Cisco Unified IP Phone Services Application Development Notes

Supporting XML Applications

Release 7.1(3)

Americas 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: 408 527-0883

Text Part Number: OL-20949-01

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CCDE, CCENT, CCSI, Cisco Eos, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Nurse Connect, Cisco Pulse, Cisco SensorBase, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flipshare (Design), Flip Ultra, Flip Video, Flip Video (Design), Instant Broadband, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Capital, Cisco Capital (Design), Cisco:Financed (Stylized), Cisco Store, Flip Gift Card, and One Million Acts of Green are service marks; and Access Registrar, Aironet, AllTouch, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Lumin, Cisco Nexus, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, Continuum, EtherFast, EtherSwitch, Event Center, Explorer, Follow Me Browsing, GainMaker, iLYNX, IOS, iPhone, IronPort, the IronPort logo, Laser Link, LightStream, Linksys, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, PCNow, PIX, PowerKEY, PowerPanels, PowerTV, PowerTV (Design), PowerVu, Prisma, ProConnect, ROSA, SenderBase, SMARTnet, Spectrum Expert, StackWise, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0910R)

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.

Cisco Unified IP Phone Services Application Development Notes
Copyright © 2004-2009 Cisco Systems, Inc. All rights reserved.

CONTENTS

Preface vii

Overview vii

Revision History vii

Audience vii

Cisco Developer Support Program vii

Organization viii

Related Documentation ix

Obtaining Documentation and Submitting a Service Request i-ix

Cisco Product Security Overview x

Document Conventions x

CHAPTER 1 **Overview** 1-1

CHAPTER 2 **New and Changed Information** 2-1

New Information for Cisco Unified Communications Manager 7.1(3) 2-1

CHAPTER 3 **CiscoIPPhone XML Objects** 3-1

Understanding Object Behavior 3-1

XML Object Definitions 3-3

CiscoIPPhoneMenu 3-3

CiscoIPPhoneText 3-4

CiscoIPPhoneInput 3-5

CiscoIPPhoneDirectory 3-6

Custom Directories 3-7

CiscoIPPhoneImage 3-7

CiscoIPPhoneImageFile 3-9

CiscoIPPhoneGraphicMenu 3-11

CiscoIPPhoneGraphicFileMenu 3-11

CiscoIPPhoneIconMenu 3-12

CiscoIPPhoneIconFileMenu 3-14

CiscoIPPhoneStatus 3-14

CiscoIPPhoneStatusFile 3-17

CiscoIPPhoneExecute 3-18

CiscoIPPhoneResponse 3-19
CiscoIPPhoneError 3-19
Custom Softkeys 3-19
XML Considerations 3-20
Mandatory Escape Sequences 3-21
XML Encoding 3-21
Application Event Handlers 3-22

CHAPTER 4 **Component APIs** 4-1
Application Management API 4-1
RTP Streaming API 4-1
Interaction Rules with Legacy RTP URI Streams 4-2
RTP Streaming Schema 4-2
Error Schema 4-4
Examples 4-5
Errors and Responses 4-5

CHAPTER 5 **Internal URI Features** 5-1
Supported URIs by Phone Model 5-1
Device Control URIs 5-2
Key 5-2
Unsupported Key URIs and Alternate Options 5-4
Display 5-5
XML Displayable Object URIs 5-6
SoftKey 5-6
QueryStringParam 5-7
Multimedia URIs 5-9
RTP Streaming 5-9
RTPRx 5-10
RTPTx 5-11
RTPMRx 5-11
RTPMTx 5-11
Play 5-12
Vibrate 5-12
Telephony URIs 5-13
Dial 5-13
EditDial 5-14
SendDigits 5-14

Application Management URIs 5-16
Init 5-16
Notify 5-16
Application 5-18

CHAPTER 6 **HTTP Requests and Header Settings 6-1**
HTTP Client Requests (HTTP GET) 6-1
HTTP Server Requests (HTTP POST) 6-1
HTTP Header Settings 6-2
HTTP Refresh Setting 6-3
MIME Type and Other HTTP Headers 6-4
Audio Clips 6-4
Content Expiration Header Setting 6-4
Set-Cookie Header Setting 6-5
HTTP Encoding Header Setting 6-6
HTTP Response Headers: Content-Type 6-6
Identifying the Capabilities of IP Phone Clients 6-7
x-CiscoIPPhoneModelName 6-7
x-CiscoIPPhoneDisplay 6-7
x-CiscoIPPhoneSDKVersion 6-8
Accept Header 6-8
Accessing IP Phone Information 6-9

CHAPTER 7 **Troubleshooting Cisco Unified IP Phone Service Applications 7-1**
Troubleshooting Tips 7-1
XML Parsing Errors 7-1
Error Messages 7-2

CHAPTER 8 **Cisco IP Phone Services Software Development Kit (SDK) 8-1**
SDK Components 8-1
Sample Services Requirements 8-3

CHAPTER 9 **IP Phone Service Administration and Subscription 9-1**
Accessing Phone Service Administration 9-1
Adding a Phone Service 9-2
Defining IP Phone Service Parameters 9-3

User Service Subscription 9-4

CHAPTER 10 **DeviceListX Report** 10-1

Benefits 10-2

Restrictions 10-2

Integration Considerations and Interoperability 10-2

Performance and Scalability 10-2

Security 10-3

Related Features and Technologies 10-3

Supported Platforms 10-3

Prerequisites 10-3

Message and Interface Definitions 10-3

DeviceList XML Object 10-3

Troubleshooting DeviceListX Reports 10-4

Error Codes 10-4

Determining Problems With the Interface 10-5

APPENDIX A **CiscoIPPhone XML Object Quick Reference** A-1

APPENDIX B **Cisco Unified IP Phone Services XML Schema File** B-1

Updated XML Parser and Schema Enforcement B-1

CiscoIPPhone.xsd B-2

INDEX

Preface

Overview

Use this document with Cisco Unified Communications Manager, Release 7.1(3) to develop and deploy customized client services for the Cisco Unified IP Phones that support Cisco Unified Phone services.

Because of the complexity of a Unified Communications network, this guide does not provide complete and detailed information for procedures that you need to perform in Cisco Unified Communications Manager or other network devices. See the "Related Documentation" section on page ix for a list of related documentation.

Revision History

Date	Updates
September 29, 2009	Initial release.
November 23, 2009	Updated the document with information on the Cisco Unified IP Phones 9971, 9951, and 8961 models. See, New Information for Cisco Unified Communications Manager 7.1(3), page 1.
December 10, 2009	Corrected the Cisco Unified IP Phone 7925G resolution from 176 x 220 to 176 x 140. See, Table 3-2 on page 9

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.

Cisco Developer Support Program

The Cisco Developer Network (CDN) portal provides access to multiple Cisco technology developer interfaces and collaborative support communities. CDN also provides formalized support services for these interfaces to enable developers, customers, and partners to accelerate their development. The

formalized process provides access to CDN Engineers who are an extension of the product technology engineering teams. CDN Engineers have access to the resources necessary to provide expert support in a timely manner.

The Cisco Developer Network Program is designed for businesses (IHV's and ISV's) interested in going to market with Cisco. The CDN Program enables members to develop compelling solutions that unify data, voice, video, and mobile communications on Cisco's powerful communications platform. The program also allows members to take advantage of Cisco's brand, market leadership position, and installed base to help drive positive business results for themselves and their customers.

For additional information about the CDN Program and CDN support services go to *http://developer.cisco.com/web/devservices*

Note The Cisco TAC does NOT provide support for this API/interface under standard hardware or software support agreements. All technical support for this API/interface, from initial development assistance through API troubleshooting/bugs in final production apps, is provided by Cisco Developer Support and requires a separate Developer Support contract. When opening cases, a Developer Support contract number must be provided to receive support.

Organization

This document comprises the following sections:

Chapter	Description
Chapter 1, “Overview”	Provides an overview of the Cisco Unified IP Phone services for developers.
Chapter 2, “New and Changed Information”	Provides details on the new and changed information in the XML service interface for the latest release of Cisco Unified Communication Manager.
Chapter 3, “CiscoIPPhone XML Objects”	Describes the general behavior and usage of each XML object.
Chapter 4, “Component APIs”	Describes additional APIs available to the Cisco Unified IP Phones.
Chapter 5, “Internal URI Features”	Describes how to implement embedded features on Cisco Unified IP Phones.
Chapter 6, “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 7, “Troubleshooting Cisco Unified IP Phone Service Applications”	Provides troubleshooting tips, XML parsing errors, and error messages.
Chapter 8, “Cisco IP Phone Services Software Development Kit (SDK)”	Provides a list of the components used in the Cisco Unified IP Services Software Development Kit (SDK) and the sample services requirements.

Chapter	Description
Chapter 9, "IP Phone Service Administration and Subscription"	Describes how to add and administer Cisco Unified IP Phone Services through Cisco Unified Communications Manager Administration.
Chapter 10, "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 Unified IP Phone Services XML Schema File"	Provides the CiscoIPPhone XML Schema.

Related Documentation

For more information about Cisco Unified IP Phones or Cisco Unified Communications Manager, refer to the following publications:

Cisco Unified IP Phone 7900 Series

These publications are available at the following URL:

http://www.cisco.com/en/US/products/hw/phones/ps379/tsd_products_support_series_home.html

Cisco Unified Communications Manager Administration

Related publications are available at the following URL:

http://www.cisco.com/en/US/products/sw/voicesw/ps556/prod_maintenance_guides_list.html

Cisco Unified Communications Manager Business Edition

Related publications are available at the following URL:

http://www.cisco.com/en/US/products/ps7273/tsd_products_support_series_home.html

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS Version 2.0.

Cisco Product Security Overview

This product contains cryptographic features and is subject to United States and local country laws governing import, export, transfer and use. Delivery of Cisco cryptographic products does not imply third-party authority to import, export, distribute or use encryption. Importers, exporters, distributors and users are responsible for compliance with U.S. and local country laws. By using this product you agree to comply with applicable laws and regulations. If you are unable to comply with U.S. and local laws, return this product immediately.

Further information regarding U.S. export regulations may be found at http://www.access.gpo.gov/bis/ear/ear_data.html

Document Conventions

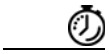
This document uses the following conventions:

Convention	Indication
bold font	Commands and keywords and user-entered text appear in **bold** font.
italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
[ ]	Elements in square brackets are optional.
{x | y | z }	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
`courier` font	Terminal sessions and information the system displays appear in `courier` font.
< >	Nonprinting characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Note Means *reader take note*.

Tip Means *the following information will help you solve a problem.*

Caution Means *reader be careful*. In this situation, you might perform an action that could result in equipment damage or loss of data.



Timesaver Means *the described action saves time*. You can save time by performing the action described in the paragraph.

Warning **Means *reader be warned*. In this situation, you might perform an action that could result in bodily injury.**

CHAPTER 1

Overview

You can use Cisco Unified 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.

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:

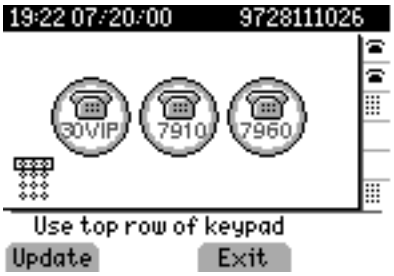
- Weather
- Stock information
- Contact information
- Company news
- To-do lists
- Daily schedule

Figure 1-1 shows a sample text menu.

Figure 1-1 Cisco Unified IP Phone Text Menu Sample

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

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



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 Unified 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 Plain Text Display Example

Figure 1-4 Graphic Image Display Example

Cisco Unified Communications Manager limits Cisco Unified IP Phone service activity to a specific Services pane in the Cisco Unified 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.

CHAPTER 2

New and Changed Information

This chapter gives details on the new and changed information in the XML service interface for the latest releases of Cisco Unified Communication Manager.

