DEVELOPER GUIDE

UC Software 5.0.0 | September 2013 |3725-49106-001 Rev A

Web Application for Polycom® Phones

6001 America Center Drive
San Jose, CA 95002
USA

Trademarks

End User License Agreement

By installing, copying, or otherwise using this product, you acknowledge that you have read, understand and agree to be bound by the terms and conditions of the End User License Agreement for this product.

Patent Information

The accompanying product may be protected by one or more U.S. and foreign patents and/or pending patent applications held by Polycom, Inc.

Open Source Software Used in this Product

This product may contain open source software. You may receive the open source software from Polycom up to three (3) years after the distribution date of the applicable product or software at a charge not greater than the cost to Polycom of shipping or distributing the software to you. To receive software information, as well as the open source software code used in this product, contact Polycom by email at OpenSourceVideo@polycom.com.

Disclaimer

While Polycom uses reasonable efforts to include accurate and up-to-date information in this document, Polycom makes no warranties or representations as to its accuracy. Polycom assumes no liability or responsibility for any typographical or other errors or omissions in the content of this document.

Limitation of Liability

Polycom and/or its respective suppliers make no representations about the suitability of the information contained in this document for any purpose. Information is provided “as is” without warranty of any kind and is subject to change without notice. The entire risk arising out of its use remains with the recipient. In no event shall Polycom and/or its respective suppliers be liable for any direct, consequential, incidental, special, punitive or other damages whatsoever (including without limitation, damages for loss of business profits, business interruption, or loss of business information), even if Polycom has been advised of the possibility of such damages.

Customer Feedback

We are striving to improve our documentation quality and we appreciate your feedback. Email your opinions and comments to DocumentationFeedback@polycom.com.

Visit the Polycom Support Center for End User License Agreements, software downloads, product documents, product licenses, troubleshooting tips, service requests, and more.

Contents

About This Guide

This guide uses a number of conventions that help you to understand information and perform tasks.

Conventions Used in This Guide

This user guide contains terms, graphical elements, and a few typographic conventions. Familiarizing yourself with these terms, elements, and conventions will help you perform certain tasks.

Information Elements

The following icons are used to alert you to various types of important information in this guide

Icons Used in this Guide

Name	*Icon*	
Note		The Note icon highlights information of interest or important information needed to be successful in accomplishing a procedure or to understand a concept.
Administrator Tip		The Administrator Tip icon highlights techniques, shortcuts, or productivity related tips.
Caution		The Caution icon highlights information you need to know to avoid a hazard that could potentially impact device performance, application functionality, or successful feature configuration.
Warning		The Warning icon highlights an action you must perform (or avoid) to prevent issues that may cause you to lose information or your configuration setup, and/or affect phone or network performance.
Web Info		The Web Info icon highlights supplementary information available online such as documents or downloads on support.polycom.com or other locations.
Timesaver		The Timesaver icon highlights a faster or alternative method for accomplishing a method or operation.
Power Tip		The Power Tip icon faster, alternative procedures for advanced administrators already familiar with the techniques being discussed.
Troubleshooting		The Troubleshooting icon highlights information that may help you solve a relevant problem or to refer you to other relevant troubleshooting resources.
Settings		The Settings icon highlights settings you may need to choose for a specific behavior, to enable a specific feature, or to access customization options.

Typographic Conventions

A few typographic conventions, listed next, are used in this guide to distinguish types of in-text information.

Typographic Conventions Used in This Guide

Convention	*Description*
Bold	Highlights interface items such as menus, soft keys, file names, and directories. Also used to represent menu selections and text entry to the phone.
Italics	Used to emphasize text, to show example values or inputs, and to show titles of reference documents available from the Polycom Support Web site and other reference sites.
Blue Text	Used for cross references to other sections within this document and for hyperlinks to external sites and documents.
`Courier`	Used for code fragments and parameter names.

Writing Conventions

This guide also uses font styles to distinguish conditional information as listed in Table 3, as shown next.

Writing Conventions Used in This Guide

Convention	*Description*
<MACaddress>	Indicates that you must enter information specific to your installation, phone, or network. For example, when you see <MACaddress>, enter your phone's 12-digit MAC address. If you see <installed-directory>, enter the path to your installation directory.
>	Indicates that you need to select an item from a menu. For example, **Settings** > **Basic** indicates that you need to select **Basic** from the **Settings** menu.
parameter.*	Used for configuration parameters. If you see a parameter name in the form `parameter.*` , the text is referring to all parameters beginning with `parameter`. See the section Reading the Feature Parameter Tables in Polycom's UC Software 5.0.0 Administrators' Guide for examples.

Coding Conventions

Sample code is shown in this guide to assist you in writing your applications. All samples are presented in the following format.

Table 1: Sample Code

```
<html>
 <body> <br/>
  Click on the link to engage the DND feature
  <a href="Key:DoNotDisturb">DNDSettings</a>
 </body>
  <softkey index="1" label="Back" action="SoftKey:Back"/>
  <softkey index="2" label="Exit" action="SoftKey:Exit"/>
</html>
```

What's in This Guide?

This developer guide is organized into eight chapters. The first chapter, *Getting Started*, introduces Polycom and Unified Communication solutions. The following chapters provide information on how to configure and deploy specific Polycom products and systems using the SDK and the latest UC software. The final chapters show you where to get help and other sources as well as outline known issues and workarounds.

Chapter 1: Getting Started gives you a quick overview of knowledge, hardware, and software you need before you begin. This chapter also provides frequently asked questions (FAQs) and resources for further help along with a checklist of things you can do before you begin writing your Web application

Chapter 2: Understanding Web Application Development on Polycom Phones gives you an overview of Web applications on Polycom phones.

Chapter 3: Getting to Know the XML API Application Interface provides information about the XML application information and gets you started with writing your Web application.

Chapter 4: Writing Your Web Application shows you how to write your Web application and provides examples.

Chapter 5: Using Configuration Parameters provides information on the phone's configurable parameters and you can modify those parameters for your application.

Chapter 6: Getting Help shares links to support documents and websites from Polycom, Polycom partners, and other information that will assist you. You will also find links to the Polycom Community, which contains a number of discussion forums you can use to share ideas with your colleagues.

Chapter 7: Troubleshooting provides help when troubleshooting your Web application development.

Chapter 8: References lists further information mentioned in this guide that can help you use the SDK along with additional information to help you write your Web application.

1: Getting Started

Getting Started provides you with information on helpful resources and recommended software tools that can aid you in Web application development on Polycom® phones with the Polycom UC software.

This Web Application Developer's Guide provides you with information on how to install and use the Polycom Software Development Kit (SDK). This guide shows you how to plan, create, and develop Web applications to run on Polycom phones using UC software.

This guide is designed to provide Web application creators with information for developing and deploying Web applications to Polycom phones. This guide is not intended for end users and does not provide user-level information on how to use any specific Web applications. You need to be familiar with creating Web applications before reading this guide.

All Polycom phones run the Polycom® UC software. The software can be downloaded from the Polycom Support Web site. Polycom also provides a simulation of different phone models with the Software Development Kit (SDK). The Polycom Software Development Kit (SDK) can be downloaded from the Applications page on the Polycom Web site.

Recommended Software Tools

Polycom recommends you use an XML editor – such as XML Notepad++ – to create and edit configuration files. Using an XML editor ensures that you create valid XML files. If the configuration files you create are not in the form of a valid XML structure, the files cannot load correctly on the phones.

For more complex applications, you need to use a fully Integrated Development Environment (IDE) like Eclipse or Microsoft Visual Studio. You can also use the Linux tool Curl to send arbitrary HTTP and XML content to a phone.

Getting Help and Support Resources

This developer guide includes a Getting Help section where you can find links to Polycom product, support, and partner sites. You can also find information about The Polycom Community, which provides access to discussion forums you can use to discuss hardware, software, and partner solution topics with your colleagues. To register with the Polycom Community, you need to create a Polycom online account.

The Polycom Community includes access to Polycom support personnel as well as user-generated hardware, software, and partner solutions topics. You can view top blog posts and participate in threads on any number of recent topics.

Planning Your XML API Interface Requirements

The XML API is supported in applications running on Polycom® SoundPoint® IP 321/331/335, 450, 550, 560, 650, and 670 desktop phones; SoundStation® IP 5000, 6000, and 7000 conference phones; SoundStation Duo conference phones; VVX® 500, 600, and 1500 business media phones; VVX® 300 and 400 desktop phones, and SoundStructure®.

The XML API is designed to provide you with flexibility when developing Polycom phone applications while securely integrating into the phone's capabilities and functions. The XML API features are supported by the Polycom browser and microbrowser, except where noted.

You'll find detailed descriptions and examples for the XML API features in Getting to Know the XML API Application Interface.

Strategies for Web Application Development

Before you start to write your application, you need to define the purpose of your application. You need to decide what tasks the application is going to perform, how complex the application will be, and if the application will run on one or several phone models. You also need to take the varying screen sizes of the phones into account to determine a common size for your application.

Creating screen mockups of your application using a visualization tool, such as PowerPoint or Visio, can aid you in your development process and help others involved understand the overall application concept.

You need to determine your audience and understand their technological capabilities. Understanding who you are creating an application for helps you determine which features are most useful for your users.

If you are new to developing Polycom phone applications, knowing a few tips to use and pitfalls to avoid before you begin can aid you in the process. Use the following Best Practices lists for guidance when developing applications to run on the Polycom microbrowser and browser.

Best Practices for Web Application Development

Consider the following when developing applications for the browser or microbrowser:

- **Using the HTTP User Agent** You can use the HTTP user agent header information to determine a variety of details about the phone, such as the model, and deliver content tailored specifically for the phone's screen size and other capabilities. You can also use JavaScript to detect the screen and/or window size for applications running on phones that support the browser.
- **Supporting Image Formats** A majority of phone models support both JPG and BMP image formats, but keep in mind that phones that support the microbrowser do not support the JPGs format. When considering the size and quality of your application's images, you need to know that compressed JPG images are better for large images, and BMP images have a better quality for smaller images but lack the compression benefit. For more information, see Image Tags.
- **Scaling Images** If an image is too large to fit a specific phone screen, ensure the server is able to scale the image to fit the phone's screen dimensions. Keep in mind that the phone's performance suffers if the phone has to scale oversized images to fit the screen.
- **Previewing Images** To preview the appearance of your application's images on the phone, set your computer monitor color depth to 16-bit. This ensures that any images your application sends to the phone are prepared using a color depth that is equal to or lower than the phones color depth capabilities.
- **Using Soft Keys** If the phone model you are developing your application for supports soft keys or shortcuts, add soft keys and shortcuts to your application to take advantage of their functionality. See Configuring Programmable Soft Keys. For more information, refer to the chapter Configuring Soft Keys in the Polycom UC Software 5.0.0 Administrators' Guide.

- **Positioning Soft Keys** Use a hard key press simulation to test your application, provided the phone supports this function. When working with simulators, simulating key presses for call screen soft keys, such as EndCall and Transfer call controls, are not reliable. Depending on the phone's configuration, you can adjust the soft key positions.
- **Encrypting Configuration Files** For security reasons, make sure you encrypt the configuration files included in the application. You will need to generate a key, which you can download to the phone and use to encrypt configuration files. See Quick Tip 67442: When Encrypting Polycom UC Software Configuration Files.
- **Pushing Sensitive Data** You can retrieve information from a HTTPS site by sending a request to the phone with a URL push. The URL push itself does not leak sensitive information, but you need to ensure the data is encrypted and avoid pushing security sensitive data directly to the phone.
- **Using HTTPS for Telephone Notification Events** Use HTTPS for telephone notification events and state polling to protect sensitive information, such as the phone MAC address, caller name, and phone number, which are contained in both.
- **Implementing User Confirmation** When including emergency push notifications, implement a user confirmation response. In cases where the push and call happen simultaneously, the push message can be lost or hidden without the user knowing the event occurred. Add a confirmation response to ensure the user receives the notification along with the call.
- **Using Tel URI** To improve the performance of the phone with your application, you need to code your application to use Tel Uri API to make calls instead of using digit key press simulation for dialing. For more information, see Using Telephone Integration URIs.
- **Removing White Space in Code** Review your HTML, JavaScript, and CSS files to remove whitespace from the code before delivery.
- **Setting the Idle Browser Refresh Cycle** There are two ways you can refresh a browser: you can set the `mb.idleDisplay.refresh` parameter to a non-zero value in seconds, or set the same parameter value to 0 and create a JavaScript code to do the refresh.
- **Providing a Link for Text Input** Although the idle browser is interactive, do not use the idle browser for text input. If you want to implement text input into the idle browser, provide a link for another page to be displayed on the main Browser.
- **Creating a Link Area** Ensure the link areas are large enough for users to easily touch with their finger. For example, avoid embedding a button inside the link area.
- **Sharing the Idle Browser** You can share the idle browser with multiple applications, so consider what the page needs to do when not in use. For example, using iframe AJAX running in the background of a hidden browser can have an impact on the performance of the phone and make it run slower. Consider monitoring the OnUnload JS event. Use the PolyUri DOM extensions `PolyUri.shownSig` and `PolyUri.hiddenSig` to trigger behavior when your application is hidden or shown.

Best Practices for Microbrowser Web Application Development

Consider the following when developing Polycom phone applications specifically to support the capabilities of the microbrowser:

- **Minimizing Onscreen Information** Make sure you do not include too much information at one time on the screen of the idle browser. Because the idle browser has no scrolling capability, excessive information cannot be viewed and slows down the performance of the phone and wastes the bandwidth.
- **Limiting Page Content** If a page contains too much information, use multiple pages to avoid a delay in displaying information or scrolling on the page. Keep in mind that the phone's memory limitation hinders the amount of content that can be displayed on the screen.
- **Avoiding Interactive Features for Idle Browser Pages** Make sure to create an application that is informational only. Because the idle browser is not interactive, avoid including user input elements like soft keys, buttons, or text input in your application.
- **Providing Clear Instructions** Make sure to include clear instructions for the user. For example, if the Select soft key is enabled on an idle page, there needs to be an indication like 'Press Select' on the page. Do not assume the user will know to press the Select soft key.
- **Stabilizing the Refresh Rate** Make sure your application does not require a frequent refresh of the idle browser. You can control how often your pages automatically refresh using JavaScript. A minimum 5 second delay is enforced by the phone.
- **Controlling the Refresh Cycle** You need to control the refresh cycle using your application's response HTTP to the server. Configure the refresh cycle in the configuration file to be as long as possible and only used as a recovery mechanism in case the HTTP connection fails due to a network problem.
- **Providing Adequate Text Elements** If a form contains text inputs and soft keys, provide a link next to the text input so the cursor can be easily moved from the text box to display the designated soft keys.
- **Limiting Cursor Navigations** If a page includes multiple buttons or links, consider adding soft keys as an additional input method for frequently used selections to avoid cursor navigations.

2: Understanding Web Application Development on Polycom Phones

Polycom has two types of browser environments for Polycom phones running Polycom UC software: browser and microbrowser. Before you write your application, you need to explore both environments and choose the one that provides the best mapping to the phone models you want to write applications for. This chapter provides an overview of each browser environment and example applications that will run in each environment.

Table 2: Polycom Phones that Support the Browser and Microbrowser shows which phones support the microbrowser and the total display area.

Table 2: Polycom Phones that Support the Browser and Microbrowser

Phone	*Browser is supported*	*Microbrowser is supported*
VVX 300/310/400/410/500/600	Yes	No
VVX 1500[1]	No (when running SIP 3.1.3 or earlier) Yes (When running SIP 3.2.2 or later.)	Yes (when running SIP 3.1.3 or earlier) No (When running SIP 3.2.2 or later.)
SoundPoint IP 321/331/335/450/ 550/560/650/670	No	Yes
SoundStation IP 5000/6000/7000	No	Yes
SoundStation Duo[2]	No	Yes
SoundStructure[3]	No	No

[1] When running SIP 3.1.3 or earlier.

[2] Only supported on the SoundStation Duo while in SIP mode (as opposed to PSTN mode).

[3] SoundStructure doesn't have a display screen.

Note: SoundStructure Supports Features of XML API

SoundStructure doesn't have a display screen, but it supports the InternalURI, Telephony Notifications, and Polling features of XML API.

