



**Avaya one-X™ Deskphone Edition
for 9600 Series IP Telephones Application
Programmer Interface (API) Guide**

16-600888
Issue 5
November 2009

Notice

While reasonable efforts were made to ensure that the information in this document was complete and accurate at the time of printing, Avaya Inc. can assume no liability for any errors. Changes and corrections to the information in this document may be incorporated in future releases.

For full legal page information, please see the complete document, Avaya Legal Page for Hardware Documentation, Document number 03-600759.

To locate this document on our Web site, simply go to <http://support.avaya.com/> and search for the document number in the search box.

Documentation disclaimer

Avaya Inc. is not responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya. Customer and/or End User agree to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation to the extent made by the Customer or End User.

Link disclaimer

Avaya Inc. is not responsible for the contents or reliability of any linked Web sites referenced elsewhere within this documentation, and Avaya does not necessarily endorse the products, services, or information described or offered within them. We cannot guarantee that these links will work all of the time and we have no control over the availability of the linked pages.

Warranty

Avaya Inc. provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product, while under warranty, is available through the following Web site: <http://support.avaya.com/>

Copyright

Except where expressly stated otherwise, the Product is protected by copyright and other laws respecting proprietary rights. Unauthorized reproduction, transfer, and or use can be a criminal, as well as a civil, offense under the applicable law.

Avaya support

Avaya provides a telephone number for you to use to report problems or to ask questions about your product. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Web site:

<http://www.avaya.com/support>

Software License

USE OR INSTALLATION OF THE PRODUCT INDICATES THE END USER'S ACCEPTANCE OF THE TERMS SET FORTH HEREIN AND THE GENERAL LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE AT <http://support.avaya.com/LicenseInfo/> ("GENERAL LICENSE TERMS"). IF YOU DO NOT WISH TO BE BOUND BY THESE TERMS, YOU MUST RETURN THE PRODUCT(S) TO THE POINT OF PURCHASE WITHIN TEN (10) DAYS OF DELIVERY FOR A REFUND OR CREDIT.

Avaya grants End User a license within the scope of the license types described below. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of licenses or units of capacity is specified in the Documentation or other materials available to End User. "Designated Processor" means a single stand-alone computing device. "Server" means a Designated Processor that hosts a software application to be accessed by multiple users. "Software" means the computer programs in object code, originally licensed by Avaya and ultimately utilized by End User, whether as stand-alone Products or pre-installed on Hardware. "Hardware" means the standard hardware Products, originally sold by Avaya and ultimately utilized by End User.

License Type(s):

Designated System(s) License (DS). End User may install and use each copy of the Software on only one Designated Processor, unless a different number of Designated Processors is indicated in the Documentation or other materials available to End User. Avaya may require the Designated Processor(s) to be identified by type, serial number, feature key, location or other specific designation, or to be provided by End User to Avaya through electronic means established by Avaya specifically for this purpose.

Third-party Components

Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the Product ("Third Party Terms"). Information identifying Third Party Components and the Third Party Terms that apply to them is available on Avaya's Web site at:

<http://support.avaya.com/ThirdPartyLicense/>

Interference

Using a cell, mobile, or GSM telephone, or a two-way radio in close proximity to an Avaya IP Telephone might cause interference.

Security

See <http://support.avaya.com/security> to locate and/or report known vulnerabilities in Avaya products. See <http://support.avaya.com> to locate the latest software patches and upgrades. For information about secure configuration of equipment and mitigation of toll fraud threats, see the Avaya Toll Fraud and Security Handbook at <http://support.avaya.com/>.

Patents

T9 Text Input and other products are covered by one or more of the following patents: U.S. Pat. Nos. 5,187,480, 5,818,437, 5,945,928, 5,953,541, 6,011,554, 6,286,064, 6,307,548, 6,307,549, and 6,636,162, 6,646,573, 6,970,599; Australia Pat. Nos. 727539, 746674, 747901; Austria Pat. Nos. AT225534, AT221222; Brazil P.I. No. 9609807-4; Canada Pat. Nos. 1,331,057, 2,227,904, 2,278,549, 2,302,595; Japan Pat. Nos. 3532780, 3492981; United Kingdom Pat. No. 2238414B; Hong Kong Standard Pat. No. HK1010924; Republic of Singapore Pat. Nos. 51383, 66959, 71979; European Pat. Nos. 1 010 057 (98903671.0), 1 018 069 (98950708.2); Republic of Korea Pat. Nos. KR201211B1, KR226206B1, 402252; People's Republic of China Pat. No. ZL96196739.0; Mexico Pat. Nos. 208141, 216023, 218409; Russian Federation Pat. Nos. 2206118, 2214620, 2221268; and additional patent applications are pending.

Table of Contents

| | |
|---|-----------|
| About This Guide | 7 |
| About this Document | 7 |
| Intended Audience | 7 |
| Document Organization..... | 8 |
| Issue Date | 9 |
| How to Use This Document..... | 9 |
| Terms Used in This Document..... | 9 |
| Conventions Used in This Document | 15 |
| Symbolic Conventions..... | 15 |
| Typographic Conventions..... | 15 |
| Online Documentation..... | 15 |
| Related Documentation..... | 16 |
| Avaya Documents | 16 |
| Other Documents | 16 |
| IETF Documents | 16 |
| Chapter 1: IP Telephone Interfaces..... | 17 |
| Overview | 17 |
| Existing Interfaces | 18 |
| Push: | 19 |
| Web: | 19 |
| Chapter 2: Push Interface Overview | 21 |
| Introduction | 21 |
| Push Feature Description | 21 |
| Push Architecture | 22 |
| The Push Flow Process | 22 |
| The Push/Pull Process – A Two-Step View | 23 |
| Push Message Flow | 24 |
| About the Push Agent | 26 |
| HTTP Server | 26 |
| Push Agent - HTTP POST Address | 27 |
| Push Types | 27 |
| Push Content Requests | 29 |
| Chapter 3: Creating Push Messages | 31 |
| Introduction | 31 |
| The Display Push Type | 31 |
| Web Browser Features..... | 31 |
| Alerts | 32 |
| Display Push Example 1: | 32 |

| | |
|---|----|
| Priorities and States | 32 |
| Display Push XML Messages..... | 35 |
| Using the <postfield> Tag | 36 |
| Display Push Example 2: | 37 |
| Push Agent..... | 37 |
| Push Content (PC) | 37 |
| The Top Line Push Type | 39 |
| Text Message Features..... | 39 |
| Alerts | 39 |
| Top Line Push Example 1: | 40 |
| Priorities and States | 40 |
| Top Line Push XML Messages | 42 |
| Using the <postfield> Tag | 43 |
| Top Line Push Example 2: | 44 |
| Push Agent..... | 44 |
| Top Line Push Content (PC) | 45 |
| The Receive Audio Push Type | 46 |
| Interrupt Screens..... | 47 |
| Receive Audio Push Features..... | 47 |
| Alerts | 47 |
| Receive Audio Push Example: | 48 |
| Priorities and States | 48 |
| Receive Audio Push XML Messages | 51 |
| RTP Port | 53 |
| Push Agent..... | 54 |
| Receive and Multicast Receive Push Content (PC) | 54 |
| Receive Audio Push Content (PC) | 55 |
| The Transmit Audio Push Type | 58 |
| Processing | 59 |
| Interrupt Screens..... | 59 |
| Transmit Audio Push Features..... | 59 |
| Alerts | 59 |
| Transmit Audio Push Example | 60 |
| Priorities and States | 60 |
| Transmit Audio Push XML Messages | 62 |
| RTP Port | 63 |
| Push Agent..... | 64 |
| Transmit Audio Push Content (PC) | 64 |
| The Subscribe Push Type | 66 |
| Subscribe Push Features | 66 |
| Alerts | 66 |
| Priorities and States | 67 |
| Successful Push Response..... | 67 |
| Subscribe XML Messages..... | 67 |
| Push Agent..... | 68 |
| Subscribe Push Content (PC) | 68 |
| Using the <Subscribe> Tag..... | 69 |
| The Phonexml Push Type (for R2.2 & R2.5+ SIP IP Telephones only)..... | 70 |
| Phone Features..... | 70 |

| | |
|---|-----------|
| Alerts | 73 |
| Phonexml Push Example 1: | 73 |
| Priorities and States | 74 |
| Using the <postfield> Tag | 75 |
| Phonexml Push Example 2: | 76 |
| Push Agent..... | 77 |
| phonexml Push Content (PC)..... | 77 |
| Chapter 4: Push Administration | 81 |
| Introduction | 81 |
| Requirements..... | 81 |
| TPSLIST..... | 81 |
| SUBSCRIBELIST | 82 |
| PUSHCAP | 82 |
| PUSHPORT | 83 |
| Security | 83 |
| Trusted Push Server List (TPSLIST)..... | 83 |
| Validation Scenarios | 85 |
| Recommendations: | 86 |
| Subscription Service | 87 |
| Subscriber Service | 87 |
| Subscription List (SUBSCRIBELIST) | 88 |
| Subscription Update..... | 89 |
| Retry Timer..... | 89 |
| Denial of Service Timer..... | 89 |
| Chapter 5: Troubleshooting the Push Interface | 91 |
| Avaya HTTP Header Extensions (x-Push-Status Codes)..... | 91 |
| HTTP Error Messages..... | 94 |
| Chapter 6: About the Web Browser | 97 |
| Introduction | 97 |
| Differences in 9600 Series IP Telephone Models | 104 |
| 9610 WML Browser Feature Differences | 104 |
| 9670G WML Browser Feature Differences | 105 |
| Summary of differences among the models in the product line | 105 |
| Physical Attributes..... | 106 |
| Display Area Specifications..... | 107 |
| Variables set during browser initialization | 110 |
| Web Browser Navigation..... | 110 |
| Multiple Paging Indicators | 113 |
| Scrolling | 113 |
| Truncation Rules and Links..... | 116 |
| Truncating Lines and Words | 116 |
| Links..... | 116 |
| Image Support..... | 116 |
| WBMP Images | 116 |

| | |
|---|------------|
| JPEG Images | 117 |
| Image Rendering..... | 117 |
| Linked Images..... | 117 |
| Number of Images Supported | 118 |
| Enabling Text Entry | 118 |
| Text Editing Modes | 120 |
| Character Set Support | 121 |
| Web History Stack..... | 122 |
| Requirements for Deck/Card Elements | 122 |
| Wireless Telephony Applications (WTA) | 122 |
| Syntax Implementation..... | 123 |
| Button-Push URI's..... | 129 |
| Administering the Avaya Menu for the Web Browser | 129 |
| Web-Related System Parameters..... | 130 |
| Chapter 7: Developing Web Pages for the Browser..... | 133 |
| Introduction | 133 |
| HTTP Support | 133 |
| HTTP Authentication | 134 |
| HTTP Protocol..... | 135 |
| HTTP Header - User Agent | 135 |
| HTTP Error Messages..... | 136 |
| Summary of WML Tags and Attributes..... | 138 |
| WML Protocol..... | 142 |
| WML Tags and Attributes..... | 143 |
| WML Document Skeleton..... | 143 |
| Text Elements | 146 |
| Anchor Elements..... | 147 |
| Image Elements | 148 |
| Image Map Support Elements (H.323 9600 Series IP Telephones only) | 149 |
| Event Elements | 151 |
| Task Elements | 156 |
| Input Elements | 158 |
| Variable Elements..... | 165 |
| Character Entities..... | 166 |
| Colors and Fonts..... | 166 |
| Access Key Input Mode (AIM)..... | 167 |
| Support for Access Keys | 167 |
| AIM Considerations..... | 170 |
| Terminating AIM..... | 172 |
| Developing Web Pages for the 9610 IP Telephone | 172 |
| 9610 Display Characteristics..... | 172 |
| Support for Cascading Style Sheets | 172 |
| Cascading Order | 173 |
| CSS2 Specifications..... | 177 |

| | |
|---|------------|
| Chapter 8: Web Applications | 181 |
| Introduction | 181 |
| Application Platform Requirements | 183 |
| Installing the Thin Client Directory on the Server | 183 |
| Pre-Installation Requirements (Apache/PHP) | 183 |
| Avaya-Provided Download Files | 183 |
| Avaya LDAP Directory Application for 9600 Series IP Telephones | 184 |
| Installing the Thin Client Directory | 185 |
| Web Application User Interface | 188 |
| Generic User Interface Screen Characteristics | 188 |
| Web Application Search Screen | 188 |
| Web Application Successful Search Screen | 189 |
| Web Application Detail Screen | 190 |
| Web Application Directory Trouble Screen | 192 |
| Directory Database Administration Interface | 196 |
| Configuring the General Directory Application Administration Screen | 198 |
| Configuring the Directory Application Search Administration Screen | 200 |
| Configuring the Directory Application Details Administration Screen | 201 |
| Configuring the Directory Application Softkey Administration Screen | 203 |

About This Guide

About this Document

This document describes how to set up two optional Avaya application interfaces, the Web browser and the Push interface. This document covers both H.323 software Release 3.1 and SIP software Releases 2.2, 2.5, or later for the 9600 Series IP Telephones and describes only the behavior of those IP telephones. The performance and behavior of the application or application server(s) are not addressed.

Intended Audience

This document is intended for application developers and System Administrators who develop or implement Web- or Push-based applications for Avaya IP Telephones. [Chapter 4: Push Administration](#) is intended for System Administrators who need to enable the Push interface and set up Subscription Server Addresses on the HTTP server.

Caution:

Avaya does not support many of the products mentioned in this document. Take care to ensure that there is adequate technical support available for the DHCP, HTTP, and LDAP servers. If the servers are not functioning correctly, the Avaya IP Telephones might not operate correctly.

Document Organization

This guide contains the following chapters:

[Chapter 1: IP Telephone Interfaces](#)

Describes the available Avaya IP Telephone interfaces.

[Chapter 2: Push Interface Overview](#)

Provides an overview of the Push technology. This chapter describes the Push Message Flow process with diagrams, and gives an overview of Push features as applicable to 9600 Series IP Telephones.

[Chapter 3: Creating Push Messages](#)

Describes the message types that can be sent (pushed) to an IP telephone in detail, and provides setup requirements and examples.

[Chapter 4: Push Administration](#)

Covers Push security features and recommendations, and server setup.

[Chapter 5: Troubleshooting the Push Interface](#)

Describes messages received during Push interface setup or processing, and provides suggested resolutions.

[Chapter 6: About the Web Browser](#)

Provides a general overview of the Web browser and application setup.

[Chapter 7: Developing Web Pages for the Browser](#)

Provides information about creating and customizing Web sites for viewing on applicable 9600 Series IP Telephones. Also describes the current capabilities and limitations of the Web browser.

[Chapter 8: Web Applications](#)

Provides information on administering an LDAP directory for applicable 9600 Series IP Telephones.

Issue Date

This is the fifth release of this document, issued in November 2009. Earlier releases were issued as follows:

- February, 2009
- September 2008
- January 2007
- July 2006

How to Use This Document

This guide is organized to help you find topics in a logical manner. Read it from start to finish for a thorough understanding of the interfaces, or use the Table of Contents to locate specific features. Note that topics/elements specific only to phones running SIP software or H.323 software are marked as such.

Terms Used in This Document

| Term | Description |
|-------------------------|--|
| ACM | <i>Avaya Communication Manager.</i> A member of a family of Avaya PBX's providing advanced call features. Avaya IP Telephones register on or login to the ACM. |
| AIM | <i>Access Key Input Mode.</i> A new text entry mode that allows a user to access a particular URL by selecting a single dialpad key. |
| Alerts | An optional notification such as a series of ring pings to alert the user to an incoming Push Message. |
| Application Area | The usable display area between the Prompt Line and Softkey labels. |
| Application Line | The display area line that indicates application-specific messages. |

| Term | Description |
|-----------------|--|
| Card | A WML card is similar to an HTML page, but WML delivers a set (deck) of closely related cards. The complete WML page comprises a collection of various cards, of which only one is visible on the browser at one time. As each of the cards is labeled by a name and ID, they can be linked together without difficulty. The WML card author determines the content of the card. The browser determines how this card will be displayed (rendered). |
| CDATA | Text, which can contain numeric or named character entities. CDATA, a DTD data type, is used only in attribute values. |
| CSS2 | <i>Cascading Style Sheets Version 2.</i> |
| Deck | A deck can be described as a stack of cards. When the browser downloads a WML page, it really is downloading a deck of cards but only one card in the deck is visible at a time. |
| DTD | <i>Document Type Definition.</i> The DTD defines the names and contents of all elements that permissible on a WML page, the order in which the elements must appear, the contents of all elements, attributes and default values. |
| Elements | Elements are the essential components that make up a single WML document. |
| FLOW | The flow type represents “card-level” information. In general, flow is used anywhere general markup can be included. |
| Focus | Since the phone has no mouse to navigate around the screen, the line buttons if shown, or the OK button are used to select a particular line on the display. Selecting a line serves to “to bring that line into focus.” Focusing on a line is used to select a line for text entry or to select a line that contains a link to another URL (card). Additionally, new titles can (not always) be presented to the user as each line on the screen is individually brought “into focus” (selected by pressing the Line or OK buttons). |
| Frame | The area in which the Web page is displayed. |
| href | The href attribute refers to either a relative or an absolute Uniform Resource Locator. |

| Term | Description |
|-------------------------|---|
| HTML | <i>Hyper Text Markup Language</i> is a text-based way of describing data for transmission over the Internet HTML is usually used with larger, color displays. |
| HTTP | <i>Hyper Text Transfer Protocol</i> , used to request and transmit pages on the World Wide Web. |
| IGMP | The Internet Group Management Protocol (IGMP) is a communications protocol used to manage the membership of Internet Protocol multicast groups. IGMP is used by IP hosts and adjacent multicast routers to establish multicast group memberships. |
| Interrupt Screen | A screen that automatically accompanies a standalone audio push. Interrupt screens provide the user with specific information about terminating the audio push. |
| IP | Internet Protocol – a suite of information exchanged message sets widely used for data transmission and increasingly used for the transmission of voice. |
| Link | The URI that is used to chain cards together. |
| Mode | Push Priority type – normal or barge – to distinguish between emergency messages and ideal messages. |
| Multicast | A technique developed to send packets from one location in the Internet to many other locations without any unnecessary packet duplication. In multicasting, one packet is sent from a source and is replicated as needed in the network to reach as many end users as necessary. |
| NMTOKEN | <i>A name token</i> , containing any mixture of name characters, as defined by the XML. |
| PBX | <i>Private Branch Exchange</i> – A generic name for a premise-based switch supporting telephony features owned by an enterprise. |
| PCDATA | <i>Parsed CDATA</i> . Text that can contain numeric or named character entities. This text can contain tags. PCDATA, a DTD data type, is used only in elements. |
| Phonexml | Push type that allows the following information/functionality to be pushed to the phone: customize display name, change the display language to specified language, clear the call-log and web history of the phone. |

| Term | Description |
|--------------------------------------|---|
| Prompt Line | The third line in the top display area. The current application uses the Prompt Line to provide context-specific prompts, hints, explanations, help, or similar information. |
| Push Agent or PA | The telephone software that is capable of receiving a Push Message from a server (Push Initiator). |
| Push Initiator or Push Server | A Web application that is capable of transmitting the Push Message to the Push Agent. |
| Push Content (PC) | A valid XML or a WML file that contains a <Response> tag as the root or a <WML> as the root. The file carries the actual information to be displayed or streamed on to an IP telephone. |
| Push Message (PM) | An XML message that contains a <Push> tag as the root. The Push Message uses a <go> tag to specify a URI to which the Push Agent can launch a request for Push Content. |
| Push State | When the telephone is in busy state such as an active phone call or user entering information in the Contacts Application. |
| Registration | The registration is a scheme of allowing an Avaya IP Telephone to authenticate itself with the Avaya Communication Manager. The Avaya Media Server switch supports registering and authenticating Avaya IP Telephones using the extension and password. |
| RTP Audio | An audio stream received from an application outside the context of a telephone call. The audio Push Message can be accompanied with an optional notification alert. |
| SUBSCRIBELIST | The subscription list for potential pushed content contains zero or more fully qualified URLs, separated by commas without any intervening spaces, up to 255 ASCII characters, including commas. The default is "" (Null). |
| Subscription Servers | A server or a database that stores the information for a Push-enabled IP telephone such as IP Address, Extension, MAC Address, etc. |
| Title Line | The second line in the top display area. Comprised of the current application title, subtitle (if applicable), and choice or Web paging indicators as applicable. |

| Term | Description |
|-------------------------------------|---|
| Top Line | The top area of the display is subdivided horizontally into a Top Line, a Title Line, and a Prompt Line, each extending across the entire width of the 318 pixel usable area for the 9600 Series IP Telephones. The Top Line contains current status information. Examples of status information are the extension number, time and date, and icons for phone- or call-related data. For the 9610 IP Telephone, which has only a Top Line, messages and prompts appear there. |
| TPSLIST | <i>List of Trusted Push Servers</i> , contains one or more domains and paths in DNS format, separated by commas without any intervening spaces, up to 255 ASCII characters, including commas. The default is "" (Null) |
| Transmit Push | Transmit Audio Push. An audio stream that is not associated with a call, based on information obtained from a pushed HTTP request. Allows end users to initiate and transmit audio pushes. Also called "XMIT Push." |
| Trusted Push Server (TPS) | Application Server with a URI that conforms to the security settings as established by the PUSH parameter in the script file. The Trusted Push Server and the Push Initiator can be the same entity. |
| Trusted Receive Server (TRS) | Application Server with a URI that conforms to the security settings as established by the PUSH parameter in the script file. The Trusted Push Server, the Push Initiator, and the Trusted Receive Server can be the same entity. |
| Type | Type specifies a tag or attribute. |
| User Agent | Software that interprets WML, WML script, WTAI and other forms of codes. |
| VDATA | A DTD data type representing a string that can contain variable references. This type is only used in attribute values. |
| W3C | <i>World Wide Web Consortium.</i> |
| WAP | <i>Wireless Application Protocol.</i> An open global standard for wireless solutions that includes WML. |
| WBMP | WBMP is a bitmap graphic format that is required for the integration of graphics into WML pages. |
| WML | <i>Wireless Markup Language</i> is a subset of XML, used by the Avaya IP Telephone Web browser to communicate with WML Servers. |

| Term | Description |
|----------------------|--|
| WML homedeck | The WML start page, derived from Homepage. |
| WMLscript | A scripting language specifically designed for programming mobile devices. It is based on ECMAScript, but has been optimized for low bandwidth communication and limited processing power and memory. |
| WML tags | WML cards/deck are composed of a number of elements. Each element begins with a descriptive tag. Tags are indicated by a pair of angled brackets that start with the < character and end with the > character. The first element inside the angled brackets is the tag name. |
| WTA | <i>Wireless Telephony Application(s)</i> . An extension of the WAE (Wireless Application Environment) that provides a set of interfaces to a mobile device's telephony functionality. |
| WTAI | <i>Wireless Telephony Application Interface</i> is a set of interfaces that extend the WAE (Wireless Application Environment) to include telephony applications. |
| x-Push-Status | The Push Agent HTTP extension header. This extension is used by the Push Agent to send the Push Message status to the Push Initiator. |
| XML | <i>eXtensible Markup Language</i> . W3C's standard for Internet Markup Languages. WML is one of these languages. |
| xml:lang | The xml:lang attribute specifies the natural or formal language of an element or its attributes. This is a DTD term. |

Conventions Used in This Document

This guide uses the following textual, symbolic, and typographic conventions to help you interpret information.

Symbolic Conventions

Note:

This symbol precedes additional information about a topic.

Caution:

This symbol is used to emphasize possible harm to software, possible loss of data, or possible service interruptions.

Typographic Conventions

This guide uses the following typographic conventions:

| | |
|-------------------|--|
| <u>Document</u> | Underlined type indicates a section or sub-section in this document containing additional information about a topic. |
| <i>"Document"</i> | Italic type enclosed in quotes indicates a reference to specific section or chapter of an external document. |
| <i>italics</i> | Italic type indicates the result of an action you take or a system response in step by step procedures. |
| Conference | In step by step procedures, words shown in bold represent a single telephone button that should be pressed/selected. |
| message | Words printed in this type are messages, prompts, code excerpts, code samples, and XML tags. |

Online Documentation

The online documentation for this guide and related Avaya documentation is located at the following URL:

<http://www.avaya.com/support>

Related Documentation

Avaya Documents

Avaya one-X™ Deskphone Edition for 9600 Series IP Telephone Administrator Guide (Document Number 16-603158) and *Avaya one-X™ Deskphone Edition for 9600 Series SIP IP Telephones Administrator Guide* (Document Number 16-600944) - These guides provide a description of administrative duties like HTTP server setup, and how to set up Push parameters in the settings file.

IP Telephone WML Server Setup Guide (Document Number 16-300507) - This guide provides information on setting up a Web server.

Avaya one-X™ Deskphone Edition for 9600 Series SIP IP Telephones Developer Guide (Document Number 16-603173) - This guide gives detailed information about how to update the xml files used to customize the 9600 Series SIP IP phones running SIP Software Release 2.2, 2.5 or later.

Customization and Hospitality Features in Avaya 9600 SIP IP Phones (Issue 1, September 2008 for SIP Software Release 2.2 or later issues for Release 2.5 and subsequent releases) - This guide discusses some of the new customization features available. It provides details about the ability to customize new screens, a new push type (phonexml), the ability to change some features via push, the various ways the backlight feature can be controlled in relation to hospitality industry needs, and more.

Other Documents

The Unicode Consortium, The Unicode Standard, Version 3.2, Addison Wesley, 2002.

Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000.

IETF Documents

The following documents provide information relevant to IP telephony and are available for free from the IETF Web site:

IETF RFC 2616: <http://www.ietf.org/rfc/rfc2616.txt?number=2616>

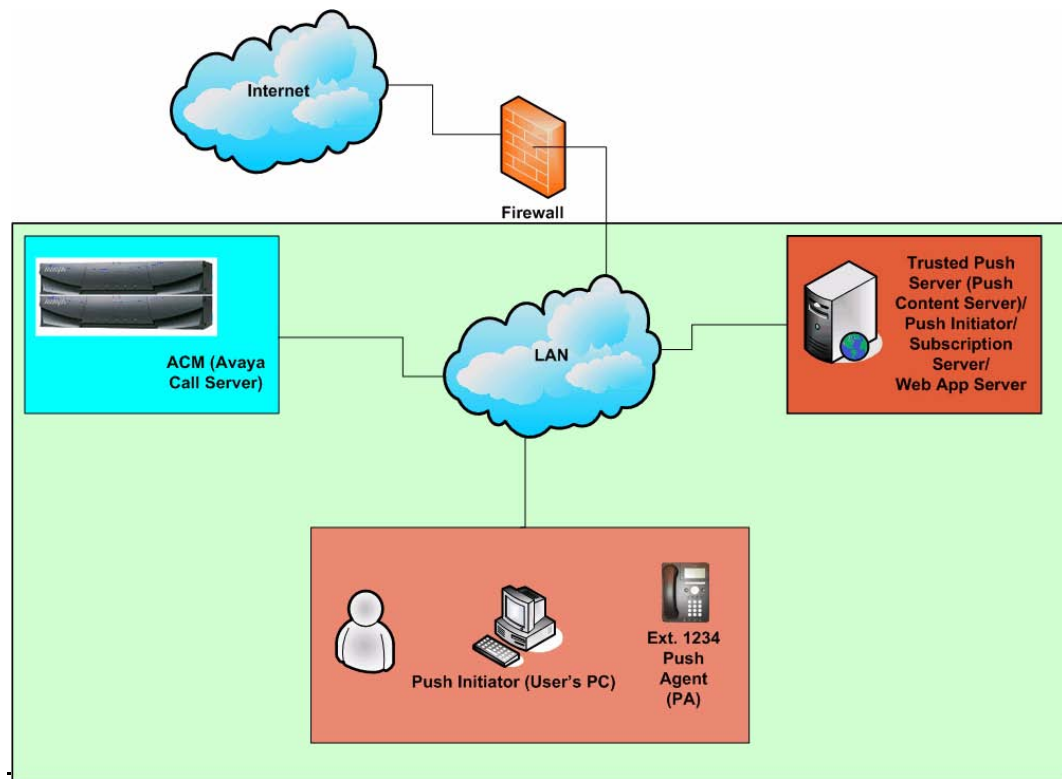
IETF 1945: <http://www.ietf.org/rfc/rfc1945.txt?number=1945>

Chapter 1: IP Telephone Interfaces

Overview

[Figure 1](#) shows a typical system-wide network diagram that includes Avaya IP Telephones, Avaya Communication Manager Servers, and application servers. The application servers include Push Servers, Subscription Servers, and a Web Application Server.

Figure 1: Typical System-Wide Network Topology



With this picture in the current context, enabling a Web server or an application server for a particular enterprise is not an additional entity.

Example:

ABC Company currently has an intranet Web server that serves the company's intranet sites and other employee information. ABC has just deployed a company-wide Avaya IP Telephone system, which offers an optional Web browser application. To set up a Web server for the Avaya IP Telephones, ABC just has to add two MIME types to their existing intranet Web server. ABC does not require an additional server or entity to enable Web functionality on Avaya 9600 Series IP Telephones. Additionally, the same intranet server can be a Push Application server or a Trusted Push Server. Similarly, a subscription server can also be resident on ABC's existing intranet server.

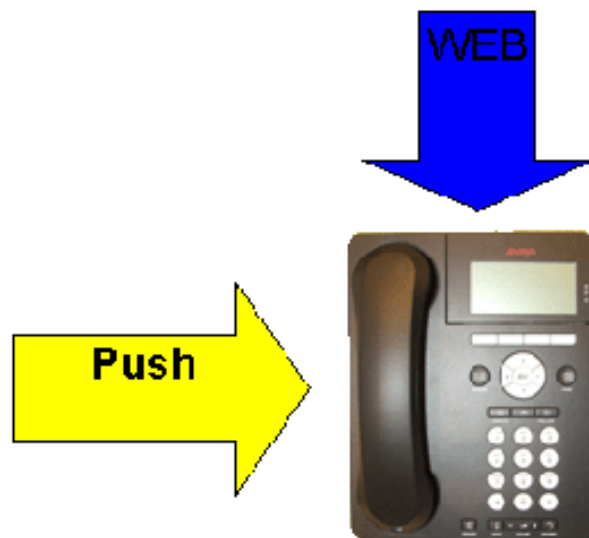
Note:

For more information on setting up MIME types or setting up a Web server, see the **WML Server Setup Guide** (Document Number 16-300507), available for download at: <http://www.avaya.com/support>.

Existing Interfaces

Avaya IP Telephones accept the interfaces shown in [Figure 2](#).

Figure 2: Avaya IP Telephone Interfaces



Push:

This interface allows an application to spontaneously push a message to an IP telephone's display without involving the user. See [Chapter 2: Push Interface Overview](#), [Chapter 3: Creating Push Messages](#), [Chapter 4: Push Administration](#), and [Chapter 5: Troubleshooting the Push Interface](#) for information about the Push interface.

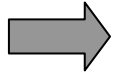
Web:

This is a Web browser Interface. Users can navigate Web applications and retrieve information about the company, news, or interactive applications such as a conference room scheduler and Company Directory lookup. For detailed information about the Web interface see:

- [Chapter 6: About the Web Browser](#),
- [Chapter 7: Developing Web Pages for the Browser](#), and
- [Chapter 8: Web Applications](#).

Chapter 2: Push Interface Overview

Introduction



Unless otherwise indicated, the information in this chapter applies to both H.323 and SIP IP Telephones.

Push is the ability for an application to send content to the Web browser, to the Top Line of the display, to the audio transducers, or set/refresh settings on the 9600 Series IP Telephones.

With the Push interface, the application can “push” unsolicited information to the telephone without the user having to click a link. Some uses of the Push interface can be:

- Broadcasting company news.
- Sending meeting reminders with conference bridge numbers, so that users don't have to search for the conference number.
- Streaming music, such as wake-up alarms in hotel rooms.
- Streaming audio announcements.
- Sending critical stock news information.
- Broadcasting critical emergency notices and weather alerts.
- Building intelligent databases to target information to an individual or groups of phones.
- Customizing the telephone interface.

This chapter provides an overview of the Push interface. [Chapter 3: Creating Push Messages](#) and [Chapter 4: Push Administration](#) provide detailed information on setting up and initiating Push Messages.

Push Feature Description

The Push interface offers several features:

- Full screen pushes, called **Display** push types.
- Single line top area text push, called **Top Line** push types.
- Audio streaming to the telephone, called **Receive Audio** push types. In addition to the unicast audio receive type (available for phones running both SIP and H.323 software), multicast audio receive pushes that facilitate packet distribution are supported, but only for 9600 Series IP Telephones running H.323 Software Release 3.0.

- Audio streaming from the telephone, called **Transmit Audio** push types.
- Refresh telephone resource pushes, called **phonexml** push types. This type of push lets you update customized screens, clear call history (call, web, contacts and user settings), and set settings (e.g. phone language, user display name and more) on the phone. This push type applies only to 9600 Series SIP IP Telephones running SIP Software Release 2.2, 2.5, or later.
- Optional alerts.
- Push priorities.
- A Security mechanism.
- A Subscription service.

A message can be pushed to a properly configured Avaya IP Telephone as a single text line, a full screen/Web page, an audio stream, or specific message to perform a given setting.

Avaya provides a security mechanism to assure that the content pushed to the phones is from a trusted source. Additionally, a subscription service allows the phones to provide necessary information to the application server such that pushes can be targeted to the individual user, a group of users, or to the enterprise. Push Messages have two priorities set by the application and can also be accompanied by an optional notification alert.

Push Architecture

The Push Flow Process

The Push interface uses the following terminology:

- ***Push Initiator***: An application capable of transmitting the Push Message to the Push Agent.
- ***Push Agent***: The telephone software resident on the Avaya IP Telephone that is capable of receiving the Push Message from the Push Initiator. The Push Agent processes the Push Message and requests the Push Content.
- ***Trusted Push Server (TPS)***: A Web server serving the Push Content that conforms to the security settings as established by the TPSLIST parameter in the script file. This can also be an existing Web server within the network, or the same server as the Push Initiator.

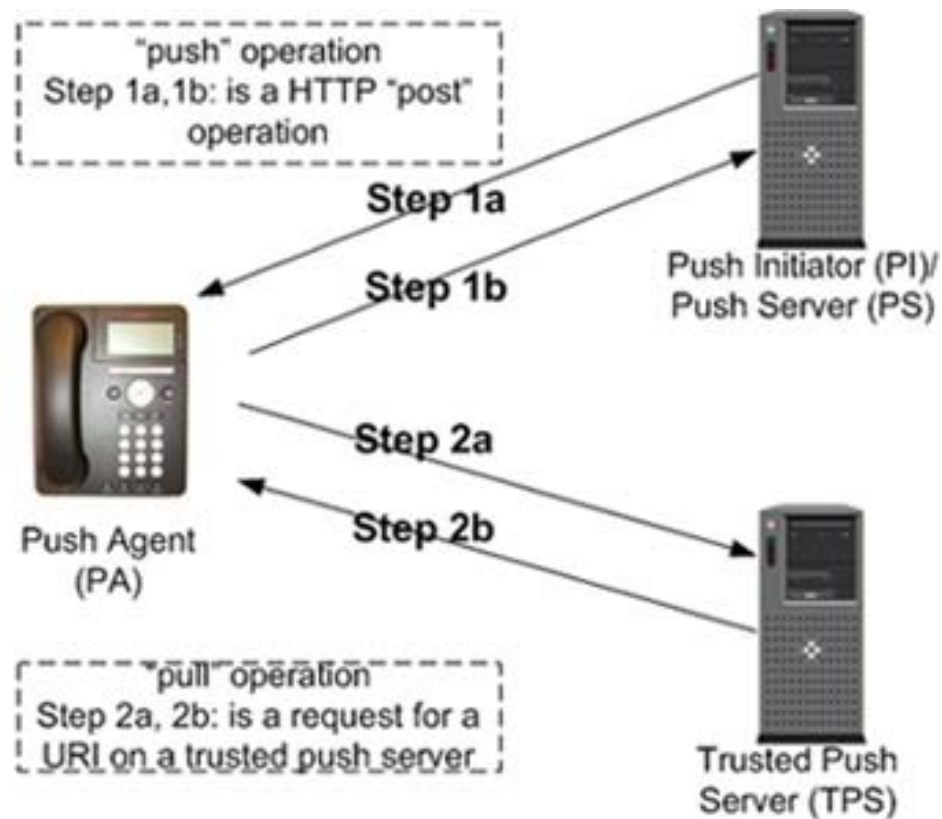
The Push/Pull Process – A Two-Step View

The Push framework is a two-step process - a “push” operation followed by a “pull” operation. See [Figure 3](#) for a visual reference to the steps involved in the Push/Pull process.

Push Operation (Step 1a and Step 1b)

The Push Initiator (PI), which is an application server, transmits a Push Message using the HTTP “POST” method to the phone’s Push Agent (PA).

Figure 3: Push/Pull Operation



Pull Operation (Step 2a and Step 2b)

The phone requests the target URI of the Push Content from a Trusted Push Server. The Push Content can be any valid WML file or an XML file with tags that instruct the endpoint to do one or more of the following:

- Set up an RTP audio stream,
- Display a message on the Top Line,
- Display a full screen message with images and links, or
- Re-subscribe with the subscription server.

