



# Developing Cisco IP IVR Applications

---

The first part of this section provides a general overview of the Cisco IP Telephony Solution application framework. It describes how to use this framework, which includes the Cisco Application Editor, to implement Cisco IP Interactive Voice Response (Cisco IP IVR) applications. The next part of the section describes how to use the Cisco Application Editor to build Cisco IP IVR applications. The last part of this section describes how to implement interactive voice response unit (VRU) scripts for integrating Cisco IP Telephony system with Cisco Intelligent Contact Manager (ICM).

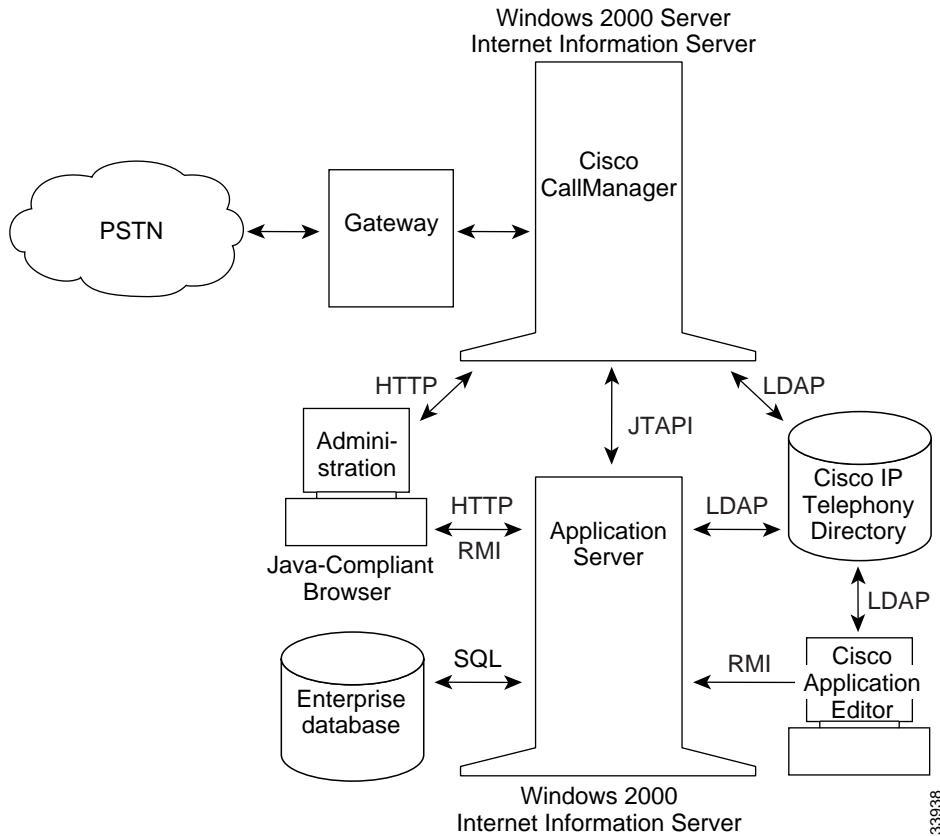
The following topics are described:

- Cisco IP Telephony Components, page 1-1
- Cisco IP AutoAttendant, page 1-7
- How the Cisco IP AutoAttendant Works, page 1-9
- Using Application Editor Steps, page 1-10
- Using the ICM VRU Interface, page 1-12
- Installation and Configuration Roadmap, page 1-13

## Cisco IP Telephony Components

The Cisco IP Telephony solution provides the components required to design, deploy, and run Cisco IP IVR and other media applications throughout an organization. These components are illustrated in Figure 1-1.

Figure 1-1 Components of the Cisco IP Telephony Solution



As shown in the illustration, a Cisco IP Telephony system is built from six main components:

- Gateway—connects the enterprise IP telephony network to the Public Switched Telephone Network (PSTN) and to other private telephone systems such as PBX, ACD.
- Cisco CallManager—provides the features required to implement IP phones, manage gateways, and enable Cisco IP IVR.