Web applications running on Polycom phones can be as simple as a list of contacts, or as complex as a nurse call system. The phone's screen size is one of the determining factors when developing applications. Polycom phones support a full browser that enables you to interact with Web pages as you

would on a computer, or a microbrowser that supports a limited set of XHTML tags and displays limited content. For a more information on which Polycom phones support a browser or a microbrowser, see

What is the Browser? and What is the Microbrowser?.

This chapter covers the following topics:

- What is the Browser?
- What is the Microbrowser?
- What is XHTML?
- What's the Difference Between the Microbrowser and Browser?
- What's in the Software Development Kit?

Admin Tip: Additional Tools to Create Applications

You can use any development languages or servers you choose, including JavaScript, PHP, Python, Django, Tomcat or Apache. Use the tools you are most comfortable using, or those that are most supported by your IT department.

Web Info: Support for Polycom Phones

You can find documentation for all Polycom phones on the Polycom Support site. Choose your phone model for specific documentation. For more information, contact your Polycom distributer.

What is the Browser?

When a URL is configured on a phone or device, the browser enables you to browse and interact with Web pages like any browser. The browser supports Web-browsing functionality that is similar to Google Chrome on Android smartphones or Apple Safari on iPhones.

Polycom phone browsers are based on the open source WebKit platform. You can find the version of the WebKit platform on your phone in the user-agent string in network captures in the User-Agent HTTP header.

The browser supports Web 2.0 applications with the following features:

- XHTML 1.1
- HTML 4.01 with partial support for HTML 5
- CCS 2.1 with partial support for CCS 3.0
- SVG 1.1 (partial support)
- JavaScript
- XML HTTP Request
- DOM
- HTTP 1.1
- AJAX

The browser display is limited by the total display area available on each Polycom phone model. If you are creating an application for a specific Polycom phone model, consider the display area as a factor in the design of your Web application.

The following table shows the total screen size (in pixels) for each phone model and the audio and video tag support for each.

Table 3: Polycom Phone and Browser Window Sizes and HTML5 Audio and Video Tag support

Phone	*Total Screen Size (pixels)*	*Full Browser Screen Size (pixels)*		*Idle Browser Screen Size (pixels)*	*HTML5 Audio Tag Support*	*HTML5 Video Tag Support*
		When toolbar-auto-hide is enabled	When toolbar-auto-hide is disabled			
VVX 600	480x272	480x252	480x217	480x190	Yes	Yes
VVX 500	320x 240	320x220	320 x 185	320 x 158	Yes	Yes
VVX 400	320x240	320x240	320x190	320x240	Yes	No
VVX 300	208x104	208x104	208x68	208x104	Yes	No
VVX 1500[1]	800 x 480	800x455	800 x 395	610 x 360	No	No

[1] When running SIP 3.2.2 or later.

What is the Microbrowser?

The microbrowser is what Polycom calls the smaller, limited-capability browser available on phones that feature smaller display screens. The microbrowser's functions are similar to the browser available on phones with larger displays, but the microbrowser supports only a subset of the XHTML 1.0 features available to the browser. For example, it can connect to Web servers hosted on the Internet or intranet to display XHTML pages, but it does not have full Web browser functionality.

The XHTML pages displayed on the microbrowser contain static or dynamic information. This information is defined as follows:

- **Static XHTML** These pages are created using XHTML editors and are hosted by the Web server. You can access these pages from the microbrowser using HTTP protocol. These XHTML pages are static because the information displayed is coded into the XHTML pages. These pages do not include information that continuously changes or contacts other services for updates.

- **Dynamic XHTML** These pages involve dynamic information updates of XHTML pages by an application hosted on the Web server. The application residing on the Web server retrieves information from an intranet or Internet data service provider like Yahoo, Exchange Server, Call Control Servers, or other enterprise servers.

When a Web or intranet URL is configured, the microbrowser downloads XHTML content into the phone's memory and parses it to identify XHTML tags. The phone renders these tags onto the display screen, and the pages display according to the graphical capabilities and display size of the phone's screen. Avoid creating complex Web pages to ensure the pages display correctly on all devices.

Note: No Support for JavaScript

The microbrowser does not support scripting such as JavaScript. All data actions entered into forms are processed by the server using POST or GET methods.

User Tip: Accessing Applications on the Microbrowser

You can launch the microbrowser on a SoundPoint IP or SoundStation IP phone by pressing the **Applications** key or access the microbrowser through the **Menu** key by selecting **Applications**.

The following table shows the difference in screen dimensions of the browser and microbrowser on Polycom phones.

Table 4: Polycom SoundPoint IP, SoundStation IP, and VVX 1500 Screen and Microbrowser Window Sizes

Phone	*Total Screen Size (pixels)*	*Microbrowser Screen Size (pixels)*
SoundPoint IP 321/331/335	102 x 33	88 x 12
SoundPoint IP 450	256 x 116	171 x 72
SoundPoint IP 550/560/650/670	320 x 160	213 x 110
SoundStation IP 5000	240 x 68	240 X 32
SoundStation IP 6000	240 x 68	240 X 32
SoundStation IP 7000	255 x 128	255 x 79
SoundStation Duo[1]	240 x 68	240 X 32
VVX 1500[2]	800 x 400	562 x 322

[1] Only supported on the SoundStation Duo while in SIP mode (as opposed to PSTN mode).

[2] When running SIP 3.1.3 or earlier.

What is XHTML?

XHTML, or eXtensible HyperText Markup Language, is a family of XML markup languages that mirror or extend versions of the widely-used Hypertext Markup Language (HTML), the language used to write Web pages. XHTML is HTML 4.01 redesigned as XML.

You need to have experience working with HTML and XHTML programming or access to someone who has experience in order to properly use and understand the information provided in this guide.

For more information, refer to the following online documents:

- W3C HTML 4.0.1 Specification
- W3C HTML 5 Specification
- W3C XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)
- W3C XHTML™ Basic 1.1 - Second Edition
- W3C XHTML™ 1.1 - Module-based XHTML - Second Edition
- W3C XHTML Tables Module - XHTML™2.0

What Are the Differences Between the Microbrowser and Browser?

The main difference between the microbrowser and the browser is that phones with larger screens are capable of displaying more complex Web content in the browser than phones using the microbrowser. Other comparisons between the microbrowser and the browser are shown in the following table:

Table 5: Microbrowser and Browser Comparison

Features	*Microbrowser*	*Browser*
XML API	programmable soft keys, telephone integration URIs, push requests, telephone notification events, phone state polling	telephone integration URIs, push requests, telephone notification events, phone state polling
Mark Up Languages	HTML 4.01 XHTML 1.0	XHTML 1.1. HTML 4.01 with partial support for HTML 5; CSS 2.1 with partial support for CSS 3.0; SVG 1.1 (partial support); JavaScript; XML HTTP Request; DOM; HTTP 1.1; AJAX

What's in the Software Development Kit?

The Polycom® Software Development Kit (SDK) provides you with a set of tools to help you develop XML API/XHTML applications. The SDK provides software-based simulators for certain VVX phones. Refer to the Software Development Kit page to learn more about which phone model simulators are supported in each of the available SDKs.

The SDK installation file installs the following components on your computer:

- Phone Simulators
- The SDK Quick Start Guide
- An Apache Tomcat Web Server
- The XML API Web Testing Tool

Launching the Polycom SDK

The Polycom SDK is a set of tools designed to assist you in developing XML API/XHTML applications for the VVX phones and provide a simulation of these phones. You need to download the SDK before creating your application. Using the SDK, you can test your application on the simulated phones and make any necessary adjustments before finalizing.

The Polycom SDK is available for download from the Polycom Support site. Refer to the SDK Quick Start Guide for more information on installing and using the SDK.

The SDK creates temporary files in the installed directory at startup. By default, Microsoft Windows Vista does not allow applications to modify/add files in certain directories due to security concerns.

To work around this issue:

1. Turn off the User Access Control (UAC) feature.
2. Install the SDK in another directory besides Program Files or Windows.

   For example, during the installation process, enter c:\PolycomSDK as the install directory

Note: Differences Between Applications Running on Phones and the SDK

It is important to consider the difference between an application running on the computer-based simulator and the same application running on a real Polycom phone. Graphic animations and other processor or memory intensive functions could perform differently in either environment. There are other slight differences in the behavior of the simulators with respect to different versions of phone software.

Troubleshooting: Before Running the Simulator

The simulator uses port 80, and if any existing applications on your computer are using port 80, you either need to shut those applications down or configure a different port other than 80 for those applications.

What’s New in the Latest Polycom UC Software Updates?

If you’ve used previous versions of the Polycom SDK to develop applications, you may notice differences in the latest capabilities supported by the phones or changes to the simulators themselves. This section

covers the range of functional improvements that were introduced with the latest UC software updates from 4.0.1 to 5.0.0.

Note: Supporting Legacy Phone Models

Certain phone models, or legacy phones, are not supported in the Polycom UC Software 4.0.x release. For the appropriate software versions to use with these legacy phones, see the Polycom UC Software/Polycom SIP Software Release Matrix. The software matrix indicates the level of software support for each Polycom phone.

The following features were introduced in UC software 4.0.1:

- SoundStation Duo conference phone
- Flexible Line Key Assignment
- SoundStructure VoIP Interface
- Support for the VVX 500 Business Media Phone

  Refer to UC Software 4.0.1 Administrator's Guide for additional information.

The following features were introduced in UC software 4.0.1 B:

- Support for SoundStructure

The following features were introduced with UC software 4.0.2:

- Browser has support for clearing the cookies and temporary Internet files from the cache.

The following features were introduced with UC software 4.0.4:

- Support for Tel URI Action on Microbrowser.
- Added ability to dial '+' character by pressing '*' twice for international dialing.
- Added ability to process DTMF entry during early media.

  Refer to UC Software 4.0.4 Release Notes for additional information.

The following features were introduced in UC software 4.1.2:

- Support for the VVX 600 Business Media Phone.

  Refer to UC Software 4.1.2 Release Notes for additional information.

The following features were introduced with UC software 4.1.4:

- Added support for VVX 300/310 and VVX 400/410 Business Media Phones.

  Refer to UC Software 4.1.4 Release Notes for additional information.

The following features were introduced with UC Software 5.0.0:

- Support for HTML5 tag <input type="number"/>.
- Added ability to delete characters with the onscreen keyboard on VVX 500 and 600 or delete soft key onVVX-300 and 400.
- Added <audio> tag items and display of a volume bar when audio is playing.

  Refer to UC Software 5.0.0 Administrators' Guide for additional information.

3: Getting to Know the XML API Application Interface

The XML API provides you with flexibility in developing applications on Polycom phones while you securely integrate into the phone's capabilities and functions. The XML API features are supported by the microbrowser and browser, except where noted.

This chapter covers the following topics:

Notes on API Security

The following should be noted about the XML API security:

- **Authenticating remote control and monitoring** The execution of each HTTP GET/POST request requires an MD5 digest authentication. The execution of each HTTP PUSH request supports MD5 digest authentication as well as TLS and HTTPS. All pushed URLs are relative URLs with the root specified in the applications.cfg configuration file.
- **Achieving confidentiality of executed content** The phone's HTTP client supports Transport Layer Security (TLS), so any data retrieved from the URL can be protected. Make sure of the confidentiality of all traffic past the initial push request by specifying a root URL that uses HTTPS.
- **Event reporting** You can protect the confidentiality of all events reported by the phone with TLS similar to push content.
- **Direct data push** When direct data push is enabled—disabled by default— small amounts of content (1KB) can be sent directly to the phone by the application server. The request will still be authenticated through HTTP digest, but all content will be in clear text on the network. Polycom recommends that you only use unencrypted data push for broadcast type alerts that do not pose any confidentiality risks.

Both `apps.push.username` and `apps.push.password` must be set for data push to be enabled.

Using Telephone Integration URIs

Internal Uniform Resource Identifiers (URIs) allow the interface to execute a set of predefined actions on the phone. These actions are similar to the manual execution of key presses on the phone.

The following are ways to execute an internal URI action:

- If the file sent to the phone contains only internal URI actions, which you can send as Data Push, ensure the file content type is `application/x-com-polycom-spipx`. Each internal URI action must be separated by a newline and the execution is performed in order from the top down.
- If an XHTML file includes internal URI, you need to define and execute the file's anchor tags in the `HREF` attribute. For example, `<a href="Key:Setup">Menu</a>`. When you select the anchor, the action is processed and executed.
- Using the PolyUri() JavaScript function, apply one of the following soft key actions in anchor tags:
  - `SoftKey:Home`
  - `SoftKey:Back`
  - `SoftKey:Exit`
  - `SoftKey:Cancel`
  - `SoftKey:Refresh`

  See also Configuring Programmable Soft Keys.

Note: Executing Internal URIs

Internal URI actions contained in a file with content type `application/x-com-polycom-spipx` can be executed only through a URL push.

Use the format `ActionType:Action` when configuring the internal URIs where:

- `ActionType` is a type of key or action to execute: `Key`, `Soft key`, `Tel`, or `Play`.
- Action is the name of the action to be executed.

The supported internal URIs are described in the following table.

Table 6: Supported Internal URIs

Action Type	*Action*
Key	**Line1 to Number of Lines supported by the Phone model., DialPad0 to DialPad9, Softkey1 to Softkey5, DialPadStar, DialPadPound, VolDown, VolUp, Headset, Handsfree, MicMute, Menu, Messages, Applications, Directories, Setup, ArrowUp, ArrowDown, ArrowLeft, ArrowRight, Backspace, DoNotDisturb, Select, Conference, Transfer, Redial, Hold, Status, Call List**
The Key URIs send the key press event to the phone. The phone processes this event as if the button had been physically pressed.	
SoftKey[1]	**Back, Cancel, Exit, Home, Refresh, Reset, Submit, Fetch**
The Soft Key URIs send the soft key press event to the phone. The phone processes this event as if the associated soft key had been physically pressed. These URIs function when the interactive microbrowser is on the screen.	

Action Type	*Action*
Tel[23]	**Number;LineIndex;ext=<extension_no>;postd=<DTMF>**
The Tel URI initiates a new call to the specified number on the specified line. The line number is optional. If the line number is not supplied, then the first available line is used. The digit map rules are followed (see Using the Local Digit Map in the Polycom UC Software 5.0.0 Administrators' Guide).	
Play[4]	**Play:<audiofile_path>**
Download and play the audio file. The supported audio formats are G.711mu-law, G.711a-law, and Liner16. The <audiofile_path> is the relative path on the application server, relative to `apps.push.serverRootURL`. The supported maximum file size is determined by `res.finder.sizeLimit`. For G.711mu-law and G.711a-law files, sample rate must be 8ksps with a sample size of 8. This is supported on all phones. For Liner16 files: Sample size must 16 for all sample rates. 16 bit PCM at 16 kHz sample rate. Sample rate of 16ksps is supported on SoundPoint IP 321/331/335, 450, 550, 560, 650, and 670, SoundStation IP 5000, 6000 and 7000, VVX 500, VVX600, VVX400, VVX410, VVX300, VVX310 and 1500. Sample rate of 32ksps and 48 ksps is supported on SoundStation IP 5000, 6000 and 7000, VVX 1500. Sample rate of 8ksps and 44.1 ksps is supported on VVX 500 and 1500.	
Action	**UpdateConfig**
Update the phone's configuration. This action works the same as selecting **Menu > Settings > Basic > Update Configuration**. Depending on which configuration parameters have changed, the phone reboots.	

1 The programmable soft key related URIs are not supported on the browser on the Polycom VVX phones.

2 The LineIndex value is case insensitive. The range of LineIndex is 'Line1' to 'Line48'.

3 If the line corresponding to the LineIndex in the Tel action is busy, the existing call on that line is held and a call is placed to the number specified in the Tel URI on that given line.

4 An error is logged in a log file, if the file is too large to play.

Keep in mind the following important notes regarding internal URIs:

- The action name and key type are case sensitive.
- For non-XHTML content containing only internal URIs, the internal URIs are executed in the order they appear in the file without any delay.
- If any URI is invalid and is in a file of only internal URIs, the entire file is rejected.

- If any invalid URI is present in a XHTML file, the execution of that URI is ignored.

