



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

16-600888  
Issue 1  
July 2006

© 2006 Avaya Inc.  
All Rights Reserved.

#### 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://www.avaya.com/support> 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://www.avaya.com/support>

#### 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.

## Contents

<b>About This Guide</b> . . . . .	<b>9</b>
About this Document . . . . .	9
Intended Audience. . . . .	9
Document Organization . . . . .	10
Issue Date . . . . .	10
How to Use This Document . . . . .	11
Terms Used in This Document . . . . .	11
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
<b>Chapter 1: IP Telephone Interfaces.</b> . . . . .	<b>17</b>
Overview . . . . .	17
Existing Interfaces. . . . .	18
<b>Chapter 2: Push Interface Overview</b> . . . . .	<b>21</b>
Introduction . . . . .	21
Push Feature Description . . . . .	22
Push Architecture . . . . .	22
The Push Flow Process . . . . .	22
The Push/Pull Process – A Two-Step View . . . . .	23
Push operation (Step 1a and Step 1b) . . . . .	23
Pull operation (Step 2a and Step 2b) . . . . .	24
Push Message Flow . . . . .	24
Step 1 - XML File Push Message Sent . . . . .	24
Step 2 – Push Agent Responds. . . . .	24
Step 3 – XML Message Parsing . . . . .	24
Step 4 – Request Launched . . . . .	25
Step 5 – Push Content Server Responds . . . . .	25
Step 6 – Message Sent to Telephone. . . . .	25
About the Push Agent. . . . .	26
HTTP Server . . . . .	26
Push Agent - HTTP POST Address. . . . .	27
Push Types. . . . .	27

<b>Chapter 3: Creating Push Messages . . . . .</b>	<b>29</b>
Introduction . . . . .	29
The Display Push Type . . . . .	29
Web Browser Features . . . . .	29
Alerts . . . . .	29
Display Push Example 1: . . . . .	30
Priorities and States . . . . .	30
Pushable vs. Non-Pushable States . . . . .	30
Successful Push Response . . . . .	31
Normal Priority . . . . .	31
Barge Priority . . . . .	32
Display Push XML Messages . . . . .	33
Display Push Message (PM). . . . .	33
Using the <postfield> Tag . . . . .	33
Display Push Example 2: . . . . .	34
Push Agent . . . . .	34
Push Content (PC). . . . .	35
The Top Line Push Type. . . . .	36
Text Message Features . . . . .	36
Alerts . . . . .	37
Top Line Push Example 1: . . . . .	37
Priorities and States . . . . .	38
Pushable vs. Non-Pushable States . . . . .	38
Successful Push Response . . . . .	38
Normal Priority . . . . .	38
Barge Priority . . . . .	39
Top Line Push XML Messages . . . . .	39
Top Line Push Message (PM) . . . . .	40
Using the <postfield> Tag . . . . .	40
Top Line Push Example 2: . . . . .	41
Push Agent . . . . .	42
Top Line Push Content (PC) . . . . .	42
Using the <Response> tag . . . . .	42
Using the <Topline> tag . . . . .	43
The Audio Push Type . . . . .	44
Interrupt Screens . . . . .	44
Audio Push Features . . . . .	44
Alerts . . . . .	45
Audio Push Example: . . . . .	45

Priorities and States . . . . .	45
Pushable vs. Non-Pushable States . . . . .	45
Successful Push Response . . . . .	46
Normal Priority . . . . .	46
Barge Priority . . . . .	46
Audio Push XML Messages . . . . .	47
Audio Push Message (PM) . . . . .	47
Using the <postfield> Tag . . . . .	48
RTP Port . . . . .	48
Push Agent . . . . .	49
Audio Push Content (PC) . . . . .	50
The Subscribe Push Type . . . . .	52
Subscribe Push Features . . . . .	52
Alerts . . . . .	53
Priorities and States . . . . .	53
Successful Push Response . . . . .	53
Subscribe XML Messages . . . . .	53
Subscribe Push Message (PM) . . . . .	54
Using the <postfield> Tag . . . . .	54
Push Agent . . . . .	55
Subscribe Push Content (PC) . . . . .	55
Using the <Response> Tag . . . . .	55
Using the <Subscribe> Tag . . . . .	55
<b>Chapter 4: Push Administration . . . . .</b>	<b>57</b>
Introduction . . . . .	57
Requirements . . . . .	57
Security . . . . .	57
Trusted Push Server List (TPSLIST) . . . . .	58
Validation Scenarios . . . . .	59
Recommendations: . . . . .	60
Subscription Service . . . . .	60
Subscriber Service . . . . .	61
Subscription List (SUBSCRIBELIST) . . . . .	61
Subscription Update . . . . .	62
Retry Timer . . . . .	62
Denial of Service Timer . . . . .	63

<b>Chapter 5: Troubleshooting the Push Interface . . . . .</b>	<b>65</b>
Avaya HTTP Header Extensions (x-Push-Status Codes) . . . . .	65
HTTP Error Messages . . . . .	67
<b>Chapter 6: About The Web Browser . . . . .</b>	<b>69</b>
Introduction . . . . .	69
Physical Attributes . . . . .	71
Display Area Specifications. . . . .	72
Web Browser Navigation . . . . .	77
Multiple Paging Indicators . . . . .	80
Scrolling . . . . .	80
Truncation Rules and Links. . . . .	82
Truncating Lines and Words . . . . .	82
Links . . . . .	83
Enabling Text Entry . . . . .	83
Text Entry Example: . . . . .	84
Text Editing Modes . . . . .	85
Character Set Support. . . . .	86
Web History . . . . .	86
Display Colors . . . . .	86
Call Interaction. . . . .	87
Requirements for Deck/Card Elements. . . . .	87
Wireless Telephony Applications (WTA) . . . . .	88
Syntax Implementation . . . . .	89
Click to Dial Functionality . . . . .	89
Add to Contacts Functionality . . . . .	94
Support for HTTP Authentication. . . . .	95
Page Loading . . . . .	96
Error Tones. . . . .	96
HTTP Protocol . . . . .	96
HTTP Header . . . . .	96
User Agent . . . . .	96
User-Agent Header String (9620 IP Telephones) . . . . .	97
User-Agent Header String (9630 IP Telephones) . . . . .	98
Web/System Interaction . . . . .	98
Idle Timer. . . . .	98
Web-Specific System Values . . . . .	99

Error Messages . . . . .	99
Summary Of WML Tags and Attributes . . . . .	101
<b>Chapter 7: Web Browser for 9620 and 9630 IP Telephones. . . . .</b>	<b>105</b>
Introduction . . . . .	105
General Background. . . . .	105
WML Tags and Attributes . . . . .	106
WML Document Skeleton . . . . .	106
Text Elements . . . . .	109
Text Formatting Tags . . . . .	110
Anchor Elements . . . . .	111
Image Elements . . . . .	112
Event Elements . . . . .	113
Task Elements . . . . .	117
Input Elements . . . . .	119
Variable Elements . . . . .	123
Character Entities . . . . .	124
Colors and Fonts . . . . .	124
Access Key Input Mode (AIM). . . . .	125
Support for Access Keys . . . . .	125
Example of Text Entry Using AIM: . . . . .	127
AIM Considerations . . . . .	129
Terminating AIM . . . . .	130
<b>Chapter 8: Web Applications . . . . .</b>	<b>131</b>
Introduction . . . . .	131
Application Platform Requirements . . . . .	133
Installing the Thin Client Directory on the Server . . . . .	133
Pre-Installation Requirements (Apache/PHP) . . . . .	133
Avaya-Provided Download Files . . . . .	133
Installing the Thin Client Directory . . . . .	134
Web Application User Interface. . . . .	137
Generic User Interface Screen Characteristics . . . . .	137
Web Application Search Screen . . . . .	138
Web Application Successful Search Screen. . . . .	139
Web Application Detail Screen . . . . .	140
Web Application Directory Trouble Screen . . . . .	141

## Contents

Directory Database Administration Interface . . . . .	144
Configuring the General Directory Application Administration Screen . . . .	146
Configuring the Directory Application Search Administration Screen. . . .	148
Configuring the Directory Application Details Administration Screen. . . .	150
Configuring the Directory Application Softkey Administration Screen . . . .	151
<b>Chapter 9: Advanced Features . . . . .</b>	<b>155</b>
Introduction . . . . .	155
Character Entities . . . . .	155
Image Support . . . . .	156
WBMP Images . . . . .	156
JPEG Images. . . . .	156
Image Rendering. . . . .	157
Scrolling Through Images. . . . .	157
Linked Images . . . . .	158
Image Size & Memory Requirements. . . . .	158
Image Justification . . . . .	159
Image Size . . . . .	161
Number of Images Supported. . . . .	161
Support for Cascading Style Sheets . . . . .	161
Cascading Order. . . . .	162
CSS2 Specifications. . . . .	168
Syntax . . . . .	168
CSS Background Properties . . . . .	169
CSS Text Properties . . . . .	169
<b>Index . . . . .</b>	<b>171</b>

# 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 applies to software Release 1.0 for the 9620 and 9630 IP Telephones and covers 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 the 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:

<a href="#">Chapter 1: IP Telephone Interfaces</a>	Describes the available Avaya IP Telephone interfaces.
<a href="#">Chapter 2: Push Interface Overview</a>	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.
<a href="#">Chapter 3: Creating Push Messages</a>	Describes the message types that can be sent (pushed) to an IP telephone in detail, and provides setup requirements and examples.
<a href="#">Chapter 4: Push Administration</a>	Covers Push security features and recommendations, and server setup.
<a href="#">Chapter 5: Troubleshooting the Push Interface</a>	Describes messages received during Push interface setup or processing, and provides suggested resolutions.
<a href="#">Chapter 6: About The Web Browser</a>	Provides a general overview of the Web browser and application setup.
<a href="#">Chapter 7: Web Browser for 9620 and 9630 IP Telephones</a>	Provides information about creating and customizing Web sites for viewing on the 9620 and 9630 IP Telephones. Also describes the current capabilities and limitations of the Web browser.
<a href="#">Chapter 8: Web Applications</a>	Provides information on administering an LDAP directory for the 9620 and 9630 IP Telephones.
<a href="#">Chapter 9: Advanced Features</a>	Provides information about creating and customizing Web sites using color cascading style sheets.

---

## Issue Date

This is the first release of this document, issued in 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 to get a thorough understanding of the interfaces or use the Table of Contents or Index to locate specific features.

---

## Terms Used in This Document

Term	Description
<b>ACM</b>	<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.
<b>AIM</b>	<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.
<b>Alerts</b>	An optional notification such as a series of ring pings to alert the user to an incoming Push Message.
<b>Application Area</b>	The usable display area between the Prompt Line and Softkey labels.
<b>Application Line</b>	The display area line that indicates application-specific messages.
<b>Card</b>	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).
<b>CDATA</b>	Text, which can contain numeric or named character entities. CDATA, a DTD data type, is used only in attribute values.
<b>CSS2</b>	<i>Cascading Style Sheets Version 2.</i>
<b>Deck</b>	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.
<b>DTD</b>	<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.
<b>Elements</b>	Elements are the essential components that make up a single WML document.

Term	Description (continued)
<b>FLOW</b>	The flow type represents “card-level” information. In general, flow is used anywhere general markup can be included.
<b>Focus</b>	Since the phone has no mouse to navigate around the screen, the line buttons if shown, or the <b>OK</b> 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 <b>Line</b> or <b>OK</b> buttons).
<b>Frame</b>	The area in which the Web page is displayed.
<b>href</b>	The <b>href</b> attribute refers to either a relative or an absolute Uniform Resource Locator.
<b>HTML</b>	<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.
<b>HTTP</b>	<i>Hyper Text Transfer Protocol</i> , used to request and transmit pages on the World Wide Web.
<b>Interrupt Screen</b>	A screen that automatically accompanies a standalone audio push. Interrupt screens provide the user with specific information about terminating the audio push.
<b>IP</b>	<i>Internet Protocol</i> – a suite of information exchanged message sets widely used for data transmission and increasingly used for the transmission of voice.
<b>Link</b>	The URI that is used to chain cards together.
<b>mode</b>	Push Priority type – normal or barge – to distinguish between emergency messages and ideal messages.
<b>NMTOKEN</b>	<i>A name token</i> , containing any mixture of name characters, as defined by the XML.
<b>PBX</b>	<i>Private Branch Exchange</i> – A generic name for a premise-based switch supporting telephony features owned by an enterprise.
<b>PCDATA</b>	<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.
<b>Prompt Line</b>	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.
<b>Push Agent or PA</b>	The telephone software that is capable of receiving a Push Message from a server (Push Initiator).

Term	Description (continued)
<b>Push Initiator or Push Server</b>	A Web application that is capable of transmitting the Push Message to the Push Agent.
<b>Push Content (PC)</b>	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.
<b>Push Message (PM)</b>	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.
<b>Push State</b>	When the telephone is in busy state such as an active phone call or user entering information in the Contacts Application.
<b>Registration</b>	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.
<b>RTP Audio</b>	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.
<b>SUBSCRIBELIST</b>	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).
<b>Subscription Servers</b>	A server or a database that stores the information for a Push-enabled IP telephone such as IP Address, Extension, MAC Address, etc.
<b>Title Line</b>	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.
<b>Top Line</b>	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 9620 and 9630 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.
<b>TPSLIST</b>	<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)
<b>Trusted Push Server (TPS)</b>	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.
<b>Trusted Receive Server (TRS)</b>	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.

---

**3 of 4**

Term	Description (continued)
<b>Type</b>	Type specifies a tag or attribute.
<b>User Agent</b>	Software that interprets WML, WMLscript, WTAI and other forms of codes.
<b>VDATA</b>	A DTD data type representing a string that can contain variable references. This type is only used in attribute values.
<b>W3C</b>	<i>World Wide Web Consortium.</i>
<b>WAP</b>	<i>Wireless Application Protocol.</i> An open global standard for wireless solutions that includes WML.
<b>WBMP</b>	WBMP is a bitmap graphic format that is required for the integration of graphics into WML pages.
<b>WML</b>	<i>Wireless Markup Language</i> is a subset of XML, used by the Avaya IP Telephone Web browser to communicate with WML Servers.
<b>WML homedeck</b>	The WML start page, derived from Homepage.
<b>WMLscript</b>	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.
<b>WML tags</b>	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.
<b>WTA</b>	<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.
<b>WTAI</b>	<i>Wireless Telephony Application Interface</i> is a set of interfaces that extend the WAE (Wireless Application Environment) to include telephony applications.
<b>x-Push-Status</b>	The Push Agent HTTP extension header. This extension is used by the Push Agent to send the Push Message status to the Push Initiator.
<b>XML</b>	<i>eXtensible Markup Language.</i> W3C's standard for Internet Markup Languages. WML is one of these languages.
<b>xml:lang</b>	The <b>xml:lang</b> 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.
<b>Conference</b>	In step by step procedures, words shown in bold represent a single telephone button that should be pressed/selected.
<code>message</code>	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 16-300698)*

This guide provides 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.

---

### 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:

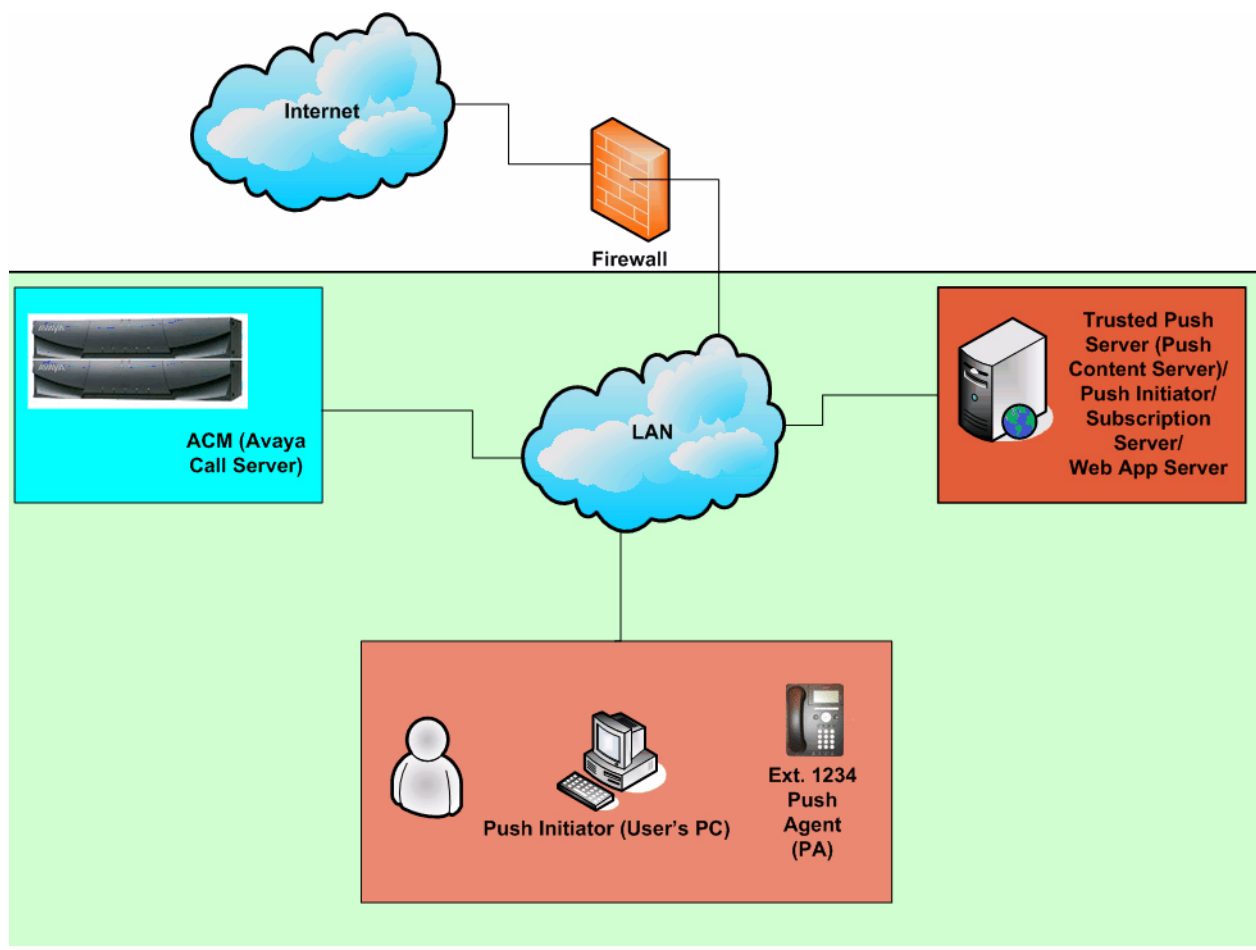
- IETF RFC 2616: <http://www.ietf.org/rfc/rfc2616.txt?number=2616>
- IETF 1945: <http://www.ietf.org/rfc/rfc1945.txt?number=1945>

# 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: Web Browser for 9620 and 9630 IP Telephones](#),
- [Chapter 8: Web Applications](#), and
- [Chapter 9: Advanced Features](#).