- Application Server—contains the Cisco IP Telephony application engine that runs Cisco IP IVR applications. Cisco IP Telephony applications are composed of a series of steps, implemented as Java Beans packaged in a .jar file.
- Cisco IP Telephony Directory—stores applications (or Cisco IP IVR scripts) and configuration information in a Lightweight Directory Access Protocol (LDAP) directory. The part of the directory that stores IP IVR applications is called the repository.
- Administration Client—starts, stops, configures, and monitors Cisco IP Telephony application engine using a standard web browser interface
- Cisco Application Editor—allows designers to create, modify, and debug Cisco IP IVR applications

These components are designed to use open standards, to be platform independent, and to be easily distributed for scalability, performance, and fault tolerance. Figure 1-1 illustrates the components running on separate machines, but they can be implemented on a single server, or distributed according to the needs of a specific organization and network. In the current release, Cisco IP Telephony application components run on Microsoft Windows 2000 systems.

## Gateway

The gateway component of the Cisco IP Telephony solution integrates Voice over IP (VoIP) network resources with the Public Switched Telephone Network (PSTN). All voice gateways provide an Ethernet interface for connecting to the VoIP network and various voice ports for connecting to different type of plain old telephone service (POTS) devices. The gateway component of the Cisco IP Telephony network can be implemented by using any of the following Cisco hardware products:

- Cisco Access Analog Station Gateway
- Cisco Access Digital Trunk Gateway
- Cisco Catalyst 6000 with the Analog Interface Module, or with the T1 and Services Module
- Cisco Voice Gateway 200 (VG200)

- Cisco IOS-based gateways with voice network modules (digital and analog)

Cisco gateway products work with one or more of the following protocols:

- Skinny Station Protocol—a Cisco protocol used for communication between the gateway and the Cisco CallManager
- H.323—A public standard that allows the interconnection of telephony and other media devices over TCP/IP networks
- Media Gateway Control Protocol (MGCP)—A public standard that simplifies gateway configuration, compared to H.323, and that allows for advanced features, such as redundancy.

When used with analog voice network modules, the Cisco VG200 supports MGCP with Cisco CallManager 3.0 or later. MGCP simplifies gateway administration and allows the use of redundant Cisco CallManager servers to eliminate a possible single point of failure in the VoIP network.

## Cisco CallManager

Cisco CallManager provides the features for which organizations have traditionally employed PBX systems (such as voice conferencing) by using open standards, such as TCP/IP, H.323, and MGCP. Because of its open architecture, Cisco CallManager allows easy deployment of robust voice applications and the integration of telephony systems with intranet applications. Cisco CallManager uses Microsoft Internet Information Server (IIS) to allow remote administration with a standard web browser.

Cisco CallManager provides the basic services required for IP phones, such as mapping IP addresses to specific devices and extensions, and for managing Cisco Access gateways. Cisco CallManager supports TAPI (Telephony Application Programming Interface) and JTAPI (Java Telephony Application Programming Interface) for deploying Cisco IP IVR and other applications. Telephony applications written to the JTAPI specification can gain the cross-platform benefits of Java. Cisco CallManager supports Cisco IP IVR by providing services to the application server through JTAPI.

## Application Server

The application server is a Cisco Media Convergence Server (Cisco MCS) that runs the Cisco Application Engine (which executes the scripts), and Cisco IP IVR applications, such as Cisco IP AA. You can deploy the Cisco Application Engine and Cisco CallManager on the same server, or on separate servers to handle more calls.

The Cisco Application Engine uses JTAPI to request and receive services from Cisco CallManager. The application engine is implemented as a Windows NT service that supports multiple applications. The application engine is distributed on a CD with the other components and applications that are included with the Cisco IP Telephony solutions package. You can buy solutions that include the server, Cisco IP IVR, and, Cisco CallManager.

The server includes three subsystems: JTAPI, Database, and ICM.

- JTAPI—manages the connection between Cisco CallManager and the application server
- Database—handles the connection between the application server and the enterprise database
- ICM—manages the connection between the application server and Cisco Intelligent Contact Manager (Cisco ICM). The ICM subsystem is only used if you are using Cisco IP IVR with Cisco IP Contact Center, which includes Cisco ICM.

## Administration

You use the Cisco CallManager Administrator to administer and configure some features for Cisco CallManager, such as adding users and configuring CTI route points and CTI ports. Cisco CallManager Administrator is implemented as a series of web pages on the Cisco CallManager server.

You can administer the Application Engine, Cisco IP IVR applications, and VRU scripts using the Application Administrator web pages. The Application Administration web pages are included on the Cisco IP Telephony solutions CD and are installed when you install the application engine.

## Cisco IP Telephony Directory

The Cisco IP Telephony Solution uses an LDAP directory for storing user profile information, Cisco IP IVR applications and VRU scripts, and network-specific configuration information, such as the location of network resources. Storing application logic and configuration information in an LDAP directory allows you to load and extract the application on any application engine in the network.