New Information for Cisco Unified Communications Manager 7.1(3)

The following are the updates made in the XML service interface for Cisco Unified Communication manager 7.1(3):

- Support for the latest 9971, 9951, and 8961 series Cisco IP phones. For more information see Table 3-1 on page 3-2
- The 9971, 9951, 8961 series IP phones does not support the CiscoIPPhoneStatus and CiscoIPPhoneStatusFile XML objects. For more information see, Table 3-1 on page 3-2.
- Key URIs supported in 9971, 9951, and 8961 series IP phones and alternate options for unsupported Key URIs. For more information see, “Key” section on page 5-2.
- Changes in softkey position, behavior, and design. For more information see, “SoftKey” section on page 5-6.

CHAPTER 3

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
- Application Event Handlers

Understanding Object Behavior

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

Regarding services, the phone does not have any concept of a state when it loads an XML page. Cisco Unified 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. 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 6, "HTTP Requests and Header Settings" for more information.

Table 3-1 shows the supported XML objects for this release.

Table 3-1 XML Objects Supported for Release 7.1(3) Cisco Unified IP Phone Services SDK

Phone Model XML Object	7905G 7906G 7911G 7912G 7931G	7920G	7921G 7925G	7940G 7960G	7941G/7941G-GE, 7942G, 7945G, 7961G/7961G-GE, 7962G, 7965G, 7970G/ 7971G-GE, 7975G, IP Communicator	6921, 6941, 6961	9971, 9951, 8961
CiscoIPPhoneMenu	X	X	X	X	X	X	X
CiscoIPPhoneText	X	X	X	X	X	X	X
CiscoIPPhoneInput	X	X	X	X	X	X	X
CiscoIPPhoneDirectory	X	X	X	X	X	X	X
CiscoIPPhoneImage	—	X[1]	X	X	X	—[2]	X
CiscoIPPhoneImageFile	—	—	X	—	X	—[2]	X
CiscoIPPhoneGraphicMenu	—	X[1]	X	X	X	—[2]	X
CiscoIPPhoneGraphicFileMenu	—	—	X	—	X	—[2]	X
CiscoIPPhoneIconMenu	X[3]	X	X	X	X	X[4]	X
CiscoIPPhoneIconFileMenu	—	—	X	—	X[5]	X[4]	X
CiscoIPPhoneStatus	—	—	—	X	X	—	—
CiscoIPPhoneStatusFile	—	—	—	—	X[5]	—	—
CiscoIPPhoneExecute	X	X[6]	X	X	X	X	X
CiscoIPPhoneResponse	X	X	X	X	X	X	X
CiscoIPPhoneError	X	X	X	X	X	X	X

1. The Cisco Unified IP Phone 7920G 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 Unified IP Phones 6921, 6941, and 6961 do not support CiscoIPPhoneGraphicFileMenu, CiscoIPPhoneGraphicMenu, CiscoIPPhoneImageFile, and CiscoIPPhoneImage because these phones use monochrome LCD.
3. The Cisco Unified IP Phones 7905G and 7912G do not support CIP images; therefore, all icons get ignored and does not display.
4. The Cisco Unified IP Phones 6921, 6941, and 6961 do not support icons; therefore, all icons are ignored and does not display.
5. The Cisco Unified IP Phones 7970G and 7971G-GE require firmware version 7.1(2) or higher to support this object, and Cisco IP Communicator requires software version 2.01 or higher.
6. The Cisco Unified IP Phone 7920G does not support Priority 1 when on a call.

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 6900 series IP phones does not display the *Title* and *Prompt* menu fields at the same time. If both *Title* and *Prompt* fields are defined at the same time, then these phones display only the *Prompt* field.

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 Unified 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 Unified IP Phone firmware controls all other pagination and wordwrap issues.

Note Cisco Unified IP Phones support the full ISO 8859-1 (Latin 1) and Shift_JIS character sets.

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>
```

Note The 6900 series IP phones does not display the *Title* and *Prompt* menu fields at the same time. If both *Title* and *Prompt* fields are defined at the same time, then these phones display only the *Prompt* field.

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 Unified Communications Manager 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.

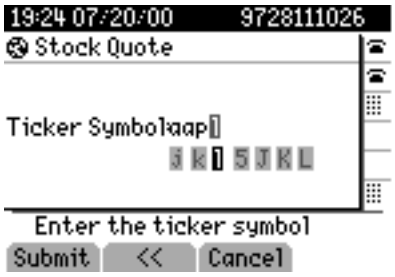
Note **Keypad navigation**: Cisco Unified IP Phones allow navigation to a specific line in a menu by pressing numeric DTMF keys. When a menu is on the display, the number for selecting the menu is on the left.

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 Unified 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 3-1 shows a sample display that is receiving input from a user.

Figure 3-1 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>
```

Note The 6900 series IP phones does not display the *Title* and *Prompt* menu fields at the same time. If both *Title* and *Prompt* fields are defined at the same time, then these phones display only the *Prompt* field.

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 Unified 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 Unified IP Phones, but it is available for your development purposes also. Figure 3-2 shows how an XML `CiscoIPPhoneDirectory` object displays on the phone.

Figure 3-2 CiscoIPPhoneDirectory Object Display Sample

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 Unified 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.

Note The 6900 series IP phones does not display the *Title* and *Prompt* menu fields at the same time. If both *Title* and *Prompt* fields are defined at the same time, then these phones display only the *Prompt* field.

Custom Directories

You can use the Cisco Unified Communications Manager 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 Unified 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 softkey</Name>
    <URL>URL of softkey</URL>
    <Position>Numerical position of the softkey</Position>
   </SoftKeyItem>
</CiscoIPPhoneImage>
```

Note The 6900 series IP phones does not display the *Title* and *Prompt* menu fields at the same time. If both *Title* and *Prompt* fields are defined at the same time, then these phones display only the *Prompt* field.

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.

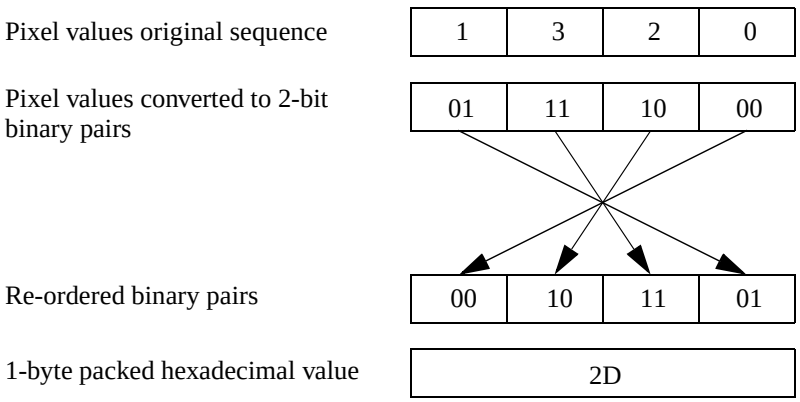
Use `Width` and `Height` to size the graphic. If the values do not match with the pixel stream specified in the `Data` field, results will be unpredictable incorrect.

`Depth` specifies the number of bits per pixel. Cisco Unified 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 Unified 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 3-3 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 3-3 Packed Pixel Translation Example



Example

The following XML code defines a `CiscoIPPhoneImage` object that displays the sequence of pixels shown in Figure 3-3 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 Unified 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 Unified IP Phones have higher-resolution displays with more color depth. The Cisco Unified IP Phone 7970G, 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 3-2).

Table 3-2 Cisco Unified IP Phones Display Image Sizes and Color Depths

Model	Resolution[1] (width x height)	Color/Grayscale /Monochrome	Color Depth (bits)
Cisco Unified IP Phones 7905G, 7906G, 7911G, 7912G[2], 7931G	N/A	Grayscale	1
Cisco Unified IP Phone 7920	128 x 59	Grayscale	1
Cisco Unified IP Phone 7921G, 7925G	176 x 140	Color	16
Cisco Unified IP Phones 7940G/60G	133 x 65	Grayscale	2
Cisco Unified IP Phones 7941G, 7941G-GE, 7942G, 7961G, 7961G-GE, 7962G	298 x 144	Grayscale	4

Table 3-2 Cisco Unified IP Phones Display Image Sizes and Color Depths (continued)

Model	Resolution[1] (width x height)	Color/Grayscale /Monochrome	Color Depth (bits)
Cisco Unified IP Phones 7945G, 7965G	298 x 156	Color	16
Cisco Unified IP Phone 7970G/7971G	298 x 168	Color	12
Cisco Unified IP Phone 7975G	298 x 168	Color	16
Cisco IP Communicator	298 x 168	Color	24
Cisco Unified IP Phones 6921, 6961	396 x 81	Monochrome	—
Cisco Unified IP Phone 6941	396 x162	Monochrome	—
Cisco Unified IP Phones 9971, 9951, 8961	498x289	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 Unified 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 Unified IP Phone 7970G.

Figure 3-4 shows a CiscoIPPhoneImageFile object on a Cisco Unified IP Phone 7970G display.

Figure 3-4 Cisco Unified IP Phone 7970G 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>
```

Note The 6900 series IP phones does not display the *Title* and *Prompt* menu fields at the same time. If both *Title* and *Prompt* fields are defined at the same time, then these phones display only the *Prompt* field.

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. When using non-ASCII character sets, the system presents the information as a bitmap graphic. To select a menu, the user enters a number from 1 to 12 using the numeric keypad (* and # are not active).

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>
```

Note The 6900 series IP phones does not display the *Title* and *Prompt* menu fields at the same time. If both *Title* and *Prompt* fields are defined at the same time, then these phones display only the *Prompt* field.

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 Cisco Unified IP Phone models, such as the Cisco Unified IP Phone 7970G and Cisco IP Communicator, have pointer devices. The Cisco Unified IP Phone 7970G 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 3-5 shows the display of the CiscoIPPhoneGraphicFileMenu.

Figure 3-5 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 3-2, "Cisco Unified 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>
```

Note The 6900 series IP phones does not display the *Title* and *Prompt* menu fields at the same time. If both *Title* and *Prompt* fields are defined at the same time, then these phones display only the *Prompt* field.

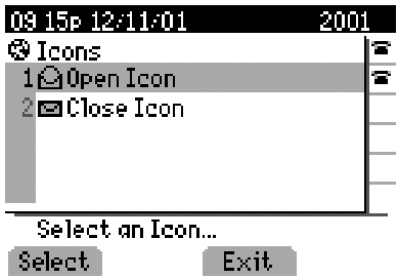
CiscoIPPhoneIconMenu

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 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 3-6 shows an IconMenu on a Cisco Unified IP Phone.

Figure 3-6 IconMenu on a Cisco Unified 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 softkey</Name>
    <URL>URL or URI of softkey</URL>
    <Position>Position information of the softkey</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>
```

Note The 6900 series IP phones does not display the *Title* and *Prompt* menu fields at the same time. If both *Title* and *Prompt* fields are defined at the same time, then these phones display only the *Prompt* field.

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 3-7).

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 3-7 CiscoIPPhoneIconFileMenu Object Display Sample

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>
```

Note The 6900 series IP phones does not display the *Title* and *Prompt* menu fields at the same time. If both *Title* and *Prompt* fields are defined at the same time, then these phones display only the *Prompt* field.

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.

The Status object cannot be closed or cleared by the user (for example, by pressing Services) because the Status object is only present on the Call plane. 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.

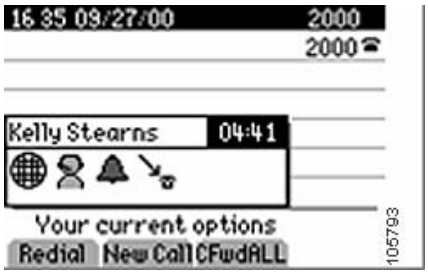
Note The CiscoIPPhoneStatus object can only be pushed (HTTP POST) to the phone; it cannot be pulled (HTTP GET).