# Chapter 2: Push Interface Overview

---

## Introduction

Push is the ability for an application to send content to the Web browser, to the Top Line of the display, or to the audio transducers of 9620 and 9630 IP Telephones.

With the Push interface, the application can spontaneously “push” 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 weather alerts
- Building intelligent databases to target information to an individual or groups of phones

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, called **Audio** push types
- 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, or an audio stream.

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 TPSSLIST 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

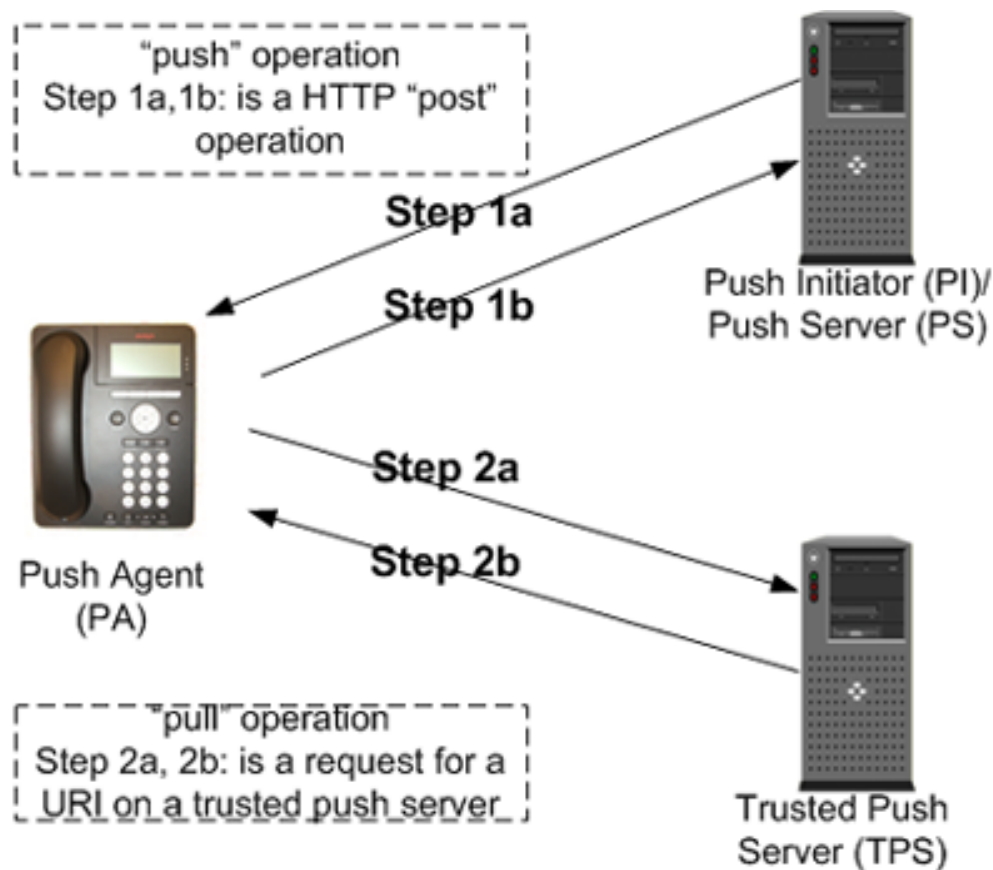
The Push framework is a two-step process - a “push” operation followed by a “pull” operation. See [Figure 3](#) for a visual reference to the steps involved in the Push/Pull process.

### 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

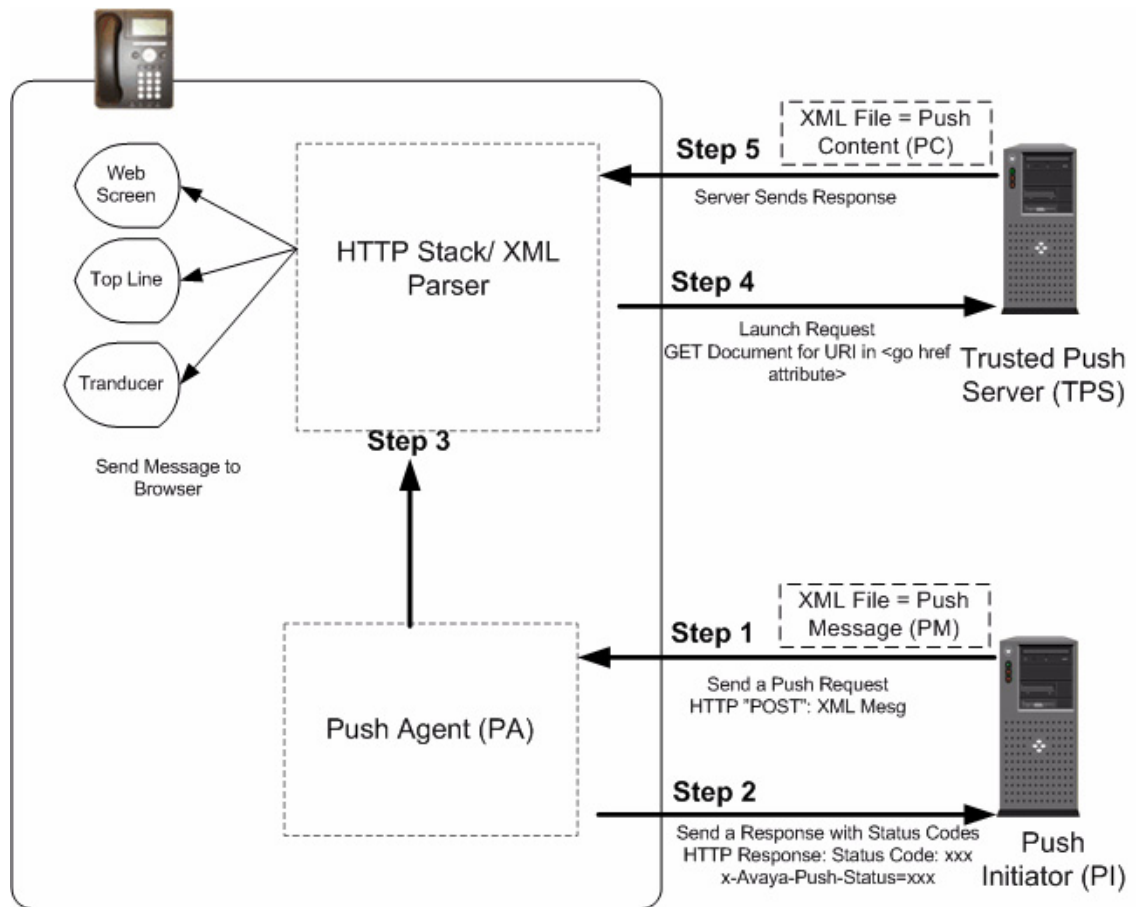
The Push Agent sends a response back to the Push Initiator with HTTP status codes. The response also contains an HTTP header extension called “x-Avaya-Push-Status” code. The x-Avaya-Push-Status indicates the outcome of a push request back to the Push Initiator. x-Avaya-Push-Status codes respond with 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 as 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.

## 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.

---

## 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 TPSSLIST 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 TPSSLIST 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-URL.

---

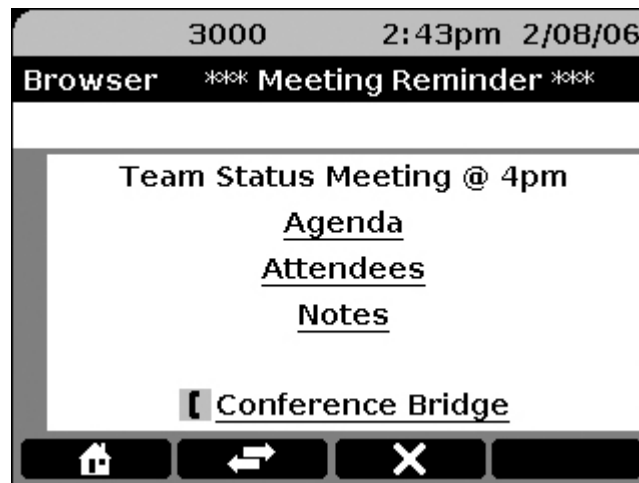
## Push Types

The 9620 and 9630 IP Telephones support the Push types described in this section.

**Display push type** - 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](#) on page 29 for more information about this type of push.

---

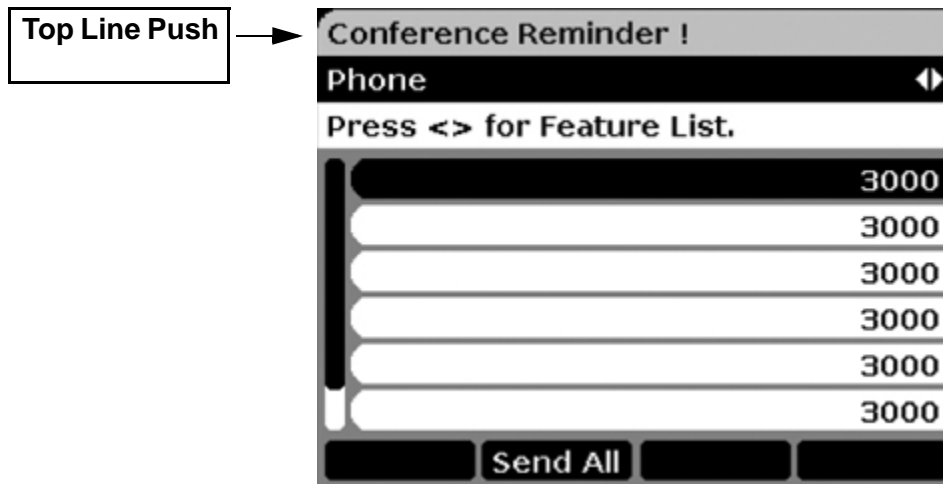
**Figure 5: Full Screen (Display) Push**



**Top Line push type** - 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](#) on page 36.

---

**Figure 6: Top Line Push**



---

**Audio push type** - The phone can receive an audio stream from an application outside the context of a telephone call. The Audio Push Message can be accompanied with an optional alert. For more information on Audio push, see [The Audio Push Type](#) on page 44.

**Subscribe push type** - 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](#) on page 52 and [Chapter 4: Push Administration](#) on page 57.

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.

# Chapter 3: Creating Push Messages

---

## Introduction

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
- Audio Push
- Subscribe Push

---

## 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.

---

## 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.

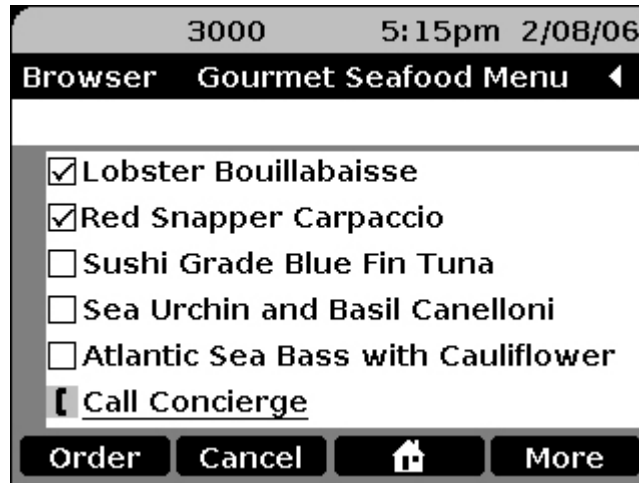
---

## 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 Display pushes to an IP telephone:

- When the user is in text entry mode on a Web text entry screen or Contacts text entry screen (for **normal** priority only).
- When the telephone is in the process of restoring a retrieved backup file.
- When a Local Procedure has been initiated.

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.

## Creating Push Messages

If the Web is in “text edit mode” 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”
<code>x-Push-Status</code>	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 two exceptions. A **barge** Display push is rejected only when the telephone is in either of these **non-pushable** states:

- When the telephone is in the process of restoring a retrieved backup file, or
- When a Local Procedure has been initiated. See the *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Administrator Guide* on the <http://www.avaya.com/support> Web site for more details on Local maintenance procedures.

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 normally 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 normally 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: Web Browser for 9620 and 9630 IP Telephones](#) 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 = <b>display</b> .
	mode	<b>normal</b> or <b>barge</b> priority.
	<go href=.../>	A fully qualified URL - to a valid WML file for Display pushes. Cannot exceed 1024 characters.
	method	HTTP <b>get</b> or <b>post</b> methods for Push Content.

---

## 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. Then the Push Agent 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: Web Browser for 9620 and 9630 IP Telephones](#) 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>

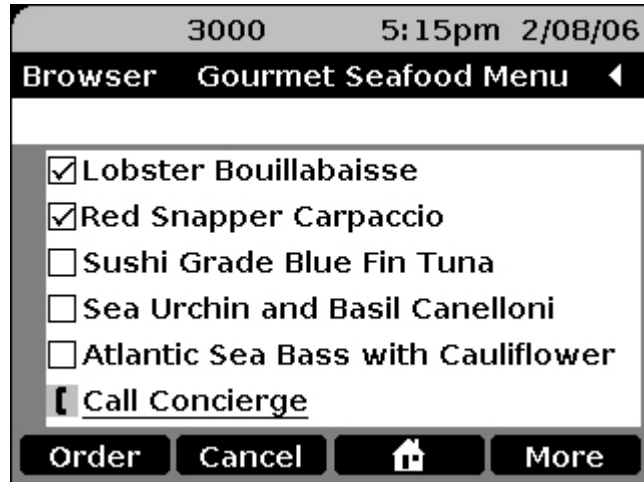
    </card>

</wml>
```

## Creating Push Messages

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.

---

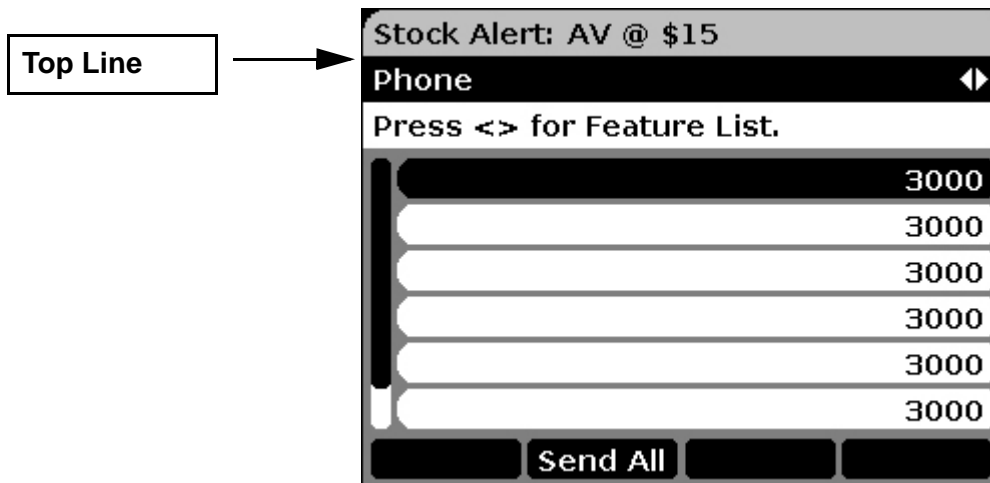
### Top Line Push Example 1:

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.
- When a Local Procedure has been initiated. See the *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Administrator Guide* on the <http://www.avaya.com/support> Web site for more details on local maintenance procedures
- When the telephone is in any text entry mode (for normal priority).
- When any Call Appearance is in the Alerting mode (for normal priority).
- When the telephone is in Soft-Hold call state (for normal priority).
- When a Top Line message cannot be displayed due to call-related messages taking Top Line precedence.

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, or
- When a Local Procedure has been initiated. See the *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Administrator Guide* on the Avaya support Web site for more details on Local maintenance procedures.

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: Web Browser for 9620 and 9630 IP Telephones](#).

**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 = <b>Top Line</b> .
	mode	<b>normal</b> or <b>barge</b> priority.
	<go href=.../>	A fully qualified URL - to a valid XML Push Content file for Top Line pushes. Cannot exceed 1024 characters.
	method	HTTP <b>get</b> or <b>post</b> methods.

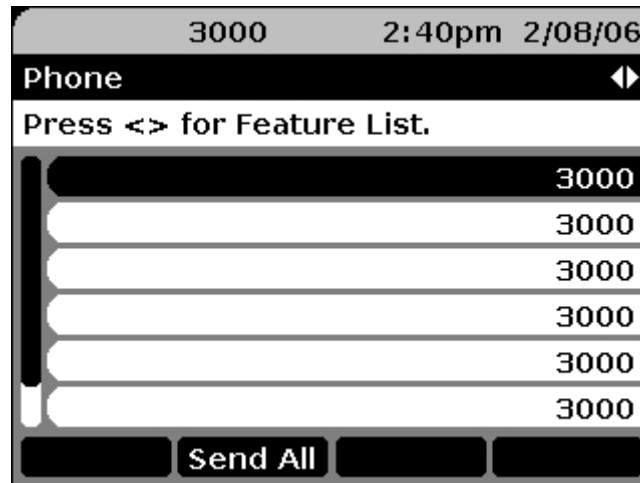
---

## 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.

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.

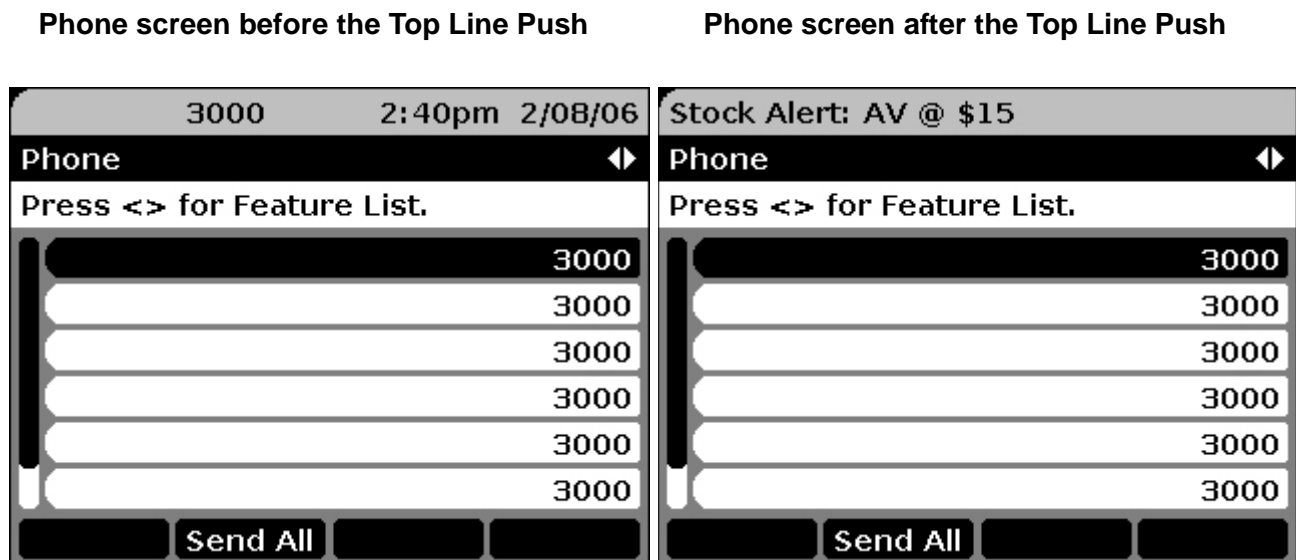
If the length of the message exceeds the given pixels, the entire message is divided. The message then fits on a single line, with the initial text and the remaining text alternating on the top display line.

The Top Line message can be 312 characters long. If the text is longer than 312 characters, all characters beyond 312 are ignored and not displayed.

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 Audio Push Type

Use an Audio push when you need to stream a particular audio message to the telephone. The 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 telephone's handling of pushed audio content mirrors the telephone's handling of 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.

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, alert the user to an incoming call. The **Volume Up** and **Volume Down** buttons work normally if the user presses them while listening to pushed audio content.

---

## Interrupt Screens

Interrupt screens automatically accompany standalone 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.

---

## Audio Push Features

The 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

---

## Alerts

An 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.

---

## Audio Push Example:

An 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

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.

---

## Pushable vs. Non-Pushable States

The following are **non-pushable** states for an IP telephone with the normal 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 Procedure has been initiated. See the *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Administrator Guide* on the <http://www.avaya.com/support> Web site for more details on Local maintenance procedures.
- The telephone is already broadcasting 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	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 Procedure has been initiated (For more information, see the *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Release 1.0 Installation and Maintenance Guide* on <http://www.avaya.com/support>).
- When the telephone is already broadcasting received pushed audio content.
- 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 a **barge** 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 Hold, and the audio is streamed immediately. If the call is still on Hold when the pushed audio stream ends, the user can select the appropriate Call Appearance button to continue the original call. Note that the far-end party does not get any indication that the current call is being placed on hold.

## Audio Push XML Messages

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

### 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"
  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: Web Browser for 9620 and 9630 IP Telephones](#).

**Table 3: 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 = <b>audio</b> .
	mode	<b>normal</b> or <b>barge</b> priority.
	<go href=.../>	A fully qualified URL - to a valid XML file for Audio push Content. Cannot exceed 1024 characters.
	method	HTTP <b>get</b> or <b>post</b> 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 POST http://server_IP_address?rtpLPort=XXXX

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

GET Parameter	Description
<b>rtpLPort</b>	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 code excerpt associated with [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. Then the Push Agent requests the Push Content from the Trusted Push Server using the URL.

**Note:**

For more information on Trusted Push Servers, see [Security](#) on page 57.

## Audio Push Content (PC)

A special XML file for RTP streaming must initiate Audio push.

