CISCO SYSTEMS

# Cisco IP Phone Services Application Development Notes

Release 4.1(3)

**Corporate Headquarters**
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Text Part Number: OL-7320-02

CONTENTS

# Preface

Use this document with Cisco CallManager 4.1(3) to develop and deploy customized client services for the Cisco IP Phones that support Cisco IP Phone services.

**Note** Developers using this guide should join the Cisco Developer Support Program because standard Cisco TAC support is limited to the Cisco AVVID installation, configuration, and Cisco-developed applications. This program provides a consistent level of dependable support while leveraging Cisco interfaces in your development projects. For more information about the program and how to join, contact us at developer-support@cisco.com.

## Contents

This document covers the following topics:

## Audience

This document provides the information needed for eXtensible Markup Language (XML) and X/Open System Interface (XSI) programmers and system administrators to develop and deploy new services.

## Organization

This document comprises the following sections.

| Chapter | Description |
|---|---|
| Chapter 1, "Overview" | Provides an overview of the Cisco IP Phone services for developers. |
| Chapter 2, "CiscoIPPhone XML Objects" | Describes the general behavior and usage of each XML object. |
| Chapter 3, "Internal URI Features" | Describes how to implement embedded features on Cisco IP Phones. |
| Chapter 4, "Cisco IP Services Software Development Kit (SDK)" | Provides a list of the components used in the Cisco IP Services Software Development Kit (SDK) and the sample services requirements. |
| Chapter 5, "HTTP Requests and Header Settings" | Provides a procedure on handling HTTP client requests, definitions for HTTP header elements, identifies the capabilities of the requesting IP phone client, and defines the Accept header. |
| Chapter 6, "IP Phone Service Administration and Subscription" | Describes how to add and administer Cisco IP Phone services through Cisco CallManager Administration. |
| Chapter 7, "Troubleshooting Cisco IP Phone Service Applications" | Provides troubleshooting tips, XML parsing errors, and error messages. |

| Chapter | Description |
|---|---|
| Chapter 8, "DeviceListX Report" | Describes how the report provides a list of the services-capable devices along with basic information about the device to identify or classify the devices based on specific criteria |
| Appendix A, "CiscoIPPhone XML Object Quick Reference" | Provides a quick reference of the CiscoIPPhone XML objects and the definitions that are associated with each. |
| Appendix B, "Cisco IP Phone XML Schema File" | Provides the CiscoIPPhone.xsd file. |

## Related Documentation

The following documents provide further information:

- *Cisco CallManager Administration Guide* (also available in the online help). Refer to the chapter on configuring Cisco IP Phone services.
- *Cisco CallManager System Guide* (also available in the online help).
- *Cisco IP Phone 7960/7940 Quick Start Guide*
  Provides instructions for users on subscribing to phone services.
- *Cisco IP Phone Administration Guide for Cisco CallManager*
  Provides administration information for Cisco IP Phones.
- *CiscoURLProxy ActiveX Component*
  Provided with the Cisco IP Services SDK.
- *LDAP Search COM Server Programming Guide*
  Provided with the Cisco IP Services SDK.
- *CipImage Release Notes*
  Provided with the Cisco IP Services SDK.

# Obtaining Documentation

Cisco documentation and additional literature are available on Cisco.com. Cisco also provides several ways to obtain technical assistance and other technical resources. These sections explain how to obtain technical information from Cisco Systems.

## Cisco.com

You can access the most current Cisco documentation at this URL:

http://www.cisco.com/univercd/home/home.htm

You can access the Cisco website at this URL:

http://www.cisco.com

You can access international Cisco websites at this URL:

http://www.cisco.com/public/countries_languages.shtml

## Ordering Documentation

You can find instructions for ordering documentation at this URL:

http://www.cisco.com/univercd/cc/td/doc/es_inpck/pdi.htm

You can order Cisco documentation in these ways:

- Registered Cisco.com users (Cisco direct customers) can order Cisco product documentation from the Ordering tool:

  http://www.cisco.com/en/US/partner/ordering/index.shtml

- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco Systems Corporate Headquarters (California, USA) at 408 526-7208 or, elsewhere in North America, by calling 800 553-NETS (6387).

## Documentation Feedback

You can send comments about technical documentation to bug-doc@cisco.com.

You can submit comments by using the response card (if present) behind the front cover of your document or by writing to the following address:

Cisco Systems
Attn: Customer Document Ordering
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

## Obtaining Technical Assistance

For all customers, partners, resellers, and distributors who hold valid Cisco service contracts, Cisco Technical Support provides 24-hour-a-day, award-winning technical assistance. The Cisco Technical Support Website on Cisco.com features extensive online support resources. In addition, Cisco Technical Assistance Center (TAC) engineers provide telephone support. If you do not hold a valid Cisco service contract, contact your reseller.

### Cisco Technical Support Website

The Cisco Technical Support Website provides online documents and tools for troubleshooting and resolving technical issues with Cisco products and technologies. The website is available 24 hours a day, 365 days a year at this URL:

http://www.cisco.com/techsupport

Access to all tools on the Cisco Technical Support Website requires a Cisco.com user ID and password. If you have a valid service contract but do not have a user ID or password, you can register at this URL:

http://tools.cisco.com/RPF/register/register.do

## Submitting a Service Request

Using the online TAC Service Request Tool is the fastest way to open S3 and S4 service requests. (S3 and S4 service requests are those in which your network is minimally impaired or for which you require product information.) After you describe your situation, the TAC Service Request Tool automatically provides recommended solutions. If your issue is not resolved using the recommended resources, your service request will be assigned to a Cisco TAC engineer. The TAC Service Request Tool is located at this URL:

http://www.cisco.com/techsupport/servicerequest

For S1 or S2 service requests or if you do not have Internet access, contact the Cisco TAC by telephone. (S1 or S2 service requests are those in which your production network is down or severely degraded.) Cisco TAC engineers are assigned immediately to S1 and S2 service requests to help keep your business operations running smoothly.

To open a service request by telephone, use one of the following numbers:

Asia-Pacific: +61 2 8446 7411 (Australia: 1 800 805 227)
EMEA: +32 2 704 55 55
USA: 1 800 553 2447

For a complete list of Cisco TAC contacts, go to this URL:

http://www.cisco.com/techsupport/contacts

## Definitions of Service Request Severity

To ensure that all service requests are reported in a standard format, Cisco has established severity definitions.

Severity 1 (S1)—Your network is “down,” or there is a critical impact to your business operations. You and Cisco will commit all necessary resources around the clock to resolve the situation.

Severity 2 (S2)—Operation of an existing network is severely degraded, or significant aspects of your business operation are negatively affected by inadequate performance of Cisco products. You and Cisco will commit full-time resources during normal business hours to resolve the situation.

Severity 3 (S3)—Operational performance of your network is impaired, but most business operations remain functional. You and Cisco will commit resources during normal business hours to restore service to satisfactory levels.

Severity 4 (S4)—You require information or assistance with Cisco product capabilities, installation, or configuration. There is little or no effect on your business operations.

## Obtaining Additional Publications and Information

Information about Cisco products, technologies, and network solutions is available from various online and printed sources.

- Cisco Marketplace provides a variety of Cisco books, reference guides, and logo merchandise. Visit Cisco Marketplace, the company store, at this URL:

  http://www.cisco.com/go/marketplace/

- The Cisco *Product Catalog* describes the networking products offered by Cisco Systems, as well as ordering and customer support services. Access the Cisco Product Catalog at this URL:

  http://cisco.com/univercd/cc/td/doc/pcat/

- *Cisco Press* publishes a wide range of general networking, training and certification titles. Both new and experienced users will benefit from these publications. For current Cisco Press titles and other information, go to Cisco Press at this URL:

  http://www.ciscopress.com

- *Packet* magazine is the Cisco Systems technical user magazine for maximizing Internet and networking investments. Each quarter, Packet delivers coverage of the latest industry trends, technology breakthroughs, and Cisco products and solutions, as well as network deployment and troubleshooting tips, configuration examples, customer case studies, certification and training information, and links to scores of in-depth online resources. You can access Packet magazine at this URL:

  http://www.cisco.com/packet

- *iQ Magazine* is the quarterly publication from Cisco Systems designed to help growing companies learn how they can use technology to increase revenue, streamline their business, and expand services. The publication identifies the challenges facing these companies and the technologies to help solve them, using real-world case studies and business strategies to help readers make sound technology investment decisions. You can access iQ Magazine at this URL:

http://www.cisco.com/go/iqmagazine

- *Internet Protocol Journal* is a quarterly journal published by Cisco Systems for engineering professionals involved in designing, developing, and operating public and private internets and intranets. You can access the Internet Protocol Journal at this URL:

  http://www.cisco.com/ipj

- World-class networking training is available from Cisco. You can view current offerings at this URL:

  http://www.cisco.com/en/US/learning/index.html

# Overview

You can use Cisco IP Phones to deploy customized client services with which users can interact via the keypad and display. Services deploy using the HTTP protocol from standard web servers, such as Microsoft IIS.

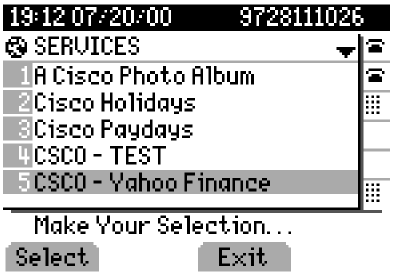
Users access these features using the **services** and **directories** buttons or menu options (availability varies by phone model). When a user presses the **services** button (or chooses the **services** menu item), a menu of configured services displays. The user then chooses a service from the list, and the phone displays the service.

The following list gives typical services that might be supplied to a phone:

- Weather
- Stock information
- Contact information
- Company news
- To-do lists
- Daily schedule

Figure 1-1 shows a sample text menu.

*Figure 1-1 Cisco IP Phone Text Menu Sample*



19:12 07/20/00 9728111026

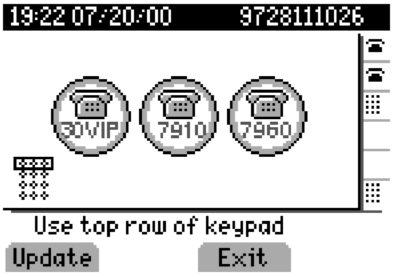
SERVICES

1 A Cisco Photo Album
2 Cisco Holidays
3 Cisco Paydays
4 CSCO - TEST
5 CSCO - Yahoo Finance

Make Your Selection...

Cisco IP Phones can also display graphic menus, as shown in Figure 1-2.

*Figure 1-2 Graphic Menu on a Cisco IP Phone Sample*



19:22 07/20/00 9728111026

Use top row of keypad

Phone users can navigate a text menu by using the Navigation button followed by the Select softkey, or by using the numeric keypad to enter a selection directly. Graphic menus currently do not support cursor-based navigation; users simply enter a number using the DTMF keypad.

When a menu selection is made, the Cisco IP Phone acts on it by using its HTTP client to load a specific URL. The return type from this URL can be plain text or one of the CiscoIPPhone XML objects. The object loads and the user interacts with the object.

Figure 1-3 and Figure 1-4 show typical displays that result from selecting a service. Figure 1-3 shows a stock quote that was generated using plain text, and Figure 1-4 displays a graphic image.

*Figure 1-3 Plan Text Display Example*

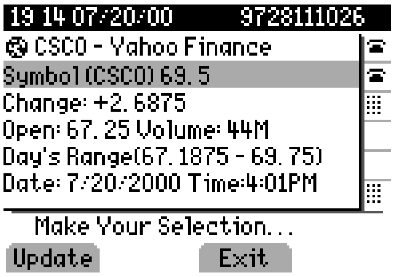
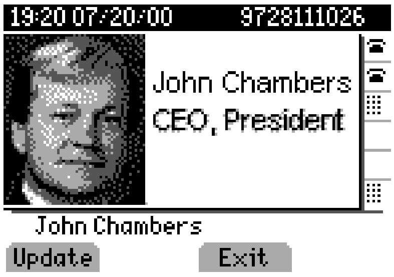

```
19 14 07/20/00        9728111026
CSCO - Yahoo Finance
Symbol (CSCO) 69.5
Change: +2.6875
Open: 67.25 Volume: 44M
Day's Range(67.1875 - 69.75)
Date: 7/20/2000 Time:4:01PM
Make Your Selection...
```

*Figure 1-4 Graphic Image Display Example*


```
19:20 07/20/00        9728111026
John Chambers
CEO, President
John Chambers
```

Cisco CallManager limits Cisco IP Phone service activity to a specific Services pane in the Cisco IP Phone display. A service cannot modify the top line of the phone display, which contains the time, date, and primary extension. A service cannot overwrite the bottom line of the display, which contains softkey definitions. The pane that displays the service sits flush with the left side of the display, and enough of the right side of the display remains intact to ensure that users can see the status of phone lines.

**Note** **HTML Disclaimer**: Phone service developers must take into consideration that the phone is *not* a web browser and cannot parse HTML. Although content is delivered to the phone through HTTP messages by using a web server, keep in mind that the content is not HTML. All content comes either as plain text or packaged in proprietary XML wrappers.

# CiscoIPPhone XML Objects

The following sections describe the general behavior and use of XML objects:

- Understanding Object Behavior
- XML Object Definitions
- Custom Softkeys
- XML Considerations

For further details on XML object syntax, refer to the Cisco IP Phone XML schema file in Appendix B, "Cisco IP Phone XML Schema File."

## Understanding Object Behavior

Creating interactive service applications is relatively easy when you understand the XML objects that are defined for Cisco IP Phones and the behavior that each object generates.

Regarding services, understand that the phone does not have any concept of a state when it loads an XML page. Cisco IP Phones can use HTTP to load a page of content in many different places, starting when the **services** button is pressed. Regardless of what causes the phone to load a page, the phone always behaves appropriately after it loads a page.

Appropriate behavior depends solely on the type of data that has been delivered in the page. This section of the document discusses the supported XML display types and how they work with Cisco IP Phones.

The web server must deliver the XML pages with a MIME type of text/xml. However, the exact mechanism required varies according to the type of web server that you are using and the server side mechanism that you are using to create your pages (for example, static files, JavaScript, CGI, and so on). See Chapter 5, "HTTP Requests and Header Settings" for more information.

Table 2-1 shows the supported XML objects for Release 4.1(3).