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 3-8 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 3-8 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>
```

Note The 6900 series IP phones does not display the *Title* and *Prompt* menu fields at the same time. If both *Title* and *Prompt* fields are defined at the same time, then these phones display only the *Prompt* field.

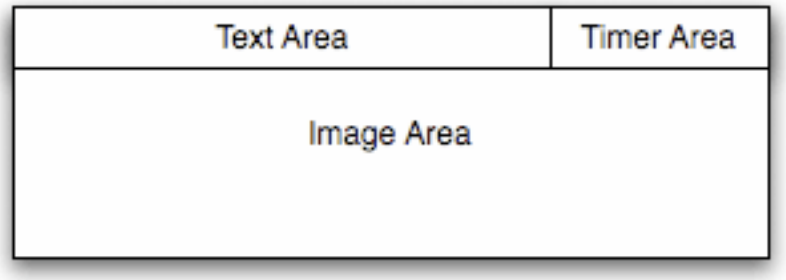
Dynamic Sizing of the Application Status Window

You can enable applications to dynamically adjust their window sizes based on the displayed content. The minimum size requirements limit the windows size so that it is a large enough size to stand out from the Overview content. For example, using a smaller window for an application allows more content from the Overview to be displayed. Sizing the window occurs upon the reception of a CiscoIPPhoneStatus or CiscoIPPhoneStatusFile object with its associated PNG file.

The Application Status window contains three main areas: (see Figure 3-9):

- Text Area
- Timer Area
- Image Area

Figure 3-9 Elements of Application Status Window



Note Self-terminating XML elements, non-declared or missing elements, and elements with the default values are all considered non-configured elements.

To allow dynamic sizing, do not configure the Text and Timer areas with any value other than the default used by the XML parser. If both elements are not configured, you can proceed, but must follow these rules:

- Do not display the Text Area and Timer Area sections of the Application Status window.
- If the LocationX element is not configured or is set to centered, and the image provided is less than the maximum width allowed, the Image Area can be resized.
- If the image provided is smaller than the minimum width, the minimum allowed window width should be used.
- If the width of the image provided is between the minimum and maximum sizes of the window, the window should be sized to display the image as well as the standard surrounding borders.
- The image height should never change.

See Table 3-3 for an overview of the maximum and minimum image area sizes by phone model. Most phone models support all sizes between the minimum and maximum. An exception is allowed for the Cisco Unified IP Phones 7940G/7960G due to resource constraints. For these phones, you should implement both the maximum size and minimum size windows ignoring all of the intermediate sizes.

Table 3-3 Application Status Window Allowable Image Sizes

Phone Models	Maximum Image Area Width	Minimum Image Area Width	Maximum Image Area Height
7940G, 7960G	106	21	21
7941G/7941G-GE, 7942G, 7945G, 7961G/7961G-GE, 7962G, 7965G	252	50	50
7970G/7971G-GE, 7975G, IP Communicator	262	50	50

Note The Cisco IP Phone models 9951, 9971, and 8961 does not support CiscoIPPhoneStatus object. See Table 3-1 on page 3-2 for information on XML objects supported on various phone models.

See Table 3-4 for an overview of the text and timer area sizes by phone model.

Table 3-4 Application Status Window Allowable Text and Timer Sizes

Phone Models	Text Area Size (WxH)	Timer Area Size (WxH)	Text Area Size No Timer (WxH)
7940G, 7960G	76x11	30x11	106x11
7941G/7941G-GE, 7942G, 7945G, 7961G/7961G-GE, 7962G, 7965G,	192x20	60x20	252x20
7970G / 7971G-GE, 7975G, IP Communicator	202x20	60x20	262x20

CiscoIPPhoneStatusFile

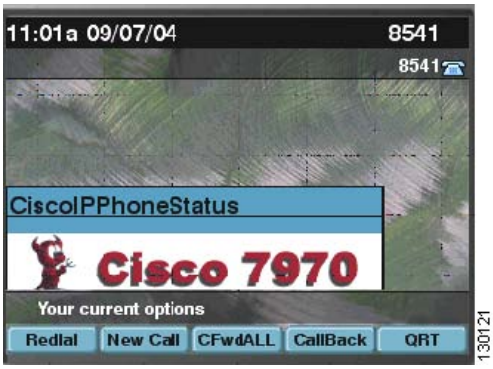
The behavior of this object is identical to the `CiscoIPPhoneStatus` object, except it uses a color PNG image instead of a grayscale CIP image for the graphics area.

The maximum image size is 262 x 50 pixels for the Cisco Unified IP Phone 7970G, but differs for other phone models. See "Dynamic Sizing of the Application Status Window" section on page 3-15 for details.

Note The Cisco IP Phone models 9951, 9971, and 8961 does not support CiscoIPPhoneStatusFile object. See Table 3-1 on page 3-2 for information on XML objects supported on various phone models.

Figure 3-10 shows how an XML `CiscoIPPhoneStatusFile` object displays on a phone.

Figure 3-10 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 Unified Communications Manager Administration Enterprise Parameters and can be overridden on a per phone basis in the Cisco Unified Communications Manager 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, then 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 Unified 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.

Note The <URLDown> tag can only contain Internal URIs—it cannot contain an HTTP URL. The "URL" in the name "URLDown" does not signify that an HTTP URL can be used.

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. To retain default softkey behavior, then you must explicitly define it 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 Chapter 5, "Internal URI Features" for more information on invoking internal softkey features.

Note If there are no custom softkeys and there is no default softkey placed in position 1, either a **Next** or **Update** softkey is assigned automatically. If the URL is a Refresh URL, the softkey will be "Next." If not, the **Update** softkey is assigned.

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 Unified 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 3-5 lists characters and their escape values.

Table 3-5 Escape Sequences for Special Characters

Character	Name	Escape Sequence
&	Ampersand	&
“	Quote	"
'	Apostrophe	'
<	Left angle bracket	<
>	Right angle bracket	>

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

XML Encoding

Because the phone firmware can support multiple encodings, the XML encoding should always be set in the XML header.

If the XML encoding header is not specified, the phone will default to the encoding specified by the current user locale.

Note This behavior is NOT compliant with XML standards, which specify UTF-8 as the default encoding, so any UTF-8 encoded XML object must have the encoding explicitly set for the phone to parse it correctly.

The encoding value specified in the XML header must match one of the encodings provided by the IP Phone in its Accept-Charset HTTP request header, as shown in the example below.

Example

The following examples illustrate UTF-8 and ISO-8859-1 encoding, respectively:

```
<?xml version="1.0" encoding="utf-8" ?>
<?xml version="1.0" encoding="iso-8859-1" ?>
```

For details on setting HTTP header encoding settings, see the “HTTP Encoding Header Setting” section on page 6-6.

Application Event Handlers

The Application Manager API (see "Application" section on page 5-18) includes an Application Management Event Handler which is supported by any displayable object, which are noted in the following table. The unsupported objects are not contained in a standard application context and are handled differently by the Application Manager API:

Supported	Unsupported
CiscoIPPhoneMenu	CiscoIPPhoneStatus
CiscoIPPhoneText	CiscoIPPhoneStatusFile
CiscoIPPhoneInput	
CiscoIPPhoneDirectory	
CiscoIPPhoneImage	
CiscoIPPhoneImageFile	
CiscoIPPhoneGraphicMenu	
CiscoIPPhoneGraphicFileMenu	
CiscoIPPhoneIconMenu	
CiscoIPPhoneIconFileMenu	

Note Support for the Application Event Handlers requires an updated XML Parser (see "Updated XML Parser and Schema Enforcement" section on page B-1 for details).

Attributes

The Application Event Handlers can be attached to a supported object by specifying the attributes:

Note An Application URI with Priority=0 is not allowed in the Application Event Handlers (see "Application" section on page 5-18).

Attribute	Description
appID	Identifies the application to which this displayable XSI object belongs. The format of the appID attribute should be in the format *vendor/product*, such as `Cisco/Unity`, but this syntax is not enforced, and the application can assign any unique identifier.
onAppFocusLost	Invoked when the application loses focus, if: • The application's context has lost focus, or • The application was navigated away from, either directly by the user, or programmatically by a refresh header or HTTP push. **Note** If a Notify URI is used as the event handler, a notification is sent with this default data: `<notifyApplicationEvent appId="appId" type="focusLost"/>`
onAppFocusGained	Invoked when the application gains focus, if: • The application is Active and the application's context has gained focus, or • The application was navigated to, either directly by the user, or by a refresh header or HTTP push. **Note** If a Notify URI is used as the event handler, a notification is sent with this default data: `<notifyApplicationEvent appId="appId" type="focusGained"/>`
onAppMinimized	Invoked when the application is minimized. An application can only be minimized programmatically by a call to App:Minimize, but this invocation could occur by direct action of the user (from a softkey invocation, for example) or from the application via a push request. `<notifyApplicationEvent appId="appId" type="minimized"/>`
onAppClosed	Invoked whenever the application closes, if: • The application's context is closed which will, in turn, close all applications in its stack, or • The application no longer exists on the context's URL stack because it was navigated out of, or because it was pruned from the URL stack (stack size exceeded). **Note** This event handler cannot contain HTTP or HTTPS URLs. **Note** If a Notify URI is used as the event handler, a notification is sent with this default data: `<notifyApplicationEvent appId="appId" type="closed"/>`

Event Handler Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="notifyApplicationEvent">
    <xs:complexType>
      <xs:attribute name="appId" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:minLength value="1"/>
            <xs:maxLength value="64"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="type" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="closed"/>
            <xs:enumeration value="minimized"/>
            <xs:enumeration value="focusLost"/>
            <xs:enumeration value="focusGained"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example

```
<CiscoIPPhoneImage appId="Cisco/Unity"
  onAppFocusLost="RTPRx:Stop; RTPTx:Stop; Notify:http:server:80:path"
  onAppFocusGained="http://server/mainpage/updateUI"
  onAppClosed="Notify:http:server:80:eventlistener/appClosed">
  ...
</CiscoIPPhoneImage>
```

CHAPTER 4

Component APIs

In addition to the primary phone XSI API, two additional component APIs are available:

- Application Management API, page 4-1
- RTP Streaming API, page 4-1

Application Management API

To address the limited application management, the Application Management API provides a smoother hand-off between the call mode and the application mode. The Application API consists of two primary components:

- Application URI—see the "Application" section on page 5-18
- Application Event Handlers—see the "Application Event Handlers" section on page 3-22

Note Support for the Application Management API requires an updated XML Parser (see "Updated XML Parser and Schema Enforcement" section on page B-1 for details).

RTP Streaming API

This XML-based RTP Streaming API allows applications to initiate and observe RTP audio streams. It extends capabilities beyond the legacy RTP streaming URIs by providing support for stream start/stop event listeners and the ability to specify other extended stream attributes, such as codec type.

Note Support for the RTP Streaming API requires an updated XML Parser (see "Updated XML Parser and Schema Enforcement" section on page B-1 for details).

The event handlers typically use the standard Notification framework (see "Notify" section on page 5-16), but they can also invoke most other URIs, with the exception of HTTP URLs.

Interaction Rules with Legacy RTP URI Streams

The RTP Streaming API allows a full-duplex stream (mode=sendReceive) to be setup as a single stream request which simplifies the usage of the API. However, in some cases, this creates some interoperability issues with the legacy RTP URIs because the legacy RTP URIs send and receive streams separately. The interaction rules between legacy RTP URI streams and the new RTP Streaming API are as follows:

- If an RTP Stop URI is invoked, and an RTP Streaming API stream is currently streaming in that same direction, then the entire RTP Streaming API stream is stopped.

  For example, if a full-duplex stream is setup through the RTP Streaming API (mode=sendReceive) and then an RTPTx:Stop URI is invoked, the stream will be stopped in both the send and receive directions (and the onStopped event handler will be called, if present).

- If the **stopMedia** request (from the RTP Streaming API) does not specify a stream ID, then the request will stop all services RTP streams, in any direction (send or receive) and of any type (multicast and unicast). This allows applications using the RTP Streaming API to stop media streams which may have been started by the legacy RTP URIs or by other applications for which a stream ID is not known.

RTP Streaming Schema

Note The **port** number parameter of the `startMedia` request is optional and if it is not specified, the phone selects an available port and returns it in the `startMediaResponse` object. The **port** parameter, if specified, must be an even number in the range of 20480-32768.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Cisco Systems, Inc. (Cisco
Systems, Inc.) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="startMedia">
    <xs:complexType>
      <xs:all>
        <xs:element name="mediaStream" type="mediaStream"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
  <xs:element name="stopMedia">
    <xs:complexType>
      <xs:all>
        <xs:element name="mediaStream">
          <xs:complexType>
            <xs:attribute name="id" type="xs:string" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
  <xs:element name="startMediaResponse">
    <xs:complexType>
      <xs:all>
        <xs:element name="mediaStream" type="mediaStream"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
  <xs:element name="notifyMediaEvent">
```

```
  <xs:complexType>
    <xs:all>
      <xs:element name="mediaStream">
        <xs:complexType>
          <xs:attribute name="id" type="xs:string" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:all>
    <xs:attribute name="origin" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="user"/>
          <xs:enumeration value="application"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="stopped"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
<xs:complexType name="mediaStream">
  <xs:all>
    <xs:element name="type">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="audio"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="codec">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="G.711"/>
          <xs:enumeration value="G.722"/>
          <xs:enumeration value="G.723"/>
          <xs:enumeration value="G.728"/>
          <xs:enumeration value="G.729"/>
          <xs:enumeration value="GSM"/>
          <xs:enumeration value="Wideband"/>
          <xs:enumeration value="iLBC"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="mode">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="send"/>
          <xs:enumeration value="receive"/>
          <xs:enumeration value="sendReceive"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="address">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:minLength value="7"/>
          <xs:maxLength value="15"/>
        </xs:restriction>
      </xs:simpleType>
```

```
      </xs:element>
      <xs:element name="port" minOccurs="0">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedShort">
            <xs:minInclusive value="20480"/>
            <xs:maxInclusive value="32768"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:all>
    <xs:attribute name="onStopped" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:minLength value="1"/>
          <xs:maxLength value="256"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="receiveVolume" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:integer">
          <xs:minInclusive value="0"/>
          <xs:maxInclusive value="100"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:schema>
```