```
<?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>
```

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) $\mu$ -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 can contain <AudioTimer> and <Url> tags. 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	X (seconds)	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. The <Url> tag has the following attributes:

Attribute	Value	Description
<b>href</b>	<i>string</i>	RTP Streaming URI Format.

RTP Streaming URI Format	IP Address	Port
<b>RTPRx</b> denotes "Receive" by the phone	rtpserver_ip_address is the IP address of the RTP streaming server.	Port Number separated by a colon. This is the receive port number or rtpLPort value from the GET request.

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

Reserved URIs	Description
<b>RTPRx://STOP</b>	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 <b>barge</b> 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.

---

## 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 character model number which is fixed and does not change
  - For a 9620 IP Telephone, the Set ID is **9620D01A**
  - For a 9630 IP Telephone, the Set ID is **9630D01A**

---

## 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://subscription_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.

**Note:**

For more information on the <postfield> tag, see [Chapter 7: Web Browser for 9620 and 9630 IP Telephones](#).

**Note:**

Postfield tags must be sent if HTTP POST method is used.

**Table 4: 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 <b>get</b> or <b>post</b> methods.

---

## 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.

## 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.

---

## Creating Push Messages

The <Url> tag has the following attributes:

Attribute	Value	Description
<code>href</code>	<i>string</i>	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.

# Chapter 4: Push Administration

---

## Introduction

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

**Note:**

Consult your System Administrator if appropriate to modify push-related settings. Also, see the *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

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.

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 these two topics in more detail.

---

## 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.

## Trusted Push Server List (TPSLIST)

Depending upon the TPSLIST setting in the script file, the Push Message is either accepted for additional processing or denied.

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.

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.

1. The administrator sets the TPSLIST values in the 46xxsettings.txt script file on the HTTP server. See the *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephones Administrator Guide* on the <http://www.avaya.com/support> Web site for more details.
2. If the value of TPSLIST is null, Push Messages cannot be received and the Push interface is disabled.
3. If the value of TPSLIST contains only a forward slash character (/), then all Push Messages are accepted, meaning any Push server is considered a Trusted Push Server.
4. Wild cards such as asterisks are not allowed for URL strings in the TPSLIST.

If the pushed URI string (the value of the Push Request's **href** attribute in the <go> tag does not begin with the proper HTTP schema, **http://**, the URI string is rejected and the response back to the Push Initiator contains 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 5: URI Examples

Validation String (Validation string Interpreted as)	Pushed URI string 'X' means <b>failure</b> and 'OK' means <b>success</b> .			
/ = 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/ /push/
"company" ("company /")	X	X	X	X
"company.com" ("company.com/")	OK	OK	OK	X
"support.company.com" ("support.company.com/")	X	OK	X	X
"company.com/push" ("company.com/push")	X	X	OK	X
"/push" ("/push")	X	X	OK	X (OK for "http:// www.hacker.com /push")
"company.com/xy/push" ("company.com/xy/push")	X	X	X	X
"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
"/" ("/")	OK	OK	OK	OK

---

## Recommendations:

If the domain from [Table 5: URI Examples](#) 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-character model number which is fixed and does not change
  - For a 9620 IP Telephone, the Set ID is **9620D01A**
  - For a 9630 IP Telephone, the Set ID is **9630D01A**

The HTTP Push Subscription Message Syntax is as follows:

```
GET:http//<subscription_server>/<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.

## Push Administration

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 *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/subscribe/,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

## 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 6: 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 Services Disabled”	Ask your system administrator to administer the TPSSLIST value on the HTTP Server. See <a href="#">Push Administration</a> for more information.
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.

1 of 2

**Table 6: x-Avaya-Push-Code Responses (continued)**

<b>x-Push-Status code</b>	<b>Reason Phrase (Description)</b>	<b>Resolution (“N/A”: Avaya IP Telephones never send the status code)</b>
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 <a href="#">Chapter 3: Creating Push Messages</a> 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 <a href="#">The Audio Push Type</a> on page 44 for more information.
301	“XML document not valid”	Check the proper XML schema as described in this API. See <a href="#">Chapter 3: Creating Push Messages</a> for more information.
302	“XML document not well-formed”	Make sure that all the XML tags are properly formed. See <a href="#">Chapter 3: Creating Push Messages</a> for more information.
303	“No element found”	Make sure that all the XML tags are properly formed. See <a href="#">Chapter 3: Creating Push Messages</a> for more information.
304	“Unknown encoding”	Make sure that all the XML tags are properly formed. See <a href="#">Chapter 3: Creating Push Messages</a> for more information.
305	“Invalid root element”	Make sure that all the XML tags are properly formed. See <a href="#">Chapter 3: Creating Push Messages</a> 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.
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.
503	“Subscription Service Unavailable”	Ask your system administrator to administer the SUBSCRIBELIST value on the HTTP Server. See <a href="#">Chapter 4: Push Administration</a> for more information.

**2 of 2**

## 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 7: HTTP Error Messages**

<b>Status Code</b>	<b>Cause of Failure</b>	<b>Failure Message Displayed to the User/Resolution (“N/A”: Avaya IP Phone will never send the status code)</b>
400	Bad Request	Due to incorrect syntax the server could not understand the request.
401	Unauthorized	The request requires user authentication.
402	Payment Required	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
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.

1 of 2

**Table 7: HTTP Error Messages (continued)**

<b>Status Code</b>	<b>Cause of Failure</b>	<b>Failure Message Displayed to the User/Resolution (“N/A”: Avaya IP Phone will never send the status code)</b>
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.
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.
<b>2 of 2</b>		

# Chapter 6: About The Web Browser

---

## Introduction

This chapter provides general information about and requirements for developing Web browser applications. The information covered applies to all 9600 Series IP Telephones to which Web application development applies, specifically:

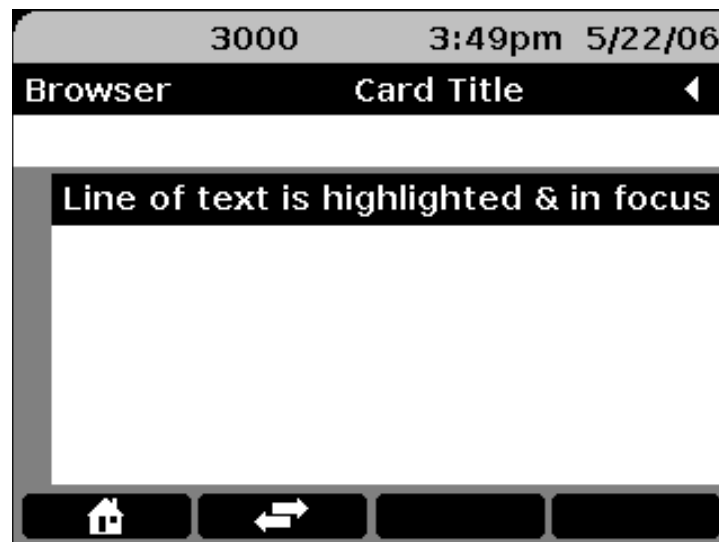
- 9620
- 9630

Basic browser principles apply to all telephones capable of maintaining a Web browser. This chapter describes those common characteristics. For specific information about Web application development for color or cascading style sheets, see [Chapter 9: Advanced Features](#). For specific information about Web application development for the other 9600 Series IP Telephones to which Web application development applies, see [Chapter 7: Web Browser for 9620 and 9630 IP Telephones](#).

[Figure 12](#) provides a schematic view of the standard Web browser display for the 9600 Series IP Telephones, and the softkeys associated with that particular display. [Figure 13](#) shows the display area in relation to the telephone itself and the Navigation cluster directly below the softkeys.

---

**Figure 12: Web Browser Display and Softkeys**



---

**Figure 13: 9620 IP Telephone**



---

**Note:**

The 9600 Series IP Telephones have differing numbers of line buttons. The 9620 has three line buttons while the 9630 has six. [Figure 13](#) shows a 9620 IP Telephone.

---

## Physical Attributes

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

- Six Line buttons are available for Web applications on the 9630. Three Line buttons are available for Web applications on the 9620.

**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.

- 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 bottom display line presents the available softkey selections. Four softkeys can be displayed at one time.
- 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 with the same functionality as a call appearance (Line button). The **OK** button provides an alternate way to select a line, a field, or an item.
- Text is in black characters when displayed against a white or light gray background. Text is in white characters when displayed against a black or dark gray background.
- When this document states that an area is **selected or highlighted**, this means the area is displayed in reverse video with a black background and white characters. The content of the area is otherwise unchanged.

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

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.

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.

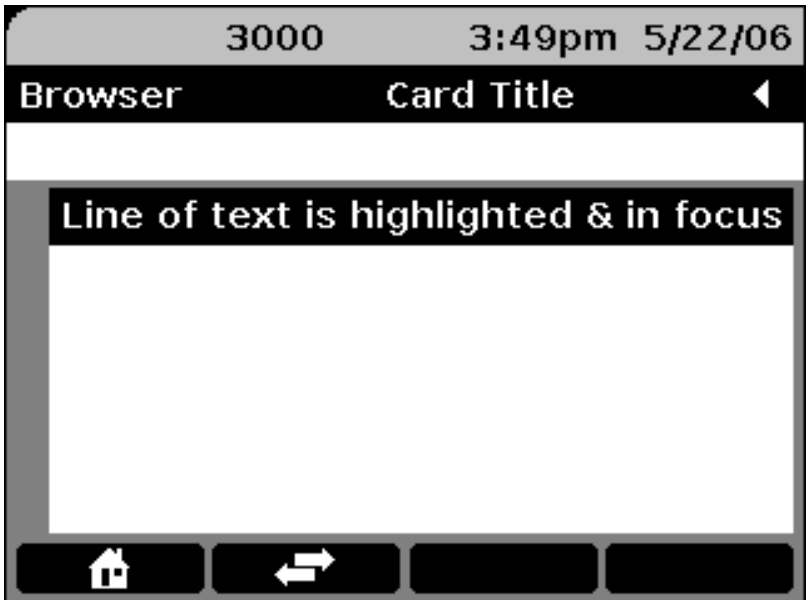
A maximum number of 96 lines per card is supported. A line corresponds to text rendered in the usable area of the display screen that contains application lines.

For deck storage, 3.5 Mbytes of memory is allocated to the Web browser.

# Display Area Specifications

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. [Figure 14](#) depicts how the Web browser renders content. [Figure 15](#) shows the height and width of the Application Area.

Figure 14: Content Rendering



Displays on 9xxx telephones are framed by a 1-pixel-wide black border, making the usable area inside 2 pixels smaller in the horizontal (H) and vertical (V) dimensions, as specified in [Figure 15](#):

Figure 15: Display Resolution and Usable Display Area

	9620	9630
Display Resolution (H x V)	320 x 160	320 x 405
Usable Area (H x V)	318 x 158	318 x 238

[Figure 16](#) is a schematic of the usable area divided by function. Not all telephone models are required to support all of the functional areas:

**Figure 16: Display Area Breakdown**

Top Area	Top Line	
	Title Line	
	Prompt Line	
Application Area	S c r o l l  B a r	Application Line 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 usable area.

The Top Line of four grayscale phones displays as black text on a light gray background. The Title Line of four grayscale phones displays white text on a black background. The Prompt Line of four grayscale phones displays black text on a white background.

The Application Area displays application content, softkey labels, and a Scroll Bar, if needed. The entire Application Area background for the four grayscale phones is always a single color, which is dark gray by default.

Text and icon content always display on Application Lines. Application lines are **23**-pixel-high horizontal areas, to align with any Line Buttons or LEDs located to the right of the display. There is a **5** pixel vertical gap of above the first Application Line and below the last Application Line for the 9620 and 9630. There is also a relatively large horizontal gap between the left side of the Application Lines and the left-side border pixel, where the Scroll Bar can be displayed. A smaller horizontal gap exists between the right side of the Application Lines and the right-side border pixel. Text and icons may **not** be displayed in gap areas around the Application Lines, although images, for example, in the Web browser, may extend across Application Lines. Text displayed directly on the Application Area background is black unless specified otherwise by the Web application.

A Scroll Bar displays vertically to the left of the Application Lines only if the display content does not fit the viewable area. When displayed, the Scroll Bar extends from the top Application Line to the bottom Application Line, inclusive, with the same gaps above and below. Small gaps remain on either side to separate the Scroll Bar from the left-side border pixel and from any other content displayed in the Application Area.

[Figure 17](#) specifies the use of pixels horizontally across the usable display area containing Application Lines.

**Figure 17: Horizontal Pixels**

	Gap	Scroll Bar	Gap	Application Line Length	Gap
320-pixel-wide displays (9620, 9630):					
<b>Width (pixels)</b>	4	10	3	297	4
<b>Pixel Numbers</b>	2-5	6-15	16-18	19-315	316-319

A single row of Softkey Labels is displayed below the Application Lines and the Scroll Bar (if present). Softkey Labels are **20** pixels high, and there is a gap below the labels to separate them from the bottom border pixel. By default, Softkey Labels are black with white text.

[Figure 18](#) specifies the use of pixels horizontally across the usable display area that contains Softkey Labels.

**Figure 18: Softkey Label Horizontal Pixels**

320-pixel-wide displays (9620, 9630)	Gap	Label 1	Gap	Label 2	Gap	Label 3	Gap	Label 4	Gap
Width (pixels)	4	75	3	75	4	75	3	75	4
Pixel Numbers	2-5	6-80	81-83	84-158	159-162	163-237	238-240	241-315	316-319

[Figure 19](#) specifies the vertical use of pixels in the Application Area and Scroll Bar.

**Figure 19: Application Area & Scroll Bar Vertical Pixels**

			9620		9630	
			Number of Pixels	Pixel Numbers	Number of Pixels	Pixel Numbers
		Border	1	1	1	1
		Top Line	20	2-21	23	2-24
		Title Line	20	22-41	23	25-47
		Prompt Line:	20	42-61	23	48-70
		Gap:	5	62-66	5	71-75
9630 Scroll Bar 135 pixels	9620 Scroll Bar 66 pixels	Application Line 1:	23	23	23	76-98
		Application Line 2:	23	23	23	99-118
		Application Line 3:	23	23	23	122-144
		Application Line 4:	n/a	23	23	145-164
		Application Line 5:	n/a	23	23	168-190
		Application Line 6:	n/a	23	20	191-210
		Gap:	5	133-137	5	211-215
		Softkey Label Line:	20	138-157	20	216-235
		Gap:	2	158-159	4	236-239

Softkey Labels are positioned above the associated softkeys buttons.