*Table 2-1 XML Objects Supported for Release 4.1(3) Cisco IP Phone Services SDK*

| Phone Model XML Object | 7905 G/ 7912G | 7920 | 7940G / 7960G | 7970G / 7971G-GE IP Communicator |
|---|---|---|---|---|
| CiscoIPPhoneMenu | X | X | X | X |
| CiscoIPPhoneText | X | X | X | X |
| CiscoIPPhoneInput | X | X | X | X |
| CiscoIPPhoneDirectory | X | X | X | X |
| CiscoIPPhoneImage | | X[1] | X | X |
| CiscoIPPhoneImageFile | | | | X |
| CiscoIPPhoneGraphicMenu | | X[1] | X | X |
| CiscoIPPhoneGraphicFileMenu | | | | X |
| CiscoIPPhoneIconMenu | X[2] | X | X | X |
| CiscoIPPhoneIconFileMenu | | | | X[3] |
| CiscoIPPhoneStatus | | | X | X |
| CiscoIPPhoneStatusFile | | | | X[3] |
| CiscoIPPhoneExecute | X | X | X | X |
| CiscoIPPhoneResponse | X | X | X | X |
| CiscoIPPhoneError | X | X | X | X |

1. The Cisco IP Phone 7920 has only a 128-by-59 display with 2 grayscale images clipping the graphic equally on both sides and providing vertical scrolling. When an image with 4 grayscale settings occurs (<Depth>2</Depth>), the phone equally splits them into 2 grayscale settings (0-1 get treated as 0 and 2-3 get treated as 1).
2. The Cisco IP Phone 7905 and 7912 do not support CIP images; therefore, all Icons get ignored and do not display.
3. The Cisco IP Phone 7970G and 7971G-GE require firmware version 7.01 or higher to support this object, and Cisco IP Communicator requires software version 2.01 or higher.

## XML Object Definitions

The following sections provide definitions and descriptions of each CiscoIPPhone XML object:

- CiscoIPPhoneMenu
- CiscoIPPhoneText
- CiscoIPPhoneInput
- CiscoIPPhoneDirectory
- CiscoIPPhoneImage
- CiscoIPPhoneImageFile
- CiscoIPPhoneGraphicMenu
- CiscoIPPhoneGraphicFileMenu
- CiscoIPPhoneIconMenu
- CiscoIPPhoneIconFileMenu
- CiscoIPPhoneStatus
- CiscoIPPhoneStatusFile
- CiscoIPPhoneExecute
- CiscoIPPhoneResponse
- CiscoIPPhoneError

## CiscoIPPhoneMenu

A menu on the phone comprises a list of text items, one per line. Users choose individual menu items by using the same mechanisms that are used for built-in menus in the phone as described in Chapter 1, "Overview".

### Definition

```
<CiscoIPPhoneMenu>
  <Title>Title text goes here</Title>
  <Prompt>Prompt text goes here</Prompt>
  <MenuItem>
   <Name>The name of each menu item</Name>
   <URL>The URL associated with the menu item</URL>
  </MenuItem>
</CiscoIPPhoneMenu>
```

**Note** The Name field under the `<MenuItem>` supports a maximum of 64 characters. This field can also accept two carriage returns to allow the MenuItem name to span three lines on the display.

The XML format allows you to specify a title and prompt that are used for the entire menu, followed by a sequence of `MenuItem` objects. Cisco IP Phones allow a maximum of 100 `MenuItems`. Each `MenuItem` includes a `Name` and an associated `URL`.

When a menu is loaded, the phone behaves the same as for built-in phone menus. The user navigates through the list of menu items and eventually chooses one by using either the Select softkey or the DTMF keys.

After the user chooses a menu option, the phone generates an HTTP request for the page with the URL or executes the uniform resource identifiers (URIs) that are associated with the menu item.

## CiscoIPPhoneText

The `CiscoIPPhoneText` XML object displays ordinary 8-bit ASCII text on the phone display. The `<Text>` message must not contain any control characters, except for carriage returns, line feeds, and tabs. The Cisco IP Phone firmware controls all other pagination and wordwrap issues.

**Note** The Cisco IP Phone supports the full ISO 8859-1 (Latin 1) character set.

### Definition

```
<CiscoIPPhoneText>
  <Title>Title text goes here</Title>
  <Prompt>The prompt text goes here</Prompt>
  <Text>The text to be displayed as the message body goes here</Text>
</CiscoIPPhoneText>
```

Two optional fields can appear in the XML message:

- The first optional field, `Title`, defines text that displays at the top of the display page. If a `Title` is not specified, the `Name` field of the last chosen `MenuItem` displays in the `Title` field.
- The second optional field, `Prompt`, defines text that displays at the bottom of the display page. If a `Prompt` is not specified, Cisco CallManager clears the prompt area of the display pane.

Many XML objects that are described in this document also have `Title` and `Prompt` fields. These fields normally behave identically to behavior described in this section.

**Note** **Non-XML Text**: This document only describes the supported CiscoIPPhone XML objects. You can also deliver plain text via HTTP. Pages that are delivered as MIME type text/html behave exactly the same as XML pages of type `CiscoIPPhoneText`. One important difference is that you cannot include a title or prompt.

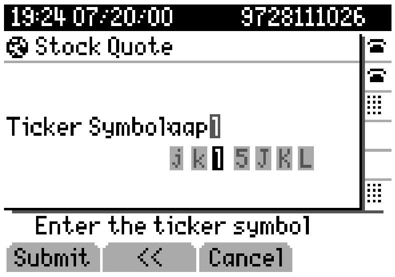
**Note** **Keypad navigation**: Cisco IP Phones allow navigation to a specific line in a menu by pressing numeric DTMF keys. When a menu is on the display, the number for selecting the menu is on the left.

When normal text displays, the numbers do not display on the left side of the screen, but the navigation capability still exists. So, a carefully written text service display can take advantage of this capability.

## CiscoIPPhoneInput

When a Cisco IP Phone receives an XML object of type `CiscoIPPhoneInput`, it constructs an input form and displays it. The user then enters data into each input item and sends the parameters to the target URL. Figure 2-1 shows a sample display that is receiving input from a user.

*Figure 2-1 Sample User Input Display*



### Definition

```
<CiscoIPPhoneInput>
  <Title>Directory title goes here</Title>
  <Prompt>Prompt text goes here</Prompt>
  <URL>The target URL for the completed input goes here</URL>
  <InputItem>
   <DisplayName>Name of the input field to display</DisplayName>
   <QueryStringParam>The parameter to be added to the target
URL</QueryStringParam>
   <DefaultValue>The default display name</DefaultValue>
   <InputFlags>The flag specifying the type of allowable
input</InputFlags>
  </InputItem>
</CiscoIPPhoneInput>
```

The `Title` and `Prompt` tags in the object delimit text are used in the same way as the identical fields in the other CiscoIPPhone XML objects.

The `URL` tag delimits the URL to which the input results are sent. The actual HTTP request sent to this server specifies the URL with a list of parameters that are appended to it as a query string. The parameters include Name/Value pairs, one for each input item.

**Note** CiscoIPPhoneInput objects do not use the HTTP POST method.

The `InputItem` tag delimits each item in the list. The number of `InputItems` must not exceed five. Each input item includes a `DisplayName`, which is the prompt that is written to the display for that particular item. Each item also has a `QueryStringParam`, which is the name of the parameter that is appended to the URL when it is sent out after input is complete. Each input item can also use the `DefaultValue` tag to set the default value to be displayed.

The final attribute for each input item comprises a set of `InputFlags`. The following table describes the input types that are currently defined.

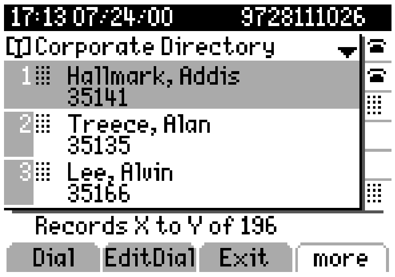
| InputFlag | Description |
|---|---|
| A | Plain ASCII text—use the DTMF keypad to enter text that consists of uppercase and lowercase letters, numbers, and special characters. |
| T | Telephone number—enter only DTMF digits for this field. The acceptable input includes numbers, #, and *. |
| N | Numeric—enter numbers as the only acceptable input. |
| E | Equation—enter numbers and special math symbols. |
| U | Uppercase—enter uppercase letters as the only acceptable input. |
| L | Lowercase—enter lowercase letters as the only acceptable input. |
| P | Password field—enter individual characters using the standard keypad-repeat entry mode. The system automatically converts accepted characters into an asterisk, keeping the entered value private. **Note** P specifies the only `InputFlag` that works as a modifier. For example, specify a value of "AP" in the `InputFlag` field to use plain ASCII as the input type and to mask the input as a password by using an asterisk (*). |

During text entry, Cisco IP Phones display softkeys to assist users with text entry. Users can navigate between fields with the vertical scroll button that is used to navigate menus, and so on.

## CiscoIPPhoneDirectory

The phone originally incorporated the `CiscoIPPhoneDirectory` XML object to support the Directory operation of Cisco IP Phones, but it is available for your development purposes also. Figure 2-2 shows how an XML `CiscoIPPhoneDirectory` object displays on the phone.

*Figure 2-2 CiscoIPPhoneDirectory Object Display Sample*


17:13 07/24/00 9728111026
Corporate Directory
1 Hallmark, Addis
35141
2 Treece, Alan
35135
3 Lee, Alvin
35166
Records X to Y of 196

### Definition

```
<CiscoIPPhoneDirectory>
  <Title>Directory title goes here</Title>
  <Prompt>Prompt text goes here</Prompt>
  <DirectoryEntry>
    <Name>The name of the directory entry</Name>
    <Telephone>The telephone number for the entry</Telephone>
  </DirectoryEntry>
</CiscoIPPhoneDirectory>
```

**Note** For the directory listing, the Cisco IP Phone displays the appropriate softkeys that are needed to dial the numbers that are listed on the display. The softkeys include the Edit Dial softkey, which allows users to insert access codes or other necessary items before dialing.

The `Title` and `Prompt` tags in the XML object have the usual semantics. A single `CiscoIPPhoneDirectory` object can contain a maximum of 32 `DirectoryEntry` objects. If more than 32 entries must be returned, use multiple `CiscoIPPhoneDirectory` objects in subsequent HTTP requests.

## Custom Directories

You can use the Cisco CallManager enterprise parameter, “URL Directories” and CiscoIPPhone XML objects to display custom directories. The “URL Directories” points to a URL that returns a `CiscoIPPhoneMenu` object that extends the **directories** menu. The request for “URL Directories” must return a valid `CiscoIPPhoneMenu` object, even if has no `DirectoryEntry` objects.

To create a custom directory, use the following optional objects in the order in which they are listed:

1. Use the `CiscoIPPhoneInput` XML object to collect search criteria.
2. Use the `CiscoIPPhoneText` XML object to display status messages or errors.
3. Use the `CiscoIPPhoneDirectory` XML object to return a list of directory entries that can be dialed.

You can omit the `CiscoIPPhoneInput` or `CiscoIPPhoneText` objects. You can display multiple `CiscoIPPhoneDirectory` objects by specifying an HTTP refresh header that points to the URL of the next individual directory object, which the user accesses by pressing the Next softkey on the phone.

## CiscoIPPhoneImage

The CiscoIPPhoneImage provides a bitmap display with a 133 x 65 pixel pane that is available to access services. Each pixel includes four grayscale settings. A value of three (3) displays as black, and a value of zero (0) displays as white.

**Note** The phone uses an LCD display, which inverts the palette.

The `CiscoIPPhoneImage` XML type lets you use the Cisco IP Phone display to present graphics to the user.

### Definition

```
<CiscoIPPhoneImage>
  <Title>Image title goes here</Title>
  <Prompt>Prompt text goes here</Prompt>
  <LocationX>Position information of graphic</LocationX>
  <LocationY>Position information of graphic</LocationY>
  <Width>Size information for the graphic</Width>
  <Height>Size information for the graphic</Height>
  <Depth>Number of bits per pixel</Depth>
  <Data>Packed Pixel Data</Data>
   <SoftKeyItem>
    <Name>Name of the soft key</Name>
    <URL>URL of soft key</Name>
    <Position>Numerical position of the soft key</Position>
   </SoftKeyItem>
</CiscoIPPhoneImage>
```

The `CiscoIPPhoneImage` object definition includes two familiar elements: `Title` and `Prompt`. These elements serve the same purpose as they do in the other CiscoIPPhone XML objects. The `Title` displays at the top of the page, and the `Prompt` displays at the bottom.

Use `LocationX` and `LocationY` to position the graphic on the phone display. Position the upper, left corner of the graphic at the pixel defined by these two parameters. Setting the X and Y location values to (0, 0) positions the graphic at the upper, left corner of the display. Setting the X and Y location values to (-1, -1) centers the graphic in the services pane of the phone display.

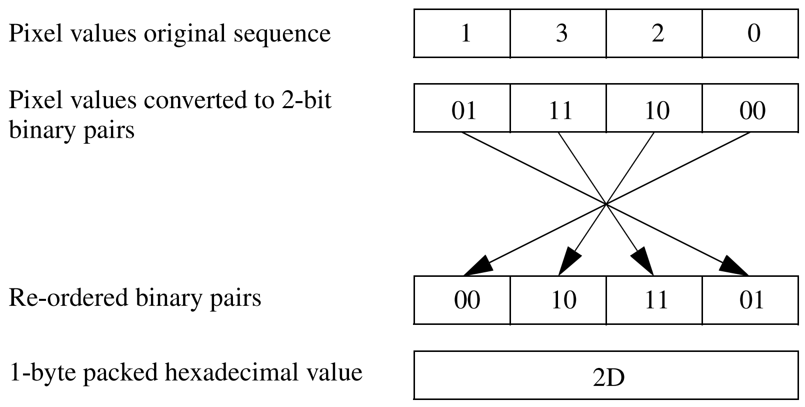
Use `Width` and `Height` to size the graphic. If the values do not match with the pixel stream specified in the `Data` field, results will be unpredictable incorrect.

`Depth` specifies the number of bits per pixel. Cisco IP Phones support a maximum value of 2. A bit depth of 1 is black and white.

The `Data` tag delimits a string of hexadecimal digits that contain the packed value of the pixels in the display. In the Cisco IP Phone, each pixel has only four possible values, which means that you can pack four pixels into a single byte. A pair of hexadecimal digits represents each byte.

Figure 2-3 provides an example of the mechanics of pixel packing. Scanning from left to right in the display, the illustration shows the process for packing consecutive pixel values of 1, 3, 2, and 0. First, the pixels get converted to 2-bit binary numbers. Then, the binary pairs get re-ordered in sets of four to create a single re-ordered byte, which two hexadecimal digits represent.

*Figure 2-3 Packed Pixel Translation Example*


Pixel values original sequence
1
3
2
0
Pixel values converted to 2-bit binary pairs
01
11
10
00
Re-ordered binary pairs
00
10
11
01
1-byte packed hexadecimal value
2D

### Definition

The following XML code defines a `CiscoIPPhoneImage` object that displays the sequence of pixels shown in Figure 2-3 as a graphic positioned at the center of the phone display:

```
<CiscoIPPhoneImage>
  <Title/>
   <LocationX>-1</LocationX>
   <LocationY>-1</LocationY>
   <Width>4</Width>
   <Height>1</Height>
   <Depth>2</Depth>
   <Data>2D</Data>
   <Prompt/>
</CiscoIPPhoneImage>
```

The graphic display comprises a contiguous stream of hexadecimal digits, with no spaces or other separators. If the number of pixels to be displayed does not represent an even multiple of four, pad the end of the pixel data with blank (zero value) pixels, so the data is packed correctly. The phone ignores the padded data.

**Tip** Before displaying a graphic image on a Cisco IP Phone, the software clears the pane dedicated to services. If a service has text or other information that must be preserved (including the title area), the information must get redrawn as part of the graphic. If the title is to be hidden, the graphic must be large enough to cover it.

