Spectralink Versity Smartphone

Web Developer’s Guide

Spectralink Versity Web API

SPECTRALINK

Versity models are covered in this document: Versity 95/96/97 Series.

721-0059-000 Rev. G
March 2025

Copyright Notice

Notice

Spectralink Corporation has prepared this document for use by Spectralink personnel and customers. The drawings and specifications contained herein are the property of Spectralink and shall be neither reproduced in whole or in part without the prior written approval of Spectralink, nor be implied to grant any license to make, use, or sell equipment manufactured in accordance herewith.

Spectralink reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult Spectralink to determine whether any such changes have been made.

NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE, OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY SPECTRALINK FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF SPECTRALINK WHATSOEVER.

Warranty

The *Product Warranty and Software License and Warranty* and other support documents are available at http://support.spectralink.com.

Contact Information

US Location
+1 800-775-5330
Spectralink Corporation
305 S. Arthur Avenue
Louisville, CO 80027
USA
info@spectralink.com

Denmark Location
+45 7560 2850
Spectralink Europe ApS
Bygholm Soepark 21 E Stuen
8700 Horsens
Denmark
infoemea@spectralink.com

UK Location
+44 1344 206591
Spectralink Europe ApS—UK branch
Suite B1, The Lightbox
Bracknell, Berkshire, RG12 8FB
United Kingdom
infoemea@spectralink.com

Document History

Rev	Change description
G	Revised tip formatting; Ch 4: add notify attribute to Push requests. Remove V92 and PIVOT references.
F	Updated text and graphics; text and format revisions for clarity.
E	Modify text for Versity 92 Series.
D	QR3 release 95/96 Series 1.7. Minor typo corrections. Add PushURI tel: behavior anomalies between Wi-Fi and LTE phones.
C	QR2 release: 95/96 Series 1.6.0.1212. Ch 4: add note to emphasize telephony integration applies only to the Biz Phone app. For Event notifications, add info on how the LTE phone can “know” its cellular IP address. For Phone state polling, delete VLANID as deprecated.
B	Rearrange and rewrite portions of the explanation of what the Web API is to improve clarity. Add Versity examples of Alertview screens. Volume tag values require single quotes. Add latitude/Longitude SafeEvent and Cellular IP address for LTE 96xx models. Add viewing a Login/Logout event. For Push Requests, add table and info to clarify differences in interrupt behavior between Versity and PIVOT.
A	Pre-release--BETA

Contents

Introduction

About This Guide

This Web Application Developer's Guide provides guidelines for developing and testing Web applications that will run on Spectralink Versity smartphones using Spectralink software. The Spectralink Versity smartphone is part of the Versity product family developed by Spectralink.

Revision F Update: The API spec includes Spectralink 97 Series smartphones. The 97 Series device uses the same API as the 95/96 Series devices, so all Versity models are covered in this document.

Who Should Read This Guide?

This guide is designed specifically to provide Web application creators with information for developing and deploying Web applications to Spectralink phones. This guide is not intended for end users and does not provide user-level information on how to use any specific Web applications.

Before reading this guide, you should be familiar with the following:

- Basic text editors, or full IDE-like Eclipse™ or Microsoft Visual Studio® for creating or writing code
- General web application and software development best practices
- JavaScript™ and CSS
- Adequate planning, creating, and testing resources needed to produce a fully deployable Web-based application

You should expect to become familiar with telephony deployment as it pertains to Spectralink smartphones including:

- Spectralink Versity smartphones, infrastructure requirements, and provisioning methods.

Recommended Software Tools

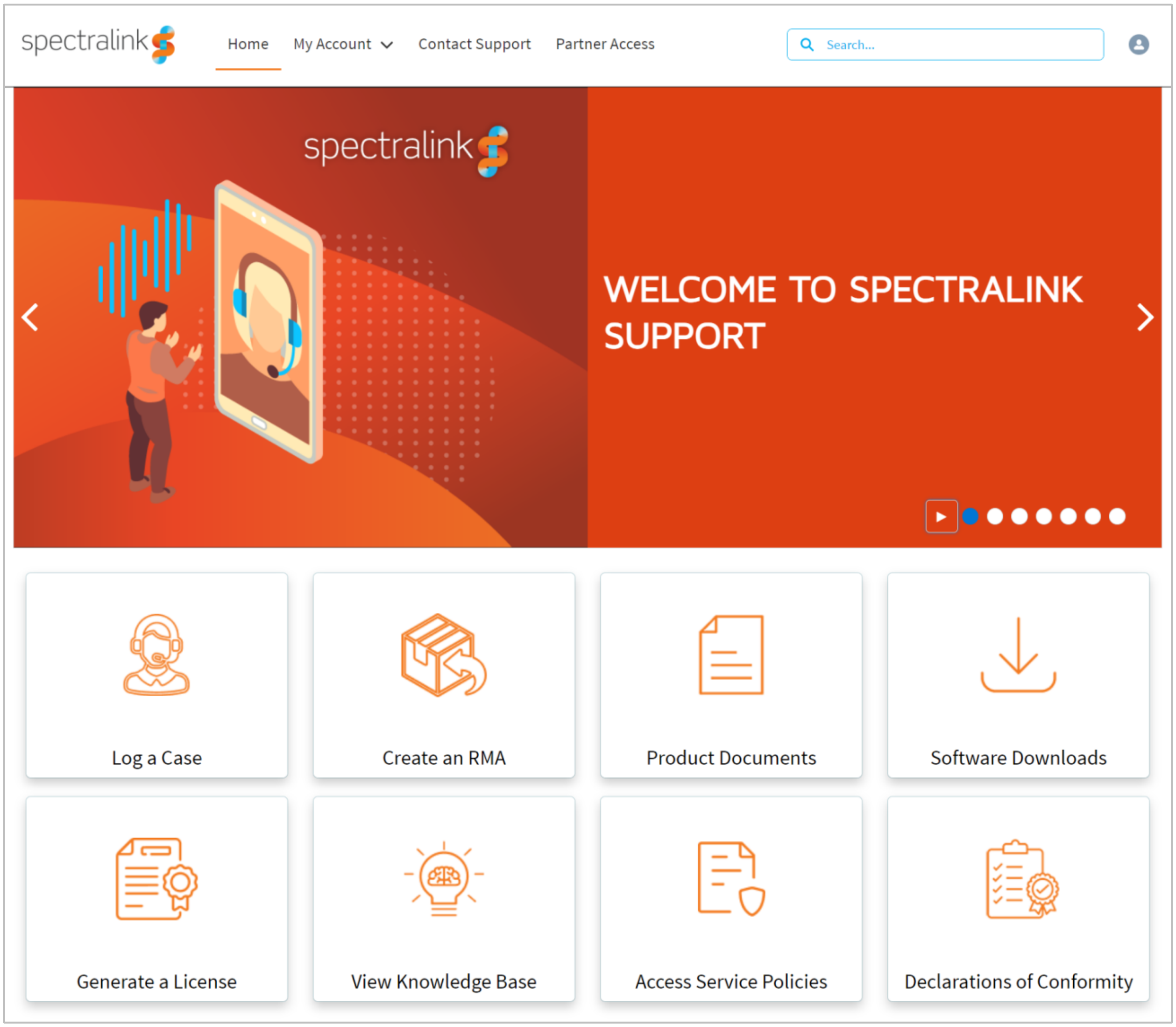
For more complex applications, you may need to use a fully Integrated Development Environment (IDE) like Eclipse or Microsoft Visual Studio.

Product Support

For support for web application development send an email to aims@spectralink.com.

Spectralink References

All Spectralink documents are available at https://spectralink.my.site.com.



Specific Documents

Spectralink Versity 95/96 software and support documents on the Spectralink support site.

Spectralink SAM software and support documents on the Spectralink support site.

Release Notes accompany every software release and provide the new and changed features and resolved issues in the latest version of the software. Please review these for the most current information about your software.

Spectralink Versity Deployment Guide provides a high-level overview of the deployment process for Spectralink Versity smartphones. This includes the interface with an EMM, the method for getting Versity connected to the wireless LAN, and the interface with the Spectralink Application Management (SAM) server.

The *Spectralink Applications Management (SAM) Administration Guide* provides information about every setting and option for the Spectralink applications that are available to the administrator on the SAM server. Time-saving shortcuts, troubleshooting tips and other important maintenance instructions are also found in this document.

The *Spectralink Applications Administration Guide* describes each Spectralink app and lists each parameter configured for each app. [not yet released]

The *Spectralink Versity User Guide* offers comprehensive instructions for using each of the Spectralink Applications deployed on the handsets.

For information on LTE technology and carrier interoperability, see the *Spectralink Versity Smartphone LTE Carrier Interoperability Guide*

For information on IP PBX and soft switch vendors, see the *Spectralink Call Server Interoperability Guide*.

Technical Bulletins and Feature Descriptions explain workarounds to existing issues and provide expanded descriptions and examples.

AP Configuration Guides explain how to correctly configure access points and WLAN controllers (if applicable) and identify the optimal settings that support Spectralink Versity smartphone. You can find them on the *VIEW Certified* webpage.

White Papers

Spectralink White Papers are available at https://www.spectralink.com/ebooks.

For details on RF deployment please see *The Challenges of Ensuring Excellent Voice Quality in a Wi-Fi Workplace* and *Deploying Enterprise-Grade Wi-Fi Telephony*.

These White Papers identify issues and solutions based on Spectralink's extensive experience in enterprise-class Wi-Fi telephony. They provide recommendations for ensuring that a network environment is adequately optimized for use with Spectralink devices.

Other references

You can find IETF RFC (Request for Comments) documents at https://www.ietf.org/standards/rfcs/.

You can find an HTML Reference at http://www.w3.org/standards/webdesign/ .

Conventions Used In This Document

Icons

Icons indicate extra information about nearby text.

CAUTION
The *Caution* icon highlights information you need to know to avoid a hazard that could potentially impact device performance, application functionality, successful feature configuration and/or affect smartphone or network performance.

NOTE
The Note icon highlights information of interest or important information that will help you be successful in accomplishing a procedure or understanding a concept.

TIP
The Tip icon highlights information that may be valuable or helpful for users to know, such as special techniques, shortcut methods, or information that will make user tasks easier to perform.

WEB
The *Web Info* icon highlights supplementary information available online such as documents or downloads on support.spectralink.com or other locations.

ADMIN TIP
This tip advises the administrator of a smarter, more productive or alternative method of performing an administrator-level task or procedure.

SETTINGS
The Settings icon highlights information to help you zero in on settings you need to choose for a specific behavior, to enable a specific feature, or access customization options.

Typography

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

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 smartphone.
Italics	Used to emphasize text, to show example values or inputs, and to show titles of reference documents available from the Spectralink Support Web site and other reference sites.
Underlined blue	Used for URL links to external Web pages or documents. If you click on text in this style, you will be linked to an external document or Web page.

Convention	*Description*
Bright orange text	Used for cross references to other sections within this document. If you click on text in this style, you will be taken to another part of this document.
`Fixed-width-font`	Used for code fragments and parameter names.

This guide also uses a few writing conventions to distinguish conditional information.

Convention	*Description*
<*MACaddress*>	Indicates that you must enter information specific to your installation, smartphone, or network. For example, when you see <*MACaddress*>, enter your smartphone'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.

Code Examples

Sample code is shown in this guide to assist you in writing your applications. All samples are presented as shown next.

TIP

Examples have wrapped lines

Be aware that the lines of code shown in this document are formatted to fit the page and may be wrapped. If you cut and paste these lines they may contain line breaks that break the code. Check for valid code before executing.