The background of a given area refers to all pixels in that area that are not being used to present text glyphs, icons, or other objects.

## About The Web Browser

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.

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 3rd 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.

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

All browser text is in the Prima Sans Bold font, 14 pixel.

On average, the character counts for display lines are:

- Top line- Lower case, 312 pixels = 36 characters
- Title line - Mixed case, 312 pixels = 35 characters
- Prompt line - Upper case, 312 pixels = 30 characters
- Each Application line - Numbers, 292 pixels = 30 digits/characters
- Softkeys -as follows:
  - 8 characters with 1 uppercase and 7 lowercase characters, or
  - 7 characters with 2 uppercase characters, 5 lowercase characters, and 1 space, or
  - 7 characters with 1 uppercase character, 6 lowercase characters, or
  - 8 characters with 7 alpha characters, 1 digit, and 1 space.

Display line character counts are based on:

- approximately 8.5 pixels per character for lowercase,
- approximately 9 pixels per character for mixed case,
- approximately 10.4 pixels per character for uppercase,
- Approximately 9.6 characters per digit for numeric, and
- 5 pixels per space for spaces.

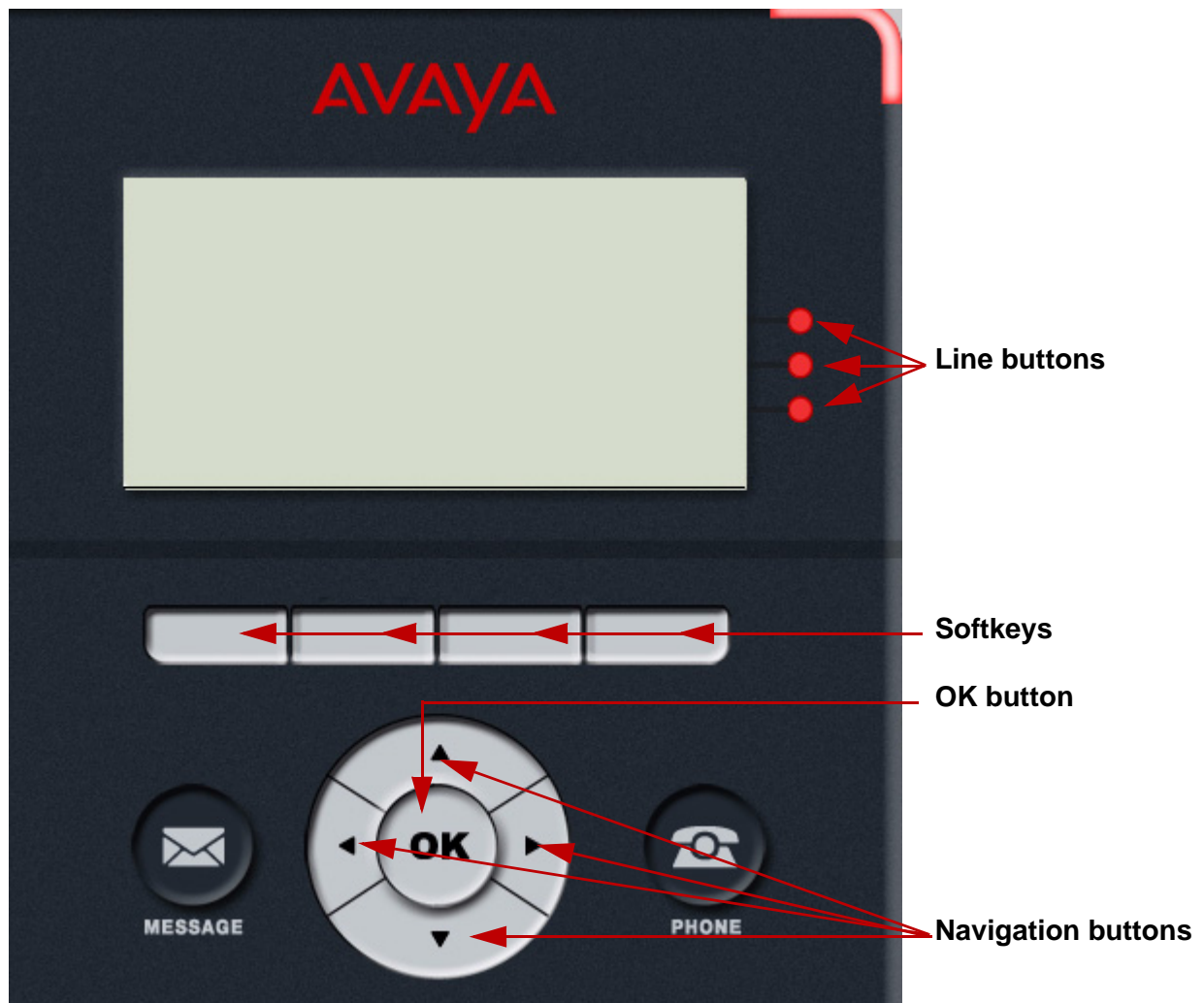
---

## Web Browser Navigation

Web navigation uses the Navigation cluster, the **OK** button, softkeys and Line buttons, as illustrated in [Figure 20](#).

---

**Figure 20: Navigation Keys and Buttons**

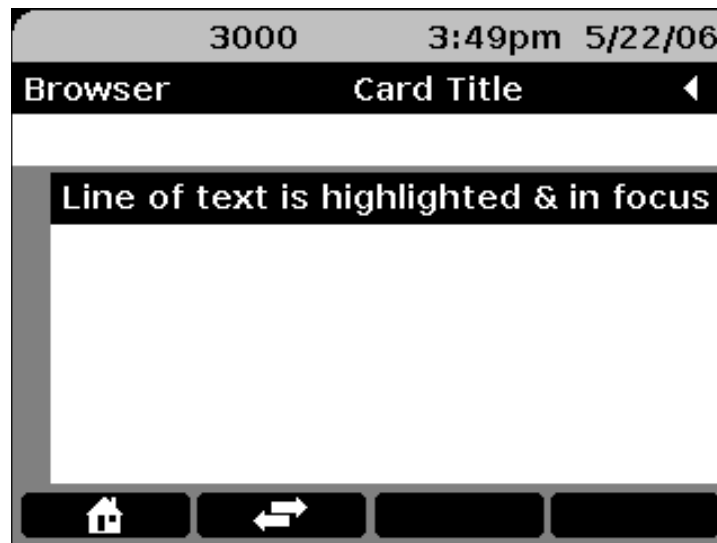


Web screen navigation works as follows:

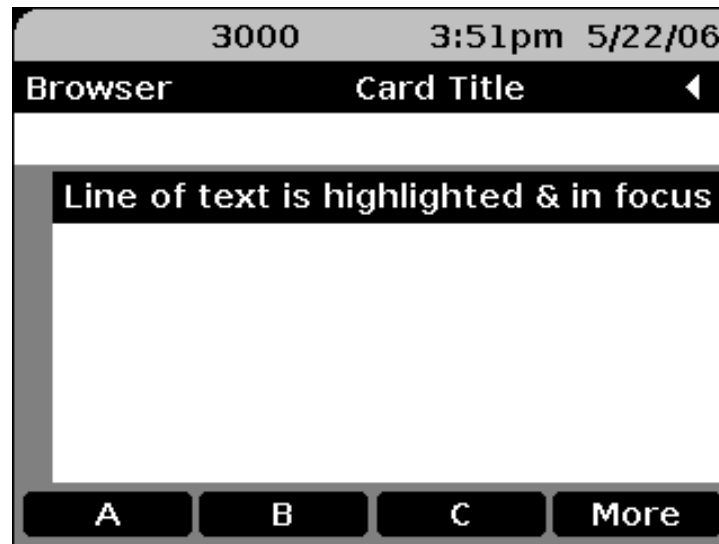
- One link is permitted per line. The link can be activated in one of two ways - pressing the right Line button, if available and if associated with a particular display line, or pressing the **OK** button if the line is in focus or highlighted.
- Line buttons, if available, or the **OK** button in the middle of the navigation cluster are used to select an option.
- Line buttons, if available, or the **OK** button in the middle of the navigation cluster are used to focus on a text entry box to allow entry of text.
- The WML card author can use the softkeys on the bottom of display for navigation as appropriate. The default softkeys labels are **Home**, **Refresh**, and **Stop**. The default softkey labels are always presented and are represented by icons. The **More** softkey is assigned to the fourth key position, 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). Default softkeys will be displayed when a page error is displayed. [Figure 21](#) shows the default softkeys. [Figure 22](#) shows how author-defined softkeys take precedence over the default softkeys.

---

**Figure 21: Default Softkeys**



---

**Figure 22: 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 always displays as a default softkey. The result of pressing **Stop** depends on the state of the display at that time that softkey is pressed, as follows:
    - If a new page is loading, the user receives an error message and the preceding page displays.
    - If a new page is not loading and no WML page has been received, nothing happens. The screen does not change, no error message displays, no error tone sounds.
    - If a new page is not loading, but a portion of the WML page has been received, the screen is updated with whatever information has been received.
  - Avaya recommends that the page author not include softkeys that are redundant with the default softkeys.

---

## Multiple Paging Indicators

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.

---

## 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 six lines 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 six lines 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.

If the **Right** or **Left** navigation button is pressed and held, 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.

The user moves up and down the lines currently displayed because different title information shows on the Prompt Line depending on which of the currently displayed lines is brought into focus. 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](#) on page 82 provides Top Line default titles. [Chapter 7: Web Browser for 9620 and 9630 IP Telephones](#) and [Chapter 9: Advanced Features](#) 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.

**Example:**

[Table 8](#) shows the relationship between moving up and down one line at a time on the current lines displayed, what appears in the Prompt Line, and what line is considered in focus. This example shows what happens as the user moves down the card one line at a time. The table shows the title displayed in the Title or Prompt Line as appropriate when its corresponding tag is in focus. In this table, for example, when line 1 is in focus, its corresponding title displays on the Top Line. When the tag associated with a line does not have a specific title, a default title displays. If the line is only text and has no actionable tags, the card title displays.

**Table 8: Relationship between Line/Page Movement, Focus, and Top Line Display**

<b>Title or Prompt Line displays:</b>	<b>Card Title on Title Line</b>	<b>Line 1 Tag Title on Prompt Line</b>	<b>Line 2 Tag Title on Prompt Line</b>	<b>Line 3 Tag Title on Prompt Line</b>	<b>Line 4 Tag Title on Prompt Line</b>
Line that is in focus	1	1	1	1	1
	2	2	2	2	2
	3	3	3	3	3
	4	4	4	4	4
	5	5	5	5	5
	6	6	6	6	6
Action					
<b>Prompt Line displays:</b>	<b>Line 5 Tag Title on Prompt Line</b>	<b>Line 6 Tag Title on Prompt Line</b>	<b>Line 7 Tag Title on Prompt Line</b>	<b>Line 8 Tag Title on Prompt Line</b>	<b>Line 9 Tag Title on Prompt Line</b>
Line that is in focus	1	1	2	3	4
	2	2	3	4	5
	3	3	4	5	6
	4	4	5	6	7
	5	5	6	7	8
	6	6	7	8	9
Action					

## Tags with Titles and Corresponding Title/Prompt Line Defaults:

Tag	Attribute	Title/Prompt Line Default Values	Appears on
<card>	Title	New Card	Title Line
<anchor>	Title	Use button to select	Prompt Line
<a>	Title	Use button to select	Prompt Line
<input>	Title	Enter text between brackets	Prompt Line
<select>	Title	Use button to select	Prompt Line
<img>	Alt	Image Not Rendered	Prompt Line

### Note:

The <img> default text displays when an image cannot be loaded. The default text does not display when an image is in focus.

The Title Line 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 <br/> 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](#) on page 106 and the corresponding Title change sample in that section. Horizontal scrolling is not supported.

### Note:

Phone calls can be made or received while in the Web browser.

---

## 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 = wrap. The default is to wrap the text.
- 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 frame width and <p> mode is nowrap, the link is truncated at the point where it no longer fits in the frame. Links are not wrapped around to the next display line. If <p> mode is set to wrap, the link continues to the adjoining line if labels are long.

- Long words that do not fit on an entire line should be hyphenated.
- URLs cannot be broken into multiple lines.

---

## Links

Rules for links within a page are simplified for development purposes:

- 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.

---

## Enabling Text Entry

Use these guidelines to enable text entry. See [Access Key Input Mode \(AIM\)](#) in [Chapter 7: Web Browser for 9620 and 9630 IP Telephones](#) for information about an alternate text entry method using access keys.

1. When a card displays, a set of square brackets containing the prompt “**Enter text here**” indicates a text entry area.
2. To start text entry, 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.
3. Text editing softkeys display at the bottom of the screen to assist the user with text entry.
4. Text (left justified) is entered between the brackets. The right bracket moves to the right edge of the screen and the left bracket remains in the same position. After exiting text entry, the new text is enclosed in brackets. The brackets cannot be deleted.
5. 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,
  - Pressing any other Line button,
  - Pressing a **Done** softkey at the bottom of the screen, or
  - Pressing the **OK** button.
6. 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.

7. A page author can customize bracketed 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.”
8. 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 is sent to 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: Web Browser for 9620 and 9630 IP Telephones](#) and [Chapter 9: Advanced Features](#) for specific information on the `<input>`, `<title>`, `<value>`, and `<ivalue>` tags.

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

For the tag `<input title="Hello there"/>` - The text box displays **[Enter text here]** as a default. No value is present, meaning the user has not entered text in the text box. Upon text entry, the value equals the typed-in text, and the “Enter Text here” prompt no longer displays.

For the tag `<input title="Hello there" value="Fill in the blanks"/>` - The text box displays **[Fill in the blanks]**. The cursor is positioned after the “s” in “blanks.” In this example, “Fill in the blanks” always appears at the beginning of the text box, regardless of what the user enters. This is contrary to the situation above where no value was specified in the `<input title>` string and the prompt [Enter text here] no longer displays after the user types the first character.

If a new attribute value is defined, the attribute takes precedence over `ivalue`. For the tag `<input title="Hello there" ivalue="Enter name here"/>` - the text box displays **[Enter name here]** and the attribute value takes precedence over `ivalue`. When the user enters text, the value now equals the typed-in text. The prompt “Enter name here” no longer displays and the text the user entered remains in the text box.

## Text Editing Modes

When a text entry area is enabled, these softkey labels display centered above the actual softkeys:

<b>Bksp</b>	<b>Done</b>	<b>Cancel</b>	<b>More</b>
-------------	-------------	---------------	-------------

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:

<b>Clear</b>	<b>Symbol</b>	<b>ABC</b>	<b>More</b>
--------------	---------------	------------	-------------

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.

---

## 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.

---

## Web History

The Web history is a list of URLs and starts from the point the user initiates the browser session for the first time. 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.

---

## Display Colors

Any line that is considered in focus displays in the reverse colors for four grayscale used for foreground and background. JPEG color images will be converted to four grayscale JPEG images.

Web authors can specify different color text and background for telephones that support color. In Release 1, neither the 9620 nor the 9630 support color images. Web authors have two options:

- use default colors on the Web page, or
- use the CSS properties described in [Advanced Features](#) to make full use of the gray tones for text and to allow inverse text and highlighting.

**Note:**

See [Chapter 9: Advanced Features](#) for specific information regarding color.

---

## Call Interaction

The user can make and receive calls when in a Web browser session.

When users re-enter the Web browser application, they are returned to the same point at which they left. If the user is entering text and leaves the Web browser using the **Phone** button, the user returns to the same state of text entry upon reentering the Web application.

No dialing is supported when in text entry mode. On-hook keypad dialing to make a phone call is not allowed when the user is entering text. Text entry is allowed while the user is off-hook. When the phone is placed once again on-hook, text entry can be re-enabled at the point where it was disabled.

When text editing is active, the user can be active on a call or receive a call, but cannot use the keypad to dial a phone number or for interactive voice recognition (IVR) prompts. The user must disable text entry prior to using keypad dialing to make a call or for IVR prompts.

---

## 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, fieldset
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.

The attributes `xml:lang`, `class`, and `id` are universal attributes associated with every wml element. The Web browser supports these tags as follows:

Attribute	Support	Comments
<code>xml:lang</code>	No - will not be used	Currently only English is supported.
<code>class</code>	No - will not be used	Only half of the phone browsers support this attribute.
<code>id</code>	Yes - will be used	

The three universal tags are not repeated in requirements for other tags. The [Summary Of WML Tags and Attributes](#) on page 101 is a complete list of all supported tags and attributes.

---

## 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 can include:

- Originating a call
- Adding, searching, and removing Contacts entries
- Sending and receiving Short Text messages
- Examining call logs, including calls made, received, or missed
- Sending DTMF tones during an active voice call
- Pushing content to the Prompt Line

The Web browser supports only those WTA functions that are available from WML using URIs.

The Web browser supports the WTAI application “click to dial” any link on the screen. 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>`, `<img>`, `<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.

The WTAI URI scheme is as follows:

**wtai://<library>/<function> (; <parameter>)\* [! <result>]**

Scheme Definition:	
< >	Indicates an enumerated operator.
[ ]	Indicates an optional section.
	Indicates a pair of mutually exclusive options.
( )*	Repeat none or multiple items.
* ( )	Repeat one or multiple items.
library	Name that identifies the library type, WTA Public uses library “wp.”
function	Function within a library, for example, “mc” for function “make call” in “wp” library.
parameter	Zero or more parameters sent to a function. Delineated by a semicolon “;”.
result	Start of result defined by “!”. Optional.