## CiscoIPPhoneImageFile

The latest generation of Cisco IP Phones have higher-resolution displays with more color depth. The Cisco IP Phone 7970, for example, has a display area of 298x168 pixels available to the Services pane and renders images in 12-bit color.

To support these more advanced displays, a new XML object allows the use of color PNG images in addition to the grayscale `CiscoIPPhoneImage` objects. The `CiscoIPPhoneImageFile` object behaves like the `CiscoIPPhoneImage` object, except for the image data. Instead of using the `<Data>` tag to embed the image data, the `<URL>` tag points to the PNG image file.

The web server must deliver the PNG image to the phone with an appropriate MIME Content-Type header, such as image/png, so the phone recognizes the content as a compressed, binary PNG image. The PNG image can be either palettized or RGB, and the maximum image size and color depth are model dependent (see Table 2-2).

*Table 2-2 Cisco IP Phones Display Image Sizes and Color Depths*

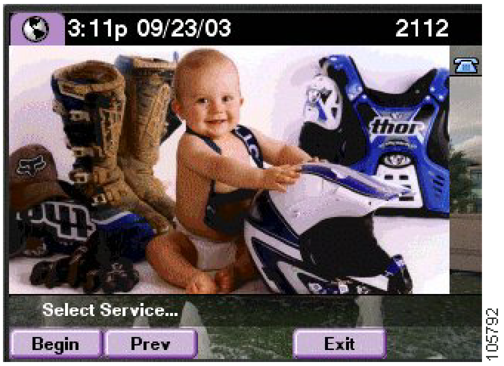
| Model | Resolution[1] (width x height) | Color/Grayscale | Color Depth (bits) |
|---|---|---|---|
| Cisco IP Phones 7905G/7912G[2] | N/A | Grayscale | 1 |
| Cisco IP Phone 7920 | 128 x 59 | Grayscale | 1 |
| Cisco IP Phones 7940G/60G | 133 x 65 | Grayscale | 2 |
| Cisco IP Phone 7970G/7971G | 298 x 168 | Color | 12 |
| Cisco IP Communicator | 298 x 168 | Color | 24 |

1. Represents the size of the display that is accessible by Services—not the full resolution of the physical display.
2. The Cisco IP Phones 7905 and 7912 have pixel-based displays, but they do not support XML images.

If the number of colors in the image is not reduced to match the phone capabilities, the image will be dithered by the phone and yield less than desirable results in most cases. To reduce the number of colors in a graphics editing program, such as Adobe Photoshop, use the “Posterize” command. The “Posterize” command takes one value as input for the number of color tones per color channel. For example, using the value of 16 (4-bits per channel = 16 tones per channel) will correctly dither the color palette of the image for the best display results on the Cisco IP Phone 7970.

Figure 2-4 shows a CiscoIPPhoneImageFile object on a Cisco IP Phone 7970 display.

***Figure 2-4 Cisco IP Phone 7970 Image File Display***



## Definition

```
<CiscoIPPhoneImageFile>
  <Title>Image Title goes here</Title>
  <Prompt>Prompt text goes here</Prompt>
  <LocationX>Horizontal position of graphic</LocationX>
  <LocationY>Vertical position of graphic</LocationY>
  <URL>Points to the PNG image</URL>
</CiscoIPPhoneImageFile>
```

## CiscoIPPhoneGraphicMenu

Graphic menus serve the same purpose as text menus: they allow a user to select a URL from a list. Use graphic menus in situations when the items may not be easy to display in a text list. For example, users might prefer to have their choices presented in a non-ASCII character set such as Kanji or Arabic.

In these cases, the system presents the information as a bitmap graphic. The user then enters a number from the DTMF keypad to make a menu selection.

### Definition

```
<CiscoIPPhoneGraphicMenu>
  <Title>Menu title goes here</Title>
  <Prompt>Prompt text goes here</Prompt>
  <LocationX>Position information of graphic</LocationX>
  <LocationY>Position information of graphic</LocationY>
  <Width>Size information for the graphic</Width>
  <Height>Size information for the graphic</Height>
  <Depth>Number of bits per pixel</Depth>
  <Data>Packed Pixel Data</Data>
  <MenuItem>
    <Name>The name of each menu item</Name>
    <URL>The URL associated with the menu item</URL>
  </MenuItem>
</CiscoIPPhoneGraphicMenu>
```

Menu items in the graphic menu have a name, like the text menu counterparts. Although the name does not display to the user, it still performs a function. The name of the menu item provides the default title that is used when the URL for the chosen item is loaded. If the loaded page has a title of its own, the phone uses that title instead.

The XML tags in `GraphicMenu` use the tag definitions for `CiscoIPPhoneImage` and `CiscoIPPhoneMenu`. Although the semantics of the tags are identical, you can have only 12 `MenuItem` objects in a `CiscoIPPhoneGraphicMenu` object. See "CiscoIPPhoneMenu" and "CiscoIPPhoneImage" for detailed descriptions.

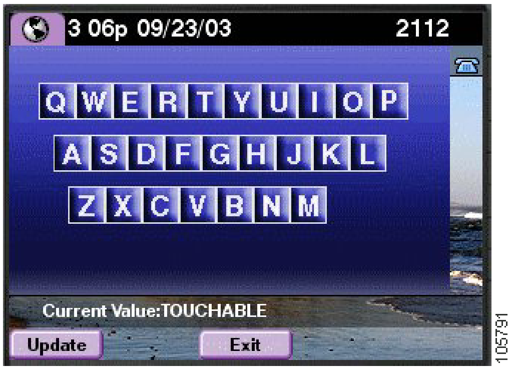
## CiscoIPPhoneGraphicFileMenu

Some of the latest Cisco IP Phone models, such as the Cisco IP Phone 7970 and Cisco IP Communicator, have pointer devices. The Cisco IP Phone 7970 uses a touchscreen overlay on the display, and the PC-based Cisco IP Communicator uses the standard Windows mouse pointer.

Because these devices can receive and process "pointer" events, a `CiscoIPPhoneGraphicFileMenu` object exposes the capability to application developers. The CiscoIPPhoneGraphicFileMenu behaves similar to the CiscoIPPhoneGraphicMenu, in that a group of options are presented by an image. When one of those objects is selected, a URL action initiates. However, the new FileMenu does not use the keypad, but uses rectangular touch areas. This rectangular touch area, `<TouchArea>`, is defined by coordinates relative to the upper-left corner of the Services display. The (X1,Y1) points specify the upper-left corner of the `<TouchArea>`, and (X2,Y2) specify the lower-right corner of the `<TouchArea>`.

Figure 2-5 shows the display of the CiscoIPPhoneGraphicFileMenu.

***Figure 2-5 CiscoIPPhoneGraphicFileMenu***



If the coordinates that are supplied in `<TouchArea>` tag exceed the dimensions of the phone display, the `<TouchArea>` rectangle will be "clipped" to fit. See Table 2-2, "Cisco IP Phones Display Image Sizes and Color Depths" for a listing of usable display resolutions for each phone model.

The `<TouchArea>` rectangles are allowed to overlap, and the first match is always taken. This allows a sense of Z-order for images where smaller touchable objects can be overlaid on top of larger ones. In this case, the smaller object `<MenuItem>` must appear before the larger one in the `<CiscoIPPhoneGraphicFileMenu>` object.

The requirements for the PNG image referenced by the `<URL>` tag match those that the CiscoIPPhoneImageFile object uses.

### Definition

```
<CiscoIPPhoneGraphicFileMenu>
  <Title>Image Title goes here</Title>
  <Prompt>Prompt text goes here</Prompt>
  <LocationX>Horizontal position of graphic</LocationX>
  <LocationY>Vertical position of graphic</LocationY>
  <URL>Points to the PNG background image</URL>
  <MenuItem>
    <Name>Same as CiscoIPPhoneGraphicMenu</Name>
    <URL>Invoked when the TouchArea is touched</URL>
    <TouchArea X1="left edge" Y1="top edge" X2="right edge" Y2="bottom
edge"/>
  </MenuItem>
</CiscoIPPhoneGraphicFileMenu>
```

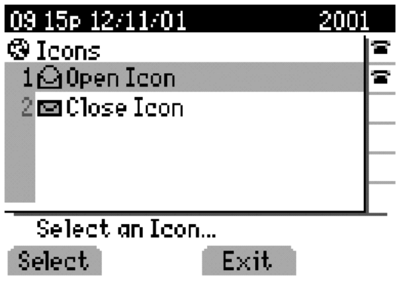
## CiscoIPPhoneIconMenu

Icon menus serve the same purpose as text menus: they allow a user to select a URL from a list. Use icon menus in situations when you want to provide additional visual information to the user to show the state or category of an item. For example, you include a a read and unread icon in a mail viewer. You can use the icons can to convey the message state.

Icons in the `CiscoIPPhoneMenu` object have a maximum width of 16 pixels and a maximum height of 10 pixels.

Figure 2-6 shows an IconMenu on a Cisco IP Phone.

*Figure 2-6 IconMenu on a Cisco IP Phone Sample*



The system presents the information as a bitmap graphic to the left of the menu item text. The user selects menu items in the same way as a `CiscoIPPhoneMenu` object.

### Definition

```
<CiscoIPPhoneIconMenu>
  <Title>Title text goes here</Title>
  <Prompt>Prompt text goes here</Prompt>
  <MenuItem>
    <IconIndex>Indicates what IconItem to display</IconIndex>
    <Name>The name of each menu item</Name>
    <URL>The URL associated with the menu item</URL>
  </MenuItem>
  <SoftKeyItem>
    <Name>Name of soft key</Name>
    <URL>URL or URI of soft key</URL>
    <Position>Position information of the soft key</Position>
  </SoftKeyItem>
  <IconItem>
    <Index>A unique index from 0 to 9</Index>
    <Height>Size information for the icon</Height>
    <Width>Size information for the icon</Width>
    <Depth>Number of bits per pixel</Depth>
    <Data>Packed Pixel Data</Data>
  </IconItem>
</CiscoIPPhoneIconMenu>
```

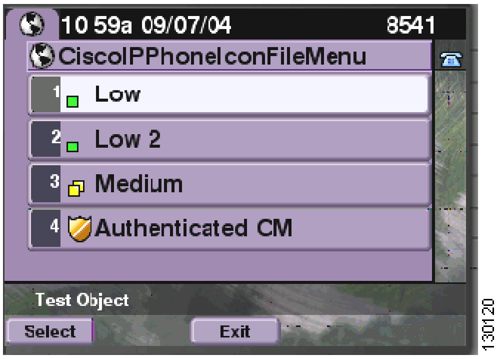
The XML tags in IconMenu use the tag definitions for CiscoIPPhoneImage and CiscoIPPhoneMenu. Although the semantics of the tags are identical, you can have only 32 MenuItem objects in a `CiscoIPPhoneIconMenu` object. See "CiscoIPPhoneMenu" and "CiscoIPPhoneImage" for detailed descriptions.

## CiscoIPPhoneIconFileMenu

This icon menu is similar to `CiscoIPPhoneMenu`, but it uses color PNG icons rather than grayscale CIP icons. Use icon menus in situations when you want to provide additional visual information to the user to show the state or category of an item. For example, you can use icons to indicate priority (see Figure 2-7).

Icons in the `CiscoIPPhoneIconFileMenu` object have a maximum width of 18 pixels and a maximum height of 18 pixels. Instead of using the `<Data>` tag to embed the image data into the `<IconItem>` tag, this object uses a `<URL>` tag to point to the PNG image file to be used for that icon.

***Figure 2-7 CiscoIPPhoneIconFileMenu Object Display Sample***


10 59a 09/07/04 8541
CiscoIPPhoneIconFileMenu
1 Low
2 Low 2
3 Medium
4 Authenticated CM
Test Object

### Definition

```
<CiscoIPPhoneIconFileMenu>
  <Title>Title text goes here</Title>
  <Prompt>Prompt text goes here</Prompt>
  <MenuItem>
    <IconIndex>Indicates what IconItem to display</IconIndex>
    <Name>The name of each menu item</Name>
    <URL>The URL associated with the menu item</URL>
  </MenuItem>
  <IconItem>
    <Index>A unique index from 0 to 9</Index>
    <URL>location of the PNG icon image</URL>
  </IconItem>
</CiscoIPPhoneIconFileMenu>
```

## CiscoIPPhoneStatus

The CiscoIPPhoneStatus object is also a displayable object, but differs from the preceding objects in that it displays on the Call plane of the phone rather than the Services plane. The CiscoIPPhoneStatus object “hovers” above the Call plane and is typically used in conjunction with CTI applications to present application status to the user.

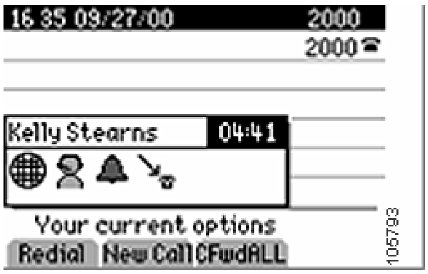
Because the Status object is only present on the Call plane, the object cannot be closed or cleared by the user (for example, by pressing Services). In order to clear the object, the phone must execute the Init:AppStatus URI. This would typically occur as the result of an application server PUSHing an Execute object to the phone that contains the Init:AppStatus URI.

The CiscoIPPhoneStatus object can be refreshed or replaced at any time. It is not necessary to clear an existing Status object before sending a new Status object. The new object simply replaces the old object.

Figure 2-8 shows the CiscoIPPhoneStatus object that contains the following visual elements:

- 106 x 21 graphics area for displaying CIP images (same image format as CiscoIPPhoneImage)
- Seedable, free-running timer (optional)
- Single-line text area (optional)

***Figure 2-8 IconMenu on a CiscoIPPhoneStatus Sample***



### Definition

```
<CiscoIPPhoneStatus>
  <Text>This is the text area</Text>
  <Timer>Timer seed value in seconds</Timer>
  <LocationX>Horizontal alignment</LocationX>
  <LocationY>Vertical alignment</LocationY>
  <Width>Pixel width of graphic</Width>
  <Height>Pixel height of graphic</Height>
  <Depth>Color depth in bits</Depth>
  <Data>Hex binary image data</Data>
</CiscoIPPhoneStatus>
```

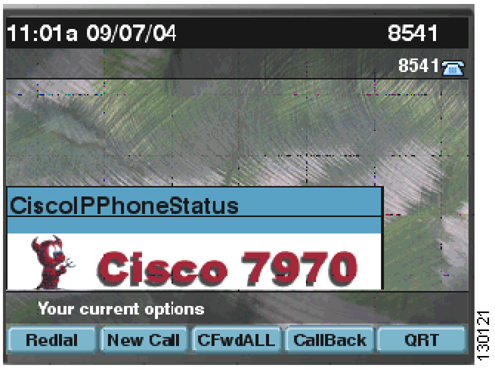
## CiscoIPPhoneStatusFile

The behavior of this object is identical to the `CiscoIPPhoneStatus` object, except it uses a color PNG image instead of a grayscale CIP image for the graphics area.