Push Message Flow

This section describes the step-by-step process to send a particular Push Message to an Avaya IP Telephone. [Figure 4](#) illustrates this process.

Step 1 - XML File Push Message Sent

The first step of the Push process is to POST a Push Message to the telephone's Push Agent. The Push Message can only be sent using the HTTP POST method. The message contains an XML file with the <Push> tag and instructions for the telephone's Push Agent to request the Push Content from a Trusted Push Server.

Note:

For more information on creating Push Messages, see [Chapter 3: Creating Push Messages](#).

Step 2 – Push Agent Responds

The Push Agent sends a response back to the Push Initiator with HTTP status codes. The response also contains an HTTP header extension called “x-Avaya-Push-Status” code. The x-Avaya-Push-Status indicates the outcome of a push request back to the Push Initiator. x-Avaya-Push-Status codes respond with errors such as Forbidden, Not in Push state, etc.

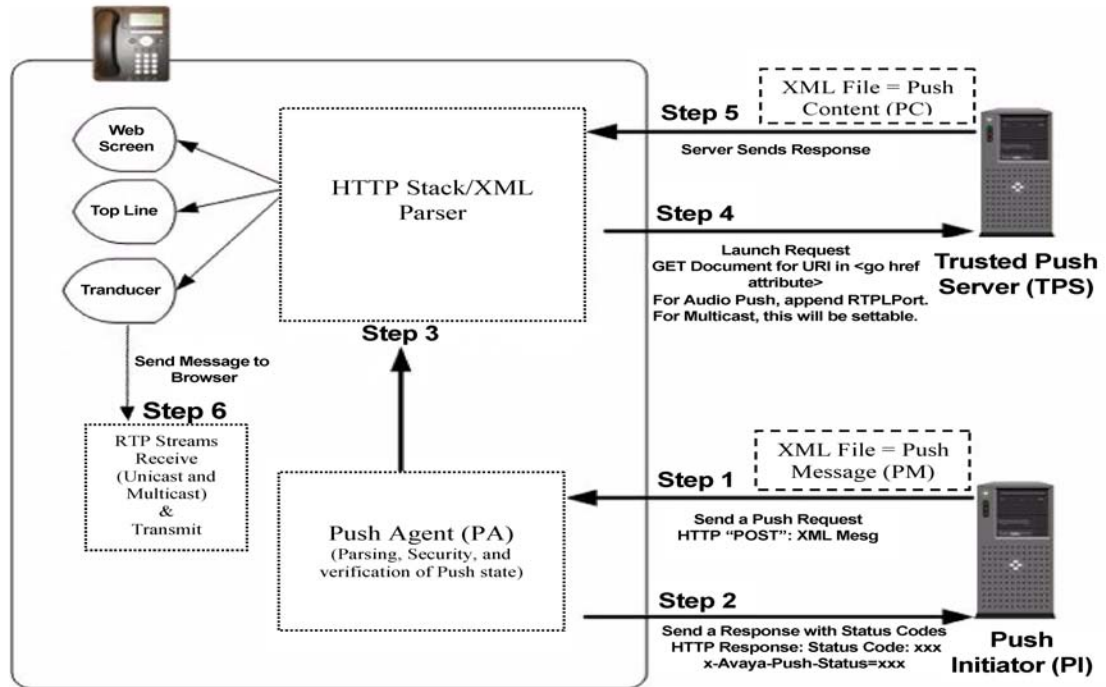
Note:

See [Chapter 3: Creating Push Messages](#) for more information on x-Avaya-Push-Status codes.

Step 3 – XML Message Parsing

The XML parser parses the Push Message and verifies that the Push Content URL is a Trusted Push Server. If the URL is not a Trusted Push Server, an HTTP 403 - Forbidden error message is sent back to the Push Initiator using Step 2 mechanisms.

Figure 4: Push Flow



Step 4 – Request Launched

Once the URL is verified from a Trusted Push Server, the Push Agent launches a request for the URL that is embedded in the <go> tag of the Push Message.

Step 5 – Push Content Server Responds

The Trusted Push Server sends a response back to the telephone with the proper Push Content. The Push Content consists of an XML file or a WML file, based on the Push type. This file is parsed by the XML Parser and the Push Content is extracted and prepared for display. If the response from the Trusted Push Server is not consistent with the value of the type attribute in the <Push> tag sent by the Push Initiator, the push is terminated.

Step 6 – Message Sent to Telephone

Once the telephone's XML parser parses the XML file, depending on the Push type, the telephone either displays or streams the message.

About the Push Agent

The 9600 Series IP Telephones provide an HTTP server in addition to the HTTP client. This allows an application server to “push” a request for:

- The Web browser to get and display a particular Web page.
- The phone's Top Line application to display a Top Line message.
- The phone to receive an audio stream from an application outside the context of a telephone call.

HTTP Server functionality is based on (or is provided by) the GoAhead Web Server 2.1, Copyright© 2004 GoAhead Software, Inc. All Rights Reserved.

Note:

For SIP software Release 2.5, the Push Agent listens only to the port specified in the PUSHPORT parameter for incoming requests.

HTTP Server

Avaya IP Telephones support an HTTP server as specified in the [IETF Documents](#) listed in [Related Documentation](#) for HTTP 1.0, HTTP 1.1, and for an HTTP client. The HTTP server uses TCP as a transport-layer protocol, and supports only one connection (socket) at a time. The HTTP client uses TCP or TLS/SSL over TCP as a transport-layer protocol.

- The Push Agent will activate the receive port 80 for the HTTP server if:
 - the phone is properly registered with a call server, and
 - the TPSLIST contains at least one non-null value, and
 - the URI for a transmitted message begins with *http://*.
- The Push Agent will activate the receive port 443 for the HTTPS server if:
 - the phone is properly registered with a call server, and
 - the TPSLIST contains at least one non-null value, and
 - the URI for a transmitted message begins with *https://*.
- The Push Agent only listens to port 80 for all incoming requests.

Push Agent - HTTP POST Address

The HTTP POST address (URL) for the IP telephone where a Push request is sent to is:

`http://<IP_Address_of_the_telephone>/forms/push`

Where, `<IP_Address_of_the_telephone>` is the IP Address of the telephone where the push is to be sent in the dotted-decimal format.

The Push Agent will process all POST methods received by the phone's HTTP server that contain the above URL. All HTTP POST requests must be sent to this URL only.

All XML messages are sent in the HTTP POST pre-defined variable called **"XMLData."**

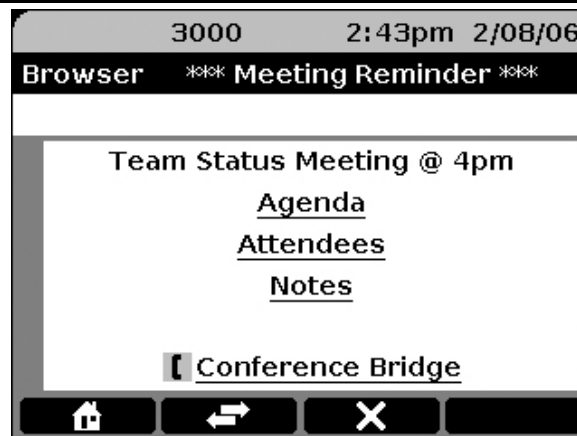
A 403 `Forbidden` error message is sent in response to a POST with an invalid Request-URI.

Push Types

The 9600 Series IP Telephones support the Push types described in this section.

Display push type – Applicable to phones running either H.323 or SIP software. Content can be pushed to the Web browser with an optional alert. The pushed page can access all Web browser features. See [The Display Push Type](#) for more information about this type of push.

Figure 5: Full Screen (Display) Push



Note:

Full screen Display pushes on the 9610 IP Telephone appear differently due to a smaller screen size and a single (Top) Line for messages and prompts.

Top Line push type - Applicable to phones running either H.323 or SIP software. Text can be pushed to the Top Line with an optional alert. Top Line pushed messages and alerts can be displayed even when the Web browser is not in focus. For more information on a Top Line push, see [The Top Line Push Type](#).

Figure 6: Top Line Push



Receive Audio push type - Applicable to phones running either H.323 or SIP software, with exceptions as noted. The phone can receive a unicast or multicast audio stream from an application outside the context of a telephone call. Multicast audio receive pushes are supported only on 9600 Series IP Telephones running H.323 Software Release 3.0 software, while unicast pushes are available for telephones running either SIP or H.323 software. The Receive Audio Push Message can be accompanied with an optional alert. For more information on Receive Audio push, see [The Receive Audio Push Type](#).

Transmit Audio push type - Applicable to phones running either H.323 or SIP software. Transmit Audio pushes allow an end user to send an audio stream through the Trusted Receive Server (TRS). For more information on Transmit Audio push, see [The Transmit Audio Push Type](#).

Subscribe push type - Applicable to phones running either H.323 or SIP software. The Push Subscription Service allows the phones to re-subscribe to the subscription application server with the phone's IP Address, user's extension, Set ID and MAC ID. For more information on the Subscribe push see, [The Subscribe Push Type](#) and [Chapter 4: Push Administration](#).

Phonexml push type – Applies to SIP IP Telephones only running SIP Software Release 2.2, 2.5, or later. The parameters associated with this push type can be used to customize the phone. For example, they can be used to clear the phone history, refresh the phone resources, and change the phone settings. For more information on Phonexml push, see [The Phonexml Push Type \(for R2.2 & R2.5+ SIP IP Telephones only\)](#).

All Push types can be delivered either with a **Normal** priority or with a **Barge** priority on an individual push basis. See the sections on each Push type in [Chapter 3: Creating Push Messages](#) for more information on priorities and states.

The system parameter PUSHCAP (Push Capabilities) is a settings file administrable value used to specify which push types and their modes (priorities) are allowed on the phone. PUSHCAP works differently for H.323 and SIP IP Telephones. See [Chapter 4: Push Administration](#) for more information.

Push Content Requests

When a Push Request is accepted, the HTTP client in the telephone sends a request for the URI specified by the href attribute of the <go> tag in the <Push> tag. The type of request (GET vs. POST) is as specified by the value of the method attribute of the <go> tag in the <Push> tag. The syntax is as specified below, where rtpLPort is the local port that the telephone will use for the associated RTP stream. Postfield data will also be included in the GET or POST request, as specified for WML if any valid <postfield> tags are received in the <Push> tag.

Note:

9600 Series H.323 S3.1 and later releases support substitutions in the value of the href attribute in a <go> element of Push request, as well as in the value of a value attribute in any <postfield> element within the <go> element. If the value of either attribute contains any of the following sub-strings, each sub-string is replaced with the corresponding parameter value before security validation occurs: \$IPADD, \$MACADDR, \$MODEL, and \$PHONEXT.

| Push Type (Push tag's type attribute value) | GET Method Syntax | POST Method Syntax |
|---|-----------------------|--------------------|
| "display" | GET URI | POST URI |
| "top line" | GET URI | POST URI |
| "audio" | GET URI?rtpLPort=xxxx | POST URI |
| "receive" | GET URI?rtpLPort=xxxx | POST URI |
| "transmit" | GET URI?rtpLPort=xxxx | POST URI |
| "subscribe" | GET URI | POST URI |

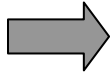
| Push Type (Push tag's type attribute value) | GET Method Syntax | POST Method Syntax |
|--|----------------------|--------------------|
| "phonexml" (for R2.2 & R2.5+ SIP phones only) | GET URI | POST URI |
| "multicast" (for H.323 R3.0+ phones only) | GETURI?rtpLPort=xxxx | POST URI |

Note:

For the POST method, parameters and values are sent in the body of the HTTP request.

Chapter 3: Creating Push Messages

Introduction



Unless otherwise indicated, the information in this chapter applies to both H.323 and SIP IP Telephones.

This chapter covers the details involved in setting up Push Messages for each type of Push:

- Display (full screen) Push
- Top Line (single line) Push
- Receive Audio Push (unicast, and for phones running H.323 Release 3.0 software, multicast)
- Transmit Audio Push
- Subscribe Push
- Phonexml Push (applies only to SIP IP Telephones running SIP Software Release 2.2, 2.5, or greater)

The Display Push Type

The Display push type is a full-screen Push. Details about this type of push are provided in the sections that follow.

Web Browser Features

When using the Display push type, you can use the entire range of features of the Web browser. Some of these features are:

- JPEG and WBMP Images.
- Form controls such as Radio buttons, check boxes, etc.
- Input elements such as text boxes.
- Hyperlinks to a series of other pages with information.
- WTAI features such as click-to-dial and add-to-Contacts.
- Full use of at least two programmable softkeys.
- Capability to push an entire thin-client Web application.

Note:

When the system parameter WMLHOME is null, a display push is allowed, but the Home and Refresh softkeys do not display and the user cannot access any other Web pages.

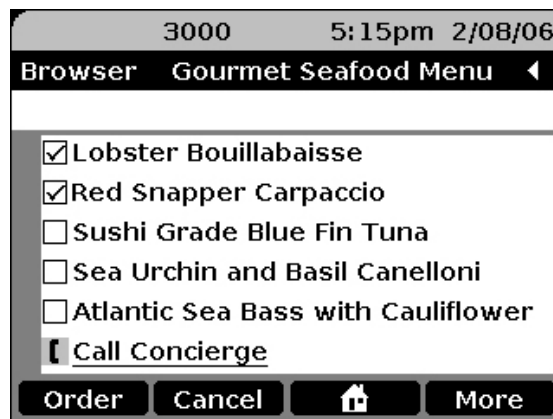
Alerts

A Display Push Message can be sent using alerts. Alerts are a number of ring pings sounded just prior to displaying the message on the screen. With the Display push type, an alert can be sounded with 1, 2, or 3 ring pings. If the **alert** attribute is not associated with the <Push> tag, then no alerts are sounded. Alternatively, if the **alert** attribute is set to "0" no alerts sound.

Display Push Example 1:

Following is an example of the Display push type. Assume that the Push Message (screen) in [Figure 7](#) will be sent to a telephone in a hotel.

Figure 7: Hotel Application Example



Priorities and States

Display Push Content is sent with one of two priorities: **normal** or **barge**. Normal priority push conditions are specified first, followed by barge priority push conditions.

Pushable vs. Non-Pushable States

The following are **non-pushable** states for normal Display pushes to an IP telephone:

- When the user is in text entry mode on a Web text entry screen.
- When the telephone is in the process of restoring a retrieved backup file (H.323 phones only).
- When a Local (Craft) Procedure has been initiated.

- When the telephone is already broadcasting any Transmit Pushed audio content.

If a phone is not in one of the above states, it is considered to be in a **pushable** state, meaning the telephone will accept a pushed message.

Successful Push Response

If the phone is in a pushable state, the Push Agent sends the Push Initiator the following response for all Push request modes and all Push request types:

| Parameter | Status Code | Response |
|----------------------|-------------|-------------------------|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 200 | "Push Message Accepted" |

Normal Priority

When the **mode** attribute in the <Push> tag is set to **normal**, the telephone state for the Display push type is as follows:

- When the telephone is in any of the **non-pushable** states, the screen currently displayed continues to be displayed (i.e., the Display push is **rejected**).
- If the telephone is in the **pushable** state, the telephone **accepts** the Push Message and generates appropriate notification tone(s).

A normal push is accepted when the telephone is in text edit mode in applications other than the Web. If a non-Web application, such as Contacts, is in text-edit mode, then *x-Push-Status* 204 "Not in Push State: Push Accepted" is sent:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|------------------------------------|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 204 | "Not in Push State: Push Accepted" |

Use Case Scenario:

Q. What happens when a user is editing or adding an entry on the Contacts screen and a **normal** priority Display push Message is sent?

A. If an alert is associated with this message, then the alert sounds first. The pushed content is loaded in the background. The message is not displayed until the user elects to view it by clicking the Web tab to launch the Web browser.

If the Web is in “text edit mode” or is transmitting audio, a subsequent normal push is rejected and x-Push-Status 205 “Not in Push State: Push Aborted” is sent. The Push Request does not proceed.

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|-----------------------------------|
| HTTP Status Code | 200 | “OK” |
| <i>x-Push-Status</i> | 205 | “Not in Push State: Push Aborted” |

Note:

If a normal Display push is denied, then the entire Display push is denied, including the Web page’s title. Hence, the application writer might choose to send two pushes - a Top Line push, followed by a Display push. Sending two messages maximizes the chance of the user viewing at least one message.

Barge Priority

When the **mode** attribute in the <Push> tag is set to **barge**, the Display Push Content is accepted as a priority message, with three exceptions. A **barge** Display push is rejected only when the telephone is in any of these **non-pushable** states:

- When the telephone is in the process of restoring a retrieved backup file (H.323 phones only).
- When a Local (Craft) Procedure has been initiated. See the applicable (SIP or H.323) *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Installation and Maintenance Guide* on the <http://www.avaya.com/support> Web site for more details on Craft procedures.

Note:

When the telephone is already broadcasting Transmit Push audio content an incoming display is not shown. Although transparent to the end user, the incoming message is received but waits under the current transmit audio push until the current push completes.

In either non-pushable state, the **barge** content, including any notification tones is discarded. In all other cases, the telephone must accept the **barge** request.

Use Case Scenario:

Q. What happens when a user is editing or adding an entry on the Contacts screen and a **barge** priority Display push message is sent?

A. If an alert is associated with this message, then the alert will first sound. Then the pushed message displays immediately. The user can leave the displayed Web page by pressing the telephone's **Phone** button.

When receiving Push Content with a barge priority, the state of the telephone is as if the user had selected the Web Access tab on the Phone Screen. For example, any incomplete task, such as restoring a retrieved backup file or performing a local procedure, is considered as having been interrupted. Additionally, the user can leave the displayed Web page by pressing the **Phone** button.

Display Push XML Messages

This section describes how to send a Display push using XML messages.

Display Push Message (PM)

To send a Display push, an application must send an **HTTP POST** request to the Push Agent in the telephone.

The format of the XML Message (PM) sent from the Push Initiator to the Push Agent is as follows:

```
<?xml version="1.0"?>

  <Push
    alert="0|1|2|3"
    type="display"
    mode="normal|barge"
  >
  <go href="http://trusted_push_server/filename.wml" method="get|post">
    <postfield name="name1" value="value1"/>
    <postfield name="name2" value="value2"/>
  </go>
</Push>
```

Using the <postfield> Tag

The Web browser interface supports the <postfield> tag. The <postfield> tag allows an application to set a name/value pair that can be sent to the source of the request. The name is set by the **name** attribute and must be a valid WML variable name. The value is set by the **value** attribute. A <go> element can contain one or more <postfield> tags. Postfield tags must be sent if HTTP POST method is used.

Note:

For more information on the <postfield> tag, see [Chapter 7: Developing Web Pages for the Browser](#) and [Chapter 8: Web Applications](#).

Table 1: Description of Elements and Attributes used in the Display Push XML Message

| Element or Tag | Attribute | Description |
|----------------|-----------------------------|--|
| <Push> | | Each Push Message must contain one valid root <Push> tag. |
| | alert=0 1 2 3 | Optional notification alerts – number of ring pings. |
| | type | Push Content = display . |
| | mode | normal or barge priority. |
| | <go href=.../> | A fully qualified URL - to a valid WML file for Display pushes. Cannot exceed 1024 characters. |
| | method | HTTP get or post methods for Push Content (i.e. GET URI; POST URI). |

Display Push Example 2:

Using our previous hotel example, the hotel is ready to serve lunch and wants to send the lunch menu directly to each room's telephone display screen, including sounding an alert to get the guest's attention. The XML payload sent as part of the Push Message is as follows:

```
<!-- Following is the XML Push Request Message sent as a POST request
embedded as part of form data -->
XMLData = <?xml version="1.0"?>
<Push alert="2" type="display" mode="normal">
<go href="http://trusted_push_server/lunch_menu.wml"
      method="get">
</go>
</Push>
<!-- The above message is part of the form data (XMLData) being sent
in Step 1 request -->
```

Push Agent

Once a Push Message is received from the Push Initiator, the Push Agent first parses the XML file for validation and tags mismatch errors. Then the Push Agent verifies that the URL in the <go> tag is part of the Trusted Push Servers. The Push Agent then requests the Push Content from the Trusted Push Server using the URL.

Note:

For more information on Trusted Push Servers, see [Chapter 4: Push Administration](#).

Push Content (PC)

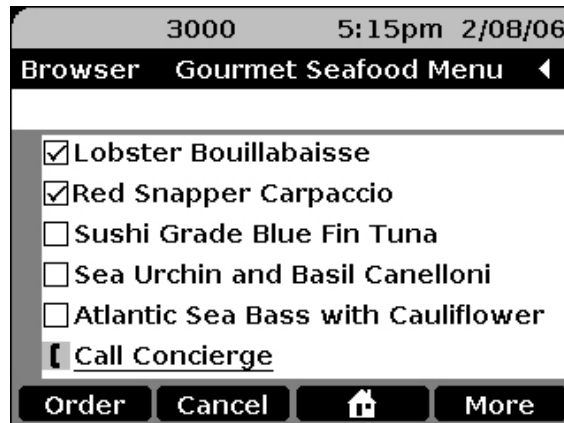
The Display push type's Push Content has to be a WML file. This WML file can contain any of the Web browser elements and features. See [Chapter 7: Developing Web Pages for the Browser](#) and [Chapter 8: Web Applications](#) for more details on elements and tags the Avaya IP Telephones support.

The outline of the “lunch_menu.wml” file (PC) from the Hotel example is as follows:

```
<!-- Server Sends Response - Push Content (PC) - File in the <Push...<go
href Url> -->
<?xml version="1.0"?>
<wml>
<card id="lunch" title="Gourmet Seafood Menu">
<p>
<select name="selection" multiple="true">
    <option value="bouillabaisse">Lobster Bouillabaisse</option>
    <option value="carpaccio">Red Snapper Carpaccio</option>
    <option value="tuna">Sushi Grade Blue Fin Tuna</option>
    <option value="canelloni">Sea Urchin and Basil
Canelloni</option>
    <option value="bass">Atlantic Sea Bass with
Cauliflower</option>
</select>
<a href="wtai://wp/mc;5551212">Call Concierge</a>
</p>
<do type="accept" name="submit" label="Order">
<go href="submit_form.php" method="get"/>
</do>
<do type="prev" label="Cancel">
<prev/>
</do>
<do type="accept" name="home" label="Home">
<go href="home.wml"/>
</do>
<do type="accept" name="help" label="More">
<go href="help.wml"/>
</do>
</card>
</wml>
```


The XML parser parses the WML file. Depending on the priorities and state of the telephone, the Push Content displays as shown in [Figure 8](#), with an alert of two ring pings.

Figure 8: Hotel Lunch Menu Display



Note:

Invalid WML Display push error messages are not displayed to the user.

The Top Line Push Type

Use the Top Line push when you only need to send a single-line text message.

Text Message Features

Some of the Top Line push features are:

- A single-line, alternating text message.
- Message displays, even if Web browser is not in focus.
- Supports UTF-8[1], ISO-888991[1], and Latin1[1] character encodings.

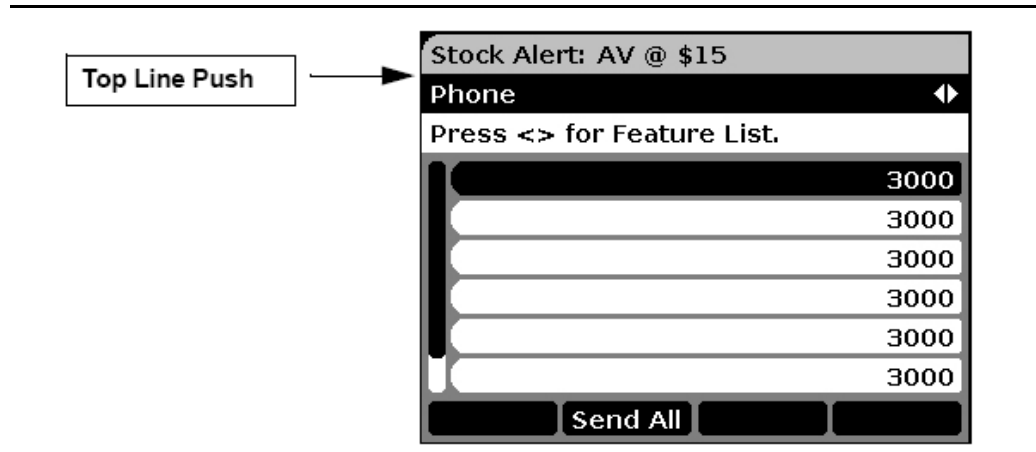
Alerts

A Top Line Push Message can be sent with alerts. Alerts are number of ring pings sounded just prior to displaying the message on the screen. For a Top Line push type, an alert can be sounded with 1, 2, or 3 ring pings. If the **alert** attribute is not associated with the <Push> tag, then no alerts are sounded. Alternatively, if the **alert** attribute is set to "0" no alerts sound.

Top Line Push Example 1:

The following Stock Alert Top Line Push Message is sent to a telephone to alert the user when a stock, AV in this example, reaches \$15 price target. [Figure 9](#) shows the pushed message that displays.

Figure 9: Stock Alert Example



Priorities and States

Top Line Push Content is sent with one of two priorities: **normal** or **barge**. Normal priority push conditions are specified first, followed by barge priority push conditions.

Pushable vs. Non-Pushable States

The following are **non-pushable** states for Top Line pushes to an IP telephone:

- When the telephone is in the process of restoring a retrieved backup file (H.323 phones only).
- When a Local (Craft) Procedure has been initiated. See the applicable SIP or H.323 *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Installation and Maintenance Guide* on the <http://www.avaya.com/support> Web site for more details on local (Craft) procedures.
- When the telephone is in any text entry mode (for normal priority).
- When a Top Line message cannot be displayed due to call-related messages taking Top Line precedence.
- When the telephone is already broadcasting any Transmit Push audio content.

If a phone is not in one of the above states, it is considered to be in a **pushable** state, meaning the telephone will accept a Pushed Message.

Successful Push Response

If the phone is in a pushable state, the Push Agent sends the Push Initiator the following response for all Push request modes and all Push request types:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|-------------------------|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 200 | "Push Message Accepted" |

Normal Priority

When the **mode** attribute in the <Push> tag is set to **normal**, the telephone state for the Top Line push type is as follows:

- If the Top Line is being used for system messages such as application help messages, then the text string is buffered until the higher-priority message is complete.
- If the phone is in a **non-pushable** state, meaning the Push Message cannot be displayed on the top display line, the Push Agent sends the Push Initiator the following responses for a **normal** priority Top Line push request. The Push request does not proceed to request Push Content.

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|-----------------------------------|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 205 | "Not In Push State: Push Aborted" |

If a text string is pushed while the top display line is displaying an earlier pushed text string, then the new text string replaces the previous one as a **normal** priority.

Note:

The expiration time on all pushed text strings is 30 seconds. All Top Line messages are discarded after 30 seconds.

Barge Priority

When the **mode** attribute in the <Push> tag is set to **barge**, the Top Line Push Content is accepted as a priority message. However, a **barge** Top Line push is rejected when the telephone is in one of these **non-pushable** states:

- When the telephone is in the process of restoring a retrieved backup file.
- When a Local (Craft) Procedure has been initiated. See the applicable SIP or H.323 *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Installation and Maintenance Guide* on the Avaya support Web site for more details on local (Craft) procedures.
- When the telephone is already broadcasting any Transmit Push audio content.

In either non-pushable state, the **barge** content, including any notification tones, is discarded. In all other cases, the telephone must accept the **barge** request and display the barge Top Line message immediately.

Top Line Push XML Messages

This section describes how to send a Top Line push with XML messages. Use the Stock Alert Example in [Figure 9](#) as a reference.

Top Line Push Message (PM)

The first step in sending a Top Line push is to send an HTTP POST request from the Push Initiator to the telephone's Push Agent. The XML Message (PM) sent from the Push Initiator to the Push Agent is as follows:

```
<?xml version="1.0"?>
<Push
  alert="0|1|2|3"
  type="Top Line"
  mode="normal|barge"
>
<go href="http://trusted_push_server/filename.xml" method="get|post">
  <postfield name="name1" value="value1"/>
  <postfield name="name2" value="value2"/>
</go>
</Push>
```

Using the <postfield> Tag

The Web browser interface supports the <postfield> tag. The <postfield> tag allows an application to set a name/value pair that can be sent to the source of the request. The name is set by the **name** attribute and must be a valid WML variable name. The value is set by the **value** attribute. A <go> element can contain one or more <postfield> tags. Postfield tags must be sent if HTTP POST method is used.

Note:

For more information on the <postfield> tag, see [Chapter 7: Developing Web Pages for the Browser](#).

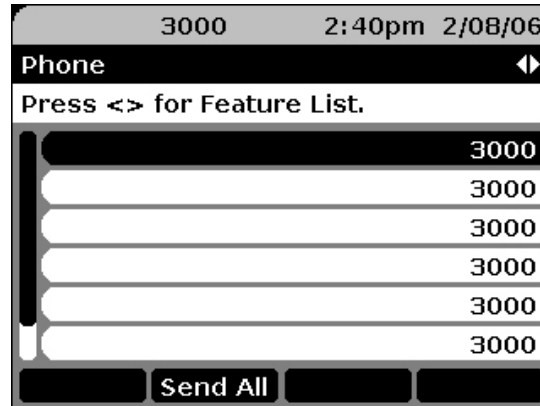
Table 2: Description of Elements and Attributes used in the Top Line Push XML Message

| Element or Tag | Attribute | Description |
|----------------|-----------------------------|--|
| <Push> | | Each Push Message must contain one valid root <Push> tag. |
| | alert=0 1 2 3 | Optional notification alerts – number of ring pings. |
| | type | Push Content = Top Line . |
| | mode | normal or barge priority. |
| | <go href=.../> | A fully qualified URL - to a valid XML Push Content file for Top Line pushes. Cannot exceed 1024 characters. |
| | method | HTTP get or post methods (i.e. GET URI; POST URI). |

Top Line Push Example 2:

Using our previous stock alert example, the price of the AV is reaching the \$15 price target. Now, the stock broker's telephone display shows:

Figure 10: Telephone Display Prior to Receiving Stock Alert Message



The code excerpt associated with the Stock Alert example to be sent as part of the Push Message is as follows:

```
<!-- Following is the XML Push Request Message sent as a POST request
embedded as part of form data -->
XMLData = <?xml version="1.0"?>
<Push alert="3" type="Top Line" mode="normal">
<go href="http://trusted_push_server/stock_alert.xml"
      method="get">

</go>
</Push>
<!-- The above message is part of the form data (XMLData) being sent
in Step 1 request -->
```

Push Agent

Once a Push Message is received from the Push Initiator, the Push Agent first parses the XML file for validation and tag mismatch errors. Then the Push Agent verifies that the URL in the <go> tag is part of the Trusted Push Servers.

Note:

For more information on Trusted Push Servers, see [Chapter 4: Push Administration](#).

Then the Push Agent requests the Push Content from the Trusted Push Server using the URL.

Top Line Push Content (PC)

The Top Line push type's Push Content has to be an XML file that contains a valid <Response> tag that contains a valid <Topline> tag. For 9600 Series SIP IP Telephones running SIP software Release 2.2 or greater, Avaya recommends that Top Line push message strings contain 28 or fewer characters; strings longer than 56 characters will be truncated.

The following is the code excerpt associated with the Stock Alert [Top Line Push Example 2](#): to be sent as part of the Push Content:

```
<!-- Server Sends Response - Push Content (PC) - File in the <Push...<go
href Url> -->
<?xml version="1.0"?>
<Response>
  <Top Line>
    Stock Alert: AV @ $15
  </Top Line>
</Response>
```

Using the <Response> tag

The <Response> tag for the Top Line push type must be the root element in the (PC) XML file.

Using the <Topline> tag

The <Topline> tag consists of the actual text message to display on the telephone's Top Line.

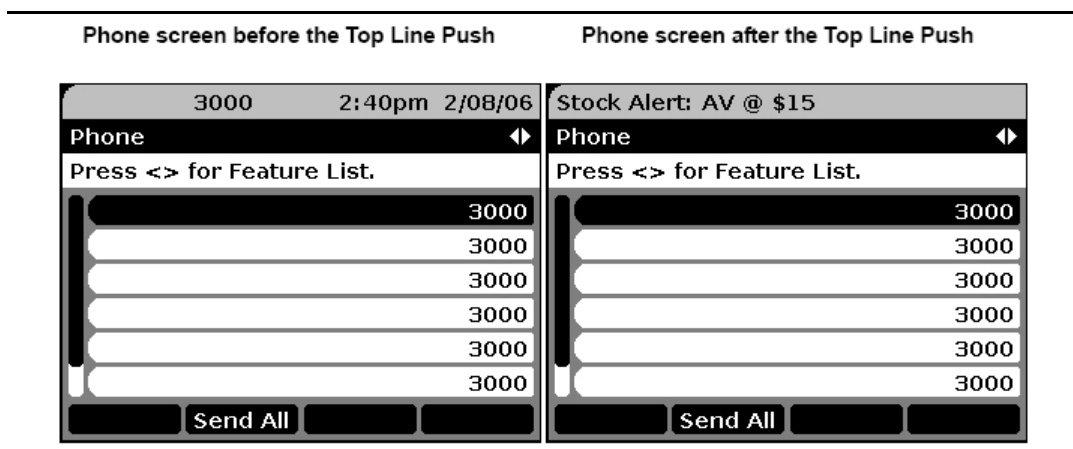
If the length of the message exceeds the given pixels, the entire message is divided. The message then fits on a single line, with the initial text and the remaining text alternating on the top display line.

A Top Line message can be approximately 320 pixels of usable space, depending on the width of each character (average 4-6 pixels wide per character) for all 9600 IP Telephones except the 9610. A single 9610 Top Line message can be approximately 160 pixels in length. When the text is longer than the allowable space, the message displays in two parts that alternate every few seconds, based on the display interval setting. This means that for all telephones except the 9610, there are up to 640 pixels (320x2) and for the 9610 there are up to 320 pixels (160x2).

The text within the <Top Line> tag can consist of different character encodings such as UTF-8, ISO-888991, and Latin1. See the [IETF Documents](#) listed under [Related Documentation](#) for information on character encoding.

The telephone's XML parser parses the XML file. Depending on the priorities and state of the telephone, the Push Content displays as shown in [Figure 11](#) with an alert ring ping of 3.

Figure 11: Stock Alert Message



The Receive Audio Push Type

Use a Receive Audio push when you need to stream a particular audio message to the telephone. The Receive Audio push is the ability to transmit RTP streams to the endpoint. This Push type provides additional capabilities such as start and stop to control audio streams. The RTP stream can also notify the phone that a stream has ended.

To the user, the way the telephone handles pushed audio content mirrors the way it handles a call. The user can switch between Speaker, Handset, and Headset as desired, adjust volume, and view LED states as with a call. The user can terminate the audio push by going on-hook, selecting a call appearance, or, if on a Web page with a "click to dial" link by selecting that link.

While the telephone is playing back pushed audio content, the call server can independently send messages that require the telephone to generate tones, for example, to alert the user to an incoming call. If the telephone is alerting while receiving an audio push and the user listens to the pushed message, alerting mutes and the call proceeds to coverage or can be answered when the push ends. The **Volume Up** and **Volume Down** buttons raise and lower the volume if the user presses them while listening to pushed audio content. The volume level is remembered for subsequent Receive Audio pushes, but has no effect on the actual microphone level for normal telephone calls.

When a Receive Audio Push is being received, the telephone may or may not be in a "pushable" state to receive a Transmit Audio push, depending on the priority of the Transmit Audio push. [Table 3](#) describes the effect of a Transmit Audio push on a Receive Audio push.

Notes:

Transmit Audio pushes allow an end user to send a non call-associated audio message. See [The Transmit Audio Push Type](#) for information on Transmit Audio pushes.

For SIP Software Release 2.2 and later, set the parameter SET_SYMMETRIC_RTP to 0 to ensure the Receive Audio Push type works successfully. This ensures that the port that the server uses to stream audio to the phone is the same as the port used by the phone to receive audio. If different ports are used, the audio will not be heard.

Interrupt Screens

Interrupt screens automatically accompany standalone Receive Audio pushes, meaning those audio pushes not accompanied by a display or other type of push. An interrupt screen has a title line, Prompt line, and Application area content. The content area provides visual information about the audio stream, for example, how to end the audio stream and return to the previous activity.

Interrupt screens have priority over normal display pushes, but barge-in pushes may have priority over interrupt screens.

Receive Audio Push Features

The Receive Audio push type provides the following features:

- Telephone connection to an incoming RTP stream
- A “pure” stream, not a telephone call
- Temporarily replacement of the active stream
- Audible alerts
- A built-in timer to display the stream for given period of time
- Multicast receive audio pushes are supported, but only for 9600 Series IP Telephones running H.323 Release 3.0 software. A multicast push prevents unnecessary packet duplication and allows the network to replicate the stream for as many users as needed. Multicast pushes are ignored by the all 46xx and all 96xx phones that do not support S3.0 H.323 software.

Alerts

A Receive Audio Push Message can be sent with accompanying alerts. Alerts are a number of ring pings sounded just prior to displaying the message on the screen. With a Receive Audio push type, an alert can be sounded with 1, 2, or 3 ring pings. If the **alert** attribute is not associated with the <Push> tag, then no alerts are sounded. Alternatively, if the **alert** attribute is set to “0” no alerts sound.