---

## 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.

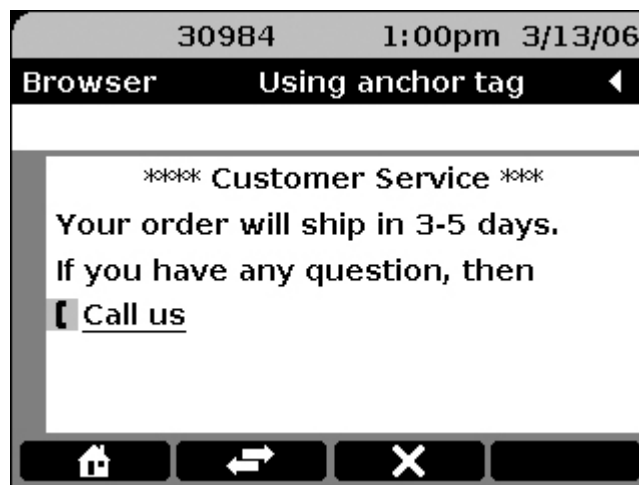
**Click-to-dial using <anchor> tag:**

```

<?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:

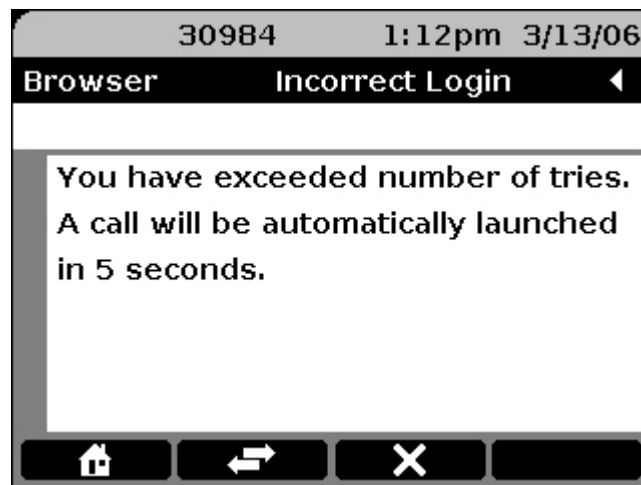


The code shows a hyperlink as **Call Us** on the Web page. When a user selects this link, the phone dials the string "5551212" or any number that follows a semicolon in the WTAI code.

### 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 *Avaya one-X™ Deskphone Edition for 9600 Series IP Telephone Administrator Guide* for configuration information.

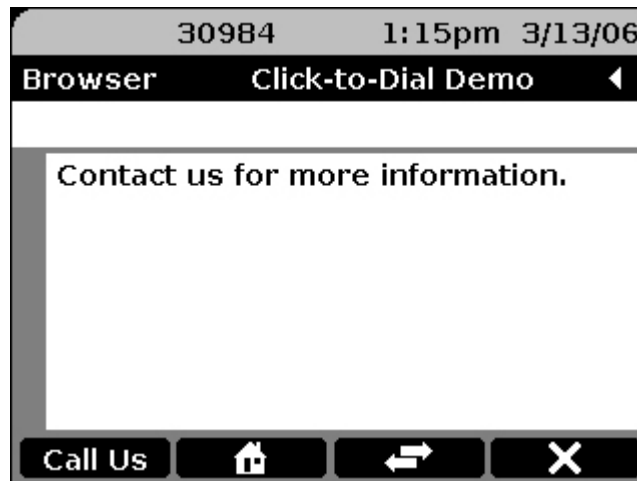
**Click-to-dial using <do> tag (softkey):**

```

<?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:

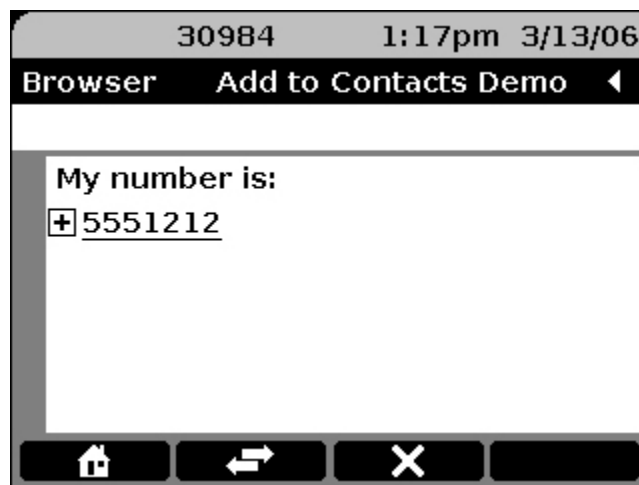
***wtai://wp/ap;number;name***

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.

---

## Support for 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.

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.

The authentication name and password and the realm are stored in reprogrammable non-volatile memory that is not overwritten if new telephone software is downloaded. The default value of the credentials and the realm are null, set at manufacture and at any other time that user-specific data is removed from the telephone.

Standard text entry and text entry for passwords is supported.

The authentication screen is as follows:

---

## Page Loading

When a page is loading, the Title Line displays `Loading`. Any additional button presses except the **Stop** softkey are not honored and the user receives an error tone. If a user tries to reach another link before a page loads, the page continues to load and the user cannot go to another link until the page is loaded.

---

## Error Tones

Error tones are consistent with other applications. When a labeled button is pressed but the button function is not specified for the current browser state, an error tone sounds. Presses of unlabeled Line buttons or softkeys are ignored. When users press a Line button or softkey inappropriately, an error tone sounds.

Text areas that can be highlighted receive no error beeps when selected.

---

## 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.

## User-Agent Header String (9620 IP Telephones)

Identify the 9620 WML Web browser using the following user-agent string:

AVAYA/SPICE/v1.0+(9620)/Std/0.1

Where:

**Avaya:** Company Name

**SPICE:** Product Family Name

**v.TBD:** Firmware version no

**9620:** Hardware indication

**Hardware Extension.** Can be used to differentiate between hardware platforms (For example, Broadcom vs. others)

**0.1 :** Minor Version Number

**Browser Type:** Std

**Browser Mode:**

**Accept-Charset**

## User-Agent Header String (9630 IP Telephones)

Identify the 9630 WML Web browser using the following user-agent string:

```
AVAYA/SPICE/v1.0+(9630)/Std/0.1
```

Where:

Avaya: Company Name

SPICE: Product Family Name

v.TBD: Firmware version no

9630: Hardware indication

Hardware Extension. Can be used to differentiate between hardware platforms (For example, Broadcom vs. others)

0.1 : Minor Version Number

Browser Type: Std

Browser Mode:

Accept-Charset

---

## Web/System Interaction

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

---

### Idle Timer

When the idle timer reaches the number of minutes equal to the WMLIDLETIME value, the telephone sends an HTTP GET for the WMLIDLEURI value. The Web application takes over the display and the Web page displays the timeout response.

---

## Web-Specific System Values

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

- **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).
- **WMLHOME** - URI of the Web application home page.
- **WMLPORT** - TCP port number used for the HTTP proxy server.
- **WMLPROXY** - IP Address, in dotted-decimal or DNS name format, of an HTTP proxy server.

If the system value **WMLHOME** is null, the telephone cannot display the Browser label under the “A” Avaya application.

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.

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.

---

## 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.

Table 9: 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	
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.

Table 9: HTTP Error Messages (continued)

Status Code	Failure Message	Meaning
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.

**2 of 2****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

Table 10: 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
<b>		No
<big>		No
 		Yes

**1 of 4**

Table 10: Summary Of WML Tags And Attributes (continued)

Tag	Attribute	Supported?
<card>		Yes
	newcontext	Yes
	onenterbackward	Yes
	onenterforward	Yes
	ontimer	Yes
	ordered	No
	style	Yes
	title	Yes
<do>		Yes
	label	Yes
	name	Yes
	optional	Yes
	type	Yes
<em>		No
<fieldset>		No
<go>		Yes
	accept-charset	Yes
	href	Yes
	method	Yes
	sendreferer	Yes
<head>		No
<i>		No
<img>		Yes
	align	Only left alignment supported
	alt	Yes
	height	No
	hspace	Yes
	localsrc	No
	src	Yes
	style	Yes
	vspace	Yes
	width	No
	style	Yes

2 of 4

Table 10: Summary Of WML Tags And Attributes (continued)

Tag	Attribute	Supported?
<input>		Yes
	accesskey	No
	ivalue	Yes
	emptyok	Yes
	format	Yes
	maxlength	Yes
	name	Yes
	size	No
	style	Yes
	tabindex	No
	title	Yes
	type	Yes
	value	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
<option>		Yes
	onpick	Yes
	style	Yes
	title	Yes
	value	Yes
<p>		Yes
	align	Yes
	mode	Yes
	style	Yes
		<b>3 of 4</b>

**Table 10: Summary Of WML Tags And Attributes (continued)**

Tag	Attribute	Supported?
<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
<strong>		No
<table>		No
<td>		No
<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

**4 of 4**

# Chapter 7: Web Browser for 9620 and 9630 IP Telephones

---

## Introduction

This chapter:

- Presents the WML portions implemented in the 9620 and 9630 IP Telephone Web Access applications. Any limitations or non-standard implementations are mentioned.
- Provides considerations to develop effective Web pages for browser viewing.

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

---

## General Background

The data types and other features supported in this browser include:

- WML 1.3, June 2002
- HTTP 1.1

The [Summary Of WML Tags and Attributes](#) on page 101 summarizes the detailed tag-related information provided in each section in this chapter.

**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.

---

## 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.

- Common tag attributes: `xml:lang`, `class`, and `id`.

The attributes `xml:lang`, `class` and `id` are universal attributes associated with each WML element.

The Web browser supports these tags:

Attribute	Comments
<code>xml:lang</code>	NOT SUPPORTED
<code>class</code>	NOT SUPPORTED
<code>id</code>	SUPPORTED

- `<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.
- `<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.

Attribute	Value(s)	Description	Comments
<b>context</b>	<i>cdata</i>	Should specify the name attribute description.	SUPPORTED.
<b>name</b>	<i>keyword</i>	Name portion of the name content value.	SUPPORTED.
<b>forua</b>	True or False	Specifies if meta data is sent to the browser. If "True" meta data must be sent to the browser	NOT SUPPORTED.
<b>http-equiv</b>	<i>cdata</i>	Sets whether the <meta> tag content is bound to an http response header.	NOT SUPPORTED.
<b>scheme</b>	<i>cdata</i>	Specifies the structure used to translate the meta data.	NOT SUPPORTED.

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. Unsupported attributes are indicated as such in the Comments column.

Attribute	Value(s)	Description	Comments
<b>newcontext</b>	true	Re-initializes the browser context.	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. SUPPORTED. <b>Note:</b> Page authors must include this attribute where it is necessary to clear the history stack.
	false	Default is "false."	
<b>ordered</b>	true	Specifies the order of card content. When ordered is set to "true" the browser displays the content in a fixed order. When ordered is set to "false" the users decide the order as they navigate between content. Default is "true."	Optional. Sets a Boolean value that provides information on how the content of the current card is arranged. Used by the browser to organize the display presentation and layout. If set to true, content is organized in a linear sequence of elements - for example, a series of ordered or non-optional input elements. If set to false, content is in no natural order - for example, a series of unordered or optional input elements. The default is true. NOT SUPPORTED.
	false		
<b>title</b>	<i>cdata</i>	The title of the card.	Can be used for title displays. SUPPORTED.
<b>onenterbackward</b>	<i>url</i>	Occurs when the user navigates into a card by means of a "prev" task.	SUPPORTED.
<b>onenterforward</b>	<i>url</i>	Occurs when the user navigates into a card by means of a "go" task.	SUPPORTED.
<b>ontimer</b>	<i>url</i>	Occurs when a "timer" expires.	SUPPORTED.
<b>style</b>	<i>property</i>	Cascading Style Sheet attribute.	SUPPORTED.

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 the Web browser supports are:

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

**Note:**

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

- `<access>` - The `<access>` tag limits access within the deck to certain cards. This tag is not supported.

## Text Elements

See [Enabling Text Entry](#) on page 83 and [Text Editing Modes](#) on page 85 for guidelines to enable text entry and facilitate text editing. See [Access Key Input Mode \(AIM\)](#) on page 125 for information about an alternate text entry method using access keys.

- `<br/>` tag - The `<br/>` 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
<code>align</code>	left right center	Aligns the paragraph. Default is “left.”	SUPPORTED.
1 of 2			

Attribute	Value	Description	Comments
<b>mode</b>	wrap nowrap	Sets whether a paragraph wraps lines or not.	SUPPORTED.
<b>style</b>	<i>property</i>	Cascading Style Sheet Attribute.	SUPPORTED
2 of 2			

If	align=left	align=center	align=right
<b>mode=wrap</b> (default)	Yes	No	No
<b>mode=nowrap</b>	Yes	Yes	Yes

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 <img> the “mode” attribute will be set to “nowrap” if the “align” attribute is either “center” or “right”. For <img> 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.

The following tags are not supported but the content inside the tags is rendered as normal text:

- <table> tag - The <table> tag specifies a table. This tag is not supported.
- <td> tag - The <td> tag defines individual cell contents in each row of a defined table. This tag is not supported.
- <tr> tag - The <tr> tag defines each row of a defined table. This tag is not supported.

---

## Text Formatting Tags

The following tags are not supported but the content inside the tags is rendered as normal text:

- <b> tag - The <b> tag specifies bold text. This tag is not supported.
- <big> tag - The <big> tag specifies large font text. This tag is not supported.
- <em> tag - The <em> tag specifies emphasized text. This tag is not supported.
- <i> tag - The <i> tag specifies italicized text. This tag is not supported.

- `<small>` tag - The `<small>` tag specifies small font size text. This tag is not supported.
- `<strong>` tag - The `<strong>` tag specifies strongly emphasized text. This tag is not supported.
- `<u>` tag - The `<u>` tag specifies underlined text. This tag is not supported.

---

## 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. The user selects the link by pressing one of the Line buttons associated with that display line.

Attribute	Value	Description	Comments
<b>href</b>	<i>url</i>	REQUIRED. Defines where to go when the user selects the link.	SUPPORTED.
<b>title</b>	<i>cdata</i>	Defines a text identifying the link.	SUPPORTED.
<b>accesskey</b>	1,2,3,4,5,6,7,8,9,0,*,#	A keypad key the user can press as a shortcut to selecting a link.	SUPPORTED.
<b>style</b>	<i>property</i>	Cascading Style Sheet Attribute	SUPPORTED.

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
<b>title</b>	<i>cdata</i>	Defines a text identifying the link.	SUPPORTED.
<b>accesskey</b>	1,2,3,4,5,6,7,8,9,0,*,#	A keypad key the user can press as a shortcut to select a link.	SUPPORTED.
<b>style</b>	<i>property</i>	Cascading Style Sheet Attribute	SUPPORTED

## Image Elements

- **<img> tag** - Use the **<img>** 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 **<img>** element can be contained in these elements: a, anchor, b, big, em, fieldset, i, p, small, strong, td, and u.

Attribute	Value	Description	Comments
<b>align</b>	top middle bottom  left right center	Aligns the image.	Left is the only value currently SUPPORTED.
<b>alt</b>	<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.	SUPPORTED. When an alt tag is not associated with an image, the Prompt Line should be empty.
<b>height</b>	<i>px (pixel)</i> <i>%</i>	Sets the height of the image.	NOT SUPPORTED. The true height parameters are determined by parsing the WBMP/JPEG information.
<b>hspace</b>	<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.	SUPPORTED. The default is 0 pixel.
<b>localsrc</b>	<i>cdata</i>	Sets an alternate representation for the image.	NOT SUPPORTED.
<b>src</b>	<i>url</i>	REQUIRED. The path to the image. Must be either a .wbmp file or a jpeg file.	SUPPORTED.
<b>vspace</b>	<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	SUPPORTED. The default is 0 pixel.
			1 of 2

Attribute	Value	Description	Comments
<b>width</b>	<i>px</i> <i>%</i>	Sets the width of the image.	NOT SUPPORTED. The true width parameters are determined by parsing the WBMP/JPEG information.
<b>style</b>	<i>property</i>	Cascading style sheet attribute.	SUPPORTED.
2 of 2			

## 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
<b>type</b>	accept prev help reset options delete unknown x- vnd.*	REQUIRED. Defines the type of the “do” element.	SUPPORTED.
<b>label</b>	<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. SUPPORTED.
<b>name</b>	<i>mmtoken</i>	Defines a name for the “do” element.	SUPPORTED.
<b>optional</b>	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. SUPPORTED.

Type	Description	Comments
accept	Acknowledgement of acceptance.	SUPPORTED.
delete	Delete item.	SUPPORTED.
help	Request for help.	SUPPORTED.
options	Options or additional operations.	SUPPORTED.
prev	Backward navigation.	SUPPORTED.
reset	Clearing or reset.	SUPPORTED.
X-*n or x-*n	Experimental.	NOT SUPPORTED.
Vnd*	Any mix of upper or lower cases. Vendor-specific.	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 no labels are given, then the “do” types have the following default labels:

Type	Default Label if no label specified
accept	Accept
delete	Delete
help	Help
options	Options
prev	Back
reset	Refresh
X-*n or x-*n	Unknown
Vnd* Any mix of upper or lower cases	AVAYA. Available for future use but currently Unknown.

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

Home	Refresh	Stop	
------	---------	------	--

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

1st DO	Refresh	Stop	Home
--------	---------	------	------

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

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

Page 1 softkeys:

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

Page 2 softkeys:

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

**Note:**

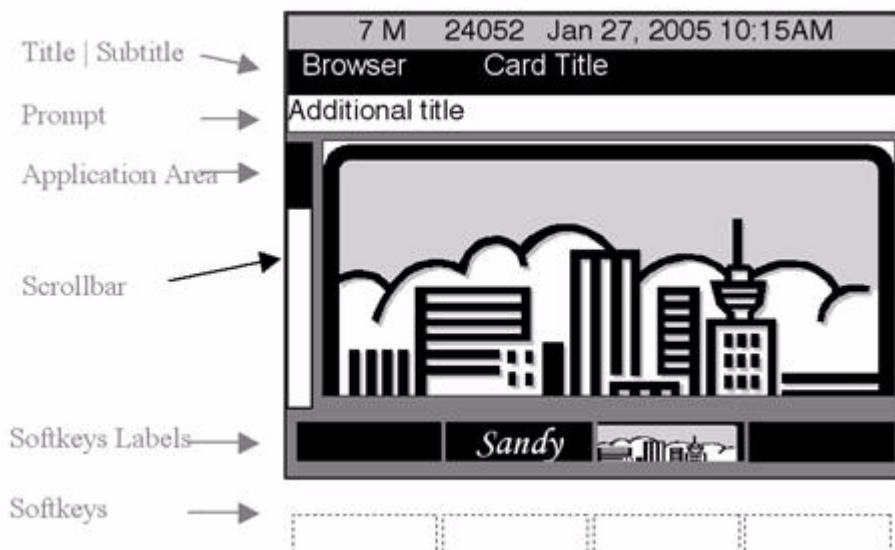
If more than one page of softkey labels are specified, pressing the **MORE** softkey automatically presents the user with the next page of labels. If the last page displays and the user presses the **MORE** softkey, the first page of labels displays. As implied in the examples, the Softkey buttons are 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.