Error Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="errorResponse">
    <xs:complexType>
      <xs:all>
        <xs:element name="type">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="InvalidURL"/>
              <xs:enumeration value="InvalidResource"/>
              <xs:enumeration value="InvalidResourceID"/>
              <xs:enumeration value="UnavailableResource"/>
              <xs:enumeration value="InvalidXML"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="data" nillable="true">
          <xs:simpleType>
            <xs:restriction base="xs:string"/>
          </xs:simpleType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Examples

Start Media

- Request

```
HTTP POST /CGI/Execute
<startMedia>
  <mediaStream
      onStopped="Notify:http:server:80:path/page"
      receiveVolume="50">
        <type>audio</type>
        <codec>G.729</codec>
        <mode>sendReceive</mode>
        <address>239.1.2.3</address>
        <port>20480</port>
  </mediaStream>
</startMedia>
```

- Response

```
HTTP200 OK
<mediaStream id="abc123"/>
```

Stop Media

- Request

```
HTTP POST CGI/Execute
<stopMedia>
  <mediaStream id="abc123"/>
</stopMedia>
```

- Response

```
HTTP 200 OK
```

If the user terminates the media stream by placing the active audio path on-hook, the following notification is sent:

```
HTTP POST /server/path/page
DATA=<notifyMediaEvent type="stopped" origin="user">
    <mediaStream id="abc123"/>
</notifyMediaEvent>
```

Errors and Responses

Error conditions and responses for the RTP Streaming API include:

Condition	Applicable Methods	HTTP Result Code	Type	Data
Authorization failed	all	401 (Authorization Failed)	N/A	N/A

Condition	Applicable Methods	HTTP Result Code	Type	Data
Request object does not comply with the API's XML schema	all	400 (BadRequest)	InfalidXML	<parser error description>
Media cannot be started because no DSP resources is available to handle the media	startMedia	400 (BadRequest)	Unavailable Resource	No Media Resource Available
Media cannot be stopped because the specified stream ID does not exist	stopMedia	400 (BadRequest)	InvalidResourceID	Unknown Media Stream ID: <streamID>

CHAPTER 5

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
- Device Control URIs
- XML Displayable Object URIs
- Multimedia URIs
- Telephony URIs
- Application Management URIs

Supported URIs by Phone Model

Table 5-1 lists the URIs that are supported for Release 7.1(3).

Table 5-1 URIs Supported for Release Cisco Unified IP Phone Services SDK

URI	7905G 7912G	7906G 7911G 7931G	7920G	7921G 7925G	7940G 7960G	7941G/7941G-GE, 7961G/7961G-GE, 7942G, 7962G, 7945G, 7965G, IP Communicator	7970G 7971G-GE 7975G	6921, 6941, 6961	9971, 9951, 8961
Key	X	X	X	X	X	X	X	X	X[1]
Softkey	X	X	X	X	X	X	X	X	X
Init	X	X	X	X	X	X	X	X	X
Dial, EditDial	X	X	X	X	X	X	X	X	X
Play	X	X	X	X	X	X	X	X	X
QueryStringParam	X	X	X	X	X	X	X	X	X
Unicast RTP	X	X	X[2]	X	X	X	X	X	X
Multicast RTP	X	X	X	X	X	X	X	X	X
Display	—	—	—	—	—	—	X	—	X

Table 5-1 URIs Supported for Release Cisco Unified IP Phone Services SDK (continued)

URI	7905G 7912G	7906G 7911G 7931G	7920G	7921G 7925G	7940G 7960G	7941G/7941G-GE, 7961G/7961G-GE, 7942G, 7962G, 7945G, 7965G, IP Communicator	7970G 7971G-GE 7975G	6921, 6941, 6961	9971, 9951, 8961
Vibrate	—	—	X	X	—	—	—	—	—
Notify[3]	—	X	—	X	—	X	X	—	X
SendDigits[3]	—	X	—	X	—	X	X	—	X
Application[3]	—	X	—	X	—	X	X	—	X

1. Key:Info, Key:Services, Key:Directories, Key:Settings, Key:AppMenu, and Key:Hold are not supported by 9971, 9951, and 8961 models.
2. Only supports one incoming and one outgoing unicast stream and does not support the Volume parameter for RTP Receive streams.
3. Requires Cisco Unified IP Phone firmware version 8.3(2) or later, which contains an updated XML parser. See the "Updated XML Parser and Schema Enforcement" section on page B-1.

Device Control URIs

These sections describe the device control URIs:

- Key
- Display

Key

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 present on the phone (hard button) or not available (softkey) when the URI is processed, the event is discarded.

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

URI Format

```
Key:n
```

Where

n = a Key name.

Table 5-2 lists the Key URIs and the phone models in which these are supported.

Table 5-2 Key URIs With Supported Phone Models

Key URIs	7905G 7912G	7906G 7911G 7931G	7920G	7921G 7925G	7940G 7960G	7941G/7941G-GE, 7961G/7961G-GE, 7942G, 7962G, 7945G, 7965G, IP Communicator	7970G 7971G-GE 7975G	6921, 6941, 6961	9971, 9951, 8961
Key:Line1 to Key:Line120	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:KeyPad0 to Key:KeyPad9	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:Soft1 to Key:Soft5	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:KeyPadStar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:KeyPadPound	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:VolDwn	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:VolUp	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:Headset	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:Speaker	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:Mute	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:NavLeft	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:NavRight	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:NavSelect	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:Info	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Key:Messages	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:Services	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Key:Directories	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Key:Settings	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Key:NavUp	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:NavDwn	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:AppMenu	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Key:Hold	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Key:Feature1 to Key:Feature120	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:Session1 to Key:Session6	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:Applications	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:Contacts	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 5-2 Key URIs With Supported Phone Models

Key URIs	7905G 7912G	7906G 7911G 7931G	7920G	7921G 7925G	7940G 7960G	7941G/7941G-GE, 7961G/7961G-GE, 7942G, 7962G, 7945G, 7965G, IP Communicator	7970G 7971G-GE 7975G	6921, 6941, 6961	9971, 9951, 8961
Key:FixedFeature 1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:FixedFeature 2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:FixedFeature 3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:NavBack	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Key:Release	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Unsupported Key URIs and Alternate Options

This section describes the unsupported Key URIs in the phone models and provides alternative options, if any, for the unsupported URIs.

Table 5-3 Unsupported Key URIs and Alternative Options

Phone Models	Unsupported URI	Description and Alternatives
9971, 9951, 8961	Key:Services	The 8900 and 9900 series IP phones does not have a services button. As such, the “Key:Services” URI is not supported in these phones. The application has to use the “Init:Services” URI and the “App:Close” URI to close the last XSI application launched from the application. If there is no application open, then the request has no effect. Additionally, the Exit softkey takes the application to the previous screen, and if the application is at the top level, it closes the application.
9971, 9951, 8961	Key:Info	The 8900 and 9900 series IP phones does not have a standalone help application. Help is provided within the context of each application.
9971, 9951, 8961	Key:Directories	In the 8900 and 9900 series IP phones the Key:Contacts URI replaces the Key:Directories URI. You can use Key:Contacts to invoke the new contacts application in these phones.
9971, 9951, 8961	Key:Settings	The 8900 and 9900 series IP phones does not have a single monolithic settings application. As such the Key:Settings URI is not supported in these phones.

Table 5-3 Unsupported Key URIs and Alternative Options

Phone Models	Unsupported URI	Description and Alternatives
9971, 9951, 8961	Key:AppMenu	The 8900 and 9900 series IP phones does not support the Key:AppMenu URI. All applications are accessed via their individual Key URIs like Applications, Contacts, and Messages.
9971, 9951, 8961	Key:Hold	The 8900 and 9900 series IP phones does not support the Key:Hold URI. The Key URI equivalents for invoking the standard fixed features are as follows: • To invoke transfer use Key:FixedFeature1 • To invoke conference use Key:FixedFeature2 • To invoke hold use Key:FixedFeature3

Display

The Display URI is available only on those Cisco Unified IP Phones that have a color backlight on the phone display, including the Cisco Unified IP Phone 7970G and 7971G-GE. Using the Display URI, you can control how long the backlight remains on or off.

Note, however, that other administrator-controlled or user-indicated display settings take precedence over the Display URI. As such, various phone states (such as phone startup, incoming and active calls, or other user input states) override the Display URI settings.

URI Format

```
Display:State:Interval
```

Where

State = whether the phone display is turned on or off or set to default to return the display to its specified state.

Interval = duration (in minutes) in which the phone state remains in the specified state (unless activated by automated or user input). Value must be an integer ranging from 0-1440 minutes. If the value is set to 0, the display remains in the indicated state indefinitely (unless activated by automated or user input).

For example:

- `Display:Off:60` turns the phone display off for 1 hour (60 minutes).
- `Display:On:10` turns the phone display on for 10 minutes.
- `Display:Off:0` turns off the display off until activated.
- `Display:Default` returns the display to its specified state for that time.

XML Displayable Object URIs

These sections describe the XML displayable object URIs:

- SoftKey
- QueryStringParam

SoftKey

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.

Note The Softkey URI is not supported in the Execute object.

URI Format

```
SoftKey:n
```

Where

n = one of the following softkey names:

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

Table 5-4 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 5-4 Valid Softkey Actions for CiscoIPPhoneObject Types

IPPhoneObject[1]	Select	Exit	Update	Submit	Search	<<	Cancel	Next	Dial	Edit Dial
CiscoIPPhoneMenu	X	X								
CiscoIPPhoneIconMenu	X	X								
CiscoIPPhoneText		X	X							

Table 5-4 Valid Softkey Actions for CiscoIPPhoneObject Types (continued)

IPPhoneObject[1]	Select	Exit	Update	Submit	Search	<<	Cancel	Next	Dial	Edit Dial
CiscoIPPhoneImage		X	X							
CiscoIPPhoneGraphicMenu		X	X							
CiscoIPPhoneInput				X	X[2]	X	X			
CiscoIPPhoneDirectory							X	X	X[3]	X[3]

1. The SoftKey URI is not allowed in an Execute object.
2. Only when used under the Directories button.
3. 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 "Telephony URIs" section on page 5-13.

