
```

Description

The ctcAbSingleStepTransfer routine transfers a current call to the third party and disconnects the assigned device.

If you need to consult with the destination party before transferring the current call, use ctcAbHoldCall, ctcABConsultationCall, and ctcAbTransferCall. See the descriptions of these routines for more information.

Restrictions

This routine is supported for channels assigned to stations, TeleSets, or trunks.

Aspect Application Bridge Message

Transfer Call Request (TCR)

The Application Bridge returns a Transfer Call Request Response message to verify that the transfer is complete. The application is then free to call other routines.

For full details of this message, refer to the Aspect Application Bridge documentation.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

callRefId

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This 32-bit integer contains the call identifier value for the call you want to transfer.

The call identifier value is the latest call reference returned by the ctcAbGetEvent or ctcAbWinGetEvent routine.

cct

type: **character string (unsigned)**
access: **read only**
mechanism: **by reference**

This argument is the address of a character string that contains the number of the CCT you want to use. The CCT contains instructions for processing the call within the Aspect CallCenter until it is connected.

This ASCII string contains a three-digit number. Specify the value 000 to use Least Cost Routing.

The maximum length for cct is specified by the literal ctcMaxDnLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length includes the null termination character (NUL).

trunk

type: **integer (unsigned)**
access: **read only**
mechanism: **by value**

This argument specifies the trunk identifier of the party you want to transfer.

The trunk identifier value is the latest value returned by the ctcAbGetEvent routine.

newCallRefId

type: **integer (unsigned)**
access: **write only**
mechanism: **by reference**

This argument is the address of a 32-bit integer into which the Aspect CallCenter writes a call identifier value for the new transferred call.

varData

type: **structure**
access: **read and write**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type ctcAbVarData. The structure is defined in a CTC/AB definitions file (see Section 1.6) and is formatted as follows:

```
struct ctcAbVarData {
  unsigned char    varDataA [ctcMaxDataALen];
  unsigned char    varDataB [ctcMaxDataBLen];
  unsigned char    varDataC [ctcMaxDataCLen];
  unsigned char    varDataD [ctcMaxDataDLen];
  unsigned char    varDataE [ctcMaxDataELen];
}
```

The structure contains the following fields:

- **varDataA**

  This field contains information corresponding to the Application Bridge variable field A. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataA is specified by the literal ctcMax-DataALen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataB**

  This field contains information corresponding to the Application Bridge variable field B. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataB is specified by the literal ctcMax-DataBLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataC**

  This field contains information corresponding to the Application Bridge variable field C. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataC is specified by the literal ctcMax-DataCLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataD**

  This field contains information corresponding to the Application Bridge variable field D. This ASCII string can contain any combination of numbers 0 through 9 and the characters * and #.

  The maximum length for varDataD is specified by the literal ctcMaxDataDLen in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

- **varDataE**

  This field contains information corresponding to the Application Bridge variable field E. This ASCII string can contain any combination of alphanumeric characters.

  The maximum length for varDataE is specified by the literal ctcMaxDataE-Len in a CTC/AB definitions file (see Section 1.6). Note that this maximum length does not include the null termination character (NUL).

stats

type: **structure**
access: **write only**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which, you allocate memory of type ctcAbStatsData. The structure is defined in a CTC/AB definitions file (see Section 1.6) and is formatted as follows:

```
struct ctcAbStatsData{
  unsigned int     rTime;
  unsigned int     qTime;
  unsigned int     tTime;
}
```

The strings in the ctcAbStatsData structure are all null-terminated. The ctcAbStatsData structure provides a statistics block for the call and contains the following fields:

- **rTime**

  A copy of the Application Bridge RTIME message is copied to this 32-bit field. The RTIME message indicates the time (in seconds) that the incoming call rang before it was answered.

- **qTime**

  A copy of the Application Bridge QTIME message is copied to this 32-bit field. The QTIME message indicates the time (in seconds) that the call was in a queue before it was connected.

- **tTime**

  A copy of the Application Bridge TTIME message is copied to this 32-bit field. The TTIME message indicates the time (in seconds) that the called remained connected to its first destination before it was transferred.

ctcAbTransferCall
Transfer a Call

Format in C

unsigned int ***ctcAbTransferCall*** *(ctcChanId channel)*

Description

The ctcAbTransferCall routine completes the transfer of a call initiated by the ctcAbConsultationCall routine. It transfers the call to a the destination device, and disconnects the assigned device.