The maximum image size is 262 x 50 pixels for the Cisco 7970, but may be different for other phone models.

Figure 2-9 shows how an XML `CiscoIPPhoneStatusFile` object displays on the phone.

***Figure 2-9 CiscoIPPhoneStatusFile Object Display Sample***



### Definition

```
<CiscoIPPhoneStatusFile>
<Text>This is the text area</Text>
<Timer>Timer seed value in seconds</Timer>
<LocationX>Horizontal alignment</LocationX>
<LocationY>Vertical alignment</LocationY>
<URL>location of the PNG image</URL>
</CiscoIPPhoneStatusFile>
```

Note that instead of using the `<Data>` tag to embed the image data, this object uses a *`<URL>`* tag to point to the PNG image file to be used for the graphics area.

## CiscoIPPhoneExecute

The `CiscoIPPhoneExecute` object differs from the other CiscoIPPhone objects. It is not a displayable object for providing user interaction. The purpose of this object is to deliver (potentially multiple) execution requests to the phone.

Like the other XML objects, the CiscoIPPhoneExecute can be either pushed (HTTP POST) or pulled (HTTP GET). Upon receiving a CiscoIPPhoneExecute object, the phone will begin executing the specified ExecuteItems. Order of execution is not guaranteed, so ExecuteItems will likely not execute in the order in which they are listed in the CiscoIPPhoneExecute object.

**Note** Limit the requests to three ExecuteItems: only one can be a URL and two URIs per `CiscoIPPhoneExecute` object, or you can send three URIs with no URL.

### Definition

```
<CiscoIPPhoneExecute>
    <ExecuteItem URL="the URL or URI to be executed"/>
</CiscoIPPhoneExecute>
```

The `<ExecuteItem>` tag of the `CiscoIPPhoneExecute` object includes an optional attribute called Priority. The Priority attribute is used to inform the phone of the urgency of the execute request and to indicate whether the phone should be interrupted to perform the request. The Priority levels determine whether the phone must be idle to perform the requested action. The Idle Timer (along with an optional Idle URL) is defined globally in the Cisco CallManager Administration Enterprise Parameters and can be overridden on a per phone basis in the Cisco CallManager Device configuration.

The following table lists the Priority levels and their behavior.

| Behavior | Description |
|---|---|
| 0 = Execute Immediately | The URL executes regardless of the state of the phone. If the Priority attribute does not get specified in the `<ExecuteItem>`, the default priority gets set to zero for backward compatibility. |
| 1 = Execute When Idle | The URL gets delayed until the phone goes idle, at which time it executes. |
| 2 = Execute If Idle | The URL executes on an idle phone; otherwise, it does not get executed (it does not get delayed). |

**Note** The Priority attribute is only used for HTTP URLs. Internal URIs always execute immediately.

### Example

The following `CiscoIPPhoneExecute` object results in the phone playing an alert "chime," regardless of the state of the phone, but waits until the phone goes idle before displaying the specified XML page:

```
<CiscoIPPhoneExecute>
    <ExecuteItem Priority="0" URL="Play:chime.raw"/>
    <ExecuteItem Priority="1" URL="http://server/textmessage.xml"/>
</CiscoIPPhoneExecute>
```

## CiscoIPPhoneResponse

The `CiscoIPPhoneResponse` object items provide messages and information resulting from `CiscoIPPhoneExecute`. As a result, a `ResponseItem` exists for each `ExecuteItems` that you send. The order differs based on completion time, and the execution order is not guaranteed.

The URL attribute specifies the URL or URI that was sent with the request. The Data attribute contains any special data for the item. The Status attribute specifies a status code. Zero indicates that no error occurred during processing of the ExecuteItem. If an error occurred, the phone returns a `CiscoIPPhoneError` object.

### Definition

```
<CiscoIPPhoneResponse>
   <ResponseItem Status=”the success or failure of the action”
   Data=”the information returned with the response” URL=”the URL or
   URI specified in the Execute object”/>
</CiscoIPPhoneResponse>
```

## CiscoIPPhoneError

The following list gives possible CiscoIPPhoneError codes:

- Error 1 = Error parsing `CiscoIPPhoneExecute` object
- Error 2 = Error framing `CiscoIPPhoneResponse` object
- Error 3 = Internal file error
- Error 4 = Authentication error

### Definition

```
<CiscoIPPhoneError Number=”x”/> optional error
 message<.CiscoIPPhoneError>
```

The text value of the `CiscoIPPhoneError` object may contain an optional error message to further describe the nature of the error condition.

## Custom Softkeys

Cisco IP Phones can use custom softkeys with any of the displayable CiscoIPPhone XML objects, excluding the `CiscoIPPhoneStatus` object which cannot control softkeys and the `CiscoIPPhoneExecute` object which is not displayable.

Softkeys can have either URL or URI “actions” associated with them. The `SoftkeyItem` can define separate actions to be taken when the softkey is pressed and released. The standard UI behavior is to execute an action when a key is released, and this action is defined by the `<URL>` tag. An action can also be taken when the softkey is initially pressed by including the optional `<URLDown>` tag. For example, you might use `<URLDown>` for a press-to-talk application in which pressing the button starts audio streaming and releasing the button stops it.

### Definition

```
<SoftKeyItem>
   <Name>Displayed sofkey label</Name>
   <URL>URL or URI action for softkey RELEASE event</URL>
   <URLDown>URL or URI action for softkey PRESS event</URLDown>
   <Position>position of softkey</Position>
</SoftKeyItem>
```

### Example

In this example, a `CiscoIPPhoneText` object has a single custom softkey defined:

```
<CiscoIPPhoneText>
  <Text>This object has one softkey named "Custom"</Text>
  <SoftKeyItem>
    <Name>Custom</Name>
    <URL>http://someserver/somepage</URL>
    <Position>4</Position>
  </SoftKeyItem>
</CiscoIPPhoneText>
```

If any custom softkeys are defined in the XML object, then all default softkeys are removed from that object. Thus, if f the default softkey behavior is required (in addition to the custom softkeys), then it must be explicitly defined in the XML object using a `<SoftKeyItem>` tag. The internal Softkey URIs can be used in the `<URL>` tag of `<SoftKeyItem>` to invoke default softkey actions from custom softkeys. See the "URIs for Invoking SoftKey Functionality" section on page 3-3 for more information on invoking internal softkey features.

### Example

The following softkey definitions would provide the custom softkey, without losing the default "Select" behavior:

```
<SoftKeyItem>
   <Name>Select</Name>
  <URL>SoftKey:Select</URL>
  <Position>1</Position>
</SoftKeyItem>
<SoftKeyItem>
  <Name>Custom</Name>
  <URL>http://someserver/somepage</URL>
  <Position>4</Position>
</SoftKeyItem>
```

## XML Considerations

The XML parser in Cisco IP Phones does not function as a fully capable XML parser. Do not include any tags other than those defined in your XML display definitions.

**Note** All CiscoIPPhone element names and attribute names are case sensitive.

## Mandatory Escape Sequences

By XML convention, the XML parser also requires that you provide escape values for a few special characters. Table 2-3 lists characters and their escape values.

*Table 2-3 Escape Sequences for Special Characters*

| Character | Name | Escape Sequence |
|---|---|---|
| & | Ampersand | &amp; |
| " | Quote | &quot; |
| ' | Apostrophe | &apos; |
| < | Left angle bracket | &lt; |
| > | Right angle bracket | &gt; |

Escaping text can be tedious, but some authoring tools or scripting languages can automate this task.

# Internal URI Features

Internal uniform resource identifiers(URIs) provide access to embedded phone features such as placing calls, playing audio files, and invoking built-in object features. These sections provide details about the available internal URIs:

- Supported URIs by Phone Model
- URIs for Pressing Buttons on the Phone
- URIs for Invoking SoftKey Functionality
- URIs to Control RTP Streaming
- Miscellaneous URIs

## Supported URIs by Phone Model

Table 3-1 lists the URIs that are supported for Release 4.1(3).

*Table 3-1 URIs Supported for Release 4.1(3) Cisco IP Phone Services SDK*

| Phone Model URI Group | 7905G / 7912G | 7920 | 7940G / 7960G | 7970 G/ 7971G IP Communicator |
|---|---|---|---|---|
| Key | X | X | X | X |
| Softkey | X | X | X | X |
| Init | X | X | X | X |
| Dial, EditDial | X | X | X | X |
| Play | X | X | X | X |

*Table 3-1 URIs Supported for Release 4.1(3) Cisco IP Phone Services SDK (continued)*

| Phone Model URI Group | 7905G / 7912G | 7920 | 7940G / 7960G | 7970 G/ 7971G IP Communicator |
|---|---|---|---|---|
| QueryStringParam | X | X | X | X |
| Unicast RTP (RTPRx, RTPTx) | X | X[1,2] | X | X |
| Multicast RTP (RTPMRx, RTPMTx) | X |  | X | X |

1. Does not support the Volume parameter for RTP Receive streams.
2. Cisco IP Phone 7920 only supports one incoming and one outgoing unicast stream.

## URIs for Pressing Buttons on the Phone

The Key URI allows a programmer to send an event that a key has been pressed. The system initiates the event as if the button was physically pressed.

Note that when buttons are pressed with this method, if the button is not be available when the URI is processed, the event is discarded.

***Example 3-1 Key:Soft1***

If the softkey set is changing and disabled while the event is being processed, the request is discarded.

Verify available softkeys by using the QA web pages that the phones web server provides to indicate the active softkey set.

## Key

URI Format:

```
Key:n
```

*Where*

**n** = a Key name.

The following is a complete listing of the Key URIs:

- Key:Line1 to Key:Line34
- Key:KeyPad0 to Key:KeyPad9
- Key:Soft1 to Key:Soft4
- Key:KeyPadStar
- Key:KeyPadPound
- Key:VolDwn
- Key:VolUp
- Key:Headset
- Key:Speaker
- Key:Mute
- Key:Info
- Key:Messages
- Key:Services
- Key:Directories
- Key:Settings
- Key:NavUp
- Key:NavDwn

## URIs for Invoking SoftKey Functionality

You can execute native softkey functionality when the phone executes a SoftKey URI. The SoftKey URI allows developers to customize softkey names and layout in the Services and Directories windows while retaining the functionality that the softkeys provide.

Softkey URIs work in menu items and in softkey items in the XML objects for which they natively occur on the phone.

## SoftKey

URI Format:

**`SoftKey:n`**

*Where*

**n** = one of the following softkey names:

- Back
- Cancel
- Exit
- Next
- Search
- Select
- Submit
- Update
- Dial
- EditDial
- <<

Table 3-2 contains valid softkey actions for each XSI object type follow. The URI invokes the native functionality that each key possesses in the given object context.

***Table 3-2 Valid Softkey Actions for IPPhoneObject Types***

| IPPhoneObject | Select | Exit | Update | Submit | Search | << | Cancel | Next | Dial | Edit Dial |
|---|---|---|---|---|---|---|---|---|---|---|
| CiscoIPPhoneMenu | X | X | | | | | | | | |
| CiscoIPPhoneIconMenu | X | X | | | | | | | | |
| CiscoIPPhoneText | | X | X | | | | | | | |
| CiscoIPPhoneImage | | X | X | | | | | | | |
| CiscoIPPhoneGraphicMenu | | X | X | | | | | | | |

*Table 3-2 Valid Softkey Actions for IPPhoneObject Types (continued)*

| IPPhoneObject | Select | Exit | Update | Submit | Search | << | Cancel | Next | Dial | Edit Dial |
|---|---|---|---|---|---|---|---|---|---|---|
| CiscoIPPhoneInput | | | | X | X[1] | X | X | | | |
| CiscoIPPhoneDirectory | | | | | | | X | X | X[2] | X[2] |

1. Only when used under the Directories button.
2. The SoftKey:Dial and SoftKey:EditDial URIs can be used only for Directory objects, but the Dial:xxx and EditDial:xxx URIs can be used as the URL of any SoftKeyItem or MenuItem. For more details, see the "Miscellaneous URIs" section on page 3-10.

## QueryStringParam URI

The QueryStringParam URI allows an application developer to collect more information from the user with less interaction. When the user performs an action with a softkey, you can either append a query string parameter to the URL of the highlighted MenuItem or append the query string parameter from the MenuItem to the URL of the softkey.

URI Format:

**`QueryStringParam:d`**

*Where*

**d** = the data to be appended to a corresponding URL.

### *Example 3-2 QueryStringParam URI in a CiscoIPPhoneMenu object*

```
<CiscoIPPhoneMenu>
  <Title>Message List</Title>
  <Prompt>Two Messages</Prompt>
  <MenuItem>
     <Name>Message One</Name>
     <URL>QueryStringParam:message=1</URL>
  </MenuItem>
  <MenuItem>
     <Name>Message Two</Name>
     <URL>queryStringParam:message=2</URL>
  </MenuItem>
  <SoftKeyItem>
     <Name>Read</Name>
     <URL>http://server/read.asp</URL>
  </SoftkeyItem>
  <SoftKeyItem>
    <Name>Delete</Name>
    <URL>http://server/delete.asp</URL>
  </SoftkeyItem>
</CiscoIPPhoneMenu>
```

Example 3-2 shows how to use the QueryStringParam URI in a `CiscoIPPhoneMenu` object. The `CiscoIPPhoneMenu` object includes two MenuItems with QueryStringParam URIs. If the user chooses the MenuItem(s) with the numeric keypad, the cursor moves to that entry, but nothing executes because the values are QueryStringParam URIs.

If the user presses either custom softkey, the currently highlighted MenuItem URI value gets appended to the softkey URL that was pressed and requested from the web server.

If you highlight the first MenuItem and press the Read softkey, the phone generates the following URL:

http://server//read.asp?message=1

### *Example 3-3 Selecting an Item with Numeric Keypad Calls the URL*

```
<CiscoIPPhoneMenu>
  <Title>Message List</Title>
  <Prompt>Two Messages</Prompt>
  <MenuItem>
    <Name>Messae One</Name>
    <URL>http://server/messages.asp?message=1</URL>
  </MenuItem>
  <MenuItem>
    <Name>Messae Two</Name>
    <URL>http://server/messages.asp?message=2</URL>
  </MenuItem>
  <SoftKeyItem>
    <Name>Read</Name>
    <Position>1</Position><URL>QueryStringParam:action=read</URL>
  </SoftKeyItem>
  <SoftKeyItem>
    <Name>Delete</Name>
    <Position>2</Position><URL>QueryStringParam:action=delete</URL>
  </SoftKeyItem>
</CiscoIPPhoneMenu>
```

The Cisco IP Phones allow you to implement the QueryStringParam URI in either manner although Example 3-3 is not as efficient as Example 3-2. Choose the best way to perform the action based on your applications needs.

Example 3-3 does have a slight advantage in that if the user chooses an item with the numeric keypad, the URL gets called. This would allow you to invoke some default behavior such as to read the message in the example. By highlighting the first message and pressing the Read softkey, the phone creates the following URL:

http://server/messages.asp?message=1&action=read

Using the QueryStringParam URI reduces the size of the XML objects that you generate by not having to repeat redundant portions of a URL in every MenuItem.