Example: 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="Press (1) to Continue" />
  </form>
  </body>
</html>
```

Chapter 1: The Versity Web API

Introducing the Spectralink Web API App

Versity smartphones ship with the Spectralink Web API app to support Web developers. The Spectralink Web API app contains the JavaScript extensions necessary to support developer requirements, as detailed in this document. It allows developers to interface with external services and provide links to frequently used websites in addition to providing a way to configure the smartphones to integrate with an XML application.

The Web API provides:

- A widget to display a set of customer-defined URLs for applications, and a special browser, WebView,
- A custom notification management tool, Alertview, that gives applications the ability to push data or a URL to the phone and have it displayed in the Alertview notification window,
- The capability for applications to receive notifications of events or poll for status.

The Alertview notification window and the App URLs widget ensure extended app availability for the Web API app.

By providing two separate activities for both pushed content and content the user has requested, content is separated and the user does not lose the content that they asked for if pushed HTML content is sent to them:

- Pushed content is delivered to the user as a standard Android notification, which is displayed in the notification drawer. It is not assumed that all pushed content is more important than the current user activity. Therefore, only Critical priority pushed content will take over the user’s foreground activity and open the Alertview. Lower priority content is queued up and shown when the user selects the notification.
- User-initiated links in the App URLs launch when the user opens the widget box containing the App URLs. The URLs available in the widget are configured by the system admin using SAM. See the *Spectralink Applications Management Administration Guide*. Once tapped, the shortcuts open applications within a browser.

Interaction with other Android Applications

Because pushed content is sent to the Android notification manager the user can choose to handle it at their convenience – except for Critical priority content, which notifies the user audibly and becomes the foreground activity. There is no adverse interaction with other running Android applications.

Interaction with phone calls

If the user is in a phone call they will not be interrupted by pushed content except for Critical priority content, which will notify them audibly and will become the foreground activity.

For information about limitations to the Web API when a 3rd-party VoIP application is used instead of the Spectralink SIP application, see Appendix A: Spectralink Versity Web API and Third-party VoIP Clients on the Versity Smartphones.

Other browsers that may be installed on the phone

Remember that the end-user may have many other apps, including browsers, on their phone. These browsers will not contain the extensions that are present in the Spectralink Web API app.

Web Development Overview

Web applications running on Spectralink phones can be as simple as a list of contacts or as complex as a nurse call system. Spectralink Versity smartphones support App URLs, where users can interact with Web pages as they would on a computer.

Development of a web application for the Versity smartphone generally follows these steps:

1 Planning. Defining the requirements of the application according to the facility's needs.

2 Familiarization with the capabilities of the Spectralink Versity smartphone.

3 Familiarization with the infrastructure requirements of the Spectralink Versity smartphones – e.g. call control (telephony server), wireless LAN, etc. You'll need to learn about the components of the entire system to implement your application. This knowledge is obtained through study of the *Spectralink Versity Deployment Guide* and VIEW Program *AP Guides*.

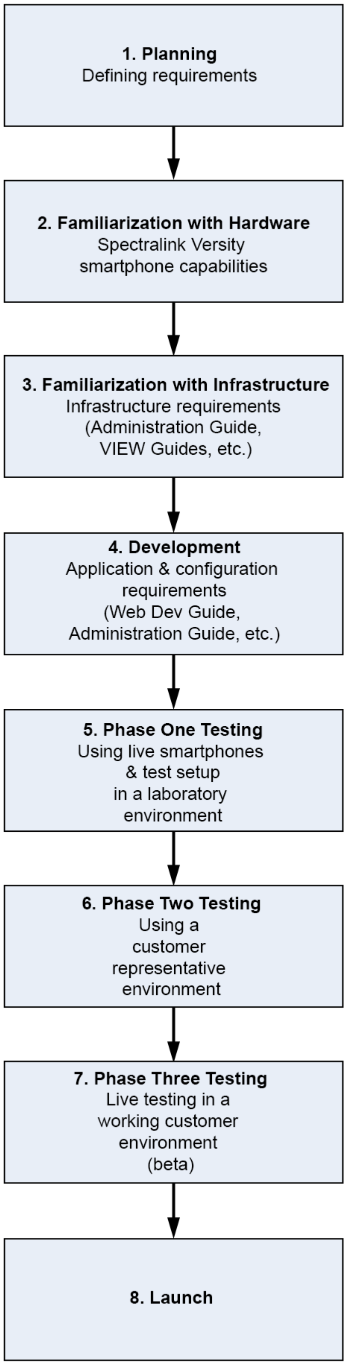
4 Application development and configuration requirements development. From your research on the requirements of the infrastructure, you will develop the application itself and configure the smartphone parameters needed to integrate with the application. The settings will become central to testing your application and ultimately will be deployed along with the application in test and customer environments.

5 The first phase of Application testing and debugging uses Spectralink Versity hardware, running your customized settings and other components to mimic a telephony deployment: a simple wireless LAN environment and call server.

6 The second phase of application testing uses Spectralink Versity smartphones deployed in a customer representative wireless LAN test environment. This test setup is detailed in this guide. During this test, applications can be tested for capacity as well as robustness for phones moving on and off the wireless LAN (due to power cycles and out of range movement).

7 The third phase of application testing is done during deployment in a working environment.

8 Launch.


1. Planning
Defining requirements
2. Familiarization with Hardware
Spectralink Versity smartphone capabilities
3. Familiarization with Infrastructure
Infrastructure requirements (Administration Guide, VIEW Guides, etc.)
4. Development
Application & configuration requirements (Web Dev Guide, Administration Guide, etc.)
5. Phase One Testing
Using live smartphones & test setup in a laboratory environment
6. Phase Two Testing
Using a customer representative environment
7. Phase Three Testing
Live testing in a working customer environment (beta)
8. Launch

Using XHTML

XHTML (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 in which web pages are written. XHTML is HTML 4.01 redesigned as XML.

You should ideally have experience working with HTML and XHTML programming or access to someone who has such experience to benefit from the information and discussion 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

ADMIN TIP
***What else do I need to create* applications?**
You can use whatever development languages or servers you choose, including JavaScript, PHP, Python®, Django®, Tomcat™ or Apache™. Use the tools you are most comfortable using – after consulting with your IT department to ensure they support your tools.

Chapter 2: Your Application & the Spectralink Versity Smartphone

The Spectralink Versity smartphone is a powerful device that enables many types of applications, including native Android applications and Web-based applications.

Applications that are standard web applications or pages can be accessed using any web browser in the same manner as a user would on a smartphone.

A web application that utilizes any of the features of the Spectralink Versity Web API will use the Spectralink Webview instead of a common browser.

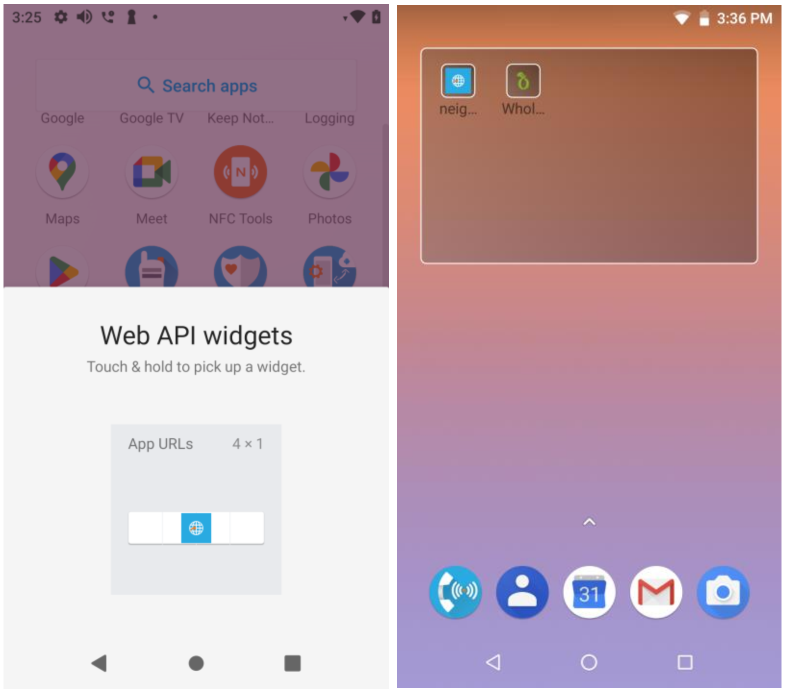
Alerts sent to the phone from a web application server will show up to the user in the Spectralink Alertview in addition to creating an Android notification in the notification bar.

What is the Spectralink Webview?

The Spectralink Webview is a pared-down browser with enhanced Javascript capabilities designed specifically for onboard applications. A user accesses the Webview by using App URLs -- programmed shortcuts that open the application. The web application shortcut widget supports up to 10 App URLs.

To make the shortcuts visible on the home screen, long press on the WebAPI icon in the Android launcher and select Widgets. Then long press and drag the widget box to a home screen.

Install the App URLs widget box. The App URLs display together in thewidget box.



The Webview supports true Spectralink Versity applications with the following features:

- HTML 5 – without video support
- CSS 3.0 – allowing only a single transition at a time
- SVG 1.1 (partial support)
- JavaScript / The Web API app is used by developers to interface with external services and provide links to frequently used websites
- Web API allows you to configure the smartphones to integrate with an XML application.DOM access
- XMLHttpRequest
- HTTP 1.1
- AJAX

What is the Spectralink Alertview?

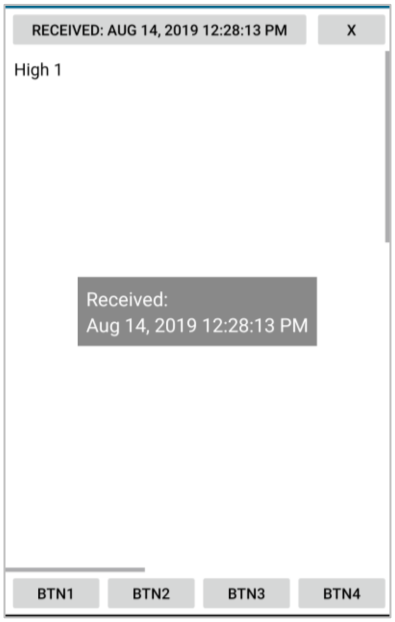
The Spectralink Alertview allows the user to see the top pushed Data or URL page (called an alert) in the queue that was sent to their Versity phone. It also sends notifications to the Android notification manager to alert the user of the number, and highest priority, of un-dismissed alerts.

It contains the following elements:

- Title bar showing the HTML <title>
- A button to dismiss the alert from the queue
- A progress bar showing the web page loading progress

- Optionally shown softkeys (Btn1-4). See PolySoftKey.

Alertview example

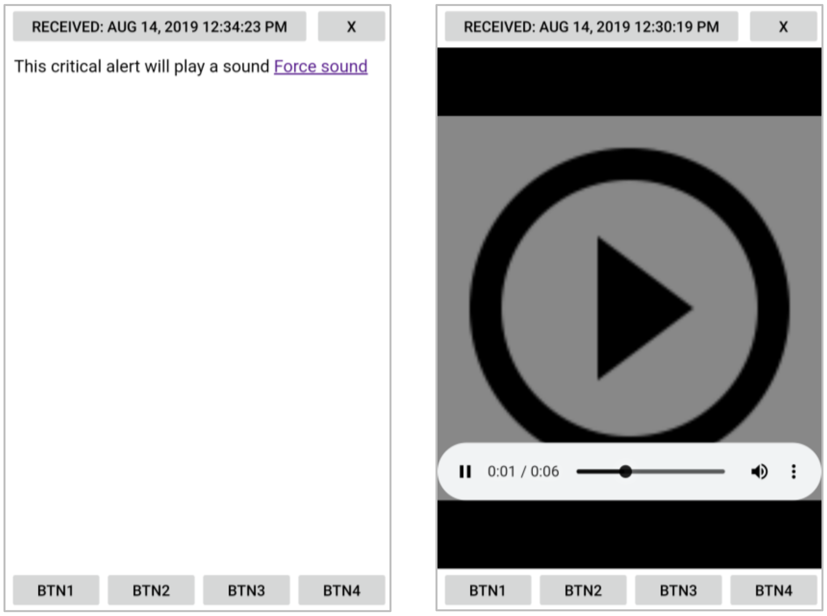


RECEIVED: AUG 14, 2019 12:28:13 PM

High 1

Received:
Aug 14, 2019 12:28:13 PM

Alertview that plays a sound. Force sound link opens a player



RECEIVED: AUG 14, 2019 12:34:23 PM

This critical alert will play a sound Force sound

RECEIVED: AUG 14, 2019 12:30:19 PM

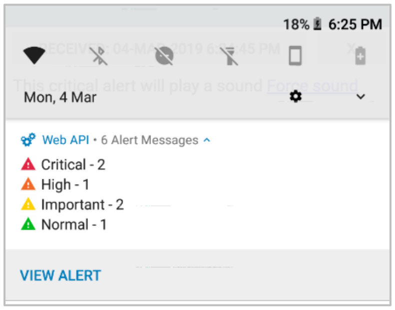
How the Spectralink Alertview is used

When a new push (Data or URL) is received by the phone, the Spectralink Alertview does the following:

- Stores the push in the queue according to priority: Highest priority first and then ordered by received time - most recent first. See Using Push Requests.
- Sends a notification to the Android notification manager to cause a notification to show up in the Notification Bar at the top of the phone.
- Plays the incoming alert notification tone, depending on the state of the phone. (This is documented in detail in Push Settings.)
- Determines whether to display the alert. If the Alertview was already on screen and the incoming alert is higher on the queue than the currently displayed alert, the new alert will replace the currently displayed alert. If the Alertview was not on screen and the new alert has critical priority, it will be displayed.

The user will always only see the top alert on the queue. If they wish to move to the next message on the queue they must dismiss the currently shown message. Once dismissed, alerts are gone and cannot be retrieved

.Alertview notification example of multiple alerts



Web API • 6 Alert Messages

Critical - 2
High - 1
Important - 2
Normal - 1

Once in the Alertview, a user can interact with your web application in the same manner as if they initiated the connection from an App URL shortcut.

Webview Applications

The user can launch web applications from the Spectralink App URL's widget box. When the user selects a web application by tapping its widget icon, the Spectralink Webview displays and the title bar shows a progress indicator as the page loads.

Handset behavior during Webview functions

While the user is in a Spectralink Webview, if there is an event in the phone application that requires the user's attention (such as an incoming phone call), the incoming phone call activity displays automatically in the foreground and the Spectralink Webview is placed on the recent activity list, just like on a consumer smartphone. The user can return to the Spectralink Webview by selecting it off the recent activity list, or by pressing the Back key.

NOTE

Webview display and other activities are not always interrupted

Not all events require the user's attention to the display, for example, Push-to-talk audio traffic plays out the rear speaker without interrupting the current activity's display.

Handset Configuration

The smartphones are configured to reach and interact with the web application through their Web API configuration, which must be set accordingly. Configuration will include:

- The smartphones must be configured to find the web application. At a minimum you will need to configure a URL location for the application itself and a label to display on the web app shortcut widget, which the user uses to initiate the application
- The smartphone must be configured to send required event notifications to the application server, minimally to notify the application of its IP address when it comes onto the wireless network.
- When applicable, the smartphone must be configured to receive incoming pushes for the application to send alerts to the smartphone.
- Other parameters can be required if your application requires telephony features such as personal alarms or emergency dial. The *Spectralink Application Management Guide* will provide you will full details about Spectralink application deployment.

Configuration information

You will need to set up your test phones with the parameters your web application requires to test your application. While you can use a full-blown Spectralink Application Management (SAM) server to configure the Web API settings in your test phones, we recommend that you simply use the manual method for a small number of phones. See *Spectralink Application Management Guide* for complete information about parameters to configure for test deployment.

Spectralink recommends testing your application against a fully functioning telephony environment as the final step of developing your application.

Web Applications and Third Party VoIP application

Some customers will decide to use a third-party Voice over IP apps to provide voice service instead of using the included Biz Phone application. Using 3rd-party VoIP applications, in effect bypassing Biz Phone, may have significant impact on the behavior of a web-based application. It is critical to understand the implications. This is described in Appendix A: Spectralink Versity Web API and Third-party VoIP Clients on the Versity Smartphones.

Chapter 3: Overview of the Spectralink Versity Web API

The Spectralink Versity Web API includes a powerful set of web-based application tools designed to integrate easily into almost any enterprise-grade application environment. There are several key functional API tools which will be fundamental elements of your application interface. The most valuable and necessary mechanisms are outlined in the sections that follow.

At a high level, the Spectralink Versity Web API uses the standard HTTP post mechanism to both send status from the phone and to allow external servers to send alert messages to the phone. The messages use XML as their syntax. The Web API also includes JavaScript callable DOM extensions inside the Spectralink Webview and Alertview. These extensions allow a web application to access capabilities not typically available within a web browser sandbox. These capabilities include playing audio files and initiating telephone calls.

The network flow diagrams provided in the following sections represent only one example of how an applications delivery platform might be architected. For example, in some instances, the web server and application server could be the same. Several of the functional blocks are indicated separately for clarity.

⚠ **CAUTION**

The API is case sensitive

Ensure you follow case guidelines exactly as any change may adversely affect the results.

See Telephony Integration for an in-depth explanation of each Web API function.

Push URL

The Push URL mechanism is typically used to create an application-generated message, alarm, or alert on the phone, to be displayed through the Spectralink Webview. This action results in the Webview receiving a URL address downloaded from the application web server.

The Push URL function supports the following two types of content URLs:

- **XHTML content URL**: The content will be displayed on Alertview. If it is not already open, it opens and displays the pushed message on Alertview.
- **URI actions content URL**: The URI actions specified in the file are executed on the smartphone.

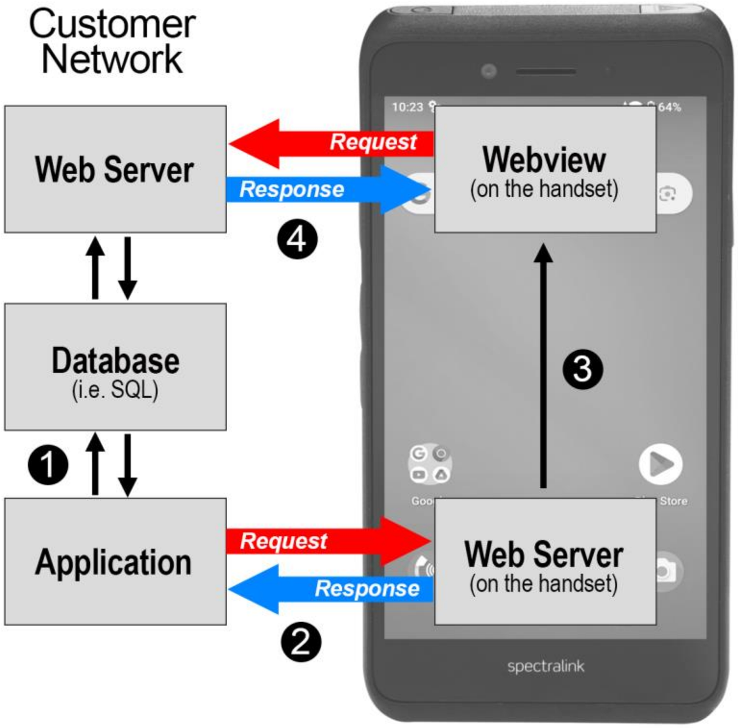
Push URL message flow

1 Customer web application queries the database for update to be sent to the handset.

2 The web application sends a URL push to the handset web server.

3 An internal trigger posts this URL to the Webview browser.

4 The Webview browser requests the URL from the customer web server, which is then displayed on the browser via the response.



Push Data

Instead of pushing a URL via the Push URL mechanism, you can push a small amount of HTML data directly from the application to the Spectralink Webview on the target smartphone(s).

This feature does not allow for URI actions in the message, but URI actions can be defined as anchors within the Push Data mechanism. This tool is useful to broadcast messages to a large group of users, avoiding the impact of many Webview browsers simultaneously requesting a URL from the web server, as would happen using the Push URL mechanism.

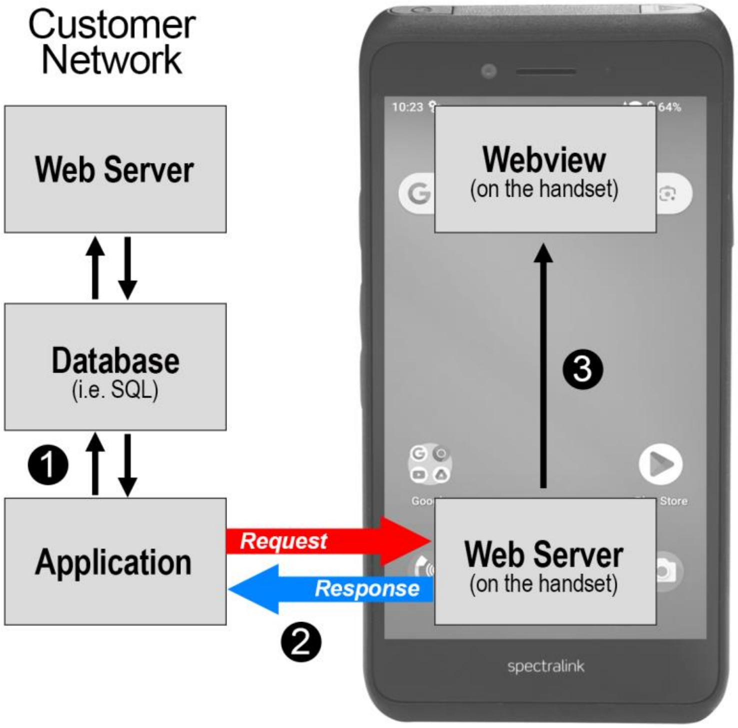
The general message flow is summarized as follows:

Push Data message flow

1 Customer application queries the database for update to be sent to the handset.

2 The web application sends a Data push to the handset web server (Request). The web server responds (Response) with an acknowledgement.

3 An internal trigger posts the HTML data to the Webview browser.



Internal URIs

Internal URIs are execution events that can be used for executing predefined actions in a specific scenario. It is similar to the manual execution of key presses. The smartphone executes Internal URIs in the order they are received. The URIs must be defined in sequence and separated by a ";" (semicolon) character or newline character and the file should be served with content type *application/x-com-spectralink-webx*. This file can be sent to a smartphone using a URL push message.

Phone State Polling

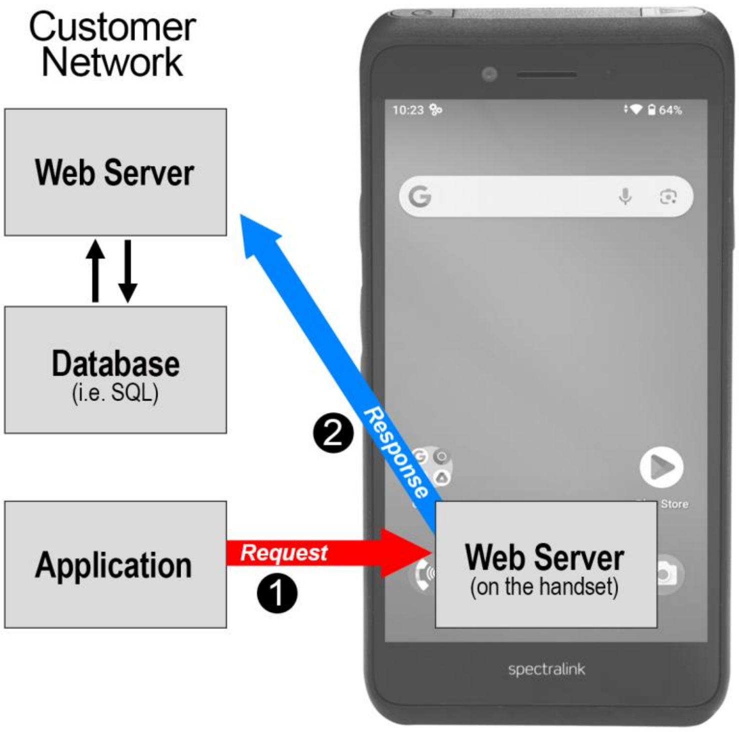
Phone state polling enables you to fetch the following smartphone configuration and call state information:

- Call Processing State
- Device Information
- Network Configuration

When the device receives any of these polling requests, it prepares the information in an XML format and sends it to the configured polling URL, or to the device that requested the Poll, depending on the state of the State Polling Response Method.

State Polling message flow

1. The application sends a poll request to the handset.
2. The handset responds with the requested information.



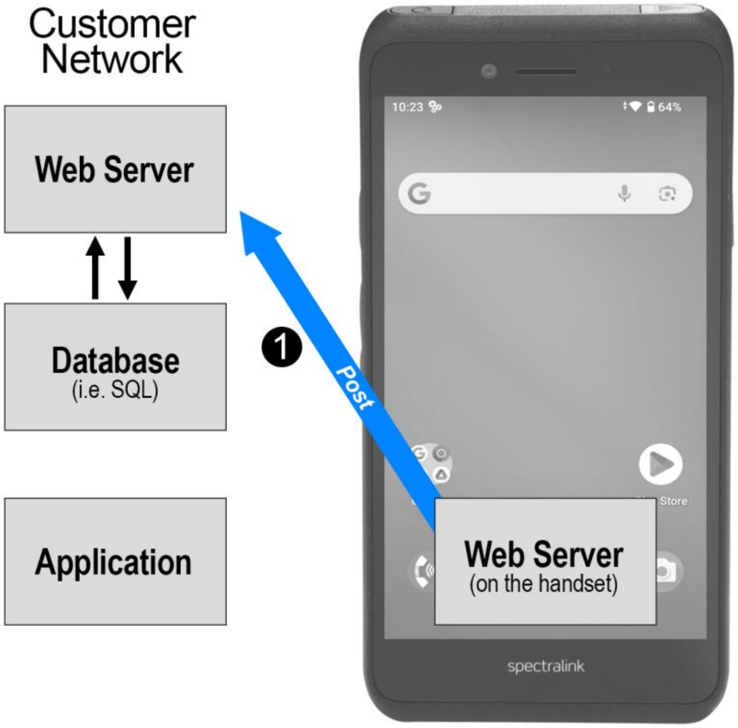
Event Notification

The smartphone-initiated event notification is based on a state change in the smartphone or a network connection event. This mechanism is used to integrate endpoint events into host application intelligence.

When an Event Notification is triggered it prepares the information in XML format and sends it to the configured event notification URL.

Push Notification Event

1 Post of state information in an XML tag group.



Chapter 4: Telephony Integration

To fully utilize the power of the Spectralink Versity Web API on a Spectralink smartphone, you will need to understand what the telephony functions are and how to write a program that utilizes them. Additionally, you will need to understand how to configure the smartphone settings to work with your application. This chapter covers the telephony functions that you can use and how the Push requests, Event notifications and Phone state polling functions operate. Configuration to enable these features is covered in the next chapter.

SPECTRALINK
Biz Phone only

Telephony integration is designed for the Spectralink Biz Phone application. All integration features detailed here are only for use with the Biz Phone app in a Wi-Fi environment. They will not work with cellular dialing or third-party phone applications.

TIP
Examples have wrapped lines

Be aware that the lines of code shown in this document are formatted to fit the page and may appear wrapped. If you cut and paste these lines, they may inadvertently contain line breaks. Check for valid code before executing.

Telephone Integration URIs

Internal Uniform Resource Identifiers (URIs) provide the interface to execute predefined actions on the phone. These actions give you as a developer action to some internal functions that normally would take manual user action to perform.

There are two (2) ways to execute an internal URI action, as follows:

- The internal URIs can be sent as Data Push where content type must be:

```
application/x-com-spectralink-webx
```

- If an XHTML file will include internal URI, they must be defined in (and executed from) anchor tags, in the href attribute (for example, <a href=tel:1234>Menu</a>). When the user selects the anchor, the action is processed and executed (in this case, dial phone number 1234).

ADMIN TIP
Executing Internal URIs

Internal URI actions contained in a file with content type “application/x-com-spectralink-webx” can be executed only through a URL push.

Use the following format when configuring the internal URIs:

```
ActionType:Action
```

where:

- ActionType is the type of action to execute (Tel, or Play)
- Action is the name/content of the action to be executed.

The supported internal URIs are described in the table shown next.

Supported Internal URIs

Action Type	*Action*
Tel[2, 3]	**Tel:[numbertodial]**
The Tel URI initiates a new call to the specified number. Any digit map rules are followed.	
Play[4]	**Play:<audiofile_path>**
Download and play the audio file. The <audiofile_path> is the relative path on the application server, relative to the Server Root URL.	

[2] The LineIndex value is case insensitive.

[3] If there are already 4 calls in progress the tel: URI request is ignored.

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

Keep in mind that the following important notes regarding internal URIs:

- The action name and key type are not 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 between URIs.
- If any URI is invalid and it is in a file of only internal URIs, the entire file is rejected.
- If any invalid URI is present in a XHTML file in an anchor tag, the execution of that URI is ignored.

NOTE

Placing a call on registration 2

Registration 2 is commonly used for in-house call systems that use registration 2 to call an extension for alerts and other messages. For example, use **tel:Reg2:2002** to place a call on registration 2 to extension 2002.

TIP

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 when specified after postd= as shown in the example below.

Example: Place a call to ***50**, and then wait two seconds before entering **44**:

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

TIP

Audio files format for Push messages

Ensure you specify the file format extension of any audio files, e.g. alertsoundA**.**wav. The preferred supported file format is WAV, G711 mu law or A law, 8KHz sampling rate, 8 bits per sample, monaural (aka mono). Audio software such as Audacity® can be used to create sound files of the correct format.

LTE and Wi-Fi variations when using Biz Phone

Spectralink has created two variations of Versity smartphones---Wi-Fi only and LTE (96xx) models. Both of these variations can use the Spectralink Biz Phone application for Wi-Fi but they handle PushURI tel: pushes differently.

ADMIN TIP

How to deploy Versity 96 Series LTE models

The summary offered here does not represent a full briefing on deployment of LTE devices. Please refer to the *Spectralink Versity Smartphone LTE Carrier Interoperability Guide*.

Use case: The push request contains a tel: link that the user taps to place a call.

- When using Biz Phone (SIP) the call is routed through Biz Phone on both Wi-Fi only and LTE models.
- On an LTE phone, if Biz Phone SIP is disabled, the call is routed through the Google cellular dialer app.
- On an LTE phone, if Biz Phone SIP is disabled and you have more than one other calling app on the phone, a chooser is displayed to allow you to select the desired app.
- On a Wi-Fi only phone, if Biz Phone SIP is disabled, Web API does not route the call anywhere.
- On a Wi-Fi only phone, if Biz Phone SIP is disabled but you have a different calling appon the phone, the call is routed through that app.

Using Push Requests

A push request is defined as an XML formatted request that 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, depending on the content of the XML.

SETTINGS

Push request parameters

See Push Settings for a list of parameters you can use to enable push requests in the smartphone.

Versity will convert PUSH request URLs to lower-case, so in effect the device will attempt to retrieve webpages and files using lower-case.

CAUTION

When Push does nothing

If a phone is in call and is sent a tel: push request that is with priority: high, normal or important), the phone accepts the push but does nothing. Only If the priority is critical will the call be placed immediately.

HTTP <URL> Push

The HTTP URL push enables an application to push a URL to a phone for the App URLs to open, such as an HTML Web page for display. The value sent within the push request is ‘relative’ because it is relative to the URL configured by the **Server root URL** parameter (the pushed URL is appended to this ‘root’ URL, and this is what the App URLs will attempt to open). This feature is asynchronous, because once the push request is received by the smartphone, it returns a 2xx or 4xx response immediately without waiting. There will be no success/failure feedback for the push handling itself. The pushing application will not know if the App URLs was able to open the pushed URL or not. The server that sends the requested page will know because it will see the page requests from the App URLs.

Use the following format when configuring the HTTP URL Push:

```
<URL priority=’X’ volume=’Y’ vibrate=’V’ ringtone=’R’ duration=’D’
notifyExternalApp=’A’ notify=’N’>URI path</URL>
```

The URL push requests support the attributes listed in the table shown next.

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 the next table. Priority must be all lower case: **priority.** The value must be enclosed in single quotes (‘).	

Attribute	*Permitted Values*
URI path[2]	**String**
Any relative URI (or relative URI path) on the configured application server.	
volume	**0 to 100**
Sets an override volume for any custom alert tone embedded in the page. (See PolyUri for more information on custom embedded alert tones.)	
vibrate	**true, false**
For internal Play: URLs, specifies whether the phone should vibrate when the notification is generated. Default value is true.	
ringtone	**String**
For internal Play: URLs, specifies the notification tone to play when the notification is generated. This must correspond to a "notification tone" installed on the phone. If not specified or if the tone is not found, then the default notification tone is played.	
duration	**Integer**
For internal Play: URLs, specifies the amount of time to vibrate (in milliseconds) when the notification is generated. If not specified, the phone will vibrate 2 times for 250 milliseconds with a 250 millisecond pause between vibrations. If < 1000, then the phone will vibrate for the supplied number of milliseconds. If the >= 1000, then the phone will vibrate in one second bursts with a one second pause between vibrations up to the amount of time specified.	
notifyExternalApp	**Android package name** **(e.g., com.spectralink.phone)**
Provides a mechanism to wake up a 3rd party app at which point it can query its external server to check for pending messages. The attribute must be in camel case as shown and the value (which is mandatory and must be enclosed in single quotes) is the package name of the application which needs to be notified. The app must register a broadcast receiver for the intent with action com.spectralink.intent.action.WEB_API_UPDATE with spectralink.permission.WEB_API_UPDATE permission to receive the broadcast. It may use an implicit or explicit broadcast registration.	
notify	**true, false**
Provides a mechanism to specify whether a pop-up notification is shown for the alert when the priority is critical. If the priority is not critical this attribute has no effect, since non-critical alerts do not show a pop-up notification. When set to true, a pop-up notification will be shown on the screen to the user in addition to being available in the notification bar on critical alerts. When set to false, the pop-up notification will not be shown, but the notification will still be available in the notification bar.	

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

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

CAUTION

Priority comes first

The order of the priority and volume setting must adhere to the order shown in the example with priority first, followed by volume. Also note that the volume value must have single quotes and the priority value must have single quotes.

ADMIN TIP

Where tags are defined

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

```
<SpectralinkIPPhone>
    <URL priority=’Normal’ volume=’100’ >/examples/media.xhtml</URL>
</SpectralinkIPPhone>
```

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

How Priority Affects URL and HTML Push Requests

Phone State	*Priority*	*Description*
Locked/Unlocked with screen off	Critical	The phone will display push request immediately: Notification sound will play, Notification in Notification Area will show, Screen wakes up, Spectralink Alertview activity starts in the foreground displaying the push.
	High	The phone plays the notification tone and Notification in the Notification Area will show, however the screen does not wakeup.
	Important	The phone plays the notification tone and Notification in the Notification Area will show, however the screen does not wakeup.
	Normal	The phone plays the notification tone and Notification in the Notification Area will show, however the screen does not wakeup.
Unlocked with screen on, not in phone call	Critical	The phone will display push request immediately: Notification sound will play, Notification in Notification Area will show, Spectralink Alertview activity comes to the foreground displaying the Push
	High	The phone plays the notification tone and Notification in the Notification Area will show.
	Important	The phone plays the notification tone and Notification in the Notification Area will show.
	Normal	The phone plays the notification tone and Notification in the Notification Area will show.
Unlocked in phone call	Critical	If “Enable notification ringtone” is set to On (see Push Settings), the phone will display push request immediately: Notification sound will play mixed in with the phone audio, Notification in Notification Area will show, Spectralink Alertview activity starts in the foreground displaying the push.
	High	The phone does not play the notification tone and Notification in the Notification Area will show.
	Important	The phone does not play the notification tone and Notification in the Notification Area will show.
	Normal	The phone does not play the notification tone and Notification in the Notification Area will show.

CAUTION

Handset must be configured to accept Push messages

See Push Settings for the settings that are required for the smartphone to receive a push request. If these are not configured any push message sent to the smartphone will be discarded.

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

- The URL that the phone ultimately ends up fetching is a concatenation of the **Server root URL** and the URL sent in the Push URL message.

SETTINGS

Server root URL

Server root URL can be defined to be Null. See Push Settings for complete information.

- Push requests are displayed as 'first-in-first-out' except for noted in the table above.
- 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 **Server root URL** is configured in a phone to be http://1.2.3.4/apps then to push the display of a XHTML page *media.xhtml*, you would send the following XHTML.

```
<SpectralinkIPPhone>
    <URL priority='Normal'>/examples/media.xhtml</URL>
</SpectralinkIPPhone>
```

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

Data Push of Complex URLs

If a URL is pushed to the phone that contains an ampersand (&), the smartphone truncates the URL at the ampersand. Two workaround options are:

- Do a data push which instantly re-directs to the desired URL

```
<SpectralinkIPPhone>
    <Data priority='%s'><html><head><title>redirecting...
</title></head>
<body onload="window.location='http://12.34.56.78/
NOTIFY.HTML?DEVID=2105010250&EVENTID=%7BA71C2393-8276-4484-A0E5-
5666DA06A5C1%7D'">redirecting...
</body>
</Data>
</SpectralinkIPPhone>
```

- Add a hidden form with which to send the data: Applies to most cases (where you are not accessing the GET variables directly w/ JavaScript).

```
<form name="form" action="someServerPage" method="POST">
<input type="hidden" name="DEVID" value="2105010250" />
<input type="hidden" name="EVENTID" value="%7BA71C2393-8276-4484-A0E5-
5666DA06A5C1%7D" />
</form>
<a href="#" onclick="document.form.submit()">notify device</a>
```

Some additional logic would be required on the server to send the correct information (accessible via the POST header of the HTTP request) back with NOTIFY.HTML but it would vary with language / framework / use case.

As a bonus, POSTed requests are considered more secure than GET style requests which include variables visible in the URL.

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 request the XHTML.

Use the following format when sending the HTML Data Push:

```
<Data priority='X' volume=’Y’ vibrate=’V’ ringtone=’R’ duration=’D’
notifyExternalApp=’A’ notify=’N’>D</Data>
```

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

HTML Push Requests

Attribute	*Permitted Values*
priority[1]	**Critical, Important, High, Normal**
Sets the priority of the push (X in the above example), which determines how and when the URL is requested. Priority must be all lower case: **priority.** The value must have single quotes (‘).	
text	**Text in HTML format**
Any text (Y in the above example).	
volume	**0 to 100**
Sets an override volume for any custom alert tone embedded in the page. (See PolyUri for more information on custom embedded alert tones.) Volume must be all lower case: **volume.** The value must have single quotes (‘).	
vibrate	**true, false**
For internal Play: URLs, specifies whether the phone should vibrate when the notification is generated. Default value is true.	
ringtone	**String**
For internal Play: URLs, specifies the notification tone to play when the notification is generated. This must correspond to a “notification tone” installed on the phone. If not specified or if the tone is not found, then the default notification tone is played.	

Attribute	*Permitted Values*
duration	**Integer**
For internal Play: URLs, specifies the amount of time to vibrate (in milliseconds) when the notification is generated. If not specified, the phone will vibrate 2 times for 250 milliseconds with a 250 millisecond pause between vibrations. If < 1000, then the phone will vibrate for the supplied number of milliseconds. If the >= 1000, then the phone will vibrate in one second bursts with a one second pause between vibrations up to the amount of time specified.	
notifyExternalApp	**Android package name (e.g., com.spectralink.phone)**
Provides a mechanism to wake up a 3rd party app at which point it can query its external server to check for pending messages. The attribute must be in camel case as shown and the value (which is mandatory and must be enclosed in single quotes) is the package name of the application which needs to be notified. The app must register a broadcast receiver for the intent with action com.spectralink.intent.action.WEB_API_UPDATE with spectralink.permission.WEB_API_UPDATE permission to receive the broadcast. It may use an implicit or explicit broadcast registration.	
notify	**true, false**
Provides a mechanism to specify whether a pop-up notification is shown for the alert when the priority is critical. If the priority is not critical this attribute has no effect, since non-critical alerts do not show a pop-up notification. When set to true, a pop-up notification will be shown on the screen to the user in addition to being available in the notification bar on critical alerts. When set to false, the pop-up notification will not be shown, but the notification will still be available in the notification bar.	

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

CAUTION

Priority value comes first

The order of the priority and volume setting must adhere to the order shown in the example with priority first, followed by volume. Also note that the volume value must have quotes and the priority value must have single quotes.

ADMIN TIP

Where tags are defined

Tags must be defined under a `<SpectralinkIPPhone>` root tag:

- Data must have a capital D: **Data**
- Volume must be all lower case: **volume**
- Priority must be all lower case: **priority**

ADMIN TIP

Prioritization of Push URL and Push Data requests

Push URL and Push Data requests follow the same priority described in HTTP <URL> Push Table*: How Priority Affects URL and HTML Push Requests*.

ADMIN TIP

Handling referenced CSS files

When performing a data push, any referenced CSS files must be an absolute path. Therefore, with a data push, the serverRoot URL is not included in the CSS file path.

Example: To push the display of an important message:

```
<SpectralinkIPPhone>
    <Data priority='Important' volume=’100’> <h1> Fire Drill at 2pm </h1>
Please exit and congregate at your appropriate location outside </Data>
</SpectralinkIPPhone>
```

CAUTION

Handset must be configured to accept Push messages

See Push Settings for the settings that are required for the smartphone to receive a push request. If these are not configured you can push a message to the smartphone but it will be discarded.

Using Event Notifications

Event Notifications allow application programs insight into what smartphones are doing, their status and their network information. Using a combination of them will allow an application to detect the power up of phones and the state of the phones.

For example, using a combination of events and phone state polls can allow an application to detect that a phone has registered with the call server (Line Registration Event) and then get the phone’s extension number, model # and firmware version (Device Info phone state poll).

The phone can be configured to send information to a specific URI if one of the following Event Notifications occurs:

- Personal Alarm Event
- Incoming Call Event
- Outgoing Call Event
- Offhook Event
- Onhook Event
- Call State Change Event
- Line Registration Event
- Line Unregistration Event

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

ADMIN TIP

Support of second registrations

If you have configured a second registration for use by your application, you may use the <LineNumber> parameter to return information to differentiate the line.

ADMIN TIP

Where is the header tag?

The header tag in the XML that identifies a Versity event notification <SpectralinkIPPhone> is not present in the event notification responses.

Viewing a Personal Alarm Event

The Alarm Event can be used by a security application to record, track or otherwise manage an alarm event that has been triggered by the SAFE application. Alarm events occur when Running, Tilt, and No movement alarms are triggered and when Panic button (duress) calls are made.

Use the following format when viewing the alarm event:

```
<SafeEvent>

  <PhoneIP> </PhoneIP>
  <CellularIP> </CellularIP>
  <MACAddress> </MACAddress>
  <BSSID> </BSSID>
  <StillAlarm> </StillAlarm>
  <TiltAlarm> </TiltAlarm>
  <RunningAlarm> </RunningAlarm>
  <DuressAlarm> </DuressAlarm>
  <LineNumber> </LineNumber>
  <TimeStamp> </TimeStamp>
  <Latitude> </Latitude>
  <Longitude> </Longitude>

</SafeEvent>
```

Alarm Notification Event Attributes

Attribute	*Permitted Values*
Phone IP IP address of the phone. For example `172.24.128.160`	**IP address**
Cellular IP (96 Series only) Cellular IP address of the LTE phone if a working SIM card is installed.	**IP address**

Attribute	*Permitted Values*
Note: In a Wi-Fi environment, the phone will not "know" its cellular IP address until it needs to use LTE. Therefore, disable Wi-Fi to allow the phone to find the LTE network. The Cellular IP address will become available at that point and continue to be used when needed.	
MACAddress	**MAC Address**
MAC address of the phone. For example: `00907a0e0f37`	
BSSID	**MAC Address**
The MAC address of the AP the phone is currently using.	
Still Alarm	**0 = no alarm, 1 = alarm**
The current state of this alarm detector.	
Tilt Alarm	**0 = no alarm, 1 = alarm**
The current state of this alarm detector.	
Running Alarm	**0 = no alarm, 1 = alarm**
The current state of this alarm detector.	
Duress Alarm	**0 = no alarm, 1 = alarm**
The current state of this alarm detector.	
LineNumber	**0 or 1**
Returns 1 if an emergency call number is configured and 0 if no emergency call number is configured.	
TimeStamp	**time**
The date and time that the event occurred on the phone. For example: `2013-05-11T13:19:53-08:00`	
Latitude	**Current coordinates obtained from GPS** Na for Wi-Fi (no GPS)
LTE only (96 Series only) `example <Latitude>40.02565302689416</Latitude>`	
Longitude	**Current coordinates obtained from GPS** Na for Wi-Fi (no GPS)
LTE only (96 Series only) `example: <Longitude>-105.22406386251863</Longitude>`	

Viewing an Incoming Call Event

The Incoming Call Event can be used by an application to send metadata about the call to the phone in real time, or to allow the application to detect that the ~~user of the phone is busy~~.

Use the following XML format when viewing the incoming call event:

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

The incoming call event contains the attributes listed in the following table.

Incoming Call Event Attributes

Attribute	*Permitted Values*
Phone IP IP address of the phone. For example `172.24.128.160`	**IP address**
Cellular IP (96 Series only) Cellular IP address of the LTE phone if a working SIM card is installed.	**IP address**
MACAddress MAC address of the phone. For example: `00907a0e0f37`	**MAC Address**
CallingPartyName The name displayed in phone's ‘From’ label in 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, then IP address of the calling party is shown. For example: `sip:172.24.128.160`	**name**
CallingPartyNumber 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.	**number**
CalledPartyName The name displayed in phone's **To** label on 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.	**name**
CalledPartyNumber 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 nonregistered line, the IP address of the called party is shown.	**number**
LineNumber Returns 1 for SIP registration 1 and 2 for SIP registration 2.	**1 or 2**

Attribute	*Permitted Values*
TimeStamp	**time**
The date and time that the event occurred on the phone. For example: `2013-05-11T13:19:53-08:00`	

When the event notification URI is set and the incoming call event is enabled to gather information, the following example shows the transmitted data for a call between one registered and one unregistered line:

```
<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 can be used by an application to detect that the user of the phone is busy in a call.