For example, for A to transfer to C an incoming call from B (where A's current call is the call from B):

1. B calls A, using ctcAbMakeCall, and A answers.
2. A place B on hold, using ctcAbHoldCall.
3. A calls C, using ctcAbConsultationCall.
4. A invokes ctcAbTransferCall when connected to C. B and C are now connected and A is disconnected.

To screen (or supervise) a transfer, A waits until speaking to C before invoking ctcAbTransferCall. For unscreened (or unsupervised) transfer, A invokes ctcAbTransferCall before C answers the telephone. You can also use ctcAbSingleStepTransfer to transfer the call directly to C. See the description of ctcAbSingleStepTransfer for more information.

Restrictions

This routine is supported for channels assigned to TeleSets only.

Aspect Application Bridge Message

Process Key Request (PKR)

CTC/AB sends a Process Key Request to the Application Bridge to simulate the user pressing a TeleSet TRANSFER key. The Application Bridge returns a Process Key Request Response message to verify that the call is transferred. The application is then free to call other routines.

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

ctcAbWinGetEvent
Get Information About Event and State Changes

Format in C

```
unsigned int ctcAbWinGetEvent (ctcChanId          channel,
                               struct  lpvASB     *lpvASB,
                               HWND               hWnd,
                               struct  ctcAbEventData *eventData)
```

Description

The ctcAbWinGetEvent routine is available on CTC/AB clients running Windows 3.1/3.11 or Windows for Workgroups only. It is a non-blocking routine that enables a CTC/AB application to receive telephony events for the assigned device.

ctcAbWinGetEvent returns the same information as ctcAbGetEvent. For details, see the description of ctcAbGetEvent.

To use ctcAbWinGetEvent, you must first set monitoring on with the ctcAbSetMonitor routine.

How ctcAbWinGetEvent Returns Event Data

When an event occurs at the assigned device, CTC/AB:

- Returns the event in the ctcAbEventData structure
- Posts a PM_CTC_EVENT completion message to the window specified by the hWnd argument

Associated with the completion message is an lParam parameter that specifies the address for the lpvASB structure. The lpvASB structure contains the routine completion status and a read-only value, for example, a pointer to the ctcAbEventData structure into which event information has been written.

The amount of information that CTC/AB returns depends on the information provided by the Aspect CallCenter. This may be different for a call that is internal to the Aspect CallCenter and for an outside call, depending on the type of trunks connected to the Aspect CallCenter.

Using ctcAbWinGetEvent With TeleSets and Trunks

For channels assigned to TeleSets or trunks, some event information is available with Application Bridge Release 6.0 only (see Tables 2–2, 2–3, and 2–4). To receive this information, a channel must be assigned to the ACD group or trunk group associated with the device before you assign to the device.

For example, to receive Application Bridge Release 6.0 events for the TeleSets in an ACD group, you:

1. Assign a channel to the ACD group
2. Assign a channel to each TeleSet in the ACD group

You do not need to assign more than one channel to the ACD group. CTC/AB returns the additional events on each channel assigned to a TeleSet in the group.

Using ctcAbWinGetEvent With InterQueue Points

If your Aspect CallCenter is part of a network of CallCenters, you can monitor calls that it receives from another CallCenter by assigning a channel to an InterQueue.

A Network InterQueue is a virtual channel used to notify a CallCenter that it will receive a call from another CallCenter in the network. When the target CallCenter receives this notification, it processes it as if the call had already been presented. For example, it can respond with an Application Bridge Call Information Message (CIM) or Call Track Information Message (CTIM). At the same time, it negotiates receipt of the call over a real trunk.

When CTC/AB receives an Application Bridge message for a call associated with the InterQueue, it generates an event shown in Table 2–5. As soon as negotiation for a trunk is complete, the call is no longer associated with the InterQueue. Instead, CTC/AB receives Application Bridge messages for the call on the trunk and generates corresponding events. Table 2–3 shows CTC/AB events that are returned for trunks.

This means that to track the progress of the call, a channel must be assigned to both the InterQueue point and the trunk used for the call.

Restrictions

The following restrictions apply:

- This routine is not supported for channels assigned to monitor channels.
- This routine is supported on CTC/AB clients running Windows 3.1/3.11 or Windows for Workgroups only.
- Not all events described are supported with Application Bridge Release 5.0. Tables 2–2, 2–3, and 2–4 indicate which events require Release 6.0.
- Aspect Application Bridge Release 5.0 does not return information for the following ctcAbEventData structure fields:

  aniDigits
  dnisDigits
  agentId
  agentGroup
  agentMode

Arguments

channel

type: **ctcChanId**
access: **read only**
mechanism: **by value**

This argument is a ctcChanId datatype that contains the channel identifier (channel ID) value returned by ctcAbAssign for the device in use.

lpvASB

type: **structure**
access: **write only**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type lpvASB. The structure is defined in CTABWI6.H installed on your system.

The lpvASB structure is formatted as follows:

```
struct  lpvASB{
  unsigned int     dwStatus;
  unsigned int     lpvDataPointer;
  unsigned int     lpvChannel;
}
```

The lpvASB structure contains the following fields:

- **dwStatus**

  On completion of the asynchronous procedure, this 32-bit field contains the routine completion status. This is a write only value.