## URIs to Control RTP Streaming

You can invoke RTP streaming via URIs in services. You can instruct the phone to transmit or receive an RTP stream with the following specifications:

- RTPRx
- RTPTx
- RTPMRx
- RTPMTx

The supported format of the RTP stream follows:

- The codec is G.711 mu-Law.
- The packet size is 20 ms.

### RTPRx

The RTPRx URI instructs the phone to receive a Unicast RTP stream or to stop receiving Unicast or Multicast RTP streams.

URI Formats:

```
RTPRx:i:p:v
RTPRx:Stop
```

*Where*

**i** = the IP Address from which the stream is coming.

**p** = the UDP port on which to receive the RTP stream. Ensure that this is an even port number within the decimal range of 20480 to 32768. If no port is specified, the phone chooses a port and returns it when initiated by a push request.

**Stop** is the parameter that will stop any active RTP stream from being received on channel one.

**v** = the optional volume setting that controls the volume of stream playout. The supplied value is a percentage of the maximum volume level of the device and must be in the range 0-100. The phone converts the specified percentage into the closest device-supported volume level setting and uses it. After the initial volume level gets set and the stream starts, you can manually change the volume level as needed. If the optional volume parameter does not get included, the current volume setting on the phone gets used as the default.

## RTPTx

Use the RTPTx URI to instruct the phone to transmit a Unicast RTP stream or to stop transmitting Unicast or Multicast RTP streams.

URI Formats:

```
RTPTx:i:p
RTPTx:Stop
```

*Where*

**i** = the IP Address to which an RTP stream is transmit ed.

**p** = the UDP port on which to transmit the RTP stream. Ensure that this is an even port number within the decimal range of 20480 to 32768.

**Stop** = the parameter that will stop any active RTP stream from being transmitted on channel one.

## RTPMRx

The RTPMRx URI instructs the phone to receive a Multicast RTP

URI Format:

```
RTPMRx:i:p:v
```

*Where*

**i** = the Multicast IP Address from which to receive an RTP stream.

**p** = the Multicast UDP port from which to receive the RTP stream. Ensure that this is an even port number within the decimal range of 20480 to 32768.

**v** is the optional volume setting that controls the volume of stream playout. The supplied value is a percentage of the maximum volume level of the device and must be in the range 0-100. The phone converts the specified percentage into the closest device-supported volume level setting and uses it. After the initial volume level gets set and the stream starts, you can manually change the volume level as needed. If the optional volume parameter does not get included, the current volume setting on the phone gets used as the default.

## RTPMTx

The RTPMTx URI instructs the phone to transmit a Multicast RTP stream.

URI Formats:

```
RTPTx:i:p
```

*Where*

**i** = the Multicast IP Address to which an RTP stream is transmit ed.

**p** = the Multicast UDP port on which to transmit the RTP stream. Ensure that this is an even port number within the decimal range of 20480 to 32768.

## Miscellaneous URIs

This section describes the following miscellaneous URIs:

- Init
- Dial
- EditDial
- Play

### Init

The Init URI allows an application to initialize a feature or data with the argument that is passed with the URI.

URI Format:

```
Init:o
```

*Where*

**o** = the Object name.

Valid object name:

**`CallHistory`**—When the phone encounters an Init:CallHistory URI, it clears the internal call history logs that are stored in the phone. This action initializes Missed Calls, Received Calls, and Placed Calls.

### Dial

The Dial URI initiates a new call to a specified number. The Dial URI invokes when it is contained in a menu item, the menu item is highlighted, and the device is taken off hook.

Activate the Dial URI by one of the following:

- Line button
- Speaker button
- Headset button
- Handset hook switch

- Normal menu item
- Softkey item selection

URI Format:

```
Dial:n
```

*Where*

**n** = the number dialed (such as **Dial:1000**).

## EditDial

The EditDial URI initiates a new call to a specified number. The EditDial URI invokes when it is contained in a menu item, the menu item is highlighted, and the device is taken off hook.

Activate the EditDial URI by one of the following:

- Line button
- Speaker button
- Headset button
- Handset hook switch
- Normal menu item
- Softkey item selection

URI Format:

```
EditDial:n
```

*Where*

**n** = the number dialed (such as **EditDial:1000**).

## Play

The Play URI downloads an audio file from the TFTP server and plays through the phone speaker. This same mechanism also plays ring files, and the format of the files is the same. You could use the Play URI to play files that are in the Ringlist.xml or those that are not.

If you want to achieve a unique audible notification, use a file that is not available from the Ringlist.xml.

URI Format:

```
Play:f
```

*Where*

**f** = the filename of a raw audio file in the TFTP path (such as **Play:Classic2.raw**).

# Cisco IP Services Software Development Kit (SDK)

The Cisco IP Services Software Development Kit (SDK) contains everything to create XML applications, including necessary documentation and sample applications. Contact Cisco Developer Services to obtain the SDK at: http://www.cisco.com/go/developersupport.

These sections describe the Cisco IP Services SDK:

- SDK Components
- Sample Services Requirements

## SDK Components

The following list contains the components that are included in the SDK:

- Documentation
  - Cisco IP Services Development Notes (in Adobe Acrobat format)
  - Cisco URL Proxy Guide (Rich Text Format)
  - Cisco LDAP Programming Guide (Microsoft Word format)
  - Cisco CIP Image Release Notes (MicrosoftWord format)
  - Cisco IP Applications Samples (Microsoft Word format)

- Development Tools
  - Cip.8bi—Adobe Photoshop plugin that allows .cip extensions to be viewed and saved.
  - Cip2Gif.exe—DOS-based program that converts .cip files to .gif.
  - Gif2Cip.exe—DOS-based program that converts .gif files to .cip.
  - ImageViewer.exe—Windows application that displays .cip graphic files.
  - Cisco CIPImage—used for converting images to and from CIP images (automatically installed)
  - Cisco URL Proxy—Proxy server that is needed to use the sample services (automatically installed).
  - Cisco LDAP Search—Service that is installed to do LDAP searches (automatically installed).
  - Microsoft XML Parser (MXSML) 3.0—Used for parsing XML data (automatically installed)
  - Cisco IP Phone Services ASP/Javascript Library (automatically installed)
  - Cisco IP Phone Services Java Library—Used by the JSP apps (manually installed - see JSP Install readme)
  - CallManager Simulator —Used for developing Phone Services without a CallManager server
  - Cisco IP Phone XML Schema (.xsd) file—Used with an XML editor to validate XML syntax
- Sample Services
  - Weather forecast lookup for any city (ASP)
  - Currency Exchange Rates and Converter (ASP
  - UPS Rates & tracking (ASP)
  - World Clock (ASP)
  - Measurement conversions (ASP)
  - US White pages/Yellow Pages search (ASP)
  - Calendar (ASP)
  - Stock Ticker (ASP)

- Stock Chart (ASP)
- Push2Phone (ASP and JSP)
- Click2Dial (ASP and JSP)
- IdleURL (ASP)
- MConference (JSP)
- Hootie (ASP)
- InterCom (ASP)
- JPEGViewer (ASP)
- Logo (ASP)
- Clock (ASP)
- Personal Service (ASP)
- WaterMark (ASP)
- Extension Mobility Controller (JSP)
- Speed Dials (JSP)
- Group MWI (JSP)
- AutoDialer (JSP)
- PhotoDirectory (JSP)
- CallerInfo (JSP)
- PushAuthenticate (ASP)
- ScreenShot (ASP)
- Integrating RS-232 devices with IP Telephony Applications (OtherApps)
- PNGViewer (ASP)
- Keyboard (ASP)
- MultiDirectory (ASP)
- Phone Push Step and Subsystem (IPCC Express / CRS)

## Sample Services Requirements

The following list contains the items that are required for the sample services to work properly:

- Microsoft IIS 4.0 or later (for ASP sample services
- Sun J2SE 1.4.2 or later and Tomcat 4.0 or later (for JSP sample services)
- Internet Connection to external websites like Yahoo.com, Cnn.com etc.
- Cisco Call Manager 4.1(2) or later.
- Cisco 7905, 7912, 7920, 7940, 7960, 7970 or other XML services-enabled IP Phone

The setup program installs a CiscoServices web project to c:\CiscoIpServices directory. The sample services get copied to the c:\CiscoIpServices\Services subdirectory, and IIS and WSH example codes are provided. The web server already senses these services; you need no further administration. You can view or edit all the source code with any text editor. For additional documentation, go to this directory: c:\CiscoIpServices\Documentation. Find tools to help develop services in c:\CiscoIpServices\Tools.

# HTTP Requests and Header Settings

Cisco IP Phones use HTTP to communicate to external applications. The phone firmware includes both an HTTP client for making requests, and an HTTP server for receiving requests. This chapter describes the capabilities of the HTTP interface.

This chapter contains the following sections:

## HTTP Client Requests (HTTP GET)

The following description designates how HTTP client requests are handled:

1. The Cisco IP Phone HTTP client performs an HTTP GET for a specified URL.
2. The HTPP server processes request and returns an XML object or plain text.
3. The phone processes the supported HTTP headers.

4. The phone parses the XML object if ContentType is `text/xml`.
5. The phone presents data and options to the user, or in the case of a CiscoIPPhoneExecute object, begins executing the URIs.

## HTTP Server Requests (HTTP POST)

The following description designates how an HTTP server request is made to the phone via an HTTP POST operation:

1. The server performs an HTTP POST in response to a case-sensitive URL of the phone with this format: http://*x.x.x.x*/CGI/Execute, where *x.x.x.x* represents the IP address of the destination Cisco IP Phone.

   The form that is posted should have a case-sensitive form field name called "XML" that contains the desired XML object. For any HTTP POST operation, the server must provide basic HTTP authentication information with the POST. The provided credentials must be of a user in the global directory with a device association with the target phone.

   If the credentials are invalid, or the Authentication URL is not set properly in the Cisco CallManager Administration, the phone will return a CiscoIPPhoneError with a value of 4 (Authentication Error) and processing will stop.
2. The phone processes the supported HTTP headers
3. The phone parses and validates the XML object
4. The phone presents data and options to the user, or in the case of a CiscoIPPhoneExecute object, begins executing the URIs.

**Tip** Any HTTP POST object is limited to 512 bytes in size. Larger objects (such as images) can only be delivered to the phone via HTTP GET. So, to push large objects to the phone, the server application must take an indirect approach. To do this, push an Execute object to the phone that contains an ExecuteItem pointing to the URL of the large object.

**Note** JTAPI also can push an XML object directly to an IP phone, with the added benefit of not requiring authentication (since the JTAPI connection itself is already authenticated). This option works particularly well for adding XML services interfaces to existing CTI applications (where the overhead of the CTI connection is already a requirement). Objects pushed via JTAPI are also limited to a maximum size of 512 bytes. See the *Cisco JTAPI Developer Guide* for more information.

## HTTP Header Settings

The following list provides definitions for HTTP header elements for Cisco IP Phone services:

- "Refresh"—sets the refresh time (in seconds) and URL
  - If no time is set or it is zero, the refresh gets set to manual.
  - If no URL is set, the current URL gets used.

  See the "HTTP Refresh Setting" section on page 5-3 section.
- ContentType —notifies the phone of the MIME type that was sent. See the "MIME Type and Other HTTP Headers" section on page 5-5 section.
- "Expires"—sets the Date/Time in GMT when the page is to expire.

  Pages that have expired before being loaded do not get added to the URL stack in the phone. The phone does not cache content. See "Content Expiration Header Setting" section on page 5-6 for more information.

## HTTP Refresh Setting

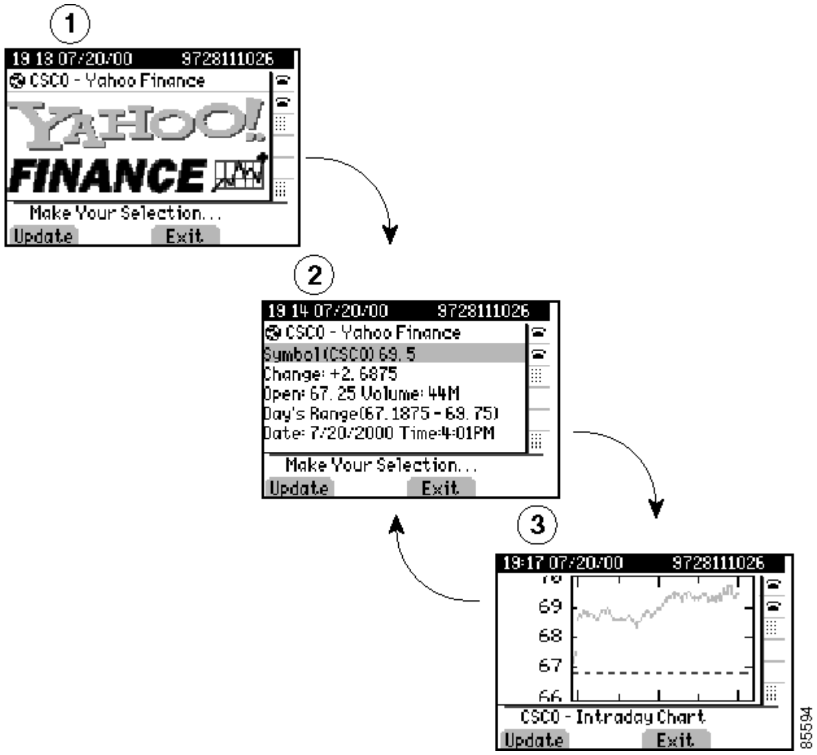
The HTTP headers that are sent with any page from an HTTP server can include a Refresh setting. This setting comprises two parameters: a time in seconds and a URL. These two parameters direct the recipient to wait the time given in the seconds parameter and then get the data to which the URL points.

The Cisco IP Phone HTTP client properly supports this setting, which gives a great deal of power to service developers. It means that a new page can replace any XML object that displays after a fixed time.

Figure 5-1 shows an example of how to use the refresh setting. This sample page shows the user the current value of Cisco stock.

1. A splash screen that displays the Yahoo logo.
2. After a very short time, it displays the numeric Cisco stock parameters.
3. Finally, it shows a graph of Cisco intraday stock performance. The display then repeatedly cycles between the final two views.

***Figure 5-1 Refresh Display Sample***



Symbol(CSCO) 69.5
Change: +2.6875
Open: 67.25 Volume: 44M
Day's Range(67.1875 - 69.75)
Date: 7/20/2000 Time:4:01PM

Refreshing the display can occur without user intervention, because the display automatically cycles if a timer parameter is specified. On any given screen, however, the user can force an immediate reload by pressing the Update softkey. Also, if a timer parameter of 0 was sent in the header, the page never automatically reloads. In this case, the display will move to the next page only when the Update softkey is pressed. If no refresh URL is specified, the current page gets reloaded.