To render JPEG or WBMP images in the softkey label area, embed <img> 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. SUPPORTED.
onenterforward	card or template	Occurs when a <go> navigates into a card. SUPPORTED.
onpick	option	Occurs when a user selects/ deselects an item. SUPPORTED.
ontimer	card or template	Occurs when the time expires. SUPPORTED.

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
<b>type</b>	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.	SUPPORTED.

- <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
<b>name</b>	<i>cdata</i>	REQUIRED. The name of the field.	SUPPORTED.
<b>value</b>	<i>cdata</i>	REQUIRED. The value of the field.	SUPPORTED.

## 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
<b>href</b>	<i>url</i>	REQUIRED.	SUPPORTED.
<b>accept-charset</b>	Charset_list	A comma- or space-separated list of character encoding the server must be able to process. The default value is "unknown."	SUPPORTED.
<b>method</b>	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 <b>value</b> attribute in an <input>, <select>, or <option> tag that is part of the current WML page.	SUPPORTED.
<b>sendreferer</b>	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."	SUPPORTED.

The accept-charset header is: **accept-charset: US-ASCII, ISO-8859-1, UTF-8;q=0.1**

- **<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. Xml:lang is not an associated attribute. User-visible side effects of the update can occur during the **<refresh>** processing.

---

## Input Elements

- **<input>** tag supported - The input tag specifies a point where the user is prompted to enter text.

Attribute	Value	Description	Comments
<b>accesskey</b>	1,2,3,4,5,6,7,8,9,0,*,#	A Dialpad button used to access a link containing selections. If the element is a Radio button, pressing the access key is a shortcut for selecting the Radio button. If the element is a check box, pressing the access key is a shortcut to check or uncheck the box. If the element is a Submit or Reset button, pressing the access key is a shortcut for pressing that button. For example, pressing a Submit button's access key submits the applicable form.	NOT SUPPORTED.
<b>name</b>	<i>nmtoken</i>	REQUIRED. The name of the variable that is set with the result of the user's input.	SUPPORTED.
<b>emptyok</b>	true false	Sets whether the user can leave the input field blank or not. Default is “true.”	SUPPORTED.
<b>1 of 2</b>			

Attribute	Value	Description	Comments
<b>format</b>		Sets the data format for the input field. Default is "M."	SUPPORTED.
	A	A = uppercase alphabetic or punctuation characters	
	a	a = lowercase alphabetic or punctuation characters	
	N	N = numeric characters	
	X	X = uppercase characters	
	x	x = lowercase 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.	
<b>ivalue</b>		The attribute <b>value</b> takes precedence over <b>ivalue</b> .	SUPPORTED.
<b>maxlength</b>	<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.	SUPPORTED.
<b>size</b>	<i>number_of_char</i>	Sets the width of the input field.	NOT SUPPORTED.
<b>tabindex</b>	<i>number</i>	Sets the tabbing position for the input field.	NOT SUPPORTED.
<b>title</b>	<i>cdata</i>	Sets a title for the input field.	SUPPORTED.
<b>type</b>	text password	Indicates the type of the input field. The default value is "text."	SUPPORTED.
<b>value</b>	<i>cdata</i>	Sets the default value of the variable in the <b>name</b> attribute.	SUPPORTED.
<b>style</b>	<i>property</i>	Cascading style sheet attribute.	SUPPORTED.

2 of 2

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** attribute is null. If the author specifies a non-null content in the **value** attribute, that content displays between brackets for that input tag.

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.

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

- **<fieldset>** tag - The fieldset tag is used to group logically related elements in a card. This tag is not supported.
- **<optgroup>** tag - Sets of **<optgroup>** brackets can be put around **<options>** in a **<select>** list. The results break a list into sublists.

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

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

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
<b>name</b>	<i>nmtoken</i>	REQUIRED. String that names the variable to which the selection results are assigned.	SUPPORTED.
<b>ivalue</b>	<i>cdata</i>	Sets the pre-selected option element. If none is specified, the first item in a list is automatically selected.	SUPPORTED.
<b>multiple</b>	true false	Sets whether multiple items can be selected. Default is "false." False is used for a single selection.	SUPPORTED.
<b>tabindex</b>	<i>number</i>	Sets the tabbing position for the select element.	NOT SUPPORTED.
<b>title</b>	<i>cdata</i>	Sets a title for the list.	SUPPORTED.
<b>value</b>	<i>cdata</i>	Sets the default value of the variable in the <b>name</b> attribute.	SUPPORTED.
<b>style</b>	<i>property</i>	Cascading style sheet attribute.	SUPPORTED.
<b>iname</b>	<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.	SUPPORTED.

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
<b>name</b>	<i>cdata</i>	REQUIRED. Sets the name of the variable.	SUPPORTED.
<b>value</b>	<i>cdata</i>	REQUIRED. Sets the value of the variable.	SUPPORTED.

- <timer> tag - The timer tag sets a timer that starts counting. This tag must be used with <onevent type="ontimer"> to be useful.

Attribute	Value	Description	Comments
<b>value</b>	<i>cdata</i>	REQUIRED. Sets the default value of the variable defined in the <b>name</b> attribute.	SUPPORTED.
<b>name</b>	<i>nmtoken</i>	Names the variable that is set with the value of the timer.	SUPPORTED.

---

## 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 `&#060;`.

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 `&lt;`. 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	"	&#34;	&quot;
ampersand	&	&#38;	&amp;
apostrophe	'	&#39;	&apos;
less than	<	&#60;	&lt;

---

## Colors and Fonts

The browser supports a four grayscale display. 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 11](#) each of the Dialpad button access keys is mapped to a different URL. For example, when the “2” button on the dialpad is pressed, the example2.com URL is launched because it is mapped to the “2ABC” button, as shown in the following code example:

```
<a href="http://example2.com/contents/wml" accesskey="2">
```

Similarly, to inform the Web server that any of the Keypad buttons were pressed, the [Table 11](#) example URLs would be launched for any of the 12-keypad buttons. Each button is assigned to a different access key and URL. Each button to be tracked must have a URL mapped to it in the page. If the page author wants to know whether any of the 12 possible Dialpad buttons are pressed, then the page must include access keys pointing to specific URLs.

Table 11: Dialpad/URL Mapping Example

Dialpad Button	Example
1	<a href="http://URL1" accesskey="1">1</a>
2ABC	<a href="http://URL2" accesskey="2">2</a>
3DEF	<a href="http://URL3" accesskey="3">3</a>
4GHI	<a href="http://URL4" accesskey="4">4</a>
5JKL	<a href="http://URL5" accesskey="5">5</a>
6MNO	<a href="http://URL6" accesskey="6">6</a>
7PQRS	<a href="http://URL7" accesskey="7">7</a>
8TUV	<a href="http://URL8" accesskey="8">8</a>
9WXYZ	<a href="http://URL9" accesskey="9">9</a>
0	<a href="http://URL10" accesskey="0">0</a>
*	<a href="http://URL11" accesskey="*">*</a>
#	<a href="http://URL12" accesskey="#">#</a>

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. The correct syntax is:

WML tag <a> or <anchor>	href="URL"	accesskey=dialpad button
-------------------------	------------	--------------------------

**Note:**  
The <input> tag does not support the accesskey attribute because AIM and standard Text Entry cannot be mixed on one page.

## Example of Text Entry Using AIM:

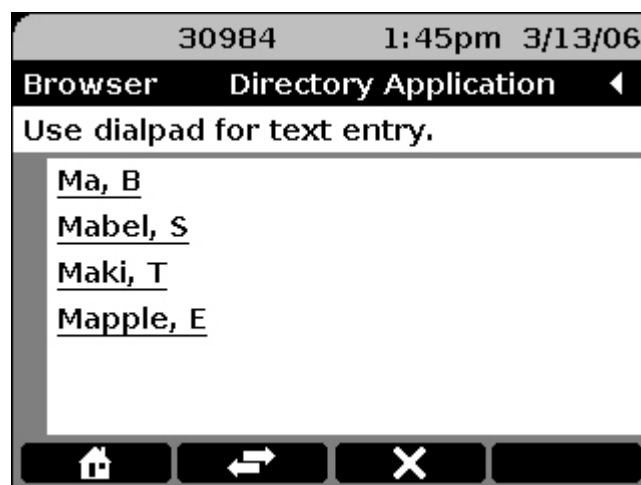
The user wants to look up a person named “Oscar” in a Directory database. The user presses a Directory softkey on the Home page that initiates the AIM application. [Figure 23](#) shows a sample starting page. The user presses the **6MNO** Dialpad button because it contains the first letter of the name 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. Figures 23 through 26 illustrate locating the name “Oscar” in the directory.

**Figure 23: Starting Page**

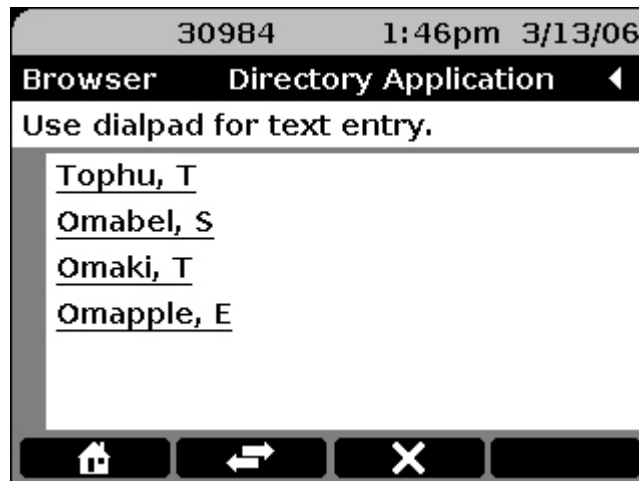


**Figure 24: User presses “6” one time for “O” and the closest match to “O” displays.**



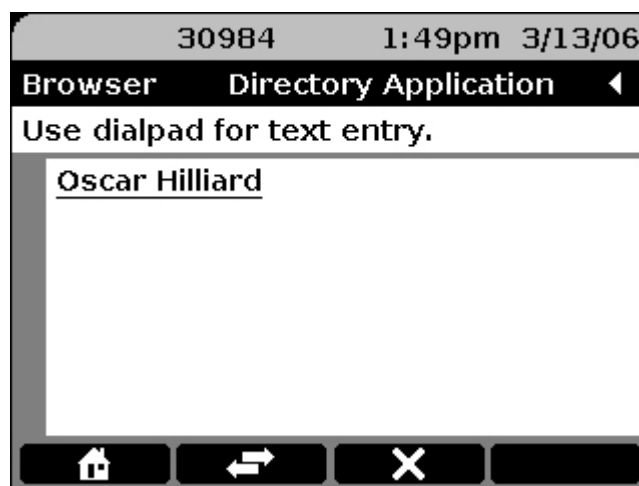
---

Figure 25: User presses “7” one time for “S,” narrowing the search.



---

Figure 26: User presses “2” one time for “C” and the search result displays the desired result - Oscar Hilliard.



---

## AIM Considerations

AIM mode cannot be enabled when the browser is not in focus.

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 launches 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 the page loads (attribute=accesskey) and can begin pressing Dialpad buttons to send button press information. As an example, you may want to have a link from your Home page to the AIM application or wherever else such a link makes sense.

When a user clicks a button that has not been mapped to a URL and the browser is in AIM mode, nothing happens.

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://URL2" 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://URL2a" 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."

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 accesskeys.

# Chapter 8: Web Applications

---

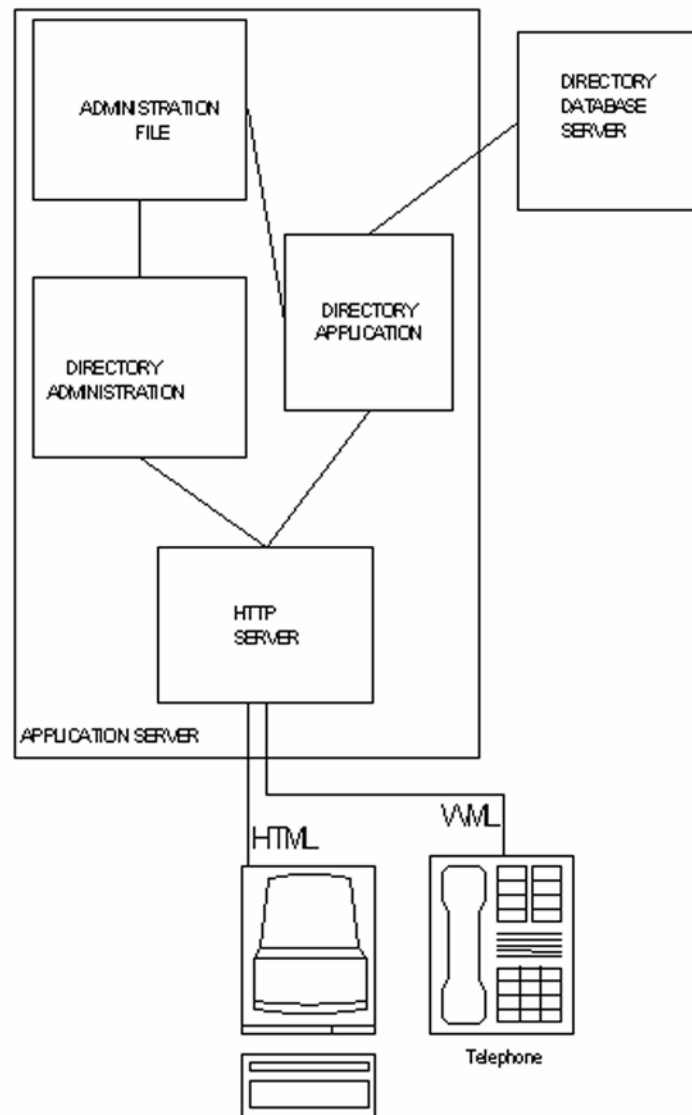
## Introduction

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 27: High-Level Thin Client Architecture](#) on page 132 provides a high-level overview of the Thin Client architecture.

**Figure 27: High-Level Thin Client Architecture**

As [Figure 27](#) 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> 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-1.0-1.0.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.

---

## 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-1.0-1.0.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-1.0-1.0.i386.rpm**.
4. To enable password control for the Directory Administration application, create a directory entry in the **httpd.conf** file as follows:

**Note:**

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-1.0.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:

#### Note:

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-1.0.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 [www.php.net](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 15: List of Drop-Down Attributes available for Search, Query and Details Administration Screens](#) on page 153. [Chapter 7: Web Browser for 9620 and 9630 IP Telephones](#) 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

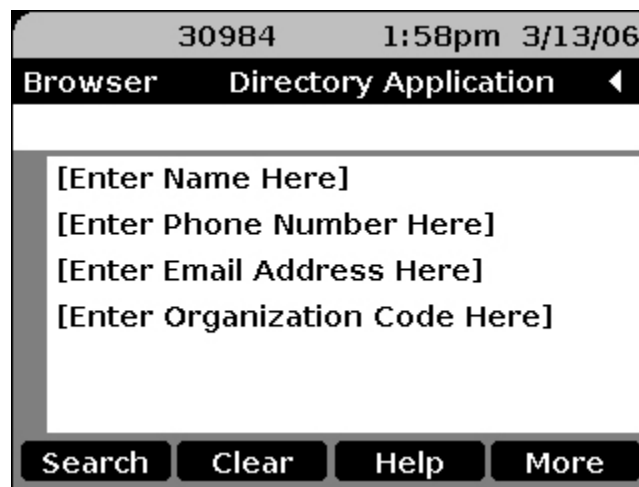
The Search screen displays upon user selection of the Directory application. At a minimum, administer these two user text entry fields:

- Enter Name Here
- Enter Phone Number Here

Either field provides basic search criteria. You can administer up to four additional text entry fields.

---

**Figure 28: Sample Search Screen**



The image shows a sample search screen interface. At the top, there is a status bar with the text "30984 1:58pm 3/13/06". Below this is a header bar with "Browser" and "Directory Application" followed by a left-pointing arrow. The main area contains four text entry fields with the following prompts: "[Enter Name Here]", "[Enter Phone Number Here]", "[Enter Email Address Here]", and "[Enter Organization Code Here]". At the bottom, there is a row of four softkeys: "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](#) on page 148.

---

## 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:

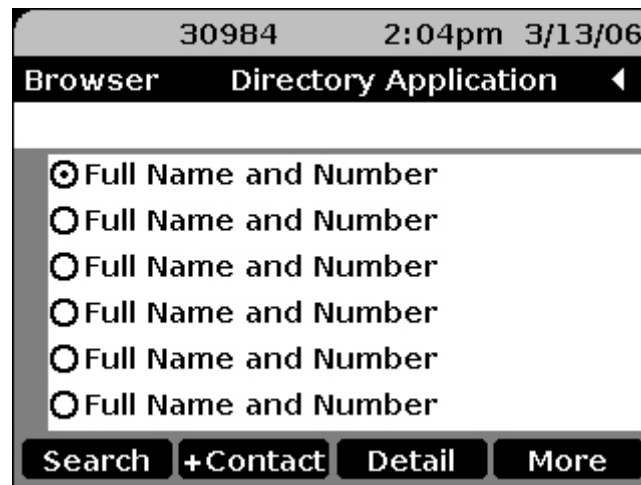
**X** found. Select choice. where **X** = the number of matches found, or

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 29: 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 30](#) for a sample Detail screen.
- **More** - Displays additional softkeys.

---

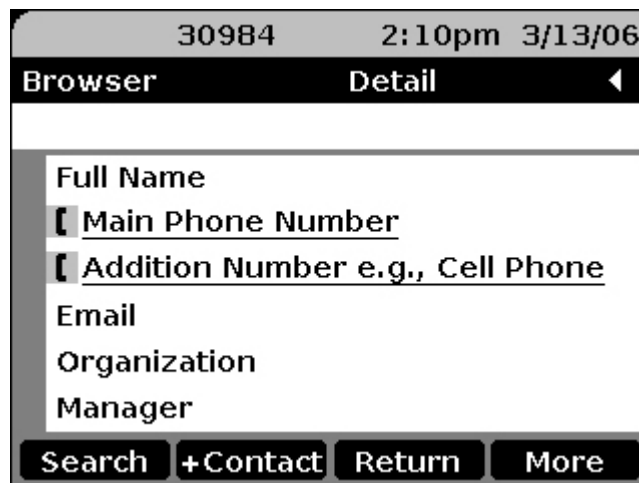
## Web Application Detail Screen

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 30: 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](#) on page 150.