  For example, Table 7 shows the code created for a link that behaves as if you pressed the Do Not Disturb key:

Table 7: Sample Code - Simulate Pressing of Do Not Disturb Key

```
<html>
    <body> <br/>
    Click on the link to engage the DND feature
    <a href="Key:DoNotDisturb">DNDSettings</a>
    </body>
    <softkey index="1" label="Back" action="SoftKey:Back" />
    <softkey index="2" label="Exit" action="SoftKey:Exit" />
</html>
```

For example, to place a call to ***50**, and then wait two seconds before entering **44**:

Table 8: Sample Code – Pauses in Calls

```
<html>
    <head>
    </head>
    <body>
        <a href="tel:*50;postd=,44">Push to Talk</a>
    </body>
</html>
```

Note: How to Indicate Pauses

A two-second pause is indicated by the comma. A one-second pause is indicated by a **p** character. The dual-tone multi-frequency (DTMF) is sent after the placed call is connected. You can use a combination of both the comma and letter “p” for any amount of time you want to use. These combinations can also be used in between the digits. For example, `postd=pp1,12p8765`.

Using Push Requests

A push request is an XML formatted request you send to a phone to tell it to process the XML content. The phone may render the data, fetch a URL, or perform an action.

See also Configuring Push Request Parameters.

HTTP URL Push

The HTTP URL push enables an application to push a URL to a phone to open its microbrowser or browser, as for example, an HTML Web page. The URL value sent within the push request is relative to the URL configured by the `apps.push.serverRootURL` configuration parameter. The pushed URL is appended to this root URL, and the microbrowser or browser attempts to open the pushed URL. This feature is asynchronous; and once the push request is received by the phone, it returns a 2xx or 4xx response immediately. There is no success/failure feedback for the push handling, and the pushing application does not know if the microbrowser or browser was able to open the pushed URL.

Use the following format when configuring the HTTP URL Push:

```
<URL priority=”X” >URI path</URL>
```

When pushing data to a phone, make sure to send the request to:

```
http://<Phone IP>/push
```

See Configuring Push Request Parameters.

The URL push requests support the attributes listed in the following table.

Table 9: URL Push Request Attributes

Attribute	*Permitted Values*
priority[1]	**Critical, Important, High, Normal**
Sets the priority of the push, which determines how and when the URL is requested. For more information, refer to Table 10: How Priority Affects URL Push Requests.	
URI path[2]	**String**
Any relative URI (or relative URI path) on the configured application server.	

[1] If attribute is absent, **Normal** is used.

[2] Currently multiple URIs in a single push request are not supported.

Note: Defining PolycomIPPhone Tags

The `<URL>` tag must be defined under a `<PolycomIPPhone>` root tag. For example:

```
<PolycomIPPhone>
    <URL priority=”Normal”>/examples/media.xhtml</URL>
</PolycomIPPhone>
```

The following table describes the results of using a specific priority when the phone is in different states.

Table 10: How Priority Affects URL Push Requests

Phone State	*Priority*	*Description*
Idle State	Critical	The phone will display push request immediately.
	High	The phone will display push request immediately.

Phone State	*Priority*	*Description*
	Important	The phone will display push request immediately.
	Normal	The phone will display push request immediately.
Non-Idle State	Critical	The phone will display push request immediately.
	High	The phone will display push request immediately, but it will appear after critical push requests. The phone will check whether the last processed message is of the same or a higher priority. If it is the same or a higher priority, then the phone will wait until the phone returns to the idle state before displaying the message.
	Important	The phone will display push request immediately, but it will appear after critical and high push requests. The phone will check whether the last processed message is of the same or a higher priority. If it is the same or a higher priority, then the phone will wait until the phone returns to the idle state before displaying the message.
	Normal	The phone will ignore push request in push queue until the phone returns to the idle state.

Keep in mind the following important notes regarding HTTP URI push:

- The URI that the phone fetches is a concatenation of the `apps.push.serverRootURL` and the URI sent in the Push URL message.
- By default, a `Back` soft key is placed on the graphic display.
- The `Back` soft key will not appear when `mb.main.autoBackKey` is set to 0 or when custom soft keys are shown using JavaScript; otherwise, it will appear.
- Push requests are displayed as first-in-first-out except for when noted in Table 66: Supported Actions in <softkey> Tag.
- All HTTP requests are challenged through HTTP Digest Authentication.
- If the phone cannot fetch the content from the pushed URI, the request is ignored.

For example, if `apps.push.serverRootURL` is configured in a phone to be `http://1.2.3.4/apps`, to push the display of a XHTML page `media.xhtml`, you would send the following XHTML:

Table 11: XHTML Code

```
<PolycomIPPhone>
    <URL priority=”Normal”>/examples/media.xhtml</URL>
</PolycomIPPhone>
```

where `media.xhtml` is hosted by a Web server at `http://1.2.3.4/apps/examples/media.xhtml`.

HTML Data Push

The data push enables you to send XHTML page content directly to a phone without the overhead of the phone having to fetch the XHTML.

Use the following format when sending the HTML Data Push:

```
<Data priority=”X” >Y</Data>
```

When pushing data to a phone, make sure to send the request to:

```
http://<Phone IP>/push
```

The attributes listed in the following table are supported by HTML push requests.

Table 12: HTML Push Requests

Attribute	*Permitted Values*
priority[1]	**Critical, Important, High, Normal**
Sets the priority of the push, which determines how and when the URL is requested. For more information on how priority affects HTML push requests, see the following table.	
text[2]	**Text in HTML format**
Any text.	

[1] If attribute is absent, **Normal** is used.

[2] The maximum content length for push request is 2KB.

Note: Where Tags Are Defined

The `<Data>` tag must be defined under a `<PolycomIPPhone>` root tag. For example:

```
<PolycomIPPhone>
    <Data priority=”Important”> <h1> Fire Drill at 2pm </h1> Please exit and congregate at your appropriate location outside </Data>
</PolycomIPPhone>
```

The following table describes the results of using a specific priority.

Table 13: How Priority Affects HTML Push Requests

Phone State	*Priority*	*Description*
Idle State	Critical	The phone will display push request immediately.
	High	The phone will display push request immediately.
	Important	The phone will display push request immediately.
	Normal	The phone will display push request immediately.
Non-Idle State	Critical	The phone will display push request immediately.

Phone State	*Priority*	*Description*
	High	The phone will display push request immediately, but it will appear after critical push requests. The phone will check whether the last processed message is of the same or a higher priority. If it is the same or a higher priority, then the phone will wait until the phone returns to the idle state before displaying the message.
	Important	The phone will display push request immediately, but it will appear after critical and high push requests. The phone will check whether the last processed message is of the same or a higher priority. If it is the same or a higher priority, then the phone will wait until the phone returns to the idle state before displaying the message.
	Normal	The phone will ignore push request in push queue until the phone returns to the idle state.

For example, to push the display of an important message:

Table 14: To Push the Display of an Important Message

```
<PolycomIPPhone>
    <Data priority=”Important”>
     <h1> Fire Drill at 2pm </h1>
     Please exit and congregate at your appropriate location outside
   </Data>
</PolycomIPPhone>
```

For example, to push the URIs for execution:

The content type must be `application/x-com-polycom-spipx`.

Table 15: To Push the URIs for Execution

```
<PolycomIPPhone>
    <Data priority=”Important”>
     Key:Line1
     Key:DialPad0
     Key:DialPad1
   </Data>
</PolycomIPPhone>
```

Using Telephony Notification Events

Telephony events allow application programs insight into how the phones operate. Using a combination of telephony events enables an application to detect the starting of the phones, the signing on of users, and the ongoing condition of the phones.

For example using a combination of events can enable an application to:

- Detect when a phone has registered through the Line Registration Event.
- Detect when multiple users log into a different phones each day through the Line Registration Event + User Login Event.

You can configure the phone to send information to a specific URI if one of the following telephony notification events occurs:

- Viewing an Incoming Call Event
- Viewing an Outgoing Call Event
- Viewing an Offhook Event
- Viewing an Onhook Event
- Viewing a Phone Lock Event
- Viewing a Phone Unlock Event
- Viewing a Call State Change Event
- Viewing a Line Registration Event
- Viewing a Line Unregistration Event
- Viewing a User Login/Logout Event

These events are XML data posted to a Web server by the phone.

For details on how to configure these events, see Configuring Telephone Event Notification Parameters.

The following table contains the attributes for all telephony notification events.

Table 16: Telephony Notification Attributes

Attribute	*Permitted Values*
Phone IP	**IP address**
IP address of the phone. For example, `172.24.128.160`.	
MACAddress	**MAC Address**
MAC address of the phone. For example, `0004f214b8e7`.	
CallingPartyName	**name**
The name displayed in the phone's From label in the screen. If the line is registered and the call is initiated from that line, then the registered line display name of the calling party is shown. If the line is not registered and the call is initiated from that line, the IP address of the calling party is shown. For example, `sip:172.24.128.160`.	

Attribute	*Permitted Values*
CallingPartyNumber	**number**
The number displayed on the phone. If the line is registered and the call is initiated from that line, the registered line number of the calling party is shown. If the line is not registered and the call is initiated using IP address from that line, the IP address of the calling party is shown.	
CalledPartyName	**name**
The name displayed in the phone's To label in the screen. If the call is received by a registered line, the registered line display name of the called party is shown. If the call is received on a non-registered line, the IP address of the called party is shown.	
CalledPartyNumber	**number**
The number displayed on the phone. If the call is received by a registered line, the registered line number of the called party is shown. If the call is received on a non- registered line, the IP address of the called party is shown.	
CalledPartyDirNum	**number**
If the line is registered, the value is the registered line number. If the line is not registered, the value is the IP address of the called party.	
CallingPartyName	**name**
If the line is registered, the value is the registered line display name. If the line is not registered, the value is the IP address of the calling party.	
CallingPartyDirNum	**number**
If the line is registered, the value is the registered line number. If the line is not registered, the value is the IP address of the calling party.	
CallDuration	**number, seconds**
The duration of the call.	
Call State	**Outgoing call states: Dialtone, Setup, RingBack** **Incoming call states: Offering** **Outgoing/Incoming call states: Connected, CallConference, CallHold, CallHeld, CallConfHold, CallConfHeld** **Shared line states: CallRemoteActive**
The call state.	
CallType	**Incoming, Outgoing**
The call type.	
TimeStamp	**time**
The date and time the event occurred on the phone. For example, `2008-07-11T13:19:53-08:00`.	
LineNumber	**number**
Line that is registered. For example, ‘1’.	

Attribute	*Permitted Values*
State The call state.	**Outgoing call states: Dialtone, Setup, Ringback** **Incoming call states: Offering** **Outgoing/incoming call states: Connected, Hold, Held, Free, Disconnected** **Unknown call state: Ncas**
LineKeyNum Line that is registered. For example, ‘1’.	**number**
LineDirNum Phone number associated with the line. For example, ‘1234’.	**phone number**
LineState The line state.	**Active, Inactive**
UserLoggedIn Name of user logged in.	**name**
UserLoggedOut Name of user logged out. The value is always Null.	**name**
UIAppearanceIndex The call appearance index. The call appearance index for the active call is denoted by an asterisk (*) character suffix.	**string**

Viewing an Incoming Call Event

The Incoming Call Event is used by an application to send metadata about the call to the phone in real time, or to detect when the phone line is busy.

Use the following format when viewing the incoming call event:

Table 17: Incoming Call Event

```
<IncomingCallEvent>
    <PhoneIP> </PhoneIP>
    <MACAddress> </MACAddress>
    <CallingPartyName> </CallingPartyName>
    <CallingPartyNumber> </CallingPartyNumber>
    <CalledPartyName> </CalledPartyName>
    <CalledPartyNumber> </CalledPartyNumber>
    <TimeStamp> </TimeStamp>
</IncomingCallEvent>
```

The following example shows the transmitted data for a call between two registered lines when the telephone notification URI is set and the incoming call event is enabled to gather information:

Table 18: Incoming Call Event

```
<IncomingCallEvent>
   <PhoneIP>172.24.132.135</PhoneIP>
    <MACAddress>0004f214b89e</MACAddress>
    <CallingPartyName>20701</CallingPartyName>
    <CallingPartyNumber>20701@172.18.186.94</CallingPartyNumber>
    <CalledPartyName>20300</CalledPartyName>
    <CalledPartyNumber>20300</CalledPartyNumber>
    <TimeStamp>2008-07-11T13:19:53-08:00</TimeStamp>
</IncomingCallEvent>
```

Viewing an Outgoing Call Event

The Outgoing Call Event is used by an application to detect when a phone is busy in a call.

Use the following format when viewing the outgoing call event:

Table 19: Code Snippet for Outgoing Call Event

```
<OutgoingCallEvent>
    <PhoneIP> </PhoneIP>
    <MACAddress> </MACAddress>
    <CallingPartyName> </CallingPartyName>
    <CallingPartyNumber> </CallingPartyNumber>
    <CalledPartyName> </CalledPartyName>
    <CalledPartyNumber> </CalledPartyNumber>
    <TimeStamp> </TimeStamp>
</OutgoingCallEvent>
```

Viewing an Offhook Event

The Offhook Event allows an application to see when a call is starting.

Use the following format when viewing the offhook event:

Table 20 Code Snippet for Offhook Event

```
<OffHookEvent>
    <PhoneIP> </PhoneIP>
    <MACAddress> </MACAddress>
    <TimeStamp> </TimeStamp>
</OffHookEvent>
```

Viewing an Onhook Event

The Onhook Event notifies an application when a call has ended. For example, this can be used for call logging information.

Use the following format when viewing the onhook event:

Table 21: Onhook Event

```
<OnHookEvent>
    <PhoneIP> </PhoneIP>
    <MACAddress> </MACAddress>
    <TimeStamp> </TimeStamp>
</OnHookEvent>
```

Viewing a Phone Lock Event

The phone lock event notifies the application when the phone is locked. The application should not send many instructions to the phone when it is locked since most operations are ignored in a locked state.

Use the following format when viewing the phone lock event:

Table 22: Phone Lock Event

```
<PhoneLockedEvent>
   <PhoneIP> </PhoneIP>
   <MACAddress> </MACAddress>
   <TimeStamp> </TimeStamp>
</PhoneLockedEvent>
```

Viewing a Phone Unlock Event

The phone unlock event notifies the application when the phone is unlocked. Instructions sent to the phone in a locked state are ignored.

Use the following format when viewing the phone unlock event:

Table 23: Phone Unlock Event

```
<PhoneUnlockedEvent>
   <PhoneIP> </PhoneIP>
   <MACAddress> </MACAddress>
   <TimeStamp> </TimeStamp>
</PhoneUnlockedEvent>
```

Viewing a Call State Change Event

The Call State Change event notifies the application of the different call states that exist on the phone. The application can perform any call related operation upon receiving these events. For example, the application can transfer an incoming call to a desired number when it receives an incoming call event.

Use the following format when viewing the call state change event:

Table 24: Call State Change Event

```
<CallStateChangeEvent CallReference=" " CallState=" ">
    <PhoneIP> </PhoneIP>
    <MACAddress> </MACAddress>
    <LineNumber> </LineNumber>
    <TimeStamp> </TimeStamp>
   <CallLineInfo>
      <LineKeyNum> </LineKeyNum>
      <LineDirNum> </LineDirNum>
      <LineState> </LineState>
      <CallInfo>
         <CallState> </CallState>
         <CallType> </CallType>
         <UIAppearanceIndex> </UIAppearanceIndex>
         <CalledPartyName> </CalledPartyName>
         <CalledPartyDirNum> </CalledPartyDirNum>
         <CallingPartyName> </CallingPartyName>
         <CallingPartyDirNum> </CallingPartyDirNum>
         <CallReference> </CallReference>
         <CallDuration> </CallDuration>
      </CallInfo>
   </CallLineInfo>
</CallStateChangeEvent>
```

Viewing a Line Registration Event

The Line Registration Event occurs when a phone registers a line to a call server. This event is useful for flagging when the phone is on and running on the network.

Use the following format when viewing the line registration event:

Table 25: Line Registration Event

```
<LineRegistrationEvent>
    <PhoneIP> </PhoneIP>
    <MACAddress </MACAddress>
    <LineNumber> </LineNumber>
    <TimeStamp> </TimeStamp>
</LineRegistrationEvent>
```

Viewing a Line Unregistration Event

The line unregistration event is useful for determining when a phone is powered off or otherwise no longer available on the network. However, the event only occurs if the phone is gracefully shutdown or restarted. If the phone experiences a power loss, the event will not occur.