Use the following XML format when viewing the outgoing call event:

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

</OutgoingCallEvent>
```

The outgoing call event contains the attributes listed in the following table.

Outgoing Call Event Attributes

Attribute	*Permitted Values*
Phone IP	**IP address**
IP address of the phone. For example:	

Attribute	*Permitted Values*
`172.24.128.160`	
Cellular IP (96 Series only)	**IP address**
Cellular IP address of the LTE phone if a working SIM card is installed.	
MACAddress	**MAC Address**
MAC address of the phone. For example: `00907a0e0f37`	
CallingPartyName	**name**
The name displayed in phone's **From** label in 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, then IP address of the calling party is shown. For example: `sip:172.24.128.160`	
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 phone's **To** label in 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 nonregistered 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 nonregistered line, the IP address of the called party is shown.	
LineNumber	**1 or 2**
Returns 1 for SIP registration 1 and 2 for SIP registration 2.	
TimeStamp	**time**
The date and time that the event occurred on the phone. For example `2013-05-11T13:19:53-08:00`	

Viewing an Offhook Event

The Offhook Event allows an application to see that the user is starting a call.

Use the following format when viewing the offhook event:

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

```
</OffHookEvent>
```

The offhook event contains the attributes listed in the following table.

Offhook Event Attribute

Attribute	*Permitted Values*
Phone IP	**IP address**
IP address of the phone. For example: `172.24.128.160`	
Cellular IP (96 Series only)	**IP address**
Cellular IP address of the LTE phone if a working SIM card is installed.	
MACAddress	**MAC Address**
MAC address of the phone. For example: `00907a0e0f37`	
LineNumber	**1 or 2**
Returns 1 for SIP registration 1 and 2 for SIP registration 2.	
TimeStamp	**time**
The date and time that the event occurred on the phone. For example: `2013-05-11T13:19:53-08:00`	

Viewing an Onhook Event

The Onhook Event notifies an application that the user has ended a call. This can be used for call logging information, for example.

Use the following format when viewing the onhook event:

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

The onhook event contains the attributes listed in the following table.

Onhook Event Attributes

Attribute	*Permitted Values*
Phone IP	**IP address**
IP address of the phone. For example `172.24.128.160`	

Attribute	*Permitted Values*
Cellular IP (96 Series only)	**IP address**
Cellular IP address of the LTE phone if a working SIM card is installed.	
MACAddress	**MAC Address**
MAC address of the phone. For example: `00907a0e0f37`	
LineNumber	**1 or 2**
Returns 1 for SIP registration 1 and 2 for SIP registration 2.	
TimeStamp	**time**
The date and time that the event occurred on the phone. For example: `2013-05-11T13:19:53-08:00`	

Viewing a Call State Change Event

The Call State Change event notifies the application of the different call states that can exist on the phone.

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

```
<CallStateChangeEvent CallReference=" " CallState=" ">
    <PhoneIP> </PhoneIP>
    <CellularIP> </CellularIP>
    <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>
```

The call state change event contains the attributes listed in the following table.

Call State Change Event Attributes

Attribute	*Permitted Values*
Phone IP IP address of the phone. For example: `172.24.128.160`	**IP address**
Cellular IP (96 Series only) Cellular IP address of the LTE phone if a working SIM card is installed.	**IP address**
MACAddress MAC address of the phone. For example: `00907a0e0f37`	**MAC Address**
LineNumber Returns 1 for SIP registration 1 and 2 for SIP registration 2.	**1 or 2**
TimeStamp The date and time that the event occurred on the phone. For example: `2008-07-11T13:19:53-08:00`	**time**
CallReference A unique identifier for a call.	**number**
LineKeyNum Used in polling to determine which registration is responding or if IP dialing has been used. Returns 1 for SIP registration 1 and 2 for SIP registration 2. Returns 4 for IP dialing.	**1, 2 or 4**
LineDirNum Phone number associated with line. For example: `1234`	**phone number**
LineState The line state.	**OK,** [any value that is not "OK" indicates a registration error that will be specific to the PBX.]
CallState The call state.	**Outgoing call states: Setup, RingBack** **Incoming call states: Offering** **Outgoing/Incoming call states: Connected, Disconnected** **CallConference, CallHold, CallHeld, CallConfHold, CallConfHeld**
CallType The call type.	**Incoming, Outgoing**
UIAppearanceIndex The call appearance index. This number simply shows the order of the call appearance on the display.	**string**
CalledPartyName If the line is registered, the value is the registered line display name.	**name**

Attribute	*Permitted Values*
If the line is not registered, the value is the IP address of the called party.	
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 Change example of offering state for line 2

```
    <CallStateChangeEvent  CallReference="2" CallState="Offering">
       <PhoneIP>172.29.101.24</PhoneIP>
       <MACAddress>00907a13b900</MACAddress>
       <LineNumber>2</LineNumber>
       <TimeStamp>2015-07-14T14:37:33-0600</TimeStamp>
       <CallLineInfo>
           <LineKeyNum>2</LineKeyNum>
           <LineDirNum>4547</LineDirNum>
           <LineState>OK</LineState>
           <CallInfo>
               <CallState>Offering</CallState>
               <CallType>Incoming</CallType>
               <UIAppearanceIndex>2</UIAppearanceIndex>
               <CalledPartyName>4547</CalledPartyName>
               <CalledPartyDirNum>4547</CalledPartyDirNum>
               <CallingPartyName>LizAvayaSIP3</CallingPartyName>
               <CallingPartyDirNum>4520</CallingPartyDirNum>
               <CallReference>2</CallReference>
               <CallDuration>0</CallDuration>
           </CallInfo>
       </CallLineInfo>
    </CallStateChangeEvent >
```

Viewing a Line Registration Event

The Line Registration Event fires whenever a phone registers one of its lines to a call server. This can be used for a number of purposes but is a useful event flagging the fact that the phone is up and running on the network. Note that this event is only sent at the first registration and not when the phone refreshes an existing registration.

Use the following format when viewing the line registration event:

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

The line registration event contains the attributes listed in the following table.

Line Registration Event Attributes

Attribute	*Permitted Values*
Phone IP IP address of the phone. For example: `172.24.128.160`	**IP address**
Cellular IP (96 Series only) Cellular IP address of the LTE phone if a working SIM card is installed.	**IP address**
MACAddress MAC address of the phone. For example: `00907a0e0f37`	**MAC Address**
LineNumber Returns 1 for SIP registration 1 and 2 for SIP registration 2.	**1 or 2**
TimeStamp The date and time that the event occurred on the phone. For example: `2013-05-11T13:19:53-08:00`	**time**

Example of line registration event: LineNumber 1

```
<LineRegistrationEvent>
       <PhoneIP>172.29.101.24</PhoneIP>
       <MACAddress>00907a13b900</MACAddress>
       <LineNumber>1</LineNumber>
       <TimeStamp>2015-07-14T13:16:28-0600</TimeStamp>
    </LineRegistrationEvent>
```

Example of line registration event: LineNumber 2

```
<LineRegistrationEvent>
       <PhoneIP>172.29.101.24</PhoneIP>
       <MACAddress>00907a13b900</MACAddress>
       <LineNumber>2</LineNumber>
       <TimeStamp>2015-07-14T13:16:30-0600</TimeStamp>
    </LineRegistrationEvent
```

Viewing a Line Unregistration Event

The line unregistration event can be useful for determining when a phone is powered off or otherwise no longer available on the network. However, the event only fires if the phone is gracefully shutdown or restarted. However, if the phone experiences a power loss (e.g. battery pack removal), the event will not be fired, so it cannot be relied on.

Use the following format when viewing the line unregistration event:

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

The line unregistration event contains the attributes listed in the following table.

Line Unregistration Event Attributes

Attribute	*Permitted Values*
Phone IP IP address of the phone. For example: `172.24.128.160`	**IP address**
Cellular IP (96 Series only) Cellular IP address of the LTE phone if a working SIM card is installed.	**IP address**
MACAddress MAC address of the phone. For example: `00907a0e0f37`	**MAC Address**
LineNumber Use 1 for SIP registration 1 and 2 for SIP registration 2	**1 or 2**
TimeStamp The date and time that the event occurred on the phone. For example: `2013-05-11T13:19:53-08:00`	**time**

Example of line unregistration event: LineNumber 1

```
<LineUnregistrationEvent>
       <PhoneIP>172.29.101.24</PhoneIP>
       <MACAddress>00907a13b900</MACAddress>
       <LineNumber>1</LineNumber>
       <TimeStamp>2015-07-14T13:15:11-0600</TimeStamp>
    </LineUnregistrationEvent>
```

Example of line unregistration event: LineNumber 2

```
<LineUnregistrationEvent>
       <PhoneIP>172.29.101.24</PhoneIP>
       <MACAddress>00907a13b900</MACAddress>
       <LineNumber>2</LineNumber>
       <TimeStamp>2015-07-14T13:15:38-0600</TimeStamp>
    </LineUnregistrationEvent>
```

Viewing a Login/Logout Event

The Login/Logout Event can be used by the Biz Phone app when SIP is enabled and there are multiple users using the same phone and each has a different extension at the same SIP server address. To use the phone, each user must login by entering unique credentials in a popup window. This is the Login event. A Logout option is provided in the app menu. Tap Logout to end the session and the Logout event is captured.

Use the following format when viewing the Login/Logout event:

```
  <UserLogInOutEvent
     <PhoneIP> </PhoneIP>
     <CellularIP> </CellularIP>
     <MACAddress> </MACAddress>
     <CallLineInfo>
          <LineKeyNum> </LineKeyNum>
          <LineDirNum> </LineDirNum>
     </CallLineInfo>
     <UserLoggedIn />
     <TimeStamp> </TimeStamp>
  </ UserLogInOutEvent
```

The Login/Logout event contains the attributes listed in the following table.

Login/Logout Event Attributes

Attribute	*Permitted Values*
Phone IP IP address of the phone. For example `172.24.128.160`	**IP address**
Cellular IP (96 Series only) Cellular IP address of the LTE phone if a working SIM card is installed.	**IP address**
MACAddress MAC address of the phone. For example: `00907a0e0f37`	**MAC Address**

Attribute	*Permitted Values*
LineKeyNum	**1, 2 or 4**
Used in polling to determine which registration is responding or if IP dialing has been used. Returns 1 for SIP registration 1 and 2 for SIP registration 2. Returns 4 for IP dialing.	
LineDirNum	**phone number**
The phone number associated with line. For example: `1234`	
TimeStamp	**time**
The date and time that the event occurred on the phone. For example: `2013-05-11T13:19:53-08:00`	

Example of LogInOut event:

```
<UserLogInOutEvent>
     <PhoneIP>172.29.101.148</PhoneIP>
     <CellularIP> </CellularIP>
     <MACAddress>00907AA7DDAF</MACAddress>
     <CallLineInfo>
          <LineKeyNum>1</LineKeyNum>
          <LineDirNum>5007</LineDirNum>
     </CallLineInfo>
     <UserLoggedIn />
     <TimeStamp>2018-09-04T09:21:53-0600</TimeStamp>
</UserLogInOutEvent>
```

```
<UserLogInOutEvent>
     <PhoneIP>172.29.101.148</PhoneIP>
     <CellularIP> </CellularIP>
     <MACAddress>00907AA7DDAF</MACAddress>
     <CallLineInfo>
          <LineKeyNum>1</LineKeyNum>
          <LineDirNum>5007</LineDirNum>
     </CallLineInfo>
     <UserLoggedOut />
     <TimeStamp>2018-09-04T09:11:52-0600</TimeStamp>
</UserLogInOutEvent>
```

Phone State Polling

The phone can be configured to send the current state information to a specific URI or to the requestor upon receipt of an HTTP Phone State Poll request.

The following types of information can be sent:

- **Receiving Call Line Information** The line registration and call state will be sent upon receipt of an HTTP request to the call state handler (http://<Phone_IP>/polling/callstateHandler).
- **Receiving Device Information** Device specific information will be sent upon receipt of an HTTP request to the device handler (http://<Phone_IP>/polling/deviceHandler).
- **Receiving Network Configuration** Network specific information will be 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 or to the sender of the request.

> **SETTINGS**
> ***Phone State Polling parameters***
>
> See State Polling for a list of parameters you can use to enable state polling in the smartphone.

Phone state polling is used to determine if a given smartphone is currently online. For example, you could send a phone state poll and wait for a response (5 sec). You can also determine of the smartphone is in a call, or what code revision it has loaded.

Receiving Call Line Information

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

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

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

ADMIN TIP

When the call info block is defined

The `<CallInfo>` block is included if and only if `<LineState>` is **OK**. Otherwise it is not included. [For Line State, any value that is not “OK” indicates a registration error that will be specific to the PBX.]

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

Call Line Information Message Attributes

Attribute	*Permitted Values*
LineKeyNum	**1, 2 or 4**
Used in polling to determine which registration is responding or if IP dialing has been used. Returns 1 for SIP registration 1 and 2 for SIP registration 2. Returns 4 for IP dialing.	
LineDirNum	**phone number**
The phone number associated with line. For example: `1234`	
LineState	**OK** [any value that is not “OK” indicates a registration error that will be specific to the PBX.]
The line state.	
CallState	**Outgoing call states: Setup, Ringback** **Incoming call states: Offering** **Outgoing/incoming call states: Connected, Disconnected** **CallConference, CallHold, CallHeld, CallConfHold, CallConfHeld**
The call state.	
CallType	**Incoming, Outgoing**
The call type.	
UIAppearanceIndex	**string**
The call appearance index. This number simply shows the order of the call appearance on the display.	
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`	

Attribute	*Permitted Values*
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`	

CallReference	**number**
An internal identifier for the call.	
CallDuration	**number, seconds**
The duration of the call in seconds.	

Receiving Device Information

Applications can use the Device Information to do things like device firmware tracking/management as well as asset tracking.

The Device Information message is returned in the following format:

```
<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.

Device Information Message Attributes

Attribute	*Permitted Values*
MACAddress	**MAC Address**
The MAC address of the phone. For example, `00907a0e0f37`	
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 Android version ID on the phone. For example `8.1.0.1.5.0.2386`	

Attribute	*Permitted Values*
BootROM The BootROM on the phone. `e.g, L9Q15000TA00`	**string**
ModelNumber The phone's model number. Versity 9540 == Wi-Fi no scanner Versity 9553 == Wi-Fi + scanner Versity 9640 == Cellular (+Wi-Fi) no scanner Versity 9653 == Cellular (+Wi-Fi) with scanner Versity 9740 == Wi-Fi no scanner Versity 9753 == Wi-Fi + scanner	**string**
TimeStamp The date and time that the event occurred on the phone. For example: `2013-05-11T13:19:53-08:00`	**time**

Receiving Network Status

The Network Configuration message returns the specific network information about the phone.