Receive Audio Push Example:

A Receive Audio push example could be a hotel wake-up message a guest schedules from the room. The guest can schedule the alarm directly from an existing Web application loaded to the IP telephone. The next morning, the application sounds the alarm using RTP streaming directly to the telephone. As an added touch, a concurrent Display push can send the Hotel Breakfast menu so customer can order breakfast directly from the telephone.

Priorities and States

Received Audio Push Content is sent with one of two priorities: **normal** or **barge**. Normal priority push conditions are specified first, followed by barge priority push conditions.

[Table 3](#) shows the result of receiving another Receive Audio push or a Transmit Audio push while a Receive Audio push is playing, depending on the push priorities.

Table 3: Transmit/Receive Audio Push Priorities

| This Audio Push Type | With this Priority | Action | This Audio Push Type | With this Priority |
|----------------------|--------------------|--------------------|----------------------|--------------------|
| Receive | Normal | DOES NOT interrupt | Receive | Normal |
| Receive | Barge-in | DOES interrupt | Receive | Normal |
| Receive | Barge-in | DOES interrupt | Receive | Barge-in |
| Transmit | Normal | DOES NOT interrupt | Receive | Normal |
| Transmit | Barge-in | DOES interrupt | Receive | Normal |
| Transmit | Barge-in | DOES interrupt | Receive | Barge-in |

Pushable vs. Non-Pushable States

The following are **non-pushable** states for an IP telephone with the normal Receive Audio push type.

- Any call appearance is alerting an incoming call.
- Any call appearance is active.
- The telephone is restoring a retrieved backup file.
- A Local (Craft) Procedure has been initiated. See the applicable SIP or H.323 *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Installation and Maintenance Guide* on the <http://www.avaya.com/support> Web site for more details on local (Craft) maintenance procedures.

- The telephone is already broadcasting any Transmit Audio pushed content.

If a phone is not in one of the above states, it is considered to be in a **pushable** state, meaning the telephone will accept a Pushed Message.

Successful Push Response

If the phone is in a pushable state, the Push Agent sends the Push Initiator the following response for all Push request modes and all Push request types:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|-------------------------|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 200 | "Push Message Accepted" |

Normal Priority

When the **mode** attribute in the <Push> tag is set to **normal**, the telephone state for the Receive Audio push type must be **pushable**. If the telephone is in the **pushable** state, then the telephone accepts the Receive Audio push. The telephone broadcasts the pushed audio stream through the currently active audio device - the Speaker, Handset, Headset, or Bluetooth headset. Once the pushed audio stream is transmitted to the user, the user can redirect it, terminate it, etc.

A Receive Audio push with a normal priority will not stream if the phone is in a **non-pushable** state. When the telephone is in a **non-pushable** state, any current audio activity continues without interruption or user notification and the Audio push stream is rejected. Error message 208 is sent and the Push Request does not proceed.

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|---|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 208 | "Not in Audio Push State: Push Aborted" |

With support for multicast pushes in software Release 3.0 (for H.323 9600 Series IP Telephones only), additional Status Codes may apply to multicast pushes and cause a non-pushable state. They are as follows:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|--|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 310 | "310 Multicast XML Failure: No RTP listening port specified" |
| <i>x-Push-Status</i> | 311 | "311 Multicast XML Failure: No Multicast IP address specified" |
| <i>x-Push-Status</i> | 312 | "312 XML Failure: RTP listening port specified is less then 10001" |
| <i>x-Push-Status</i> | 313 | "313 XML Failure: RTP listening port specified is greater then 65535" |
| <i>x-Push-Status</i> | 314 | "314 Multicast XML Failure: Multicast IP address is not in the expected range of 239.0.0.0 to 239.255.255" |

Additionally, the phone reserves the rtpLPort specified in the Push Message. If the port is not available, then the following x-Push-Status error is returned:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|---|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 209 | "209 Not in Audio Push State: RTP listening port not available" |

Barge Priority

When the **mode** attribute in the <Push> tag is set to **barge**, the Receive Audio Push Content is accepted as a priority message. However, a **barge** Audio push is rejected when the telephone is in one of these non-pushable states:

- When the telephone is in the process of restoring a retrieved backup file.
- When a Local (Craft) Procedure has been initiated (For more information, see the applicable SIP or H.323 *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Installation and Maintenance Guide* on <http://www.avaya.com/support>).

In either non-pushable case, the **barge** audio content, including any notification tones, is discarded and the RTP socket is closed. In all other cases the telephone accepts the barge audio request regardless of any user activity and broadcasts the audio push content through the Speaker.

Note:

A barge Receive Audio push will interrupt and play while the telephone is broadcasting received pushed audio content.

Use Case Scenario:

Q. What happens when a user is on the call and a **barge** Receive Audio Push Message is sent?

A. If a user is on the call and a barge Audio Push Message is sent, the current active call is placed on [local] Hold, and the audio is streamed immediately. If the call is still on Hold when the pushed audio stream ends, the user is reconnected to the call. Note that the far-end party does not get any indication that the current call is being placed on hold.

Receive Audio Push XML Messages

This section describes how to send an Audio push with XML messages.

Receive Audio Push Message (PM)

The first step in sending a Receive Audio push is to send an HTTP POST request from the Push Initiator to the telephone's Push Agent. Following is the XML Message (PM) format sent from the Push Initiator to the Push Agent:

```
<?xml version="1.0"?>
<Push
  alert="0|1|2|3"
  type="audio"
  mode="normal|barge"
>
  <go href="http://trusted_push_server/filename.xml"
  method="get|post">
    <postfield name="name1" value="value1"/>
    <postfield name="name2" value="value2"/>
  </go>
</Push>
```

Receive Multicast Audio Push Message (available only for 9600 Series IP Telephones running H.323 Software Release 3.0 or greater)

The first step in sending a Receive Multicast Audio push is to send an HTTP Post request from the Push Initiator to the telephone's Push agent. Following is the XML Message (PM) format sent from the Push Initiator to the Push Agent:

```

    <?xml version="1.0"?>
    <Push
      alert="0|1|2|3"
      type="multicast"
      mode="normal|barge"
    >
      <set name="rtpLPort" value="xxxx"/> /*set the listening port on the
phone*/
      <set name="multicastip" value="x.x.x.x"/> /*set the multicast ip*/
      <go href="http://trusted_push_server/filename.xml"
method="get|post">
        <postfield name="name1" value="value1"/>
        <postfield name="name2" value="value2"/>
      </go>
    </Push>

```

Using the <postfield> Tag

The Web browser interface supports the <postfield> tag. The <postfield> tag allows an application to set a name/value pair that can be sent to the source of the request. The name is set by the **name** attribute and must be a valid WML variable name. The value is set by the **value** attribute. A <go> element can contain one or more <postfield> tags. Postfield tags must be sent if the HTTP POST method is used.

Note:

For more information on the <postfield> tag, see [Chapter 7: Developing Web Pages for the Browser](#).

Table 4: Description of Elements and Attributes used in the Receive Audio Push XML Message

| Element or Tag | Attribute | Description |
|----------------|---------------|--|
| <Push> | | Each Push Message must contain one valid root <Push> tag. |
| | alert=0 1 2 3 | Optional notification alerts – number of ring pings. the default value is "0." |
| | Type | Push Content = receive or audio or multicast . |
| | Mode | normal or barge priority. |

| Element or Tag | Attribute | Description |
|----------------|----------------|---|
| | Set | Applicable only when type=multicast. A multicast push will be rejected with the appropriate response error codes if invalid values are specified for either of the following set attributes. Multicast pushes apply only to 9600 Series IP Telephones running H.323 Software Release 3.0 or greater. rtpLPort – the rtp listening port on the phone; the valid port range is 10001 to 65535. multicastip –Specifies to which multicast group or IP address the phone should subscribe; the valid range is 239.0.0.0 to 239.255.255.255. |
| | <go href=.../> | A fully qualified URL - to a valid XML file for Audio push Content. Cannot exceed 1024 characters. |
| | method | HTTP get or post methods. |

RTP Port

An RTP port is needed to stream audio to the telephone. The Push Agent sends the port information to the Trusted Push Server in the GET string. The GET string contains the variable “rtpLPort,” the telephone’s local port to be used for audio streaming.

The Push Agent responds to the server with:

| Push Message Type | Syntax |
|-------------------------------------|-----------------------------------|
| “receive” or “audio” or “multicast” | GET URI?rtpLPort=xxxx POST URI |

The RTP Local port information sent by the Push Agent to the Push Initiator from the GET request is as follows:

| GET Parameter | Description |
|---------------|---|
| rtplPort | RTP Local port that will be used for streaming. |

Note:

The rtplPort is generated dynamically and might be different for every new Audio Stream Push session. The telephone will play the audio stream only on this specified port. The rtp stream is checked while playing to verify that it is actually being streamed from the "remote_ip_address".

The code excerpt associated with [Receive Audio Push Example](#): (the hotel wake-up message), which will be sent as part of the Push Message, is as follows:

```
<!-- Following is the XML Push Request Message sent as a POST request
embedded as part of form data -->
XMLData = <?xml version="1.0"?>
    <Push alert="3" type="audio" mode="barge">
        <go href="http://trusted_push_server/wake_up.xml" method="get">
        </go>
    </Push>
<!-- The above message is part of the form data (XMLData) being sent
in Step 1 request -->
```

Push Agent

Once a Push Message is received from the Push Initiator, the Push Agent first parses the XML file for validation and tag mismatch errors. Then the Push Agent verifies that the URL in the <go> tag is part of the Trusted Push Servers. The Push Agent then requests the Push Content from the Trusted Push Server using the URL.

Note:

For more information on Trusted Push Servers, see [Security](#).

Receive and Multicast Receive Push Content (PC)

Receive type pushes (including multicast type pushes applicable to phones running H.323 Software Release 3.0) must be a valid XML file that contains a valid <Response> tag. The valid <Response> tag must contain a valid <Audio> tag, and the <Audio> tag contain a <Url> tag with an href attribute that begins with "RTPRx://".

Receive Audio Push Content (PC)

A special XML file for RTP streaming must initiate the Receive Audio push. To be considered valid, the xml file must contain a valid <Response> tag that contains a valid <Audio> tag.

```
<?xml version="1.0"?>
<Response>
  <Audio
    packetsize="10|20|30|40|50|60"
    codec = "PCMU | PCMA"
  >
    <AudioTimer value="30"/>
    <Url href="RTPRx://remote_ip_address:remote_port"/>
    <Promptline>
      This text goes on the Prompt Line
    </Promptline>
  </Audio>
</Response>
```

Note:

For multicast receive audio pushes, the rtp stream is checked while playing to ensure that it is actually being streamed from the "remote_ip_address" indicated in the href attribute. Thus, developers must take the precaution of specifying the correct IP address in the XML file.

Each <Response> can contain only one <Audio> tag with the following attributes:

| Attribute | Value | Description |
|--------------|-------------------------------------|--|
| codec | PCMU PCMA | Silence suppression on. G.711 Annex A (no CID frames) μ -law A-law Default is PCMU. |
| packetsize | 10 20 30 40 50 60 (milliseconds) | Default is 40 milliseconds. |
| <Promptline> | | Provides the prompt line text to accompany the Audio Interrupt Screens. |

Each <Audio> tag must contain one valid <Url> tag and may contain one <AudioTimer> tag and/or one <Promptline> tag. < The <AudioTimer> tag is used as an inter-packet timer. This timer is set every time a packet is received. After an administrable duration where no packets are received the RTP stream is terminated. This tag has the following attributes:

| Attribute | Value | Description |
|--------------|--------------------|--|
| value | <i>X (seconds)</i> | Default is 20 seconds. The range is 5 to 30 seconds. |

Each <Url> tag consists of information for the RTP streaming server and the local receive port of the telephone. If the <Url> tag has an unspecified format, the push will be terminated. A valid <Url> tag has the following attributes:

| Attribute | Value | Description |
|-------------|-------------------|---|
| href | <i>URI string</i> | URI to specify or control the RTP stream. |

| RTP Streaming URI Format | IP Address | Port |
|--------------------------|-------------------------------------|--|
| RTPRx://URI:port | The URI of the Trusted Push Server. | RTPRx denotes "Receive" by the phone. Port Number is separated by a colon. This is the receive port number or rtpLPort value from the GET request. For multicast, the RTPRx URI must be that of the server or endpoint transmitting (or streaming) audio. |

The following reserved URIs can be used in the **href** attribute to control an audio stream:

| Reserved URIs | Description |
|---------------------|---|
| RTPRx://STOP | Can be used to stop an audio streaming on the receive end. To be successful, all Push requests that result in sending the RTPRx://STOP must have a barge priority. |

Note:

For example, if an audio stream originator wanted to explicitly stop an audio stream the following would be sent in new Push Content:

```
<!-- Following is the XML Push Request Message sent as a POST request
embedded as part of form data -->
XMLData = <?xml version="1.0"?>
<Push type="audio" mode="barge">
    <go href="http://trusted_push_server/stop_audio.xml" method="get">
    </go>
</Push>
<!--The message below is from the Push Content file called
"stop_audio.xml" from above go href URI -->
<?xml version="1.0"?>
<Response>
    <Audio>
        <Url href="RTPRx://STOP">
    </Audio> <
</Response>
```

Audio quality depends on the streaming source providing the audio at an appropriate pace. The pace depends on the packet size. If you are using 40ms packets, then the packets should be separated by 40ms.

Transmitting the packets too slowly results in odd silences when the telephone has no audio to play out its Speaker. Transmitting the packets too quickly results in broken sound, as the telephone is forced to drop packets it cannot maintain in its buffer.

The error in the pace measured at the telephone is called jitter. If the jitter stays below the size of one packet, then jitter does not impact the audio quality. Note that a +2ms jitter on one packet cancels out a -2ms jitter on the next. However, 40ms packets, each separated by 38ms, means that the jitter grows +2ms with each packet. After 3 seconds, the jitter would be +150ms, and the telephone would need to drop audio to maintain its buffers.

Use Case Scenario:

Q. What is the best use for a **Receive Multicast Audio Push**?

A. Easily and quickly transmitting audio to multiple users without having to send individual push messages. For example, a school principal wants to transmit a message to teachers and students. The message can be pre-recorded ("Reminder. Students will be dismissed at 2:00 tomorrow due to scheduled building maintenance.") or can be a live emergency broadcast to all other phones in the system ("Due to extreme weather conditions, school will close in one half-hour. Students needing assistance or transportation should contact my office immediately."). The principal's phone is administered to transmit the pushed message (without duplication) to all telephones in each classroom, which get the message simultaneously as a receive multicast audio push.

The Transmit Audio Push Type

Transmit Audio pushes allow an end user to send an audio stream through the Trusted Receive Server (TRS). An exception is the 9610 IP Telephone, which can only send Transmit Audio pushes using the handset. Transmit Audio pushes have nearly identical characteristics and formats as Receive Audio pushes initiated by the administrator. Exceptions and processing differences are noted in this section.

The user initiates a Transmit Audio push by activating the microphone using either the Speaker, handset, or headset. A Voice Alert interrupt screen allows the user to initiate the Transmit Audio push by pressing the Talk softkey.

While transmitting pushed audio content, the user can switch between Speaker, Handset, and Headset as desired, adjust volume, and view LED states as with a call. The user can initiate this type of push while on a call by placing the active call on Hold, as with a barge-in Audio push. The user terminates the Transmit Audio push by going on-hook, selecting a call appearance, or by pressing the **Cancel** or **Exit** softkey from the Voice Alert interrupt screen.

When a Transmit Audio push is in progress, the telephone is considered not to be in a "pushable" state. Thus the telephone cannot receive a normal or barge-in Audio push or a Display push at the same time.

Processing

The Trusted Push Server/Trusted Receive Server (TRS) sends a barge-in push with type=audio to the telephone. The telephone provides the appropriate response based on the "Push State" query. If the response is Status Code 200 (OK), the telephone sends a request for the Push Content.

The TRS returns the Push Content with the correct response xml tag. The Push content must contain the remote IP Address and the remote port to which to transmit.

If the XML message is correctly parsed, the telephone displays the Transmit Push Interrupt screen, instructing the user to start to transmit using the **Talk** softkey. The user either speaks into the handset, Speaker microphone, or headset to transmit audio.

The audio timer available with Audio push types is not applicable to Transmit Audio push types. To stop an audio stream in play as a result of a transmit push a user can hang up the telephone or the TRS server can send a STOP Pushed content command. In this case, the audio streaming stops without any user intervention.

Interrupt Screens

An interrupt screen automatically accompanies a Transmit Audio push. The interrupt screen has a title line, Prompt line, and Application area content. The content area provides visual information about the transmit audio stream, for example, how to initiate the message or end the audio stream and return to the previous activity. Interrupt screens continue to display until the user either stops talking and takes action to end the push, hangs up (goes on-hook) to terminate the push, or cancels the push using an interrupt screen softkey.

Interrupt screens have priority over normal display pushes, but barge-in pushes may have priority over interrupt screens.

Transmit Audio Push Features

The Transmit Audio push type allows a user to transmit a non-call-associated audio push based on information obtained in response to a pushed HTTP request.

Alerts

A Transmit Audio Push Message can be sent with accompanying alerts. Alerts are a number of ring pings sounded just prior to displaying the message on the screen. With an Audio push type, an alert can be sounded with 1, 2, or 3 ring pings. If the **alert** attribute is not associated with the <Push> tag, then no alerts are sounded. Alternatively, if the **alert** attribute is set to "0" no alerts sound.

Transmit Audio Push Example

A third party visual voice mail application is an example of transmit push implementation. In this example a user can respond to a voice mail message viewed on the Phone screen by sending a non-call associated RTP stream directly back to the voice mail server.

Priorities and States

Audio Push Content is sent with one of two priorities: **normal** or **barge**. Normal priority push conditions are specified first, followed by barge priority push conditions. [Table 5](#) describes the result of a Receive Audio push on a Transmit Audio push.

Table 5: Receive/Transmit Audio Push Priorities

| This Audio Push Type | With this Priority | Action | This Audio Push Type | With this Priority |
|----------------------|--------------------|--------------------|----------------------|--------------------|
| Receive | Barge-in | DOES NOT interrupt | Transmit | Normal |
| Receive | Normal | DOES NOT interrupt | Transmit | Normal |
| Receive | Barge-in | DOES NOT interrupt | Transmit | Barge-in |
| Receive | Normal | DOES NOT interrupt | Transmit | Barge-in |

Pushable vs. Non-Pushable States

The following are **non-pushable** states for an IP telephone with the normal Transmit Audio push type.

- Any call appearance is alerting an incoming call.
- Any call appearance is active.
- The telephone is restoring a retrieved backup file.
- A Local (Craft) Procedure has been initiated. See the applicable SIP or H.323 Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Installation and Maintenance Guide on the <http://www.avaya.com/support> Web site for more details on local (Craft) maintenance procedures.

If a phone is not in one of the above states, it is considered to be in a **pushable** state, meaning the telephone will accept a Pushed Message.

Successful Push Response

If the phone is in a pushable state, the Push Agent sends the Push Initiator the following response for all Push request modes and all Push request types:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|-------------------------|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 200 | "Push Message Accepted" |

Normal Priority

When the **mode** attribute in the <Push> tag is set to **normal**, the telephone state for the Audio push type must be **pushable**. If the telephone is in the **pushable** state, then the telephone accepts the Audio push. The telephone broadcasts the pushed audio stream through the currently active audio device - the Speaker, Handset, Headset, or Bluetooth headset. Once the pushed audio stream is transmitted to the user, the user can redirect it, terminate it, etc.

An Audio push with a normal priority will not stream if the phone is in a **non-pushable** state. When the telephone is in a **non-pushable** state, any current audio activity continues without interruption or user notification and the Audio push stream is rejected. Error message 208 is sent and the Push Request does not proceed.

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|---|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 208 | "Not in Audio Push State: Push Aborted" |

Barge Priority

When the **mode** attribute in the <Push> tag is set to **barge**, the Audio Push Content is accepted as a priority message. However, a **barge** Audio push is rejected when the telephone is in one of these non-pushable states:

- When the telephone is in the process of restoring a retrieved backup file.
- When a Local (Craft) Procedure has been initiated (For more information, see the applicable SIP or H.323 *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Installation and Maintenance Guide* on <http://www.avaya.com/support>).
- When the telephone is already broadcasting transmitted pushed audio content.

In either non-pushable case, the **barge** audio content, including any notification tones, is discarded and the RTP socket is closed. In all other cases the telephone accepts the barge audio request regardless of any user activity and broadcasts the audio push content through the Speaker.

Use Case Scenario:

Q. What happens when a user is on the call and the interrupt screen indicates a **Transmit Audio** Push Message is sent?

A. If a user is on the call and a Transmit Audio Push Message is sent, the current active call is placed on [local] Hold while the user proceeds to transmit the audio stream. If the call is still on Hold when the pushed audio stream ends, the user is reconnected to the call. Note that the far-end party does not get any indication that the current call is being placed on hold.

Transmit Audio Push XML Messages

This section describes how to send an Audio push with XML messages.

Transmit Audio Push Message (PM)

The first step in sending an Audio push is to send an HTTP POST request from the Push Initiator to the telephone's Push Agent. Following is the XML Message (PM) format sent from the Push Initiator to the Push Agent:

```
<?xml version="1.0"?>
<Push
  alert="0|1|2|3"
  type="audio|transmit"
  mode="normal|barge"
>
  <go href="http://trusted_push_server/filename.xml"
  method="get|post">
    <postfield name="name1" value="value1"/>
    <postfield name="name2" value="value2"/>
  </go>
</Push>
```

Using the <postfield> Tag

The Web browser interface supports the <postfield> tag. The <postfield> tag allows an application to set a name/value pair that can be sent to the source of the request. The name is set by the **name** attribute and must be a valid WML variable name. The value is set by the **value** attribute. A <go> element can contain one or more <postfield> tags. Postfield tags must be sent if the HTTP POST method is used.

Note:

For more information on the <postfield> tag, see [Chapter 7: Developing Web Pages for the Browser](#).

Table 6: Description of Elements and Attributes used in the Audio Push XML Message

| Element or Tag | Attribute | Description |
|----------------|---------------------|--|
| <Push> | | Each Push Message must contain one valid root <Push> tag. |
| | alert=0 1 2 3 | Optional notification alerts – number of ring pings. the default value is “0.” |
| | type | Push Content = audio or transmit . |
| | mode | normal or barge priority. |
| | <go href=.../> | A fully qualified URL - to a valid XML file for Audio push Content. Cannot exceed 1024 characters. |
| | method | HTTP get or post methods. |

RTP Port

An RTP port is needed to stream audio to the telephone. The Push Agent sends the port information to the Trusted Push Server in the GET string. The GET string contains the variable “rtpLPort,” the telephone’s local port to be used for audio streaming.

The Push Agent responds to the server with:

| Push Message Type | Syntax |
|-------------------|-----------------------------------|
| “audio” | GET URI?rtpLPort=xxxx POST URI |
| “transmit” | GET URI?rtpLPort=xxxx POST URI |

The RTP Local port information sent by the Push Agent to the Push Initiator from the GET request is as follows:

| GET Parameter | Description |
|-----------------|--|
| rtpLPort | An RTP IP port is needed to stream audio from the telephone to the server. This is conveyed in the XML Audio Respond in the URI field: URI href="RTPTx://<IP>:<port>". |

Note:

The rtpLPort is generated dynamically and might be different for every new Audio Stream Push session. The telephone will play the audio stream only on this specified port.

Push Agent

Once a Push Message is received from the Push Initiator, the Push Agent first parses the XML file for validation and tag mismatch errors. Then the Push Agent verifies that the URL in the <go> tag is part of the Trusted Push Servers. The Push Agent then requests the Push Content from the Trusted Push Server using the URL.

For more information on Trusted Push Servers, see [Security](#).

Transmit Audio Push Content (PC)

Transmit audio push content must be a valid XML file that contains a valid <Response> tag. The <response> tag must contain a valid <Audio> tag and the <Audio> tag must contain a <Url> tag with an href attribute that begins with "RTPTx://". If the user wants to reply to a message using a non call-associated audio stream, the following syntax is used.

```
<?xml version="1.0"?>
<Response>
  <Audio packetsize="10|20|30|40|50|60" codec = "PCMU | PCMA" >
    <AudioTimer value="30"/>
    <Url href="RTPTx://URI from Push Request:remote_port"/>
    <Promptline>
      This text goes on the Prompt Line
    </Promptline>
  </Audio>
</Response>
```

The Audio Timer value does not apply to Transmit Audio pushes because as long as the handset, headset, or Speaker is active and as long as the server does not stop the session, RTP is sent. The XML syntax <Audio Timer value> remains, but the telephone ignores the timer value for a Transmit Audio push.

Each <Response> can contain only one <Audio> tag with the following attribute:

| Attribute | Description |
|--------------|---|
| <Promptline> | Provides the prompt line text to accompany Audio Interrupt Screens. |

Each <Url> tag consists of information for the RTP streaming server and the local receive port of the telephone. The <Url> tag has the following attributes:

| Attribute | Value | Description |
|-----------|--------|---------------------------|
| href | string | RTP Streaming URI Format. |

| RTP Streaming URI Format | URI/IP Address | Port |
|--------------------------|------------------------------------|--|
| RTPTx://URI:port | URI of the Trusted Receive Server. | RTPTx denotes "Transmit" to the phone. URI is the URI/IP address of the Trusted Receive Server that will be receiving the RTP, and port is the receive port number of the device receiving the audio. Port Number is separated by a colon. |

The following reserved URIs can be used in the href attribute to control an audio stream:

| Reserved URIs | Description |
|---------------|--|
| RTPTx://START | Used to start audio streaming at the transmitting end. |
| RTPTx://STOP | Used to stop an audio streaming from the transmitting end. |

The Subscribe Push Type

The Subscribe push is used as a subscription service for the IP telephones. The subscription service allows an intelligent application to get the telephone's information from an external database without having to query for the target telephone.

Subscribe Push Features

The features associated with the Subscribe push type are:

- Sends re-subscription requests when the Push type is Subscribe.
- No user interface is involved, since the subscription service is launched in the background.
- This feature is recommended to build databases of IP telephones.

The telephone provides the following information while subscribing to a particular subscription server:

- User's Telephone Extension
- IP Address of the telephone
- MAC Address of the telephone
- Set ID, an eight or nine character model number which is fixed and does not change:
 - For a 9610 IP Telephone, the Set ID is **9610D01A**
 - For a 9620 IP Telephone, the Set ID is **9620D01A**
 - For a 9620C IP Telephone, the Set ID is **9620D03C**
 - For a 9620L IP Telephone, the Set ID is **9620D02L**
 - For a 9630 IP Telephone, the Set ID is **9630D01A**
 - For a 9640 IP Telephone, the Set ID is **9640D01A**
 - For a 9640G IP Telephone, the Set ID is **9640GD01A**
 - For a 9650 IP Telephone, the Set ID is **9650D01A**
 - For a 9650C IP Telephone, the Set ID is **9650D02C**
 - For a 9670G IP Telephone, the Set ID is **9670GD01A**

Alerts

Alerts are not applicable to the Subscribe push type, as there is no user interaction.

Priorities and States

Normal and barge priorities are not applicable to the Subscribe push type.

The Subscription service is initialized during registration or the IP telephone boot-up process. Once the telephone is properly registered (logged in) with the media server, the telephone subscribes to the server listed in the **SUBSCRIBELIST** parameter.

Note:

For more information on SUBSCRIBELIST and boot-time subscription service, see [Chapter 4: Push Administration](#).

Successful Push Response

If the phone is in a pushable state, the Push Agent sends the Push Initiator the following response for all Push request modes and all Push request types:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|-------------------------|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 200 | "Push Message Accepted" |

Subscribe XML Messages

The following sections describe how to send a re-subscription Push request.

Subscribe Push Message (PM)

The first step in sending a Subscribe push is to send an HTTP POST request from the Push Initiator to the Push Agent in the telephone. Following is the format of the XML Message (PM) sent from the Push Initiator to the Push Agent:

```
<?xml version="1.0"?>
<Push type="subscribe">
  <go href= "http://ip_address/subscribe.xml"  method="get|post">
    <postfield name="name1" value="value1"/>
    <postfield name="name2" value="value2"/>
  </go>
</Push>
```

Using the <postfield> Tag

The Web browser interface supports the <postfield> tag. The <postfield> tag allows an application to set a name/value pair that can be sent to the source of the request. The name is set by the **name** attribute and must be a valid WML variable name. The value is set by the **value** attribute. A <go> element can contain one or more <postfield> tags. Postfield tags must be sent if HTTP POST method is used.

Note:

For more information on the `<postfield>` tag, see [Chapter 7: Developing Web Pages for the Browser](#).

Table 7: Description of Elements/Attributes used in the Subscribe Push XML Message

| Element or Tag | Attribute | Value | Description |
|----------------|----------------|------------|---|
| <Push> | | | Each Push Message must contain one valid root <Push> tag. |
| | type | subscribe | For the Subscribe push type, set type to subscribe. |
| | <go href=.../> | <Url> | A fully qualified URL - to an XML file with <Response> and the associated embedded <Subscribe> tag. |
| | method | get post | HTTP get or post methods (i.e. GET URI; POST URI). |

Push Agent

Once a Push Message is received from the Push Initiator, the Push Agent first parses the XML file for validation and tag mismatch errors. The Push Agent verifies that the URL in the `<go>` tag is part of the Trusted Push Servers. The Push Agent then requests the Push Content from the Trusted Push Server using the URL.

Note:

For more information on Trusted Push Servers, see [Security](#) in [Chapter 4: Push Administration](#).

Subscribe Push Content (PC)

The Push Content for the Subscribe push type must be an XML file. This XML file contains the URL for the subscription server and the type of subscription request made. The XML file must contain a valid `<Response>` tag that contains a valid `<Subscribe>` tag.

Using the <Response> Tag

The `<Response>` tag for the Subscribe push type must be the root element in the (PC) XML file.

Using the <Subscribe> Tag

Following is the syntax for the Subscribe XML file:

```
<?xml version="1.0"?>
<Response>
  <Subscribe type="all|me">
    <Url href= "Subscription_Server_Url" />
  </Subscribe>
</Response >
```

Each < Response> can contain only one <Subscribe> tag with the following attributes:

| Attribute | Value | Description |
|-----------|-------|--|
| type | "all" | Instructs the endpoint to re-subscribe to all servers in the SUBSCRIBELIST. |
| | "me" | Instructs the endpoint to re-subscribe to the server in the href attribute of the <Url> tag. This url string must exactly match one of the subscription server list servers in the SUBSCRIBELIST variable for this type of subscription to proceed. If the matching fails the subscription request is aborted. |

The <Url> tag has the following attributes:

| Attribute | Value | Description |
|-----------|--------|--|
| href | string | Contains the URL of the subscription server for which the endpoint must subscribe. Maximum length cannot exceed 1024 characters. |

An exact string-based comparison matches the subscription URIs against the SUBSCRIBELIST values. If the subscription server URI does not exactly match the values in the SUBSCRIBELIST, the subscription request is aborted.

Note:

See [Subscription Service](#) in [Chapter 4: Push Administration](#) for more information.

The Phonexml Push Type (for R2.2 & R2.5+ SIP IP Telephones only)

Use the Phone XML push to refresh phone-defined resources, clear history and set specific settings on a SIP IP Telephone running Release 2.2, 2.5 or later software only.

Phone Features

The phone xml push features and their associated parameters are displayed in the tables below:

Table 8: Clear Phone History

| Parameter Name | Value Type | Default | Description |
|---------------------|----------------------------|---------|--|
| CallHistory | boolean value (true/false) | | Clears the call log and redial values, and removes the Redial Softkey from the screen |
| WebHistory: | Boolean value (true/false) | | Clears the web history. |
| userSettingsHistory | | | Resets all user settings (from the Options menu) to the default values defined in the xls config file. |
| contactsHistory | | | Clears the Contacts history. |

Table 9: Refresh Phone Resource

| Parameter Name | Value Type | Default | Description |
|-----------------------|------------|---------|---|
| CurrentContent | <url> | “ “ | Refresh the phone display, updating all aspects of the display specified in the settings file under the CURRENT_CONTENT parameter. The url string should point to the appropriate content xml file. |
| UserPreferredLanguage | <url> | “ “ | Download a given language translation for the CurrentContent. The url should point to the language translation file that needs to be downloaded. |

Table 10: Set Phone Settings

| Parameter Name | Value Type | Default | Description |
|-------------------------|---------------|---------|---|
| UserDisplayName | <stringValue> | “ “ | Set the user's name as a replacement for the phone extension. |
| UserPreferredLanguage | <stringValue> | “ “ | Set the language to be used on the phone display to a language that has already been downloaded to the phone. If the requested language has not been downloaded to the phone, English will display. |
| CurrentSkin | <stringValue> | “ “ | Set the skin to be used on the phone. The skins that can be used are defined in the settings file, and are downloaded when the phone boots up. (this parameter is only applicable for the 9640). |
| ShowPhoneScreenOnCall | <intValue> | 0 | Enable this option to display the customized phone screen when there is a call in progress, as well as when there is not. Possible values are “1” and “0”, where 1 = enable and 0 = disable. |
| ShowPhoneScreenOn Alert | <intValue> | 1 | Enable this option to display the customized phone screen when the phone is alerting. Possible values are “1” and “0”, where 1 = enable and 0 = disable. |
| DisplayBrightness | <intValue> | 4 | Change the brightness of the phone display. Possible values are 1,2,3,4,5 where 5 is the brightest. |
| DisplayCallTimers | <intValue> | 1 | Enable this option if you want the phone to display the duration of active calls. Possible values are “1” and “0”, where 1 = enable and 0 = disable. |

| Parameter Name | Value Type | Default | Description |
|----------------------|---------------|---------|---|
| UseVisualAlerting | <intValue> | 1 | Enable this option to activate visual alerting on the phone. Possible values are "1" and "0", where 1 = enable and 0 = disable. |
| EffectOfRedialButton | <intValue> | 1 | Set this parameter to 1 to enable redialing for a single number or to 0 to enable redialing for a list of numbers. |
| DefaultAudioPath | <intValue> | 1 | Set the default audio path for the phone. Set this parameter to 1 to set the default audio path to speaker phone, or to 2 to default to the headset. |
| ButtonClicksEnabled | <intValue> | 1 | Enable or disable button clicks. Possible values are "1" and "0", where 1 = enable and 0 = disable. |
| ErrorToneEnabled | <intValue> | 1 | Enable this option to enable error tones on the phone. Possible values are "1" and "0", where 1 = enable and 0 = disable. |
| PersonalLabels | <stringValue> | " | Set personal labels on the phone. |
| CurrentLogo | <stringValue> | "" | Set the logo to be displayed on the phone. Logos that can be displayed are defined in the settings file, and downloaded to the phone when it is booted up. |
| TimeFormat | <intValue> | 0 | Set this option to change the format of the time on the phone display. Possible values are "0" and "1" where 0 displays the 12 hour clock and 1 displays the 24 hour clock. |

| Parameter Name | Value Type | Default | Description |
|-------------------|---------------|---------|--|
| WmlBrowserIdleUri | <stringValue> | “ “ | Specify the url that will be displayed when the WML idle time expires. See Using a Push to reset the value of WMLIDLEURL to turn off backlighting. |
| SwitchBacklight | <intValue> | 1 | When set to "1" this parameter turns on the display backlight. When set to "0" this parameter turns off the display backlight. |

Alerts

A phonexml Push Message should be sent with alerts equal to 0. If the alert attribute is not associated with the <Push> tag, the push message will be considered invalid and will be rejected. Any phonexml message being sent with alert other than zero will result in rejecting the push message.

Phonexml Push Example 1:

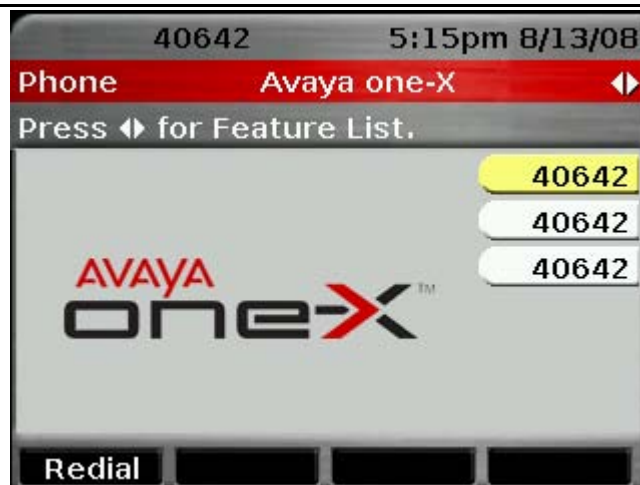
The following example describes sending a phonexml push message to the phone to change the user display name to **Richard**.

Note:

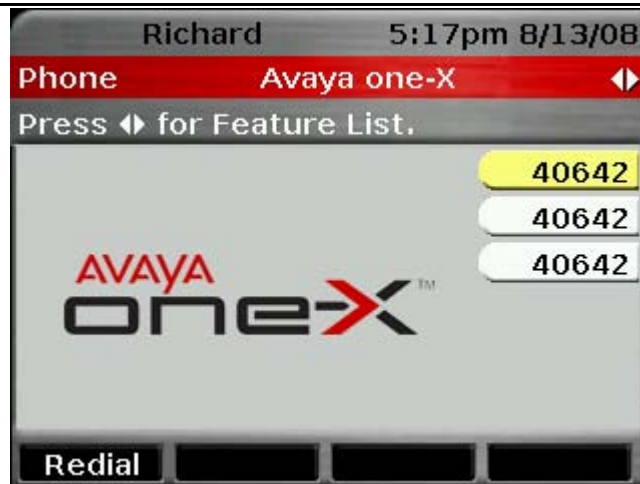
The default user display name is the phone extension.