Use the following format when viewing the line unregistration event:

Table 26: Line Unregistration Event

```
<LineUnregistrationEvent>
    <PhoneIP> </PhoneIP>
    <MACAddress> </MACAddress>
    <LineNumber> </LineNumber>
    <TimeStamp> </TimeStamp>
    </LineUnregistrationEvent>
```

Viewing a User Login/Logout Event

The UserLogin/Logout Event is used to detect when a profile is used to sign into or out of a phone.

Use the following format when viewing the user login/logout event:

Table 27: User Login/Logout Event

```
<UserLoginOutEvent>
    <PhoneIP> </PhoneIP>
    <MACAddress> </MACAddress>
    <CallLineInfo>
        <LineKeyNum> </LineKeyNum>
        <LineDirNum> </LineDirNum>
    </CallLineInfo>
    <UserLoggedIn> </UserLoggedIn>
    <TimeStamp> </TimeStamp>
</UserLoginOutEvent>

<UserLoginOutEvent>
    <PhoneIP> </PhoneIP>
    <MACAddress> </MACAddress>
    <CallLineInfo>
        <LineKeyNum> </LineKeyNum>
        <LineDirNum> </LineDirNum>
    </CallLineInfo>
    <UserLoggedOut/>
    <TimeStamp> </TimeStamp>
</UserLoginOutEvent>
```

Note: How Event Values are Generated

The `LineKeyNum` and `LineDirNum` fields are generated for each registered line on the phone. The `UserLoggedIn` field is generated when a user logs into a phone. The `UserLogedOut` field is generated when a user logs out of a phone.

Using Phone State Polling

You can configure the phone to send the current state information to a specific URI or to the requestor upon receipt of an HTTP request. The following types of information can be sent to an URI:

- **Receiving Call Line Information** The line registration and call state is sent upon receipt of an HTTP request to the call state handler: `http://<Phone_IP>/polling/callstateHandler`.
- **Receiving Device Information** Device- specific information is sent upon receipt of an HTTP request to the device handler: `http://<Phone_IP>/polling/deviceHandler`.
- **Receiving Network Configuration** Network-specific information is sent upon receipt of an HTTP request to the network handler: `http://<Phone_IP>/polling/networkHandler`.

Two HTTP transactions occur:

- The application sends an HTTP request to a particular handler in the phone.
- The Phone posts the state, in XML format, to a preconfigured Web server.

See Configuring Phone State Polling Parameters for the parameters that control this feature.

Receiving Call Line Information

The Receiving Call Line Information is useful for providing additional information about the caller, such as information available through a contact management system.

The Call Line Information message is returned in the following format:

Table 28: Call Line Information

```
<CallLineInfo>
    <LineKeyNum> </LineKeyNum>
    <LineDirNum> </LineDirNum>
    <LineState>Active</LineState>
    <CallInfo>
        <CallState> </CallState>
        <CallType> </CallType>
        <UIAppearanceIndex> </UIAppearanceIndex>
        <CalledPartyName> </CalledPartyName>
        <CalledPartyDirNum> </CalledPartyDirNum>
        <CallingPartyName> </CallingPartyName>
        <CallingPartyDirNum> </CallingPartyDirNum>
        <CallReference> </CallReference>
        <CallDuration> </CallDuration>
    </CallInfo>
</CallLineInfo>
```

Note: When the Call Information Block is Defined

The `<CallInfo>` block is included if `<LineState>` is ‘Active’. Otherwise, it is not included.

The call line information message contains the attributes listed in the following table.

Table 29: Call Line Information Message Attributes

Attribute	*Permitted Values*
LineKeyNum	**number**
The line that is registered. For example, ‘1’.	

Attribute	*Permitted Values*
LineDirNum	**phone number**
The phone number associated with the line. For example, ‘1234’.	
LineState	**Active, Inactive**
The line state.	
CallState	**Outgoing call states: Dialtone, Setup, Ringback** **Incoming call states: Offering** **Outgoing/incoming call states: Connected, CallConference, CallHold, CallHeld, CallConfHold, CallConfHeld** **Shared line states: CallRemoteActive**
The call State.	
CallType	**Incoming, Outgoing**
The call type.	
UIAppearanceIndex	**string**
The call appearance index. The call appearance index for the active call is denoted by an asterisk (*) character suffix.	
CallingPartyName	**name**
If the line is registered, the value is the registered line display name. If the line is not registered, the value is the IP address of the calling party.	
CallingPartyDirNum	**number**
If the line is registered, the value is the registered line number. If the line is not registered, the value is the IP address of the calling party.	
CalledPartyName	**name**
If the line is registered, the value is the registered line display name. For example ‘45343’. If the line is not registered, the value is the IP address of the called party. For example `10.243.1.32`.	
CalledPartyDirNum	**number**
If the line is registered, the value is the registered line number. For example ‘45344’. If the line is not registered, the value is the IP address of the called party. For example `10.243.1.32`.	
CallDuration	**number, seconds**
The duration of the call in seconds.	

Receiving Device Information

Applications use the Device Information for tasks like device firmware tracking/management or asset tracking.

The Device Information message is returned in the following format:

Table 30: Device Information

```
<DeviceInformation>
    <MACAddress> </MACAddress>
    <PhoneDN> </PhoneDN>
    <AppLoadID> </AppLoadID>
    <BootROMID> </BootROMID>
    <ModelNumber> </ModelNumber>
    <TimeStamp> </TimeStamp>
</DeviceInformation>
```

The device information message contains the attributes listed in the following table.

Table 31: Device Information Message Attributes

Attribute	*Permitted Values*
MACAddress	**MAC Address**
The MAC address of the phone. For example, `0004f214b8e7`.	
PhoneDN	**string**
A list of all registered lines, including expansion modules, and their directory numbers delimited by commas. For example, `Line1:6744,Line2:4534,Line3:4534`.	
AppLoadID	**string**
The application load ID on the phone. For example, ‘4.0.1.18754 27-Feb-11 20:07’.	
BootROMID	**string**
The BootROM on the phone. For example, ‘5.0.0.11646’.	
ModelNumber	**string**
The phone’s model number. For example, ‘SoundPoint IP 650’.	
TimeStamp	**time**
The date and time that the event occurred on the phone. For example, ‘2008-07-11T13:19:53-08:00’.	

Receiving Network Configuration

The Network Configuration message returns network information about the phone.

The Network Configuration message is returned in the following format:

Table 32: Network Configuration

```
<NetworkConfiguration>
    <DHCPServer> </DHCPServer>
    <MACAddress> </MACAddress>
    <DNSSuffix> </DNSSuffix>
    <IPAddress> </IPAddress>
    <SubnetMask> </SubnetMask>
    <ProvServer> </ProvServer>
    <DefaultRouter> </DefaultRouter>
    <DNSServer1> </DNSServer1>
    <DNSServer2> </DNSServer2>
    <VLANID> </VLANID>
    <DHCPEnabled> </DHCPEnabled>
</NetworkConfiguration>
```

The network configuration message contains the attributes listed in the following table.

Table 33: Network Configuration Message Attributes

Attribute	*Permitted Values*
DHCPServer	**IP address**
The DHCP server IP address. For example, ‘192.168.1.1’.	
MACAddress	**MAC Address**
The MAC address of the phone. For example, ‘0004f214b8e7’.	
DNSSuffix	**host name**
The DNS domain suffix. For example ‘polycom.com’.	
IPAddress	**IP address**
The IP address of the phone. For example ‘192.168.1.5’.	
SubnetMask	**IP address**
The subnet mask: For example ‘255.255.255.0’.	
ProvServer	**IP address**
The IP address of the provisioning server or a host name. For example ‘192.168.1.10’.	
DefaultRouter	**IP address**
The IP address of the default router or IP gateway. For example ‘192.168.1.1’.	
DNSServer1	**IP address**
The configured IP address of DNS Server 1. For example ‘192.168.1.250’.	

Attribute	*Permitted Values*
DNSServer2 The configured IP address of DNS Server 2. For example ‘192.168.1.250’.	**IP address**
VLANID The phone’s 802.1Q VLAN identifier. For example ‘45’.	**Null, 0 to 4094**
DHCPEnabled If DHCP is enabled and set to ‘Yes’.	**Yes, No**

4: Writing Your Web Application

This chapter provides you with information you need to know while writing your Web application.

The topics in this chapter include:

- Developing Your Browser Application
- Developing Microbrowser-Specific Applications

Web Info: Polycom Developer Community

Polycom has an active Developer Community Forum where you can find more sample codes as well as answers to many common developer questions.

Developing Your Browser Application

You need to ensure you develop a Web application that runs on the browser available on VVX® phones. You can see which phones support the browser in Table 2: Polycom Phones that Support the Browser and Microbrowser.

This section contains information on:

- Supporting HTTP
- Launching the Browser from VVX Phones
- Navigating and Form Editing on the Main Browser
- Viewing the Idle Browser
- Using Browser JavaScript DOM Extensions

Refer to What is the Browser to view the standards supported in the browser.

Supporting HTTP

The browser is a fully compliant HTTP/1.1 user agent as described in RFC 2616.

The browser supports:

- **Cookies** Cookies are stored in the flash file system; they are preserved when the phone reboots or is reconfigured. Cookies are shared between the idle display browser, if available, and the main browser.
- **Refresh headers**
- **HTTP proxies**
- **HTTP proxy authentication** The phone supports login credentials that authenticate your phone to the server.

- **HTTPS by HTTP over TLS** The browser supports the TLS protocol v1 only. It is not backward compatible with SSL v2 or SSL v3.
- **Custom CA certificates**

Web Info: To View Trusted Certificate Authorities

For more information on CA certificates, see Technical Bulletin 17877: Using Custom Certificates With SoundPoint IP Phones.

Launching the Browser from VVX Phones

By default, the main browser loads the home page, which is specified in the `mb.main.home` configuration parameter, once you press the **Applications** icon on the Home screen on the VVX phones. Alternatively, you can launch the browser by pressing the **Applications** key on the VVX 1500 phone.

You can return to the browser from an application by pressing the **Applications** key or selecting the **Applications** icon. Even though you cannot see the browser application, it is still active. Pending transactions occur in the background and become visible when the browser is brought to the foreground. Examples of the main browser on VVX phones are shown in Figure 1: VVX 1500 Main Browser and Figure 2: VVX 500 Main Browser. The main browser on the VVX 500 and VVX 1500 phones runs in full screen mode only.

Figure 1: VVX 1500 Main Browser

Figure 2: VVX 500 Main Browser

Navigating and Form Editing on the Main Browser

Navigation and Form Editing is slightly different on each phone. The different keyboard keys and the presence or absence of a touch-screen display affects the Navigation and Form Editing. However, you can navigate the browser as you would in any major Web browser.

Note: Error Indications in Browser

Network-related errors, such as HTTP 404, will be handled in the page body as in any major Web browser. JavaScript errors will be indicated by an icon in the title bar, but not accompanied by any text messages.

You can use the navigation keys on the non-touch sensitive phones to scroll the Web page in each direction. For the touch sensitive phones, the on-screen navigation performs in the same manner. You can swipe the screen to scroll the Web page in each direction.

The following buttons display on the toolbar on the browser:

- **Home**
- **Stop/Refresh** depending on whether the page has completely downloaded yet
- **Keyboard pop-up** when focus is on an input widget
- **Navigation** Up, down, left, and right buttons appear only if scrolling is available in those directions

  Holding down the navigation keys on the phone speeds up scrolling.
- **Exit**
- **Encoding** Ascii, Cyrillic, Katakana, Latin, and Unicode
- **Text entry mode** 123, ABC, abc, and Abc
- **Left and Right Tab** Only displayed on VVX 300/310 and 400/410 non touch phones and navigates between the focusable elements in the webpage.

See Figure 1: VVX 1500 Main Browser and Figure 2: VVX 500 Main Browser for an example of the toolbar and buttons on each phone.

Form editing in the browser behaves the same as any major Web browser. When an input field is invoked, the keyboard displays at the bottom of the screen, and the input field is centered at the top of the screen. The keyboard is removed from the screen once you click on the screen. This enables you to click the Submit button next to the entry field without closing the keyboard widget.

Viewing the Idle Browser

The idle display browser functions independently of the main browser and is capable of rendering the same content. The idle browser's home page is configured via the `mb.idleDisplay.home` configuration parameter. The idle display browser does not accept user input and only displays when there are no phone calls in progress and the phone is in the idle user interface.

Examples of the Idle browser on VVX phones are shown in Figure 3: VVX 1500 Idle Browser and Figure 4: VVX 500 Idle Browser.

Note: Line Activity Can't be Viewed

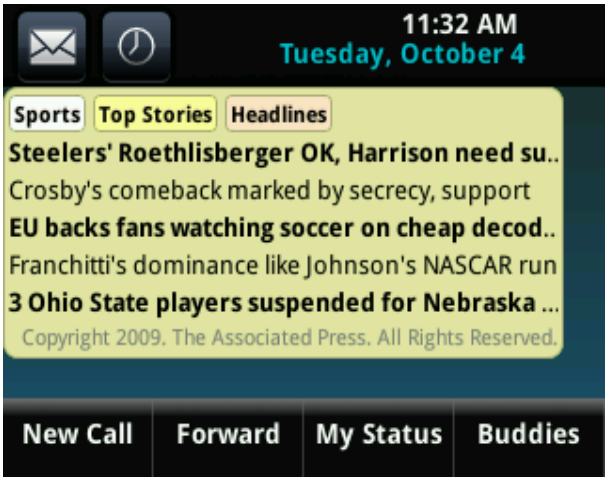
The idle browser doesn't display line activity on VVX 300, 400, 500 and 600 phones.

Note: Viewing Idle Browser on VVX 300 and VVX 400

In addition to configuring the idle browser URL, you need to configure the Screen Saver Type (`up.screenSaver.type=2`) parameter for the idle browser.

Figure 3: VVX 1500 Idle Browser

Figure 4: VVX 500 Idle Browser



Using Browser JavaScript DOM Extensions

The browser provides access to phone-specific Document Object Model (DOM) JavaScript extensions. The DOM is created by the browser after parsing an XHTML file. JavaScript's primary role in the browser is to modify properties of the DOM. The DOM is a collection of every object defined in the XHTML including every button, label, and image. A Web application uses JavaScript to modify DOM properties similarly to other XHTML objects.

For more JavaScript examples, see JavaScript Examples for the Browser.

This section provides information on the following custom DOM extensions:

- PolySoftKey
- PolyUri
- Sample Browser Web Applications

PolySoftKey

The PolySoftKey DOM object provides control over the soft keys in the browser. You can use the PolySoftKey DOM object to hide or show the default or custom defined soft keys as well as respond to pressed soft keys.

Note: Soft key Support

Soft keys are only supported on the microbrowser. If you are making an application for the browser on VVX phones, use HTML input type='button'. See Control of Soft Keys.

The JavaScript PolySoftKey.* custom DOM extensions are as follows:

- **PolySoftKey.customSoftkeyEvent.connect({function})** Connects the JavaScript function supplied to the callback when a custom soft key is pressed. Refer to Table 34: PolySoftKey DOM Extension.

- **PolySoftKey.setSoftkeyLabel(int, string)** Sets the label of a given custom soft key (0 to 3).
- **PolySoftKey.hideToolbar()** Enables the application to hide the soft key toolbar.
- **PolySoftKey.showToolbar()** Returns the soft key toolbar.
- **PolySoftKey.getSoftkeyPoint(int)** Returns a JavaScriptON object with the X, Y coordinates of the soft key. When used in combination with `hideToolBar()`, you can replace hard key button events with any type of HTML object.
- **PolySoftKey.resetAllDefaults()** Clears all custom defined key labels.
- **PolySoftKey.resetDefaultKey(int)** Clears custom key label (0 to 3).

The PolySoftKey custom DOM extension example is shown next.