The new generation 8900 / 9900 series IP phones have the following enhancements to their display:

- The positions of the sofkeys have been changed. Moving from left to right, the Exit is the first softkey followed by the Submit/Select/Update/Next softkey, and finally the Delete softkey.
- In the submenu screens, the back arrow icon ( ) replaces the << or Exit softkeys, and it is placed in the first (extreme left) position.
- The phone displays error messages, like XML Parse error or HTTP failures, in a new window.

QueryStringParam

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 5-1 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 5-1 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 5-2 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 Unified IP Phones allow you to implement the QueryStringParam URI in either manner although Example 5-2 is not as efficient as Example 5-1. Choose the best way to perform the action based on your applications needs.

Example 5-2 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.

Multimedia URIs

These sections describe the multimedia URIs:

- RTP Streaming
- Play
- Vibrate

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

Note For some Cisco Unified IP Phone models, the RTP Streaming URIs have been deprecated by the RTP Streaming API. See the "RTP Streaming API" section on page 4-1.

The supported format of the RTP stream is as follows:

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

The following list gives these possible CiscoIPPhoneError codes:

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

Interaction with Call Streaming

- Existing Tx URI streams will be terminated if a new call begins or an existing call is resumed
- Tx URI stream requests received when a call is active will be rejected with an `errorNo=4 unauthorized`. If a call is in a Held state (connected but not actively streaming), the Tx URI request will be accepted, but will be terminated if the call is resumed.

Note Returning `errorNo=4` allows the application to distinguish this error from the normal `errorNo=1 busy` response.

- Existing Rx URI streams will be terminated if a new call begins or an existing call is resumed.

The user has no explicit mechanism for terminating the Rx URI stream independent of the call. Thus, if the Rx stream is not terminated automatically, it would continue to play. For example, a user is listening to Internet radio feed and gets an incoming call. The user answers the call, which either closes or minimizes the Internet radio XSI application. Otherwise, the user has no intuitive way to stop the music stream.

- New Rx URI stream requests received during an active call will be accepted (whisper), but the volume parameter of the URI will be ignored.

  If the Rx URI request was done via push, then the associated application is responsible for using push Priority attributes and for stopping and starting the stream.

  If the user initiates the Rx URI via an application, then the user likely is not concerned about having the audio mixed with the current call. However, they should also be presented with an option to stop the application, when needed.

- For the Rx URI, the Mute indicator light is only lit when both these conditions are met:
  - There are no active transmit streams from either a call or an XML services stream, and
  - There is at least one active receive stream

  For example, if an active call is ended or put on hold while a Rx URI stream is active, the Mute indicator will light.

- If a Rx or Tx URI request is received and there is already an active XML services stream in that direction, then a response with `errorNo=1 Tx/Rx is already active` will be returned. The previous stream must be terminated (either by the user or by an RTP Stop URI) before a new stream can be started.

  This response provides visibility to the application if the phone is currently busy. It then allows the application to decide whether or not to terminate the existing stream and start a new one, rather than being controlled by the phone firmware.

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 = 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 = 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 transmitted.

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.

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 the phone is equipped with an MWI light, it will be flashing while the audio file is playing, providing a visual alert as well.

Note The Play URI is a synchronous request. If the request is pushed to the phone via HTTP, the HTTP response (CiscoIPPhoneResponse object) is not returned until after the playback has completed.

Interaction with Incoming Calls

The Play URI and incoming calls (ringing) have equal priority access to the DSP ringer resources resulting in the following interactions:

- If a Play URI is currently playing, an incoming call (ringing) will not preempt the Play URI; the Play URI will finish playing first.
- If the phone is ringing and a Play URI request is sent to the phone, the execution of the Play URI defers until the phone stops ringing (the DSP ringer resource becomes available) and then the Play URI will play.

URI Format

```
Play:f
```

Where

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

The audio files for the rings must meet the following requirements for proper playback on Cisco Unified IP Phones:

- Raw PCM (no header)
- 8000 samples per second
- 8 bits per sample
- uLaw compression
- Maximum ring size—16080 samples
- Minimum ring size—240 samples
- Number of samples in the ring is evenly divisible by 240.
- Ring starts and ends at the zero crossing.

To create PCM files for custom phone rings, you can use any standard audio editing packages that support these file format requirements.

Vibrate

The Vibrate URI is available on the Cisco Unified IP Phones 7920G, 7921G, and 7925G wireless phone models, and it enables third-party applications to invoke the phone's vibration capabilities for silent alerts, similar to the way in which the Play URI plays audible alerts. If the Vibrate parameters are not specified or if the device is unable to support custom Vibrate sequences, the device will execute its default vibrate sequence.

URI Format

```
Vibrate:vibrateDuration:silenceDuration:count
```

Where

vibrateDuration = duration (in milliseconds) in which the vibrate state remains on. Value must be an integer ranging from 0-65536 milliseconds.

silenceDuration = duration (in milliseconds) in which the vibrate state remains off. Value must be an integer ranging from 0-65536 milliseconds.

count = number of times to repeat the vibrate on and off sequence.

For example:

- `Vibrate:1000:0:1` initiates a single vibrate for 1 second.
- `Vibrate:500:1500:5`initiates five vibrations each lasting for 500 ms. followed by 1500 ms of silence.

Telephony URIs

These sections describe the telephony URIs:

- Dial
- EditDial
- SendDigits

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:{dialSequence}[:{useAppUI}:{applicationId}[:audibleFeedback]]
```

Where

dialSequence = The sequence of DTMF digits to be dialed. Commas represent 1 second pauses.

Value Type: String

Values: minLength=0, no maxLength, can only contain 0123456789#*ABCD,

Default value: N/A

useAppUI = Specifies whether or not this application will be used as the user interface for this call. A value of true will cause the application to keep UI focus when the call is made instead of switching to the Call UI application. The appId must be specified or this param will have no effect – it will always be false.

Value Type: boolean

Values: 0 or 1 (0=false 1=true)

Default value: 0

applicationId = The unique name of the XSI web application requesting this call

Value Type: String

Values: minLength=1, no maxLength, cannot contain semicolons – should be in the format Company/Product.

Default value: Nil which means this dial request will not be associated with any application

audibleFeedback = Whether or not to provide audible feedback to the user when the DTMF digits are dialed.

Value Type: Boolean

Values: 0, 1 (0=false 1=true)

Default value: 1

EditDial

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

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**).

SendDigits

The SendDigits URI instructs the phone to send a specified sequence of DTMF digits in-band within the media stream of the current active (streaming) call.

Audible feedback to the user can be enabled or disabled and an optional application ID can be specified to ensure that the DTMF digits will only be sent to the call which is associated with a specific application.

URI Format

```
SendDigits:dtmfSequence:audibleFeedback::applicationId
```

Where

dtmfSequence = the sequence of DTMF digits to be sent. Value must contain only 0123456789#*ABCD

audibleFeedback = indicates whether to provide audible feedback to the user as the DTMF digits are entered. Values can be 0 (false) or 1 (true).

applicationId = optional identifier of the application associated with the call which must receive the DTMF digits. Value must be 0-64 and cannot contain colons. The default value is null indicating that the active call should receive the DTMF digits, regardless of any application association.

For example:

- Make a call using a calling card service that implements these steps:
  1. Connects to a 800 calling card service (using the Dial URI)
  2. Application waits to give call time to connect
  3. Dials the destination number, ensuring that the digits can only be dialed from this application.
  4. Pauses 2 seconds
  5. Dials the calling card number
  6. Pauses 1 second
  7. Dials the pin number

```
<CiscoIPPhoneExecute>
  <ExecuteItem URL="Dial:918005551212:1:Cisco/Dialer"/>
</CiscoIPPhoneExecute>
<CiscoIPPhoneExecute>
  <ExecuteItem URL="SendDigits:6185551212,,987654321,1234:1:Cisco/Dialer"/>
</CiscoIPPhoneExecute>
```

Error and Response

When the SendDigits URI is invoked via an Execute object, it will use the standard URI Status and Data values in ResponseItems:

Condition	Status	Data
Executed successfully	0 (Success)	Success
URI syntax is invalid	1 (Parse error)	Invalid URI
URI is not supported	6 (Internal error)	URI not found
Unable to execute URI because there currently is no active (streaming) call	6 (Internal error)	No Active Call
Unable to execute URI because the current active (streaming) call is not associated with the specified application	6 (Internal error)	No Active Call for Application

Condition	Status	Data
Phone is temporarily unable to execute URI due to some other transient issue	6 (Internal error)	<Failure>

Application Management URIs

These sections describe the application management URIs:

- Init
- Notify
- Application

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.

`Services`—When the phone encounters an Init:SERVICES URI, it closes the Services application. If Services is not currently open, it has no effect.

`Messages`—When the phone encounters an Init:Messages URI, it closes the Messages application. If Messages is not currently open, it has no effect.

`Directories`—When the phone encounters an Init:Directories URI, it closes the Directories application. If Directories is not currently open, it has no effect.

Notify

The Notify URI generates network notifications to back-end applications. This feature is most useful for XSI objects that support action handlers (such as displayable XSI objects and RTP streams). For example, use the Notify URI to deliver notifications to back-end applications when an XSI application is closed or when an RTP stream is terminated.

You can also specify the Notify URI in place of most fields that accept a generic URI, including softkeys and menu items. For example, you can call the Notify URI from a softkey or menu item to trigger a back-end event that does not require an interface change, such as manipulating the state of audio streams or other non-visual resources. The Notify URI also works in conjunction with the QueryStringParam URI, such that the exact contents of the QueryStringParam data will be used as the Notify URI data.

The Notify URI is not made in the context of an XSI application session and does not contain any HTTP cookie or session information. Thus, the back-end application cannot rely on HTTP cookies or session information to uniquely identify the client or application. Instead, the application must embed any necessary information in the Notify path and data fields, or leave the data field empty and rely on any default information provided by the specific event handler.

Note The Notify URI is not supported in the Execute object.

URI Format

```
Notify:protocol:host:port:path:credentials:data
```

Where

protocol = network protocol to use for the Notify connection; http is the only supported protocol.

host = network host designated to receive the notification. Value must be entered as a hostname or IP address.

port = network port to use for the Notify connection. Value must be a number from 1-65535.

path = protocol-specific information. Value cannot contain colons or semicolons.

credentials = optional protocol-specific credentials used to authenticate to the server. For HTTP, this is a base64-encoded version of `userid:password`. Value cannot contain colors or semicolons. If the credentials parameter is not specified or if it is null, no Authorization header will be included in the request. The HTTP notification service will retry the request 3 times before failing and logging an error message.

data = optional application-specific event data. Value cannot contain semicolons.

For example:

- Called from RTP onStreamStopped Event Handler, no credentials, with data:

```
Notify:http:myserver:8080:path/streamhandler?event=stopped:
:myStreamStoppedData

HTTP POST /path/streamhandler?event=stopped HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=”UTF-8”
Host: myserver:8080
Content-Length: 23

DATA=myStreamStoppedData
```

- Called from RTP onStreamStopped Event Handler, no credentials, no data:

```
Notify:http:server:8080:path/streamhandler?event=stopped

HTTP POST /path/streamhandler?event=stopped HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=”UTF-8”
Host: myserver:8080
Content-Length: 40

DATA=<notifyStreamStopped id=”stream1”/>
```

- Called from SoftKey, with credentials, with data:

```
Notify:http:myserver:8080:path/streamhandler?event=stopped:
8fh4hf7s7dhf :myStreamStoppedData
```