Figure 12: Display Name Example

Before sending a phonexml push to change the user display name:



After sending phonexml push:



Priorities and States

Phonexml push is supported with ONLY **barge** priority. Normal phonexml push will be rejected by the phone.

Pushable vs. Non-Pushable States

The only states a registered telephone can be in, for which it rejects barge-in page and/or text string pushed content, are:

- When a Local Procedure has been initiated, with “initiated” meaning the complete Local Procedure has been specified, including PROCPSWD if applicable, and the terminating press of the “#” key.

In this case, the barge-in content (including any notification tones) is discarded. In all other cases, the telephone must accept the barge-in request.

If the telephone is already broadcasting any transmit or received pushed audio content, the phoneXML push is loaded in the background.

Successful Push Response

If the phone is in a pushable state, the Push Agent sends the Push Initiator the following response for all Push request modes and all Push request types:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|-------------------------|
| HTTP Status Code | 200 | “OK” |
| <i>x-Push-Status</i> | 200 | “Push Message Accepted” |

Normal Priority

When the **mode** attribute in the <Push> tag is set to **normal**, the telephone will reject phonexml push and return the following response to the push initiator:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|-----------------------------|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 401 | "Push Services not allowed" |

Phonexml Push XML Messages

This section describes how to send a phonexml push with XML messages. Use the user display name example in [Figure 14](#) as a reference.

Phonexml Push Message (PM)

The first step in sending a phonexml push is to send an HTTP POST request from the Push Initiator to the telephone's Push Agent. The XML Message (PM) sent from the Push Initiator to the Push Agent is as follows:

```
<?xml version="1.0"?>
<Push
  alert="0"
  type="phonexml"
  mode="barge"
>
<go href="http://trusted_push_server/filename.xml" method="get|post">
<postfield name="name1" value="value1"/>
<postfield name="name2" value="value2"/>
</go>
</Push>
```

Using the <postfield> Tag

The Web browser interface supports the <postfield> tag. The <postfield> tag allows an application to set a name/value pair that can be sent to the source of the request. The name is set by the **name** attribute and must be a valid WML variable name. The value is set by the **value** attribute. A <go> element can contain one or more <postfield> tags. Postfield tags must be sent if HTTP POST method is used.

Note:

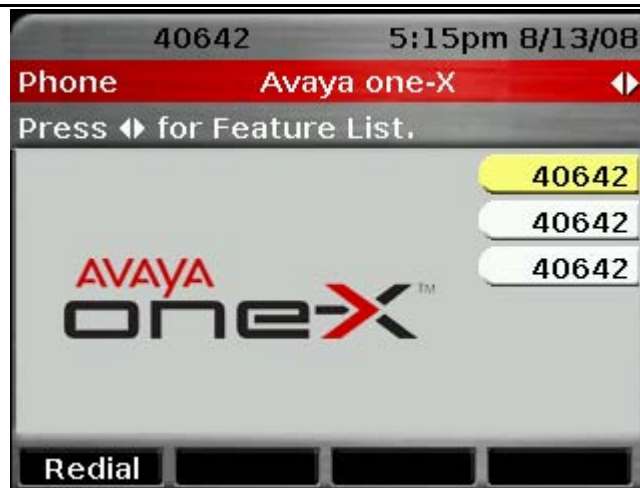
For more information on the <postfield> tag, see [Chapter 7: Developing Web Pages for the Browser](#).

| Element or Tag | Attribute | Description |
|----------------|----------------|---|
| <Push> | | Each Push Message must contain one valid root <Push> tag. |
| | alert=0 | Phonexml push does not support alert. Alert should be equal to zero at all times. |
| | type | Push Content = phonexml . |
| | mode | Barge priority. |
| | <go href=.../> | A fully qualified URL - to a valid XML Push Content file for phonexml pushes. Cannot exceed 1024 characters. See note above for H.323 Release 3.1 or later additions. |
| | method | HTTP get or post methods (i.e. GET URI; POST URI). |

Phonexml Push Example 2:

Using our previous user display name example, we want to display the user name to replace the phone extension.

Figure 13: Telephone Display prior to changing user display name



The code excerpt associated with the user display name to be sent as part of the Push Message is as follows:

```
<!-- Following is the XML Push Request Message sent as a POST request
embedded as part of form data -->
XMLData = <?xml version="1.0"?>
<Push alert="0" type="phonexml" mode="barge">
<go href="http://trusted_push_server/displayName.xml"
      method="get">

</go>
</Push>
<!-- The above message is part of the form data (XMLData) being sent
in Step 1 request -->
```

Push Agent

Once a Push Message is received from the Push Initiator, the Push Agent first parses the XML file for validation and tag mismatch errors. Then the Push Agent verifies that the URL in the <go> tag is part of the Trusted Push Servers.

Note:

For more information on Trusted Push Servers, see [Chapter 4: Push Administration](#).

Then the Push Agent requests the Push Content from the Trusted Push Server using the URL.

phonexml Push Content (PC)

The phonexml push type's Push Content has to be a valid XML file.

The following is the code excerpt associated with the user display name to be sent as part of the Push Content:

```
<!-- Server Sends Response - Push Content (PC) - File in the <Push...<go
href Url> -->
<?xml version="1.0"?>
<Response>
<SetSettingsRequest>
  <data>
    <name>UserDisplayName</name>
    <value>
      <stringValue>Richard</stringValue>
    </value>
  </data>
</SetSettingsRequest>
</Response>
```

Using the <Response> tag

The <Response> tag for the phonexml push type must be the root element in the (PC) XML file.

Using the <SetSettingsRequest> tag

The < SetSettingsRequest > is one of the supported tags by a phonexml response message. A list of tags that is supported by phonexml will be covered in the following section.

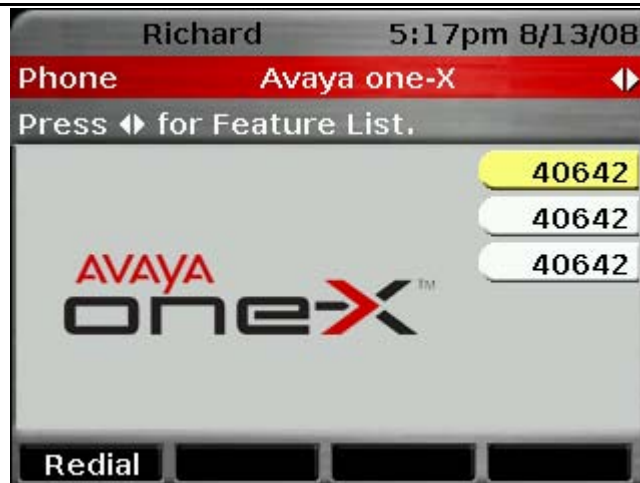
SetSettingsRequest tag is used to instruct to change some settings. A table detailing the supported settings and parameters can be found on pages 66-67.

If the length of the name exceeds the given pixels, the name will be truncated.

The SetSettingsRequest can only contain data elements. Data element includes a pair (name and value). Multiple data can be associated with a single SetSettingsRequest response.

The telephone's XML parser parses the XML file. Depending on the priorities and state of the telephone, the Push Content displays as shown in [Figure 11](#) with no alert.

Figure 14: User Display name



Using the <data> tag

The data tag is only associated with <SetSettingsRequest> request.

It contains a child <name> element which hold parameter name.

It also contains a child <value> element which holds the value of the parameter to be set. The <value> element one possible value of the three below choices to represents the type of data:

1. <StringValue>
2. <IntValue>
3. <BoolValue>

Using the <RefreshResourceRequest tag

The following example is a typical RefreshResourceRequest from the TPS server instructing the phone to download a new content file and update the content of the customized screen. If the content file is invalid, the screen will not be updated.

```
<?xml version="1.0"?>
<Response>
  <RefreshResourceRequest>
    <name>CurrentContent</name>
    <url>http://135.27.67.164/push/content.xml</url>
  </RefreshResourceRequest>
</Response>
```

The following example is a RefreshResourceRequest from the TPS server instructing the phone to download a language translation file. This translation file represents the actual content of the content file in a specific language. Avaya recommends that you have a translation file for every supported language.

```
<?xml version="1.0"?>
<Response>
  <RefreshResourceRequest>
    <name>UserPreferredLanguage</name>

    <url>http://135.27.67.164/push/FrenchTranslationFile.xml</url>
  </RefreshResourceRequest>
</Response>
```

Detailed information about the Content xml file can be found in the *Avaya one-X™ Deskphone Edition for 9600 Series SIP IP Telephones Developer Guide* available on the Avaya support Web site.

Using the <ClearPhoneHistoryRequest> tag

The following is an example of the type of response received from the TPS to instruct the phone to clear its web history, call history and user settings history.

Any child elements of ClearPhoneHistoryRequest that are not presented, will be treated with a value of False.

```
<?xml version="1.0"?>
<Response>
  <ClearPhoneHistoryRequest>
    <webHistory>true</webHistory>
    <callHistory>true</callHistory>
  </ClearPhoneHistoryRequest>
</Response>
```

Using a Push to reset the value of WMLIDLEURL to turn off backlighting

To enable this feature where the phone initiates a request when the WMLIDLETIME expires, a valid URL must be set to WMLIDELURL. To disable this feature an empty value is set to WMLIDELURL.

The user sees a line appearance button (for example, a “don’t show ads” link to a wml page) or a link in a WML page. When pressed, the phone sends a request to the server to fetch the page. After the server receives the request, the server sends a push message to the phone (phonexml setSettingRequest to set WMLIDELURL = null), and no advertisement is pushed to a web page on this phone.

Since the WMLIDELURL becomes an empty string, the phone will not initiate any wml page request during the night time (the phone will turn off backlighting after the BACKLIGHTOFF timer expires).

The previous scenario describes how the user “turns off” WMLIDELURL. A push message can be sent to the phone without the user’s request to disable WMLIDELURL. For example, a phonexml push can be sent where WMLIDELURL = null at 10pm, and then another phonexml push can be sent to enable backlighting where WMLIDELURL=http://www.xxxx.com/idle.wml” at 9am.

To disable WMLIDLEURL:

```
<SetSettingsRequest>
  <clientId>1</clientId>
  <requestId>2</requestId>
  <data>
    <name>WmlBrowserIdleUri</name>
    <value>
      <stringValue/>
    </value>
  </data>
</SetSettingsRequest>
```

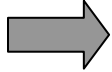
To enable WMLIDLEURL:

```
<SetSettingsRequest>
  <clientId>1</clientId>
  <requestId>2</requestId>
  <data>
    <name>WmlBrowserIdleUri</name>
    <value>
      <stringValue>http://mydefaultidleurl.com/index.wml
    </stringValue>
  </value>
</data>
</SetSettingsRequest>
```

Chapter 4: Push Administration

Introduction

This chapter covers the administrative actions required for the Push interface to work properly.



Unless otherwise indicated, the information in this chapter applies to both H.323 and SIP IP Telephones.

Note:

Consult your System Administrator if appropriate to modify push-related settings. Also, see the applicable H.323 or SIP *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Administrator Guide* on the <http://www.avaya.com/support> Web site for more information.

Requirements

For information on setting system parameters using the settings file and for descriptions and values of the system parameters that apply to pushing content, see the *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephone Administrator Guide* (Document Number 16-603158) for H.323 IP Telephones or the *Avaya one-X™ Deskphone Edition for 9600 Series SIP IP Telephones Administrator Guide* (Document Number 16-600944, for SIP Software Release 2.2, 2.5 or later, as applicable) for SIP IP Telephones.

TPSLIST

To enable the Push interface, the non-null parameter TPSLIST (Trusted Push Server List) must be administered in the 46xxsettings.txt script file on the HTTP server.

Note:

The 46xxsettings file is used for both 9600 Series and 4600 Series IP Telephones.

The remaining sections in this chapter discuss this topic in more detail.

SUBSCRIBELIST

Use the SUBSCRIBELIST parameter to define the list of subscription servers to which all the telephones will subscribe. The SUBSCRIBELIST parameter is not required to send Push Messages. The remaining sections in this chapter discuss this topic in more detail.

PUSHCAP

Use the system parameter PUSHCAP (Push Capabilities) to specify which push types and their modes (priorities) are allowed on the phone. Each of four (H.323) or five (SIP Release 2.2, R2.5+) PUSHCAP digits controls a different type of push. The rightmost digit controls Top line pushes, the next digit to the left controls display (web) pushes, the next digit to the left of that controls Audio receive pushes (including multicast in Software Release 3.0 for H.323 phones only), the next digit to the left of that controls Audio transmit pushes, and the leftmost digit controls Phone XML pushes (for SIP phones running SIP Software Release 2.2, 2.5, or later only).

Each digit of PUSHCAP can have one of the following values. Any other value will be treated as 0.

| PUSHCAP VALUE | MEANING |
|---------------|--|
| 0 | Push type is completely disabled |
| 1 | Barge-in only push mode is allowed |
| 2 | Normal and barge-in push modes are allowed |

If PUSHCAP has only 3 digits, the fourth (and fifth for SIP phones running Release 2.2, 2.5, or later) leftmost digit is considered to be "0".

The following example shows the push mode associated with each digit of PUSHCAP with a value of 2101.

| Phonexml value (SIP Release 2.2, 2.5, or later only) | Audio transmit value | Audio receive value (also used for multicast in H.323 Software Release 3.0) | Display (Web) push value | Top Line value |
|--|----------------------|---|--------------------------|----------------|
| 1 | 2 | 1 | 0 | 1 |

A value of 2222 for H.323 or 22222 for SIP means that all four or five types are enabled for barge-in and normal-mode pushes. A value of 0000 for H.323 or 00000 for SIP (Release 2.2, 2.5+) would disable all push types completely.

If the port specified by the system parameter PUSHPORT is already in use, the PUSHCAP value is set to "0000" (H.323) or "00000" (for SIP Release 2.2, 2.5+). If the PUSHCAP value is "0000" (or "00000" for SIP Release 2.2, 2.5+) a port will not be opened for the HTTP server and Push capability will remain disabled.

If a Push request message is disabled based on the PUSHCAP value, the push request will be rejected and the response back to the Push Initiator will contain the following parameters:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|-------------------------|
| HTTP Status Code | 200 | "OK" |
| <i>x-Push-Status</i> | 201 | "Push service disabled" |

PUSHPORT

Use the system parameter PUSHPORT to indicate the TCP listening port number used for the telephone's HTTP server. The Push Agent listens only to the PUSHPORT specified for all incoming requests. The default port is "80" but this value can be 2 to 5 ASCII numeric digits "80" through "65535".

Security

Push security is validated by a domain-based authentication algorithm. If the Push Content URL is not part of the Trusted Push Server, the Push Message is denied. For more information, see the sections that follow on [Trusted Push Server List \(TPSLIST\)](#) and [Validation Scenarios](#).

Trusted Push Server List (TPSLIST)

The URI received in the Push message is validated against the value(s) of TPSLIST in the settings file, and the Push message is either accepted for additional processing or denied.

TPSLIST contains zero or more URI authority elements, each of which may optionally include a scheme of "http://" and/or a path, as specified in Section 3 of IETF RFC 3986. Each authority element must be resolvable into one or more specific IP addresses, otherwise it will be considered invalid.

An administrator can set the security level using the following domain-based security values for the TPSLIST parameter in the 46xxsettings.txt script file.

| System Value Name | Application Software Default | Usage (Value Range; References) |
|-------------------|------------------------------|--|
| TPSLIST | "" (Null) | List of Trusted Push Servers. Contains zero or more domain/path strings, separated by commas without any intervening spaces. Up to 255 ASCII characters, including commas are allowed. |

Note:

Effective with SIP software Release 2.5, the push URL must match the TPSLIST address value exactly or the telephone will reject the push request. For example, if the push URL is of the form:
http://148.147.171.100:80/push.wml then the TPSLIST value must be set as 148.147.171.100:80. The port number in the push URL (shown as "80" in the example) must also be present in the TPSLIST value.

Specifically, the List of Trusted Push Servers contains one or more domains and paths in a DNS format or an IP Address in dotted-decimal format, separated by commas.

- The administrator sets the TPSLIST values in the 46xxsettings.txt script file on the HTTP server. See the applicable SIP or H.323 *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Administrator Guide* on the <http://www.avaya.com/support> Web site for more details.
- If the value of TPSLIST is null, Push Messages cannot be received and the Push interface is disabled.
- Wild cards such as asterisks are not allowed for URL strings in the TPSLIST.

The pushed URI string (the value of the href attribute in the <go> tag in the push request) is compared to each of the values of TPSLIST after all percent-encoded characters (see Section 2.1 of IETF RFC 3986) in the URI string have been replaced by their single-octet equivalent. To be considered valid,

- the beginning of the pushed URI string must begin with "http://",
- the pushed URI string must contain an authority element that exactly matches the authority element of one of the values of TPSLIST, and
- if the TPSLIST value also includes a path, the beginning of the path in the pushed URI string must exactly match the path in the TPSLIST value.
- For SIP phones running Software Release 2.2, 2.5, or later, the response from the Trusted Push Server must be consistent with the value of the type attribute in the <Push> tag sent by the Push Initiator.

Otherwise, the push request will be rejected and the response back to the Push Initiator will contain the following parameters:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|-------------------------|
| HTTP Status Code | 403 | “Forbidden” |
| <i>x-Push-Status</i> | 402 | “Push security failure” |

To validate a particular pushed URI string, a string-based comparison is done against the values of the TPSLIST administered in the 46xxsettings.txt script file.

Validation Scenarios

Table 11: URI Examples

| Validation String (Validation string Interpreted as) | Pushed URI string ‘X’ means failure and ‘OK’ means success. | | | |
|---|--|--|---|---|
| / = domain/path separator | Example 1 http://www. company.com/ | Example 2 http://support. company.com/ | Example 3 http://www. company.com /push/ | Example 4 http://www. hacker.com/ |
| “company” (“company/”) | X | X | X | X |
| “company.com” (“company.com/”) | X | X | X | X |
| “support.company.com” (“support.company.com/”) | X | OK | X | X |
| “company.com/push” (“company.com/push”) | X | X | X | X |
| “/push” (“/push”) | X | X | X | X |
| “company.com/raj/push”\n (“company.com/raj/push”) | X | X | X | X |

| Validation String (Validation string Interpreted as) | Pushed URI string 'X' means failure and 'OK' means success. | | | |
|--|--|----|----|---|
| "http://www.company.com" ("http://www.company.com/") | OK | X | OK | X |
| "http://support.company.com" ("http://support.company.com/") | X | OK | X | X |
| "http://support.company.com/push" ("http://support.company.com/push") | X | X | X | X |
| "http://www.company.com/push" ("http://www.company.com/push") | X | X | OK | X |
| "/" ("/") | X | X | X | X |

Recommendations:

If the domain from [Table 11](#) is "**associate.hacker.com**" and the TPS string is "**company.com**", then the domain "**hackercompany.com**" will also match the TPS string and pass the security validation. For example, if the TPS string is "**company.com**" & the URLs are:

http://www.company.com - this URL is acceptable.
http://associate.company.com - this URL is acceptable.
http://www.hackercompany.com - this URL is acceptable.

In this case "**hackercompany**" also passes the security validation.

To avoid such security issues, the administrator must set the TPS string to "**.company.com**" (where there is a "**dot**" before "**company**").

For example, if the TPS string is "**.company.com**" and the URLs are:

http://www.company.com - this URL is acceptable.
http://associate.company.com - this URL is acceptable.
http://www.hackercompany.com - this URL is not acceptable.

In this case "**hackercompany**" fails the security validation.

Subscription Service

The phone provides a set of values such as the IP Address and user telephone number to subscription servers when the telephones register with the media server. An explicit subscription message can also be sent to the phone using `<Push type="subscribe"... />` to re-subscribe the phones.

Note:

See [Chapter 3: Creating Push Messages](#) for more details on sending a re-subscribe Push Request to the telephone.

Applications must maintain their own database of information for each phone. Databases can be built with respect to a group of telephones or an individual phone.

A typical example in a corporate work environment is to have distribution lists according to specific teams. An application can only target pushes to that team such as "team meeting reminders" with conference bridge numbers, etc.

Once an application builds a database that has the required information of all of the telephones in the network, it can target Push Content or an audio stream to a specific phone or a group of phones. Additionally, Push Initiator Servers can poll the phone to update its database.

Note:

A phone in a logoff state does not unsubscribe from the Subscription server.

Subscriber Service

Using the Push Subscription Service, the phone makes the following values known to the trusted subscription service server:

- IP Address of the Phone
- User's Extension
- MAC Address
- Set ID, an eight or nine character model number which is fixed and does not change
 - For a 9610 IP Telephone, the Set ID is **9610D01A**
 - For a 9620 IP Telephone, the Set ID is **9620D01A**
 - For a 9620C IP Telephone, the Set ID is **9620D03C**
 - For a 9620L IP Telephone, the Set ID is **9620D02L**
 - For a 9630 IP Telephone, the Set ID is **9630D01A**

- For a 9640 IP Telephone, the Set ID is **9640D01A**
 - For a 9640G IP Telephone, the Set ID is **9640GD01A**
 - For a 9650 IP Telephone, the Set ID is **9650D01A**
 - For a 9650C IP Telephone, the Set ID is **9650D02C**
- For a 9670G IP Telephone, the Set ID is **9670GD01A**

The HTTP Push Subscription Message Syntax is as follows:

```
GET:http://<server_IP address>/<script_name>?MAC="xxx"&Extn="xxx"&ip="xxx"& SetID="xxx"
```

Upon log on to the call server, the Push Subscription service updates all values. No unsubscribe event is sent upon user logoff.

Subscription List (SUBSCRIBELIST)

This list specifies the Push administrative requirements for setting a new system value for the subscription-trusted list of servers. Specifically, the Subscription list contains one or more fully qualified URLs, separated by commas.

An administrator can set the subscription server addresses using the following domain-based values for the SUBSCRIBELIST parameter in the 46xxsettings.txt script file.

| System Value Name | Application Software Default | Usage (Value Range; References) |
|-------------------|------------------------------|---|
| SUBSCRIBELIST | "" (Null) | List of Trusted Subscription Servers, contains zero or more domain/path strings, separated by commas without any intervening spaces. Can be up to 255 ASCII characters, including commas. |

Specifically, the List of Trusted Subscription Servers contains one or more fully qualified URLs of the subscription servers, separated by commas.

The values for SUBSCRIBELIST are set by the administrator in the 46xxsettings.txt script file on the HTTP server. See the applicable SIP (Release 2.2, 2.5, or later as applicable) or H.323 *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Administrator Guide* on the <http://www.avaya.com/support> Web site for more details.

Note that the Subscription Servers specified in the SUBSCRIBELIST parameter are considered to be Trusted Push Server. Hence, no additional security validation is done for the subscription server URIs.

The syntax of the Trusted Subscription List is a series of fully qualified URIs of the form:

```
<scheme>://<host>:<port>/<url-path>
```

If `http://10.0.1.101/subscribe.asp`, `http://company.com/subscribe/`, and `http://abc.company.com:8000/cgi/subscribe` are lists of the three subscription servers, then use the following syntax in the `46xxsettings.txt` script file to administer these URLs as the subscription servers:

```
SET SUBSCRIBELIST
http://10.0.1.101/subscribe.asp,http://company.com/sub
scribe/,http://abc.company.com:8000/cgi/subscribe
```

Subscription Update

Use the Subscribe push type to make an asynchronous request to a phone to re-subscribe and update all the values defined in the syntax examples above.

Note:

See [Chapter 3: Creating Push Messages](#) for more details on sending a Subscribe Push Request to the telephone.

An exact string-based comparison is done to match the subscription URIs against the values in the SUBSCRIBELIST. If the subscription server URI does not exactly match the values in the SUBSCRIBELIST, the subscription request is aborted.

Retry Timer

If the subscription server is unresponsive to the initial subscription message from the phone at boot up and does not return a Message 200 (OK), then the subscription service retry timer will send subscription messages to the subscription server every 20 minutes, up to five times.

The retry timer does not apply to a subscription update from the subscription server.

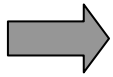
Denial of Service Timer

The Denial of Service Subscription Timer limits the number of subscription updates requested by the trusted subscription service. The Push Agent following a successful subscription will accept only one subscription request per minute. This implies that the Push Agent in the telephone can handle one subscription request per minute.

The denial of service timer starts following a Push Request for Subscribe. No Subscribe request is accepted for the next minute. If another Subscribe request is received within that time, the response is as follows:

| Parameter | Status Code | Reason Phrase |
|----------------------|-------------|-----------------------|
| HTTP Status Code | 200 | “OK” |
| <i>x-Push-Status</i> | 503 | “Service Unavailable” |

Chapter 5: Troubleshooting the Push Interface



Unless otherwise indicated, the information in this chapter applies to both H.323 and SIP IP Telephones.

Avaya HTTP Header Extensions (x-Push-Status Codes)

The Push Agent sends a response back to the Push Initiator with HTTP status codes.

Note:

See [Push Message Flow](#) and [Figure 4](#) in [Chapter 2: Push Interface Overview](#) for an overview of the Push process.

In addition to the HTTP response codes, an additional HTTP header extension called “x-Push-Status” is sent.

The x-Push-Status header is used to indicate the outcome of a Push Request back to the Push Initiator. The header is included in all responses to the Push Initiator for Push Requests. General guidelines for error codes for x-Push-Status are as follows:

- Status-Code 200-299: Push Service Response and Rejection codes, Push State codes
- Status-Code 300-399: XML parsing codes
- Status-Code 400-499: Trusted Domain Security codes
- Status-Code 500-599: General rejection reasons

Table 12: x-Avaya-Push-Code Responses

| x-Push-Status code | Reason Phrase (Description) | Resolution (“N/A”: Avaya IP Telephones never send the status code) |
|--------------------|-----------------------------|---|
| 200 | “Push Message Accepted” | This reports that the telephone has successfully accepted the push. |
| 201 | “Push Service Disabled” | Ask your system administrator to administer the TPSLIST value on the HTTP Server and to verify that the PUSHCAP value is properly set for the different types of pushes. See Chapter 4: Push Administration for more information. |

| x-Push- Status code | Reason Phrase (Description) | Resolution (“N/A”: Avaya IP Telephones never send the status code) |
|--------------------------------|---|---|
| 202 | “Internal Error: Push Aborted” | There was a problem processing your request. Re-send the Push request. |
| 204 | “Not in Push State: Push Accepted” | The Push Message was accepted but the user’s telephone is busy, for example, in text-edit mode. The Push Message is loaded in the background. Once the user launches the Web browser on the phone, the user will be able to view the Push Message. Try sending a “barge” priority message instead if you want the user to view the message immediately. |
| 205 | “Not In Push State: Push Aborted” | The Push Message was rejected because the phone is in a non-pushable state. Try sending a “barge” priority message instead. See Chapter 3: Creating Push Messages for more information. |
| 208 | “Not in Audio Push State: Push Aborted” | The audio stream was rejected because the user is currently on the call or busy. Try sending a “barge” priority message instead. See The Receive Audio Push Type for more information. |
| 209 | “Not in Audio Push State: RTP listening port not available” | Change the RPTL port. Applies only to H.323 phones running software Release 3.0 and greater. |
| 301 | “XML document not valid” | Check the proper XML schema as described in this API. See Chapter 3: Creating Push Messages for more information. |
| 302 | “XML document not well-formed” | Make sure that all the XML tags are properly formed. See Chapter 3: Creating Push Messages for more information. |
| 303 | “No element found” | Make sure that all the XML tags are properly formed. See Chapter 3: Creating Push Messages for more information. |
| 304 | “Unknown encoding” | Make sure that all the XML tags are properly formed. See Chapter 3: Creating Push Messages for more information. |

| x-Push- Status code | Reason Phrase (Description) | Resolution (“N/A”: Avaya IP Telephones never send the status code) |
|--------------------------------|--|--|
| 305 | “Invalid root element” | Make sure that all the XML tags are properly formed. See Chapter 3: Creating Push Messages for more information. |
| 306 | “Empty Post Content” | If you are using the HTTP POST method in the Push Message, include postfield tags with the message. |
| 307 | “Push Content too large” | The XML file is larger than 5Kb. Send an XML file less than 5K bytes. |
| 310 | “Multicast XML Failure: No RTP listening port specified” | Set the RTPL port value in the <set name> element for the audio receive push. Applies only to H.323 phones running software Release 3.0 and greater. |
| 311 | “Multicast XML Failure: No Multicast IP address specified” | Set the multicastip value in the <set name> element for the audio receive push. Applies only to H.323 phones running software Release 3.0 and greater. |
| 312 | “XML Failure: RTP listening port specified is less than 10001” | The RTPL port value must be within the range 10001 and 65535. Applies only to H.323 phones running software Release 3.0 and greater. |
| 313 | “XML Failure: RTP listening port specified is greater than 65535” | The RTPL port value must be within the range 10001 and 65535. Applies only to H.323 phones running software Release 3.0 and greater. |
| 314 | “Multicast XML Failure: Multicast IP address is not in the expected range of 239.0.0.0 to 239.255.255.” | Set the multicastip value to a valid IP address within the specified range. Applies only to H.323 phones running software Release 3.0 and greater. |
| 402 | “Push security failure” | The security service failed to validate the Trusted Push Server. |
| 501 | “Invalid Push URI” | Recheck the URI included in the <go href.../> request for fully qualified URL string. Also, make sure the length of the URI is no longer than 1024 characters. |

| x-Push-Status code | Reason Phrase (Description) | Resolution (“N/A”: Avaya IP Telephones never send the status code) |
|---------------------------|------------------------------------|--|
| 503 | “Subscription Service Unavailable” | Ask your system administrator to administer the SUBSCRIBELIST value on the HTTP Server. See Chapter 4: Push Administration for more information. |

HTTP Error Messages

The following standard HTTP Error Messages will be supported. Status codes starting with “4” indicate a client error in which the request contains bad syntax or cannot be fulfilled. Status codes starting with “5” indicate a server error in which the server failed to fulfill an apparently valid request.

Table 13: HTTP Error Messages

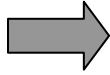
| Status Code | Cause of Failure | Failure Message Displayed to the User/Resolution (“N/A”: Avaya IP Phone will never send the status code) |
|--------------------|-------------------------|---|
| 400 | Bad Request | Due to incorrect syntax the server could not understand the request. |
| 401 | Unauthorized | The request requires user authentication. |
| 402 | Push Security Failure | N/A |
| 403 | Forbidden | The server has indicated it will not respond to your query. |
| 404 | Not Found | The server has not found anything matching the request. |
| 405 | Method Not Allowed | N/A |
| 406 | Not Acceptable | N/A |

| Status Code | Cause of Failure | Failure Message Displayed to the User/Resolution ("N/A": Avaya IP Phone will never send the status code) |
|--------------------|---------------------------------|---|
| 407 | Proxy Authentication Required | The request requires user authentication. |
| 408 | Request Time-out | N/A |
| 409 | Conflict | The request could not be completed due to a conflict with the current state of the resource. |
| 410 | Gone | The requested resource is no longer available. |
| 411 | Length Required | The server refuses to accept the request. |
| 412 | Precondition Failed | N/A |
| 413 | Request Entity Too Large | The server refuses to accept the request because the request entity is too large. |
| 414 | Request-URI Too Large | The request-URI is longer than the server can interpret. |
| 415 | Unsupported Media Type | The server refuses to accept the request because it's in an unsupported format. |
| 416 | Requested range not satisfiable | N/A |
| 417 | Expectation Failed | N/A |
| 500 | Internal Server Error | The server encountered an unexpected condition that prevented it from fulfilling the request. |
| 501 | Not Implemented | N/A |
| 502 | Bad Gateway | The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request. |

| Status Code | Cause of Failure | Failure Message Displayed to the User/Resolution ("N/A": Avaya IP Phone will never send the status code) |
|--------------------|----------------------------|---|
| 503 | Service Unavailable | The server is currently unable to handle the request due to a temporary overloading or maintenance of the server. |
| 504 | Gateway Time-out | The server, while acting as a gateway or proxy, did not receive a timely response from the upstream server specified. |
| 505 | HTTP Version not supported | The server does not support, or refuses to support, the HTTP protocol version that was used in the request message. |

Chapter 6: About the Web Browser

Introduction



The information in this chapter applies to both H.323 and SIP 9600 Series IP Telephones.

This chapter provides general information about and requirements for developing Web browser applications for the 9600 Series IP Telephones, which currently include the following eleven models:

- 9610
- 9620/9620C/9620L
- 9630/9630G
- 9640/9640G
- 9650/9650C
- 9670G

Unless otherwise indicated, the information covered applies to all 9600 Series IP Telephones to which Web application development applies.

Note:

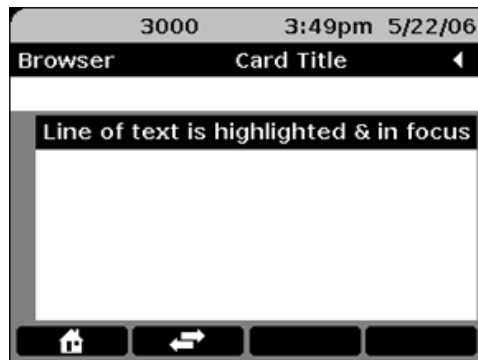
Due to its smaller display area and unique functionality, the 9610 IP Telephone Web browser has somewhat different capabilities. For more information, see [9610 WML Browser Feature Differences](#). The 9670G IP Telephone is a high-end, touch screen telephone which also has somewhat different capabilities than the other 9600 Series IP Telephones; see [9670G WML Browser Feature Differences](#) for more information.

Basic browser principles apply to all telephones capable of maintaining a Web browser. This chapter describes those common characteristics.

For detailed information about Web application development for 9600 Series IP Telephones, see [Chapter 6: About the Web Browser](#). For specific information about Web application development for color phones or cascading style sheets, see [Chapter 7: Developing Web Pages for the Browser](#). For specific information about administering Web applications, see [Chapter 8: Web Applications](#).

[Figure 15](#) provides a schematic view of the standard Web browser display for the 9600 Series IP Telephones (with the exception of the 9610 and 9670G), and the softkeys associated with that particular display.

Figure 15: Web Browser Display and Softkeys



[Figure 16](#) shows the display area in relation to the telephone itself and the Navigation cluster directly below the softkeys for all phones except the 9610 and 9670G.

Figure 16: 9620/9620C/9620L IP Telephone



Figure 17: 9630/9630G/9640/9640G IP Telephone



Figure 18: 9650/9650C IP Telephone



Figure 19: 9610 IP Telephone



Figure 20: 9670G IP Telephone



Note:

The 9600 Series IP Telephones have differing numbers of line buttons. The 9620/9620C and 9620L phones have no line buttons, but have three LEDs to indicate call appearance/line activity. The 9630/9630G and 9640/9640G have six line buttons. The 9650 has three line buttons but also has 8 auxiliary buttons, with an "Aux (shift)" button to access 8 additional auxiliary buttons. The 9670G has six lines, however, when the Quick Touch panel is enabled only three lines are visible. The 9610 is a single-line telephone and has no line buttons.

Differences in 9600 Series IP Telephone Models

While the WML Browser is expected to operate in a consistent manner on all 9600 Series IP Telephone models, some physical design differences and feature differences of specific models force differences in browser operation. For example, some models can display six Application Lines at a time, but some can display only three at a time. Some models have a Line Button associated with each Application Line, and others do not. Some displays support color, and some do not. However, most differences are found in the models at the low end of the product line (the 9610) and at the high end of the product line (the 9670G). The major differences for these two models are summarized below.

9610 WML Browser Feature Differences

There are no Line Buttons on the 9610 IP Telephone.

There is a 6 line application area.

The 9610 does not have a Title Line or a Prompt Line, so card and element titles are displayed on the Top line, the name of the "Browser" application (or an alternate name specified in a <meta> element) is not displayed at all, and left and right arrow icons are not displayed for the History Stack. Titles can be presented to the user on the Top Line as each line on the screen is individually brought "into focus." Content may also be rendered on the Top Line for this phone.

Only two softkeys can be displayed at a time.

Only the **Up** or **Down** navigation controls with the **OK** button can be used to select a particular line on the display, or "to bring that line into focus."

Default **Home**, **Refresh** and **Stop** softkeys are not supported, but the 9610 does support an **Exit** softkey.

The rendering area is much smaller than the other 9600 Series IP Telephones - approximately 19 characters including icons can fit on each display line

Users access Web pages via links embedded in the 9610 Menu screen and connected pages. The labels for URLs in the menu screen are not underlined.

Text entry is not supported, but the 9610 does support Access Key Input Mode (AIM).

Because text entry is not supported, HTTP authentication cannot be supported either. If the server sends the authentication, the telephone displays the error message "Incorrect WML syntax."

Since the 9610 is intended to be used in shared common areas, the History Stack is cleared every time the Idle Application is displayed, and the WTAI Add to Phonebook URI function is not supported because end users cannot create or edit Contacts.