- **lpvDataPointer**

  This 32-bit field contains a read only value. For example, if you are monitoring multiple channels, you can use this field to identify the ctcAbEventData structure into which event information has been written.

- **lpvChannel**

  On completion of the asynchronous procedure, this 32-bit field contains the identifier for the channel for which you want event information.

hWnd

type: **HWND**
access: **read only**
mechanism: **by reference**

This handle specifies the window where CTC/AB returns the PM_CTC_EVENT message generated by an event at the assigned device.

eventData

type: **structure**
access: **write only**
mechanism: **by reference**

This argument contains the address of a fixed-format structure, for which you allocate memory of type ctcAbEventData. The structure is defined in one of the CTC/AB definitions files (see Section 1.6) and is formatted as follows:

```
struct ctcAbEventData{
  unsigned int     refId;
  unsigned int     trackNumber;
  unsigned int     trackNode;
  unsigned int     oldRefId;
  unsigned int     oldTrackNumber;
  unsigned int     oldTrackNode;
  unsigned int     event;
  unsigned int     otherPartyType;
  unsigned char    otherParty [ctcMaxDnLen];
  unsigned int     otherPartyTrunk;
  unsigned char    aniDigits [ctcMaxDnLen];
  unsigned char    dnisDigits [ctcMaxDnLen];
  unsigned char    lineId [ctcMaxDnLen];
  unsigned char    agentId [ctcMaxDnLen];
  unsigned char    agentGroup [ctcMaxDnLen];
  unsigned int     agentMode;
  unsigned char    varDataA [ctcMaxDataALen];
  unsigned char    varDataB [ctcMaxDataBLen];
  unsigned char    varDataC [ctcMaxDataCLen];
  unsigned char    varDataD [ctcMaxDataDLen];
  unsigned char    varDataE [ctcMaxDataELen];
  unsigned char    subType [ctcMaxSubTypeLen];
  unsigned int     rTime;
  unsigned int     qTime;
  unsigned int     tTime;
  unsigned char    monitorParty [ctcMaxDnLen];
  unsigned char    nestedMonitorChannel [ctcMaxDnLen];
}
```

The strings in this structure are all null-terminated. For a description of these fields, refer to the description of the ctcAbGetEvent routine.

3

Error and Condition Values Returned

Table 3–1 provides a brief description of the errors and condition values that can be returned by CTC/AB routines.

Use Table 3–1 in conjunction with the ctcAbErrMsg routine. This routine provides the name of the condition or error associated with a returned value. For more information, refer to the description of ctcAbErrMsg routine in Chapter 2.

3.1 Mapping Errors to Routines

It is not possible to specify which specific errors and conditions can be returned for each CTC/AB routine. However to help you determine and isolate problems, Table 3–1 shows the source of the reported condition: the CTC/AB API, the CTC/AB server, or the Aspect CallCenter.

The following general guidelines apply:

- Errors from the CTC/AB API are usually returned for programming errors. For example, if you specify an invalid type or argument.
- Condition values from the CTC/AB server are usually associated with resources, or CTC/AB management.
- Condition values from the Aspect CallCenter are often returned when there is a problem with the device state or the call reference. For example, when you provide an invalid call reference, or try to perform an operation and the device is in the wrong state for that operation.

Table 3–1 Condition Values Returned

Condition	From	Description
ctcBadObjState	Aspect CallCenter	The object is in the incorrect state for the service. The Aspect CallCenter is unable to provide more specific information.
ctcBindFail	CTC/AB API	An RPC network binding handle cannot be created from the serverName and networkType arguments for ctcAbAssign.
ctcComFail	CTC/AB server	Insufficient virtual memory has been detected during the communications initialization procedure.
ctcCondError	CTC/AB server	An internal error has occurred on the CTC/AB server.
ctcCondWaiting	CTC/AB server	An internal error has occurred on the CTC/AB server.
ctcDeadLock	CTC/AB server	An internal error has occurred on the CTC/AB server.
ctcEventDataLost	CTC/AB server	A number of events have occurred at the same time and some event data has been lost.
ctcEventInProgress	CTC/AB server	The ctcAbGetEvent or ctcAbWinGetEvent routine has already been called.
ctcInitFail	CTC/AB server	Insufficient virtual memory has been detected during the communications initialization procedure.
ctcInsMem	CTC/AB server	Insufficient virtual memory available, either on the CTC/AB server or the CTC/AB client, to complete the routine. Check your application's use of memory and ask your system manager to check the system parameters.
ctcInvAgentMode	CTC/AB API	The agentMode argument for ctcAbSetAgentStatus contains an invalid value.
ctcInvalidDest	Aspect CallCenter	The specified called party is invalid.
ctcInvalidFeature	Aspect CallCenter	The request specified an invalid feature.
ctcInvCallIdentifier	Aspect CallCenter	The call identifier is invalid.
ctcInvCct	CTC/AB server	The specified CCT is invalid.
ctcInvChan	CTC/AB API	An invalid channel identifier was specified. Specify the channel identifier as returned by the ctcAbAssign routine.
ctcInvClassOfService	Aspect CallCenter	The specified Class Of Service (COS) is not recognized by the Aspect CallCenter.
ctcInvDevIdentifier	Aspect CallCenter	The device identifier is invalid.