Table 34: PolySoftKey DOM Extension

```
PolySoftKey.customSoftkeyEvent.connect(skCallBack);

PolySoftKey.setSoftkeyLabel(0, "one");
PolySoftKey.setSoftkeyLabel(1, "Two");
PolySoftKey.setSoftkeyLabel(2, "Three");
PolySoftKey.setSoftkeyLabel(3, "Four");

function skCallBack(key, skEvent){
   switch(key){
   case 0:
      document.getElementById("eventStuff").innerHTML = "SK 1 was pressed";
      break;
   case 1:
      document.getElementById("eventStuff").innerHTML = "SK 2 was pressed";
      break;
   case 2:
      document.getElementById("eventStuff").innerHTML = "SK 3 was pressed";
      break;
   case 3:
      document.getElementById("eventStuff").innerHTML = "SK 4 was pressed";
      break;
   }
   document.getElementById("eventValue").innerHTML = skEvent;
}

// hide the tool bar
function hideSKs(){
   PolySoftKey.hideToolBar();
}

// show the tool bar
function showSKs(){
   PolySoftKey.showToolBar();
}
```

```
// get the styled points of the SKs so app can add whatever object they
// want to that area after calling hideToolBar()
function getSKPoints(){
   // Returns a JSON object with two properties, X & Y.  To convert to
   // JS object you must use the eval function on the JSON object.
   var one = PolySoftKey.getSoftkeyPoint(0);
   var oneObj = eval('(' + one + ')'); //to help avoid syntax errors,
                                       //wrap with '(' ')' chars

   var two = PolySoftKey.getSoftkeyPoint(1);
   var twoObj = eval('(' + two + ')');

   var three = PolySoftKey.getSoftkeyPoint(2);
   var threeObj = eval('(' + three + ')');

   var four = PolySoftKey.getSoftkeyPoint(3);
   var fourObj = eval('(' + four + ')');

   document.getElementById("points").innerHTML = oneObj.X + ":" + oneObj.Y + "," +
twoObj.X + ":" + twoObj.Y + "," + threeObj.X + ":" + threeObj.Y + "," + fourObj.X +
":" + fourObj.Y;
}
```

PolyUri

The PolyUri custom DOM extension gives you a few general controls/notifications, such as a notification when the browser is hidden or shown as opposed to other applications on the phone. It also enables you to push a URI back to the phone—in a loopback fashion—from a loaded Web page. See Using Telephone Integration URIs.

The JavaScript PolyUri.* custom DOM extensions are as follows:

- **PolyUri.pushUri(string)** Enables you to push any Polycom internal URI. For example, Play:: and Tel::.
- **PolyUri.shownSig.connect({function})** Connects the JavaScript function supplied to the callback when the browser is visible. For example, after being resumed from the call appearance list.
- **PolyUri.hiddenSig.connect({function})** Connects the JavaScript function supplied to the callback when the browser is not visible. For example, when put on hold in the call appearance list.

The PolyUri custom DOM extension example is shown next.

Table 35: PolyUri DOM Extension

```
PolyUri.shownSig.connect(appShown);
PolyUri.hiddenSig.connect(appHidden);

function appShown(){
// Pushes a play request whenever the browser is shown
PolyUri.pushUri("play:http://172.24.134.28:8080/sdk/demos/sounds/DingLing.wav");
}
function appHidden(){
}
```

Sample Browser Web Applications

This section presents a sample application in attachment **mip3.zip** that you can use as a starting point for writing your Web application. Additional applications can be found in attachment **app_examples.zip**.

Developing Microbrowser-Specific Applications

When designing your application, consider developing a Web application that runs on the microbrowser available on SoundPoint® IP and SoundStation® IP phones. See Table 2: Polycom Phones that Support the Browser and Microbrowser.

This section contains information on:

- Supporting XHTML Elements
- Supporting HTTP
- Launching the Microbrowser from the Phone
- Navigating and Form Editing Behavior on the Main Browser
- Viewing the Idle Browser
- Sample Microbrowser Web Applications

Supporting XHTML Elements

The microbrowser supports a subset of XHTML elements which are derived from HTML 4.01.

The supported elements and attributes are:

- Basic Tags
- Link Tags
- Input Tags
- Image Tags

- Table Tags
- Meta Information Tags
- Audio Tags
- Ordered and Unordered List Tags

Unsupported elements and attributes are described in Unsupported XHTML elements on the Microbrowser.

Basic Tags

The following basic tags are supported:

- <!DOCTYPE>—Defines the document type
- <!--…-->—Defines a comment

<!DOCTYPE>

Place the `<!DOCTYPE>` declaration first in your document before the `<html>` tag. This tag tells the browser which XHTML specification the document uses. XHTML 1.0 specifies three XML document types: Strict, Transitional, and Frameset.

XHTML 1.1 specifies one XML document type: Strict.

For example, `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">`

This tag does not support any attributes.

XHTML Strict

Use this DTD for a clean markup free of presentational clutter.

For example, `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">`

XHTML Transitional

Use this DTD for XHTML's presentational features.

For example, `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`

XHTML Frameset

Use this DTD for frames.

For example, `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">`

<!--…-->

Use the comment tag to insert a comment in the source code. A comment will be ignored by the browser. You can use comments to explain your code, which can help you when you edit the source code at a later date.

This tag does not support any attributes.

Link Tags

The following link tag is supported:

- <a>—Defines an anchor

<a>

Use the <a> tag to define an anchor. An anchor can be used to create a link to another document by using the HREF attribute or to create a bookmark inside a document by using the name or id attribute.

The <a> tag supports the attributes listed in the following table.

Table 36: <a> Tag Attributes

Attribute	*Value(s)*
href[1]	**URL**
The target URL of the link. For example, 'http://www.polycom.com'.	
name[2]	**section_name**
Names an anchor. Use this attribute to create a bookmark in a document.	

[1] The microbrowser supports both http:// and tel:// URL schemes as well as internal URIs. When a tel:// URL is selected, the phone switches to the telephony application and dials the number specified in the URL. Currently the number is dialed as-is; however, full support for tel:// URL parsing as specified in RFC 2806 will be available in a future release. sip:// URLs are not supported at this time.

[2] This attribute is parsed, but not used.

An example of the <a> tag is in the following table.

Table 37: <a> Tag Example

```
<a href="http://www.polycom.com" name="link1" />Link to Polycom</a>
```

Input Tags

The following input tags are supported:

- <form>—Defines a form
- <input>—Defines an input field

Note: Nesting Forms Are Supported

The microbrowser supports both the GET and POST methods for submitting forms. Nesting forms within tables is supported. However, nesting of one form tag within another is not supported and can lead to unexpected results.

<form>

Use the `<form>` tag to create a form for user input. A form can contain text fields, check boxes, radio buttons, etc. Forms are used to pass user data to a specified URL.

The `<form>` tag supports the attributes listed in the following table.

Table 38: *<form>* Tag Attributes

Attribute	*Value(s)*
action	**URL**
A URL that defines where to send the data when the submit button is pushed. For example, ‘http://www.google.com’.	
method[1]	**get, post**
The HTTP method for sending data to the action URL. The default is **get**. method=**get**: This method sends the form contents in the URL: `URL?name=value&name=value`. method=**post**: This method sends the form contents in the body of the request.	
name	**form_name**
Defines a unique name for the form.	

[1] If the form values contain non-ASCII characters or exceeds 100 characters, you must use method=**post**.

An example for the `<form>` tag is in the following table.

Table 39: <form> Tag Example

```
<form name=”appForm” method=”get” >
   Input elements …..
</form>
```

<input>

Use the `<input>` tag to define the start of an input field that allows you to enter data. In XHTML, the `<input>` tag must be properly closed.

The <input> tag supports the attributes listed in the following table.

Table 40: *<input>* Tag Attributes

Attribute	*Value(s)*
checked[1]	**checked**
Indicates that the input element should be checked when it first loads.	
type	**checkbox, hidden, password, radio, reset, submit, text**
Indicates the type of input element. The default value is **text**.	
Content	**alphanumeric, numericalpha, numeric**
Used with type=”text.” Alphanumeric sets the element’s input mode to accept alphabets and allows you to change the input mode to accept numeric values. Numericalpha sets the element’s input mode to accept numeric values and allows you to change the input mode to enter alphabets. Numeric sets the element’s input mode to accept only numeric values.	
name[2]	**field_name**
Defines a unique name for the input element.	
value[3]	**value**
For buttons, reset buttons, and submit buttons: Defines the text on the button. For image buttons: Defines the symbolic result of the field passed to a script. For checkboxes and radio buttons: Defines the result of the input element when clicked. The result is sent to the form's action URL. For hidden, password, and text fields: Defines the default value of the element.	

[1] Used with type=**checkbox** and type=**radio**

[2] This attribute is required with type=**checkbox**, type=**hidden**, type=**password**, type=**text**, and type=**radio**

[3] This attribute is required with type=**checkbox** and type=**radio**

Image Tags

The following image tag is supported:

- <img>—Defines an image

The microbrowser supports images stored in uncompressed .bmp or in .jpg format.

- While all BMP bit depths will be displayed to the best of the phone’s ability, Polycom recommends the image format most suitable for the target platform be chosen. For example:
  - The SoundPoint IP 601 LCD supports four levels of grey, so a 16-color BMP format would be most appropriate.
  - The SoundPoint IP 670 LCD supports 12-bit color.
- JPEG images are not supported on SoundPoint IP/SoundStation IP phones except for 321/331/335, 450, 550, 560, 650, and 670 desktop phones; SoundStation IP 5000, 6000, and 7000 conference phones; and SoundStation Duo conference phones.

You can scroll images up and down; however, images that are too wide will be truncated.

Various platforms have differing limits due to memory. There are also differing pixel limits for devices of differing pixel depth. A 1 bit per pixel image 160x80 requires only 1600 bytes. For a 24-bit picture, the memory requirement is 38400 bytes.

There are several limits depending on the source data. This involves the cache limits in configuration, and the display converted data, which is dependent on available RAM and is limited in the code depending on the platform.

The `<img>` tag defines an image.

Note: Certain Image Elements Are Not Support in XHTML 1.0

The `align`, `border`, `hspace`, and `vspace` attributes of the image element are not supported in XHTML 1.0 Strict DTD. The image is not scaled, up or down, when only one of **width** or **height** is used. However, scaling works when both are used together.

The `<img>` tag supports the attributes listed in the following table.

Table 41: *<img>* Tag Attributes

Attribute	*Value(s)*
src	**URL**
The URL of the image to display. For example, ‘http://www.topxml.com/images/topxml_site.gif’.	
height	**pixels or %**
Specifies the height of the image in pixel or percent. For example, 30.	
width	**pixels or %**
Specifies the width of the image in pixel or percent. For example, 30.	

An example for the `<img>` tag is shown next.

Table 42: <img> Tag Example

```
<img src="http://www.topxml.com/images/topxml_site.gif" height="30px" width="10%" />
```

Table Tags

The following table tags are supported:

- <table>—Defines a table
- <caption>—Defines a table caption
- <th>—Defines a table header
- <tr>—Defines a table row
- <td>—Defines a table cell
- <thead>—Defines a table header
- <tbody>—Defines a table body
- <tfoot>—Defines a table footer

Note: XHTML Table Must be Correctly Formatting For Proper Phone Display

XHTML tables must include `<tbody>` and `</tbody>` tags to be displayed properly on the phone.

<table>

The `<table>` tag defines a table. Inside a `<table>` tag you can put table headers, table rows, table cells, and other tables. The `align` and `bgcolor` attributes of the table element are not supported in XHTML 1.0 Strict DTD.

The `<table>` tag supports the attributes listed in the following table.

Table 43: *<table>* Tag Attributes

Attribute	*Value(s)*
align Aligns the table. Deprecated. Use **styles** instead.	**left, center, right**
border Specifies the border width. Set **border**=0 to display tables with no borders.	**pixels**
cellpadding Specifies the space between the cell walls and the contents.	**pixels or %**
cellspacing Specifies the space between the cells.	**pixels or %**
width Specifies the width of the table.	**pixels or %**

An example for the `<table>` tag is shown next.

Table 44: <table> Tag Example

```
{
<table align=”center” border=”1px” cellpadding=”1px” cellspacing=”1px” width=”100%”
/>
```

<caption>

This element defines a table caption. The `<caption>` tag must be inserted directly after the `<table>` tag. You can specify only one caption per table. The caption is usually centered above the table. The `align` attribute of the caption element is not supported in XHTML 1.0 Strict DTD.

The `<caption>` tag supports the attributes listed in the following table.

Table 45: <*caption*> Tag Attributes

Attribute	*Value(s)*
align Aligns the caption. Deprecated. Use **styles** instead.	**left, right, top, bottom**
id Defines a unique name for the map tag.	**unique_name**
class The class of the element.	**class_rule, style_rule**
title Text to display in a tool tip.	**tooltip_text**
style An inline style definition.	**style_definition**
dir Sets the text direction.	**ltr (left to right), rtl (right to left)**
lang Sets the language. For example, EN=English, DE=German.	**language_code**
xml:lang Sets the language. For example, EN=English, DE=German.	**language_code**

<th>

This tag defines a table header cell in a table. The text within the `<th>` element usually renders in bold. The `bgcolor`, `height`, `width`, and `nowrap` attributes of the `<th>` element are not supported in XHTML 1.0 Strict DTD.

The `<th>` tag supports the attributes listed in the following table.

Table 46: <*th*> Tag Attributes

Attribute	*Value(s)*
abbr	**abbr_text**
Specifies an abbreviated version of the content in a cell.	
align	**left, right, center, justify, char**
Specifies the horizontal alignment of cell content text.	
axis	**category_name**
Defines a name for a cell.	
bgcolor	**rgb(x,x,x), #xxxxxx, colorname**
Specifies the background color of the table cell. Deprecated. Use **styles** instead.	
char	**character**
Specifies which character to align text on. Used if **align**=char.	
charoff	**pixels, %**
Specifies the alignment offset to the first character to align on. Used if **align**=char.	
class	**class_rule, style_rule**
The class of the element.	
colspan	**number**
Indicates the number of columns this cell should span.	
dir	**ltr (left to right), rtl (right to left)**
Sets the text direction.	
headers	**header_cells_id**
A space-separated list of cell IDs that supply header information for the cell. This attribute allows text-only browsers to render the header information for a given cell.	
height	**pixels**
Specifies the height of the table cell. Deprecated. Use **styles** instead.	
id	**unique_name**
Defines a unique name for the map tag.	

Attribute	*Value(s)*
lang	**language_code**
Sets the language. For example, EN=English, DE=German.	
nowrap	**nowrap**
Whether to disable or enable automatic text wrapping in this cell. Deprecated. Use **styles** instead.	
rowspan	**number**
Indicates the number of rows this cell should span.	
title	**tooltip_text**
Text to display in a tool tip.	
scope	**col, colgroup, row, rowgroup**
Specifies if this cell provides header information for the rest of the row that contains it (row), or for the rest of the column (col), or for the rest of the row group that contains it (rowgroup), or for the rest of the column group that contains it.	
style	**style_definition**
An inline style definition.	
valign	**top, middle, bottom, baseline**
Specifies the vertical alignment of the cell content.	
width	**pixels, %**
Specifies the width of the table cell in pixels or a percentage. Deprecated. Use **styles** instead.	
xml:lang	**language_code**
Sets the language. For example, EN=English, DE=German.	

<tr>

This tag defines a row in a table. The `bgcolor` attribute of the `<tr>` element are not supported in XHTML 1.0 Strict DTD.

The `<tr>` tag supports the attributes listed in the following table.

Table 47: *<tr>* Tag Attributes

Attribute	*Value(s)*
align	**left, right, center, justify, char**
Specifies the horizontal alignment of cell content text.	

<td>

This tag defines a cell in a table. The `bgcolor`, `height`, `width`, and `nowrap` attributes of the `<td>` element are not supported in XHTML 1.0 Strict DTD.

The `<td>` tag supports the attributes listed in the following table.

Table 48: *<td>* Tag Attributes

Attribute	*Value(s)*
align Specifies the horizontal alignment of cell content text.	**left, right, center, justify, char**
colspan Indicates the number of columns this cell should span.	**number**
rowspan Indicates the number of rows this cell should span.	**number**

<thead>

This tag defines a table header. The `<thead>`, `<tfoot>` and `<tbody>` tags enable you to group rows in a table. When you create a table, you want to have a header row, rows with data, and a row with totals. This division enables browsers to support scrolling of table bodies independently of the table header and footer. When long tables are printed, the table header and footer information repeats on each page that contains table data.

Note: Table Tag Order Must Be Followed

The `<thead>` tag must have a `<tr>` tag inside. If you use the `<thead>`, `<tfoot>` and `<tbody>` tags, you must use every element. They should appear in this order: `<thead>`, `<tfoot>` and `<tbody>`, so browsers can render the footer before receiving all the data. You must use these tags within the table element.

The `<thead>` tag supports the attributes listed in the following table.