Since the 9610 can display only about half as many characters per line as the other telephone models, it does not display a Handset icon for the WTAI Make Call (Click to Dial) URI function.

The home page URL is specified by the system parameter WMLSMALL, not by the system parameter WMLHOME.

9670G WML Browser Feature Differences

A touch screen is provided for navigation, scrolling and selection; there are no Line Buttons or Navigation Buttons. Right and left page control arrows are touched to facilitate navigating card stacks/pages.

- An on-screen keyboard is provided for text entry.
- Up to 5 softkeys can be displayed at a time.
- The scroll bar is rendered on the right side of the display.

Summary of differences among the models in the product line

| Feature | 9610 | 9620 9620L | 9620C | 9630 9630G | 9640 9640G | 9650 | 9650C | 9670G |
|--------------------------------------|------|---------------|-------|---------------|---------------|------|-------|-------|
| Top Line | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Title Line | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Prompt Line | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Application Lines | 6 | 3 | 3 | 6 | 6 | 3 | 3 | 6 |
| Line Buttons | 0 | 0 | 0 | 6 | 6 | 3 | 3 | 0 |
| Max. Line Width (characters) | ~15 | ~30 | ~30 | ~30 | ~30 | ~30 | ~30 | ~40 |
| Max. Image Width (pixels) | 153 | 297 | 297 | 297 | 297 | 297 | 297 | 564 |
| Max. Image Height per line (pixels) | 20 | 23 | 23 | 23 | 23 | 23 | 23 | 48 |
| Maximum Image Size without scrolling | 120 | 69 | 69 | 138 | 138 | 138 | 138 | 288 |
| Softkeys per screen | 2 | 4 | 4 | 4 | 4 | 4 | 4 | 5 |

| Feature | 9610 | 9620 9620L | 9620C | 9630 9630G | 9640 9640G | 9650 | 9650C | 9670G |
|-------------------------|---------------------|---------------------|-------|---------------------|---------------|---------------------|-------|-------|
| Softkey Height (pixels) | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 38 |
| Color support | 4- level gray scale | 4- level gray scale | Yes | 4- level gray scale | Yes | 4- level gray scale | Yes | Yes |
| Navigation Buttons | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No |
| Touch screen | No | No | No | No | No | No | No | Yes |
| Text Input | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| WTAI Add to Phonebook | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes |

Note:

The Maximum Image Height is the image height per application line multiplied by the maximum number of lines (96).

Physical Attributes

The IP telephones that support the Web browser have these characteristics:

- Six Line buttons are available for Web applications on the 9630 and 9640. Three Line buttons are available for Web applications on the 9650. The 9610, 9620, and 9670G have no line buttons. Except for the 9670G, telephones without line buttons use the **Up** and **Down** Arrows and **OK** button for that functionality. The 9670G has a touch screen, therefore the user touches the desired line or on-screen button to obtain the desired result.

Note:

Web authors should be aware of the possible Line Buttons on the right side of the display, and should design their pages to align Web page text as necessary.

- For all telephones except the 9610, the top display area consists of a Top Line, a Title Line, and a Prompt Line for messages, text entry prompts, and page titles. The 9610 has a Top Line only on which titles can be displayed.

- The bottom display line presents the available softkey selections. Four softkeys can be displayed at one time for all phones except the 9610 and 9670G. The 9610 can display only two softkeys, while the 9670G can display up to five softkeys.
- With the exception of the 9670G, each telephone has a navigation cluster consisting of:
 - **Left** and **Right** navigation buttons to navigate back to a previous page and forward to another page, if one exists,
 - **Up** and **Down** navigation buttons to scroll up and down, one line or one page at a time, and,
 - An **OK** button functionality similar to a call appearance (Line button). The **OK** button cannot be used to navigate to a specific line, but provides an alternate way to select a line, a field, or an item.
- The 9670G has a touch screen and navigation is touch-based.

When this document states that an area is **selected or highlighted**, this means the area is displayed in reverse video. The content of the area is otherwise unchanged. On the 9670G, if a rendered WML element is touched, the element will be highlighted and activated. If the Next Field touch object on the On-Screen Keyboard is enabled and touched, the next <input> element on the card will be activated, cycling back to the beginning of the card when the end of the card has been reached.

For all telephones except the 9670G, which has a right-side scroll bar, the frame in which a Web page displays sits next to the left scroll bar area and flush with the right side of the display. The telephone determines the last horizontally addressable pixel and the last vertically addressable pixel.

For H.323 software Releases 3.1 and later and SIP software Releases 2.5 and later, one WML deck will be stored in volatile memory, and if any operation (except a refresh, Home key activation, or expiration of the WMLIDLETIME timer) requires the rendering of content from that deck, the cached version will be used and the deck will not be re-downloaded.

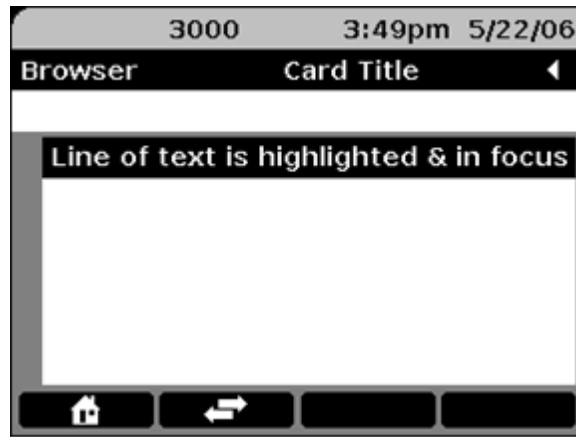
Display Area Specifications

The information in this section applies to all 9600 Series IP Telephones except the 9610, which has a smaller display area and the 9670G, which has a larger display area.

The Web browser renders text and icon content in the Application Area, between the top of the top Application Line and the bottom of the bottom Application Line. Image content may extend across Application Lines. WBMP or JPEG images can appear in the softkey label area. A maximum number of 96 lines per card is supported. A line corresponds to text rendered in the usable area of the display screen that contains application lines.

[Figure 21](#) depicts how the Web browser renders content.

Figure 21: Content Rendering



[Figure 22](#) is a schematic of the display area divided by function for all telephones except the 9610 and 9670G IP Telephones.

Figure 22: Display Area Breakdown (non-9610, non-9670G)

| | | |
|------------------|----------------------------|------------------|
| Top Area | Top Line | |
| | Title Line | |
| | Prompt Line | |
| Application Area | S c r o l l | Application Line |
| | B a r | Application Line |
| | Softkey Labels | |

The Top Area is subdivided vertically into a Top Line, a Title Line, and a Prompt Line. Each top area line extends across the entire width of the display.

The Application Area displays application content, softkey labels, and a Scroll Bar, if needed.

Text and icon content always display on Application Lines. Application lines align with any Line Buttons or LEDs located to the right of the display.

For all but the 9670G, a Scroll Bar displays vertically to the left of the display if the content does not fit the viewable area; the 9670G displays the scroll bar to the right of the display. When displayed, the Scroll Bar extends from the top Application Line to the bottom Application Line, inclusive.

A single row of Softkey Labels is displayed below the Application Lines and the Scroll Bar (if present). Softkey Labels are positioned above the associated softkeys buttons.

The Title Line is the second line in the Top Area. The Title Line is comprised of the current application Title, a subtitle if the application provides one, and choice or Web paging indicators. The Title is left justified on the Title Line. The Title is followed by a minimum of two spaces, followed by the subtitle. The subtitle is centered in the remaining area to the right of the title and to the left of the choice indicators. If an icon such as the Page History Indicator exists, that icon is right justified. The 9610 IP Telephone does not have a Title Line.

If the Title Line text exceeds the pixels allocated for that line, any characters that follow the last whole character that fits are truncated.

The Prompt Line is the third line in the Top Area. The current application uses the Prompt Line to provide context-specific prompts, hints, explanations, help, or similar information. Application specific help messages are managed by each application. All Prompt Line text is left justified. The 9610 IP Telephone does not have a Prompt Line.

If text on the Prompt Line exceeds the pixels allocated for that line, any characters that follow the last whole character are truncated.

When Web browser is invoked to render a URL, the Title Line, Prompt Line, Application area and Softkey labels are immediately cleared to their default background color.

Variables set during browser initialization

The following variables are supported and will be set to the value specified below when the browser is first initialized:

| Variable Name | Value |
|---------------|--|
| IPADD | Value of the system parameter IPADD. |
| MACADDR | Value of the system parameter MACADDR. |
| MODEL | Value of the system parameter MODEL. |
| PHONEXT | Value of the system parameter NVPHONEXT. |

Changing the value of these variables via a <setvar> element is not allowed. The variables Xmap and Ymap are also supported for use in image maps.

Web Browser Navigation

Web navigation uses the Navigation cluster, the **OK** button, softkeys, and Line buttons, as illustrated in [Figure 23](#). Note that the 9670G does not have a Navigation cluster; navigation on the 9670G is touch-based.

Figure 23: Navigation Keys and Buttons (non-9670G)



Web screen navigation works as follows:

- One link is permitted per line. The link can be activated in one of two ways - pressing the right Line button, if available and if associated with a particular display line, or pressing the **OK** button if the line is highlighted. For the 9670G, touching the line button/link activates it.
- Line buttons, if available, or the **OK** button in the middle of the navigation cluster are used to select an option. For the 9670G, touching the desired line/option selects it.
- Line buttons, if available, or the **OK** button in the middle of the navigation cluster are used to activate a text entry box to allow entry of text. For the 9670G, the On Screen Keyboard is used for text entry.

- The WML card author can use the softkeys on the bottom of display for navigation as appropriate. For all telephones except the 9610 the default softkeys labels are **Home** and **Refresh**; for phones running SIP Software Release 2.2, 2.5, or later, **Home** displays as a default when WMLHOME has been set to a value. The default softkeys are represented by icons and are always displayed when a WML page has been fully loaded by the browser. By default, **Home** is displayed on the first softkey and **Refresh** is displayed on the second softkey, although they may shift to other softkeys if the WML card contains additional softkey definitions. The default softkeys will also be displayed in their default positions when a page error is displayed. The **Home** softkey, if pressed (or touched, for the 9670G), will cause the web page specified by WMLHOME to be displayed. The **Refresh** softkey, if pressed (or touched, for the 9670G), will cause the current page to be reloaded.
- A **Stop** softkey is displayed on the third softkey only during web page loading. If pressed/touched, the previously displayed page will be redisplayed.
- The default softkey labels are always presented and are represented by icons. The **More** softkey is assigned to the fourth key position (or the second position for the 9610), and will only be displayed if the Web author defines more than one softkey. The Web author can define additional softkeys and softkey labels. Additional softkey labels (if defined) are assigned to the initial softkey label positions (beginning with Softkey 1). [Figure 24](#) shows the default softkeys and [Figure 25](#) shows how author-defined softkeys take precedence over the default softkeys.

Figure 24: Default Softkeys

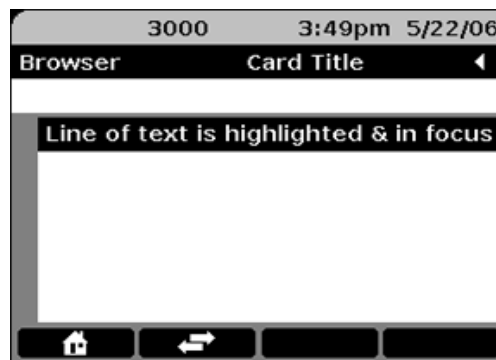
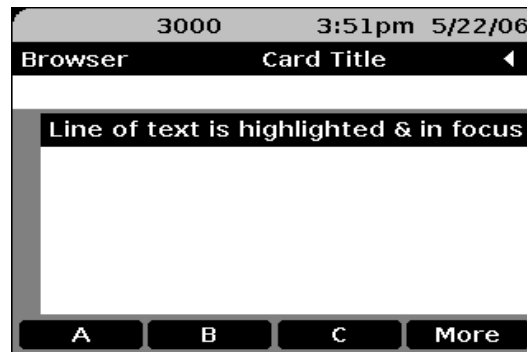





Figure 25: Author-defined Softkeys



- The **Left** navigation button takes the user to the previous card. The **Right** navigation button takes the user to the next card only if it is available in the history stack.
- The **Home** softkey () takes the user to the initial home Web page as conveyed in the settings.
- The **Refresh** softkey () reloads the same page on the screen.
- The **Stop** () softkey is only displayed as a default softkey when a page is loading. If **Stop** is pressed, page loading stops and the preceding page displays.
- Avaya recommends that the page author not include softkeys that are redundant with the default softkeys.

Multiple Paging Indicators

Except for the 9610 and 9670G, when the screen displays one of multiple pages in the Web history stack, right justified **Left** and **Right Arrows** appear on the Prompt Line. The **Left Arrow** indicates a previously viewed page is available in the history stack. The **Right Arrow** indicates the user can go forward to see the next page in the history stack. The user must press the **Left** or **Right** navigation button to view the previous or next page, as applicable. For the 9670G, **Left** and **Right** page control arrows display instead.

Scrolling

The Title Line displays the Card Title, which does not change as scrolling occurs. The Prompt Line displays title information for each line, which changes as the user scrolls the page and brings a different line into focus.

To move up and down on the lines of text currently displayed:

- The **Down** navigation button brings each line into focus, one line at a time starting with line 1 to last line above the softkeys. Each time a new line is brought into focus, you can show a new help message on the Prompt Line. Pressing and holding the **Down** navigation button moves down one screen at a time.
- The **Up** navigation button brings each line into focus, one line at a time starting from the last line above the softkeys to line 1. Each time a new line is brought into focus, you can show a new help message on the Prompt Line. Pressing and holding the **Up** navigation button moves up one screen at a time.
- The user can move up and down (bring “into focus”) displayed text as desired.

When a Web page is initially displayed, the card title is shown on the Title Line.

With the exception of the 9670G, if the **Right** or **Left** navigation button is pressed, the next or previous page displays. In this case, the Title Line displays the card title, rather than an element title. The Prompt Line is either empty if no line is in focus or displays any applicable help messages.

Each time a new line is brought into focus, a new title can display on the Prompt Line, if the title is included in the tag coding. When you code a tag without a title, the Prompt Line shows the default associated with that tag. The chart of Tags with Titles and Corresponding Title/Prompt Line Defaults provides Top Line default titles. [Chapter 7: Developing Web Pages for the Browser](#) and [Chapter 8: Web Applications](#) provide tag-specific information.

The user can move to additional text lines that do not fit on the current screen, to view all the text lines contained in one card. It appears to the users as if they are navigating to the next page of text, but they are really descending down six more lines of text on the currently displayed card.

Tags with Titles and Corresponding Title/Prompt Line Defaults:

| Tag | Attribute | Title/Prompt Line Default Values (Top Line for 9610) | Appears on |
|----------|--------------|--|---|
| <card> | title | New Card | Title Line |
| <anchor> | title | Use button to select Touch to select (9670 only) | Prompt Line; Top Line for 9610 only. |
| <a> | title | Use button to select Touch to select (9670 only) | Prompt Line; Top Line for 9610 only. |

| Tag | Attribute | Title/Prompt Line Default Values (Top Line for 9610) | Appears on |
|----------|-----------|---|--|
| <input> | title | Enter text between brackets Enter text (9670 only) | Prompt Line; Not applicable for 9610. |
| <option> | title | Use button to select Touch to select (9670 only) | Prompt Line; Top Line for 9610 only. |
| | alt | | |

Tags with Titles and Corresponding Application Line Defaults:

| Tag | Attribute | Title/Prompt Line Default Values (Top Line for 9610) | Appears in |
|-------|-----------|--|---|
| | alt | Image Not Rendered, i.e., "Image?" displays | The application area where the image should have been |

Note:

Since the 9610 IP Telephone does not have Title and Prompt Lines, any title tags appear on the Top Line but follow the same guidelines as with other 9600 Series IP Telephones.

The Title Line (or Top Line, for the 9610) displays the card title when the page is first displayed. The Prompt Line displays additional card titles when the user scrolls down the screen and comes across a text element (<p> tag or
 tag). When pages are downloading, **Loading** displays on the Title Line. The left side of the Title Line displays **Browser** as a default, which you can customize. For more information, see [WML Document Skeleton](#) and the corresponding Title change sample in that section. Horizontal scrolling is not supported.

Truncation Rules and Links

Truncating Lines and Words

- Wrapping capability is supported the same way on <p>, <a>, or <anchor> tags
- The browser breaks long lines of text into multiple lines. This is equivalent to setting the <p> mode to wrap. The default is to wrap the text.
- Long words that do not fit on an entire line should be hyphenated.

Links

- A maximum of one link can be displayed per line.
- Regular text will not be displayed on the same line as a link.
- A link causes an automatic line break to occur both before and after the link.
- Each link associated with <a> and <anchor> WML tags can take up the width of one display line. If the link would display wider than the line width and <p> mode is nowrap, the link is truncated at the point where it no longer fits on the line. If <p> mode is set to wrap, the link continues to the adjoining line if the text is too long.

Image Support

WBMP Images

The Web browser supports rendering of Wireless Bitmap (WBMP) images. WBMP is a graphic format optimized for mobile computing devices. At present WBMP supports only a simple picture format that is restricted to bi-level, black and white pixel images.

A WBMP image is identified using a TypeField value, which describes encoding information such as pixel and palette organization, compression, and animation. The TypeField value also determines image characteristics according to WAP documentation. An Image Type Identifier represents Type Field values. Currently, there is only one type of WBMP specified. The Image Type Identifier label for this is **0**.

This image type has the following characteristics:

- No compression
- One bit color (white=1, black=0)
- One bit deep (monochrome)

WBMP is part of the Wireless Application Protocol, Wireless Application Environment Specification Version 1.1. The WBMP specification is available at: <http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html>.

Many utilities are available as shareware to convert other graphical file formats to the WBMP format.

JPEG Images

In addition to WBMP, the Web browser supports renderings of JPEG images. JPEG stands for the Joint Photographic Experts Group, and is a digital image file format designed for maximal image compression. JPEG uses “lossy” compression in such a way that, when the image is decompressed, the human eye does not find the loss too obvious. The amount of compression is variable and the extent to which an image may be compressed without too much degradation depends partly on the image and partly on its use. JPEG is designed for compressing either full-color or gray-scale images of natural, real-world scenes. It works well on photographs, naturalistic artwork, and similar material. JPEG does not work as well on lettering, simple cartoons, or line drawings. JPEG handles only still images.

Image Rendering

The mime type for the WBMP is image/vnd.wap.wbmp. The mime type for the JPEG image is as follows:

| Extension | Mime Type |
|-----------|------------|
| jpe | image/jpeg |
| jpeg | image/jpeg |
| jpg | image/jpeg |

Without a mime type, the Web servers cannot process JPEG files.

Text is not rendered on the same line as an image. If a page author attempts to include both text and image on the same line, the image is rendered first, followed by the text.

Linked Images

Images can be a part of a link and be selectable. An image inside an <a> or an <anchor> tag can be selected to activate a <go> task. For information see [Anchor Elements](#).

The browser follows the WML 1.3 specs which allow images to be used as hyperlinks. The `img` element can be contained within the following elements: `a`, `anchor`, `p` and `do` tags for softkeys as described in [Chapter 7: Developing Web Pages for the Browser](#).

Number of Images Supported

A maximum of 16 images can be displayed per wml file, in either the middle content area or softkey area. For any additional images that exceed this maximum number, the “alt” text is presented.

Enabling Text Entry

Use these guidelines to enable text entry. See [Access Key Input Mode \(AIM\)](#) in [Chapter 7: Developing Web Pages for the Browser](#) for information about an alternate text entry method using access keys.

Note:

The 9610 IP Telephone Web browser does not support text entry as described in this section, but does support [Access Key Input Mode \(AIM\)](#).

1. When a card displays, the prompt “**Enter text here**” indicates a text entry area.
2. To start text entry for all but the 9670G phone, the user either presses the right Line button associated with the text line/field or uses the **Up** or **Down** navigation button(s) and **OK** to bring the line/field into focus. Either action causes a cursor that indicates where to begin text entry to replace the text entry prompt. For the 9670G, the user touches the desired line/field (scrolling to it first if necessary) and the on-screen keyboard displays to allow text entry.
3. Text editing softkeys display at the bottom of the screen to assist the user with text entry or for the 9670G, on the on-screen keyboard.
4. The user exits text entry mode by:
 - Pressing the Line button associated with the line on which text entry occurred to de-focus the line for all but the 9670G phone,
 - Pressing any other Line button for all but the 9670G phone,
 - Pressing a **Done** softkey at the bottom of the screen (or for the 9670G, on the on-screen keyboard), or
 - Pressing the **OK** button for all but the 9670G.

5. To re-enter text entry mode, the user either presses the Line button associated with the text line/field or scrolls to the desired line and presses **OK**. Text entry can proceed when a cursor appears to the right of the existing text. The **Clear** softkey is used to delete existing text, as appropriate. For the 9670G, the user touches the desired line/field to display the on-screen keyboard, using the **Done** or **Clear** softkey(s) as applicable.
6. A page author can use the **ivalue** attribute to customize text to assist a user in text entry. For example, the author can prompt "Enter name here," "Enter phone number here," or "Enter email address here."
7. On-hook keypad dialing to make a phone call is not allowed when in Text Entry mode. If the user is already off-hook, text entry is allowed. When the phone is again placed on-hook, text entry can be re-enabled at the point at which it was disabled. When text entry is active, the user can be active on a call or receive a call, but cannot use the keypad to dial a number or for interactive voice recognition (IVR) prompts. In this case, the user must disable text entry prior to using the keypad to dial a call or for IVR prompts.

Text Entry Example:

An `<input>` tag used for a text entry box has these attributes:

- **title** - The title displays on the Prompt Line.
- **value** - A value displays in the text entry box rather than the standard "Enter text here" prompt. When the user selects the line or brings the line in focus, the cursor displays at the end of the value. When the value field is empty, focusing in erases the "Enter text here" prompt and places the cursor at the beginning of the entry field. The **value** attribute takes precedence over an **ivalue** attribute, as explained in the examples that follow.

Note:

See [Chapter 7: Developing Web Pages for the Browser](#) and [Chapter 8: Web Applications](#) for specific information on the **input**, **title**, **value**, and **ivalue** tags.

Here is how the user experiences a value during text entry:

"Hello there" is displayed on the Prompt line for the tag `<input title="Hello there"/>`
- The text box displays **Enter text here** as a default. No value is present, meaning the web page does not provide a default input value. Upon text entry, the value equals the typed-in text, and the "Enter Text here" prompt no longer displays.

"Hello there" is displayed on the Prompt line for the tag `<input title="Hello there" ivalue="Fill in the blanks"/>` - The text box displays **Fill in the blanks**, which is replaced by any text input by the user.

For the tag `<input title="Enter quantity" value="1"/>`, "Enter quantity" is displayed on the Prompt line and "1" is displayed in the input field. If the user wants a different value, the "1" must first be deleted.

Text Editing Modes

When a text entry area is enabled, these softkey labels display centered above the actual softkeys for all but the 9670G IP Telephone, which instead uses the on-screen keyboard for editing text:

| | | | |
|-------------|-------------|---------------|-------------|
| Bksp | Done | Cancel | More |
|-------------|-------------|---------------|-------------|

Note:

An illustration of the 9670G's on-screen keyboard appears at the end of this section. The 9670G user guide describes entering and editing text using the on-screen keyboard.

The labels appear at the bottom of the display and are activated by pressing the softkey buttons directly below them.

The **Bksp** softkey is used for destructive backspacing during text entry. Pressing this softkey deletes text that is backspaced over. This softkey can only be used if there is a character to the left of the cursor. Text to the right of the cursor moves left with the cursor.

The **Done** softkey exits text entry mode. When **Done** is pressed, several actions occur:

- the line containing the text entry area no longer displays as in focus,
- the cursor no longer displays, and
- the previous softkey labels and operation are restored.

The **Cancel** softkey exits edit mode without changing the content of the field currently in focus.

Selecting **More** causes another line of softkeys to replace the current line:

| | | | |
|--------------|---------------|------------|-------------|
| Clear | Symbol | ABC | More |
|--------------|---------------|------------|-------------|

The **Clear** softkey appears only when an in focus field contains a value, and deletes the contents of that field.

The **Symbol** softkey causes a set of special characters to display. The symbol selected displays in the cursor position.

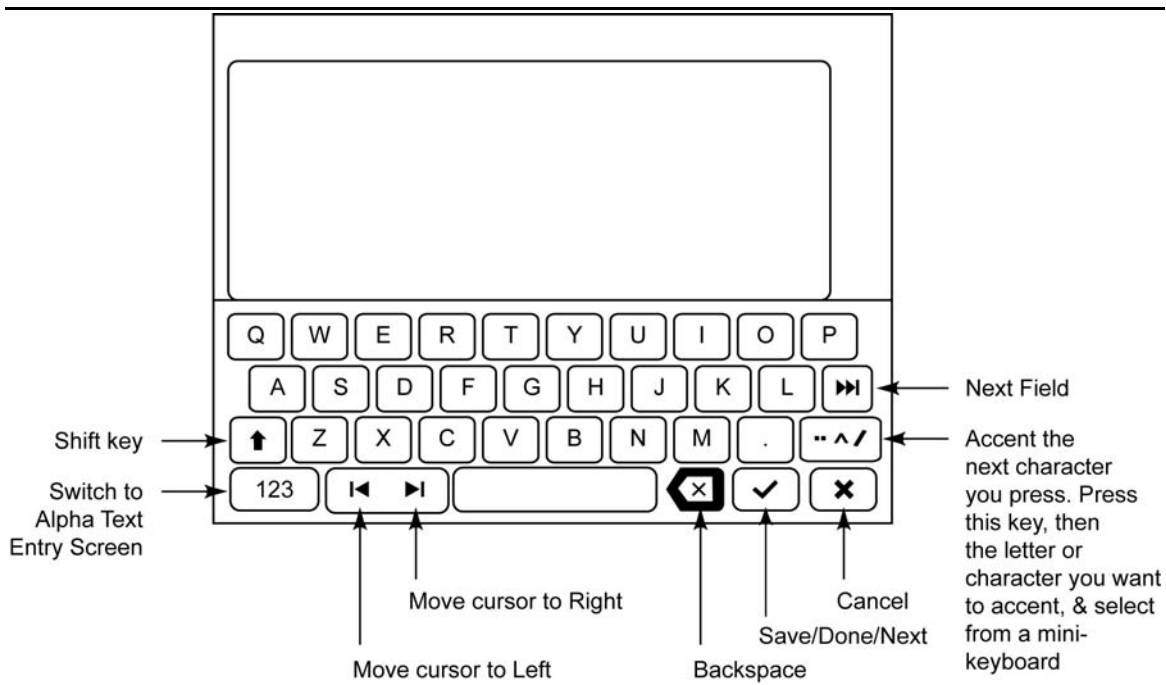
The **ABC** softkey offers the option to change case as well as to go from alphabetic character entry to numeric character entry. Selecting this softkey causes the character to the right of the cursor to cycle from uppercase to lowercase to Title case to numeric. Selecting a different case or mode displays that choice in the softkey label field. For example, changing from upper to lowercase changes the softkey label from **ABC** to **abc**. Changing to title case causes the softkey to change again to **Abc**. Changing to numeric changes the softkey label to **123**.

The **More** softkey on the second group of softkey labels changes the labels back to **Bksp**, **Done**, and **Cancel**.

Note:

The * and # dial pad keys can be used to enter the * and # characters in any text mode. The cursor automatically advances after one of these characters is entered.

Figure 26: On-screen Keyboard for 9670G IP Telephone Text Entry:



Character Set Support

Character set support differs depending on the IP telephone model. The Web browser supports ISO-8859-1 (Latin 1) encoded WML for any languages available on a specific telephone model. For 9600 Series IP Telephones running SIP software Release 2.2, 2.5, or later, WML encoded by the UTF-8 encoding of Unicode is supported.

Web History Stack

The Web History Stack is a list of URLs that starts from the point the user initiates the browser session for the first time. Each time a card or an error message is rendered except for a Refresh operation, the URL is added to the next position in the stack.

Up to 100 URLs can be stored in memory. Any URLs exceeding the allowable maximum are deleted in the order stored in memory, meaning the first URLs stored are the first URLs deleted.

Requirements for Deck/Card Elements

Valid WML elements include the following:

| Type of Elements | Element |
|------------------------------------|--|
| Deck/card elements | wml, card, template, head, access, meta |
| Event elements | do, ontimer, onenterforward, onenterbackward, onpick, onevent, postfield |
| Tasks | go, prev, refresh, noop |
| Variables | setvar |
| User input | input, select, option, optgroup |
| anchors, Images, Timers, Variables | a, anchor, img, setvar, timer |

Unsupported tags (not well-formed) cause the error message **Page cannot be rendered** to appear in the Prompt Line. The error message **Page contains invalid tags** displays on the Prompt Line if the XML parser fails.

Wireless Telephony Applications (WTA)

Wireless Telephony Applications (WTA) are those applications designed to interact with the telephony-related functions present in a phone. These applications include:

- Originating a call ("click to dial")
- Adding a Contacts entry

Note:

The 9610 Web browser supports only the "click to dial" WTA function.

The Web browser supports the WTAI application “click to dial” any link on the screen. With the exception of the 9610, a telephone handset icon displays to the left of a “click to dial” link when the link first displays.

The **Add to Phonebook** WTAI function “wtai://wp/ap;” adds a name and number to the telephone Contacts application. A Contacts icon displays to the left of an “add to phonebook” link. The wtai syntax is supported as an **href** attribute. As such, any tags supporting the **href** attribute can use the “add to Contacts” function. These tags are <a>, <anchor>, , <do>, <onevent>, <select>, <option>, and <optgroup>.

When a user activates the **Add to Phonebook** function, the Web browser transfers the name and number to the Contacts application. The user can edit the entry according to the current Contacts application functionality.

Syntax Implementation

Click to Dial Functionality

To enable the click to dial functionality, use the following syntax:

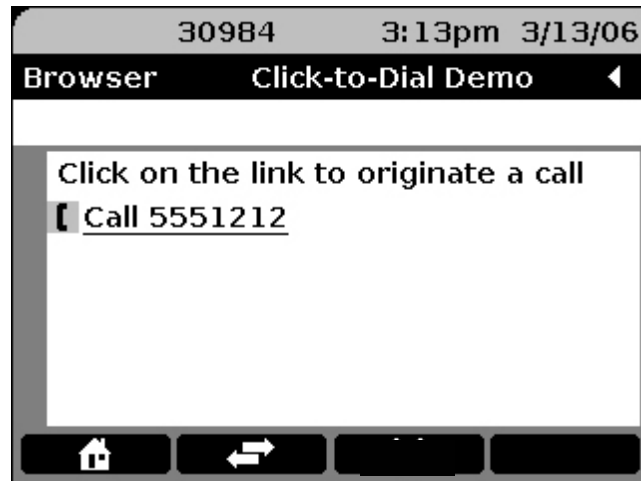
wtai://wp/mc ; *number*

You can embed this code into any valid WML tag that implements href or a hyperlink by associating these tags with a <go> tag. Examples of tags you can associate with the <go> tag are the <a> tag, <anchor>, <do>, <option>, or <onevent> tags.

Click-to-dial using <a> tag:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
    "http://www.wapforum.org/DTD/wml13.dtd">
<wml>
<card id="callid1" title="Click-to-Dial Demo">
    <p>
        Click on the link to originate a call
        <a href="wtai://wp/mc;5551212">Call
            5551212</a>
    </p>
</card>
</wml>
```

The generated code is rendered as the following diagram:



The code shows a hyperlink as **Call 5551212** on the Web screen of an Avaya 9620 IP Telephone. A phone icon precedes this hyperlink, indicating the hyperlink is a “click-to-dial” number. When a user selects this link, the phone dials the string “5551212” or any phone number that follows a semicolon in the WTAI code.

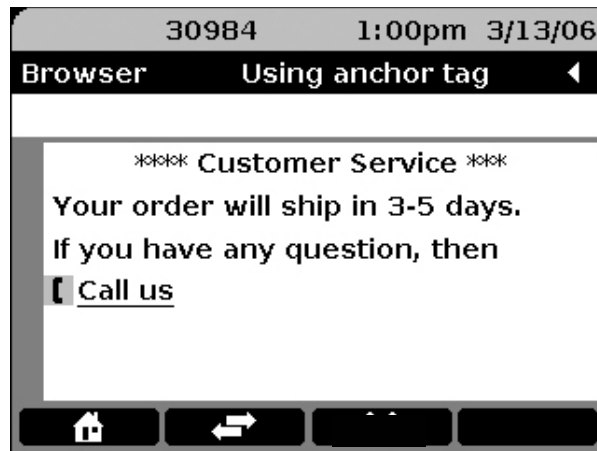
Note:

A phone icon is generated only when an `<a>` tag or `<anchor>` tag is used.

Click-to-dial using `<anchor>` tag:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
    "http://www.wapforum.org/DTD/wml13.dtd">
<wml>
    <card id="callid1" title="Using anchor tag">
        <p>
            <p align="center">***Customer Service***</p>
            <p>
                Your order will ship in 3-5 days.
                If you have any questions, then
                <anchor>Call us
                <go href="wtai://wp/mc;5551212"/>
                </anchor>
            </p>
        </card>
    </wml>
```

The generated code is rendered as the following diagram:

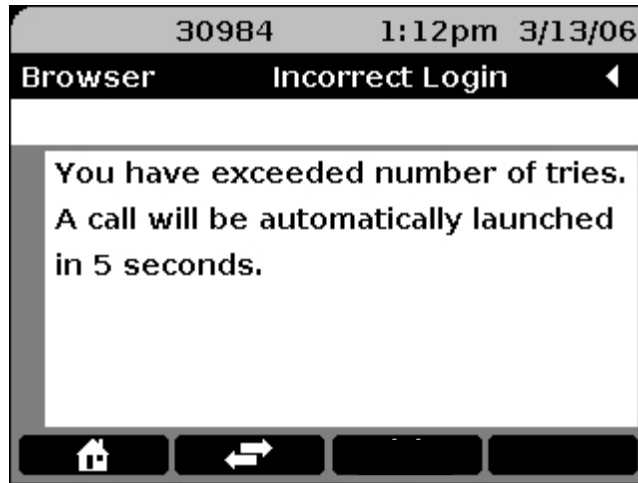


The code shows a hyperlink as **Call Us** on the Web page. When a user selects this link, the phone dials the string “5551212” or any number that follows a semicolon in the WTAI code.

Click-to-dial using <onevent> tag:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
    "http://www.wapforum.org/DTD/wml13.dtd">
<wml>
  <card id="callid3" title="Incorrect Login">
    <onevent type="ontimer">
      <go href="wtai://wp/mc;+ 1888 555 1212"/>
    </onevent>
    <timer value="50"/>
    <p>
      You have exceeded number of tries.
      A call will be automatically launched in 5
seconds.
    </p>
  </card>
</wml>
```

The generated code is rendered as the following diagram:



The code automatically dials the number "1 888 555 1212" after 5 seconds, once the Web page loads.

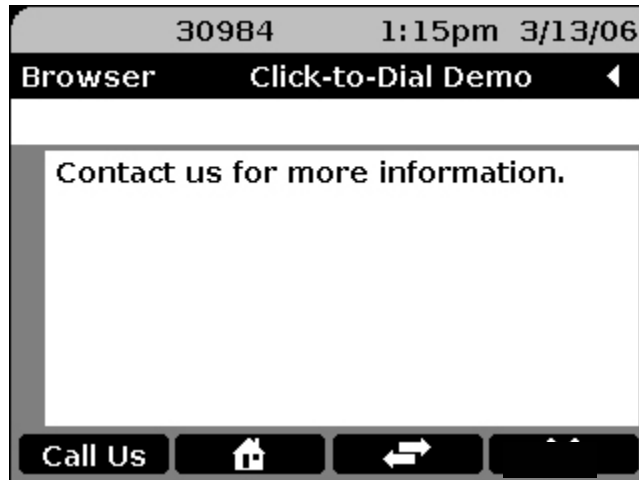
Note:

When using international numbers with click to dial functionality, you must include a '+' before the country code. You must include a space following the country code, but before the national number. The enhanced local dialing rules will then be handled correctly. See the applicable *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephone Administrator Guide* for configuration information.

Click-to-dial using <do> tag (softkey):

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
    "http://www.wapforum.org/DTD/wml13.dtd">
<wml>
    <card id="callid4" title="Click-to-Dial Demo">
        <p>
            Contact us for more information.
        </p>
        <do type="accept" label="Call Us" name="dotag1">
            <go href="wtai://wp/mc;+1 8005552525"/>
        </do>
    </card>
</wml>
```


The generated code is rendered as the following diagram:



Add to Contacts Functionality

Add to Contacts is referred to as Add to Phone Book by WTAI. When a user clicks Add to Contacts, the Web browser transfers the name and number to the telephone's Contacts application. The Contacts application allows users to edit and save the entry to their Contacts list.