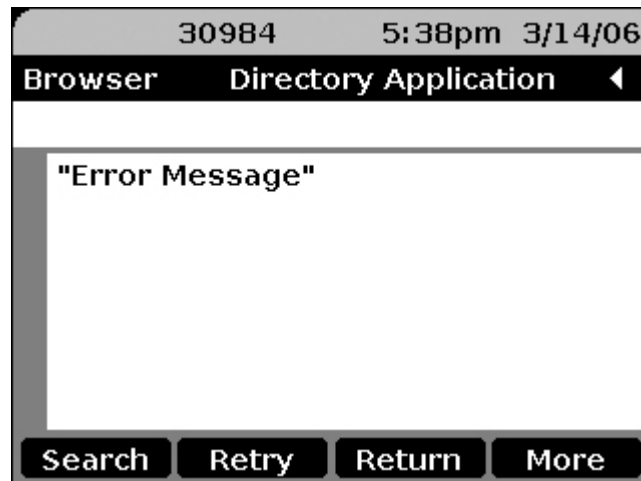
---

## 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 31](#).

---

**Figure 31: 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 12](#).

**Table 12: 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.
If any of these system values are null (except <b>DIRUSERID</b> and <b>DIRSRVRPWD</b> , 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.

1 of 2

**Table 12: Search Failure Causes and Corresponding Trouble Screen Error Messages (continued)**

<b>Cause of Search Failure</b>	<b>LDAP Result Code</b>	<b>Trouble Screen Message</b>
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.
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.

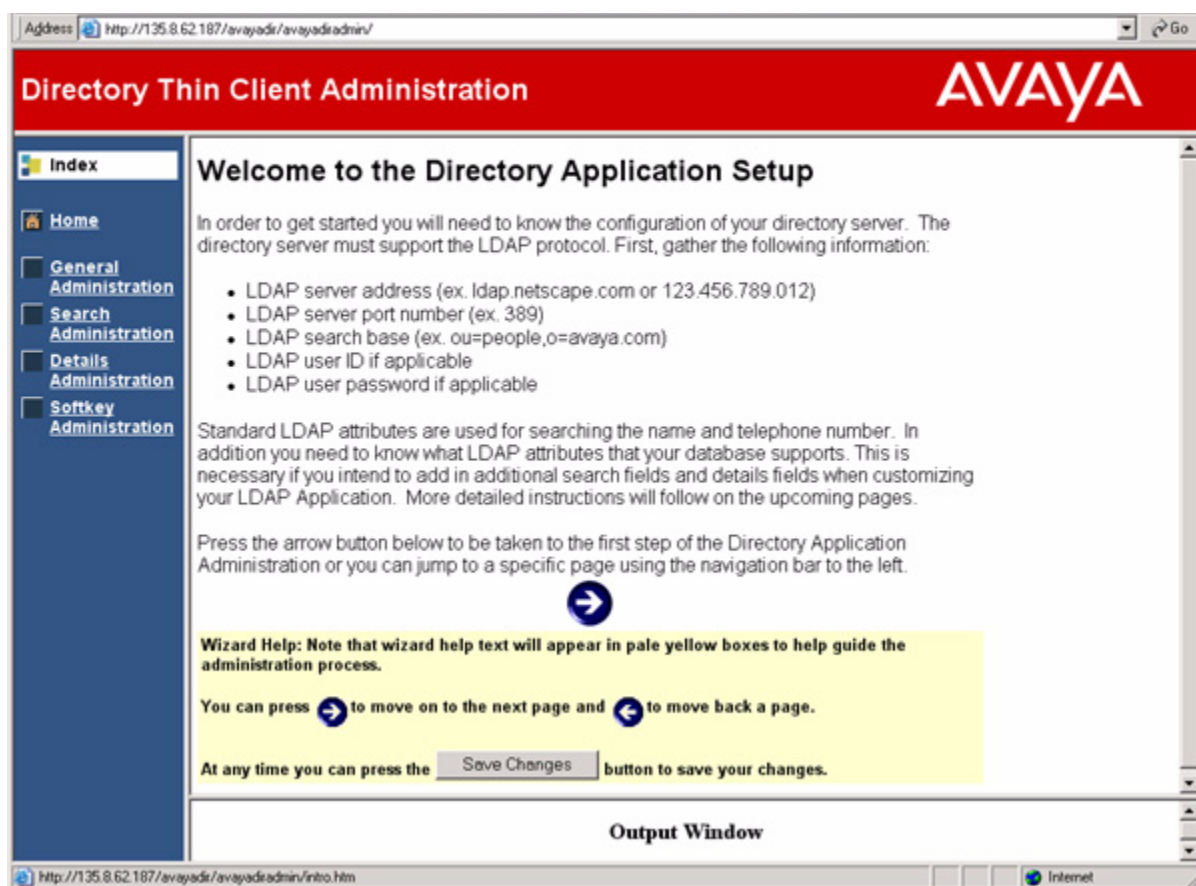
**2 of 2**

## 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 32: 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 33: Directory Application Administration Screen**

General Administration	
<b>Application Title:</b> 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.
<b>Directory Server:</b> 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.
<b>Topmost Distinguished Name (Search Root):</b> ou=people, o=yourcompany.com	Put the search root (ex. "ou=people, o=avaya.com") here.
<b>Port Network:</b> 389	The default LDAP server port is 389. The default is 636 for SSL enabled LDAP systems.
<b>Max number of hits:</b> 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.
<b>Directory User id (optional):</b>	If your LDAP directory requires a user name for searching then enter it here.
<b>Directory Password (optional):</b>	If your LDAP directory requires a password for searching then enter it here.
<b>Search Time:</b> 30 Seconds	Select the maximum search time that the application will wait for to receive results from the LDAP server.
<b>Coding:</b> 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 13](#) shows the Administration screen fields, their associated key names, default values, and descriptions:

**Table 13: 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.

3. 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 28: Sample Search Screen](#) on page 138 for an illustration of the user interface Search screen

1. 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 34: 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 ]

2. 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 14](#).

**Table 14: Search Administration Screen Fields**

Search Screen Line #	Search Field/ Search Object	LDAP Attribute Name	Associated (Customer-Defined) Label (20 characters maximum)
1	Name (fixed)	<b>cn</b>	Customer administrable.
2	Main Phone Number (fixed)	<b>phoneNumber</b>	Customer administrable.
3	E-mail (default)	<b>mail</b>	Customer administrable.
4	Customer administrable	<b>Customer administrable</b>	Customer administrable.
5	Customer administrable	<b>Customer administrable</b>	Customer administrable.
6	Customer administrable	<b>Customer administrable</b>	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 15: List of Drop-Down Attributes available for Search, Query and Details Administration Screens](#) on page 153 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 35: Details Administration Screen**

Address: http://135.8.62.187/avayads/avayadadmin/

**Directory Thin Client Administration** **AVAYA**

**Details Administration**

Line	Displayed Attribute	Label (Max 8 Characters)
1	Common LDAP Attributes: (cn) full name (multiple formats) Attribute: cn	
2	Common LDAP Attributes: (telephoneNumber) telephone number Attribute: telephoneNumber	
3	Common LDAP Attributes: (mobile) cellular telephone number Attribute: mobile	Cell Phone:
4	Common LDAP Attributes: (mail) e-mail address Attribute: mail	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](#) on page 140.

**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 15: List of Drop-Down Attributes available for Search, Query and Details Administration Screens](#) on page 153 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 15: List of Drop-Down Attributes available for Search, Query and Details Administration Screens](#) on page 153. 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 36: Softkey Administration Screen

Address <http://135.8.62.187/avaya/avayaadmin/> Go

## Directory Thin Client Administration

### AVAYA

[Index](#)  
[Home](#)  
[General Administration](#)  
[Search Administration](#)  
[Details Administration](#)  
[Softkey Administration](#)

### Softkey Administration

Softkey	From Attribute	To Attribute
1	<div>Common LDAP Attributes: (manager) DN of the entry of this person's supervisor</div> <div>Attribute: manager</div>	<div>Common LDAP Attributes: (dn) unique name of the entry, defined s</div> <div>Attribute: dn</div>
2	<div>Common LDAP Attributes: (employeeNumber) unique Personnel Number</div> <div>Attribute: employeeNumber</div>	<div>Common LDAP Attributes: (supervisorID) Supervisor Identification</div> <div>Attribute: supervisorID</div>
3	<div>Common LDAP Attributes: (departmentNumber) department ID/cost center</div> <div>Attribute: departmentNumber</div>	<div>Common LDAP Attributes: (departmentNumber) department ID/co</div> <div>Attribute: departmentNumber</div>
4	<div>Common LDAP Attributes: — Select an attribute from the attribute list, or enter one in the text area below —</div> <div>Attribute: </div>	<div>Common LDAP Attributes: — Select an attribute from the attribute lis</div> <div>Attribute: </div>

Output Window

Table 15: 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



# Chapter 9: Advanced Features

---

## Introduction

This chapter describes the capabilities and limitations of the Web browser for the following advanced features:

- Images
- Color
- Cascading Style Sheets

This chapter also provides considerations to develop effective Web pages for browser viewing.

This chapter is intended for IP telephone Web browser [Web page] designers, and assumes that readers are somewhat familiar with WML. This chapter is not intended to provide technical details on setting up a Web server, nor does it provide information on Web server technologies.

---

## 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 `&#060;`.

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 `&lt;`. The set of characters defined by the World Wide Web Consortium are fully supported in the Web browser in conformance with the standard.

For convenience, here are a few of these key symbols:

Description	Symbol	Numeric Entity	Name Entity
double quotation	"	<code>&amp;#34;</code>	<code>&amp;quot;</code>
ampersand	&	<code>&amp;#38;</code>	<code>&amp;amp;</code>
apostrophe	'	<code>&amp;#39;</code>	<code>&amp;apos;</code>
less than	<	<code>&amp;#60;</code>	<code>&amp;lt;</code>

---

## 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)

The Web browser does not support graphic animation.

WBMP is part of the Wireless Application Protocol, Wireless Application Environment Specification Version 1.1. The WBMP specification is available at:

<http://www.wapforum.org/docs/technical1.1/>.

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
<b>jpe</b>	image/jpeg
<b>jpeg</b>	image/jpeg
<b>jpg</b>	image/jpeg

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

Due to the current browser implementation, 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.

---

## Scrolling Through Images

Users can scroll through an image using the current navigation schema described in [Web Browser Navigation](#) in Chapter 6. The **Up** and **Down** navigation buttons move up and down the page to a portion of the image that is not initially visible on the display.

If the image is larger than the six display lines, pressing and holding the **Up** navigation button moves up an entire screen and displays the previous six lines of the image. If the image is larger than the six display lines, pressing and holding the **Down** navigation button moves down an entire screen and displays the next six lines of the image.

When the user focuses on the first line of an image and the image spans multiple lines on the screen, the following occurs when the user presses the **Down** navigation button:

- The focus drops to the bottom line of the image if the entire image can fit on the screen. For example, assume the image spans lines 1 to 5, and line 1 is in focus. The user presses the **Down** navigation button, and the focus shifts to the bottom of the image on line 5. If the user focuses on line 2 and presses the **Down** navigation button, again, the last line of the image on the screen is put in focus.
- If the image does not fit on the screen, pressing the **Down** navigation button changes the focus to the last line on the screen. For example, if the image spans lines 1 to 6 and does not fit on the screen, the focus shifts to the bottom of the screen on line 6.
- If the image is larger than the six display lines, the focus moves to line 7 and the next six lines of image display.

When the user focuses on the last line of an image and the image spans multiple lines on the screen, the following occurs when the user presses the **Up** navigation button:

- The focus goes to the first line of the image if the entire image can fit on the screen. For example, if the image spans lines 1 to 5, the user focuses on line 5 and presses the **Up** navigation button. The focus shifts to the first line of the image on line 1. If the user focuses on line 2 and presses line up, again the first line of the image on the screen will be in focus.
- If the image does not fit on the screen, pressing the **Up** navigation button shifts the focus to the first line on the screen. For example, if the image spans lines 1 to 6 and does not fit on the screen, the focus shifts to the bottom of the screen on line 6.

As another example, an image takes up three of the six available display lines. The user scrolls to any of the three lines displaying the image, presses **OK** and the entire three lines of the image are brought into focus. The image itself does not change color when in focus, but the screen area outside of the image changes color when in focus.

---

## 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](#) on page 111.

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.

---

## Image Size & Memory Requirements

All supported images can have up to 3 Mbytes of volatile memory per WML file. The image width and height must be smaller than or equal to 297x138 pixels. Larger images are reduced to 297x128 pixels or not rendered at all, in which case alternate text displays instead.

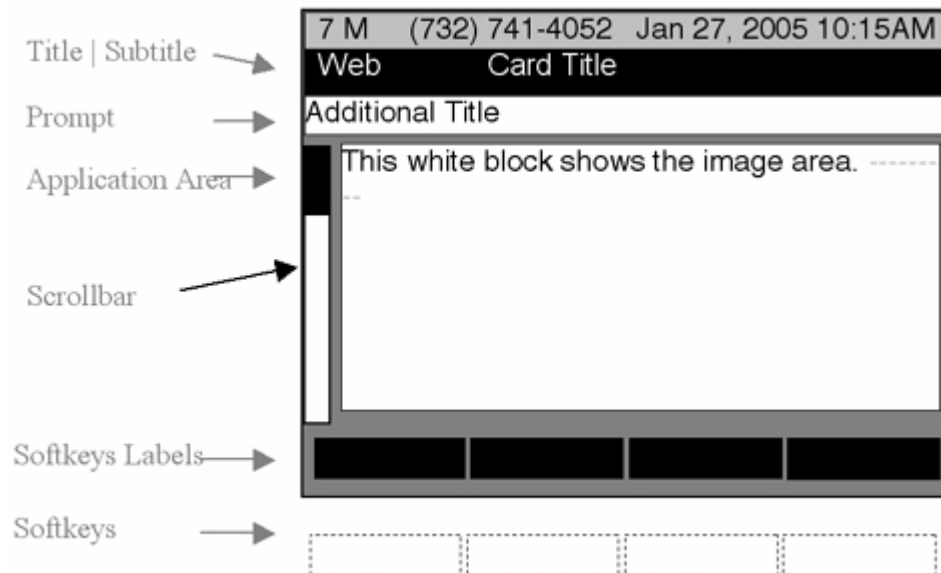
The browser processes images successfully as long as the upper limit of 3 Mbytes is not reached. The browser checks for sufficient memory as image processing starts. If sufficient memory to process the entire image is not found, the alternate text for the image displays instead of the actual image. For example, there are ten images, the first four of which use 1.5 Kbytes. But the fifth image is 2.0 Mbytes, and the browser does not have sufficient memory to process that image. The browser will continue to process the next image if sufficient memory for that is available. The browser displays a page with first four images, then an image for which the alternate text is displayed, followed by the sixth image and so on. Alternate text is displayed only for those images where:

- the browser could not successfully parse the image file because of memory limitations,
- the image could not be found on the server (404 error), or
- the image file is corrupted.

When an image tag exists inside a <do> tag, and it results in a failed attempt to retrieve the image through http, the <do> tag's label attribute value (text) displays on the softkey label.

## Image Justification

Figure 37: Image Area



The Image Area is the screen area that is available for the browser to render that particular image. If an image overflows the image area, the graphic engine expands to the right and downward until the image completes or truncates the image at the image area border (screen glass). The **hspace** and **vspace** values define the Image Area.

## Advanced Features

The Application Area starts at pixel 17 and ends at pixel 313, making the width 297. The numbers in the bullet list that follows are relative positions within the 0 to 296 pixel range. Images can be rendered starting at pixel 0 relative to the Application Area as follows:

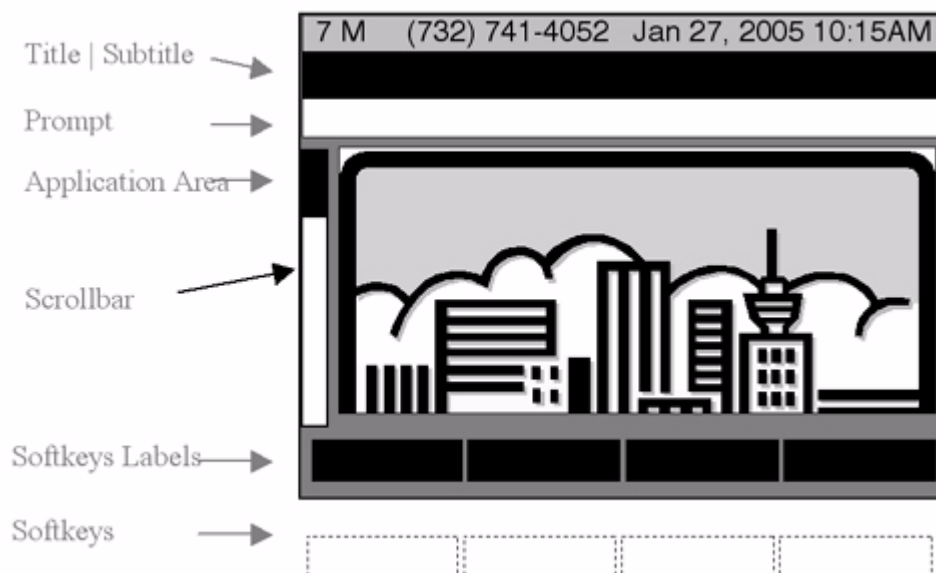
- A 0 (zero) pixel border on the left and right sides of all images displayed, as well as on top of any image displayed. The border area is empty.
- A 0 (zero) pixel border at the bottom.
- No 0 (zero) pixel border on the top of the remaining lines of images that span multiple lines.

Left-justified images begin at pixel 0. Center-justified images center at pixel 150. Right-justified images end at pixel 296. All positions are relative to the Application Area.

The default values for hspace and vspace are 0 pixels. This default value is honored only if the authors have not specified their own hspace and vspace values.

---

**Figure 38: Image Justification for the Entire Screen**



---

## Image Size

The maximum width of an image is 297 pixels. The maximum height is 23 pixels of height x 96 lines is 2208 pixels. Images greater than those maximums are truncated. This is translated as the maximum image width being from pixel 1 to pixel 297. The maximum height is 96 lines. Images greater than the maximums are truncated.

Each Application line is 20 pixels in height, including one black pixel at the bottom of the area, making the maximum supported image height 96 lines x 145 pixels.