```
HTTP POST /path/streamhandler?event=stopped HTTP/1.1
Accept: */*
Authorization: Basic 8fh4hf7s7dhf
Content-Type: application/x-www-form-urlencoded; charset="UTF-8"
Host: myserver:8080
Content-Length: 23
```

- Called from SoftKey, no credentials, no data

```
Notify:http:server:8080:path/streamhandler?event=stopped

HTTP POST /path/streamhandler?event=stopped HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset="UTF-8"
Host: myserver:8080
Content-Length: 5
```

- Called from SoftKey with QueryStringParam URI:

```
<CiscoIPPhoneMenu>
  <MenuItem>
    <Name>Voicemail1</Name>
    <URL>QueryStringParam:id=1</URL>
  </MenuItem>
  <MenuItem>
    <Name>Voicemail2</Name>
    <URL>QueryStringParam:id=2</URL>
  </MenuItem>
  <SoftKeyItem>
    <Name>Play</Name>
    <URL>Notify:http:vmailSrvr:8080:path/play</URL>
  </SoftKeyItem>
</CiscoIPPhoneMenu>
```

If the Voicemail2 menu item was selected when the Play softkey was pressed, the following notification would be sent:

```
HTTP POST /path/play HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset="UTF-8"
Host: vmailSrvr:8080
Content-Length: 9

DATA=id=2
```

Application

The Application URI is a component of the Application Management API, which provides an improved hand-off between call mode and application mode. The Application URI allows applications to request changes to their application or window state. Applications can request to change focus, to be minimized, or to be closed.

Note The other component of the Application Management API is the Application Management Event Handler, see the "Application Event Handlers" section on page 3-22 for details.

When an Application URI request is made, it has a specific application associated with it (not just the application context) and that action can only be taken on that specific application. The Application specified in the **appId** parameter (of the displayable XML object) must be active at the time the action is requested, or an error will be returned.

This prevents open, but not active, applications which are buried on the application "stack" from closing the entire application context which would also close the active application, potentially disrupting the user's interaction with the application. This also means that if an application closes or becomes non-active (for example, if user navigates out of an application, or a new application is pushed to the context) any pending Application URI requests are immediately cancelled.

Note The 6900 series IP phones cannot add phone service under application due to hardkey mapping.

URI Format

```
App:action:priority:idleTimer:applicationId
```

Where

action = action to be taken with the application. Values include:

- RequestFocus—Makes a request to the application manager to bring the application context (window) containing this application into focus (maximize). This is a request, not a demand, as higher priority applications may prevent the application from actually gaining focus. Applications must use onAppFocusGained event handlers (see the "Application Event Handlers" section on page 3-22) to know when focus is actually gained.
  - If the requested application is Open, but not currently Active, this request will not succeed (error response).
  - If the application already has focus, the request has no effect.
- ReleaseFocus—Makes a request to the application manager to relinquish focus to another application context (essentially, a "move-to-back" request). Applications must use onAppFocusLost event handlers to know when focus is actually lost (see the "Application Event Handlers" section on page 3-22).
  - If the application does not have focus, the request has no effect.
  - If there are no other applications open (available to receive focus) then this application will retain focus.
- Minimize—Makes a request to the application manager to minimize the application context containing this application. This request always results in the application (eventually) being minimized. If the application has focus when this URI executes, the onAppFocusLost event handler will be invoked first, then the onAppMinimize handler (see the "Application Event Handlers" section on page 3-22).
  - If the requested application is Open, but not currently Active, this request will not succeed (error response).
  - If the application is already minimized, the request has no effect.
- Close—Makes a request to the application manager to close the application context containing this application.
  - If the requested application is open, but not currently active, this request will not succeed (error response). This request will result in the application context (and all applications within that context) being closed.
  - If the application has focus when this URI executes, the onAppFocusLost event handler will be invoked prior to the onAppClosed event handler (which will always be invoked).

priority = priority at which the action should be take. Values include:

- 0—Do immediately, even if user is interacting with the phone. This priority is unavailable if the Application URI is contained within an Application Management Event Handler (see the "Application Event Handlers" section on page 3-22).
- 1—Do when user is done interacting with the phone.
- 2—Do only if the user is not interacting with the phone.

idleTimer = duration of time (in seconds) the phone or application must be idle before the action should be taken. Values must range from 10-86400 (seconds); default is 60 seconds. The idleTimer value has no effect on priority=0 requests. Any pending timers are automatically cancelled when the displayable object changes for an application context.

applicationId = optional identifier of the application on which the action should be taken. Values must range in length from 1-64 string characters and cannot contain colons. The default value is the application of the displayable object in which the URI is defined.

Note If the Application URI is used in an ExecuteItem, you must specify the applicationId because the application context of the request cannot be inferred.

Error and Response

All Application URI requests are asynchronous, so the only return value indicates that the URI was successfully parsed and that the specified application was valid and currently active in its context. The application is notified of the actual state change asynchronously via the event handlers.

Condition	Status	Data
Executed successfully	0 (Success)	Success
URI syntax is invalid	1 (Parse error)	Invalid URI
Unknown application ID	6 (Internal error)	Unknown Application ID
Request made to change state of an application that is not current active	6 (Internal error)	Application is not Active

CHAPTER 6

HTTP Requests and Header Settings

Cisco Unified 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)
- HTTP Server Requests (HTTP POST)
- HTTP Header Settings
- Identifying the Capabilities of IP Phone Clients
- Accept Header
- Accessing IP Phone Information

HTTP Client Requests (HTTP GET)

The following description designates how HTTP client requests are handled:

1. The Cisco Unified IP Phone HTTP client performs an HTTP GET for a specified URL.
2. The HTTP 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 Unified 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 Unified Communications Manager 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 Unified Communications Manager JTAPI Developer Guide* for more information.

HTTP Header Settings

The following list provides definitions for HTTP header elements for Cisco Unified 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 6-3 section.
- ContentType —notifies the phone of the MIME type that was sent. See the "MIME Type and Other HTTP Headers" section on page 6-4 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 6-4 for more information.
- "Set Cookie" - see "Set-Cookie Header Setting" section on page 6-5
- "HTTP Encoding Header Setting" section on page 6-6

HTTP Refresh Setting

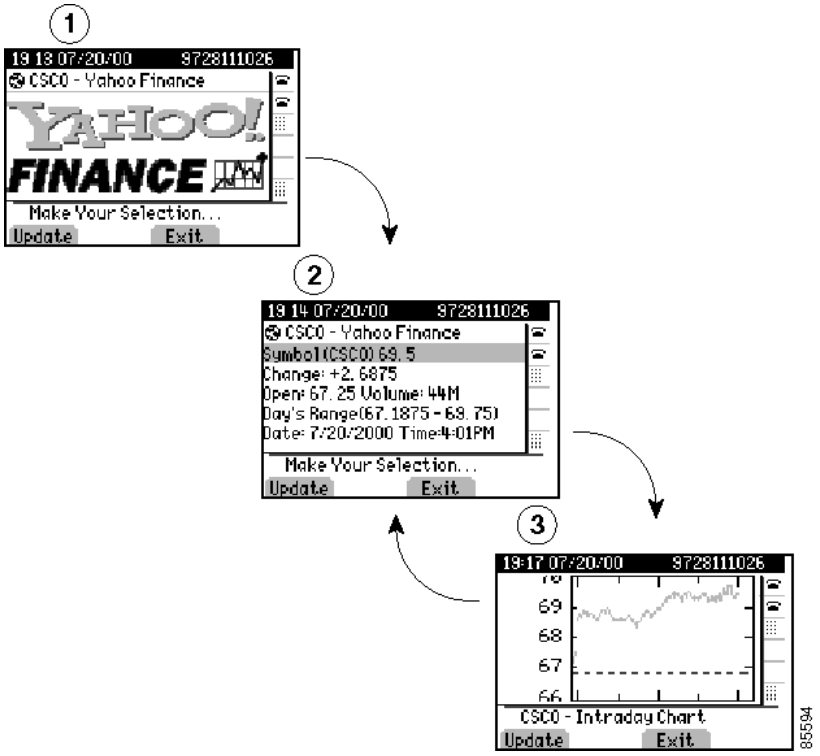
The HTTP headers that are sent with any page from an HTTP server can include a Refresh setting. This setting comprises two parameters: a time in seconds and a URL. These two parameters direct the recipient to wait the time given in the seconds parameter and then get the data to which the URL points.

The Cisco Unified 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 6-1 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 6-1 Refresh Display Sample



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 JavaScript 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 Unified IP Phone rings. Refer to the chapter on “Custom Phone Rings” in the *Cisco Unified Communications Manager System Guide* (also available in the online help).

Note The audio file should not be longer than five 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 Unified 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. On the latest generation phones which are capable of running multiple applications concurrently (Cisco Unified IP Phones 7970G, 7971G, 7961G, 7941G, 7911G), the session state is also cleared whenever the application window closes. This behavior is consistent with PC-based browsers and provides better security since anyone attempting to reopen a secure application would be forced to authenticate. 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.

HTTP Encoding Header Setting

The encoding header controls language and character settings related to localization.

Accept-Language

Cisco Unified IP Phones populate the Accept-Language HTTP request header in compliance with the HTTP specification.

For example, the Accept-Language value advertised by a phone configured for the English_United_States user locale would look like:

```
Accept-Language: en-US
```

Accept-Charset

As of this release, the phones are capable of handling UTF-8 encoding and, depending on phone model, some degree of Unicode support.

The phone models (such as the 7940, 7960, 7905 and 7912) can handle UTF-8 encoding, but will only recognize characters which can be represented by the default encoding of the phone's current user locale. For example, if the phone is currently configured to use the English_United_States locale, then it will only be able to display UTF-8 characters which map to the ISO-8859-1 character set.

The phone models (such as the 7970, 7971, 7941, 7961, and 7911) provide UTF-8 and true Unicode support. These phones provide support for more multi-byte character sets and user locales like Japanese and Chinese.

In addition to the character set for the currently configured user locale, the new phone models will also support ISO-8859-1 characters in their font files.

All phones will advertise their supported encodings using the standard HTTP Accept-Charset header. Per HTTP standard, q-values are used to specify preferred encodings. The older phone models, with more limited UTF-8 support, will specify a lower q-value for UTF-8 than the default user locale encoding.

For example, an older phone model configured with the English_United_States user locale would include an Accept-Charset header similar to the following:

```
Accept-Charset: iso-8859-1, utf-8;q=0.8
```

A newer phone model with Unicode support would advertise an Accept-Charset similar to the following:

```
Accept-Charset: utf-8,iso-8859-1;q=0.8
```

HTTP Response Headers: Content-Type

Because the phones are capable of supporting multiple character encodings, HTTP responses returned to the phones should include the 'charset' parameter on the HTTP Content-Type header. Examples of responses including the “charset” parament are shown below:

```
Content-Type: text/xml; charset=ISO-8859-1
```

```
Content-Type: text/xml; charset=UTF-8
```

```
Content-Type: text/plain; charset=Shift_JIS
```

HTTP standards state that if the encoding is not explicitly specified, ISO-8859-1 is the default.Cisco Unified IP Phones are typically compatible with this spec, but not fully compliant.

If 'charset' is not specified, the phones will use the default encoding for the currently configured user locale. So to avoid possible problems where the phone's default encoding may NOT be ISO-8859-1, the web server should explicitly set the Content-Type charset (which must match one of the Accept-Charset values specified by the phone).

Identifying the Capabilities of IP Phone Clients

XML services are supported on many Cisco Unified IP Phones, so web application servers must 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 Unified 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 Unified 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 Table 5-1 table to find which IP phone models support the URIs documented in this SDK.

Note Beginning with the IP Phone Services SDK 3.3(3), the SDK version number matches the minimum Cisco Unified Communications Manager software that is required to support it. For example, SDK version 3.3(4) gets supported only on Cisco Communications Manager 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 Unified 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 Unified 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 Unified 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 6-1 lists the available URLs and their purpose.

Table 6-1 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.
/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

CHAPTER 7

Troubleshooting Cisco Unified 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 Unified 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 another web browser
- 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 Unified 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 3, “CiscoIPPhone XML Objects” for more information.
- Validate XML applications developed prior to Cisco Unified IP Phone firmware release 8.3(2) against the more recent XML parser (see the “Updated XML Parser and Schema Enforcement” section on page B-1 for details). Some of examples of the types of errors you might encounter include:

- CiscoIPPhoneMenu Object—If the field <Name> is missing for a <MenuItem>, the original parser would stop rendering from that <MenuItem> onwards. The new parser will display a blank line in the menu list and continue to render any subsequent <MenuItem> definitions.
- CiscoIPPhoneDirectory Object—If the field <Name> is not present, the old original parser would not display the directory entry, the new parser will display the directory entry, but there will be no <Name> associated with it.
- CiscoIPPhoneInput Object—The URL and QueryStringParam fields are mandatory. The original parser would not report an error on the missing URL and on submit request would display a "Host not Found: message. If the QueryStringParam field is missing, the updated parser will report an error.
- SoftKeyItem—The Position field is mandatory. If the Position field is not present, the updated XML parser will report an error.

Error Messages

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

- `XML Error[4] = XML Parser error (Invalid Object)`
- `XML Error[5] = Unsupported XML Object (not supported by this phone model)`
- `HTTP Error[8] = Unknown HTTP Error`
- `HTTP Error[10] = HTTP Connection Failed`

CHAPTER 8

Cisco IP Phone Services Software Development Kit (SDK)

The Cisco IP Phone Services Software Development Kit (SDK) contains everything that you require to create XML applications, including necessary documentation and sample applications. Contact Cisco Developer Services to obtain the SDK at:
http://developer.cisco.com/web/ipps

These sections describe the Cisco IP Phone 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 (PDF format)
  - Cisco URL Proxy Guide (Rich Text Format)
  - Cisco LDAP Programming Guide (Microsoft Word format)
  - Cisco CIP Image Release Notes (Microsoft Word format)
  - Cisco IP Applications Samples (Microsoft Word format)

- Development Tools
  - Cip.8bi—Adobe Photoshop plug-in 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 Unified IP Phone Services ASP/Javascript Library (automatically installed)
  - Cisco Unified 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 Cisco Unified Communications Manager server
  - Cisco Unified 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) - Not supported on Cisco Unified IP Phones 7905G and 7912G
  - 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 (Cisco Unified Contact Center 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 Unified Communications Manager 4.1(2) or later.
- Cisco Unified IP Phones that supports XML services

The setup program installs a CiscoServices web project to **c:\CiscoIpServices directory**. The sample services are copied to **c:\CiscoIpServices\Services** subdirectory, and IIS and WSH example codes are provided. The web server already senses these services and you do not require 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**.

CHAPTER 9

IP Phone Service Administration and Subscription

Cisco Unified Communications Manager administrators maintain the list of services to which users can subscribe. Administrators must use Cisco Unified Communications Manager Administration to add and administer Cisco Unified IP Phone services.

Note This chapter provides just a brief overview about managing IP Phone services. For detailed up-to-date instructions, refer to the *Cisco Unified Communications Manager Administration Guide* available at the following URL:
http://www.cisco.com/en/US/products/sw/voicesw/ps556/tsd_products_support_series_home.html

These sections provide an overview about administering Cisco Unified IP Phone Services using Cisco Unified Communications Manager Administration.

- Accessing Phone Service Administration
- Adding a Phone Service
- Defining IP Phone Service Parameters
- User Service Subscription

Accessing Phone Service Administration

To access phone service administration, open Cisco Unified Communications Manager Administration and choose **Device > Device Settings > 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 Unified Communications Manager 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 Unified Communications Manager Administration and choose **Device > Device Settings > Phone Services**:

The Cisco Unified Services Configuration page in Cisco Unified Communications Manager Administration contains the fields as shown in Table 9-1.

Table 9-1 IP Phone Service Configuration Settings

Field	Description
Service Information	
Service Name	Enter the name of the service as it will display on the menu of available services in Cisco Unified CM User Options. Enter up to 32 characters for the service name.
ASCII Service Name	Enter the name of the service to display if the phone cannot display Unicode.
Service Description	Enter a description of the content that the service provides.
Service URL	Enter the URL of the server where the IP phone services application is located. Make sure that this server remains independent of the servers in your Cisco Unified Communications Manager cluster. Do not specify a Cisco Unified Communications Manager server or any server that is associated with Cisco Unified Communications Manager (such as a TFTP server or directory database publisher server). For the services to be available, the phones in the Cisco Unified Communications Manager cluster must have network connectivity to the server.
Service Category	Select a service application type.
Service Type	Select whether the service will be provisioned to the Services, Directories, or Messages button.
Service Vendor	For XML services, you can leave this field blank.
Service Version	You can leave this field blank for XML services.

Table 9-1 IP Phone Service Configuration Settings (continued)

Field	Description
Enable	Select this check box to enable the service, or clear the check box to disable the service without deleting it. **Note** You cannot delete default services. Use this field if a default service exists, but you do not want to make it available for subscription.
Enterprise Subscriptions	Select this check box to automatically provision the new service to all devices in the enterprise without requiring individual subscription. If this option is selected, the service automatically gets provisioned and does not get presented for user subscription. **Note** Be aware that this check box is available for selection only when the service is created. You cannot modify it.

Defining IP Phone Service Parameters

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 Unified IP Phone Service Parameter Configuration page in Cisco Unified Communications Manager Administration contains the fields as described in Table 9-2.

Table 9-2 IP Phone Service Parameter Settings

Field	Description
Service Parameter Information	
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 Cisco Unified CM User Options; 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 Cisco Unified CM User Options, 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 parameter entry, select the Parameter is a Password (mask contents) check box in the Configure IP phone service Parameter window in Cisco Unified Communications Manager Administration.

Tip If you change the service URL, remove a Cisco Unified 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 Cisco Unified CM User Options. 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.

CHAPTER 10

DeviceListX Report

The DeviceListX Report is no longer supported as of Cisco Unified Communications Manager Release 5.0. Retrieving real-time information from Cisco Unified Communications Manager is now supported via the Cisco Unified Communications Manager AXL Serviceability API.

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 DeviceListX does not support all devices. If you have a device that you need to support, contact Cisco Developer Support to verify whether it is supported: http://developer.cisco.com/web/ipps

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 Unified Communications Manager 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 Unified Communications Manager Administration can access the report.

Note To minimize processing overhead on the Cisco Unified Communications Manager 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 Unified Communications Manager cluster size for the targeted release without impacting core features, such as delaying dial tone. On multiserver Cisco Unified Communications Manager clusters, the report can access only from the publisher server. In large clusters where the publisher is not a Cisco Unified Communications Manager server, no possibility exists of impacting the system performance as perceived by a user.

This report is not intended for use during real time, so 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 Unified Communications Manager Administration, inherits its security from that web site, so no security issues directly relate to this report. If the Cisco Unified Communications Manager 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 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 Unified Communications Manager Administration reporting infrastructure. The following releases support DeviceListX.asp:

- Cisco CallManager Release 3.2(3)SPB
- Cisco Unified 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 Unified Communications Manager 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 Unified CallManager system that contains the report.

Note Beginning with Cisco Unified CallManager 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 10-1 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 "" = No known IP address "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 "" = Device not found "1" = Device registered "2" = Device found but not currently registered

Example 10-1 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.

Procedure

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 "Document Conventions" section on page x.

APPENDIX A

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

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

Object	Definition
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>
CiscoIPPhoneGraphicMenu	<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>
CiscoIPPhoneGraphicFileMenu	<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>

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

Object	Definition
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>
CiscoIPPhoneIconFileMenu	<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>

APPENDIX B

Cisco Unified IP Phone Services XML Schema File

These sections provide details about the XML schema supported on Cisco Unified IP Phones:

- Updated XML Parser and Schema Enforcement
- CiscoIPPhone.xsd

Updated XML Parser and Schema Enforcement

In order to provide a stable and consistent platform upon which to build enhancements to IP phones services, Cisco released an updated XML parser beginning with firmware release 8.3(2). As a result, many Cisco Unified IP Phones now contain this updated XML parser which provides a more rigid enforcement of the XML schema. This updated parser provides more error logging information when it encounters XML schema violations, and it enables developers to debug their applications more efficiently.

Cisco recommends that developers verify that their existing applications conform to the XML schema to avoid incompatibilities with any XML enhancements, particularly if you want to incorporate new URIs.

The following Cisco Unified IP Phones implement this new XML parser: 7906G, 7911G, 7921G, 7925G, 7931G, 7941G/7941G-GE 7942G, 7945G, 7961G/7961G-GE, 7962G, 7965G, 7970G/ 7971G-GE, 7975G, 6921, 6941, 6961

CiscoIPPhone.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Cisco Systems, Inc. (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="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="CiscoIPPhoneTouchAreaMenuItemType">
    <xsd:sequence>
      <xsd:element name="Name" 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" minOccurs="0">
        <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="CiscoIPPhoneTouchAreaType" minOccurs="0"/>
```

```
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CiscoIPPhoneTouchAreaType">
  <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" 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="Telephone" minOccurs="0">
      <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="CiscoIPPhoneInputItemType">
  <xsd:sequence>
    <xsd:element name="DisplayName" 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="QueryStringParam">
      <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">
      <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" minOccurs="0">
        <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="CiscoIPPhoneMenuItemType">
    <xsd:sequence>
      <xsd:element name="Name" minOccurs="0">
        <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" minOccurs="0">
        <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">
        <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">
        <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" minOccurs="0">
```

```
      <xsd:simpleType>
        <xsd:restriction base="xsd:hexBinary">
          <xsd:maxLength value="40"/>
          <xsd:minLength value="0"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CiscoIPPhoneIconMenuItemType">
  <xsd:sequence>
    <xsd:element name="Name" minOccurs="0">
      <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" minOccurs="0">
      <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" 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="CiscoIPPhoneIconFileItemType">
  <xsd:sequence>
    <xsd:element name="Index">
      <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">
      <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="CiscoIPPhoneKeyType">
  <xsd:sequence>
    <xsd:element name="Key">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="KeyPad0"/>
          <xsd:enumeration value="KeyPad1"/>
```

```
            <xsd:enumeration value="KeyPad2"/>
            <xsd:enumeration value="KeyPad3"/>
            <xsd:enumeration value="KeyPad4"/>
            <xsd:enumeration value="KeyPad5"/>
            <xsd:enumeration value="KeyPad6"/>
            <xsd:enumeration value="KeyPad7"/>
            <xsd:enumeration value="KeyPad8"/>
            <xsd:enumeration value="KeyPad9"/>
            <xsd:enumeration value="KeyPadStar"/>
            <xsd:enumeration value="KeyPadPound"/>
            <xsd:enumeration value="NavUp"/>
            <xsd:enumeration value="NavDown"/>
            <xsd:enumeration value="NavLeft"/>
            <xsd:enumeration value="NavRight"/>
            <xsd:enumeration value="NavSelect"/>
            <xsd:enumeration value="NavBack"/>
            <xsd:enumeration value="PushToTalk"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="URL" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:minLength value="0"/>
            <xsd:maxLength value="256"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="URLDown" 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="CiscoIPPhoneSoftKeyType">
    <xsd:sequence>
      <xsd:element name="Name" 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="Position">
        <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" minOccurs="0">
        <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" 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="CiscoIPPhoneDisplayableType">
    <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="SoftKeyItem" type="CiscoIPPhoneSoftKeyType" minOccurs="0"
maxOccurs="8"/>
      <xsd:element name="KeyItem" type="CiscoIPPhoneKeyType" minOccurs="0"
maxOccurs="32"/>
    </xsd:sequence>
    <xsd:attribute name="keypadTarget" use="optional" default="application">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="application"/>
          <xsd:enumeration value="applicationCall"/>
          <xsd:enumeration value="activeCall"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="appId" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:minLength value="1"/>
          <xsd:maxLength value="64"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="onAppFocusLost" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:minLength value="1"/>
          <xsd:maxLength value="256"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="onAppFocusGained" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:minLength value="1"/>
          <xsd:maxLength value="256"/>
        </xsd:restriction>
```

```
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="onAppMinimized" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:minLength value="1"/>
          <xsd:maxLength value="256"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="onAppClosed" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:minLength value="1"/>
          <xsd:maxLength value="256"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:element name="CiscoIPPhoneExecute">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ExecuteItem" type="CiscoIPPhoneExecuteItemType" maxOccurs="3"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ResponseItem" type="CiscoIPPhoneResponseItemType"
maxOccurs="3"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneError">
    <xsd:complexType>
      <xsd:attribute name="Number" type="xsd:unsignedShort" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneText">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="CiscoIPPhoneDisplayableType">
          <xsd:sequence>
            <xsd:element name="Text" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:minLength value="0"/>
                  <xsd:maxLength value="4000"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneInput">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="CiscoIPPhoneDisplayableType">
          <xsd:sequence>
            <xsd:element name="URL">
              <xsd:simpleType>
```

```
                <xsd:restriction base="xsd:string">
                  <xsd:minLength value="1"/>
                  <xsd:maxLength value="256"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="InputItem" type="CiscoIPPhoneInputItemType" minOccurs="0"
maxOccurs="5"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneDirectory">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="CiscoIPPhoneDisplayableType">
          <xsd:sequence>
            <xsd:element name="DirectoryEntry" type="CiscoIPPhoneDirectoryEntryType"
minOccurs="0" maxOccurs="32"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneImage">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="CiscoIPPhoneDisplayableType">
          <xsd:sequence>
            <xsd:element name="LocationX" default="0" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:short">
                  <xsd:minInclusive value="-1"/>
                  <xsd:maxInclusive value="132"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationY" default="0" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:short">
                  <xsd:minInclusive value="-1"/>
                  <xsd:maxInclusive value="64"/>
                </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" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:hexBinary">
                  <xsd:maxLength value="2162"/>
                  <xsd:minLength value="0"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneImageFile">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="CiscoIPPhoneDisplayableType">
          <xsd:sequence>
            <xsd:element name="LocationX" default="0" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:short">
                  <xsd:minInclusive value="-1"/>
                  <xsd:maxInclusive value="297"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationY" default="0" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:short">
                  <xsd:minInclusive value="-1"/>
                  <xsd:maxInclusive value="167"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="URL">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="256"/>
                  <xsd:minLength value="1"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneMenu">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="CiscoIPPhoneDisplayableType">
          <xsd:sequence>
            <xsd:element name="MenuItem" type="CiscoIPPhoneMenuItemType" minOccurs="0"
maxOccurs="100"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
```

```
  </xsd:element>
  <xsd:element name="CiscoIPPhoneIconMenu">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="CiscoIPPhoneDisplayableType">
          <xsd:sequence>
           <xsd:element name="MenuItem" type="CiscoIPPhoneIconMenuItemType" minOccurs="0"
maxOccurs="32"/>
            <xsd:element name="IconItem" type="CiscoIPPhoneIconItemType" minOccurs="0"
maxOccurs="10"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneIconFileMenu">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="CiscoIPPhoneDisplayableType">
          <xsd:sequence>
           <xsd:element name="MenuItem" type="CiscoIPPhoneIconMenuItemType" minOccurs="0"
maxOccurs="32"/>
            <xsd:element name="IconItem" type="CiscoIPPhoneIconFileItemType" minOccurs="0"
maxOccurs="10"/>
          </xsd:sequence>
          <xsd:attribute name="IconIndex" type="xsd:unsignedShort" use="optional"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneGraphicMenu">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="CiscoIPPhoneDisplayableType">
          <xsd:sequence>
            <xsd:element name="LocationX" default="0" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:short">
                  <xsd:minInclusive value="-1"/>
                  <xsd:maxInclusive value="132"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationY" default="0" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:short">
                  <xsd:minInclusive value="-1"/>
                  <xsd:maxInclusive value="64"/>
                </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" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:hexBinary">
                  <xsd:maxLength value="2162"/>
                  <xsd:minLength value="0"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="MenuItem" type="CiscoIPPhoneMenuItemType" minOccurs="0"
maxOccurs="12"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneGraphicFileMenu">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="CiscoIPPhoneDisplayableType">
          <xsd:sequence>
            <xsd:element name="LocationX" default="0" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:short">
                  <xsd:minInclusive value="-1"/>
                  <xsd:maxInclusive value="297"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="LocationY" default="0" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:short">
                  <xsd:minInclusive value="-1"/>
                  <xsd:maxInclusive value="167"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="URL">
              <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:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneStatus">
    <xsd:complexType>
```

```
<xsd:sequence>
  <xsd:element name="Text" 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" minOccurs="0">
    <xsd:simpleType>
      <xsd:restriction base="xsd:unsignedShort">
        <xsd:minInclusive value="0"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="LocationX" default="0" minOccurs="0">
    <xsd:simpleType>
      <xsd:restriction base="xsd:short">
        <xsd:minInclusive value="-1"/>
        <xsd:maxInclusive value="105"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="LocationY" default="0" minOccurs="0">
    <xsd:simpleType>
      <xsd:restriction base="xsd:short">
        <xsd:minInclusive value="-1"/>
        <xsd:maxInclusive value="20"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="Width">
    <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">
    <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">
    <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" minOccurs="0">
    <xsd:simpleType>
      <xsd:restriction base="xsd:hexBinary">
        <xsd:minLength value="0"/>
        <xsd:maxLength value="557"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
```

```
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CiscoIPPhoneStatusFile">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Text" 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" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:unsignedShort">
              <xsd:minInclusive value="0"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="LocationX" default="0" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:short">
              <xsd:minInclusive value="-1"/>
              <xsd:maxInclusive value="261"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="LocationY" default="0" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:short">
              <xsd:minInclusive value="-1"/>
              <xsd:maxInclusive value="49"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="URL">
          <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:element>
</xsd:schema>
```