The Cisco IP Telephony system includes a default LDAP directory server, which stores application configuration information for Cisco CallManager and the application framework. (The default directory is established as part of the Cisco CallManager installation and configuration.) You can use the default directory server to manage your user profiles, or you can integrate your Cisco IP Telephony application system with an existing enterprise Netscape Directory Server 4.0 or Microsoft Active Directory system. If you use the default directory, you can migrate later to a production directory server, using the MetaLink utility, which is included with the plug-in.

## Cisco Application Editor

The Cisco Application Editor lets application designers modify existing applications, such as the Cisco IP AA, or to use steps for creating new applications. The Cisco Application Editor is a visual scripting tool that allows designers to drag from a palette and drop them in the main design pane.

The Cisco Application Editor also supports a “debug mode” that allows an administrator to register a configured application for debugging.

Once testing completes, the designer uploads the completed application to the directory, from where it is deployed to production Cisco IP Telephony application engine on the network. The application designer can also download existing scripts from the LDAP Directory, modify and test them, then upload the revised scripts.

## Running Cisco IP Telephony Applications

The application server is a Cisco Media Convergence Server (Cisco MCS) that runs the Cisco Application Engine (which executes the scripts), and Cisco IP IVR applications, such as Cisco IP AA. You can deploy the application engine (along

with the other application server components) and Cisco CallManager on the same server, or on separate servers for higher port densities. (Port density is the number of ports that an application server can support.)

During run time, the application engine loads application logic and configuration information from the Cisco IP Telephony directory. Then, it checks for file corruption and improper construction, and executes each step in the sequence specified in the application.

You use a standard web browser as an administrative client to enable and disable an application and to monitor Cisco Application Engine activity. Real-time reports include total system activity and statistics regarding specific applications.

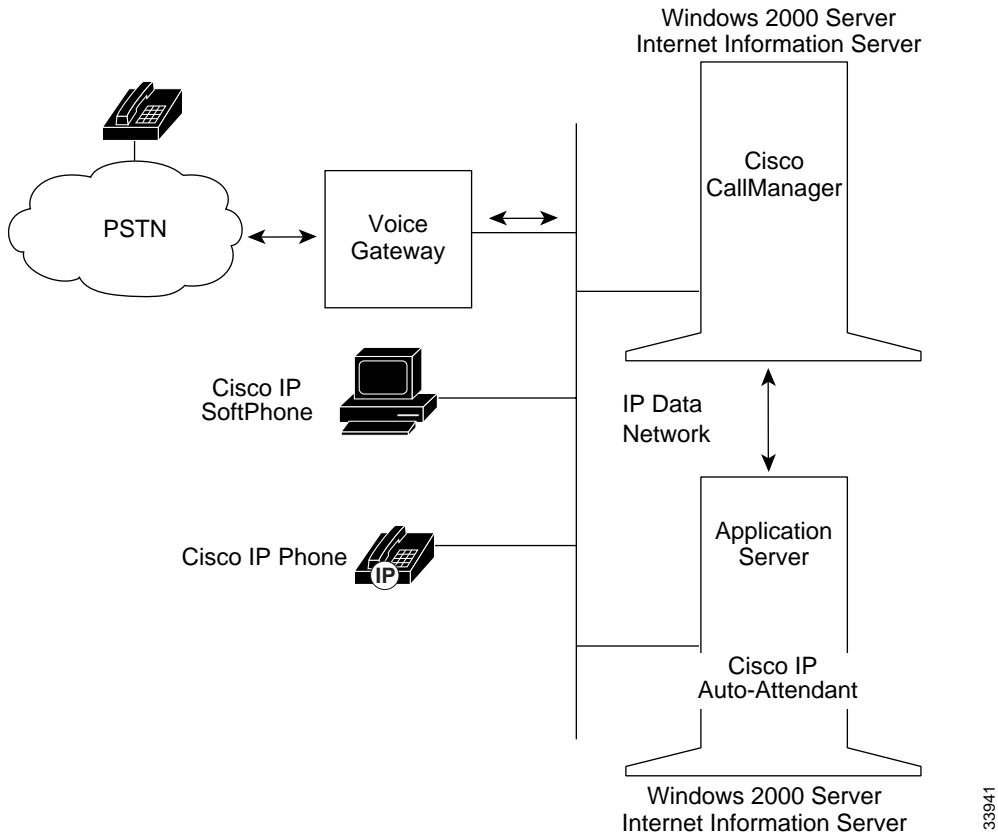
Cisco IP AA is an application built with Application Editor steps that provides a simple call answering and forwarding service. Cisco IP AA provides dial-by-name via telephone keypad mapping, which allows callers to find an extension by the first few characters of the associated user name. More details are provided in “Cisco IP AutoAttendant” section on page 1-7 and “How the Cisco IP AutoAttendant Works” section on page 1-9.