JPEG images that are smaller than the normal softkey area can appear in the softkey label area. Such images are centered in the label area with the label background color visible around the JPEG. Any JPEG not fitting in the same area as a softkey label is truncated. Scaling to reduce images to fit the screen is not supported.

---

## 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.

---

## 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 <em> 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:**

The 9620 and 9630 only support four grayscale color. 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 16](#) shows which WML tags are parent and which tags inherit properties.

**Table 16: 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				

1 of 4

Table 16: WML Inheritance Table (continued)

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.
					go	postfield			
						setvar			
					img				
					prev	setvar			
					refresh	setvar			
				b					
				big					
				br					
				do	go	postfield			
						setvar			
					noop				
					prev	setvar			
					refresh	setvar			
				em					
				fieldset					
				i					
				input					

Table 16: WML Inheritance Table (continued)

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.
				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					

**Table 16: WML Inheritance Table (continued)**

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					
<b>4 of 4</b>									

[Table 17](#) 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 17: 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 below
p	Yes	Yes
a	Yes	Yes
anchor	Yes	Yes
img	Yes	Yes
do	Yes	Yes
onevent		
postfield		
go		
noop		
prev		
refresh		
input	Yes	Yes
optgroup	Yes	Yes
option	Yes	Yes
select		
setvar		
timer		

**Note:**

The <br/> 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
<b>background-color</b>	Sets the background color of an element.	<i>color-hex</i> <i>color-name</i> transparent	Supported.

Property	Description	Default Value	Notes
<b>background-color</b>	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
<b>color</b>	Sets the color of text.	<i>color</i> (default)	Supported.

Property	Description	Useful for
<b>color</b>	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.



# Index

## Numerical

9620 and 9630 IP Telephones	
Anchor Elements . . . . .	<a href="#">111</a>
Character Entities. . . . .	<a href="#">124</a> , <a href="#">155</a>
Colors and Fonts . . . . .	<a href="#">124</a>
Event Elements. . . . .	<a href="#">113</a>
Image Elements . . . . .	<a href="#">112</a>
Image Support . . . . .	<a href="#">156</a>
Input Elements . . . . .	<a href="#">119</a>
Task Elements . . . . .	<a href="#">117</a>
Text Formatting Tags . . . . .	<a href="#">110</a>
Variable Elements . . . . .	<a href="#">123</a>
Web Browser. . . . .	<a href="#">105</a>
WML Tags and Attributes . . . . .	<a href="#">106</a>
9620 and 9630 IP Telephones, Web Interface . . . . .	<a href="#">155</a>

## A

Access Key Input Mode (AIM) . . . . .	<a href="#">125</a>
Access Keys, Support for. . . . .	<a href="#">125</a>
Add-to-Speed Dial Functionality, for WTA Applications	<a href="#">94</a>
Administering the Push Interface . . . . .	<a href="#">57</a>
AIM Considerations . . . . .	<a href="#">129</a>
AIM, Terminating . . . . .	<a href="#">130</a>
Alerts	
Audio Push. . . . .	<a href="#">45</a>
Display. . . . .	<a href="#">29</a>
Subscribe Push. . . . .	<a href="#">53</a>
Topline Push . . . . .	<a href="#">37</a>
Anchor Elements . . . . .	<a href="#">111</a>
Audio Push	
Alerts . . . . .	<a href="#">45</a>
Barge Priority. . . . .	<a href="#">46</a>
Normal Priority . . . . .	<a href="#">46</a>
Priorities and States . . . . .	<a href="#">45</a>
Push Agent. . . . .	<a href="#">49</a>
Push Content. . . . .	<a href="#">50</a>
Push Response . . . . .	<a href="#">46</a>
Pushable vs. Non-Pushable States . . . . .	<a href="#">45</a>
Using the postfield Tag . . . . .	<a href="#">48</a>
XML Messages . . . . .	<a href="#">47</a>
Audio Push Example 1 . . . . .	<a href="#">45</a>
Audio Push Features. . . . .	<a href="#">44</a>
Audio Push Type . . . . .	<a href="#">44</a>
Avaya HTTP Header Extensions, for Push Interface . . . . .	<a href="#">65</a>

## B

Brief Feature Description . . . . .	<a href="#">22</a>
-------------------------------------	--------------------

## C

Call Interaction . . . . .	<a href="#">87</a>
Cascading Style Sheets, Support for. . . . .	<a href="#">161</a>
Character Entities . . . . .	<a href="#">124</a> , <a href="#">155</a>
Character Set Support . . . . .	<a href="#">86</a>
Click to Dial Functionality, for WTA Applications . . . . .	<a href="#">89</a>
Colors and Fonts . . . . .	<a href="#">124</a>
Colors, Display, for Web Browser . . . . .	<a href="#">86</a>
Creating Push Messages . . . . .	<a href="#">29</a>
Creating Web Sites	
General Background. . . . .	<a href="#">105</a>
Creating Web Sites for Other IP Telephones	
Summary Of WML Tags and Attributes . . . . .	<a href="#">127</a>
WML Document Skeleton . . . . .	<a href="#">106</a>
CSS2 Specifications . . . . .	<a href="#">168</a>

## D

Deck/Card Elements, for Web Browser. . . . .	<a href="#">87</a>
Dial Pad/URL Mapping Example. . . . .	<a href="#">126</a>
Directory Database Administration Interface . . . . .	<a href="#">144</a>
Display Area Breakdown . . . . .	<a href="#">73</a>
Display Push	
Barge Priority . . . . .	<a href="#">32</a>
Normal Priority . . . . .	<a href="#">31</a>
postfield Tag . . . . .	<a href="#">33</a>
Push Agent . . . . .	<a href="#">34</a>
Push Content . . . . .	<a href="#">35</a>
Push Message (PM). . . . .	<a href="#">33</a>
XML Messages . . . . .	<a href="#">33</a>
Display Push Example 1 . . . . .	<a href="#">30</a>
Display Push Example 2 . . . . .	<a href="#">34</a>
Display Push Type . . . . .	<a href="#">29</a>

## E

Error Messages, for Web Browser . . . . .	<a href="#">99</a>
Error Tones, in Web Browser . . . . .	<a href="#">96</a>
Event Elements . . . . .	<a href="#">113</a>
Existing Interfaces, for Avaya IP Telephones . . . . .	<a href="#">18</a>

## Index

---

### H

Hotel Application Example . . . . .	<a href="#">30</a>
HTTP Authentication, Support for . . . . .	<a href="#">95</a>
HTTP Error Messages . . . . .	<a href="#">67</a>
HTTP Header, in Web Browser . . . . .	<a href="#">96</a>
HTTP POST Address, Push Agent . . . . .	<a href="#">27</a>
HTTP Protocol. . . . .	<a href="#">96</a>
HTTP Server . . . . .	<a href="#">26</a>

---

### I

Idle Timer . . . . .	<a href="#">98</a>
Image Elements . . . . .	<a href="#">112</a>
Image Justification . . . . .	<a href="#">159</a>
Image Rendering . . . . .	<a href="#">157</a>
Image Size . . . . .	<a href="#">161</a>
Image Size & Memory Requirements . . . . .	<a href="#">158</a>
Image Support. . . . .	<a href="#">156</a>
Images Supported . . . . .	<a href="#">161</a>
Images, JPEG . . . . .	<a href="#">156</a>
Images, Scrolling Through . . . . .	<a href="#">157</a>
Images, WBMP . . . . .	<a href="#">156</a>
Input Elements . . . . .	<a href="#">119</a>
Installing the Thin Client Directory on the Server . . . . .	<a href="#">133</a>
Interrupt Screens . . . . .	<a href="#">44</a>
IP Telephone Interfaces . . . . .	<a href="#">17</a>

---

### J

JPEG Images . . . . .	<a href="#">156</a>
-----------------------	---------------------

---

### L

Links, for Web Browser. . . . .	<a href="#">82</a>
---------------------------------	--------------------

---

### M

Messages	
Display Push . . . . .	<a href="#">33</a>
Messages, for Audio Push . . . . .	<a href="#">47</a>
Messages, for Subscribe Push . . . . .	<a href="#">53</a>
Messages, for Topline Push . . . . .	<a href="#">39</a>

---

### N

Navigation, for Web Browser . . . . .	<a href="#">77</a>
Network Topology, illustration of . . . . .	<a href="#">17</a>

---

### O

Overview, of IP Telephone Interfaces . . . . .	<a href="#">17</a>
--	--------------------

---

### P

Page Loading, in Web Browser . . . . .	<a href="#">96</a>
Paging Indicators, for Web Browser . . . . .	<a href="#">80</a>
Physical Attributes, of Web Browser . . . . .	<a href="#">71</a>
Pixels	
Horizontal. . . . .	<a href="#">74</a>
Postfield Tag, in Topline Push . . . . .	<a href="#">40</a>
Postfield Tag, Using for Audio Push . . . . .	<a href="#">48</a>
Postfield Tag, Using for Subscribe Push . . . . .	<a href="#">54</a>
Postfield Tag, Using the, for Display Push . . . . .	<a href="#">33</a>
Priorities and States . . . . .	<a href="#">30</a>
Audio Push . . . . .	<a href="#">45</a>
Display Push . . . . .	<a href="#">30</a>
Subscribe Push . . . . .	<a href="#">53</a>
Topline Push . . . . .	<a href="#">38</a>
Push Administration . . . . .	<a href="#">57</a>
Push Agent	
Audio Push . . . . .	<a href="#">49</a>
Display Push . . . . .	<a href="#">34</a>
Subscribe Push . . . . .	<a href="#">55</a>
Topline Push . . . . .	<a href="#">42</a>
Push Agent - HTTP POST Address . . . . .	<a href="#">27</a>
Push Agent, About the . . . . .	<a href="#">26</a>
Push Architecture. . . . .	<a href="#">22</a>
Push Content . . . . .	<a href="#">25</a>
Audio. . . . .	<a href="#">50</a>
Subscribe Push . . . . .	<a href="#">55</a>
Topline Push . . . . .	<a href="#">42</a>
Push Content (PC)	
Display Push . . . . .	<a href="#">35</a>
Push Feature Description . . . . .	<a href="#">22</a>
Push Flow Process . . . . .	<a href="#">22</a>
Push Interface	
Denial of Service Timer . . . . .	<a href="#">63</a>
Overview . . . . .	<a href="#">21</a>
Requirements . . . . .	<a href="#">57</a>
Retry Timer . . . . .	<a href="#">62</a>
Security. . . . .	<a href="#">57</a>
Subscription Service. . . . .	<a href="#">60</a>
Troubleshooting . . . . .	<a href="#">65</a>
Validation Scenarios. . . . .	<a href="#">59</a>
Push interface . . . . .	<a href="#">19</a>
Push Message	
Subscribe Push . . . . .	<a href="#">54</a>
Topline . . . . .	<a href="#">40</a>
Push Message Flow . . . . .	<a href="#">24</a>
Push Messages, Creating. . . . .	<a href="#">29</a>
Push operation . . . . .	<a href="#">23</a>
Push Response	
Display Push . . . . .	<a href="#">31</a>
Subscribe Push . . . . .	<a href="#">53</a>
Push Types . . . . .	<a href="#">27</a>
Push/Pull Process, in Push Interface. . . . .	<a href="#">23</a>

Pushable vs. Non-Pushable States . . . . .	<a href="#">38</a>
Pushable vs. Non-Pushable States, for Display push . . . . .	<a href="#">30</a>
Push-Code Responses . . . . .	<a href="#">65</a>

## R

Recommendations, for setting up TPSLIST . . . . .	<a href="#">60</a>
Requirements, for Push Interface . . . . .	<a href="#">57</a>
Response tag, Using for Topline Push . . . . .	<a href="#">42</a>
RTP Port . . . . .	<a href="#">48</a>

## S

Scrolling . . . . .	<a href="#">80</a>
Security, for Push Interface . . . . .	<a href="#">57</a>
Setting Up Push Messages . . . . .	<a href="#">29</a>
Softkeys	
Author-Defined . . . . .	<a href="#">79</a>
Default . . . . .	<a href="#">78</a>
Standard Text Entry Example . . . . .	<a href="#">84</a>
Stock Alert Example . . . . .	<a href="#">37</a>
Stock Alert Message . . . . .	<a href="#">43</a>
Stock Alert Message, Telephone Display Prior to Receiving . . . . .	<a href="#">41</a>
Subscribe Push	
Alerts . . . . .	<a href="#">53</a>
Features . . . . .	<a href="#">52</a>
Priorities and States . . . . .	<a href="#">53</a>
Push Agent . . . . .	<a href="#">55</a>
Push Content . . . . .	<a href="#">55</a>
Push Message . . . . .	<a href="#">54</a>
Push Response . . . . .	<a href="#">53</a>
XML Messages . . . . .	<a href="#">53</a>
Subscribe Push Type . . . . .	<a href="#">52</a>
Subscribe Tag, Using the . . . . .	<a href="#">55</a>
Subscriber Service, for Push Interface . . . . .	<a href="#">61</a>
Subscription List, for Push Interface . . . . .	<a href="#">61</a>
Subscription Service . . . . .	<a href="#">60</a>
Subscription Update . . . . .	<a href="#">62</a>
Successful Push Response . . . . .	<a href="#">31</a> , <a href="#">38</a>
Syntax Implementation, for WTA Applications . . . . .	<a href="#">89</a>
System Values . . . . .	<a href="#">99</a>
System/Web Interaction . . . . .	<a href="#">98</a>

## T

TAPI (Telephony Application Programmer's Interface) . . . . .	<a href="#">19</a>
Task Elements . . . . .	<a href="#">117</a>
Terminating AIM . . . . .	<a href="#">130</a>
Text Editing Modes, for Web Browser . . . . .	<a href="#">85</a>
Text Elements . . . . .	<a href="#">109</a>
Text Entry Example . . . . .	<a href="#">84</a>

Text Entry, for Web Browser . . . . .	<a href="#">83</a>
Text Formatting Tags . . . . .	<a href="#">110</a>
Text Message . . . . .	<a href="#">36</a>
Text Message Features, for Topline Push . . . . .	<a href="#">36</a>
Thin Client Directory, Avaya-Provided Download Files . . . . .	<a href="#">133</a>
Thin Client Directory, Installing on the Server . . . . .	<a href="#">133</a>
Thin Client Directory, Installing the . . . . .	<a href="#">134</a>
Timer	
Denial of Service . . . . .	<a href="#">63</a>
Retry . . . . .	<a href="#">62</a>
Timer, Idle . . . . .	<a href="#">98</a>
Top Line Defaults, for Web Browser . . . . .	<a href="#">82</a>
Topline Push	
Alerts . . . . .	<a href="#">37</a>
Barge Priority . . . . .	<a href="#">39</a>
Normal Priority . . . . .	<a href="#">38</a>
postfield Tag . . . . .	<a href="#">40</a>
Priorities and States . . . . .	<a href="#">38</a>
Push Agent . . . . .	<a href="#">42</a>
Push Content . . . . .	<a href="#">42</a>
Push Message . . . . .	<a href="#">40</a>
Push Response . . . . .	<a href="#">38</a>
Pushable vs. Non-Pushable States . . . . .	<a href="#">38</a>
Text Message Features . . . . .	<a href="#">36</a>
XML Messages . . . . .	<a href="#">39</a>
Topline Push Example 1 . . . . .	<a href="#">37</a>
Topline Push Example 2 . . . . .	<a href="#">41</a>
Topline Push Type . . . . .	<a href="#">36</a>
Topline tag, using for Topline Push . . . . .	<a href="#">43</a>
TPSList (See Trusted Push Server List) . . . . .	<a href="#">58</a>
Troubleshooting the Push Interface . . . . .	<a href="#">65</a>
Truncation Rules, for Web Browser . . . . .	<a href="#">82</a>
Trusted Push Server List . . . . .	<a href="#">58</a>
Two Step View – Push/Pull Process . . . . .	<a href="#">23</a>
Typical System-Wide Network Topolgy. . . . .	<a href="#">17</a>

## U

URI Examples, for Push Interface Validation . . . . .	<a href="#">59</a>
---	--------------------

## V

Validation Scenarios, for Push Interface . . . . .	<a href="#">59</a>
Variable Elements . . . . .	<a href="#">123</a>

## W

WBMP Images . . . . .	<a href="#">156</a>
Web Application User Interface . . . . .	<a href="#">137</a>
Web Applications . . . . .	<a href="#">131</a>

## Index

Web Browser	
About the . . . . .	<a href="#">69</a>
Call Interaction . . . . .	<a href="#">87</a>
Character Set Support . . . . .	<a href="#">86</a>
Display Colors . . . . .	<a href="#">86</a>
Error Messages . . . . .	<a href="#">99</a>
Moving Up and Down a Card . . . . .	<a href="#">80</a>
Navigation . . . . .	<a href="#">77</a>
Page Loading . . . . .	<a href="#">96</a>
Physical Attribute . . . . .	<a href="#">71</a>
Relationship between Line/Page Movement, Focus, and Top Line Display . . . . .	<a href="#">81</a>
Requirements for Deck/Card Elements. . . . .	<a href="#">87</a>
Summary of WML Tags and Attributes . . . . .	<a href="#">101</a>
Support for HTTP Authentication. . . . .	<a href="#">95</a>
Text Editing . . . . .	<a href="#">85</a>
Text Entry . . . . .	<a href="#">83</a>
Truncation Rules and Links . . . . .	<a href="#">82</a>
Web Browser Features. . . . .	<a href="#">29</a>
Web Browser Features, with Display Push Type . . .	<a href="#">29</a>
Web Browser for 9620 and 9630 IP Telephones . . .	<a href="#">105</a>
Web History . . . . .	<a href="#">86</a>
Web Interface . . . . .	<a href="#">105</a>
Web/System Interaction. . . . .	<a href="#">98</a>
Web-Specific System Values . . . . .	<a href="#">99</a>
Wireless Telephony Applications (WTA) . . . . .	<a href="#">88</a>
WML Tags and Attributes . . . . .	<a href="#">106</a>
WML Tags and Attributes, Summary for Web Browser . . . . .	<a href="#">101</a>

---

## X

XML Message Parsing . . . . .	<a href="#">24</a>
XML Messages	
Audio Push . . . . .	<a href="#">47</a>
Display Push . . . . .	<a href="#">33</a>
Subscribe Push . . . . .	<a href="#">53</a>
Topline Push . . . . .	<a href="#">39</a>