The Network Configuration message is returned in the following format:

```
<NetworkConfiguration>
    <DHCPServer> </DHCPServer>
    <MACAddress> </MACAddress>
    <DNSSuffix> </DNSSuffix>
    <IPAddress> </IPAddress>
    <CellularIP> </CellularIP> [96 Series only]
    <SubnetMask> </SubnetMask>
    <ProvServer> </ProvServer>
    <DefaultRouter> </DefaultRouter>
    <DNSServer1> </DNSServer1>
    <DNSServer2> </DNSServer2>
    <DHCPEnabled> </DHCPEnabled>
</NetworkConfiguration>
```

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

Network Configuration Message Attributes

Attribute	*Permitted Values*
DHCPServer The DHCP server IP address. For example, `192.168.1.1`	**IP address**

Attribute	*Permitted Values*
MACAddress The MAC address of the phone. For example, `00907a0e0f37`	**MAC Address**
DNSSuffix The DNS domain suffix. For example `spectralink.com`	**host name**
IPAddress The IP address of the phone. For example `192.168.1.5`	**IP address**
Cellular IP (96 Series only) Cellular IP address of the LTE phone if a working SIM card is installed.	**IP address**
SubnetMask The subnet mask: For example `255.255.255.0`	**IP address**
ProvServer The IP address of the CMS server or a host name, if defined. For example `192.168.1.10`	**IP address**
DefaultRouter The IP address of the default router (or IP gateway). For example `192.168.1.1`	**IP address**
DNSServer1 The configured IP address of DNS Server 1. For example `192.168.1.250`	**IP address**
DNSServer2 The configured IP address of DNS Server 2. For example `192.168.1.250`	**IP address**
DHCPEnabled If DHCP is enabled, set to `Yes`	**Yes, No**

Chapter 5: Writing Your Web Application

Supported Standards

The Spectralink App URLs supports true Spectralink Versity applications—nearly indistinguishable from a desktop application, provides immediate feedback and updates information without a deliberate refresh—with the following features:

- HTML 5 – no video
- CSS 3.0 – only one active transition / animation at a time.
- SVG 1.1
- JavaScript. Supports ECMA-262 with extensions.
- XMLHttpRequest
- HTTP 1.1
- AJAX

HTTP Support

The App URLs is a fully compliant HTTP/1.1 user agent as described in RFC 2616. For more information, see http://www.ietf.org/rfc/rfc2616.txt?number=2616.

The App URLs supports:

- **Cookies** Cookies are stored in the flash file system; they are preserved when the phone reboots or is reconfigured.
- Refresh headers
- HTTP proxies
- **HTTP proxy authentication** The phone's login credentials or the user's name and password can be used to authenticate the user with the server.
- **HTTPS by HTTP over TLS** The App URLs will support the TLS protocol v1 only. It is not backward compatible with SSL v2 or SSL v3.
- Custom CA certificates

Using JavaScript DOM Extensions

The Spectralink App URLs and Spectralink Alertview provide access to phone-specific Document Object Model (DOM) JavaScript extensions. The DOM is created by the App URLs after parsing an XHTML file. JavaScript's primary role in the App URLs is to modify properties of the DOM. The DOM is a collection of every object defined in the XHTML, for example, every button, every label, and every image. A web application can use JavaScript to modify DOM properties just like any other XHTML object.

PolySoftKey

The PolySoftKey DOM object provides control over the soft keys in the App URLs. You can use it to hide or show the default or custom defined soft keys and to respond to soft key presses performed by the user.

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

- **PolySoftKey.connect(“{function}”)** Connects the JavaScript function supplied to the callback that is made when a custom soft key was pressed (refer to the example below)
- **PolySoftKey.setSoftkeyLabel(int, “string”)** Used to set the label of a given custom soft key (0 to 3) to string.
- **PolySoftKey.hideToolBar()** Allows the application to hide the soft key toolbar
- **PolySoftKey.showToolBar()** Brings back the soft key toolbar
- **PolySoftKey.resetAllDefaults()** Clears all custom defined key labels
- **PolySoftKey.resetDefaultKey(int)** Clears custom key label (0 to 3)
- **PolySoftKey.enableBack(int, “string”)** Labels the key (0 to 3) with the string. When the user presses it, the brower’s back button is invoked.**PolySoftKey.enableForward(int, “string”)** Labels the key (0 to 3) with the string. When the user presses it, the brower’s forward button is invoked.
- **PolySoftKey.enableRefresh(int, “string”)** Labels the key (0 to 3) with the string. When the user presses it the browser’s refresh button is invoked.
- **PolySoftKey.clearMessages(int, “string”)** Labels the key (0 to 3) with the string. When the user presses it, the Webview is closed

The PolySoftKey custom DOM extension example is shown next.

Example: PolySoftKey DOM Extension

```
PolySoftKey.connect(“skCallBack”);

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

function skCallBack(key, skEvent){
   if (skEvent.indexOf("pressed") != -1){  // ignore the “released” event
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;
   } // if
}

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

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

ADMIN TIP

Key press events

A user pressing a softkey will generate two key events, pressed and released. And accordingly a connected Javascript softkey callback function will be called twice. Often the keypress only needs to be handled once, so one approach is to act off just the “pressed” or “released” string, e.g.

```
  if(skEvent.indexOf(“pressed”) != -1)
  {
  Document.getElementById(“demo”).innerHTML=”Key pressed”;
}
```

PolyUri custom DOM extension

The PolyUri custom DOM extension gives you a few general controls/notifications such as notification when the App URLs is hidden or shown, as opposed to other applications on the phone. It also allows you to push a URI (see Push URL) back to the phone—in a sort of loopback fashion—from a loaded Web page. This allows a push to play a custom embedded alert Wave file.

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

- **PolyUri.pushUri(string)** Enables you to push any Spectralink internal URI. For example, Play:: and Tel::)

The PolyUri custom DOM extension example is shown next.

Example: PolyUri DOM Extension