## MIME Type and Other HTTP Headers

Although delivering pages with the proper MIME type and other formatting items is not difficult, it requires moderately in depth knowledge of your web server. The following code excerpt, written in Java Script and used with Microsoft IIS and ASP, sets these values in a few lines:

```
<%@ Language=JavaScript %>
<%
Response.AddHeader( "Refresh",
                    "3; url=http://services.cisco.com/s/q.asp");
Response.ContentType = "text/xml";
//
// Additional page content here
//
%>
```

Usually, you can set the MIME type for pages in any web server by simply performing an association to the .xml file extension. Your web server documentation should explain how to accomplish this. This action allows you to serve static pages without the need for writing script.

If you want to deliver dynamic content by using the other supported HTTP headers, you will need to understand how to generate the HTTP headers by using the desired programming language and have common gateway interface (CGI) or script access on the target web server.

## Audio Clips

You can serve audio clips to the phone from a web server by using the “audio/basic” MIME type setting. When this MIME type is used, the body of the response should contain raw audio data in the same format that is used for custom Cisco IP Phone rings. Refer to the chapter on “Custom Phone Rings” in the *Cisco CallManager System Guide* (also available in the online help).

**Note** The audio file should not be longer than 5 seconds.

Use the following ASP sample script to set the MIME type and to serve the file that is specified in the #include command:

```
<%@ Language=JavaScript%>
<%
Response.ContentType = "audio/basic";
%><!--#include file="filename.raw" --><% Response.End();%>
```

Using script to generate the MIME header when playing a sound provides an advantage because you may also include a refresh header to take the phone to a subsequent URL. Usually, you can set the MIME type for pages in any web server by simply performing an association to the .xml or .raw file extension. Your web server documentation should explain how to accomplish this. This action allows you to serve static pages without the need for writing script.

## Content Expiration Header Setting

The expiration header can control which URLs are added to the phone URL history. This behavior differs slightly from traditional web browsers but is implemented to perform the same function. Disable the back button functionality to avoid calling a URL twice.

This functionality allows you to make the content of any page that is sent to the phone expire. When a user presses the Exit softkey, the user goes back to the last URL that did not expire when it was loaded. This differs from traditional browsers by not considering the current freshness of the data but the freshness of the data when the URL was requested. This requires you to have a page expire when it is first loaded and to not set a time and date in the future.

The following example shows how to have content on IIS expire by using Active Server Page (ASP):

```
<%@ Language=JavaScript %>
<%
   Response.ContentType = "text/xml";
   Response.Expires = -1;
%>
```

The “Expires” property specifies the number of minutes to wait for the content to expire. Setting this value to -1 subtracts 1 minute from the request time and returns a date and time that have already passed.

## Set-Cookie Header Setting

A “cookie” is a term for a mechanism that the Web server uses to give the client a piece of data and have the client return the data with each request. The two traditional uses for cookies are:

- For Web sites to store a unique identifier and/or other information on the client's file system. The information is available to the Web server on subsequent visits.
- To track a unique identifier for state management. The client returns the cookie with each request and the server uses this identifier to index information about the current session. The identifier is commonly referred to as a session ID. Most Web servers have a built-in session management layer that uses this second type of cookie, which is commonly referred to as a session cookie.

The following example shows the Set-Cookie header that is returned to the browser when a request method is used:

```
Set-Cookie: ASPSESSIONIDGQGQGRLS=OCPNMLFDBJIPNIOOKFNFMOAL; path=/
```

The Cisco IP Phone can receive and use a total of four cookies per host per session and can store information for up to eight sessions at once. Each cookie can be up to 255 bytes in size. These cookies are available until the server terminates the session or the client session has been idle for more than 30 minutes. If the client is connecting to a new server and all session resources are in use, the client clears and reuses the session with the longest inactivity time.

When using ASP on IIS the default server configuration automatically generates a session cookie and sends it to the client using the Set-Cookie header. This enables you to utilize the Session object from within ASP to store and retrieve data spanning multiple requests for the life of the session. When using JSP on Tomcat, the default configuration generates and issues a session cookie.

## Identifying the Capabilities of IP Phone Clients

Because XML services are now supported across a wide range of Cisco IP Phones, web application servers now need to identify the capabilities of the requesting IP phone to optimize the content returned to the phone. For example, if the requesting phone is a Cisco IP Phone 7960, which cannot support color PNG images, the application server must be able to identify this and return a gray scale CIP image instead.

The IP phone client request to send the relevant information from the IP phone to the web server application includes three (3) HTTP headers:

- x-CiscoIPPhoneModelName
- x-CiscoIPPhoneDisplay
- x-CiscoIPPhoneSDKVersion

## x-CiscoIPPhoneModelName

This Cisco-proprietary header contains the Cisco manufacturing Model Name of the device, which can typically be found by going to **Settings > Model Information**, but varies between different models. Some examples of manufacturing Model Names are CP-7960, CP-7960G, CP-7940G, CP-7905G, and CP-7970G.

## x-CiscoIPPhoneDisplay

This Cisco-proprietary header contains the display capabilities of the requesting device with the following four parameters (listed in the order in which they appear):

- Width (in pixels)
- Height (in pixels)
- Color depth (in bits)
- A single character indicating whether the display is color ("C") or gray scale ("G")

These parameters get separated by commas as shown in the following example of a Cisco IP Phone 7970 header:

```
x-CiscoIPPhoneDisplay: 298, 168, 12, C
```

**Note** The pixel resolutions advertised by the device define the area of the display accessible by the phone services; not the actual resolution of the display.

## x-CiscoIPPhoneSDKVersion

This Cisco-proprietary header contains the version of the IP Phone Services SDK that the requesting phone supports. The HTTP header does not specify which URIs are supported. Therefore, you must check the "Supported URIs" matrix in the IP Phone Services SDK to determine which URIs are supported based on the Phone Model Name and supported SDK version.

See the "URIs Supported for Release 4.1(3) Cisco IP Phone Services SDK" to find out which IP phone models support the URIs that are documented in this SDK.

**Note** Beginning with the IP Phone Services SDK 3.3(3), the SDK version number matches the minimum Cisco CallManager software that is required to support it. For example, SDK version 3.3(4) gets supported only on Cisco CallManager version 3.3(4) or later.

## Accept Header

The Accept header represents a standard HTTP header that is used to inform web servers about the content-handling capabilities of the client.

Cisco IP Phones include proprietary content-types to indicate which XML objects are supported. These proprietary content-types all begin with x-CiscoIPPhone, to indicate Cisco IP Phone XML objects, followed by a slash "/", followed by either a specific XML object or a "*" to indicate all objects.

For example, x-CiscoIPPhone/* indicates that all XML objects defined in the specified version of the SDK are supported, and x-CiscoIPPhone/Menu specifies that the `<CiscoIPPhoneMenu>` object gets supported.

As the example illustrates, the name of the XML object can be derived directly from the content-type by appending the sub-type (the part after the slash) onto "CiscoIPPhone." The content-type can also include an optional version to indicate support for a particular SDK version of that object. If a version is not specified, then the x-CiscoIPPhoneSDKVersion is implied. The syntax of the version number may vary, but, in general, will be as follows:

```
<major version>.<minor version>.<maintenance version>
```

Here are some examples of typical content-types:

x-CiscoIPPhone/*;version=3.3.3

x-CiscoIPPhone/Text

x-CiscoIPPhone/Menu;version=3.3.4

## Accessing IP Phone Information

Cisco IP Phones have an embedded web server to provide a programming interface for external applications and a debugging and management interface for system administrators.

You can access the administrative pages using a standard web browser and pointing to the IP address of the phone with: /http://<*phoneIP*>/, where *phoneIP* is the IP address of the specific phone.

These device information pages are available in either HTML format, for manual debugging purposes, or in XML format for automation purposes. Table 5-1 lists the available URLs and their purpose:

***Table 5-1 Device Information URLs***

| HTML URL | XML URL | Description |
|---|---|---|
| /DeviceInformation | /DeviceInformationX | General device information |
| /NetworkConfiguration | /NetworkConfigurationX | Network configuration information |
| /EthernetInformation | /EthernetInformationX | Ethernet counters |
| /PortInformation?n | /PortInformationX?n | Detailed port information, where *n* is a model-specific ethernet port identifier, typically in the range 1- 3. |
| /DeviceLog?n | /DeviceLogX?n | Device logging, debug, and error messages, where *n* is a model-specific log number, typically in the range 0 - 2. |
| /StreamingStatistics?n | /StreamingStatisticsX?n | Current RTP streaming stats, where 'n' is model-specific RTP stream identifier, typically in the range 1-3. |

Table 5-1 Device Information URLs (continued)

| HTML URL | XML URL | Description |
|---|---|---|
| /CGI/Execute[1] | | The target URL of a phone push (HTTP POST) request. |
| /CGI/Screenshot[1] | | Returns an exact snapshot of the current phone display. The size and format of the image returned is model-specific. |

1. Password-protected CGI script

# IP Phone Service Administration and Subscription

Cisco CallManager administrators maintain the list of services to which users can subscribe. These sections provide details about administering Cisco IP Phone Services using Cisco CallManager Administration.

- Overview
- Adding a Phone Service
- Defining IP Phone Service Parameters in Cisco CallManager Administration
- User Service Subscription

## Overview

Administrators must use Cisco CallManager Administration to add and administer Cisco IP Phone services. To access phone service administration, open Cisco CallManager Administration and choose **Feature** > **Cisco IP Phone Services**:

- Phone services can have any number of parameters associated with them.
- You can specify phone service parameters as optional or required, depending on how the phone service application defines them.
- Users can subscribe to any service configured in their cluster, using their User Options web pages.
- Service subscriptions currently occur on a device basis.

A URL constitutes the core of each service. When a service is chosen from the menu, the URL gets requested via HTTP, and a server somewhere provides the content. The Service URL field shows this URL entry. For the services to be available, the phones in the Cisco CallManager cluster must have network connectivity to the server.

### Example

`http://`*<servername>*`/ccmuser/sample/sample.asp`

*Where*

<*servername*> designates a fully qualified domain name or an IP address.

## Adding a Phone Service

To access phone service administration, open Cisco CallManager Administration and choose **Feature** > **Cisco IP Phone Services**:

The Cisco IP Phone Services Configuration page in Cisco CallManager Administration contains the following fields:

*Table 6-1 Adding a Cisco IP Phone Service*

| Field | Description |
|---|---|
| Service Name | Enter the name of the service as it will display on the menu of available services in the Cisco IP Phone User Options application. Enter up to 32 characters for the service name. |
| Service Description | Enter a description of the content that the service provides. |

*Table 6-1 Adding a Cisco IP Phone Service (continued)*

| Field | Description |
|---|---|
| Service URL | Enter the URL of the server where the Cisco IP Phone Services application is located. Make sure that this server remains independent of the servers in your Cisco CallManager cluster. Do not specify a Cisco CallManager server or any server that is associated with Cisco CallManager (such as a TFTP server or directory database publisher server).<br><br>For the services to be available, the phones in the Cisco CallManager cluster must have network connectivity to the server.<br><br>When defining the service URL, you can embed a special #DEVICENAME# substitution tag within the URL. This tag provides a convenient method for IP phones to pass their device name to a web application server. For example, if a service URL was defined in CCMAdmin as: `http://myserver/myscript?name=#DEVICENAME#`, when a phone actually makes the HTTP request for the service, the requested URL will appear as: `http://myserver/myscript?name=SEP000123456789` |
| Character Set | If you are using a language other than English for Service Name and Description, choose the character set for that language. Text that is input by the user displays incorrectly if the wrong character set is chosen. |

**Tip**

- You can insert, update, or delete a service definition.
- After a service is inserted, you can insert, update, or delete service parameter definitions.
- When you delete a Cisco IP Phone subscription, Cisco CallManager removes all service information, user subscriptions, and user subscription data from the database.
- The **Update Subscriptions** button rebuilds all user subscriptions if the service has been modified after subscriptions exist.

## Defining IP Phone Service Parameters in Cisco CallManager Administration

Each service can have a list of parameters. You can use these parameters, which are appended to the URL when they are sent to the server, to personalize a service for an individual user. Examples of parameters include stock ticker symbols, city names, or user IDs. The service provider defines the semantics of a parameter.

The Cisco IP Phone Service Parameter Configuration page in Cisco CallManager Administration contains the following fields:

*Table 6-2 Defining IP Phone Service Parameters*

| Field | Description |
|---|---|
| Parameter Name | Enter the exact query string parameter to use when you build the subscription URL; for example, symbol. |
| Parameter Display Name | Enter a descriptive parameter name to display to the user in the Cisco IP Phone User Options application; for example, Ticker Symbol. |
| Default Value | Enter the default value for the parameter. This value displays to the user when a service is being subscribed to for the first time; for example, CSCO. |
| Parameter Description | Enter a description of the parameter. The user can access the text that is entered here while the user is subscribing to the service. The parameter description should provide information or examples to help users input the correct value for the parameter. |
| Parameter is Required | If the user must enter data for this parameter before the subscription can be saved, check the Parameter is Required check box. |
| Parameter is a Password (mask contents) | You can mask entries in the Cisco IP Phone User Options application, so asterisks display rather than the actual user entry. You may want to do this for parameters such as passwords that you do not want others to be able to view. To mask a parameter entry, check the Parameter is a Password (mask contents) check box in the Configure Cisco IP Phone Service Parameter window in Cisco CallManager Administration. |

**Tip** If you change the service URL, remove a Cisco IP Phone service parameter, or change the Parameter Name of a phone service parameter for a phone service to which users are already subscribed, be sure to click **Update Subscriptions** to update all currently subscribed users with the changes. If you do not do so, users must resubscribe to the service to rebuild the URL correctly.

## User Service Subscription

End users can configure service subscriptions using the CCMUser web site. After users log in and choose a device, a list of services that are assigned to the phone displays. The user can then configure these services, adding additional ones or removing un-used services. These password-protected windows are authenticated via the LDAP directory.

Users can personalize their services using the User Options pages to:

- Customize the name of the service.
- Enter any available service parameters.
- Review the description of each parameter.

After all the required fields are set, the user clicks **Subscribe** to add the services. A custom URL gets built and stored in the database for this subscription. The service then appears on the device services list.

# Troubleshooting Cisco IP Phone Service Applications

This chapter contains the following sections:

- Troubleshooting Tips
- XML Parsing Errors
- Error Messages

## Troubleshooting Tips

The following tips apply to troubleshooting Cisco IP Phone service applications:

- Microsoft Internet Explorer 5 or higher can display the XML source with its default style sheet.
- Understand that standard IP troubleshooting techniques are important for HTTP errors.
- Externally verify name resolution (Phone has DNS set).
- If DNS is suspected, use IP addresses in URLs.
- Browse the URL in question with Microsoft Internet Explorer or download and verify with Netscape Navigator.
- Use a logged telnet session to verify that the desired HTTP headers are returned (Telnet to the server on port 80; then, enter get /path/page).

## XML Parsing Errors

The following tips apply to troubleshooting XML parsing errors in Cisco IP Phone services applications:

- Verify the object tags (the object tags are case sensitive).
- Verify that "&" and the other four special characters are used per the restrictions while inside the XML objects. See Chapter 2, "CiscoIPPhone XML Objects" for more information.

## Error Messages

The following error messages may appear on the prompt line of the Cisco IP Phone display.