To enable the add to Contacts functionality, use the following syntax:

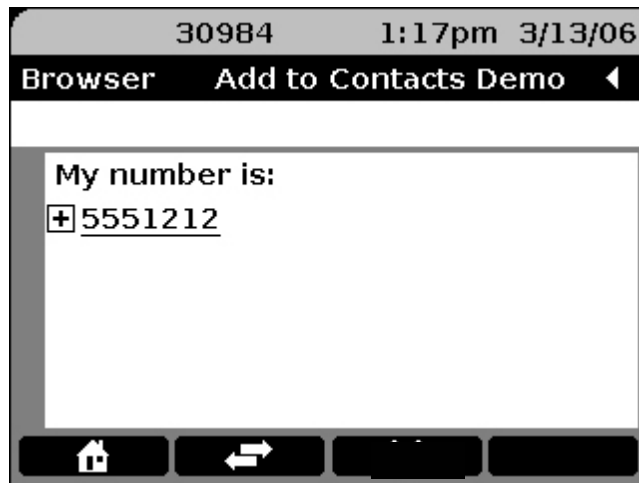
wtai://wp/ap;number;name

This code can be embedded into any valid WML tag that implements href or a hyperlink by associating these tags with a <go> tag. Examples of tags you can associate with the <go> tag are the <a> tag, <anchor>, <do>, <option>, or <onevent> tags.

Add to Contacts using <a> tag:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
    "http://www.wapforum.org/DTD/wml13.dtd">
<wml>
    <card id="addap1" title="Add to Contacts Demo">
        <p>
            My number is:
            <a href="wtai://wp/ap;5551212;My
Company">5551212</a>
        </p>
    </card>
</wml>
```

The generated code is rendered as the following diagram:



The code adds the entry to the Contacts group with the name “My Company” on the telephone’s Contacts screen. A **Save** icon precedes this hyperlink, indicating the hyperlink is an “add to Contacts” number. When the user selects this link, the Web browser transfers the name and number, for example, “My Company” and “5551212,” to the telephone’s Contacts application. Users can then edit and save the entry to their Contacts list.

Note:

A Save icon is generated only when an `<a>` tag or an `<anchor>` tag is used.

Button-Push URI's

For H.323 9600 Series IP Telephones running software Release 3.1 or later, a "button" URI scheme is supported that uses the following syntax:

- button:*ButtonName*
- where *ButtonName* corresponds to a button on the telephone.

When an element containing an href attribute with a value that begins with a "button" scheme is activated, it has the same effect as if the corresponding button on the telephone had been pushed. If the telephone does not have a corresponding button, activating the element has no effect.

The following *ButtonName* values are supported:

| Button Name | Description |
|-------------|---|
| Phone | Corresponds to the Phone button. The 9610 does not have a Phone button. |
| A | On the 9610, corresponds to the Home button. On the 9670, corresponds to the A/Home button. On all other 96xx telephones, corresponds to the A/Menu button. |

Administering the Avaya Menu for the Web Browser

Customizing the Avaya Menu to display a Browser entry and/or Web application links might require collaboration between the developer and the system administrator. The applicable (H.323 or SIP) *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Administrator Guide* describes Avaya Menu administration and includes sample screen illustrations for several scenarios. Refer to the appropriate administrator guide for detailed information.

Web-Related System Parameters

This section describes Web-related system parameters, values, and validation rules.

The interaction between the telephone and the Web application involves several system parameters. Each is listed, followed by specific processing requirements of note to Web developers.

| System Parameter | Description, Purpose, Dependencies, and Guidelines |
|------------------|---|
| AMADMIN | For telephones running H.323 protocol, the URI used to obtain the AvayaMenuAdmin.txt file, which can be used to provide access to WML applications under the A (AVAYA) Menu (or, for the 9670G, the Home screen). Specify the HTTP server and directory path to the administration file. Do not specify the administration file name. AMADMIN does not apply to 9610 menu administration, nor does it apply to telephones running the SIP protocol. |
| IDLEAPP | Applicable only to the 9610 IP Telephone. If a Web Idle Timer (WMLIDLETIME) has been administered and that timer expires, the 9610 display changes to the Idle application specified here. The Idle application is either one of the existing local applications ("Menu" or "Directory"), or a URL, depending on the contents of the system value IDLEAPP. |
| WMLEXCEPT | Web application HTTP proxy server exception domains. These are domains the proxy server will not use. The WMLEXCEPT value is a list of one or more domains, separated by commas without any intervening spaces (up to 127 total ASCII characters, including commas). Set this parameter only if a proxy server is used and if there are exception domains (domains for which the proxy server will not be used). |
| WMLHOME | URI of the Web application home page for all 9600 IP Telephones except the 9610. Zero or one URL (0-255 ASCII characters, including spaces, if any). If the system value WMLHOME is null, the telephone will not display the Browser label under the "A" Avaya Menu. If WMLHOME is null, a display push is allowed, but the Home softkey does not display and the user cannot access any other Web pages. |

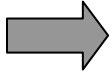
| System Parameter | Description, Purpose, Dependencies, and Guidelines |
|------------------|--|
| WMLIDLETIME | Idle time before displaying the Web page. The number of minutes of inactivity after which the Web browser will be displayed if WMLIDLEURI is not null. The default is 10 minutes. Valid values range from 1 to 999 minutes (16.65 hours). When the idle timer reaches the number of minutes equal to the WMLIDLETIME value, the telephone sends an HTTP GET for the URI specified by the system value WMLIDLEURI. The response is displayed in the Web page display area. |
| WMLIDLEURI | Idle time Web page URI. The URI that specifies the Web page the browser displays after an idle interval as specified by WMLIDLETIME. Value: Zero or one URI (0-255 ASCII characters, no spaces). Null is valid but if Null, no page displays. Avaya recommends that WMLIDLEURI be specified for telephones in public areas through the use of a GROUP parameter. The idle timer is only reset if WMLIDLEURI is non-null such that an HTTP GET can be sent. Do not use this parameter for 9610 IP Telephone(s); use IDLEAPP instead with WMLIDLETIME. |
| WMLPORT | TCP port number for the HTTP proxy server, if applicable (1-5 ASCII numeric characters from "0" to "65535"). Null is not a valid value. |
| WMLPROXY | One HTTP proxy server IP Address in dotted decimal or DNS Name format (0-255 ASCII characters). Set this parameter only if Web pages requiring a proxy server will be supported or if the 9670G Avaya applications will be used. If WMLPROXY is null, or if WMLPROXY cannot be resolved into a valid IP Address, an HTTP proxy server is not used. If WMLPROXY is resolved into a valid IP Address and WMLEXCEPT is null, the HTTP proxy server is used for all Web transactions. If WMLEXCEPT is not null, the HTTP proxy server is only used for URLs whose domains are not on the WMLEXCEPT list. |
| WMLSMALL | 9610 Web browser (only). Zero (0) to 255 ASCII characters. Zero or one URL. Other 9600 Series telephones use WMLHOME instead. Set this value in the 46xxsettings file and not the AvayaMenuAdmin.txt file used for other 9600 Series IP Telephone settings. If WMLSMALL is null, the telephone will not display the Web links in the Main menu. |

Note:

If the Web pages accessed by the telephone are completely on the customer's intranet, WMLPROXY, WMLPORT and WMLEXCEPT need not be set. If WMLPROXY is null, the values of WMLPORT and WMLEXCEPT do not matter. WMLPROXY will be used only in WML- and Push-related HTTP message exchange; WMLPROXY should not be used for backup/restore and AMADMIN-related HTTP operations.

Chapter 7: Developing Web Pages for the Browser

Introduction



The information in this chapter applies to both H.323 and SIP 9600 Series IP Telephones.

This chapter:

- Describes HTTP authentication, protocol, header information, and error messages.
- Presents the WML portions implemented in the 9600 Series IP Telephone Web Access applications. Any limitations or non-standard implementations are mentioned.
- Provides considerations to develop effective Web pages for browser viewing.
- Describes special considerations for developing Web pages for the 9610 IP Telephone, which supports a subset of the WML Browser used by all other 9600 Series IP Telephones. To develop Web pages for the 9610, see [Developing Web Pages for the 9610 IP Telephone](#).
- Describes support for Cascading Style Sheets.

This chapter is not intended to provide technical details on setting up a Web server, nor does it provide information on Web server technologies.

HTTP Support

The browser uses an HTTP client to communicate with Web servers and supports HTTP 1.1. The browser supports WML 1.3 (WML June 2000). For more information, see <http://www.wapforum.org/>.

For 9600 Series IP Telephones running [H.323] software Releases S2.002 and S3.0 or greater (only), HTTP over TLS/SSL is supported and negotiation to SSL 3.0 is allowed for browser operation.

HTTP Authentication

An authentication input screen displays when a response code of 401 (Unauthorized Response) or a response code of 407 (Proxy Authentication Required) is received. The authentication input screen has a text entry area for the user to enter a name and password, and two softkeys labeled **Submit** and **Cancel**. The words **HTTP Authentication** appear on the Prompt Line.

Note:

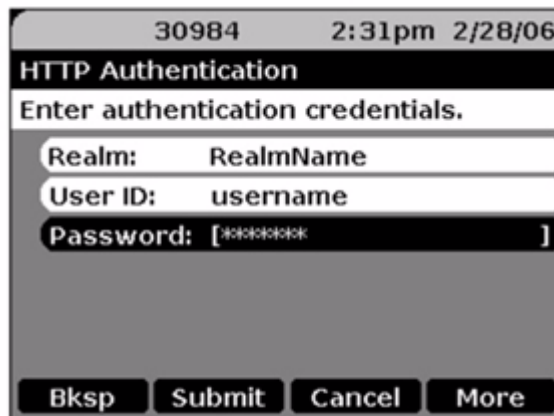
Any type of Push message sent to the phone while the HTTP Authentication screen is displayed will be rejected.

The **Submit** softkey is enabled whenever there is at least one character in one of the input fields. If the **Submit** softkey is selected, the original HTTP request is reissued and contents of the user-input fields used to generate the appropriate authentication header.

The **Cancel** softkey is always enabled. If **Cancel** is selected, the contents of the user input controls are discarded, and the previous screen redisplay.

Standard text entry and text entry for passwords is supported.

The authentication screen is as follows:



Note:

Because it does not support text entry, the 9610 IP Telephone does not support HTTP authentication. If the server sends the authentication the 9610 displays the error message "Incorrect WML syntax."

Once the user logs out, the login credentials are cleared from the phone.

HTTP Protocol

The Web browser uses an HTTP client to communicate with Web servers.

HTTP Header - User Agent

The User-Agent request-header field contains the Web browser's signature. This is for:

- tracking requests,
- statistical purposes,
- tracing protocol violations, and
- automated user agent recognition to tailor responses to avoid particular user agent limitations.

User agents should include the User-Agent request-header field with requests. The field can contain multiple product tokens and comments identifying the agent (browser) and any sub-products that form a significant part of the user agent. By convention, the product tokens are listed in order of their significance in identifying the application.

```
User-Agent = "User-Agent" ":" 1*(product | comment)
```

```
Example: User-Agent: CERN-LineMode/2.15 libwww/2.17b3
```

The User-Agent header is included in all HTTP messages initiated by the browser, formatted as specified below for the appropriate IP telephone. The User-Agent header will be included in all HTTP messages initiated by the browser, formatted as follows:

User-Agent:

```
AVAYA/SoftwarePlatform/vMajorSoftwareVersion+(model4)/Std/  
MinorSoftwareVersion
```

where:

- *SoftwarePlatform* is "SPICE" for the H.323 96xx software platform, and "Spark" for the SIP 96xx software platform.
- *MajorSoftwareVersion* is the software version number,
- *model4* is the first four characters of the telephone's model identifier
- *MinorSoftwareVersion* is any finer-grained internal software version identifier.

For example, the User-Agent header for a 9620D01A running SPICE software release 1.0 (minor release 0.1) would be:

```
User-Agent: AVAYA/SPICE/v1.0+(9620)/Std/0.1
```

HTTP Error Messages

The Web browser supports the following HTTP standard Error Messages. The 4xx Status Codes indicate a client error in which the request contains bad syntax or cannot be fulfilled. The 5xx Status Codes indicate a server error in which the server failed to fulfill an apparently valid request.

Note:

Unsupported WML 1.3 tags are not rendered, and do not cause the browser to fail. Unknown tags and misspelled tags generate error messages.

Table 14: HTTP Error Messages

| Status Code | Failure Message | Meaning |
|-------------|-------------------------------|--|
| 400 | Bad Request | The request could not be understood by the server due to incorrect syntax. |
| 401 | Unauthorized | The request requires user authentication. This message triggers display of the HTTP Authentication screen. |
| 402 | Payment Required | |
| 403 | Forbidden | The server has indicated it will not respond to your query. |
| 404 | Not Found | The server has not found anything matching the request. |
| 405 | Method Not Allowed | |
| 406 | Not Acceptable | |
| 407 | Proxy Authentication Required | The request requires user authentication. This message triggers display of the HTTP Authentication screen. |
| 408 | Request Time-out | |
| 409 | Conflict | The request could not be completed due to a conflict with the current state of the resource. |
| 410 | Gone | The requested resource is no longer available. |
| 411 | Length Required | The server refuses to accept the request. |
| 412 | Precondition Failed | |

| Status Code | Failure Message | Meaning |
|-------------|---------------------------------|--|
| 413 | Request Entity Too Large | The server refuses to accept the request because the request entity is too large. |
| 414 | Request-URI Too Large | The request-URI is longer than the server can interpret. |
| 415 | Unsupported Media Type | The server refuses to accept the request because it is in an unsupported format. |
| 416 | Requested range not satisfiable | |
| 417 | Expectation Failed | |
| 500 | Internal Server Error | The server encountered an unexpected condition that prevented it from fulfilling the request. |
| 501 | Not Implemented | |
| 502 | Bad Gateway | The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed attempting to fulfill the request. |
| 503 | Service Unavailable | The server is currently unable to handle the request due to a temporary overload or maintenance. |
| 504 | Gateway Time-out | The server, while acting as a gateway or proxy, did not receive a timely response from the upstream server specified. |
| 505 | HTTP Version not supported | The server does not support, or refuses to support, the HTTP protocol version used in the request message. |

Note:

When the client server receives a Status Code 431, it safely assumes there was something wrong with its request and treats the response as if it had received a Status Code 400.

Summary of WML Tags and Attributes

The [Table 15: Summary of WML Tags and Attributes](#) that follows summarizes the detailed tag-related information provided in each remaining section in this chapter.

Table 15: Summary of WML Tags and Attributes

| Tag | Attribute | Supported? |
|----------|-----------------|------------|
| <a> | | Yes |
| | accesskey | Yes |
| | href | Yes |
| | title | Yes |
| | style | Yes |
| <access> | | No |
| <anchor> | | Yes |
| | accesskey | Yes |
| | title | Yes |
| | style | Yes |
| <area> | | Yes |
| | coords | Yes |
| | href | Yes |
| | shape | Yes |
| | | No |
| <big> | | No |
| | | Yes |
| <card> | | Yes |
| | newcontext | Yes |
| | onenterbackward | Yes |
| | onenterforward | Yes |
| | ontimer | Yes |
| | ordered | No |
| | style | Yes |
| | title | Yes |

| Tag | Attribute | Supported? |
|------------|----------------|-------------------------------|
| <do> | | Yes |
| | label | Yes |
| | name | Yes |
| | optional | Yes |
| | type | Yes |
| | | No |
| <fieldset> | | No |
| <go> | | Yes |
| | accept-charset | Yes |
| | href | Yes |
| | method | Yes |
| | sendreferer | Yes |
| <head> | | No |
| <i> | | No |
| | | Yes |
| | align | Only left alignment supported |
| | alt | Yes |
| | height | No |
| | hspace | Yes |
| | localsrc src | No |
| | style | Yes |
| | vspace | Yes |
| | width | Yes |
| | style | No |
| | | Yes |

| Tag | Attribute | Supported? |
|------------|-----------------|------------|
| <input> | | Yes |
| | accesskey | No |
| | autoselect | Yes |
| | ivalue | Yes |
| | emptyok | Yes |
| | format | Yes |
| | inputformat | Yes |
| | maxlength | Yes |
| | name | No |
| | size | Yes |
| | style | No |
| | tabindex | Yes |
| | title | Yes |
| | type | Yes |
| | value | Yes |
| <map> | | Yes |
| <meta> | | Yes |
| | content | Yes |
| | name | Yes |
| | forua | No |
| | http-equiv | No |
| | scheme | No |
| <noop> | | Yes |
| <onevent> | | Yes |
| | onenterbackward | Yes |
| | onenterforward | Yes |
| | onpick | Yes |
| | ontimer | Yes |
| <optgroup> | | Yes |
| | style | Yes |
| | title | Yes |

| Tag | Attribute | Supported? |
|-------------|-----------|------------|
| <option> | | Yes |
| | onpick | Yes |
| | style | Yes |
| | title | Yes |
| | value | Yes |
| <p> | | Yes |
| | align | Yes |
| | mode | Yes |
| | style | Yes |
| <postfield> | | Yes |
| | name | Yes |
| | value | Yes |
| <prev> | | Yes |
| <refresh> | | Yes |
| <select> | | Yes |
| | iname | Yes |
| | ivalue | Yes |
| | multiple | Yes |
| | name | Yes |
| | style | Yes |
| | tabindex | No |
| | title | Yes |
| | value | Yes |
| <setvar> | | Yes |
| | name | Yes |
| | value | Yes |
| <small> | | No |
| | | No |
| <table> | | No |
| <td> | | No |

| Tag | Attribute | Supported? |
|------------------------|-----------------|------------|
| <template> | | Yes |
| | onenterbackward | Yes |
| | onenterforward | Yes |
| | ontimer | Yes |
| | style | Yes |
| <timer> | | Yes |
| | name | Yes |
| | value | Yes |
| <tr> | | No |
| <u> | | No |
| <wml> | | Yes |
| {Universal Attributes} | xml:lang | No |
| | class | No |
| | id | Yes |

WML Protocol

The Web browser renders pages with spaces before the XML prolog. The strict rules of XML specify that pages with any characters before the XML prolog, including spaces, are invalid and should be rejected. However, many WML pages available on the World Wide Web contain such spaces. To facilitate compatibility with existing content, the Web browser relaxes this constraint.

WML Tags and Attributes

WML Document Skeleton

Certain tags define the basic framework of a WML document. The tags listed below make up the basic skeleton of a WML document. The designated IP telephones support these tags unless otherwise indicated in the [Table 15: Summary of WML Tags and Attributes](#).

Common tag attribute: **id**. The attribute **id** is a universal attribute associated with each WML element.

<wml> tag - The <wml> tag defines a deck of cards and encloses all information about the deck. This tag is a required WML element, must contain at least one <card> tag, and can additionally contain one <head> tag.

<head> tag - The <head> tag is an optional WML tag. This tag contains information that relates to the deck as a whole, including meta-data and access control elements. The <head> tag can optionally contain at least one <meta> element or one <access> element. The <head> tag is used to customize a Title Line display. Since the 9610 does not have a Title Line, its Web browser does not support this tag.

<meta> tag - The optional <meta> tag is contained between multiple <head> tags. This tag gives values for the parameters that describe the content of the deck. The 9610 IP Telephone does not support the <meta> tag. All 96xx telephones except the 9610 will render the value of the content attribute of a <meta> element as the Title on the Title Line if and only if the <meta> element also contains a name attribute with a value of "title".

| Attribute | Value(s) | Description | Comments |
|----------------|----------------|--|---|
| content | <i>cdata</i> | Should specify the name attribute description. | Supported by all 9600 Series IP Telephones except the 9610. |
| name | <i>keyword</i> | Name portion of the name content value. | Supported by all 9600 Series IP Telephones except the 9610. |

The page author can customize the title “Browser” using the <meta> tag with the *name* attribute equal to “title”. For example:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.WAPforum.org/DTD/wml_1.1.xml">
<wml>
<head>
<meta name="title" content="Da Browser" />
</head>
<card>
...
</card>
</wml>
```

<card> tag - A single WML file can contain multiple cards, supporting the analogy of a “deck” of “cards” within a single WML file. A “card” is essentially the specification of one specific WML page. This is a mandatory tag.

The card element attributes supported by the Web browser are as follows.

| Attribute | Value(s) | Description | Comments |
|------------|----------|---|--|
| newcontext | true | Re-initializes the browser context. Default is “false.” | Clears out the current WML browser context, which entails emptying the navigation stack history and clearing out all variables. When newcontext is set to true the Web browser clears its history buffer and captures and buffers the WML card title attribute of the WML card with the newcontext tag. This buffered attribute is used as the title for every WML element with a null or missing title attribute, including top-level cards. When set to true, the Web browser home page will not be loaded. Note: Page authors must include this attribute where it is necessary to clear the history stack. |
| | false | | |
| title | cdata | The title of the card. | Can be used for title displays. |

| Attribute | Value(s) | Description | Comments |
|------------------------|-----------------|---|----------|
| onenterbackward | <i>url</i> | Occurs when the user navigates into a card by means of a “prev” task. | |
| onenterforward | <i>url</i> | Occurs when the user navigates into a card by means of a “go” task. | |
| ontimer | <i>url</i> | Occurs when a “timer” expires. | |
| style | <i>property</i> | Cascading Style Sheet attribute. | |

Note:

If a onenterforward or onenterbackward attribute is defined for a <card> tag and the <card> tag also has an <onevent> tag defined with a onenterforward or onenterbackward event type, the attribute defined in the card tag supersedes the <onevent> binding.

<template> tag - The <template> tag defines a template for all the cards in a deck. The “code” in the <template> tag is added to each card in the deck. Only one <template> tag for each deck can be specified. This tag can only contain <do> and <onevent> tags.

The **template** tag attributes supported by the Web browser are:

| Attribute | Value(s) | Description | Comments |
|------------------------|-----------------|---|----------|
| onenterbackward | <i>url</i> | Occurs when the user navigates into a card by means of a “prev” task. | |
| onenterforward | <i>url</i> | Occurs when the user navigates into a card by means of a “go” task. | |
| ontimer | <i>url</i> | Occurs when the “timer” expires. | |
| style | <i>property</i> | Cascading Style Sheet attribute. | |

Note:

The implication for rendering WML pages is that the local environment always overrides a global template for <do> types with the same name.

Text Elements

See [Enabling Text Entry](#) and [Text Editing Modes](#) for guidelines to enable text entry and facilitate text editing. See [Access Key Input Mode \(AIM\)](#) for information about an alternate text entry method using access keys.

**
 tag** - The
 tag tells the browser to add a line break to the text at the point the element is written.

<p> tag - The <p> tag specifies a paragraph of text with alignment and line wrapping properties. All text data must be contained inside this tag. Only <do> tags, wml and card elements can exist outside the <p> tag. When rendered, this tag causes a line to be skipped.

| Attribute | Value | Description | Comments |
|--------------|-------------------------|--|----------|
| align | left right Center | Aligns the paragraph. Default is "left." | |
| mode | wrap nowrap | Sets whether a paragraph wraps lines or not. | |
| style | <i>property</i> | Cascading Style Sheet Attribute. | |

When the mode has been set to "nowrap" then all wml tags embedded inside this <p> tag get the "nowrap" behavior. For example, if an image is embedded inside, the image is restricted to the line. The browser truncates a line on the horizontal boundary. Similarly, for any anchors and <a> tags embedded inside, the label is truncated to the line boundary and the link is restricted.

When the mode has been set to "wrap" then all elements inside the <p> tag receive the wrap behavior. Images inside will bleed vertically into the adjoining lines. <a> and <anchor> tags will also continue to the adjoining line if the labels are long.

For each tag except the "mode" attribute will be set to "nowrap" if the "align" attribute is either "center" or "right". For tags the mode will not change.

Even though the "mode" attribute value might be changed, any embedded tag will inherit the original "mode" attribute value.

The alignment attribute is honored for all tags except for input tags.

Anchor Elements

<a> tag - <a> elements define <go> tasks that require a URL link specification. All <a> tags are rendered as underlined. All <a> nested tags like **br** and **img** are supported. A maximum of six anchors can be rendered on the screen at one time for all telephones except the 9610, which has a maximum of 5 anchors. The user selects the link by pressing one of the Line buttons associated with that display line.

| Attribute | Value | Description | Comments |
|------------------|-------------------------|---|----------|
| href | <i>url</i> | REQUIRED. Defines where to go when the user selects the link. | |
| title | <i>cdata</i> | Defines a text identifying the link. For telephones with a touch screen, the default value is "Touch to select". For other telephones, the default value is "Use button to select". | |
| accesskey | 1,2,3,4,5,6,7,8,9,0,*,# | A keypad key the user can press as a shortcut to selecting a link. | |
| style | <i>property</i> | Cascading Style Sheet Attribute | |

The Web browser supports the WTA Click to Dial application for any link on the screen. The browser also supports the WTAI Add to Phonebook function (wtai://wp/ap;) to add names and numbers to Contacts. The WTAI syntax is supported as an href attribute.

<anchor> tag - <anchor> elements define <go> tasks that require a URL link specification. All anchors are rendered as underlined. All <anchor> nested tags (br, go, img, prev, and refresh) are supported. A maximum of six anchors can be rendered on the screen at one time. The user selects a link by pressing the Line button associated with that display line. You cannot specify more than one <onevent> tag inside an <anchor> tag.

| Attribute | Value | Description | Comments |
|------------------|-------------------------|---|----------|
| title | <i>cdata</i> | Defines a text identifying the link. For telephones with a touch screen, the default value is "Touch to select". For other telephones, the default value is "Use button to select". | |
| accesskey | 1,2,3,4,5,6,7,8,9,0,*,# | A keypad key the user can press as a shortcut to select a link. | |
| style | <i>property</i> | Cascading Style Sheet Attribute | |

Image Elements

** tag** - Use the tag to place an image in the text flow. Use either monochrome wbmp (wireless bitmap) format or color JPEG format to code images for display. Images can be used as hyperlinks, as per WML 1.3. The element can be contained in these elements: a, anchor, and p.

| Attribute | Value | Description | Comments |
|---------------|--|--|---|
| align | top middle bottom left right center | Aligns the image. | Left is the only value currently supported. |
| alt | <i>cdata</i> | REQUIRED. Sets an alternate text to be displayed if the image is not displayed. If this is not supplied, either default text displays (if available) or one of the following messages displays: "Image not displayed" (for the Prompt Line) or "No image" for a softkey label. | When an alt tag is not associated with an image, the Prompt Line should be empty. |
| hspace | <i>px</i> <i>%</i> | Sets white space to the left and right of the image. Specify the value in pixels, for example, "10" or as a percentage of the available screen size. | The default is 0 pixel. |
| src | <i>url</i> | REQUIRED. The path to the image. Must be either a .wbmp file or a jpeg file. | |
| vspace | <i>px</i> <i>%</i> | Sets white space above and below the image. Specify the value in pixels, for example, "10" or as a percentage of the available screen size | The default is 0 pixel. |
| style | <i>property</i> | Cascading style sheet attribute. | |

An element may contain one or more <map> elements. 9600 Series IP Telephones that do not have a touchscreen ignore <map> elements contained within an element. 9600 Series IP Telephones with a touchscreen (for example, the 9670) ignore all but the first <map> element within an element as well as all <map> elements within an element used as softkey labels. When an element that contains a <map> element is activated on a telephone with a touchscreen, the variable Xmap will be set to the x-coordinate,

and the variable Ymap will be set to the y-coordinate, in pixels relative to the upper-left corner of the image (x=0, y=0), of the point at which the element was activated. An HTTP GET is then sent for the value of the url attribute in the first <area> element (see below) that contains the coordinates of the pixel by which the element was activated. The <area> elements will be checked to see whether they contain the pixel in the order in which they are listed in the <map> element, however, if more than 20 <area> elements are listed, any beyond the first 20 will be ignored. If the element is contained within an <a> or an <anchor> element, and if the point at which the element was activated (including points on any Application Line on which the image is rendered that are outside the area of the image) is not contained within one of the <area> elements, the <a> element or the <anchor> element will be activated.

Image Map Support Elements (H.323 9600 Series IP Telephones only)

<map> tag – This element is not visually rendered and is basically used as a “container” for one or more <area> elements. The <map> element is contained by the element.

<area> tag – This element is not visually rendered and is contained by the <map> element. Note that the <area> element is not specified in standard WML, but is specified in HTML.

| Attribute | Value | Description | Comments |
|---------------|---|--|----------|
| coords | A sequence of non-negative integers separated by commas | Specifies x,y coordinates of pixels relative to the upper left corner of the image, which is point “0.0”, except for the radius of a circle (see the shape attribute below), for which this value specifies the radius in pixels relative to the center of the circle. The sequence of integers is referred to as “i1, i2,i3,i4....” in the <area> attributes described below. | |
| href | <i>uri</i> | Specifies the URI to be processed when the area is activated. | |

| Attribute | Value | Description | Comments |
|--------------|---------|--|---|
| shape | | Specifies the shape of the area. | Mandatory. If this attribute is omitted or if the value is invalid, the <area> element is ignored. |
| | rect | Specifies the area is a rectangle with an upper left corner at i1,i2 and a lower right corner at i3,i4. | Any additional coordinates are ignored. If the value of the cords attribute contains fewer than four integers, the <area> element is ignored. |
| | poly | Specifies the area is a polygon with a vertex at i1,i2, another vertex at i3,i4, another vertex at i5,i6, etc. | If an odd number of coordinates are specified, the last one is ignored. If the last vertex is not the same as the first vertex a line connecting them will be assumed. If fewer than 3 distinct vertices or more than 50 total vertices are specified, the <area> element is ignored. |
| | circle | Specifies the area is a circle with a center at i1,i2 and a radius of i3. | Any additional coordinates are ignored. If the value of the cords attribute contains fewer than three integers, the <area> element is ignored. |
| | default | Specifies the entire image. | The coords attribute is ignored. |

Event Elements

<do> tag - The <do> tag is a card-level user interface. It serves as a general mechanism to activate a task, usually performed by the user clicking a word or phrase in the display. A task is performed in response to an event. There are four tasks in WML: **go**, **noop**, **prev**, and **refresh**.

The mandatory **type** attribute provides information about the intent of the element, helping to improve processing. If the Web browser does not recognize the specified type, the specified type is treated as unknown. For example, testing, experimental, and vendor specific types would be unknown.

| Attribute | Value | Description | Comments |
|-----------------|---|--|--|
| type | accept prev help reset options delete exit unknown | REQUIRED. Defines the type of the “do” element. | The exit attribute applies only to the 9610 IP Telephone. |
| label | <i>cdata</i> | Creates a label for the “do” element. | Optional. Creates a string label for the element. The telephone browser imposes a six character limit. |
| name | <i>mmtoken</i> | Defines a name for the “do” element. | |
| optional | true false | If set to true, the browser ignores this element. If set to false, the browser does not ignore this element. Default is “false.” | Optional. |

| Type | Description | Default Label | Comments |
|---------|-----------------------------------|---------------|--------------------------|
| exit | Exit the page | Exit | Supported for 9610 only. |
| accept | Acknowledgement of acceptance. | Accept | |
| delete | Delete item. | Delete | |
| help | Request for help. | Help | |
| options | Options or additional operations. | Options | |
| prev | Backward navigation. | Back | |
| reset | Clearing or reset. | Refresh | |
| unknown | | Unknown | |

<do> tags are rendered as centered softkey labels on the bottom display line. <do> tags are specified per WML page and therefore are page context-sensitive. <do> tags with an unspecified name attribute default to the type attribute value. A <do> tag with a <noop> tag embedded within renders the <noop> on a softkey, but pressing that softkey has no effect. The eight “do” types are labeled either specifically in a WML page or by a browser-dependent label.

If the total number of softkeys to be displayed is greater than the maximum number of softkeys that can be displayed simultaneously by the telephone, one less than the maximum number of softkeys will be rendered on the leftmost softkeys and a “More” softkey will be rendered on the rightmost softkey. When the More softkey is pressed, the remaining softkeys will be rendered in groups of one less than the maximum number of softkeys that can be displayed simultaneously (or fewer, if less than that number remain to be displayed), with the More softkey remaining on the rightmost softkey. After the last group of softkeys has been displayed, if the More softkey is pressed again, the first group of softkeys will be redisplayed.

If no labels are given, then the “do” types have the following default labels:

| Type | Default Label if no label specified |
|---------|--|
| exit | Exit (applicable only to the 9610). *See note below this table. |
| accept | Accept |
| delete | Delete |
| help | Help |
| options | Options |
| prev | Back |
| reset | Refresh |

Note:

Exit functionality is provided only for the 9610 to exit the Browser and render the Main Menu. Exit functionality can also be provided using the <do> tag type “exit” so that an Exit softkey can be rendered on any desired Web page. The Exit softkey is presented by default as the first softkey on the topmost page. Therefore, if an author codes an exit tag on a topmost page it will be ignored. The syntax is: <do type=“exit” label=SK1>.

The only behavior for the Exit syntax is to exit the browser and render the main menu.

If no <do> tags were specified, only the default softkeys display:

| | | | |
|------|---------|--|--|
| Home | Refresh | | |
|------|---------|--|--|

If one <do> tag was specified, these softkeys display:

| | | | |
|--------|------|---------|--|
| 1st DO | Home | Refresh | |
|--------|------|---------|--|

If multiple <do> tags are specified, display them as follows:

Page 1 softkeys:

| | | | |
|--------|--------|--------|------|
| 1st DO | 2nd DO | 3rd DO | MORE |
|--------|--------|--------|------|

Page 2 softkeys:

| | | | |
|--------|--------|------|------|
| 4th DO | 5th DO | Etc. | MORE |
|--------|--------|------|------|

Page *n* softkeys:

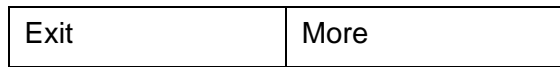
| | | | |
|----------------|------|---------|------|
| <i>n</i> th DO | Home | Refresh | MORE |
|----------------|------|---------|------|

Note:

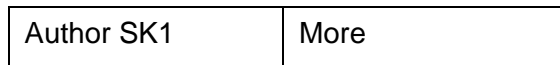
As implied in the examples, the Softkey buttons are labeled in sequential order of the <do> tags. When an image tag exists inside a <do> tag, and the attempt to retrieve the image through HTTP fails, the <do> tag label displays as the softkey label.

For the 9610 IP Telephone, if more than two <do> tags are specified for a WML page, they will be displayed as follows:

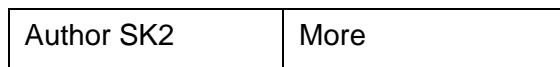
Page 1 softkeys for a 9610:



Page 2 softkeys for a 9610:



Page 3 softkeys for a 9610:

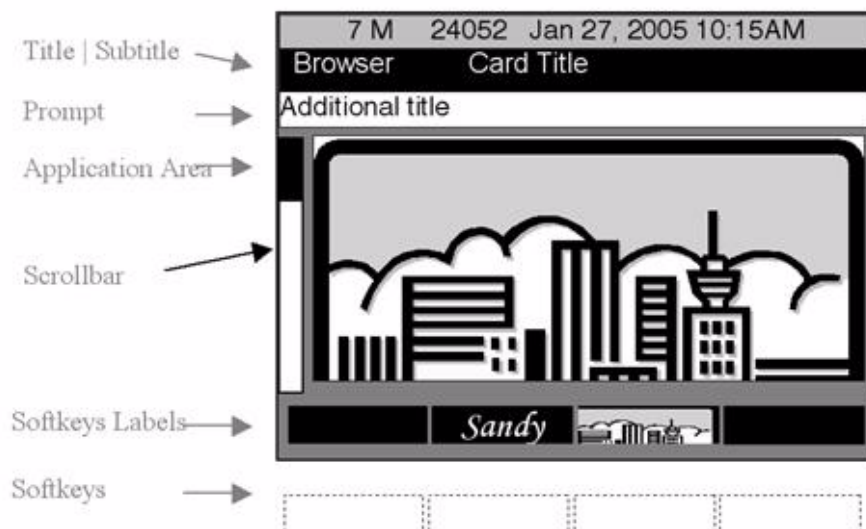


To render JPEG or WBMP images in the softkey label area, embed tags in the <do> tag. For example:

```
<do type="example-accept" label="SK1">
  <go href="example-#card2"/>
  
</do>
```

To render JPEG or WBMP images in the softkey area, use the CSS2 property background-image with a <do> tag. The example that follows shows a <do> tag with embedded images. Notice that the <do> tag has an image that does not completely fit in Softkey 3. The default background color shows where the image does not cover the softkey label area.

The default value of “No image” is used for the Alt text when the image cannot be loaded on a softkey.



<onevent> tag - The onevent tag serves as a container for code that you want executed automatically when one of the four intrinsic events occurs. The onevent element binds (associates) the tasks (code) to the event for the element. You must specify the intrinsic event using the mandatory **type** attribute. For example, when a user presses the Back softkey, instead of being routed to the previous screen, the user is directed to another specified page because this tag carries out an onevent backward event.