```
function onPageLoad(){
   // Pushes a play request whenever the page is loaded
  PolyUri.pushUri("play:http://123.45.67.890:8080/sounds/dingling.wav");
}
```

Chapter 6: Configuring the Parameters Required by the Spectralink Versity Web API

Handsets depend on certain settings for site-specific information. These settings are documented in the *Spectralink Application Management Guide* and can be configured on the smartphone's admin menu or through SAM. When using SMS, refer to *Spectralink Application Management Guide.*

The parameters described in this chapter include those for:

- Web applications
- Push requests
- Event notifications
- Phone state polling

Web API Settings

Web API settings enable the smartphone to display the label or name of your application in the web application shortcut widget point to the URL where the application resides.

There is a top-level Enable/Disable setting for the Web API. It must be enabled for the Web API features to function.

Web Application Shortcuts Settings

The phone can be configured to show up to 10 web application shortcuts in the web application shortcut widget. The settings are configured in pairs with a Shortcut title and Shortcut URL for each shortcut desired.

Web Application Shortcut Settings

Parameter	*Permitted Values*	*Default*
Shortcut title The descriptive text that displays in the Applications menu	**String**	**null**
Shortcut URL The URL of an application	**URL String**	**null**
The label and URL of up to 10 applications.		

State Polling Settings

The State Polling parameters are used to control state polling responses from the phone when it receives a poll request.

Phone State Polling Settings

Parameter	*Permitted Values*	*Default*
Authentication username	**string**	**null**
Enter the user name that the phone requires to authenticate phone state polling. This must be non-null for state polling to be functional.		
Authentication password	**string**	**null**
Enter the password that the phone requires to authenticate phone state polling. This must be non-null for state polling to be functional.		
response method	**Requester, URL**	**Requester**
The method of sending requested polled data. If URL, the requested polled data is sent to a configured URL. If Requester, the data is sent in the HTTP response.		
URL	**URL**	**null**
The URL to which the phone sends call processing state/device/network information, if the state polling response method is set to URL. The protocol used can be either HTTP or HTTPS.		

Push Settings

The push request parameters are used to control the behavior, security and allowed priorities of pushes to the phone.

> **SETTINGS**
> ***Enabling Data and URL Push***
>
> Both the push username and push password must be non-null for Data and URL Push to be enabled.

Push Settings

Parameter	*Permitted Values*	*Default*
Push authentication username	**string**	**null**
The username required to cause the phone to accept an incoming push Data/URL. Used with the username to respond to the MD5 HTTP Digest Challenge from the smartphone. Both the push authentication username and push authentication password must be non-null for Data and URL Push to be enabled.		
Push authentication password	**string**	**null**
The username required to cause the phone to accept an incoming push Data/URL. Used with the username to respond to the MD5 HTTP Digest Challenge from the smartphone. Both the push authentication username and push authentication password must be non-null for Data and URL Push to be enabled.		

Parameter	*Permitted Values*	*Default*
Push message priority	**All, Critical, High, Important, Normal, None**	**All**
Configures the allowed incoming priority push data/URL commands. (None) Discard all push messages (Normal) Allows only normal push messages (Important) Allows only important push messages (High) Allows only high priority push messages (Critical) Allows only critical push messages (All) Allows all priority push messages		
Server root URL	**URL**	**null**
The URL of the application server you enter here is combined with the pushed URL and sent to the phone's App URLs. For example, if the application server root URL is http://172.24.128.85:8080/sampleapps and the pushed URL is /examples/sample.html, the URL that is sent to the App URLs is http://172.24.128.85:8080/sampleapps/examples/sample.html. Can be either HTTP or HTTPS.		
Enable notification ringtone	**On/Off**	**Off**
If off, there is no sound when an alert is received, except for a possible custom embedded tone. If on, the phone's selected default notification sound is played.		

Event Notification Settings

Event notification settings are used to control what phone events are sent to what URL. An unlimited number of URLs can be configured to receive any combination of events, and a user readable name can be defined for the Event URL definition.

Event Notification Settings

Parameter	*Permitted Values*		*Default*
Name	**string**		**null**
A human readable name for the Event URL definition.			
Event URL	**URL string**		**null**
The URL where the event notification post will be sent. E.g. http://www.myserver.com/phone_event_handler.php			
Events to receive	**None** **All** **State Change** (phone) **Incoming** (phone call) **Registration** (SIP Line) **UnRegistration** (SIP Line)	**Off Hook** (phone) **On Hook** (phone) **Outgoing** (phone call) **Login/out** **CallState Connected** **CallState Disconnected**	**None**
Select which combination of events should be sent to the Event URL. None and All are exclusive, of course, but any combination of the other settings is allowed.			

Chapter 7: Debugging, Troubleshooting and Best Practices

The best App URLs for testing your app is the Spectralink Versity smartphone's built-in Spectralink App URLs. You next best option is either the Chrome™ browser or Safari®. They can be used to test rendering issues on the computer before testing them on the phone's Spectralink App URLs or Spectralink Alertview.

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

A useful debugging process is as follows:

1. Use Firebug (in Firefox) or 'Inspect' (in Chrome) to check for JavaScript errors.
2. User Firebug (in Firefox) or 'Inspect' (in Chrome) to check that all asynchronous requests are working properly.
3. Determine if there are server errors; if there are, use the generated error messages / Server logs to figure out the error.

Repeat this process until there are no errors.

Troubleshooting App URLs Application Errors

Pushed message is not getting displayed in Spectralink Alertview

Push message will be displayed in Spectralink Alertview based on the priority of the message. See HTTP <URL> Push. Another cause is if a URL is pushed to the phone that contains an ampersand (&), the phone truncates the URL at the ampersand. Format the URL differently or use AJAX to load additional information after the page is loaded.

Server Not Found

Usually occurs on the phone after a URL Push when the Server Root URL setting is set incorrectly and the phone cannot resolve the concatenated 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 additional page elements embedded in it that are not valid, the phone will first show a blank page with a very slow moving (or even stopped) progress bar and will eventually render only the elements it was able to retrieve. Check that the URLs for any additional page elements are correct and reachable by the phone (firewalls, VLANs, for example, can present barriers).

Best Practices during Web Application Development

As with any software development project, there are a range of approaches you can follow. If you are new to developing Spectralink Web applications, it may help to know a few tips to use and pitfalls to avoid before you begin. Use the following lists for guidance to the best practices to use when developing applications to run on the Spectralink App URLs and Alertview.

The following points apply when developing applications for the Spectralink App URLs and Alertview:

- **Using the HTTP User Agent** The application 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 and screen size and other capabilities. Applications running on phones that support the App URLs can also use JavaScript to detect the screen and/or window size.
- **Supported Image Formats** Spectralink Versity supports GIF, PNG, JPG, and BMP image formats. Where image size is a concern, compressed JPG images are better for large images. For smaller images, the BMP image format provides better quality but lacks the compression benefit.
- **Pushing Sensitive Data** Avoid pushing security sensitive data direct to the phone. A URL push can be used to push a request to the phone to get the information from a HTTPS site, so the data will be encrypted. The URL push itself should not leak sensitive information.
- **Using HTTPS for Event Notifications** You may want to use HTTPS for event notifications and state polling because they contain sensitive information such as the phone MAC address, caller name and phone number.
- **Implement a User Confirmation** When including push notifications, be sure to implement a user confirmation response. Adding a confirmation response will ensure the user actually viewed the notification.
- **Using Tel URI** Your application should use TelUri API to make phone calls.
- **Remove white space in code** If concerned about Data Push content length (must be <2kb) you may process HTML, JavaScript, and CSS files to remove whitespace and shrink before delivery.
- **Use lower case for PUSH requests** Versity will convert PUSH request URLs to lower-case, so in effect the device will attempt to retrieve web-pages and files using lower-case.

Notes on API Security

With respect to the security of the REST API, the following should be noted:

- **Authenticating remote control and monitoring** The execution of each HTTP PUSH request requires MD5 digest authentication which can be further secured inside HTTPS. All pushed URLs are relative URLs with the root specified in the smartphone's configuration.
- **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** The confidentiality of all events reported by the phone can be also be protected by TLS in the same way that push content is. Simply specify an HTTPS URL for the destination for Event Notifications.
- **Data push** When data push is enabled, content 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 wired network (wireless security will encrypt the traffic through the air). Spectralink recommends that you only use unencrypted data push for broadcast type alerts that do not pose any confidentiality risks.

Chapter 8: Testing

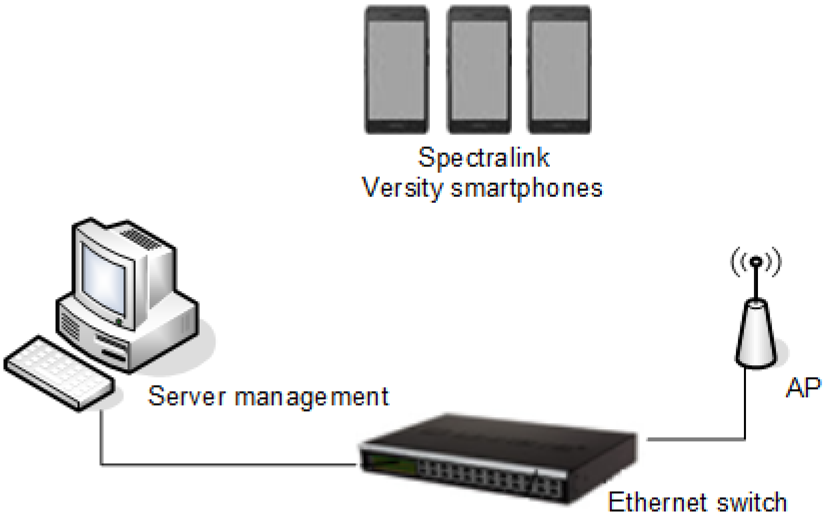
We recommend two levels of application testing, each with progressively more stringent requirements:

1. Using a controlled test environment with an Versity smartphone, web application server, and telephony server.
2. Using a fully functional system.

Setting up a Controlled Test Environment

A controlled test environment uses a Spectralink Versity smartphones working in a wireless LAN with a PC that is configured to function as a telephony server as well as, perhaps, providing all other server functions. At least one Spectralink VIEW Certified AP is required. This setup will give you adequate verification of the workability of your application before it is deployed in a working facility.

Controlled Test Environment



Test Hardware

Hardware to be purchased from Spectralink:

- Two Spectralink Versity smartphones