- `XML Error[4] = XML Parser error (Invalid Object)`
- `HTTP Error[8] = Unknown HTTP Error`
- `HTTP Error[10] = HTTP Connection Failed`

Refer to the *Cisco IP Phone Administration Guide for Cisco CallManager* for more information.

# DeviceListX Report

The DeviceListX Report provides a list of the services-capable devices along with basic information about the device to identify or classify the devices based on specific criteria. The report also includes the current device status and the IP address information that is obtained from the Real-Time Information Service.

These sections provide details about the DeviceListX Report:

- Benefits
- Restrictions
- Integration Considerations and Interoperability
- Performance and Scalability
- Security
- Related Features and Technologies
- Supported Platforms
- Prerequisites
- Message and Interface Definitions
- DeviceList XML Object
- Troubleshooting DeviceListX Reports

**Note** Not all device types are supported by DeviceListX . If you have a device that you need to support, contact Cisco Developer Support to verify whether it is supported:
http://www.cisco.com/go/developersupport

When a third-party developer initiates an **HTTP GET** request for the DeviceListX.asp report page, the system retrieves the following information about phones that are registered to a Cisco CallManager server from the database:

- Device Type
- Device Name
- Device Description
- Calling Search Space
- Device Pool
- IP Address
- Real-Time Information

The completed list of data gets formatted into a simple XML object and gets returned in the HTTP Response to the developer.

## Benefits

DeviceListX provides access to critical real-time data that was previously unavailable to third-party developers. In particular, the ability to list currently registered devices along with their IP address allows developers to easily build push, broadcast, and CTI-type applications.

## Restrictions

Only users with administrative privileges to the Cisco CallManager Administration can access the report.

**Note** To minimize processing overhead on the Cisco CallManager server, access to the DeviceListX report gets rate-limited to once per minute. Any attempt to pull the report more frequently will fail. In practice, the developer application should pull and cache the DeviceListX report, refreshing only as often as required, typically every few hours or daily.

## Integration Considerations and Interoperability

The interface allows HTTP 1.1 or HTTP 1.0 **GET** requests for the report. The report returns data that is encapsulated by using XML version 1.0.

## Performance and Scalability

You can run this report on the largest supported Cisco CallManager cluster size for the targeted release without impacting core features, such as delaying dial tone. On multiserver Cisco CallManager clusters, the report can access only from the publisher server. In large clusters where the publisher is not a Cisco CallManager server, no possibility exists of impacting the system performance as perceived by a user.

Because this report is not intended for use during real time, this interface should provide a mechanism for developers to poll for the data on a daily or hourly basis. Give consideration to the frequency of polling and the time of day to prevent unnecessary burden on the system during peak usage times.

## Security

This report, which is within the Cisco CallManager Administration, inherits its security from that web site, so no security issues directly relate to this report. If the Cisco CallManager Administration changes how it implements security with additions, such as SSL, this report benefits from that enhancement.

## Related Features and Technologies

DeviceListX acts as an independent interface, which is a real-time complement to the AVVID XML-Layer Database API (AXL), where AXL provides access to static, persisted data, and DeviceListX provides access to dynamic, volatile information.

## Supported Platforms

For the DeviceListX.asp page to function requires Cisco CallManager Administration reporting infrastructure. The following releases support DeviceListX.asp:

- Cisco CallManager Release 3.2(3)SPB
- Cisco CallManager Release 4.0(1) and later

## Prerequisites

You can access this feature when devicelistX.asp resides in the C:\ciscoWebs\Admin\reports directory of the Cisco CallManager publisher server.

## Message and Interface Definitions

Use the following URL to access the report via HTTP:

```
http://x.x.x.x/CCMAdmin/reports/devicelistx.asp
```

*where*

**x.x.x.x** can either be the IP address or hostname of the Cisco CallManager system that contains the report.

**Note** Beginning with CiscoCallManager 4.1 release, the DeviceListX report can only be accessed via secure HTTP (HTTPS), so the URL must begin with “https:” rather than “http:”.

## DeviceList XML Object

Third-party applications that reside elsewhere on the network commonly use the interface. The application makes an HTTP request for the report and gets a response that contains a DeviceList XML object. The XML object follows:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<DeviceList>
<Device t="" n="" d="" c="" p="" i="" s="" />
</DeviceList>
```

***Table 8-1 DeviceList XML Object Attributes***

| Attribute Name | Field Name | Description |
|---|---|---|
| t | Device Type | Numeric enumeration value that is specified in the database. |
| n | Device Name | String value that specifies the device name. |
| d | Device Description | String value that is specified in the database. |
| c | Device Calling Search Space | String value that is specified in the database. |
| p | Device Pool | String value that is specified in the database. |
| i | Device IP Address | Last known IP address as reported by the Real-Time Information Service<br>"" = No known IP address<br>"x.x.x.x" = Last known IP address |
| s | Device Status | Numeric enumeration for the current device status as reported by the Real-Time Information Service<br>"" = Device not found<br>"1" = Device registered<br>"2" = Device found but not currently registered |

***Example 1 DeviceList Object with Data***