The intrinsic events are:

| Event | Permitted Tags | Description |
|-----------------|-------------------------|--|
| onenterbackward | card or template | Occurs when a <prev> navigates back onto a card. |
| onenterforward | card or template | Occurs when a <go> navigates into a card. |
| onpick | option | Occurs when a user selects/deselects an item. |
| <u>ontimer</u> | <u>card or template</u> | <u>Occurs when the time expires.</u> |

The template element creates code that is inserted into all cards in a single deck. The nested tags are: go, noop, prev, and refresh. There are no visual implications for supporting the <onevent> tag. If there is more than one <onevent> tag defined with the same type in a deck/card, then only the last <onevent> tag will be rendered. Specifying more than one <onevent> tag inside an <anchor> tag is an error.

| Attribute | Value | Description | Comments |
|-------------|--|--|----------|
| type | onenterbackward onenterforward onpick ontimer | REQUIRED. Specifies the type of the "onevent" element. onenterbackward - Triggered when a <prev> goes to a previous card. onenterforward - Triggered when a <go> goes to a card. onpick - Triggered when an item is selected/unselected. ontimer - Triggered when a timer expires. | |

<postfield> tag - Use the postfield tag to set a name/value pair that can be transmitted during a URL request to an origin server, the request's source. The **name** attribute sets the name, which must be a valid WML variable name. The **value** attribute sets the value. There are no visual rendering implications with this tag.

| Attribute | Value | Description | Comments |
|--------------|--------------|-----------------------------------|----------|
| name | <i>cdata</i> | REQUIRED. The name of the field. | |
| value | <i>cdata</i> | REQUIRED. The value of the field. | |

Task Elements

<go> tag - The go element can contain one or more postfield elements. If a go element destination is a card within the same deck, all postfield elements are ignored. The go element can also contain one or more setvar elements. Unlike postfield elements, there are no destination limitations on passing information contained in the setvar elements. The <go> nested tags, postfield and setvar, are supported.

| Attribute | Value | Description | Comments |
|-----------------------|---------------|--|----------|
| href | <i>url</i> | REQUIRED. | |
| accept-charset | Charset_list | A comma- or space-separated list of character encoding the server must be able to process. The default value is "unknown." | |
| sendreferer | true false | If set to true, the browser sends the URL of the current deck with the request, which allow servers to perform simple access control on decks, based on which decks are linking to them. Default is "false." | |

| Attribute | Value | Description | Comments |
|---------------|-----------------|---|----------|
| method | post get | Sets how to send the data to the server. Default method is get. When method=get, the data is sent as a request with <i>?data</i> appended to the URL. A get can be used only for a limited amount of data, which is a disadvantage. If you send sensitive information it is displayed on the screen and saved in the Web server's logs. With method="post", the data is sent as a request with the data sent in the body of the request. This method has no limit, and sensitive information is not visible. The data sent in a get method is limited to ASCII characters. The data sent in a post method can include non-ASCII characters that are included as part of a value attribute in an <input>, <select>, or <option> tag that is part of the current WML page. | |

<noop> tag - The noop tag dictates that no operation should be done. This tag can be used on the card level to prevent an event that is specified on the deck level by the template element from occurring. This tag can only be contained in either a do or onevent element.

An example of noop is to use a <do> tag to add a "Back" link to the card. When users click the "Back" link, generally they should be taken back to the previous card. However, the <noop> tag prevents this operation. When the user clicks on the "Back" link nothing happens.

<prev> tag - The prev tag specifies navigation to the previous URL in the history.

<refresh> tag - The refresh tag specifies a refresh task whereby whatever card is being displayed is refreshed. This task specifies the need for an update of the user agent context as specified by the contained <setvar> elements. This tag can only be nested inside an anchor, do, or onevent element. User-visible side effects of the update can occur during the <refresh> processing.

Input Elements

<input> tag - The input tag specifies a point where the user is prompted to enter text. This tag is supported by all 9600 Series IP Telephones except the 9610 IP Telephone, which supports only Access Key Input Mode (AIM) but does not support text entry.

| Attribute | Value | Description | Comments |
|----------------|----------------|---|----------|
| name | <i>nmtoken</i> | REQUIRED. The name of the variable that is set with the result of the user's input. | |
| emptyok | true false | Sets whether the user can leave the input field blank or not. Default is "true." | |
| format | | Sets the data format for the input field. Default is "M." | |
| | A | A = uppercase alphabetic or symbol characters (not lowercase alphabetic or numeric characters) | |
| | a | a = lowercase alphabetic or symbol characters (not uppercase alphabetic or numeric characters) | |
| | N | N = numeric characters | |
| | X | X = uppercase alphabetic, numeric, or symbol characters (not lowercase alphabetic characters) | |
| | x | x = lowercase alphabetic, numeric, or symbol characters (not uppercase alphabetic characters) | |
| | M | M = any character, but is treated as uppercase for data entry | |
| | m | m = any character, but is treated as lowercase for data entry | |
| | *f | *f = Any number of characters. Replace the f with one of the letters above to specify what characters the user can enter. | |
| | nf | nf = Replace the n with a number from 1 to 9 to specify the number of characters the user can enter. Replace the f with one of the letters above to specify what characters the user can enter. The user cannot exit the input box unless the correct number or type of characters is entered. The user does not receive an error message if incorrect data is entered. | |

| Attribute | Value | Description | Comments |
|------------------|------------------|---|----------|
| ivalue | | Specifies text to be rendered in the input field before the <input> element is activated. The default value is "Enter text here". The attribute value takes precedence over ivalue . The ivalue attribute is not specified in standard WML. The value of ivalue is only used for display and it does not become the value of the variable specified by the name attribute. | |
| maxlength | <i>number</i> | Sets the maximum number of characters the user can enter in the field. If the number of characters entered exceeds this value, display "Maxlength reached" on the Prompt Line. For H.323 only, the default value is 200, which is used if a maxlength attribute is not specified or if a value greater than 200 is specified. | |
| title | <i>cdata</i> | Sets a title for the input field. For the 9670G the default value is "Enter text". For all other 96xx telephones the default value is "Enter text between brackets". | |
| type | text password | Indicates the type of the input field. The default value is "text." | |
| value | <i>cdata</i> | Sets the default value of the variable in the name attribute. | |
| style | <i>property</i> | Cascading style sheet attribute. | |

| Attribute | Value | Description | Comments |
|--------------------|-----------------------------|---|----------|
| inputformat | <i>alpha</i> | Specifies the initial text entry mode for the input. The default value is "alpha". If inputformat is not specified, the initial text entry mode is set to "abc". <i>Num</i> sets the initial mode to "123". <i>alpha</i> sets the initial mode to "abc". <i>ALPHA</i> sets the initial mode to "ABC". <i>Alpha</i> sets the initial mode to "Abc". On the 9670G, this element also specifies which keyboard layout is initially displayed when the <input> element is activated; <i>alpha</i> displays the Alpha keyboard layout. <i>ALPHA</i> displays the Alpha keyboard layout in UPPERCASE text entry mode. <i>Alpha</i> displays the Alpha keyboard layout with the Shift key activated. <i>Num</i> displays the Number/Symbol keyboard layout. | |
| autoselect | <i>true</i> <i>false</i> | Applies only to 9600 Series IP Telephones running H.323 software Release 2.02 and greater. Specifies whether or not the input element is to be automatically activated when it is rendered. The default value is "false". If this attribute is set to "true" for more than one <input> element in a card, only the first one will be activated. | |

The input tag causes an automatic line break before and after input text.

Only one input tag can exist per display line.

When a user views a page with the input tag specified, the first thing that shows up in the Top Line is the card title, if specified. When the user scrolls to the first line containing input, the Top Line shows the input box title if specified, otherwise the card title is shown. The Top Line displays the card title for all non-input text.

When the input box is selected, a vertical line (the "cursor") appears at the left side of the input box.

The attribute **type** password should only be used when it is important to not display the user's password on the screen. Asterisks are displayed instead. It is also important that the password not be cached.

The phrase **Enter text here** appears for all input tags if the **value** and **ivalue** attributes are null. If non-null content is specified in the **value** attribute, that

content displays for that input tag. If the **value** is null but **ivalue** is non-null, the value of **ivalue** will be displayed.

Only the correct size, type, and number of characters are accepted in to the input box. For example, if alpha text is specified and the user types in a symbol or numeric text, the user input is not accepted. The screen repaints and the user has to re-enter the text. If the wrong kind of text is typed, the user receives an error tone. If the "n" (number) value is specified and the user types in the incorrect number of characters, that input is rejected.

When an <input> element is rendered:

- An input field will be displayed without any text being displayed to the left of the left square bracket,
 - a line break will be rendered before and after the input field,
- if the variable specified by the name attribute is already bound to a value (including a default value specified by a **value** attribute), the bound value will be displayed in the input field, otherwise, the value of the **ivalue** attribute will be displayed in the input field.

If an <input> element is activated while text entry is disabled:

- On the 9670G, an On-Screen Keyboard will be displayed and the card will be re-rendered in the available Application Line Area space above the keyboard; the input field will be on the first Application Line if the field is on the first line of the card, or on the last visible Application Line if the field is on the last line of the card, or with the input field centered vertically in the visible portion of the Application Line Area if the field is on any other line of the card. If more than one <input> element is contained on the card, the Next Field touch object on the On-Screen Keyboard will be enabled, otherwise the Next Field touch object on the On-Screen Keyboard will be disabled.
- The text specified by the **ivalue** attribute, if present, will be removed from the input field.
- The value of the **value** attribute, if any, will be displayed in the text input field.
- Text entry will be enabled except that a **Done** softkey will be substituted for the **Save** softkey or, for the 9670, a **Done** touch object will be substituted for the **Save** touch object.

While text entry is enabled:

- If the user tries to enter a character that is not consistent with the value of the format attribute, an error beep tone will be generated and "Invalid character. Try again" will be displayed on the Prompt Line.
- Any attempt to scroll away from the input field or to highlight another element (without activating it) will be ignored without generating an error beep tone.

- If the value of the emptyok attribute is false and no text has been entered, if the Done or Cancel softkey or touch object is pressed, or if an attempt is made to re-activate the <input> element or to activate any other element (either of which would normally disable text entry for the current <input> element), an error beep tone will be generated and "Input field cannot be empty" will be displayed on the Prompt Line.
- If the entered text complies with the value of the emptyok attribute:
 - if the Cancel softkey or touch object is pressed, text entry will be disabled, the <input> element will remain highlighted, and the value bound to the variable specified by the name attribute will not be changed;
 - if the Done softkey or touch object is pressed, text entry will be disabled, the entered text will be bound to the variable specified by the name attribute, and the <input> element will remain highlighted;
 - if a different element is activated, text entry will be disabled, the entered text will be bound to the variable specified by the name attribute, and the other element will be activated;
 - if the <input> element is activated again, text entry will be disabled, the entered text will be bound to the variable specified by the name attribute, and the <input> element will remain highlighted with text entry disabled.

When text entry is disabled, if no value has been bound to the variable specified by the name attribute, the value of the ivalue attribute will be re-displayed in the input field. Otherwise, the value bound to the variable specified by the name attribute will be displayed in the input field. On telephones with a touch screen, the On-Screen Keyboard will be removed from the display and additional card content, if available, will be rendered on the Application Lines previously occupied by the keyboard.

When text entry is terminated:

- If a value has been bound to the variable specified by the name attribute, and if the <input> element contains an onsubmit attribute:
 - if the value of the onsubmit attribute does not begin with a “#” character, the value will be processed as a URI, but if the value is not a valid URI, the onsubmit attribute will be ignored;
 - if the value of the onsubmit attribute begins with a “#” character, the value (except for the “#”) will be compared to the values of the id attributes of any <do> elements contained in the same <card> element as the <input> element. If it matches, the task element(s) contained in the <do> element will be executed;

- if the value of the onsubmit attribute begins with a “#” character but does not match the value of an id attribute of a <do> element as specified above, the value will be processed as a URI fragment name, but if the fragment name does not match the value of an id attribute of a <card> element in the deck, the onsubmit attribute will be ignored.
- If a value has been bound to the variable specified by the name attribute, the value will be displayed in the input field.
- If a value has not been bound to the variable specified by the name attribute, the value of the ivalue attribute will be re-displayed in the input field;
- On the 9670, the On-Screen Keyboard will be removed from the display and additional card content, if available, will be rendered on the Application Lines previously occupied by the keyboard.

See [Text Elements](#) for other text entry guidelines.

<optgroup> tag - Sets of <optgroup> brackets can be put around <options> in a <select> list. The results break a list into sub lists.

| Attribute | Value | Description | Comments |
|--------------|-----------------|--|----------|
| Title | <i>cdata</i> | Sets a title for the optgroup element. | |
| style | <i>property</i> | Cascading style sheet attribute. | |

<option> tag - A set of option tags is needed to specify each individual item in a list. This tag must be used with the select tag.

| Attribute | Value | Description | Comments |
|---------------|-----------------|---|----------|
| onpick | <i>url</i> | Sets what is going to happen when a user selects an item. | |
| title | <i>cdata</i> | Sets a title for the option | |
| value | <i>cdata</i> | Sets the value to use when setting the “name” variable in the select element. | |
| style | <i>property</i> | Cascading style sheet attribute. | |

The following can occur:

If an onpick attribute is specified, the user simply presses the associated Line button to go to that specified URL.

If no onpick is specified, the user must choose and use the select (Do tag) softkey. Pressing the Line button checks the option if a radio button is specified, or checks/unchecks a specified check box. If there is a radio box and multiple="false" value in the <select> tag), clicking on the Line button keeps the radio button checked.

A WML page will not specify both a do type (select softkey) and onpick on the same page. Either the do type or onpick specifies the URL of the next card.

Line buttons toggle the state. When the Line button is initially pressed, a choice is selected. Pressing the same Line button again deselects the choice. However, this is not always true. If the option corresponds to a check box (multiple="true" value in the <select> tag) it is true.

If an onpick attribute is defined for an <option> tag and the <option> tag also has an <onevent> tag defined, the onpick attribute will supersede the onevent binding.

<select> tag - The select tag allows for the definition of a list, embedded in a card. This tag allows the user to choose inputs from a list rather than having to type a value. The select tag must be used with the option tag.

| Attribute | Value | Description | Comments |
|-----------------|-----------------|--|----------|
| name | <i>nmtoken</i> | REQUIRED. String that names the variable to which the selection results are assigned. | |
| ivalue | <i>Cdata</i> | Sets the pre-selected option element. If none is specified, the first item in a list is automatically selected. | |
| multiple | true false | Sets whether multiple items can be selected. Default is "false." False is used for a single selection. | |
| title | <i>Cdata</i> | Sets a title for the list. | |
| value | <i>Cdata</i> | Sets the default value of the variable in the name attribute. | |
| style | <i>property</i> | Cascading style sheet attribute. | |
| iname | <i>Index</i> | This optional attribute specifies the name of the variable that will be assigned the value of the index result. The index result is the position of the selected item in the select list. If multiple selections are permitted, the index result is a semicolon-delimited list of the indices. The indexing starts at one. A zero signifies that no selection has been made. | |

Option tags are nested within select tags to determine the number of multiple-choice selections.

The following defines the graphic template for rendering single and multiple-choice selections:

- Single selection - a modified radio box is rendered as follows for single selection of multiple choices:
 - a complete empty circle indicates that the user did not select this item.
 - a complete empty circle with a black round dot in the center of the circle indicates the user choice.
- Multiple selection - a check box is rendered as follows for multiple selections of multiple choices:
 - an empty check box indicates that the user did not select this item.
 - a check box with an X centered in the box indicates the user choice.

Variable Elements

<setvar> tag - There are no visual rendering implications with this tag.

| Attribute | Value | Description | Comments |
|--------------|--------------|---|----------|
| name | <i>cdata</i> | REQUIRED. Sets the name of the variable. | |
| value | <i>cdata</i> | REQUIRED. Sets the value of the variable. | |

If the value of a name attribute is equal to one of the following variable names the <setvar> element is ignored: IPADD, MACADDR, MODEL, or PHONEXT.

<timer> tag - The timer tag sets a timer that starts counting. This tag must be used with <onevent type="ontimer"> to be useful. The Web browser sets the timer value to 10 or to the value sent in the timer tag, whichever is greater. A minimum setting of 10 equates to a minimum timer setting of 1 second.

| Attribute | Value | Description | Comments |
|--------------|----------------|--|----------|
| value | <i>cdata</i> | REQUIRED. Sets the default value of the variable defined in the name attribute. | |
| name | <i>nmtoken</i> | Names the variable that is set with the value of the timer. | |

In addition to the standard WML requirements for initializing, starting and stopping a timer, the timer will be stopped any time a different application or interrupt screen is displayed on the telephone, and the timer will be reinitialized and restarted every time the WML Browser application is redisplayed on the telephone, even if the same card is displayed that has been displayed previously.

Character Entities

As with any syntactic language, WML has certain characters that have special meaning. The two most obvious of these characters are the **<** and **>** symbols, which surround all tags. These characters cannot be typed in directly if the designer's intent is to display these characters. Thus, all characters that can be displayed in a Web browser have numeric values assigned to them. The numeric values are entered into the source Web page as **&#nnn;** where **nnn** is a three-digit value. For example, the **<** symbol is entered as **<**.

In addition, many of these characters also have names assigned. Name values are entered into the source Web page as **&name;** where **name** is the WML name associated with this character. For example, the **<** symbol would be entered as **<**. The browser fully supports the set of characters defined by the World Wide Web Consortium in conformance with the standard.

For convenience, here are a few of these key symbols:

| Description | Symbol | Numeric Entity | Name Entity |
|------------------|--------|----------------|-------------|
| double quotation | “ | " | " |
| Ampersand | & | & | & |
| Apostrophe | ‘ | ' | ' |
| less than | < | < | < |

Colors and Fonts

With the exception of the 9620C, 9640/9640G, 9650C, and 9670G, the browser supports a four grayscale display; the 9620C, 9640/9640G, 9650C, and 9670G IP Telephones support color. Only a normal font weight is supported. Bold, italic and different font sizes are not supported. The font the telephone uses defines characters to have at most six pixels in width.

Access Key Input Mode (AIM)

The Web browser considers cards that include the accesskey attribute and which require the user to enter text to be in a new Text Entry mode. That new mode is called Access Key Input Mode (AIM). Unlike Text Entry mode, AIM arbitrates dialpad use between the phone and Web applications that use accesskey attributes. When a Web page with one or more valid accesskey attributes loads completely, the Web page takes control of the dialpad.

With Text Entry, text entry is considered complete once a URL is launched and the Web relinquishes control of the dialpad. AIM works differently than standard text editing by maintaining control of the dialpad over the course of loading one or multiple new Web pages. AIM stays in effect until either:

- the user selects another application tab,
- the phone goes off-hook to make or receive a call, or
- a Web page without any valid accesskey attribute is completely loaded.

Support for Access Keys

The accesskey attribute assigns a particular key on the phone to an element. Its purpose is to allow the user to activate a particular element using a single key. For the Web browser, an access key is a single Dialpad button and the element is a URL. The access keys associated with the IP telephone Web browser are as follows:

| Accesskey |
|-----------|
| 1 |
| 2ABC |
| 3DEF |
| 4GHI |
| 5JKL |
| 6MNO |
| 7PQRS |
| 8TUV |
| 9WXYZ |
| 0 |
| * |
| # |

In [Table 16](#) each of the Dialpad button access keys is mapped to a different URL.

Table 16: Dialpad/URL Mapping Example

| Dialpad Button | Example |
|----------------|--|
| 1 | <code>1</code> |
| 2ABC | <code>2</code> |
| 3DEF | <code>3</code> |
| 4GHI | <code>4</code> |
| 5JKL | <code>5</code> |
| 6MNO | <code>6</code> |
| 7PQRS | <code>7</code> |
| 8TUV | <code>8</code> |
| 9WXYZ | <code>9</code> |
| 0 | <code>0</code> |
| * | <code>*</code> |
| # | <code>#</code> |

Button presses do not distinguish among the characters a button represents. For example, if a user presses Dialpad button 2, it is impossible to distinguish anything other than the user pressed the second button. No refinement is made to inform the Web server that the user meant **2** or **A** or **B** or **C**.

An AIM application is one where the page currently being loaded contains `<a>` or `<anchor>` tags with access keys defined. The application developer has a choice of mapping every keypad entry to a URL or can opt to leave some unmapped keypad entries. The `<a>` or `<anchor>` tags support the `accesskey` attribute.

Example of Text Entry Using AIM:

The user wants to look up a person named "Oscar" in a Directory database. The user presses a Directory softkey on the Home page that initiates the AIM application. [Figure 27](#) shows a sample starting page. The user presses the **6MNO** Dialpad button because it contains the first letter of the name to be found. In most cases, AIM avoids the user having to enter the entire first and/or last name. AIM also avoids multiple key presses to select letters, for example, pressing **6** four times to select the letter **O**.

The user then presses the dialpad number key that corresponds to the second letter in the name once, etc. There is no need to press a number key multiple times to select a letter.

[Figure 27](#) through [Figure 30](#) illustrates locating the name “Oscar” in the directory.

Figure 27: Starting Page



Figure 28: User presses “6” one time for “O” and the closest match to “O” displays.

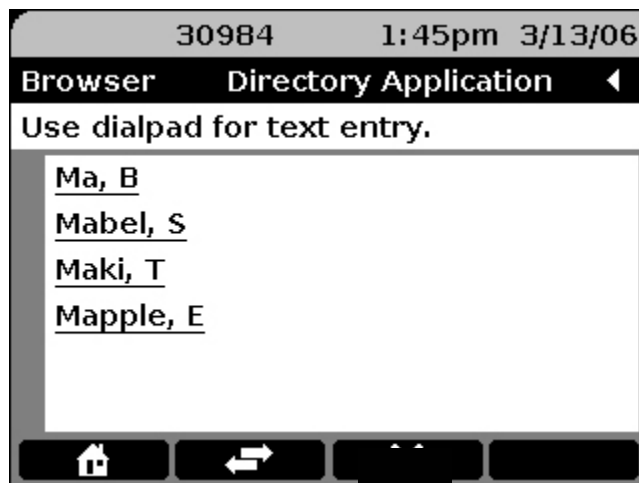


Figure 29: User presses “7” one time for “S,” narrowing the search.

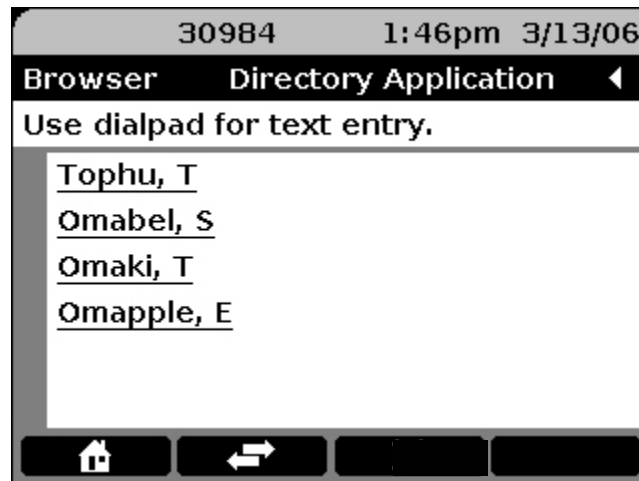
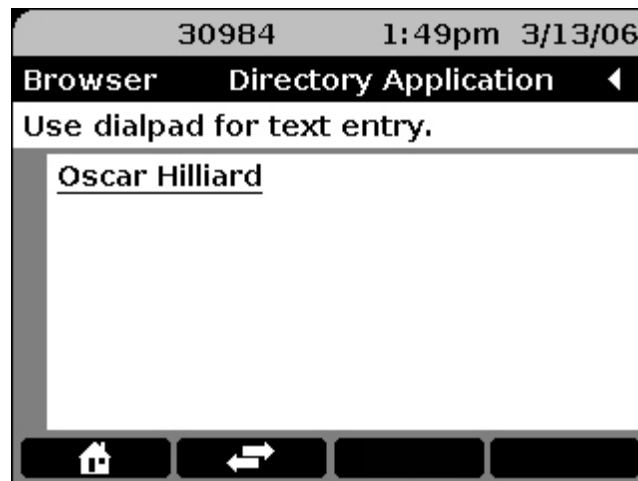


Figure 30: User presses “2” one time for “C” and the search result displays the desired result - Oscar Hilliard.



AIM Considerations

AIM maintains dialpad control while loading one, or multiple, new Web pages as long as the accesskey attribute is part of the page.

When a user presses a Dialpad button, or “access key,” the browser retrieves a URL without displaying any characters to the user. The button press information is sent to the URL and the Web page author returns subsequent screens. What the user views is up to the page author. Access key supplies the mechanism to capture button press information and send that information to a Web server.

The user is automatically in AIM when a page containing an accesskey attribute loads and can begin pressing Dialpad buttons to send button press information.

Having a **Clear** softkey on the AIM page allows a user to clear previous search results and start a new search. The page author must set up a **Clear** softkey that loads the first AIM page to allow this action.

The Web pages resulting from Dialpad button presses are part of the history stack. This is normal browser behavior.

AIM is re-enabled when the phone goes back on-hook and the current Web page has valid access keys.

An error beep sounds if the user presses any Dialpad buttons while a new page is loading.

If the accesskey attribute is on the page, then:

- Text Entry/Editing (TE) softkeys do not appear when the user is in AIM. AIM does not have the default Text Editing softkeys such as **alpha**, **backwards**, etc.
- The <a> and <anchor> links that would normally appear on the screen are hidden.
- Normal text entry (<input> tags) is prohibited because it cannot co-exist with access keys. If, by error, regular Text Entry and access keys are on the same page, AIM rules take precedence. This means when the accesskey attribute appears on a page, the regular text entry (TE) softkeys do not appear.
- If the user enters the AIM Application, and there is an input field for TE, the field will not get focus. Focus does not occur, regardless of how many times the user selects the Line button next to the TE input field. Since TE is prohibited when access keys are present on a page, TE does not get focus.
- The server must send back a resulting page with the access keys for each Dialpad button in the page. Doing so maintains AIM across pages and retains dialpad ownership. The URLs associated with each accesskey can differ from the starting page in case the proxy server caches Web pages.

For example, after the page with access keys loads and the user pushes the dialpad, the Web server launches the URL:

```
<a href="http://www.example.com/URL2.wml" accesskey="2">2<\a>
```

In the resulting page, the page author can send a different URL to be associated with the same access key:

```
<a href="http://www.example.com/URL2a.wml" accesskey="2">2<\a>
```

Terminating AIM

The user ends AIM by:

- going off-hook,
- selecting another application button,
- Selecting (loading) a Web page that does not contain any valid accesskey attributes.

When the accesskey URL is invalid or there is an error loading such a URL (404, 403, etc.) the Top Line displays the message “Page Cannot Be Rendered.” and the user can press the **Back** softkey to return to the previous page.

Pressing any of the navigation scroll buttons does not terminate AIM. This is because the user might want to scroll to see results, then continue a search. AIM is re-enabled when the user goes back on hook and the current Web page has valid access keys.

Developing Web Pages for the 9610 IP Telephone

The 9610 IP Telephone is a “walk-up” model designed for public areas like hotel lobbies. As such, the 9610 supports several different browser capabilities than other 9600 Series models. The features and general requirements described in [Chapter 6: About the Web Browser](#) apply to 9610 IP Telephones unless otherwise indicated within the text or as notes preceding/following applicable text.

9610 Display Characteristics

The browser renders on a 160x160 pixel display using 4-level grayscale color.

The 9610 uses the Swiss 721 Bold Condensed font due to its smaller display area.

The Top Line accommodates approximately 158 pixels or 22 digits/characters on the average. The remaining [Application] Lines accommodate approximately 150 pixels, or 20-21 digits/characters on average.

Support for Cascading Style Sheets

XHTML and **Cascading Style Sheets** (CSS) are designed to separate content from its presentation. XHTML and WML tags were originally designed to define the content of a document. In this way, the same content can be rendered on diverse devices. Most XHTML elements are semantic elements, that is, they convey meaning about their content rather than information on how to display it. For example, the element contains content that should be emphasized. It is up to the browser to figure out how to render the emphasis, with a different

typeface, a louder voice, or in another way. Style sheets are a way to manage a Web page's overall look such as the page background, background color, or font color.

A style is a rule that tells the browser how to render a particular tag's contents. Each tag has a number of style properties associated with it, whose values define how that tag is rendered by the browser. A rule defines a specific value for one or more tag properties. Style Sheets allow style information to be specified in many ways. The Web browser supports the inline style where a style attribute and tag along with a list of properties and their values are specified. The browser uses those style properties and values to render the tag's contents.

The browser supports CSS2. CSS2 is compatible with both WML and XHTML and can be re-used if the browser evolves to XHTML. For more information, see <http://www.w3.org/TR/CSS21/cascade.html>.

Cascading Order

If more than one style is specified for a WML element, the multiple style definitions cascade into one style definition. Many factors affect the precedence of styles, including:

- The default style of the browser on the phone, that is, how elements are presented in the absence of presentation rules from a style sheet.
- Inheritance rules under which style rules are inherited by elements.
- A style rule that is more specific takes precedence over a less specific, conflicting rule. For example, a style rule applied to a class of elements takes precedence over a rule that applies to an element in general. A rule that applies to an element ID takes precedence over both the class of elements style and that of an element in general.
- The most recent style rule has only the scope of the current tag and its embedded tags. Once the current tag closes, the tag rule with a higher hierarchy replaces the closed tag.
- If a style rule in a style sheet conflicts with a presentational attribute of a WML element, the style sheet takes precedence.

Note:

Color references are only for illustration purposes.

In lieu of a specific rule for a particular tag element, properties and their values for tags within tags are inherited from their parent tag. Thus, setting a property for the <wml> tag effectively applies that property to every tag in the body of the document, except for those that specifically override it. To make all the text in the page blue, code the following:

```
wml style="color:blue;"
```

rather than creating a rule for every tag used in the page.

This inheritance extends to any level. If the page author later creates a <p> tag with different color text, the style-conscious browser displays all the contents of <p> tag and all its included tags in that new color. When the <p> tag ends, the color reverts to that of the <wml> tag. For example, for WML:

- <card> tags inherit all <wml> tag properties.
- All <card> properties are inherited by <p> and <do> tags.
- All <p> tag properties are inherited by other wml tags.

This inheritance is not restricted to its immediate child tags, but can cascade further down.

Also note that some inherited properties may not have meaning in the scope of the child tag. For example, a background color defined for a <card> tag has no meaning for a <onevent> tag that has no visual rendering.

[Table 17](#) shows which WML tags are parent and which tags inherit properties.

Table 17: WML Inheritance Table

| Col1 | Col2 | Col3 | Col4 | Col5 | Col6 | Col7 | Col8 | Col9 | Col10 |
|-----------------|--|--|--|--|--|--|--|--|--|
| No inheritance. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. |
| wml | | | | | | | | | |
| | card | do | go | postfield | | | | | |
| | | | | setvar | | | | | |
| | | | noop | | | | | | |
| | | | prev | setvar | | | | | |
| | | | refresh | setvar | | | | | |
| | | onevent | go | postfield | | | | | |
| | | | | setvar | | | | | |
| | | | noop | | | | | | |
| | | | prev | setvar | | | | | |
| | | | refresh | setvar | | | | | |
| | | timer | | | | | | | |
| | | p | | a | br | | | | |
| | | | | | img | | | | |
| | | | | anchor | br | | | | |
| | | | | | go | postfield | | | |
| | | | | | | setvar | | | |
| | | | | | img | | | | |

| Col1 | Col2 | Col3 | Col4 | Col5 | Col6 | Col7 | Col8 | Col9 | Col10 |
|-----------------|--|--|--|--|--|--|--|--|--|
| No inheritance. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. |
| | | | | | prev | setvar | | | |
| | | | | | refresh | setvar | | | |
| | | | | b | | | | | |
| | | | | big | | | | | |
| | | | | br | | | | | |
| | | | | do | go | postfield | | | |
| | | | | | | setvar | | | |
| | | | | | noop | | | | |
| | | | | | prev | setvar | | | |
| | | | | | refresh | setvar | | | |
| | | | | em | | | | | |
| | | | | fieldset | | | | | |
| | | | | i | | | | | |
| | | | | input | | | | | |
| | | | | img | | | | | |
| | | | | select | | | | | |
| | | | | | optgroup | optgroup | | | |
| | | | | | | option | onevent | go | postfield |
| | | | | | | | | | setvar |
| | | | | | | | | noop | |
| | | | | | | | | prev | setvar |
| | | | | | | | | refresh | setvar |
| | | | | | option | onevent | go | postfield | |
| | | | | | | | | setvar | |
| | | | | | | | noop | | |
| | | | | | | | prev | setvar | |
| | | | | | | | refresh | setvar | |
| | | | | small | | | | | |
| | | | | strong | | | | | |
| | | | | table | tr | td | | | |
| | | | | u | | | | | |

| Col1 | Col2 | Col3 | Col4 | Col5 | Col6 | Col7 | Col8 | Col9 | Col10 |
|-----------------|--|--|--|--|--|--|--|--|--|
| No inheritance. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. | Tag in column inherits from tag in column 1. |
| | head | access | | | | | | | |
| | | Meta | | | | | | | |
| | template | do | go | postfield | | | | | |
| | | | | setvar | | | | | |
| | | | noop | | | | | | |
| | | | prev | setvar | | | | | |
| | | | refresh | setvar | | | | | |
| | | onevent | go | postfield | | | | | |
| | | | | setvar | | | | | |
| | | | noop | | | | | | |
| | | | prev | setvar | | | | | |
| | | | refresh | setvar | | | | | |

[Table 18](#) provides a list of CSS2-affected WML tags. Applicable tags are indicated by a **Yes**. The Color column applies to the foreground color, while the Background Color column is the alternative used to set the background color.

Table 18: WML Tags to Which CSS2 Applies

| WML Tag | CSS2 Property | |
|----------|---------------|------------------------------|
| | Color | Background Color |
| wml | Yes | Yes |
| card | Yes | Yes |
| template | Yes | Yes |
| br/ | Yes | Yes - *see note that follows |
| p | Yes | Yes |
| a | Yes | Yes |
| anchor | Yes | Yes |
| img | Yes | Yes |
| do | Yes | Yes |

| WML Tag | CSS2 Property | |
|-----------|---------------|------------------|
| | Color | Background Color |
| onevent | | |
| postfield | | |
| go | | |
| noop | | |
| prev | | |
| refresh | | |
| input | Yes | Yes |
| optgroup | Yes | Yes |
| option | Yes | Yes |
| select | | |
| setvar | | |
| timer | | |

Note:

The
 tag's color setting needs to support color property so the inverse color shows when a line with this tag is in focus.

CSS2 Specifications

The browser supports Cascading Style Sheets Version 2 to render color backgrounds, text and images.

Syntax

The CSS2 syntax the browser uses is made up of three parts - a wml tag with an attribute called style, a property, and a value:

```
WML tag style= "property: value"
```

The property is the attribute that will be changed, and each property can take a value. The property and value are separated by a colon and surrounded by quotes.

To specify more than one property, separate each property with a semi-colon:

```
style="property1:value1; property2: value2; ... ; propertyN: valueN"
```

The browser uses CSS2 with inline styles. The style attribute can be used with every WML tag. The style attribute can contain only the CSS properties the browser supports. The code example that follows shows how to change the color:

```
<p style="color: sienna">
```

This is a paragraph

```
</p>
```

Other examples include:

```
<wml> tag:
  <wml style= "|properties|" .. </wml>
<card> tag:
  <card style= "|properties|" title="Card Title"> .. </card>
<p> tag:
  <p style= "|properties|" mode="wrap"> .. </p>
<a> tag:
  <a style= "|properties|" href="...">Link </a>
```

CSS Background Properties

The Background properties control the element background color, set an image as the background, repeat a background image vertically or horizontally, and position an image on a page.

| Property | Description | Value | Supported |
|-------------------------|--|--|------------|
| background-color | Sets the background color of an element. | <i>color-hex</i> <i>color-name</i> transparent | Supported. |

| Property | Description | Default Value | Notes |
|-------------------------|---|---------------|---|
| background-color | Sets the background color of content and padding. Used to paint the background color of the screen. | transparent | If a background color is set, set the foreground to a contrasting color, so that its content remains visible. The transparent value allows the color of the parent element to show. |

CSS Text Properties

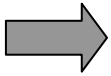
Text properties allow for control of the text appearance. You can change the color of text, increase or decrease the space between characters in a text, align text, decorate text, indent the first line in text, and more.

| Property | Description | Value | Supported |
|--------------|-------------------------|---------------------------|------------|
| color | Sets the color of text. | <i>color</i> (default) | Supported. |

| Property | Description | Useful for |
|--------------|--|-------------------------|
| color | Sets the foreground color. Used to paint the icons, brackets for text editing, text, numbers, symbols, horizontal line under the Top Line, horizontal line above the softkeys. | All displayed elements. |

Chapter 8: Web Applications

Introduction



The information in this chapter applies to both H.323 and SIP 9600 Series IP Telephones.

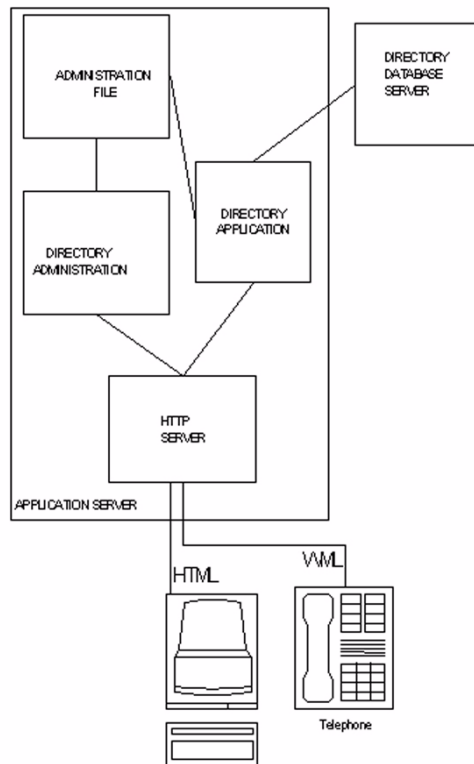
If you have a corporate database that supports the Lightweight Directory Access Protocol (LDAP), the Avaya Thin Client Directory application can communicate with that database. IP telephone users can then use their phones to search for names, telephone numbers, or other information. Using search results, users can call a person directly, store a number on Contacts list, and view more details about the person.