(continued on next page)

Table 3–1 (Cont.) Condition Values Returned

Condition	From	Description
ctcInvGroup	Aspect CallCenter	The specified ACD group or trunk group is not recognized by the Aspect CallCenter.
ctcInvLogId	CTC/AB API	The specified logical identifier is invalid or does not exist. Make sure you specify the same logical identifier as defined on the CTC/AB server, which specifies the Aspect CallCenter in use.
ctcInvMonitorMode	CTC/AB API	The monitorMode argument for ctcAbSetMonitor contains an invalid value.
ctcInvNetType	CTC/AB API	The networkType argument for ctcAbAssign contains an invalid or unsupported RPC protocol sequence string.
ctcInvokeLockError	CTC/AB server	An internal error has occurred on the CTC/AB server.
ctcInvokeUnlockError	CTC/AB server	An internal error has occurred on the CTC/AB server.
ctcInvServerName	CTC API	The serverName parameter for ctcAbAssign contains an invalid CTC/AB server name or address string.
ctcInvTeam	Aspect CallCenter	The specified team is not recognized by the Aspect CallCenter.
ctcLCBFail	CTC/AB server	Insufficient virtual memory has been detected during the communications initialization procedure.
ctcLCBQLockError	CTC/AB server	An internal error has occurred on the CTC/AB server.
ctcLCBQUnlockError	CTC/AB server	An internal error has occurred on the CTC/AB server.
ctcLinkConnectFail	CTC/AB server	The link between the CTC/AB server and the Aspect CallCenter has failed.
ctcLinkDown	CTC/AB server	The link between the CTC/AB server and the Aspect CallCenter is down.
ctcLogIdTooLong	CTC/AB API	Logical identifier is too long.
ctcMonAreadyOn	CTC/AB API	Monitoring is already set on for this channel.
ctcMonitorOff	CTC/AB API	Monitoring is set off for this channel so the call to ctcAbGetEvent has been returned.
ctcMonMaxExceeded	CTC/AB API	The maximum number of monitors for the CTC/AB server has been reached.
ctcMonNotOn	CTC/AB API	Monitoring is not set on for this channel.
ctcMutexLocked	CTC/AB server	An internal error has occurred on the CTC/AB server.

(continued on next page)

Table 3–1 (Cont.) Condition Values Returned

Condition	From	Description
ctcNetWriteErr	CTC/AB API	The CTC/AB client cannot communicate with the CTC/AB server because the link is down or there are insufficient resources on the CTC/AB client.
ctcNoEvent	CTC/AB API	The dontWait argument for ctcAbGetEvent is set to TRUE and there is no event data at the CTC/AB server for this channel.
ctcNotMon	CTC/AB server	Monitoring is not currently enabled.
ctcNoUnlock	CTC/AB server	An internal error has occurred on the CTC/AB server.
ctcParseErr	CTC/AB server	The CTC/AB server could not parse the message from the Aspect CallCenter. This indicates an internal error. Report the problem to Dialogic.
ctcRcvReqRej	CTC/AB server	The Aspect CallCenter rejected a message or request from the CTC/AB server. This indicates an internal error. Report the problem to Dialogic.
ctcReadError	CTC/AB server	The read data request on the link between the CTC/AB server and the Aspect CallCenter has returned an error. This could indicate that the Aspect CallCenter has stopped or restarted the link, or that there may be a problem with the link hardware.
ctcRequestPending	Aspect CallCenter	The request is waiting to be processed by the Aspect CallCenter.
ctcSuccess	CTC/AB API	The routine completed successfully.
ctcTimeout	CTC/AB server	The Aspect CallCenter did not respond to the request from the CTC/AB server. There may be a problem with the link between the CTC/AB server and the Aspect CallCenter, or the Aspect CallCenter may be too busy to respond.
ctcUCBFail	CTC/AB server	The UCB initialization procedure detected insufficient virtual memory on the CTC/AB server.
ctcUnsupProc	CTC/AB API	The specified procedure is not supported for the assigned device.
ctcXmitError	CTC/AB server	The send data request on the link between the CTC/AB server and the Aspect CallCenter has returned an error condition.

Index

A

B

C

L

M

N

O

P

Q

R

S

T

U

W