```
<?xml version="1" encoding="iso-8859-1"?>
<DeviceList>
<Device t="35" n="SEP000123456789" d="Auto 2010" c=”” p="Default"
i="10.1.1.1" s="1"/>
</DeviceList>
```

## Troubleshooting DeviceListX Reports

These sections can assist you in troubleshooting DeviceListX Reports:

- Error Codes
- Determining Problems With the Interface

## Error Codes

The error codes that are specific to this report interface follow.

**Error Message** `1001 Too many simultaneous requests for Device List. Please wait at least 60 seconds and try again.`

**Explanation** When two or more clients attempt to get the list at the same time, or if the list is long, overlapping requests can result (first request is processing when the second request attempts processing).

**Recommended Action** Request information only as often as necessary.

**Note** Cisco recommends that you wait longer than 60 seconds between requests.

**Error Message** `1002 Too many consecutive requests for Device List. Please wait at least 60 seconds and try again.`

**Explanation** Because the system is busy, it cannot process a Device List.

**Recommended Action** Request information only as often as necessary. Because the real-time status of every device gets checked, Device List represents a CPU-intensive process.

**Note** Cisco recommends that you wait longer than 60 seconds between requests.

## Determining Problems With the Interface

Use the following procedure to determine whether a problem exists with the interface and determine the root cause of the problem.

**Step 1** Check the Windows NT Event Logs for error messages that pertain to the IIS server and the SQL server.

**Start > Programs > Administrative Tools > Event Viewer**

**Step 2** Check for error messages or successful completion of a request in the IIS log files, which are typically located in

C:\WINNT\System32\LogFiles\W3SVC1

The date of the log provides part of the log name. All times in the log files specify GMT for noted events. The IIS logs appear in chronological order and can easily be searched by specific query event.

**Step 3** Use a web browser, such as IE, to request the URL of the devicelistx.asp web page. A successful request yields a well-formed XML object of all the device information.

**Step 4** Use a Sniffer trace to view the HTTP GET request and response transaction between the third-party application and the report.

**Step 5** If you need further assistance, see the "Obtaining Technical Assistance" section on page xi.

# CiscoIPPhone XML Object Quick Reference

Table A-1 provides a quick reference of the CiscoIPPhone XML objects and the definitions that are associated with each.

*Table A-1 CiscoIPPhone XML Object Quick Reference*

| Object | Definition |
|---|---|
| CiscoIPPhoneMenu | `<CiscoIPPhoneMenu>`<br>`  <Title>Title text goes here</Title>`<br>`  <Prompt>Prompt text goes here</Prompt>`<br>`  <MenuItem>`<br>`    <Name>The name of each menu item</Name>`<br>`    <URL>The URL associated with the menu item</URL>`<br>`  </MenuItem>`<br>`</CiscoIPPhoneMenu>` |
| CiscoIPPhoneText | `<CiscoIPPhoneText>`<br>`  <Title>Title text goes here</Title>`<br>`  <Prompt>The prompt text goes here</Prompt>`<br>`  <Text>Text to display as the message body goes here</Text>`<br>`</CiscoIPPhoneText>` |

*Table A-1 CiscoIPPhone XML Object Quick Reference (continued)*

| Object | Definition |
|---|---|
| `CiscoIPPhoneInput` | `<CiscoIPPhoneInput> <Title>Directory title goes here</Title> <Prompt>Prompt text goes here</Prompt> <URL>The target URL for the completed input goes here</URL> <InputItem> <DisplayName>Name of input field to display</DisplayName> <QueryStringParam>The parameter to be added to the target URL</QueryStringParam> <DefaultValue>Value</DefaultValue> <InputFlags>The flag specifying the type of allowable input</InputFlags> </InputItem> </CiscoIPPhoneInput>` |
| `CiscoIPPhoneDirectory` | `<CiscoIPPhoneDirectory> <Title>Directory title goes here</Title> <Prompt>Prompt text goes here</Prompt> <DirectoryEntry> <Name>The name of the directory entry</Name> <Telephone>The telephone number for the entry</Telephone> </DirectoryEntry> </CiscoIPPhoneDirectory>` |
| `CiscoIPPhoneImage` | `<CiscoIPPhoneImage> <Title>Image title goes here</Title> <Prompt>Prompt text goes here</Prompt> <LocationX>Position information of graphic</LocationX> <LocationY>Position information of graphic</LocationY> <Width>Size information for the graphic</Width> <Height>Size information for the graphic</Height> <Depth>Number of bits per pixel</Depth> <Data>Packed Pixel Data</Data> </CiscoIPPhoneImage>` |
| `CiscoIPPhoneImageFile` | `<CiscoIPPhoneImageFile> <Title>Image Title goes here</Title> <Prompt>Prompt text goes here</Prompt> <LocationX>Horizontal position of graphic</LocationX> <LocationY>Vertical position of graphic</LocationY> <URL>Points to the PNG image</URL> </CiscoIPPhoneImageFile>` |

*Table A-1 CiscoIPPhone XML Object Quick Reference (continued)*

| Object | Definition |
|---|---|
| `CiscoIPPhoneGraphic` | `<CiscoIPPhoneGraphicMenu>` |
| `Menu` | `  <Title>Menu title goes here</Title>` |
| | `  <Prompt>Prompt text goes here</Prompt>` |
| | `  <LocationX>Position information of graphic</LocationX>` |
| | `  <LocationY>Position information of graphic</LocationY>` |
| | `  <Width>Size information for the graphic</Width>` |
| | `  <Height>Size information for the graphic</Height>` |
| | `  <Depth>Number of bits per pixel</Depth>` |
| | `  <Data>Packed Pixel Data</Data>` |
| | `  <MenuItem>` |
| | `    <Name>The name of each menu item</Name>` |
| | `    <URL>The URL associated with the menu item</URL>` |
| | `  </MenuItem>` |
| | `</CiscoIPPhoneGraphicMenu>` |
| `CiscoIPPhoneGraphic` | `<CiscoIPPhoneGraphicFileMenu>` |
| `FileMenu` | `  <Title>Image Title goes here</Title>` |
| | `  <Prompt>Prompt text goes here</Prompt>` |
| | `  <LocationX>Horizontal position of graphic</LocationX>` |
| | `  <LocationY>Vertical position of graphic</LocationY>` |
| | `  <URL>Points to the PNG background image</URL>` |
| | `  <MenuItem>` |
| | `    <Name>Same as CiscoIPPhoneGraphicMenu</Name>` |
| | `    <URL>Invoked when the TouchArea is touched</URL>` |
| | `    <TouchArea X1="left edge" Y1="top edge" X2="right` |
| | `edge"Y2="bottom edge"/>` |
| | `  </MenuItem>` |
| | `</CiscoIPPhoneGraphicFileMenu>` |
| `CiscoIPPhoneIconMenu` | `<CiscoIPPhoneIconMenu>` |
| | `  <Title>Title text goes here</Title>` |
| | `  <Prompt>Prompt text goes here</Prompt>` |
| | `  <MenuItem>` |
| | `    <IconIndex>Indicates what IconItem to display</IconIndex>` |
| | `    <Name>The name of each menu item</Name>` |
| | `    <URL>The URL associated with the menu item</URL>` |
| | `  </MenuItem>` |
| | `<IconItem>` |
| | `    <Index>A unique index from 0 to 9</Index>` |
| | `    <Height>Size information for the icon</Height>` |
| | `    <Width>Size information for the icon</Width>` |
| | `    <Depth>Number of bits per pixel</Depth>` |
| | `    <Data>Packed Pixel Data</Data>` |
| | `  </IconItem>` |
| | `</CiscoIPPhoneIconMenu>` |

*Table A-1 CiscoIPPhone XML Object Quick Reference (continued)*

| Object | Definition |
|---|---|
| CiscoIPPhoneIconFile Menu | `<CiscoIPPhoneIconFileMenu>` `<Title>Title text goes here</Title>` `<Prompt>Prompt text goes here</Prompt>` `<MenuItem>` `<IconIndex>Indicates what IconItem to display</IconIndex>` `<Name>The name of each menu item</Name>` `<URL>The URL associated with the menu item</URL>` `</MenuItem>` `<IconItem>` `<Index>A unique index from 0 to 9</Index>` `<URL>location of the PNG icon image</URL>` `</IconItem>` `</CiscoIPPhoneIconFileMenu>` |
| CiscoIPPhoneStatus | `<CiscoIPPhoneStatus>` `<Text>This is the text area</Text>` `<Timer>Timer seed value in seconds</Timer>` `<LocationX>Horizontal alignment</LocationX>` `<LocationY>Vertical alignment</LocationY>` `<Width>Pixel width of graphic</Width>` `<Height>Pixel height of graphic</Height>` `<Depth>Color depth in bits</Depth>` `<Data>Hex binary image data</Data>` `</CiscoIPPhoneStatus>` |
| CiscoIPPhoneStatusFile | `<CiscoIPPhoneStatusFile>` `<Text>This is the text area</Text>` `<Timer>Timer seed value in seconds</Timer>` `<LocationX>Horizontal alignment</LocationX>` `<LocationY>Vertical alignment</LocationY>` `<URL>location of the PNG image</URL>` `</CiscoIPPhoneStatusFile>` |
| CiscoIPPhoneExecute | `<CiscoIPPhoneExecute>` `<ExecuteItem URL=”The URL or URI to be executed”/>` `</CiscoIPPhoneExecute>` |
| CiscoIPPhoneError | `<CiscoIPPhoneError Number=”x”/>` |
| CiscoIPPhoneResponse | `<CiscoIPPhoneResponse>` `<ResponseItem Status”the success or failure of the action”Data=”the information associated with the request” URL=”the URL or URI specified in the Execute object”/>` `</CiscoIPPhoneResponse>` |

# Cisco IP Phone XML Schema File

## CiscoIPPhone.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Kelly
Stearns (Cisco Systems, Inc.) -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified"
version="3.3.4">
    <xsd:complexType name="CiscoIPPhoneExecuteItemType">
        <xsd:attribute name="Priority" use="optional">
            <xsd:simpleType>
                <xsd:restriction base="xsd:unsignedByte">
                    <xsd:minInclusive value="0"/>
                    <xsd:maxInclusive value="2"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="URL" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="256"/>
                    <xsd:minLength value="1"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:complexType name="CiscoIPPhoneExecuteType">
        <xsd:sequence>
            <xsd:element name="ExecuteItem"
type="CiscoIPPhoneExecuteItemType" maxOccurs="3"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CiscoIPPhoneResponseItemType">
        <xsd:sequence>
            <xsd:element name="Status" type="xsd:short"/>
            <xsd:element name="Data">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="URL">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="256"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CiscoIPhoneResponseType">
        <xsd:sequence>
            <xsd:element name="ResponseItem"
type="CiscoIPPhoneResponseItemType" maxOccurs="3"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CiscoIPPhoneTextType">
        <xsd:sequence>
            <xsd:element name="Title" nillable="true" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Prompt" nillable="true" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Text">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="4000"/>
                        <xsd:minLength value="0"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="SoftKey" type="CiscoIPPhoneSoftKeyType"
minOccurs="0" maxOccurs="8"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CiscoIPPhoneInputItemType">
            <xsd:sequence>
                <xsd:element name="DisplayName" nillable="true">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="32"/>
                            <xsd:minLength value="0"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="QueryStringParam" nillable="false">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="32"/>
                            <xsd:minLength value="1"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="InputFlags" nillable="false">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="A"/>
                            <xsd:enumeration value="T"/>
                            <xsd:enumeration value="N"/>
                            <xsd:enumeration value="E"/>
                            <xsd:enumeration value="U"/>
                            <xsd:enumeration value="L"/>
                            <xsd:enumeration value="AP"/>
                            <xsd:enumeration value="TP"/>
                            <xsd:enumeration value="NP"/>
                            <xsd:enumeration value="EP"/>
                            <xsd:enumeration value="UP"/>
                            <xsd:enumeration value="LP"/>
                            <xsd:enumeration value="PA"/>
                            <xsd:enumeration value="PT"/>
                            <xsd:enumeration value="PN"/>
                            <xsd:enumeration value="PE"/>
                            <xsd:enumeration value="PU"/>
                            <xsd:enumeration value="PL"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="DefaultValue" nillable="true">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="32"/>
                            <xsd:minLength value="0"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CicsoIPPhoneInputType">
            <xsd:sequence>
                <xsd:element name="Title" nillable="true" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:minLength value="0"/>
                            <xsd:maxLength value="32"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="Prompt" nillable="true" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:minLength value="0"/>
                            <xsd:maxLength value="32"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="URL" nillable="false">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="256"/>
                            <xsd:minLength value="0"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="InputItem"
type="CiscoIPPhoneInputItemType" maxOccurs="5"/>
                <xsd:element name="SoftKeyItem"
type="CiscoIPPhoneSoftKeyType" minOccurs="0" maxOccurs="8"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CiscoIPPhoneGraphicMenuType">
            <xsd:sequence>
                <xsd:element name="Title" nillable="true" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Prompt" nillable="true" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationX" nillable="false" 
minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:short">
                        <xsd:minInclusive value="-1"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationY" nillable="false" 
minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:short">
                        <xsd:minInclusive value="-1"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Width">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedShort">
                        <xsd:minInclusive value="1"/>
                        <xsd:maxInclusive value="133"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Height">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedShort">
                        <xsd:minInclusive value="1"/>
                        <xsd:maxInclusive value="65"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Depth">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedShort">
```

```
                        <xsd:minInclusive value="1"/>
                        <xsd:maxInclusive value="2"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Data">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:hexBinary">
                        <xsd:maxLength value="2162"/>
                        <xsd:minLength value="1"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="MenuItem"
type="CiscoIPPhoneMenuItemType" minOccurs="0" maxOccurs="12"/>
            <xsd:element name="SoftKeyItem"
type="CiscoIPPhoneSoftKeyType" minOccurs="0" maxOccurs="8"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CiscoIPPhoneGraphicFileMenuType">
        <xsd:sequence>
            <xsd:element name="Title" nillable="true" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Prompt" nillable="true" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationX" nillable="false"
minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:short">
                        <xsd:minInclusive value="-1"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationY" nillable="false"
minOccurs="0">
                <xsd:simpleType>
```

```
                        <xsd:restriction base="xsd:short">
                            <xsd:minInclusive value="-1"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="URL" nillable="false">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="256"/>
                            <xsd:minLength value="1"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="MenuItem"
type="CiscoIPPhoneTouchAreaMenuItemType" minOccurs="0"
maxOccurs="32"/>
                <xsd:element name="SoftKeyItem"
type="CiscoIPPhoneSoftKeyType" minOccurs="0" maxOccurs="8"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CiscoIPPhoneTouchAreaMenuItemType">
            <xsd:sequence>
                <xsd:element name="Name" nillable="true">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:minLength value="0"/>
                            <xsd:maxLength value="32"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="URL" nillable="true">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:minLength value="0"/>
                            <xsd:maxLength value="256"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="TouchArea"
type="CiscoIPPhoneTouchArea"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CiscoIPPhoneTouchArea">
            <xsd:attribute name="X1" type="xsd:unsignedShort"
use="required"/>
            <xsd:attribute name="Y1" type="xsd:unsignedShort"
use="required"/>
```

```
        <xsd:attribute name="X2" type="xsd:unsignedShort"
use="required"/>
        <xsd:attribute name="Y2" type="xsd:unsignedShort"
use="required"/>
    </xsd:complexType>
    <xsd:complexType name="CiscoIPPhoneDirectoryEntryType">
        <xsd:sequence>
            <xsd:element name="Name" nillable="true">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="32"/>
                        <xsd:minLength value="0"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Telephone" nillable="false">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="32"/>
                        <xsd:minLength value="1"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CiscoIPPhoneDirectoryType">
        <xsd:sequence>
            <xsd:element name="Title" nillable="true" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Prompt" nillable="true" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="DirectoryEntry"
type="CiscoIPPhoneDirectoryEntryType" maxOccurs="32"/>
            <xsd:element name="SoftKey" type="CiscoIPPhoneSoftKeyType"
minOccurs="0" maxOccurs="8"/>
        </xsd:sequence>
```

```
</xsd:complexType>
<xsd:complexType name="CiscoIPPhoneMenuItemType">
    <xsd:sequence>
        <xsd:element name="Name" nillable="true">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:minLength value="0"/>
                    <xsd:maxLength value="64"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element name="URL" nillable="true">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="256"/>
                    <xsd:minLength value="0"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CiscoIPPhoneIconItemType">
    <xsd:sequence>
        <xsd:element name="Index" type="xsd:unsignedShort"/>
        <xsd:element name="Width" nillable="false">
            <xsd:simpleType>
                <xsd:restriction base="xsd:unsignedShort">
                    <xsd:minInclusive value="1"/>
                    <xsd:maxInclusive value="16"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element name="Height" nillable="false">
            <xsd:simpleType>
                <xsd:restriction base="xsd:unsignedShort">
                    <xsd:minInclusive value="1"/>
                    <xsd:maxInclusive value="10"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element name="Depth">
            <xsd:simpleType>
                <xsd:restriction base="xsd:unsignedShort">
                    <xsd:minInclusive value="1"/>
                    <xsd:maxInclusive value="2"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
```

```
                <xsd:element name="Data">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:hexBinary">
                            <xsd:maxLength value="40"/>
                            <xsd:minLength value="1"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CiscoIPPhoneIconMenuItemType">
            <xsd:sequence>
                <xsd:element name="Name" nillable="true">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:minLength value="0"/>
                            <xsd:maxLength value="64"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="URL" nillable="true">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="256"/>
                            <xsd:minLength value="0"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="IconIndex" nillable="false" 
minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:short">
                            <xsd:minInclusive value="0"/>
                            <xsd:maxInclusive value="9"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CiscoIPPhoneSoftKeyType">
            <xsd:sequence>
                <xsd:element name="Name" nillable="true">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="32"/>
                            <xsd:minLength value="0"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="Position" nillable="false">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:unsignedShort">
                            <xsd:minInclusive value="1"/>
                            <xsd:maxInclusive value="8"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="URL" nillable="true">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="256"/>
                            <xsd:minLength value="0"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="URLDown" nillable="true" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:minLength value="0"/>
                            <xsd:maxLength value="256"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CiscoIPPhoneMenuType">
            <xsd:sequence>
                <xsd:element name="Title" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:minLength value="0"/>
                            <xsd:maxLength value="32"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="Prompt" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:minLength value="0"/>
                            <xsd:maxLength value="32"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="MenuItem"
    type="CiscoIPPhoneMenuItemType" minOccurs="0" maxOccurs="100"/>
```

```
                <xsd:element name="SoftKeyItem"
type="CiscoIPPhoneSoftKeyType" minOccurs="0" maxOccurs="8"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CiscoIPPhoneImageType">
            <xsd:sequence>
                <xsd:element name="Title" nillable="true" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="32"/>
                            <xsd:minLength value="0"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="Prompt" nillable="true" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="32"/>
                            <xsd:minLength value="0"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="LocationX" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:short">
                            <xsd:minInclusive value="-1"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="LocationY" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:short">
                            <xsd:minInclusive value="-1"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="Width">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:unsignedShort">
                            <xsd:minInclusive value="1"/>
                            <xsd:maxInclusive value="133"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="Height">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:unsignedShort">
                            <xsd:minInclusive value="1"/>
                        <xsd:maxInclusive value="65"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Depth">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedShort">
                        <xsd:minInclusive value="1"/>
                        <xsd:maxInclusive value="2"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Data" nillable="false">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:hexBinary">
                        <xsd:maxLength value="2162"/>
                        <xsd:minLength value="1"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="SoftKeyItem"
type="CiscoIPPhoneSoftKeyType" minOccurs="0" maxOccurs="8"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CiscoIPPhoneImageFileType">
        <xsd:sequence>
            <xsd:element name="Title" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="32"/>
                        <xsd:minLength value="0"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Prompt" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="32"/>
                        <xsd:minLength value="0"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationX" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:short">
                        <xsd:minInclusive value="-1"/>
                    </xsd:restriction>
                </xsd:simpleType>
                </xsd:element>
                <xsd:element name="LocationY" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:short">
                            <xsd:minInclusive value="-1"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="URL" nillable="false">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="256"/>
                            <xsd:minLength value="1"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="SoftKeyItem"
type="CiscoIPPhoneSoftKeyType" minOccurs="0" maxOccurs="8"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CiscoIPPhoneIconMenuType">
            <xsd:sequence>
                <xsd:element name="Title" nillable="true" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:minLength value="0"/>
                            <xsd:maxLength value="32"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="Prompt" nillable="true" minOccurs="0">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:minLength value="0"/>
                            <xsd:maxLength value="32"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="MenuItem"
type="CiscoIPPhoneIconMenuItemType" minOccurs="0" maxOccurs="32"/>
                <xsd:element name="IconItem"
type="CiscoIPPhoneIconItemType" minOccurs="0" maxOccurs="10"/>
                <xsd:element name="SoftKeyItem"
type="CiscoIPPhoneSoftKeyType" minOccurs="0" maxOccurs="8"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="CiscoIPPhoneStatusType">
            <xsd:sequence>
            <xsd:element name="Text" nillable="true" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Timer" nillable="false" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedShort">
                        <xsd:minInclusive value="0"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationX" nillable="false"
minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:short">
                        <xsd:minInclusive value="-1"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationY" nillable="false"
minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:short">
                        <xsd:minInclusive value="-1"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Width" nillable="false">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedShort">
                        <xsd:minInclusive value="1"/>
                        <xsd:maxInclusive value="106"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Height" nillable="false">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedShort">
                        <xsd:minInclusive value="1"/>
                        <xsd:maxInclusive value="21"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Depth" nillable="false">
```

```
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedShort">
                        <xsd:minInclusive value="1"/>
                        <xsd:maxInclusive value="2"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Data" nillable="false">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:hexBinary">
                        <xsd:minLength value="1"/>
                        <xsd:maxLength value="557"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CiscoIPPhoneStatusFileType">
        <xsd:sequence>
            <xsd:element name="Text" nillable="true" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Timer" nillable="false" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedShort">
                        <xsd:minInclusive value="0"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationX" nillable="false"
minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:short">
                        <xsd:minInclusive value="-1"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationY" nillable="false"
minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:short">
                        <xsd:minInclusive value="-1"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="URL" nillable="false">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="1"/>
                        <xsd:maxLength value="256"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CiscoIPPhoneIconFileItemType">
        <xsd:sequence>
            <xsd:element name="Index" nillable="false">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:unsignedShort">
                        <xsd:minInclusive value="0"/>
                        <xsd:maxInclusive value="9"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="URL" nillable="false">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="1"/>
                        <xsd:maxLength value="256"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="CiscoIPPhoneIconFileMenuType">
        <xsd:sequence>
            <xsd:element name="Title" nillable="true" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="Prompt" nillable="true" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:minLength value="0"/>
                        <xsd:maxLength value="32"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="MenuItem"
type="CiscoIPPhoneIconMenuItemType" minOccurs="0" maxOccurs="32"/>
            <xsd:element name="IconItem"
type="CiscoIPPhoneIconFileItemType" minOccurs="0" maxOccurs="10"/>
            <xsd:element name="SoftKeyItem"
type="CiscoIPPhoneSoftKeyType" minOccurs="0" maxOccurs="8"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="CiscoIPPhoneMenu" type="CiscoIPPhoneMenuType"/>
    <xsd:element name="CiscoIPPhoneImage"
type="CiscoIPPhoneImageType"/>
    <xsd:element name="CiscoIPPhoneImageFile"
type="CiscoIPPhoneImageFileType"/>
    <xsd:element name="CiscoIPPhoneIconMenu"
type="CiscoIPPhoneIconMenuType"/>
    <xsd:element name="CiscoIPPhoneDirectory"
type="CiscoIPPhoneDirectoryType"/>
    <xsd:element name="CiscoIPPhoneGraphicMenu"
type="CiscoIPPhoneGraphicMenuType"/>
    <xsd:element name="CiscoIPPhoneGraphicFileMenu"
type="CiscoIPPhoneGraphicFileMenuType"/>
    <xsd:element name="CiscoIPPhoneInput"
type="CicsoIPPhoneInputType"/>
    <xsd:element name="CiscoIPPhoneText" type="CiscoIPPhoneTextType"/>
    <xsd:element name="CiscoIPPhoneExecute"
type="CiscoIPPhoneExecuteType"/>
    <xsd:element name="CiscoIPPhoneResponse"
type="CiscoIPhoneResponseType"/>
    <xsd:element name="CiscoIPPhoneError">
        <xsd:complexType>
            <xsd:attribute name="Number" type="xsd:unsignedShort"
use="required"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="CiscoIPPhoneStatus"
type="CiscoIPPhoneStatusType"/>
    <xsd:element name="CiscoIPPhoneIconFileMenu"
type="CiscoIPPhoneIconFileMenuType"/>
    <xsd:element name="CiscoIPPhoneStatusFile"
type="CiscoIPPhoneStatusFileType"/>
</xsd:schema>
```

INDEX