Table 49: *<thead>* Tag Attributes

Attribute	*Value(s)*
align Specifies the horizontal alignment of cell content text.	**left, right, center, justify, char**
char Specifies which character to align text on. Used if **align**=char.	**character**

Attribute	*Value(s)*
charoff	**pixels, %**
Specifies the alignment offset to the first character to align on. Used if **align**=char.	
class	**class_rule, style_rule**
The class of the element.	
dir	**ltr (left to right), rtl (right to left)**
Sets the text direction.	
id	**unique_name**
Defines a unique name for the map tag.	
lang	**language_code**
Sets the language. For example, EN=English, DE=German.	
title	**tooltip_text**
Text to display in a tool tip.	
style	**style_definition**
An inline style definition.	
valign	**top, middle, bottom, baseline**
Specifies the vertical alignment of the cell content.	
xml:lang	**language_code**
Sets the language. For example, EN=English, DE=German.	

<tbody>

This tag defines a table body.

The `<tbody>` tag supports the attributes listed in the following table.

Table 50: <*tbody*> Tag Attributes

Attribute	*Value(s)*
align	**left, right, center**
Specifies the horizontal alignment of cell content text.	

<tfoot>

This tag defines table footer.

The `<tfoot>` tag supports the attributes listed in the following table.

Table 51: *<tfoot>* Tag Attributes

Attribute	*Value(s)*
align Specifies the horizontal alignment of cell content text.	**left, right, center, justify, char**
char Specifies which character to align text on. Used if **align**=char.	**character**
charoff Specifies the alignment offset to the first character to align on. Used if **align**=char.	**pixels, %**
class The class of the element.	**class_rule, style_rule**
dir Sets the text direction.	**ltr (left to right), rtl (right to left)**
id Defines a unique name for the map tag.	**unique_name**
lang Sets the language. For example, EN=English, DE=German.	**language_code**
title Text to display in a tool tip.	**tooltip_text**
style An inline style definition.	**style_definition**
valign Specifies the vertical alignment of the cell content.	**top, middle, bottom, baseline**
xml:lang Sets the language. For example, EN=English, DE=German.	**language_code**

Meta Information Tags

The following META information tags are supported:

- <head>—Defines information about the document

<head>

The <head> element contains information about the document. The browser does not display the head information. The following tags can be in the head section: <base>, <link>, <meta>, <script>, <style>, and <title>.

This tag does not support any attributes.

Note: No Title Bar in Microbrowser

Due to space constraints, there isn't a static title bar at the top of the microbrowser window as there is in most other browsers. The title is displayed in large bold text in the first line of the page and is scrolled off the screen as you navigate down the page.

Audio Tags

The following audio tags are supported:

- <audio> - Defines audio.

<audio>

The <audio> tag enables you to define audio elements that play wav files on the phone.

The <audio> tag supports the attributes listed in the following table.

Table 52: <audio> Tag Attributes

Attribute	*Value(s)*
src	**string**
Audio file source which can either be relative to the XHTML file or an absolute path. Only ".wav" files are supported by the phones.	
description	**string**
Labels the <audio> tag.	
buttons	**string ','**
Creates soft keys. Use this attribute with links. Separate each soft key with a semicolon: ";".	
links	**section_name**
Creates soft keys. Use this attribute with buttons. Separate each link with a space: " ".	

An example of the <audio> tag is shown next.

Table 53: <audio> Tag Example

```
<audio src="flute16Ksps.wav" description="This is a short description"
buttons="Details Back" links="details.xhtml SoftKey:Back" />
```

Ordered and Unordered List Tag

The following tags are supported:

- - Defines an ordered list.
- - Defines an unordered list.

List tags (<li>) are used to define ordered or unordered lists. The first item in the <li> tag must be a link with an <a> anchor tag and the HREF member set to the desired URI. You cannot have images or text before the link.

The <ol> tag defines an ordered list, which can be numerical or alphabetical. This tag does not support any attributes.

An example of an ordered list is shown next.

Table 54: <ol> Tag Example

```
<ol>
    <li><a href="target1.html">link 1</a></li>
    <li><a href="target2.html">link 2</a></li>
    <li><a href="target3.html">link 3</a></li>
</ol>
```


The <ul> tag defines an unordered list. This tag does not support any attributes.

An example of an unordered list is shown next.

Table 55: <ul> Tag Example

```
<ul>
    <li><a href="target1">link 1</a></li>
    <li><a href="target2">link 2</a></li>
    <li><a href="target3">link 3</a></li>
    <li><a href="target4">link 4</a></li>
</ul>
```

Supporting HTTP

The microbrowser is a fully compliant HTTP/1.1 user agent:

The microbrowser supports:

- Cookies
- Refresh headers
- HTTP proxies
- HTTPS over SSL/TLS
- Custom CA certificates

The following are exceptions:

- There is no sophisticated caching. The HTML cache refresh META tag is not supported.
- Any images in the body of a document with the same URL are assumed to be the same image. The image is loaded from the microbrowser's memory instead of making another request to the server.
- When a new page is requested, the microbrowser's internal memory is cleared and all components of the new page are downloaded from the server.

Web Info: To View Trusted Certificate Authorities

For more information on CA certificates, see Technical Bulletin 17877: Using Custom Certificates With SoundPoint IP Phones.

Launching the Microbrowser from the Phone

The first time the **Applications** key is pressed, the main microbrowser loads the home page specified in the `mb.main.home` configuration parameter. Subsequent presses of the **Applications** key simply toggle between the microbrowser and phone applications. The active page remains loaded in memory when you toggle.

Whenever there is an event in the phone application that requires your attention, the telephony application is brought to the foreground automatically.

While you are viewing the microbrowser, if there is an event in the phone application that requires your attention, the phone application displays automatically in the foreground. You can return to the microbrowser by pressing the **Applications** key. Even though you cannot see the microbrowser application, it is still active, and pending transactions will complete in the background and be immediately visible when the browser is brought to the foreground.

Examples of the main microbrowser on SoundPoint IP 450 and 670 are shown in Figure 5: SoundPoint IP 670 Main Browser and Figure 6: SoundPoint IP 450 Main Browser.

Figure 5: SoundPoint IP 670 Main Browser

Figure 6: SoundPoint IP 450 Main Browser

Navigating and Form Editing Behavior on the Main Browser

You can navigate through pages with the up and down arrow keys. Items are highlighted on the page in the order they appear in the XHTML source.

To select a link to follow or a text box to toggle, you must press the **Select** key. This will generate a request for the linked page or toggle the selection on the displayed page. You can enter text and move to the next selectable item when you are finished by using the Up and Down arrow keys. Long menus can be scrolled past by using the * and # keys as page up and page down respectively. If there is a large area of the page without editable fields, the page is only scrolled by one screen for each push of the arrow key.

To submit data, you need to scroll to and select a submit button on the page or press the **Submit** soft key, when available.

You can return to the previous page by pressing the **Back** soft key. The Left arrow key performs a similar function unless you are editing a text field. The **Refresh** and **Home** soft keys behave in the expected manner—reloading the current page and reloading to the phone's home page respectively.

You can enter text into text boxes using the dial pad. When editing text, you can change to uppercase letters, lowercase letters, or numeric entry modes by pressing a special soft key. You can undo edits by pressing the **Cancel** soft key.

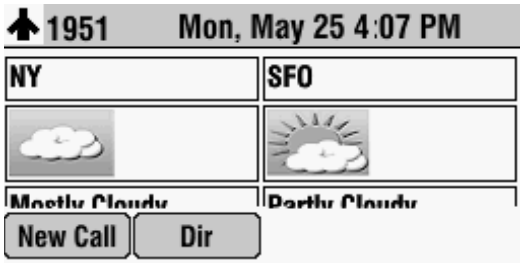
Viewing the Idle Browser

The idle display microbrowser is independent of the main microbrowser but is capable of rendering the same content. The idle browser's home page is configured via the `mb.idleDisplay.home` configuration parameter. The idle display microbrowser does not accept any user input and will only appear when there are no phone calls in progress and the phone is in the idle user interface state. The content on the idle display microbrowser is updated based on a configurable refresh timer or by honoring the value of the Refresh header.

Examples of the idle microbrowser on the SoundPoint IP 670 and SoundStation IP 7000 phones are shown in Figure 7: SoundPoint IP 670 Idle Browser and Figure 8: SoundStation IP 7000 Idle Browser.

Figure 7: SoundPoint IP 670 Idle Browser

Figure 8: SoundStation IP 7000 Idle Browser



Sample Microbrowser Web Applications

This section provides three sample applications that you can use as a starting point for writing your application for the microbrowser:

- Static XHTML Application
- Dynamic XHTML Application
- XML API Application

Note: Applications Can Be Developed on Any Web Server

Even though Tomcat is used in the following examples, you are free to use any Web server you choose. Static and dynamic XHTML applications can be developed using any Web server technologies: ASP.net, Java Servlets, Java Server Pages, CGI-PERL, PHP, etc.

Note: Careful When Copying Sample Code

Be careful when copying the lines of code in the following tables because the lines may have wrapped over to the next line. If you cut and paste these lines, the code can inadvertently contain line breaks. Check that the code is valid before executing.

Static XHTML Application

The following instructions show you how to create a static XHTML application that displays ‘Hello World!’ in the main browser.

To develop a static XML application:

1. Use the following table to create a `sample.xhtml`:

Table 56: Sample Code - Hello World

```
<html>
<head>
<title>Sample Application</title>
</head>
<body>
<p>HelloWorld!</p>
</body>
</html>
```

2. Configure the Web server to serve the above XHTML file.

   For example, if you are using Apache Tomcat to try this example, put this file into the `webapps\PLCM` folder of Tomcat.

3. Configure SoundPoint IP and SoundStation IP phones to point to the XHTML file in the **applications.cfg** configuration file.

   For this example, change `mb.main.home` to `http://<WEBSERVER_ADDRESS:PORT>/PLCM/Sample.xhtml` .

4. Reboot the phone.

5. On a SoundPoint IP phone, press the **Applications** key.

   The text ‘Hello World!’ displays on the screen.

Dynamic XHTML Application

The following instructions show you how to create a dynamic XHTML application that displays a stock ticker in the main browser.

To develop a dynamic XML application:

1. Use the following to create a `addstock.xhtml example`:

Table 57: Sample Code - addstock.XHTML Example

```
<html xmlns="http://www.w3.org/1999/xhtml">
<!- - HEADER START - ->
<head>
<title>Stocks</title>
</head>
<!- - HEADER END - ->
<!- - BODY START - ->
<body>
<!- - ADD STOCK FORM START - ->
<form method="POST" action="GetQuote.jsp">
<p>Symbol<input type="text" name="stockname"/>
<input type="submit" value="Get Quote"/></p>
</form>
<!- - ADD STOCK FORM END - ->
</body>
<!- - BODY END - ->
</html>
```

2. Configure the Web server to serve the above XHTML file.

   For example, if you are using Apache Tomcat to try this example, then put this file into the `webapps\PLCM` folder of Tomcat.

3. Write an application that is going to retrieve the stock information from a data service provider.

   For the following table, the application will retrieve stock information from Yahoo and send it to the microbrowser. This application is written using a Java Server Page (JSP). Name the file **GetQuote.jsp**.

Table 58: Retrieving Stock Information Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<%@page
import="java.io.File,java.io.IOException,java.net.URL,java.awt.image.BufferedImage,j
avax.imageio.ImageIO"%>
<html>
<head>
<title>Stock Quote</title>
</head>
<body>
<%
// GETTING THE PATH WHERE BMP FILE HAS TO BE SAVED
String bmpFilePath = application.getRealPath(File.separator) + "quote.bmp";
```

```
// DEFINE URL FROM WHERE CONTENT TO BE RETRIEVED
String stockUrl = "http://ichart.yahoo.com/t?s=";
// RETRIEVE THE STOCK SYMBOL FROM REQUEST
String stockSymbol = "PLCM"; // DEFAULT TO PLCM
if ( request.getParameter("stockname") != null ) {
stockSymbol = request.getParameter("stockname");
}
readAndConvertContentToBmp(stockUrl + stockSymbol, bmpFilePath, stockSymbol);
%>
<%!
// READ THE CONTENT FROM GIVEN URL AND THEN CONVERT THE CONTENT TO A BMP FILE
private void readAndConvertContentToBmp(String a_stockUrl, String a_filePath, String
a_name) throws IOException {
try {
BufferedImage stockImage = ImageIO.read(new URL(a_stockUrl));
ImageIO.write(stockImage, "bmp", new File(a_filePath));
}
catch (IOException ex) {   throw ex;     }
}
%>
<!-- START DISPLAY BMP FILE -->
<img src="quote.bmp"/>
<!-- END DISPLAY BMP FILE -->
</body>
</html>
```

4. Configure the Web server to deploy the above JSP file.

   For example, if you are using Apache Tomcat to try this example, put this file into the `webapps\PLCM` folder of Tomcat.

5. For this example, change `mb.main.home` to `http://<WEBSERVER_ADDRESS:PORT>/PLCM/AddStock.xhtml`.

6. Reboot the phone.

7. On a SoundPoint IP phone, press the **Applications** key.

   The `AddStock.xhtml` displays on the screen.

8. Enter a stock symbol, and select the **Get Quote** soft key.

   The stock quote for the entered stock symbol displays on the screen.

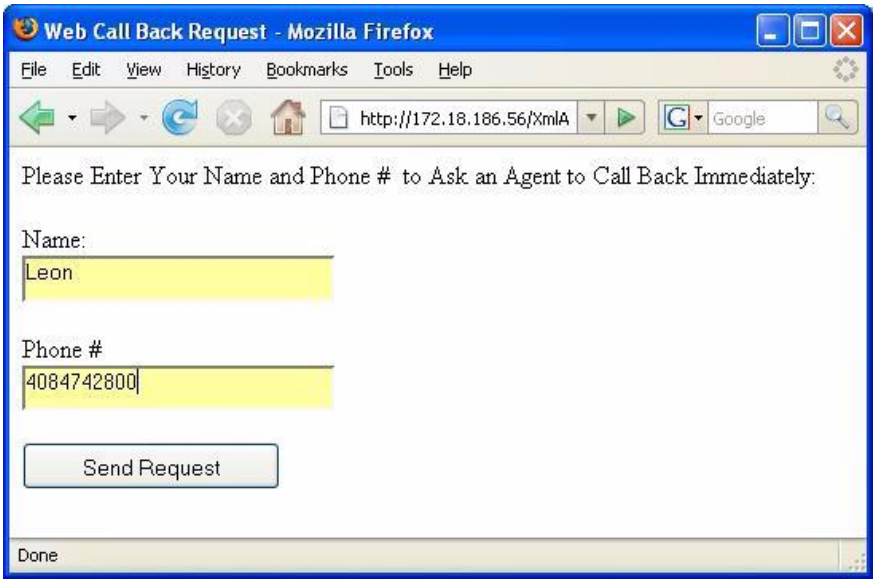
XML API Application

The following instructions show you how to create a XML API application that provides a callback request in the main browser.

This example uses a Telephone Integration URI:

- This is an ASP.NET sample for an IIS Server.

- A customer is browsing a company's Web site on the internet. They come upon this web page, `http://A_Web_Site/WebCallback.aspx`, and enter their name and phone number as shown next.



- After the customer clicks Send Request, the page shown below is pushed to the customer support agent's phone.

The customer support agent can call the customer by pressing the **Select** key because the highlighted link contains a Tel URI with the customer's phone number.

To develop an XML API application:

1 Using your integrated development environment (IDE) of choice, create a new file and name this file `webcallback.aspx`.

Table 59: webcallback XHTML Example

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="WebCallback.aspx.cs"
Inherits="WebCallback" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Web Call Back Request</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        Please Enter Your Name and Phone #  to Ask an Agent to Call Back
Immediately:<br />
        <br />
        Name:<br />
        <asp:TextBox ID="BoxName" runat="server" Height="23px"
Width="192px"></asp:TextBox><br />
        <br />
        Phone #<br />
        <asp:TextBox ID="BoxNumber" runat="server" Height="22px"
Width="192px"></asp:TextBox><br />
        <br />
        <asp:Button ID="Button1" runat="server" Height="30px"
OnClick="Button1_Click" Text="Send Request"
            Width="162px" /></div>
    </form>
</body>
</html>
```