This chapter provides the information you need to install and administer Avaya's Thin Client Directory. It has four primary sections:

- [Application Platform Requirements](#) - Describes the operating environment for the Thin Client Directory application.
- [Installing the Thin Client Directory on the Server](#) - Lists the Avaya-provided download files needed for installation, pre-installation requirements, and step-by-step installation instructions.
- [Web Application User Interface](#) - Describes and illustrates the Directory application screens with which IP telephone users perform Directory searches and review search results.
- [Directory Database Administration Interface](#) - Describes and illustrates the administration screens with which you define LDAP attributes and configure the user interface screens.

[Figure 31](#) provides a high-level overview of the Thin Client architecture.

Figure 31: High-Level Thin Client Architecture



As [Figure 31](#) shows, the Directory application and its administration are co-resident with an HTTP server. Administration screens allow all Directory application parameters like the directory database server's IP Address, allowable search fields, etc. to be set using a PC browser. The Web browser can be co-resident on the Directory application server.

When a 9620 IP Telephone user starts a directory search, the user's browser sends the search criteria to the Directory application. The Directory application sends a query based on administered parameters to the directory database, usually located on a separate server. The directory database server then returns search results to the Directory application. The Directory application formats the results in the appropriate markup language and sends the results back to the end user. The user then has several options regarding the search results.

Application Platform Requirements

The LAN Administrator or System Administrator must provide and configure the LDAP server and the operating environment on which to install the Thin Client Directory.

The recommended server configuration is Red Hat for Linux 8.0 or greater software. This version facilitates optimal, automatic Thin Client Directory application installation. Other configurations, not recommended by Avaya, require HTTP/Apache 2.0 and PHP Version 4.2.0, with PHP Version 4.2.4 preferred.

Installing the Thin Client Directory on the Server

Pre-Installation Requirements (Apache/PHP)

Before you install the Thin Client Directory application, you must install the PHP Apache module included with Red Hat 8.0. If necessary, you can download this module for free from the

<http://www.php.net/downloads.php> Web site. Go to <http://www.php.net/downloads.php> for installation instructions. Otherwise, the distribution you download will contain its own set of installation instructions.

If you are not using Red Hat 8.0 or greater, Apache must be configured to accept PHP so the Web server recognizes it. This process differs depending on whether PHP is being installed on Linux or Windows. Further configuration variations depend on the Apache version installed.

Avaya-Provided Download Files

Two Thin Client Directory application versions are available from the Avaya Web site at <http://avaya.com/support>.

The recommended download version (avayadir-1.1-1.1.i386.rpm) is for Red Hat Linux 8.0 or greater installations only. This download allows the Red Hat Package Manager to automatically install the required directories and associated files in the correct locations.

Note:

A README file containing Thin Client Directory application installation instructions is also available from the Avaya Web site, if needed.

The other application version available is a WinZip-readable file. This file is for those installations with Windows or any other operating system not using Red Hat for Linux 8.0 or greater. This version requires that you select specific files to download. You must also perform additional server/file customization to properly install the Thin Client Directory application.

The download contains these directories:

- **avayadiradmin** - Files needed for the HTML administration part of the LDAP application. Includes the PHP files needed for administration.
- **avayadirclient** - Files necessary for the telephone/user interface. These files perform the search query and return search results to the telephone's display screen.
- **avayadir.ini** - Files containing settings that control the administration and client application. This is a protected directory that cannot be browsed. During the unzip process, it is placed in the same root as the other two sub-directories. If desired, you can move this directory outside the HTML path, providing the new path is PHP-accessible.
- **avayadirinclude** - Common files shared between the Directory administration and client (end user) interface.
- **avayadirerror** - Text files for search-related error message generation.
- **avayadirhelp** - Text files containing end user Directory assistance.

Avaya LDAP Directory Application for 9600 Series IP Telephones

The Avaya LDAP Directory Application for 4600 Series IP Telephones requires a compatibility patch for use with 9600 Series IP Telephones. This patch resolves issues with "Error – 32" and "No Results Found" messages.

Administrator Instructions

To install the patched version of the LDAP Directory Application compatible with 9600 Series IP Telephones, take the following steps:

1. If the Avaya LDAP Directory Application is not already installed on your HTTP server, install it by downloading the "4620 LDAP Directory Application Release 1.1" from the 4600 IP Telephones software downloads page at <http://www.avaya.com/support>. Once installed, note the location where the files were installed. As downloaded, the installation directory is named **avayadir**. The folder avayadir will reside under the documentRoot for the HTTP server. The location will vary depending on which HTTP server you are using.
2. Download "Required LDAP Directory Application 9600 Compatibility Patch for Release 1.1" from the 9600 IP Telephones software downloads page at <http://www.avaya.com/support>. This download is a new version of the **avayadirresults.php** file to replace the one included in the 4600 version.

3. Replace the installed **avayadirresults.php** file with the new file of the same name provided in the patch. Find the installed file in the **avayadirclient** folder under the **avayadir** folder.
4. After applying the patch it is not necessary to restart any servers or endpoints, but the browser page must be refreshed. Test the patch by searching for a known result.
5. Ensure that the LDAP search results are shown correctly on both the 9600 and 4600 Series IP Telephone displays.

End User Instructions

Instruct your Directory users that whenever they perform a search, they must:

1. Scroll down to the line of search results that shows the party they want to dial, see more information about, or add to their Contacts list.

Note:

Scrolling to the desired entry applies even when a search results in only one entry. Also, ensure that the Prompt Line shows the name of the party you want to select before proceeding to the next step.

2. Press **OK** to select that entry.
3. Press the softkey representing the action to take, for example, **AddToSD** to add the directory entry to the Contacts list, **Details** to see more information about the selected entry, or **Call** to dial the selected party.

Installing the Thin Client Directory

Installations using Red Hat for Linux 8.0 (or greater):

1. Login at the root.
2. Copy this download file to the Linux system: **avayadir-1.1-1.1.i386.rpm**.
3. Run the following command from the command line to extract the files to the **/var/www/html/avayadir** directory:
rpm -ivh avayadir-1.1-1.1.i386.rpm.
4. To enable password control for the Directory Administration application, create a directory entry in the **httpd.conf** file as follows:

The correct filename is **httpd.conf**, not **http.conf**.

```
<Directory "/var/www/html/avayadir/avayadiradmin">
AuthType Basic
AuthName "Password Required"
AuthUserFile "/var/www/password/avayadirpasswd"
Require user ldap
</Directory>
```

5. The default user/password combination is **ldap/ldap**. To change the password, run **"htpasswd /var/www/passwd/avayadirpasswd ldap."**
6. Open the file **/etc/php.ini** for editing.
7. Set the option **"short_open_tag = On"** in php.ini.
8. Uncomment the line **"extension=ldap.so"** in php.ini.
9. To finish, restart the Web server by running **"/sbin/service httpd restart."**
10. Now test everything out by pointing a browser at the newly created directory structure such as **<http://yourserver/avayadir/avayadiradmin/index.htm>**.

Installation for any other Unix-based operating system:

1. Download the winzip file and run: **unzip avayadir-1.1.zip**
2. Copy the entire tree that was created by running unzip under the documentRoot of the httpd server. For example, if your directory is **/var/www/html**, the directory created is **/var/www/html/avayadir**.
3. Use the command **"chown apache:apache /var/www/html/avayadir/avayadirini"** to change the user and group of the directory **/var/www/html/avayadir/avayadirini** to **user:apache, group:apache**.
4. Run **"chmod 755 /var/www/avayadir/avayadirini"** to change the permission of the **/var/www/html/avayadir/avayadirini** to **755**.
5. To enable password control for the Directory Administration application, create a directory entry in the httpd.conf file as follows:

The correct filename is **httpd.conf**, *not* **http.conf**.

```
<Directory "/var/www/html/avayadir/avayadiradmin">
AuthType Basic
AuthName "Password Required"
AuthUserFile "/var/www/password/avayadirpasswd"
Require user ldap
</Directory>
```

6. The default user/password combination is **ldap/ldap**. To change the password, run **"htpasswd /var/www/passwd/avayadirpasswd ldap"**.
7. Open the file **/etc/php.ini** for editing.
8. Set the option **"short_open_tag = On"** in php.ini.
9. Uncomment the line **"extension=ldap.so"** in php.ini.
10. To finish, restart the Web server by running **"/sbin/service httpd restart."**

11. Now test everything out by pointing a browser at the newly created directory structure such as
<<http://yourserver/avayadir/avayadiradmin/index.htm>>.

Installation for Windows with Apache:

1. Extract the file **avayadir-1.1.zip** to the documentRoot folder.

Note:

Making LDAP/PHP work with Apache is not necessarily easy. This procedure contains only the basics. For further information, you can download a free white paper available on the Avaya support Web site. After accessing the Avaya support Web site, make the following selections: Telephone Devices & User Agents, then IP Telephones & User Agents, then 4600 IP Telephones and SDK and Browser Information. The white paper referenced applies to both 4600 Series IP Telephones and 9600 Series IP Telephones.

The documentRoot location varies based on Web server installation. This is the directory where the Web server originates the files it serves.

2. Go to <http://www.php.net/> to determine how to install and configure PHP for your server.
3. Check your Web server's installation instructions to determine how to enable directory-level password control. We *strongly recommend* that you enable password protection for the directory administration folder **avayadiradmin**.
4. Open the file **php.ini** for editing. This file is typically located in the Windows folder **c:\windows**.
5. In php.ini, set the option "**short_open_tag = On**".
6. Uncomment the line "**extension=php_ldap.dll**".
7. Save the updated **php.ini** file.
8. To finish, restart the Web server.
9. Now test everything out by pointing a browser at the newly created directory structure such as:
<<http://yourserver/avayadir/avayadiradmin/index.htm>>.

Web Application User Interface

This section describes the user interface screens required for the Directory application. The Directory application's phone screens are accessed using the IP telephone Web Access application. Therefore, any Directory user interface screen you administer must use LDAP attributes only. Some examples are provided in [Table 22](#) in [Chapter 7: Developing Web Pages for the Browser](#) which provides detailed information about how Web pages/screens are rendered. Once you familiarize yourself with the user interface, see [Directory Database Administration Interface](#) for instructions on completing the associated administration screens.

Note:

Specific user instructions regarding the Directory application are not provided in the *respective IP Telephone User Guides*. We do not provide detailed information because the Directory user interface screens are considered part of a Web based application that you can customize.

Generic User Interface Screen Characteristics

All Directory application phone screens have similar layouts with:

- A display area.
- Line buttons down the right side related to a text entry field or data item.
- Softkeys below the display that start screen-related actions, such as **Search** or **Call**.
- Web browser navigation buttons down the right side of the display, which allow the user to move forward, back and return to the Home page.

Note:

Standard softkey labels on text entry screens are translated into the user's language. The Directory application itself, and associated help or error messages are in English only.

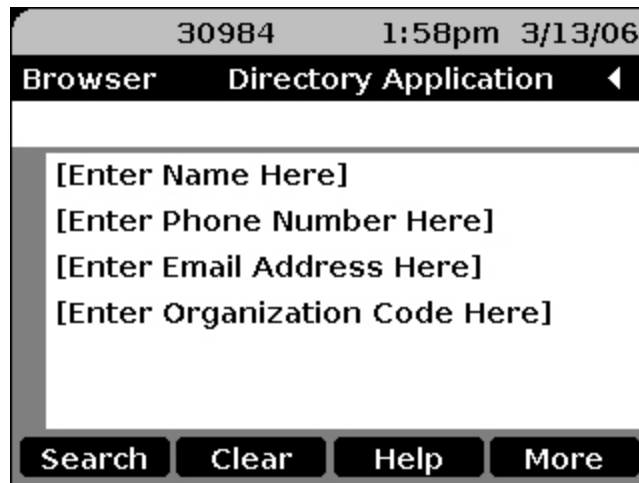
Web Application Search Screen

The Search screen displays upon user selection of the Directory application. At a minimum, administer these two user text entry fields:

- Enter Name Here
- Enter Phone Number Here

Either field provides basic search criteria. You can administer up to four additional text entry fields.

Figure 32: Sample Search Screen



The image shows a sample search screen on a handheld device. At the top, a status bar displays '30984' on the left and '1:58pm 3/13/06' on the right. Below this is a header bar with 'Browser' on the left and 'Directory Application' on the right, followed by a left-pointing arrow. The main content area contains four text input fields, each with a placeholder label: '[Enter Name Here]', '[Enter Phone Number Here]', '[Enter Email Address Here]', and '[Enter Organization Code Here]'. At the bottom of the screen is a row of four buttons: 'Search', 'Clear', 'Help', and 'More'.

The softkeys at the bottom of the screen function as follows:

- **Search** - Sends user input to the Directory application to initiate a search.
- **Clear** - Discards user input.
- **Help** - Retrieves a Help page specific to the Search screen.
- **More** - Displays additional softkeys.

Search responses take one of two forms. A successful search, one returning at least one telephone number when a name was provided as search criteria, displays the Successful Search screen. This screen offers options to call the number found, add it to a Contacts list or review more detail. An unsuccessful search, for example, no name found, error report(s) and/or unintelligible responses, displays the Directory Trouble screen.

Note:

You complete the Search Administration screen to administer the [user interface] Search screen. See [Configuring the Directory Application Search Administration Screen](#).

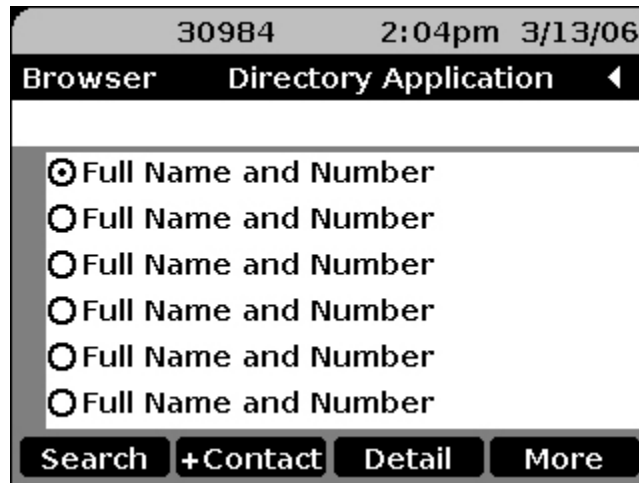
Web Application Successful Search Screen

The Successful Search screen displays when at least one match results from a user-submitted search. The top display line provides one of these messages:

X found. Select choice. where **X** = the number of matches found, or
More results - please try again and refine search.
to indicate more than 96 matches were found.

This screen's display area provides the name and phone number of up to 96 matches. If the search returns more than 96 matches, only the first 96 are shown and the rest are lost. The user can scroll through the matches using the Web browser navigation key to move forward one page. To select an entry, the user presses the Line button to the left of that entry.

Figure 33: Sample Successful Search Screen



The softkeys across the bottom of the display function as follows:

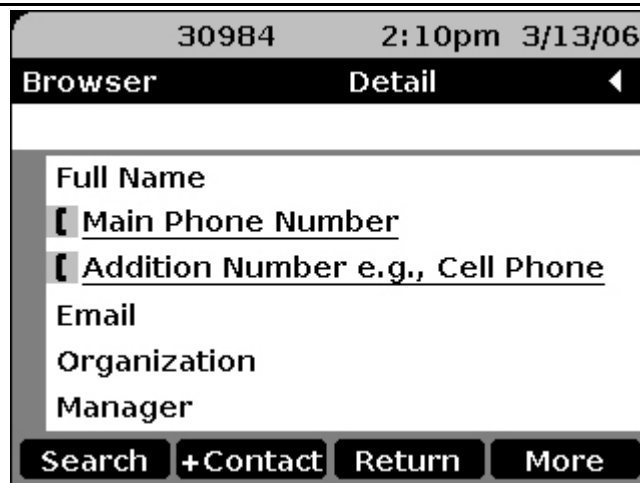
- **Search** - Displays the Search screen, to allow the user to enter new criteria and start another search.
- **+Contact** - Allows the user to add a selected name and phone number to a **Contacts** button.
- **Detail** - Displays more directory information on the person selected, such as a department, secondary contact, manager, etc. (as administered). See [Figure 34](#) for a sample Detail screen.
- **More** - Displays additional softkeys.

Web Application Detail Screen

The Detail screen displays when a user selects the **Detail** button on the Successful Search screen. Depending on how you administer it, this screen provides additional information about the person selected on the Successful Search screen. The selected person's Full Name and Main Telephone Number show on the first two lines as a default. You can administer the first two lines to show different data. You can administer four additional display lines to provide specific corporate or personal information about the person. Examples of data you can administer to appear as follows. (You can use any valid LDAP attribute in place of the sample data.)

- **Additional Phone Number** - a cell phone or other related telephone number.
- **E-mail** - the person's business e-mail address.
- **Organization** - the department or organization to which this person belongs.
- **Other** - any other pertinent information, such as the name of the person's manager or assistant.

Figure 34: Sample Detail Screen



A “click to dial” icon (📞) to the left of the Main Phone Number allows the user to call the person directly from the Detail screen. Using this icon instead of a **Call** softkey saves a softkey for your customization. Three softkeys are labeled as follows, the fourth softkey is available for your use:

- **Search** - Displays the Search screen, to allow the user to enter new criteria and start another search.
- **+Contacts** - Allows the user to add a selected name and phone number to a Contacts button.
- **Return** - Displays the Search screen, including the **Name** field entered by the user to start the most recent search.
- **More** - Displays additional softkeys.

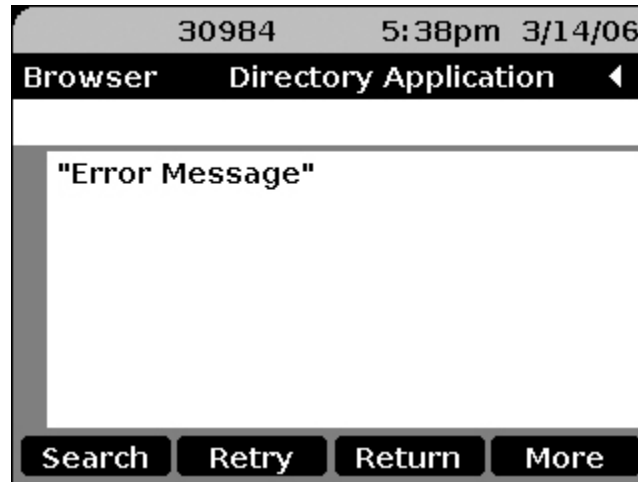
Note:

You administer the [user interface] Detail screen on the Details Administration screen. See [Configuring the Directory Application Details Administration Screen](#).

Web Application Directory Trouble Screen

Unsuccessful Directory searches occur for several reasons. For example, an inability to connect to the server for a search or finding no Directory listings that match the search criteria produce unsuccessful searches. Any search-related problem displays as an error message on the Directory Trouble screen, shown in [Figure 35](#).

Figure 35: Sample Directory Trouble Screen



The Trouble Screen's softkeys function as follows:

- **Search** - Displays the Search screen, to allow the user to enter new criteria and start another search.
- **Retry** - Allows the user to restart the search using the same search criteria.
- **Return** - Displays the Search screen, with the **Name** field the user entered to start the most recent search.
- **More** - Displays additional softkeys.

Possible reasons for search failure and the resulting messages displayed on the Trouble screen follow in [Table 19](#).

Table 19: Search Failure Causes and Corresponding Trouble Screen Error Messages

| Cause of Search Failure | LDAP Result Code | Trouble Screen Message |
|--|--|--|
| N/A. The corresponding LDAP Result Code represents a successful search. | 0 | No message displayed. The search was successful. |
| Operations/Protocol error. | 1, 2 | Operations/Protocol error. |
| Server-generated timeout. | 3 | Server timed out. |
| More than 96 entries match the Search criteria. | 4 | Size limit exceeded. |
| Various unexpected errors. You should never receive these result codes. | 5, 6, 10, 11, 12, 13, 14, 18, 19, 20, 21, 34, 36, 54, 64, 65, 66, 67, 68, 69, 71 | Invalid response. |
| Authentication not accepted. | 7, 8, 48, 49, 50, 53 | Authentication error. |
| Telephone Number not recognized. | 16, 17 | Telephone Number not recognized. |
| Object not found. | 32 | No results found. |
| Server responds with "null" as data. | N/A | No results found. |
| Server not available. | 51, 52 | Server not available. |
| Other, unspecified failure. | 80 | An unknown problem has occurred. |

| Cause of Search Failure | LDAP Result Code | Trouble Screen Message |
|---|------------------|--|
| If any of these system values are null (except DIRUSERID and DIRSRVRPWD , which are optional and might remain null), and the user tries to access the Directory application, the endpoint receives a Trouble Screen. | N/A | Insufficient Administrative Information. |
| The Directory server does not respond at all within an administrable amount of seconds. The default is 10 seconds. | N/A | Unable to contact server. |
| When the Directory application receives a request for a Search screen, it sends a Search screen in response only if supplied with the minimum administrative information. Otherwise, the endpoint receives a Trouble Screen. | N/A | Insufficient Administrative Information. |
| When the Directory application receives a request for a search from an endpoint, it initiates a connection to the directory database server. If the connection succeeds, a query is formatted and sent to the database server based on the input received from the endpoint. If the input received from the endpoint is null, the endpoint receives a Trouble Screen. | N/A | Insufficient Query Information. |
| If a connection to the database server cannot be established, or if the connection fails before a response is received, the endpoint receives a Trouble Screen. | N/A | Connection Failure. |

| Cause of Search Failure | LDAP Result Code | Trouble Screen Message |
|---|--|------------------------|
| When the Directory application receives a successful response from the database server, the endpoint receives a Successful Result screen. If no matching database entries are returned, the endpoint receives a No Match Result screen. If the database returns an error, the endpoint receives a Trouble screen. | N/A | No Match Result. |
| Cannot be determined. | 9, 22 - 31, 35, 37 - 47, 55 - 63, 70, 72 - 79, and 81 - 90 | Unknown Error. |

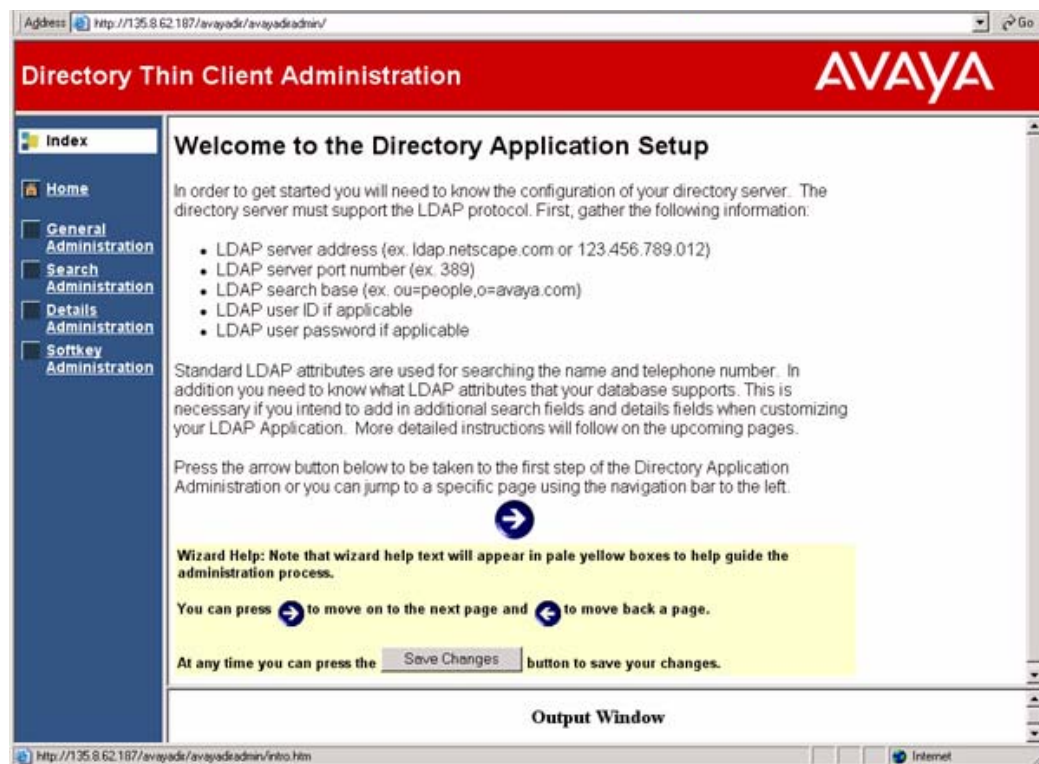
Directory Database Administration Interface

The Directory application file you download from the Avaya Web site contains five primary screens on which you administer and customize the Thin Client Directory. Additionally, each administration screen has embedded Help to guide you through the administrative process.

The primary screens are:

1. **Welcome screen** - The Home Page for administering your Directory application. This screen provides pre-administration requirements, basic administration information, links to all other administration screens, and a link to administrative Help.

Figure 36: Welcome Screen



Note:

The Welcome screen (Home page) provides a checklist of the values required to set up general administration, such as the LDAP Server Address. Ensure that you have this required information before starting to configure the General Directory application Administration screen.

2. **General Directory Application Administration screen** - You provide general information about your Directory application, such as: the Application Title displayed at the top of the first user interface screen, the LDAP Server Address, the search root and port network identification, optional User ID and Password for accessing the application, and the amount of time to be allowed for a search.
3. **Directory Application Search Administration screens** - You specify required and optional LDAP search attributes that display on the [user interface] Search screen.
4. **Details Administration screen** - You specify the detail information the user sees on the [user interface] Detail screen, such as an e-mail address for a person found, etc.
5. **Softkey Administration screen** - Allows you to optionally specify additional softkeys, to appear below the [user interface] Detail screen's display area.

Note:

The Directory application administration interface is in the English language only.

Each screen has required and optional parameters. The input fields have a definition and/or explanation of what is required to their right in the yellow areas. There can also be yellow Help areas at the bottom of a screen to help you populate the screens correctly. You can select the **Home** option from the left side of any administration screen to return to the Welcome screen (Home page).

The bottom of each screen provides navigation and save options, as shown here:



After entering the screen values, press the **Save Changes** button to save your entries. Then use the **Right Arrow** or **Left Arrow** buttons to move from that screen to another. Pressing an arrow button without first saving what you entered or changed displays a dialog box. The dialog box allows you to:

- confirm that you do not want to retain your entries or any changes you've made to existing values, or
- allows you to select **Cancel** to return to the screen and save the data.

Configuring the required information in accordance with the instructions in this section allows the Thin Client Directory application to communicate properly with the LDAP server. After configuring and saving the required information, test the Directory application to ensure that:

- the user interface screen values are correct,
- the application is interfacing properly with the LDAP server, and
- the Directory application server is interfacing properly with the end user's phone.

Configuring the General Directory Application Administration Screen

To configure the General Directory Application Administration screen:

1. From the Welcome screen, select the **General Administration** screen link. Alternately, select the **Right Arrow** icon at the bottom of the Welcome screen.

Figure 37: Directory Application Administration Screen

| General Administration | |
|--|---|
| Application Title: LDAP Directory | This is the label that will appear at the top of the initial directory search page on the telephone. All users will see this. |
| Directory Server: ldap.yourcompany.com | Put your LDAP server address here. An IP address or fully qualified DNS name here (ex. 123.456.789.12 or ldap.netscape.com) will work. |
| Topmost Distinguished Name (Search Root): ou=people, o=yourcompany.com | Put the search root (ex. "ou=people, o=avaya.com") here. |
| Port Network: 389 | The default LDAP server port is 389. The default is 636 for SSL enabled LDAP systems. |
| Max number of hits: 96 | The maximum number of results that can be returned is 96. This is due to the number of lines of text that can be displayed on the 4620. |
| Directory User id (optional): <input type="text"/> | If your LDAP directory requires a user name for searching then enter it here. |
| Directory Password (optional): <input type="password"/> | If your LDAP directory requires a password for searching then enter it here. |
| Search Time: 30 Seconds | Select the maximum search time that the application will wait for to receive results from the LDAP server. |
| Coding: Latin 1 | This is the coding that your LDAP server will use. Currently only Latin 1 is supported. |

Output Window

2. All fields except **Directory User ID** and **Directory Password** are required. [Table 20](#) shows the Administration screen fields, their associated key names, default values, and descriptions:

Table 20: Administration Screen Fields

| Field Title | Key Name | Default | Description |
|--|---------------|--|---|
| Application Title | DIRSVRNAME | 4620 Directory Application | Label appearing at top of the user interface's Directory Search screen. |
| Directory Server | DIRSRVR | Null | LDAP server address, IP Address or fully qualified DNS name. |
| Topmost Distinguished Name (Search Root) | DIRTOPDN | Null | The search root base, usually "ou=people" or o=company name. Note that spaces and other special characters might need to be treated as specified in RFC 2253, <i>Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names</i> . |
| Port Network | DIRLDAPPORT | 389 for LDAP 686 for SSI-enabled LDAP | Directory Server Port. |
| Max number of hits | | 96 | The maximum number of result entries that can be displayed on the 4620. |
| Directory User ID | DIRUSERID | Optional | User name for authorized Directory search, if required. |
| Directory Password | DIRSRVRPWD | Optional | User's password for authorized Directory search, if required. |
| Directory Search Time | DIRSEARCHTIME | 10 seconds | Maximum amount of time the application waits for search results (01 - 59 seconds). |
| Directory Coding | DIRCODING | Latin 1 | No other option is currently available. |

- Press the **Save Changes** button as stated at the bottom of the screen to save the values entered. When you complete the final administration screen, you can review all values on all screens.

Configuring the Directory Application Search Administration Screen

The Search screen's administration requires that you provide labels for the LDAP attributes that appear on the user interface Search screen. These attributes are the labeled search fields the end user sees when the Search screen displays.

Any Customer-Defined Label you create populates the **Label** value of the **Enter Label Here** text entry box on the Search Administration screen. This label also displays as the text entry prompt on the user interface Search screen. See [Figure 38](#) for an illustration of the user interface Search screen

- From the Welcome screen, select the **Search Administration** screen link. Alternately, select the **Right Arrow** icon at the bottom of the General Administration screen.

Figure 38: Search Administration Screen

| Line | LDAP Attribute | Associated Label: (Maximum 20 characters) |
|------|---|--|
| 1 | cn (Attribute represents all the versions of a person's full name) | [Enter Name Here] |
| 2 | telephoneNumber (Attribute represents the main phone number) | [Enter Telephone Number Here] |
| 3 | Common LDAP Attributes: (mail) e-mail address | [Enter Email Address Here] |
| 4 | Common LDAP Attributes: (organizationNumber) Organization Code | [Enter Organization Code Here] |
| 5 | Common LDAP Attributes: - Select an attribute from the attribute list, or enter one in the text area below - | [Enter Here] |

- Enter the **search fields, corresponding LDAP attribute names**, and associated **labels** by which your end users can search your corporate Directory. The Search Administration screen contains the fields shown in [Table 21](#).

Table 21: Search Administration Screen Fields

| Search Screen Line # | Search Field/Search Object | LDAP Attribute Name | Associated (Customer-Defined) Label (20 characters maximum) |
|----------------------|----------------------------|-------------------------------------|---|
| 1 | Name (fixed) | <code>cn</code> | Customer administrable. |
| 2 | Main Phone Number (fixed) | <code>phoneNumber</code> | Customer administrable. |
| 3 | E-mail (default) | <code>mail</code> | Customer administrable. |
| 4 | Customer administrable | <code>Customer administrable</code> | Customer administrable. |
| 5 | Customer administrable | <code>Customer administrable</code> | Customer administrable. |
| 6 | Customer administrable | <code>Customer administrable</code> | Customer administrable. |

Example: Line 3 above shows a search field “E-mail” with the LDAP attribute “`mail`.” If you enter the Associated Label in column 4 as “E-mail Address” the end users’ Search screen third line prompts: “Enter E-mail Address Here.”

You can populate fields with well-known LDAP attributes from an Avaya-provided drop-down list. [Table 22](#) provides a list of allowable attributes you can use to label such fields.

Configuring the Directory Application Details Administration Screen

The Detail screen’s administration requires you to provide the LDAP attributes to display on the user interface Details screen. These attributes are the details the end user sees about a selected person when the Details screen displays.

1. From the Welcome screen, select the **Details Administration** screen link. Alternately, select the **Right Arrow** icon at the bottom of the Search Administration screen.

Figure 39: Details Administration Screen

| Line | Displayed Attribute | Label (Max 8 Characters) |
|------|---|--------------------------|
| 1 | Common LDAP Attributes: (cn) full name (multiple formats) | |
| 2 | Common LDAP Attributes: (telephoneNumber) telephone number | |
| 3 | Common LDAP Attributes: (mobile) cellular telephone number | Cell Phone: |
| 4 | Common LDAP Attributes: (mail) e-mail address | Email: |
| | Common LDAP Attributes: (organizationNumber) Organization Code | Organization: |

2. Enter the LDAP attribute names that represent the detail information you want to display about a person found by a search. These entries appear on the user interface Detail screen, as shown in the [Web Application Detail Screen](#).

Note:

We assume that detail information has, at minimum, the name and telephone number. You can change these defaults and provide different attributes, if desired.

To override an attribute that does not appear in a drop-down list, change the Use Other radio box next to the appropriate displayed attribute from “Yes” to “No.” Then enter the custom attribute in the Other text entry field.

Labels are not required because the detail attribute should be unique enough for end user identification. If the attribute does not provide a sufficient description, you can include a label of up to 8 or less characters. Doing so, however, reduces the number of characters in the text display area accordingly.

You can populate LDAP attributes from an Avaya-provided drop-down list. [Table 22](#) provides a list of allowable attributes you can use to label such fields.

Configuring the Directory Application Softkey Administration Screen

Avaya provides specific softkeys with specific functions on each user interface screen. Where space is available in the softkey area at the bottom of a [user interface] screen, you can optionally configure up to five additional softkeys. Then you can link them to specific Detail screen display fields. For example, you might have configured “Manager” as a detail screen attribute, which shows a [found] person’s manager as part of the detail information. Linking a softkey to that field can provide a report/list of *any* person in the directory having that manager as part of his or her own detail information.

The Softkey Administration screen lists all attributes you previously defined on the Detail Administration screen as “From” attributes. You configure softkeys by providing a “To” attribute that establishes a link between the two attributes, and which is used as the softkey’s label. You can populate LDAP “To” attributes with values from an Avaya-provided drop-down list. Find these values in [Table 22](#). You can also provide a specific label for the new softkey, using the minimum number of characters that display in the screen’s softkey label area.

Figure 40: Softkey Administration Screen

The screenshot displays the 'Softkey Administration' screen within the 'Directory Thin Client Administration' application. The interface includes a sidebar with navigation options: Index, Home, General Administration, Search Administration, Details Administration, and Softkey Administration. The main content area is titled 'Softkey Administration' and contains a table for configuring softkeys. The table has three columns: 'Softkey', 'From Attribute', and 'To Attribute'. There are four rows for configuring softkeys, each with a 'Softkey' number (1, 2, 3, 4) and corresponding 'From Attribute' and 'To Attribute' fields. The 'From Attribute' field includes a 'Common LDAP Attributes' dropdown and an 'Attribute' text input. The 'To Attribute' field includes a 'Common LDAP Attributes' dropdown and an 'Attribute' text input. The 'Softkey' column also has a 'Common LDAP Attributes' dropdown. The bottom of the screen features an 'Output Window'.

| Softkey | From Attribute | To Attribute |
|---------|--|---|
| 1 | Common LDAP Attributes: (manager) DN of the entry of this person's supervisor Attribute: manager | Common LDAP Attributes: (dn) unique name of the entry, defined e Attribute: dn |
| 2 | Common LDAP Attributes: (employeeNumber) unique Personnel Number Attribute: employeeNumber | Common LDAP Attributes: (supervisorID) Supervisor Identification Attribute: supervisorID |
| 3 | Common LDAP Attributes: (departmentNumber) department ID/cost center Attribute: departmentNumber | Common LDAP Attributes: (departmentNumber) department ID/co Attribute: departmentNumber |
| 4 | Common LDAP Attributes: -- Select an attribute from the attribute list, or enter one in the text area below -- Attribute: | Common LDAP Attributes: -- Select an attribute from the attribute lis Attribute: |

Output Window

Table 22: List of Drop-Down Attributes available for Search, Query and Details Administration Screens

| Field | LDAP Attribute |
|----------------------|---|
| person | sn cn userPassword telephoneNumber description |
| organizationalPerson | title registered address telexNumber teletexTerminalIdentifier telephoneNumber internationalISDNNumber facsimileTelephoneNumber street postOfficeBox postalCode postalAddress physicalDeliveryOfficeName ou st l |
| inetOrgPerson | businessCategory carLicense departmentNumber employeeNumber employeeType givenName homePhone homePostalAddress initials labeledURL mail manager mobile pager roomNumber secretary uid userCertificate;binary x500uniqueIdentifier |