Cisco IP IVR includes steps which the developer can use to create application logic with the Cisco Application Editor. Cisco IP IVR includes steps that you can use to access an enterprise database, provide menu options, reference XML documents, interact with Cisco IP Contact Center, and other useful functions.

## Cisco IP AutoAttendant

The Cisco IP AA application, illustrated in Figure 1-2, works with Cisco CallManager to receive calls on specific telephone extensions and to allow the caller to select an appropriate extension.

Figure 1-2 Using the Cisco IP AutoAttendant



The Cisco IP AA application provides the following functionality:

- Answers a call
- Plays a user-configurable Welcome prompt
- Plays a Main Menu prompt, asking the user to perform one of three actions:
  - Press “0” for the operator
  - Press “1” to enter an extension number
  - Press “2” to spell by name

- If the user chooses to type letters (option 2), the application compares the letters entered with the available extensions.
  - If one matches, the system prompts the user for confirmation of the name, and transfers the call to that user’s primary extension.
  - If there is more than one match, the application prompts the user to select the correct extension.
  - If there are too many matches, the application prompts the user to enter more characters.
- When the user has specified the destination, the application transfers the call.
  - If the line is busy, does not answer, or is not in service, the application informs the user accordingly and replays the Main Menu prompt.

## How the Cisco IP AutoAttendant Works

The Cisco IP AA is a software application built using Cisco Application Editor steps and based on the Cisco IP IVR framework. Applications built using the Cisco IP Telephony application framework are composed of a series of application steps written in Java 1.1 and packaged as .jar files.

You use a standard web browser as an administrative client to start and stop an application and to monitor application engine activity.

Cisco IP AA provides a simple call answering and forwarding service. Cisco IP AA provides telephone keypad mapping, which allows callers to identify an extension by the first few characters of the associated user name.

You can configure one or more instances of the Cisco IP AA on a Cisco Application Engine using the Application Administration server web pages.

To be recognized by the Application Engine as an AutoAttendant, the AutoAttendant is stored in a file named “aa.aef.”

You administer Cisco IP AA and custom Cisco IP IVR applications using the Application Administration server, which is a web application that is installed on the application server. Network users with the required privileges can perform remote administration using Microsoft Internet Explorer 5.x (or later).

The Application Administration web pages let you start and stop applications, configure system parameters, and monitor application activity.

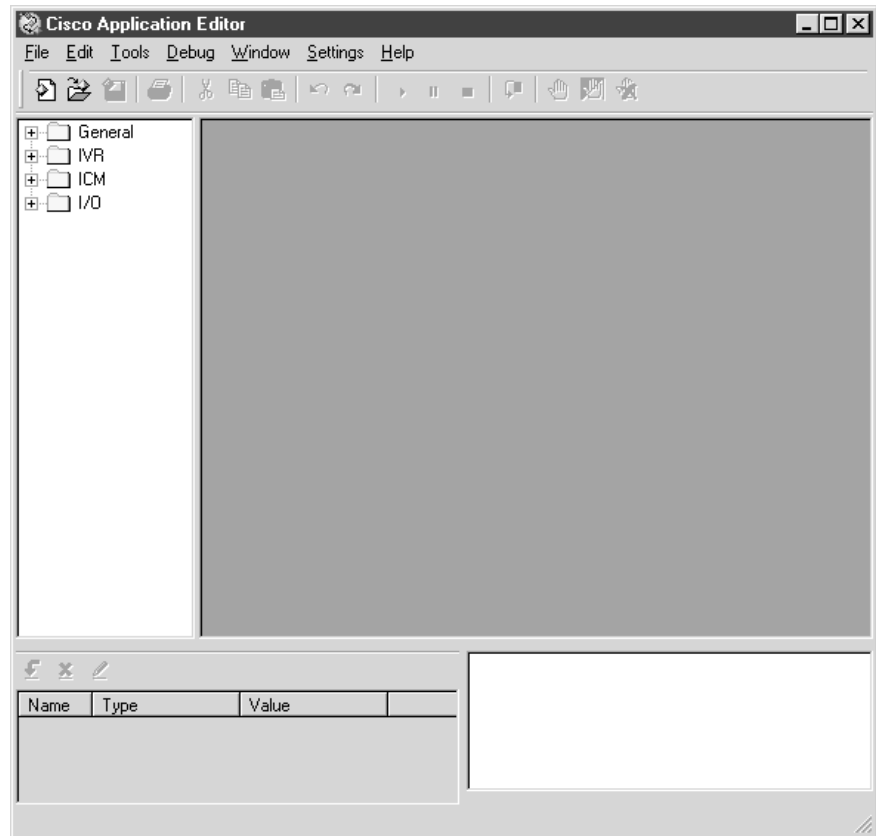
When you run an application, the application engine loads application logic and configuration information from the directory, and executes application steps in the order specified in the application.