2 Using the IDE of your choice, create a file called `webcallback.aspx.cs`.

Table 60: webcallback.aspx.cs Example

```
using System;
using System.IO;
using System.Text;
using System.Data;
using System.Configuration;
using System.Net;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Threading;

public partial class WebCallback : System.Web.UI.Page
{
    public static ManualResetEvent allDone = new ManualResetEvent(false);
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        String phoneNum = BoxNumber.Text ;
        String name = BoxName.Text;

        //send a push request to the phone with the IP address
        //NOTE: Change this hardcoded IP address
        callbackReq("172.18.103.32", phoneNum, name);

    }


    private void callbackReq(String phoneIP, String phoneNum, String name)
    {
        String strLoc = "http://" + phoneIP + "/push";
        String[] cred = { "Polycom", "456" };

        NetworkCredential myCred = new NetworkCredential(cred[0], cred[1]);

        CredentialCache myCache = new CredentialCache();

        myCache.Add(new Uri(strLoc), "Digest", myCred);
```

```
        string result = "";

        // Create the web request
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(strLoc);

        WebRequestState myRequestState = new WebRequestState();
        myRequestState.request = request;

        request.Method = "POST";
        request.Credentials = myCache;

        myRequestState.createPostData(phoneNum, name);

        IAsyncResult r = (IAsyncResult)request.BeginGetRequestStream(
            new AsyncCallback(ReadCallback), myRequestState);

        allDone.WaitOne();

        // Get response
        HttpWebResponse response = (HttpWebResponse)request.GetResponse();

        // Get the response stream
        StreamReader reader = new StreamReader(response.GetResponseStream());

        // Read the whole contents and return as a string
        result = reader.ReadToEnd();

        reader.Close();
        response.Close();

    }

    private static void ReadCallback(IAsyncResult asynchronousResult)
    {

        WebRequestState myRequestState =
(WebRequestState)asynchronousResult.AsyncState;
        WebRequest myWebRequest = myRequestState.request;

        // End the Asynchronus request.
        Stream streamResponse =
myWebRequest.EndGetRequestStream(asynchronousResult);

        byte[] byteArray = Encoding.UTF8.GetBytes(myRequestState.getPostData());

        // Write the data to the stream.
        streamResponse.Write(byteArray, 0, byteArray.Length);
        streamResponse.Close();
        allDone.Set();
```

```
    }

}

public class WebRequestState
{
    public String postData = null;

    public WebRequest request;
    public WebRequestState()
    {
        request = null;
    }

    public String getPostData()
    {
        return postData;
    }

    public void createPostData(String phoneNum, String name)
    {
        postData =
            "<PolycomIPPhone><Data Priority=\"Critical\">" +
            "<title>Customer Web Call Back Request</title>" +
           " <h1>Customer Name : " + name + " </h1> <br></br>" +
            "<a href=\"tel://" + phoneNum + ";Line1\">Callback to Customer</a>" +
            "</Data></PolycomIPPhone>";

    }
}
```

3. Configure the Web server to deploy the preceding files.
4. Change the below configuration parameters to the following:
   - Set `apps.push.username` to **Polycom**.
   - Set `apps.push.password` to **456**.

   The phone's IP address is hardcoded in `webcallback.aspx.cs` to **172.18.103.32** for this example. You must change this to another value, noted in the code.
5. Reboot the phone.

   After a customer enters their name and phone number on the Web page, the Customer Web Call Back Request page appears on the phone with IP address hardcoded in the `webcallback.aspx.cs` file.

5: Using Configuration Parameters

This chapter shows you how to configure Polycom phones to run your newly-created Web application using the configuration files that accompany the Polycom® UC software. The configuration parameters dictate the behavior of the phone once it is running the executable specified in the master configuration file.

Caution: Only Knowledgeable Administrators Should Modify the Configuration Files

Only a knowledgeable system administrator needs to modify configuration files. If you apply incorrect parameters, your phone cannot be used. The configuration files that accompany a specific UC software release must be used only with that software. Failure to do this can render the phone unusable.

The topics in this chapter include:

You can make changes to the configuration parameters by modifying the configuration files on the provisioning server using an XML editor. For detailed information on modifying the configuration files, refer to chapters Setting Up the Provisioning Sever and Configuration Methods in the Polycom's UC Software 5.0.0 Administrators' Guide. If you are configuring a single phone, you can also use the Polycom Web Configuration Utility. For information on how to configure files using the Web Configuration Utility, see the Polycom Web Configuration Utility Guide on the support site.

The parameters described in this chapter include those for:

- Web applications
- Push requests
- Telephone event notifications
- Phone state polling
- Soft Keys

A sample configuration is shown in Sample Configuration.

Configuring Web Application Parameters

The parameters shown in the following table control the home page, proxy, and size limits that are used by the microbrowser and browser when it is selected to provide services.

Table 61: Microbrowser and Web Browser Parameters

Parameter	*Permitted Values*	*Default*
mb.idleDisplay.home	**Null or any fully formed valid HTTP URL. Length up to 255 characters.**	**Null**
The URL for the microbrowser/browser home page that is shown on the idle display microbrowser/browser Home page. For example: http://www.example.com/xhtml/frontpage. If Null, there is no idle display microbrowser/browser. Note that the microbrowser/browser idle display will displace the idle display indicator.		
mb.idleDisplay.refresh	**0 or an integer > 5**	**0**
The time period in seconds that the microbrowser/browser 's idle display will refresh. If set to 0, the idle display microbrowser/browser does not refresh. The minimum refresh period is 5 seconds (values from 1 to 4 are ignored, and 5 is used). Note: If an HTTP Refresh header is detected, it will be respected, even if this parameter is set to 0. The refresh parameter will be respected only in the event that a refresh fails. Once a refresh is successful, the value in the HTTP refresh header, if available, will be used.		
mb.main.autoBackKey[2]	**0 or 1**	**1**
If 0, the phone does not provide a **Back** soft key; all soft keys are created and controlled by the application. If 1, the phone automatically supplies a **Back** soft key in all main browser screens. The **Back** soft key will take you back to the previous page in the browser history.		
mb.main.home	**Any fully formed valid HTTP URL. Length up to 255 characters.**	**Null**
The URL of the microbrowser/browser's Home page. For example: http://www.example.com/xhtml/frontpage/home. If blank, the microbrowser/browser will notify the user that a blank home-page was used.		
mb.main.idleTimeout	**0 - 600, seconds**	**40**
The timeout, in seconds, for the interactive microbrowser/browser . If the interactive microbrowser/browser remains idle for the defined period of time, the phone returns to the idle browser. If 0, there is no timeout.		
mb.main.statusbar	**0 or 1**	**0**
If 0, the status bar does not display. If 1, the status bar displays and status messages are shown.		
mb.main.toolbar.autoHide.enabled[2]	**0 or 1**	**1**
If 0, the toolbar displays continually. If 1, the toolbar disappears if not selected.		
mb.proxy	**Null or domain name or IP address in the format <address>:<port>**	**Null. Default port = 8080**
The address of the HTTP proxy to be used by the microbrowser/browser. If blank, normal unproxied HTTP is used by the microbrowser /browser.		

1 Change causes phone to restart or reboot.

Configuring Push Request Parameters

The `<apps.push/>` parameters are used to control push server controls. Requests sent to the phone's push server must be in the format of `http://<Phone IP>/push`. The push control request parameters are listed in the following table.

Table 62: Push Request Parameters

Parameter	*Permitted Values*	*Default*
apps.push.alertSound	**0 or 1**	**0**
If 0, there is no sound when an alert is pushed. If 1, there is sound.		
apps.push.messageType	**0 to 5**	**0**
Set which push priority messages should display on the phone. The possible values are: **0**: (None) Discard push messages **1**: (Normal) Allows only Normal priority push messages **2**: (Important) Allows only Important priority push messages **3**: (High) Allows only High priority push messages **4**: (Critical) Allows only Critical priority push **5**: (All) Allows all push messages of any priority		
apps.push.password	**string**	**null**
The password to access the push server URL.		
apps.push.secureTunnelEnabled	**0 or 1**	**1**
If 0, the Web server is not connected through a secure tunnel. If 1, the Web server is connected through a secure tunnel.		
apps.push.secureTunnelPort	**1 to 65535**	**443**
The port that the phone uses to communicate to the Web server when the secure tunnel is used.		
apps.push.secureTunnelRequired	**0 or 1**	**0**
If 0, communications to the Web server do not require a secure tunnel. If 1, communications require a secure tunnel.		
apps.push.serverRootURL	**URL**	**null**
The application server root URL. The relative URL received from the HTTP push message is appended to this root URL, and the combined URL is sent to the microbrowser. For example, if the application server root URL is http://172.24.128.85:8080/sampleapps and the relative URL is /examples/sample.html, the URL that is sent to the microbrowser is http://172.24.128.85:8080/sampleapps/examples/sample.html. Can be either HTTP or HTTPS.		
apps.push.username	**string**	**null**
The user name to access the push server URL. Note: To enable the push functionality, the parameters `apps.push.username` and `apps.push.password` must both be set, not null.		

Configuring Telephone Event Notification Parameters

The `<apps.telNotification/>` parameter is used to control telephone notification events. The telephone event notification parameters are listed in the following table.

Table 63: Telephone Event Notification Parameters

Parameter	*Permitted Values*	*Default*
apps.telNotification.callStateChangeEvent	**0 or 1**	**0**
If 0, call state change notification is disabled. If 1, notification is enabled.		
apps.telNotification.incomingEvent	**0 or 1**	**0**
If 0, incoming call notification is disabled. If 1, notification is enabled.		
apps.telNotification.lineRegistrationEvent	**0 or 1**	**0**
If 0, line registration notification is disabled. If 1, notification is enabled.		
apps.telNotification.offhookEvent	**0 or 1**	**0**
If 0, off-hook notification is disabled. If 1, notification is enabled.		
apps.telNotification.onhookEvent	**0 or 1**	**0**
If 0, on-hook notification is disabled. If 1, notification is enabled.		
apps.telNotification.outgoingEvent	**0 or 1**	**0**
If 0, outgoing call notification is disabled. If 1, notification is enabled.		
apps.telNotification.userLogInOutEvent	**0 or 1**	**0**
If 0, user login/logout notification is disabled. If 1, notification is enabled.		
apps.telNotification.URL	**URL**	**null**
The URL to which the phone sends notifications of specified events. Can be either HTTP or HTTPS.		
apps.telNotification.x.URL	**URL**	**null**
The URL to which the phone sends notifications of specified events, where x 1 to 9. Can be either HTTP or HTTPS. These parameters, combined with `apps.telNotification.URL` allow you to register up to 10 URLs to receive events.		

Note: Limitation for Server URLs

The configured events will be sent to all server URLs. You cannot configure a few events for a specific server and the remaining events for another server.
The Web Configuration Utility also does not allow you to set parameters individually for each server URL. If you select an event for any server, it is selected for all 10 server URLs.

Configuring Phone State Polling Parameters

The `<apps.statePolling/>` parameter is used to control state polling events. The telephone state polling notification event parameters are listed in the following table.

Table 64: Telephone Event Notification Parameters

Parameter	*Permitted Values*	*Default*
apps.statePolling.password The password to access the state polling URL.	**string**	**null**
apps.statePolling.URL The URL to which the phone sends call processing state/device/network information. The protocol used can be either HTTP or HTTPS. Note: To enable state polling, the parameters `apps.statePolling.URL`, `apps.statePolling.username`, and `apps.statePolling.password` must be set to non-null, numerical values.	**URL**	**null**
apps.statePoling.responseMode The mode of sending requested polled data. If 1, requested polled data is sent to a configured URL. If 0, the data is sent in the HTTP response back to the requestor.	**0 or 1**	**1**
apps.statePolling.username The user name to access the state polling URL.	**string**	**null**

Configuring Programmable Soft Keys

Soft keys appear on the bottom of the screen on Polycom phones. Soft keys are programmed using tags that create soft keys similar to the UC software (see Figure 3: VVX 1500 Idle Browser).

The programmable soft key tag, `<softkey>`**,** defines a soft key.

The `<softkey>` tag enables you to create a soft key with a customizable label, position, and action. You can execute different actions, listed in Table 66: Supported Actions in <softkey> Tag, by pressing the soft key on your phone.

You can only use soft keys within the interactive microbrowser. Use the following format when configuring the `<softkey>` tag:

```
<softkey index="W" name="X" label="Y" action="Z" />
```

Note: Programmable Soft Key Feature is Not Supported on VVX phones

The programmable soft key tag is not supported in the browser on all VVX phones. However, the same functionality can be created when in the browser through the HTML button tag, `<button></button>`, or through JavaScript soft key hooks. For more information, see JavaScript Examples for the Browser.

The `<softkey>` tag supports the attributes listed in the following table.

Table 65: `<softkey>` Tag Attributes

Attribute	*Permitted Values*
index	**Numeric, 1 to 8**
Position of the soft key.	
name	**String**
Text displayed on soft key when the `Softkey:Submit` action is used. It is ignored for all other actions. Use in cases where more than one `Softkey:Submit` action appears on a page.	
Label[1]	**String**
Text displayed on soft key. The maximum length is 9 characters.	
action	**URI**
Supported actions must be one of those listed in the table shown next.	

[1] If empty or absent, default action name is displayed.

The following table lists the supported actions in the `<softkey>` tag and the action's names and descriptions.

Table 66: Supported Actions in <*softkey*> Tag

Action	*Default Action Name*	*Description*
SoftKey: Home	Home	Moves to configured home page.
Softkey:Back	Back	Moves to previous page
SoftKey:Exit	Exit	Exits microbrowser
SoftKey:Cancel	Cancel	Cancels action
SoftKey:Refresh	Refresh	Refreshes current page
SoftKey:Fetch; <URI>	Fetch	Fetches the page from the given URI
SoftKey:Reset	Reset	Clears all input fields in the form
SoftKey:Submit	Submit	Submits the form
Key:VolDown	VolDown	Decreases volume by 1 unit
Key:VolUp	VolUp	Increases volume by 1 unit
Key:DoNotDisturb	Do not disturb	Enables Do Not Disturb feature
Key:Headset	Headset	Enables use of microphone

Action	*Default Action Name*	*Description*
Key:Handsfree	Hands-free	Enables use of speaker
Key:Messages	Messages	Opens the **Messages** menu
Key:Applications	Applications	Opens the **Applications** menu
Key:MicMute	Mute	Mutes the phone when the call state
Key:Directories	Directories	Opens the **Directories** menu
Key:Menu	Menu	Opens the main menu
Key:Setup	Setup	Opens the **Settings** menu

Depending on the microbrowser state, a number of predefined soft keys exist as described in the following table.

Table 67: Predefined Soft keys and the Microbrowser Display State

Microbrowser State	*Predefined Soft Key*
Fetching pages	Home, Refresh, Back, Stop
Rendering pages	Home, Refresh, Back, Exit
Edits—Input element is on focus	Home, Character Encoding, Back, Exit. Character Encoding lists possible character sets for the input elements in the page.

Note: When Default Soft Keys Override Custom Soft Keys

The soft keys from the `Fetching pages` and `Edit Active` soft key groups override any custom soft keys defined in the current XHTML. The soft keys from the `Rendering page` soft key group appear if no custom soft keys are defined. The soft keys that display vary between the SoundPoint IP and SoundStation IP phones.

Keep in mind the following important notes regarding `<softkey>` tags:

- All actions are case sensitive, and the correct character case is required.
- If you leave the soft key action name empty, the soft key tag is ignored.
- The `Reset` and `Submit` soft key tags must exist inside the `<form>` tag they are expected to act upon.
- The `Reset` and `Submit` soft key tags can exist inside a single form element. If there are multiple forms inside an XHTML document, the XHTML `Submit` and `Reset` input elements must be used.
- Indexes do not need to be sequential. A missing index will result in an empty space, and no soft key is displayed.

- An index greater than eight is ignored.
- By default, a `Back` soft key is placed on the graphic display even if one is not defined.

Note: When the Back Soft Key Doesn't Display

When `mb.main.autoBackKey` is set to 0, the `Back` soft key will not display.

- When using more than one `Submit` soft key on a page, change the name to distinguish between each.

