CHAPTER 2

Writing Client Applications

This chapter provides background information about CTI client applications. It discusses the following topics:

- Design considerations for client applications
- Various client application models
- Programming languages and environments you can use to write client applications
- The CTI Desktop sample applications

Client Application Guidelines

This section lists some guidelines and requirements you should take note of before you write a client application.

Design Considerations

A good application starts with a good design. Make sure you have a design that meets the needs of your business. Following are some design issues to consider before you start to write your client application:

- What specific task(s) do you want your application to accomplish?
- What will your application be modeling? (Calls? An agent? The phone?)

- What exactly do you want to appear on the screen? (Customer information? Agent information? Call information?)
- Will your application be handling multiple calls simultaneously, and if so, how?
- Will your application just do screen pops, or will it perform third-party call-control functions—for example, answer, transfer, conference—as well? (Recommendation: It is usually best to get the screen pop working first, and then move on to introducing third-party call-control functions.)

Client Application Requirements

Depending on the development and desktop environments chosen, certain requirements apply.

The following requirements apply to Cisco CTI Desktop components (Softphone Controls, Desktop Control Server, CTIClient):

- You can write applications in any language that supports the Microsoft Component Object Model (COM) binary interface. This includes both low order languages (for example, Visual C++) and high order languages (such as Visual Basic or PowerBuilder). CTI Desktop also supports scripting performed within an OLE container (such as Internet Explorer).
- If your program is a CTIClient implementation, it must define an asynchronous event handler. All languages that support OC96 ActiveX controls have such a mechanism.
- Client applications can run on any platform that supports the OLE2 protocol (for example, Windows 95, Windows 98, Windows NT).

Applications that host Cisco JavaClient must be written in Java and deployed in a 1.1.5+ JVM.

CTI Client Application Models

You can use either of two principal client models to integrate call center applications with ICM software: *agent workstation (desktop)* or *CTI Bridge (All Events service)*.

These two models—as well as the relations of the CTI clients, the CTI Server, and ICM software—are discussed in the *Cisco ICM Software CTI Product Description* guide. The most significant difference between these two models is that:

- The agent workstation client application connects a single agent workstation to the CTI environment—the application is interested only in the data and events associated with the single agent's teleset
- The CTI Bridge client application is interested in all the events associated with all the agent telesets—the CTI Bridge application acts as an intermediary between the agent workstations and the rest of the CTI environment

Currently, if you wish to develop a CTI Bridge application, you must interface directly to the CTI Server.

Call Data

As discussed in the *Cisco ICM Software CTI Product Description* guide, an important source of information for the CTI client is the call data. The possible kinds of call data available are:

- Those that are completely defined as to type, content, and format—for example, ANI and DNIS
- Those completely defined as to type and format, but more loosely defined as to the precise content—for example, Caller-Entered Digits (CED) and Wrapup Data
- Those defined only as to format—the ten user-defined call variables
- Those completely user-defined—expanded call variables, which may consist of a user-defined number of named variable and named array fields (available only if the Expanded Call Context (ECC) feature is explicitly enabled in ICM software). For specifics on enabling, creating, and naming ECC variables, see the *Cisco ICM Software Script Editor Guide*.

For specifics on the handling of call data, consult the documentation for the particular Cisco CTI product offerings that you are using to develop your application. The following table lists elements (controls, methods, messages, classes, functions) that interact with the call data.

Table 2-1 Interacting with Call Data

Product Offering	Elements that Interact with Call Data
Softphone Controls (controls)	Answer/Release, Call Appearance Manager, Conference, Make Call, Transfer
Desktop Control Server (methods)	GetCallVariableArrayByName, GetCallVariableByName, GetNamedArrayBoundsByName, MakeCall, MakeCC, ReleaseCall, SingleStepConferenceCall, SingleStepTransferCall
CTIClient (methods)	BlindConferenceCall, BlindTransferCall, ConferenceCallEx, GetCallVariableArrayByName, GetCallVariableByName, GetNamedArrayBoundsByName, MakeCall, MakeCallEx, MakeConsultCall, MakeConsultCallEx, PushCallData, ReleaseCall, SetCallVariableArrayByName, SetCallVariableByName, TransferCall, TransferCallEx
CTI Server (messages)	AGENT_PRE_CALL_EVENT, BEGIN_CALL_EVENT, CALL_DATA_UPDATE_EVENT, CALL_TRANSLATION_ROUTE_EVENT, CONFERENCE_CALL_REQ, CONSULTATION_CALL_REQ, MAKE_CALL_REQ, MAKE_PREDICTIVE_CALL_REQ, RELEASE_CALL_REQ, SET_CALL_DATA_REQ, SNAPSHOT_CALL_CONF, TRANSFER_CALL_REQ
JavaClient (classes)	CallObject, ConferenceCallReqEvent, ConsultationCallReqEvent, MakeCallReqEvent, ReleaseCallReqEvent, TransferCallReqEvent

Service Models

There are four basic service models: Client Events, All Events, Peripheral Monitor, and Client Monitor.