## Using Application Editor Steps

Cisco Application Editor steps are included with Cisco IP IVR. The steps allow you to create interactive voice response applications, including accessing customer information from enterprise databases.

The Cisco Application Editor, illustrated in Figure 1-3 is a visual programming environment that lets you assemble steps into Cisco IP IVR applications, using the resources and infrastructure provided by the Cisco IP Telephony application framework.

Figure 1-3 The Cisco Application Editor



You do not need to understand Java programming to use the Cisco Application Editor and to construct Cisco IP IVR applications. You simply assemble a script by dragging the step icons from the Palette on the left of the workspace to the design window on the right side of the workspace. The Cisco Application Editor automatically supplies the code required to connect the different steps. You then validate, create variables, and debug the application, viewing the output in the lower right window. Finally, you can upload the finished application to your Cisco IP Telephony Directory or application engine.

The Cisco IP Telephony application framework manages the details of interfacing with Cisco CallManager, and managing access to the other required network resources.

You configure the location and authentication information for network resources, stored in the User Preferences (LDAP) directory, using the Application Administration pages that are included with the Cisco IP Telephony application engine.

## Using the ICM VRU Interface

Cisco Intelligent Contact Manager (Cisco ICM) software provides a central control system that directs calls to various other human and automated systems, such as interactive voice response units (VRU), and automatic call distribution (ACD) systems. Cisco ICM uses scripts that can direct calls based on various criteria, such as time of day or the availability of subsystems. The Cisco Application Engine ICM VRU interface allows the integration of Cisco IP Telephony components with Cisco ICM.

**Note**

---

If you are not using Cisco IP Contact Center and Cisco ICM, you do not need to use the ICM VRU Interface.

---

The ICM VRU interface allows Cisco ICM scripts to invoke Cisco Application Editor steps and logic from the Cisco Application Engine. As a result, calls are handled centrally by the Cisco ICM and directed to the VoIP network based on user entries and stored information, retrieved and manipulated using application editor steps.

Figure 1-4 illustrates how the different components of this solution work together. When a call is received by Cisco ICM from a POTS device, ICM applies the rules contained in the running ICM script to determine how to route the call. When integrated with the Cisco IP Telephony framework, ICM scripts send requests through the ICM VRU interface of the Cisco Application Engine, through a peripheral gateway. When the application engine receives this message, it determines which ICM VRU script to run, or where the call should be redirected, based on the information received. An ICM VRU script can contain a number of application steps that together perform a useful task, like prompting callers for input.

When the ICM VRU script completes processing, the results of the process are passed back through the peripheral gateway to the Cisco ICM, which then executes the next command in the ICM script.



Perform the following steps to implement the Cisco IP Telephony system components and to deploy the applications you need for your organization.

#### Procedure

---

- Step 1** Install and configure Cisco IP IVR.  
Do this to install an instance of Cisco IP IVR, including the Cisco Application Engine, the Cisco IP AA, and the Application Administration pages on the application server.
- Step 2** Create an application script with the Cisco Application Editor.  
Do this to prepare the application for deployment. Refer to Chapter 3, “Creating an Application.”
- Step 3** Configure Cisco CallManager (and the Cisco ICM system, if you are using Cisco IP IVR with Cisco ICM).  
Do this to create CTI ports, CTI route points, and users. Refer to the *Cisco CallManager Administration Guide*.
- Step 4** Upload application script to the Cisco IP Telephony Directory.  
Do this to register your application so that it is ready to run from anywhere in the network that the directory server is available. Refer to Chapter 4, “Administering Applications.”
- Step 5** Set up and test your application.  
Do this to verify the application setup. See Chapter 4, “Administering Applications.”
-