  For example, to create a simple page:

Table 68: Sample Code - Hello World

```
<html>
    <p> Hello World! </p><br/>
    <softkey index="1" label="Home" action="SoftKey:Home" />
    <softkey index="2" label="Refresh" action="SoftKey:Refresh" />
    <softkey index="4" label="Exit" action="SoftKey:Exit" />
    <softkey index="3" label="Back" action="SoftKey:Back" />
</html>
```

Sample Configuration

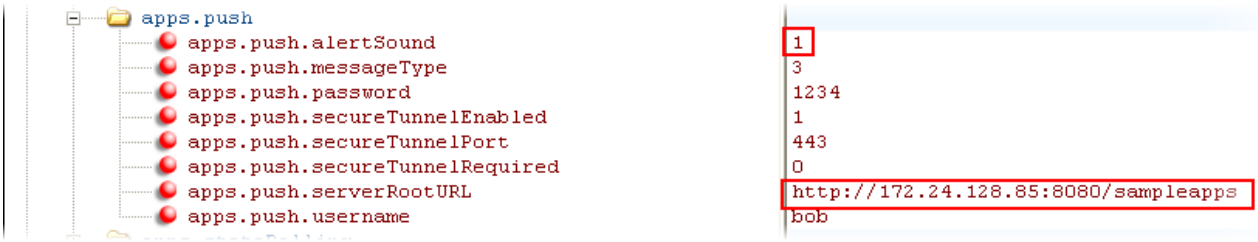
The following sample configuration shows you how to enable your users to use the Web applications that you developed.

In this example:

- `mb.proxy` is set to the address of the desired HTTP proxy to be used by the microbrowser or browser. For example, **10.11.32.103:8080**.
- `mb.idleDisplay.home` is set to the URL used for the microbrowser or browser idle screen home page. For example, `http://10.11.32.128:8080/sampleapps/idle`.
- `mb.idleDisplay.refresh` is set to the period in seconds between refreshes of the idle display microbrowser or browser's content.

- `mb.main.home` is set to the URL used for the microbrowser or browser home page. For example, `http://10.11.32.128:8080/sampleapps/login`.

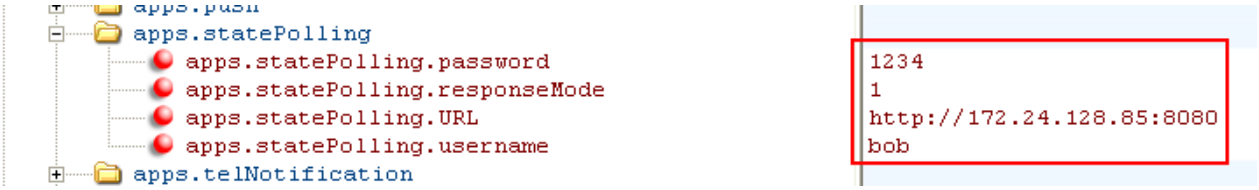
- `apps.push.alertSound` is set to 1, so a sound is played when an alert is pushed.
- `apps.push.messageType` is set to the appropriate display priority. For example, **3** Important Priority messages only.
- `apps.serverRootURL` is set to the application server root URL. For example, `http://172.24.128.85:8080/sampleapps`.
- `apps.push.username` is set to the appropriate user name. For example, **bob**.
- `apps.push.password` is set to the appropriate password. For example, **1234**.



- `apps.telNotification.URL` is set to the URL where notifications should be sent. For example, `http://172.24.128.85:8080`.
- `apps.telNotification.offhookEvent` is set to 1 to enable notifications for off-hook events.
- `apps.telNotification.onhookEvent` is set to 1 to enable notifications for on-hook events.
- `apps.telNotification.userLogInOut` is set to 1 to enable notifications for user login and logout events.

- `apps.telNotification.callStateChange` is set to 1 to enable notifications for call state change events.

- `apps.statePolling.URL` is set to the location where requested state polling information should be sent. For example, `http://172.24.128.85:8080`.
- `apps.statePolling.responseMode` is set to send the requested state polling information to the configured URL, 1, instead of back to the requestor.
- `apps.statePolling.username` is set to the appropriate username. For example, **bob**.
- `apps.statePolling.password` is set to the appropriate password. For example, **1234**.



For other references to Web information, see References look for the Web Info icon:

6: Getting Help

This section provides a list of Polycom documents referred to in this guide as well as partner resources you can use. If you are looking for help or technical support for your phones, the following types of documents are available on the Polycom Voice Support site:

- Quick Start Guides, which describe how to assemble phones.
- Quick User Guides, which describe the basic phone features.
- User Guides, which describe both basic and advanced phone features.
- Administrator's Guide, which provides instructions for installing, provisioning, and administering Polycom phones.
- Feature Descriptions and Technical Notifications such as Feature Profiles, Engineering Advisories, Technical Bulletins, and Quick Tips which describe workarounds to existing issues and provide expanded descriptions and examples.
- Release Notes, which describe the new and changed features and fixed problems in the latest version of the software.

Polycom and Partner Resources

For more information about installing, configuring, and administering Polycom products, refer to Documents and Downloads on Polycom Support.

The Polycom Community

The Polycom Community gives you access to the latest developer and support information. Participate in discussion forums to share ideas and solve problems with your colleagues. To register with the Polycom Community, create a Polycom online account. When logged in, you can access Polycom support personnel and participate in developer and support forums to find the latest information on hardware, software, and partner solutions topics.

Check out the Polycom Technology and Solution Developer's Forum at the Polycom Community.

7: Troubleshooting

The browsers that best match Polycom phone browsers are Chrome or Safari, as they are built on WebKit as well. You can use Chrome or Safari to test rendering issues on the computer before testing them on the phone's browser.

When debugging Web pages, the Inspect Element in the phone simulator and in Chrome is very helpful in finding coding issues. Also, Firebug is a useful Firefox add-on that can be used to debug Web pages.

To debug Web pages:

1. Use Firebug (in Firefox) or Inspect (in Chrome or the phone simulator) to check for JavaScript errors.
2. Use Firebug or Inspect to check that all asynchronous requests are working properly.
3. Determine if there are server errors.

   If there are server errors, use the generated error messages Apache logs to figure out the error.
4. Repeat this process until there are no errors.

Polycom has an active Developer Community Forum where you can find more sample code as well as answers to many common developer questions. You can also post questions to Polycom support experts and other developers.

This chapter also presents problems, likely causes, and corrective actions. Problems are organized into the following categories:

- Understanding Microbrowser Application Errors

Understanding Microbrowser Application Errors

The following table describes possible solutions to microbrowser application errors.

Table 69: Troubleshooting Microbrowser Application Errors

Improperly formatted tables can cause error 'XML Error (17,75) mismatched tag'.
Solution: Correct the improperly formatted table.

The following table describes possible solutions to browser application errors

Table 70: Troubleshooting Browser Application Errors

Pushed message is not displaying in browser.
Push message is displayed in the browser based on the priority of the message. See Table 10: How Priority Affects URL Push Requests
The Server Not Found error usually occurs on the phone after a URL Push when the `apps.push.serverRootURL` parameter is set incorrectly and the phone cannot resolve the URL to a valid page.
Partial page is rendered on a Data Push after a long delay. If a Data Push is sent with URLs for an additional page with invalid elements embedded, the phone will first show a blank page with a very slow moving, or even stopped, progress bar and eventually only render the elements it was able to retrieve. Check that the URLs for any additional page elements are correct and reachable by the phone. For example, firewalls and VLANs can present barriers.

8: References

This chapter provides you with a list of recommended references that can help you when writing your application. When writing your application, refer to the following resources for additional help.

- You can locate Polycom's UC Software 5.0.0 Administrators' Guide on the Polycom Support Web site.
- You can locate all Technical Bulletins and Notifications, Feature Descriptions, and phone documentation referred to in the Administrators' Guide on the Polycom Support site.
- You can find Request For Comments (RFC) documents by entering the RFC number.
- You can find an HTML Reference at W3 HTML & CSS .
- You can find Third Party Software on the Polycom Support site.

Additional Information

This section provides you with information on the following topics:

- Unsupported XHTML elements on the Microbrowser
- JavaScript Examples for the Browser

Unsupported XHTML elements on the Microbrowser

This section provides information on XHTML elements not supported by the microbrowser.

The unsupported elements and attributes are listed in the following table.

Table 71: Unsupported Elements and Attributes

Tag Type	*Tag Description*
Basic Tags	<html>—Defines HTML document. <body>—Defines document's body. <h1> to <h6>—Defines header 1 to header 6. <p>—Defines a paragraph. —Inserts a single line break. <hr>—Defines a horizontal rule.
Character Format Tags	<b>—Defines bold text. <font>—Deprecated. Defines text font, size, and color. <i>—Defines italic text. —Defines emphasized text.

Tag Type	*Tag Description*
	<big>—Defines big text. <strong>—Defines strong text. <small>—Defines small text. <sup>—Defines superscripted text. <sub>—Defines subscripted text. <bdo>—Defines the direction of text display. <u>—Deprecated. Defines underlined text.

Output Tags	<pre>—Defines preformatted text.
	<code>—Defines computer code text.
	<tt>—Defines teletype text.
	<kbd>—Defines keyboard text.
	<var>—Defines a variable.
	<dfn>—Defines a definition term.
	<samp>—Defines sample computer code.
	<xmp>—Deprecated. Defines preformatted text.

Block Tags	<acronym>—Defines an acronym. <abbr>—Defines an abbreviation. <address>—Defines an address element. <blockquote>—Defines a long quotation. <center>—Deprecated. Defines centered text. <q>—Defines a short quotation. <cite>—Defines a citation. <ins>—Defines inserted text. <del>—Defines deleted text. <s>—Deprecated. Defines strikethrough text. <strike>—Deprecated. Defines strikethrough text.
Link Tags	<a>—Defines an anchor. The following attributes are not supported: charset, cords, hreflang, rel, rev, shape, target, type, id, class, title, style, dir, lang, xml:lang, tabindex, and accesskey.

Tag Type	*Tag Description*
	<link>—Defines a resource reference.
Frame Tags	<frame>—Defines a sub window (frame). <frameset>—Defines a set of frames. <noframes>—Defines a noframe section. <iframe>—Defines an inline sub window (frame).
Input Tags	<form>—Defines a form. The following attributes are not supported: accept, accept charset, enctype, target, class, id, style, title, dir, lang, and accesskey. <input>—Defines an input field. The following attributes are not supported: accept, align, alt, disabled, maxlength, readonly, size, arc, type:button, type:file, type:image, class, is, style, title, dir, lang, and accesskey. <textarea>—Defines a text area. <button>—Defines a push button. <select>—Defines a selectable list. <optgroup>—Defines an option group. <option>—Defines an item in a list box. <label>—Defines a label for a form control. <fieldset>—Defines a fieldset. <legend>—Defines a title in a fieldset. <isindex>—Deprecated. Defines a single-line input field.
List Tags	—Defines an unordered list. —Defines an ordered list. —Defines a list item. <dir>—Deprecated. Defines a directory list. <dl>—Defines a definition list. <dt>—Defines a definition term. <dd>—Defines a definition description. <menu>—Deprecated. Defines a menu list.

Tag Type	*Tag Description*
Image Tags	<img>-Defines an image. The following attributes are not supported: alt, align, border, hspace, ismap, longdesc, usemap, vspace, id, class, title, style, xml:lang, and lang <map>—Defines an image map. <area>—Defines an area inside an image map.
Table Tags	<table>—Defines a table. The following attributes are not supported: bgcolor, frame, rules, summary, id, class, title, style, dir, lang, and xml:lang. <col>—Defines attributes for table columns. <tr>—Defines a table row. The following attributes are not supported: bgcolor, char, charoff, valign, id, class, title, style, dir, lang, and xml:lang. <td>—Defines a table cell. The following attributes are not supported: abbr, axis, bgcolor, char, charoff, headers, height, nowrap, scope, valign, width, id, class, title, style, dir, lang, and xml:lang. <tbody>—Defines a table body. The following attributes are not supported: align:justify, align:char, char, charoff, valign, id, class, title, style, dir, lang, and xml:lang. <colgroup>—Defines groups of table columns.
Style Tags	<style>—Defines a style definition. —Defines a section in a document. —Defines a section in a document.
Meta Information Tags	<head>—Defines information about the document. No attributes are supported. <title>—Defines the document title. <meta>—Defines meta information <base>—Defines a base URL for all the links in a page <basefont>—Deprecated. Defines a base font
Programming Tags	<script>—Defines a script <noscript>—Defines a noscript section <applet>—Deprecated. Defines an applet <object>—Defines an embedded object

Tag Type	Tag Description
	<param>—Defines a parameter for an object

JavaScript Examples for the Browser

This section provides JavaScript examples that work in conjunction with the browser on Polycom® VVX® 500 and 1500 phones.

Note: Examples have Wrapped Lines

The lines of code shown below may have wrapped. If you cut and paste these lines, they can inadvertently contain line breaks. Check for valid code before executing.

Control of Soft Keys

The following example shows how to control soft keys.

Table 72: Soft Key Control Example

```
html>
   <head>
      <Title>Softkey JavaScript Test</Title>
      <script type="text/javascript">
         // PolySoftKey is the exported DOM object
         // Registers a JavaScript function to be executed when a custom softkey
event occurs
         PolySoftKey.customSoftkeyEvent.connect(skCallBack);

         PolySoftKey.setSoftkeyLabel(0, "one");
         PolySoftKey.setSoftkeyLabel(1, "Two");
         PolySoftKey.setSoftkeyLabel(2, "Three");
         PolySoftKey.setSoftkeyLabel(3, "Four");

         function skCallBack(key, skEvent){
            switch(key){
            case 0:
               document.getElementById("eventStuff").innerHTML = "SK 1 was pressed";
               break;
            case 1:
               document.getElementById("eventStuff").innerHTML = "SK 2 was pressed";
               break;
            case 2:
               document.getElementById("eventStuff").innerHTML = "SK 3 was pressed";
```

```
                break;
             case 3:
                document.getElementById("eventStuff").innerHTML = "SK 4 was pressed";
                break;
             }
             document.getElementById("eventValue").innerHTML = skEvent;
         }

          // hide the tool bar
          function hideSKs(){
             PolySoftKey.hideToolBar();
          }

          // show the tool bar
          function showSKs(){
             PolySoftKey.showToolBar();
          }

          // get the styled points of the SKs so app can add whatever object they
want to that area
          // after calling hideToolBar()
          function getSKPoints(){
             // Returns a JSON object with two properties, X & Y.  To convert to JS
object you must
             // use the eval function on the JSON object.
             var one = PolySoftKey.getSoftkeyPoint(0);
             var oneObj = eval('(' + one + ')'); //to help avoid syntax errors, wrap
with '(' ')' chars

             var two = PolySoftKey.getSoftkeyPoint(1);
             var twoObj = eval('(' + two + ')');

             var three = PolySoftKey.getSoftkeyPoint(2);
             var threeObj = eval('(' + three + ')');

             var four = PolySoftKey.getSoftkeyPoint(3);
             var fourObj = eval('(' + four + ')');

             document.getElementById("points").innerHTML = oneObj.X + ":" + oneObj.Y
+ "," + twoObj.X + ":" + twoObj.Y + "," + threeObj.X + ":" + threeObj.Y + "," +
fourObj.X + ":" + fourObj.Y;
             }
       </script>
    </head>
    <body onload="onInit()">
       <div id="showButton">
          <input type='button' onclick='showSKs()' value='Show Softkeys'/>
       </div>
       <div id="hideButton">
          <input type='button' onclick='hideSKs()' value='Hide Softkeys'/>
```

```
      </div>
      <div id="eventText">
         <p>Last Click: <b id='eventStuff'>0</b> </p>
         <p>Event Value: <b id='eventValue'>0</b> </p>
      </div>
      <div id="clickPos">
         <input type='button' onclick='getSKPoints()' value='Show Points'/>
         <p id="points"></p>
      </div>
   </body>
</html>
```

Keypad Captures

The following example shows how to capture keypad keys, and how to map keypad keys to an HTML button t to click when you press the **1** dial pad key.

Table 73: Keypad Capture Example

```
<title>JavaScript key press event</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<script type="text/javascript">
document.onkeyup = KeyCheck;

function KeyCheck(e)
{
var KeyID = (window.event) ? event.keyCode : e.keyCode;

switch (KeyID)
{ case 49: document.Form1.KeyName.click(); break; default: break;}

}
</script>
</head>

<body>
<form name="Form1">
<input type="button" name="KeyName" value="Click or press (1) to Continue" />
</form>
</body>
</html>
```