In the Client Events model, the CTI client receives the call and agent state change events associated with a specific ACD phone. This is the service model that corresponds to the agent workstation client application.

In the All Events model, the CTI client receives all call and agent state change events associated with any ACD phone. This is the service model that is usually used with the CTI Bridge client application.

In the Peripheral Monitor model, the CTI client may dynamically add and remove devices and/or calls that it wishes to receive call and agent state events for. This would be used by a CTI Bridge client application that is not interested in truly monitoring all events. For example, the CTI client may only be interested in the calls associated with a particular IVR. The client could monitor all events and discard the ones not associated with the given IVR. However, using the Peripheral Monitor model would allow the client to only receive the events it is interested in.

A variant that can be used with the above models is Client Monitor. This allows a CTI client to receive notification when any other CTI client sessions are opened or closed, and allows monitoring of the activities of these sessions. This may prove useful in debugging or in providing a CTI client for use by a supervisor.

CTI Server Protocol Versions

Various CTI Server protocol versions were introduced with various versions of ICM software. Different components of Cisco CTI function with different protocol versions. An earlier protocol version can always be used with an ICM software version that is later than the one where it was introduced.

Table 2-2 relates each CTI Server protocol version with the ICM version it was released with.

Table 2-3 indicates which components of Cisco CTI run with which protocol versions.

Table 2-2 CTI Server Protocol Versions & ICM Versions

CTI Server Protocol Version	ICM Version
5	3.0 SP2+, 4.0
6	4.1
7	4.5
8	4.6

Table 2-3 CTI Server Protocol Versions & CTI Components

CTI Server Protocol Version	Components
5	(CTI Desktop 2.5, 4.0) CTI Server, CTIClient, DCS, Softphone Controls (CTI Desktop 4.1) JavaClient
6	(CTI Desktop 4.1) CTI Server, CTIClient, DCS, Softphone Controls (CTI Desktop 4.1.2, 4.1.3) CTI Server, CTIClient, DCS, Softphone Controls, JavaClient

Cisco CTI Desktop Sample Applications

The CTI Desktop includes sample applications that demonstrate simple client application functions. In addition to the executable file for each application, the Cisco CTI CD includes, as part of the CTI Desktop, the related source files, header files, resource files, and documentation. You can both run the sample applications and use the source files as a guideline to follow when you write your own client applications.

The sample applications and their associated files reside in the subdirectories underneath the subdirectory named samples. The samples directory contains the following subdirectories:

- **CTIClient**. This directory contains sample applications that use the CTIClient DLL to communicate with the CTI Server. It contains the following subdirectories:
  - **Screenpop**. This directory contains HTML, Visual Basic, and Visual C++ versions of a sample CTIClient screenpop application.
  - **Delphi**. This directory contains sample code for CTIClient usage in Delphi.
  - **Phonefordemo**. This directory contains a Visual Basic version of a sample soft phone built with CTIClient.
  - **PowerBuilder**. This directory contains sample code for CTIClient usage in PowerBuilder.

- **DCS**. This directory contains sample applications that use the Desktop Control Server to communicate with the CTI Server. It contains the following subdirectories:
  - **Vcsamp**. This directory contains a Visual C++ version of a sample DCS screenpop application.
  - **Vbsamp**. This directory contains a Visual Basic version of a sample DCS application that processes a single customer call.
- **Softphone**. This directory contains sample Softphone Controls applications. It contains the following subdirectories:
  - **EnterpriseAgentPhone**. This directory contains a Softphone compliant with Enterprise Agent.
  - **IPCC Phone**. This directory contains a Softphone compliant with IPCC.
  - **GeoTelDemo**. This directory contains a fully functional Softphone Controls application written in Visual Basic.
  - **IE4**. This directory contains a Softphone Controls application accessible from Internet Explorer.
  - **Screenpop**. This directory contains a Softphone Controls screenpop application written in Visual Basic.
  - **Sft_vb_ddesamp**. This directory contains a sample application using DDE for interprocess communication.
  - **Softphone**. This directory contains the Visual Basic code used to create CTI Desktop Softphone.