INDEX

Symbols

<< softkey 5-6

Numerics

8961
- Color depth 3-10
- Resolution 3-10
- URIs supported 5-1, 5-3
- XML Objects supported 2-1, 3-2

9951
- Color depth 3-10
- Reolution 3-10
- URIs supported 5-1, 5-3
- XML Objects supported 2-1, 3-2

9971
- Color depth 3-10
- Resolution 3-10
- URIs supported 5-1, 5-3
- XML Objects Supported 3-2

A

accept header, support for 6-8

audio types, supported 6-4

C

Cancel softkey 5-6

CGI script
- Execute 6-9
- Screenshot 6-9

Cisco IP Phone Services Software Development Kit (SDK)
- components 8-1
- development tools 8-2
- sample services
  - description 8-2
  - requirements 8-3

CiscoIPPhone XML Objects
- CiscoIPPhoneDirectory 3-6
- CiscoIPPhoneError 3-19
- CiscoIPPhoneExecute 3-18
- CiscoIPPhoneGraphicFileMenu 3-11
- CiscoIPPhoneGraphicMenu 3-11
- CiscoIPPhoneIconFileMenu 3-14
- CiscoIPPhoneIconMenu 3-12
- CiscoIPPhoneImage 3-7
- CiscoIPPhoneImageFile 3-9
- CiscoIPPhoneInput 3-5
- CiscoIPPhoneMenu 3-3
- CiscoIPPhoneResponse 3-19
- CiscoIPPhoneStatus 3-14
- CiscoIPPhoneStatusFile 3-17
- CiscoIPPhoneText 3-4
- supported phone models 3-2
- understanding 3-1

Cisco Unified Communications Manager
- adding services 9-2
- defining service parameters 9-3
- subscribing to services 9-4

Cisco Unified IP Phone models
- accessing information about 6-9
- supported displays and colors 3-9
- supported URIs 5-1, 5-3
- supported XML objects 3-2, 3-16, 3-17

client requests, using HTTP 6-1
content expiration, header setting 6-4
cookie behavior 6-5

D

development tools, included with SDK 8-2
Device control URI
  Display 5-5
  Key 5-2
DeviceInformation 6-9
device information URLs 6-9
DeviceList XML Object
  attributes 10-4
  description 10-3
DeviceListX Report 10-1
DeviceLog 6-9
Dial, used as URI 5-13
Dial softkey 5-6
directories
  creating 3-6
  customizing 3-7
documentation
  additional i-ix

E

EditDial, used as URI 5-14
EditDial softkey 5-6
error codes, description 3-19, 5-9
EthernetInformation 6-9
Execute, CGI script 6-9
executing items
  priority levels 3-18
  sending requests 3-18
Exit softkey 5-6

F

featuers, supported 2-1

G

graphic menus, creating 3-11
graphics
  displaying color images 3-9
  displaying grayscale images 3-7
  PNG support 3-9, 3-14
  using with status 3-14

H

header settings, for HTTP 6-2
HTML URL
  DeviceInformation 6-9
  DeviceLog?n 6-9
  EthernetInformation 6-9
  EthernetInformation?n 6-9
  NetworkConfiguration 6-9
  StreamingStatistics?n 6-9
HTTP
  client requests 6-1
  header settings 6-2
    accept 6-8
    content expiration 6-4
    MIME type 6-4
    refresh setting 6-3
    set-cookie 6-5
    x-CiscoIPPhoneDisplay 6-7
    x-CiscoIPPhoneModelName 6-7
    x-CiscoIPPhoneSDKVersion 6-8
  how used 6-1
  server requests 6-1
HTTP GET 6-1
HTTP POST 6-1

I

icon menus, creating
- color 3-14
- grayscale 3-12

Init, used as URI 5-16

input forms
- creating 3-5
- supported input types 3-6

M

menus
- graphic 3-11
- icon, color 3-14
- icon, grayscale 3-12
- text 3-3

MIME type, for HTTP 6-4

N

NetworkConfiguration 6-9

Next softkey 5-6

P

Play, used as URI 5-12

PNG images
- displaying on screen 3-9
- using in menus 3-14

PortInformation 6-9

priority levels, for executing items 3-18

R

refresh setting, for HTTP 6-3

RTPMTx, used as URI 5-11

S

Screenshot, CGI script 6-9

SDK. *See* Cisco IP Services Software Development Kit (SDK)

Search softkey 5-6

Select softkey 5-6

server requests, using HTTP 6-1

services
- adding to Cisco Unified Communications Manager 9-2
- defining parameters of in Cisco Unified Communications Manager 9-3
- requirements for samples 8-3
- samples included with SDK 8-2
- subscribing to in Cisco Unified Communications Manager 9-4

set-cookie, header setting 6-5

softkey
- << 5-6
- Cancel 5-6
- Dial 5-6
- EditDial 5-6
- Exit 5-6
- Next 5-6
- Search 5-6
- Select 5-6
- Submit 5-6
- Update 5-6
- valid actions for object types 5-6

status, displaying for applications 3-14, 3-17

StreamingStatistics 6-9

Submit softkey 5-6

T

text menus, creating 3-3

text messages, displaying 3-4

troubleshooting
- error messages 7-2

tips 7-1
XML parsing errors 7-1

U

uniform resource identifiers (URI)
description 5-1
miscellaneous
Dial 5-13
EditDial 5-14
Init 5-16
Play 5-12
to control RTP streaming 5-9
RTPMTx 5-11
Update softkey 5-6
URI. *See* uniform resource identifiers

X

x-CiscoIPPhoneDisplay 6-7
x-CiscoIPPhoneModelName 6-7
x-CiscoIPPhoneSDKVersion 6-8
XML schema file B-1
XML URL
DeviceInformationX 6-9
DeviceLogX?n 6-9
EthernetInformationX 6-9
NetworkConfigurationX 6-9
PortInformationX?nX 6-9
StreamingStatisticsX?n 6-9