- Battery Packs, chargers, and power supplies for each smartphone

Hardware provided by participant:

- One 100/1000Mbit Switch
- VIEW Certified wireless LAN infrastructure AP (we recommend Cisco 1142 in Autonomous mode)
- PC to run DHCP server, and Syslog server
- PC running a virtual SIP PBX (usually the same PC)
- PC to run web application server (usually the same PC).

Test Software

Required software provided by participant

Software	*Description*
DHCP server	DHCP server
Packet analyzer software	Useful for debugging.
Virtual SIP PBX	Virtual SIP PBX software can be downloaded from various sites at no charge.

Setup Overview

Tests require the following setup, unless otherwise indicated.

1. Connect the network switch to the following (only one PC is needed):
   - ○ One wired LAN data PC
   - ○ One PC running virtual SIP PBX software
   - ○ One AP (the second AP will be added only when indicated in this plan)
   - ○ One wired LAN packet analyzer PC "spanning" port specific to the wired device of interest.
2. Associate the wireless data PC to the AP.
3. Register all of the Spectralink smartphones to the virtual SIP PBX. See Appendix B: Call Server Setup Example for an example setup.

Setting up the PC

The PC will serve several functions and each function must be configured:

- Wired data: configure the PC as a
  - ○ DHCP server
  - ○ Syslog server
- Virtual telephony call server (virtual SIP PBX)

- Wired and wireless packet analyzer using Wireshark or similar software

Wired data PC

The data PC will be used as the DHCP server, and Syslog Server. Attach the wired data PC to the network switch. Load all applicable server software.

DHCP server and Syslog Server setup

These are the usual functions. Instructions to set up these functions are not described in Spectralink documents.

Wired/Wireless packet analyzer setup

Attach Ethernet cable to the spanned monitor port on the switch or use a hub. Install Wireshark or similar packet analyzer with wired and wireless 802.11a/b/g/n capabilities.

Setting up the Versity smartphones

Handsets must be configured to associate with the wireless LAN and find the CMS from which they will download the code and s.

Full information is available in online references: *Spectralink Versity Administration Guide.*

Required settings

- Wi-Fi
  - Add the Wi-Fi network settings to match how you setup your View Certified AP
- Logging
  - Configure Syslog to point to your SYSLOG server
- SIP Phone
  - Configure your SIP server, SIP server port, Extension number, Username, Password, Audio DSCP value (0x2e) and Call Control DSCP value (0x28)
- Web API
  - Enable the API
  - Configure Phone state polling (see State Polling Settings).
  - Configure Push settings (see Push Settings).
  - Configure Event Notifications (see Event Notification Settings).
  - Configure Web application shortcuts to point to your app so the user can launch it (see Web Application Shortcuts Settings).

Conduct the Test

Once the hardware is set up and the files are downloaded and configured, you will be able to make calls and run the application.

Setting up a Working System Test

A working system test is done in close coordination with an existing installation. The telephone administrator will need the Web API settings you have customized for your application. Additionally, of course, the application itself and any application server must be configured.

Appendix A: Spectralink Versity Web API and Third-party VoIP Clients on the Versity Smartphones

As the Spectralink Versity smartphones are Android based and since they can run many different Voice over IP apps, some customers will decide to use one of these 3rd-party Voice over IP apps to provide voice service instead of using the included Spectralink SIP application.

Using 3rd-party VoIP applications, in effect bypassing the Spectralink SIP application, may have significant impact to the behavior of web based application. It is critical to understand the implications. But first, you should understand some of the interactions between the Spectralink SIP app and the Web API.

Spectralink Versity Web API and Spectralink SIP Application Dependencies

The Spectralink Versity Web API can send state and event information related to the Spectralink SIP telephony Application, including:

- Call State
- Call State Change
- Incoming Call
- Outgoing Call
- On Hook
- Off Hook
- Line Registration Complete
- Line Unregistration Complete
- Phone Extension Number (DN)

However as other 3rd party VoIP Applications do not communicate their internal state, the Web API cannot detect the above information. This may have significant impact to Web applications.

Important Workability Notes

The following parts of the Web API will not work, or will have reduced functionality, if the Spectralink SIP App is not used:

- The Call State Poll will not return any valid values.
- The Device Info State Poll will not return a valid Phone DN (Extension). All other information will be valid.
- No Notification Events will be send: Call State Change, Incoming Call, Outgoing Call, On Hook, Off Hook, Line Registration, Line Unregistration.

The following functionality in the Web API will still work, with some conditions:

- PushURI tel: pushes, and anchor tags in the Spectralink App URLs, will still initiate phone calls if the 3rd-Party VoIP client listens to the standard Android "place call intent". However, post connection dialing may not be supported by the 3rd Party Client.
- PushURI play: pushes may not play in the same way: mixing and volume levels may differ.

The following functionality in the Web API will work fully even if the Spectralink SIP App is not used:

- Push HTML Data
- Push HTML URL
- Network Info Phone State Poll
- Javascript DOM Extensions in the Spectralink App URLs/Alertview: including Softkeys and PolyBarcode

Web Application Shortcut Widget

Appendix B: Call Server Setup Example

NOTE

Trixbox is a deprecated product

A while ago Fonality acquired the trixbox product. As of this writing, Fonality has merged with NetFortris. The trixbox screens shown in the examples below is no longer available. The example screens serve only as an example of how you might go about setting up a generic call server and is not meant literally as a product recommended for installation.

The initial test setup relies on simple virtual devices to provide telephony services. We have used the trixbox® CE software from Fonality® to provide basic call server functionality. Installed on your PC, it will allow your PC to serve as a PBX to register the Spectralink devices and provide PBX services.

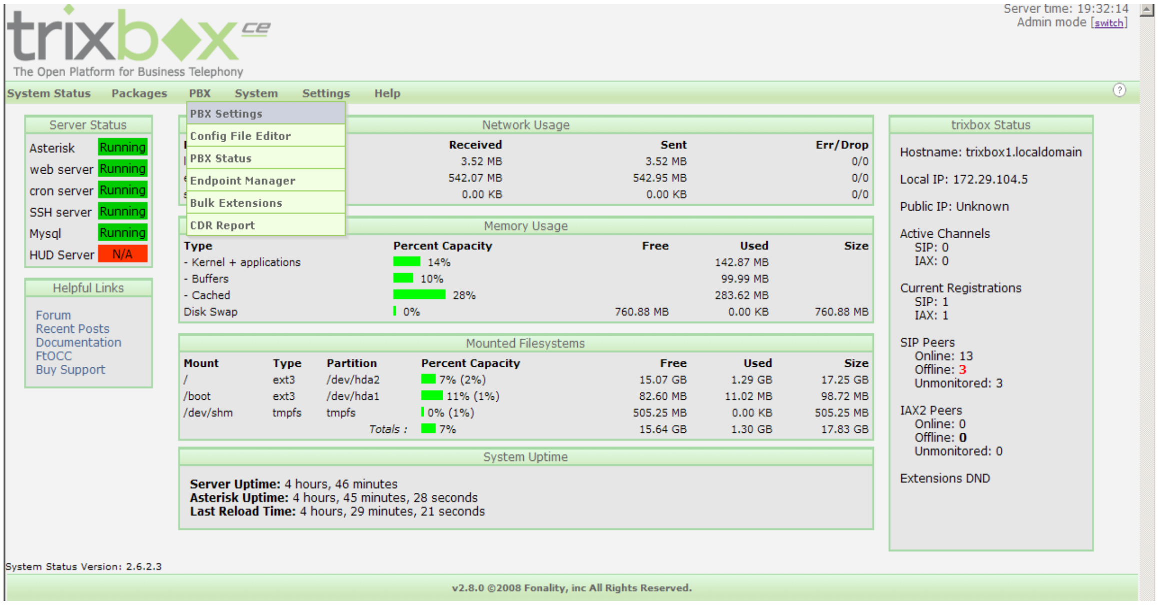
To set up the trixbox:

1. Obtain and install trixbox per provided instructions.
2. On the console, login into your new trixbox with the username: root and the password you selected during installation
3. When you log in the system will tell you what IP address it received from your DHCP server. You can give the system a permanent address now by typing **system-config-network** or setting the IP address from the GUI.
4. If you reconfigured the IP address, restart the network device by running **service network restart**.
5. To continue configuration connect to your system with a web browser using the assigned IP you specified in the previous step.
6. Click on **Switch next to User Mode** this will open the Admin GUI. Login with
   - user: **maint**
   - pass: **password**
7. Click on **Asterisk > PBXconfig**. This will open the PBX configuration GUI

To create device extensions:

When you get to the point in the trixbox setup that you have opened the PBX configuration GUI, please follow these instructions to create the correct extensions in the PBX.

1. Highlight and select **PBX > PBX Settings.**



Server time: 19:32:14
Admin mode [switch]

Server Status

Asterisk	Running
web server	Running
cron server	Running
SSH server	Running
Mysql	Running
HUD Server	N/A

Network Usage

	Received	Sent	Err/Drop
	3.52 MB	3.52 MB	0/0
	542.07 MB	542.95 MB	0/0
	0.00 KB	0.00 KB	0/0

Memory Usage

Type	Percent Capacity	Free	Used	Size
- Kernel + applications	14%		142.87 MB	
- Buffers	10%		99.99 MB	
- Cached	28%		283.62 MB	
Disk Swap	0%	760.88 MB	0.00 KB	760.88 MB

Mounted Filesystems

Mount	Type	Partition	Percent Capacity	Free	Used	Size
/	ext3	/dev/hda2	7% (2%)	15.07 GB	1.29 GB	17.25 GB
/boot	ext3	/dev/hda1	11% (1%)	82.60 MB	11.02 MB	98.72 MB
/dev/shm	tmpfs	tmpfs	0% (1%)	505.25 MB	0.00 KB	505.25 MB
		Totals :	7%	15.64 GB	1.30 GB	17.83 GB

System Uptime

Server Uptime: 4 hours, 46 minutes
Asterisk Uptime: 4 hours, 45 minutes, 28 seconds
Last Reload Time: 4 hours, 29 minutes, 21 seconds

trixbox Status

Hostname: trixbox1.localdomain

Local IP: 172.29.104.5

Public IP: Unknown

Active Channels
SIP: 0
IAX: 0

Current Registrations
SIP: 1
IAX: 1

SIP Peers
Online: 13
Offline: 3
Unmonitored: 3

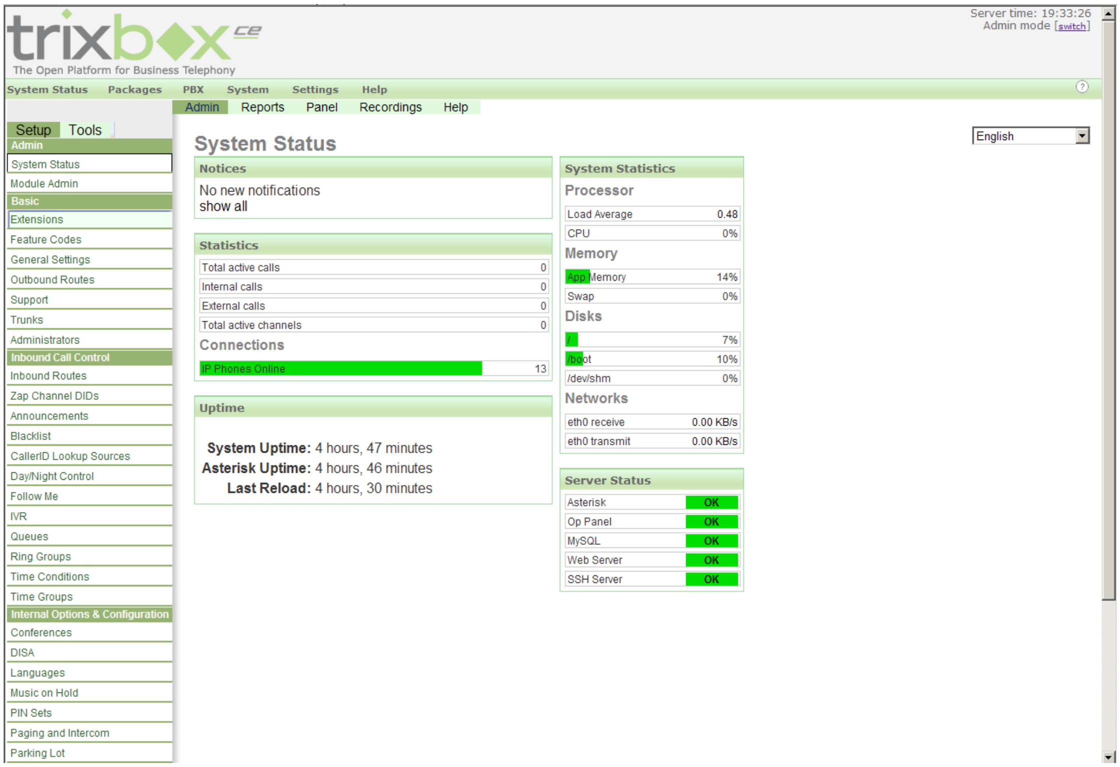
IAX2 Peers
Online: 0
Offline: **0**
Unmonitored: 0

Extensions DND

System Status Version: 2.6.2.3

v2.8.0 ©2008 Fonality, inc All Rights Reserved.

2 Highlight and select **Extensions** under **Basic** on the left of the screen.



Server time: 19:33:26
Admin mode [switch]

System Status

Notices

No new notifications
show all

Statistics

Total active calls	0
Internal calls	0
External calls	0
Total active channels	0

Connections

IP Phones Online	13

Uptime

System Uptime: 4 hours, 47 minutes
Asterisk Uptime: 4 hours, 46 minutes
Last Reload: 4 hours, 30 minutes

System Statistics

Processor

Load Average	0.48
CPU	0%

Memory

App Memory	14%
Swap	0%

Disks

/	7%
/boot	10%
/dev/shm	0%

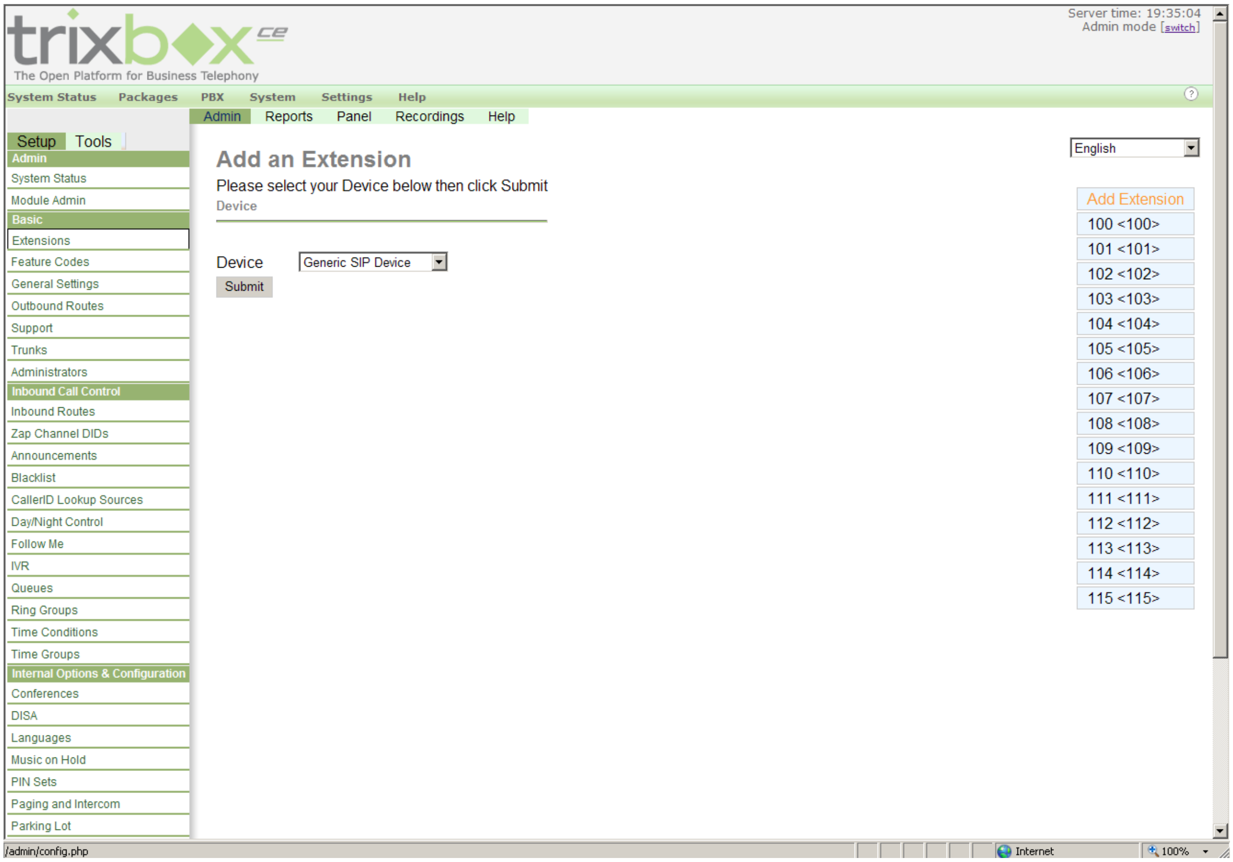
Networks

eth0 receive	0.00 KB/s
eth0 transmit	0.00 KB/s

Server Status

Asterisk	OK
Op Panel	OK
MySQL	OK
Web Server	OK
SSH Server	OK

3 Select **Generic SIP Device** from the **Device** dropdown menu and click the **Submit** button

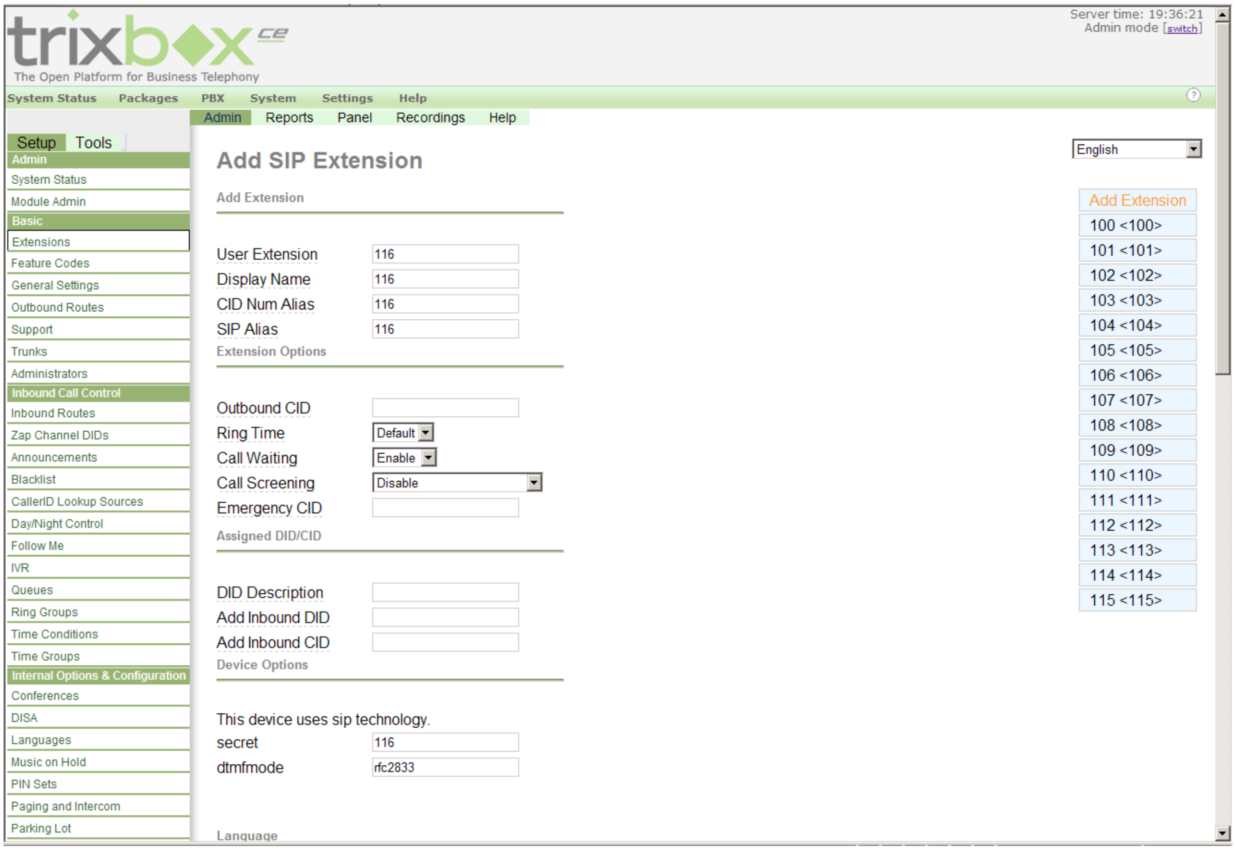


Add an Extension

Please select your Device below then click Submit

Device: Generic SIP Device

4 You may want to create extensions 100 and 101 for testing. Enter the extension number in each of the following fields; **User Extension**, **Display Name**, **CID Num Alias**, **SIP Alias,** and **secret.**



Add SIP Extension

Add Extension

User Extension	116
Display Name	116
CID Num Alias	116
SIP Alias	116

Extension Options

Outbound CID	
Ring Time	Default
Call Waiting	Enable
Call Screening	Disable
Emergency CID	

Assigned DID/CID

DID Description	
Add Inbound DID	
Add Inbound CID	

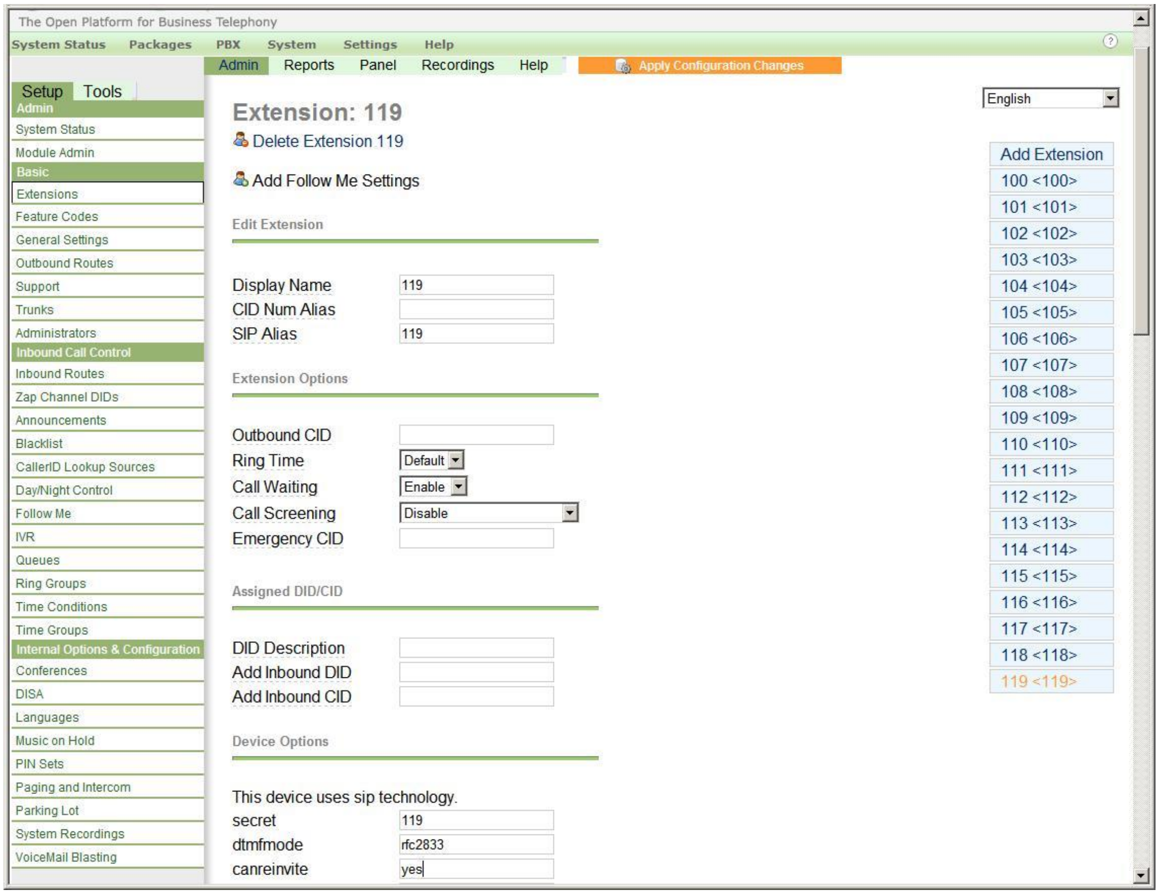
Device Options

This device uses sip technology.

secret	116
dtmfmode	rfc2833

Language

5 Return to each extension you have created by clicking on its number in the Add Extension bar on the right of the screen. Set **canreinvite** to **yes** to allow the phones to continue a phone call without issuing SIP reinvites.



Extension: 119

Edit Extension

Display Name	119
CID Num Alias	
SIP Alias	119

Extension Options

Outbound CID	
Ring Time	Default
Call Waiting	Enable
Call Screening	Disable
Emergency CID	

Assigned DID/CID

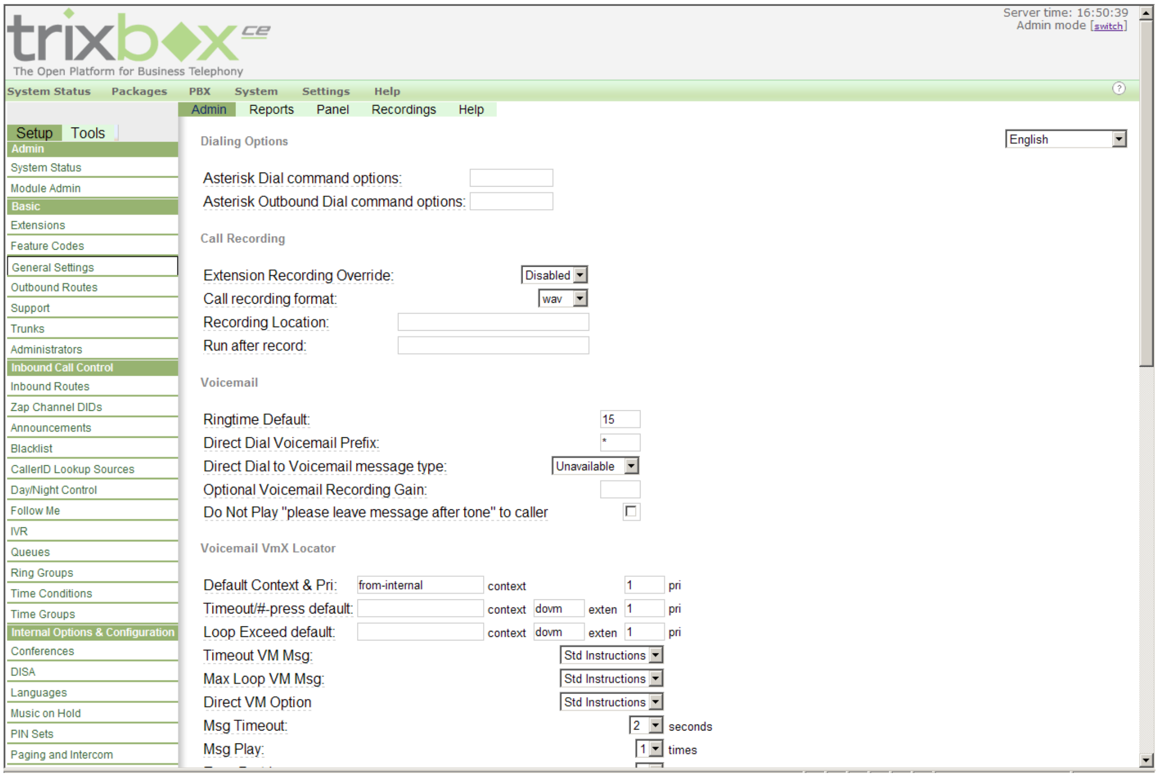
DID Description	
Add Inbound DID	
Add Inbound CID	

Device Options

This device uses sip technology.

secret	119
dtmfmode	rfc2833
canreinvite	yes

6 Select **General Settings** under **Basic** and remover “tr” from **Asterisk Dial command options** field to allow peer-to-peer audio.



Dialing Options

Asterisk Dial command options:

Asterisk Outbound Dial command options:

Call Recording

Extension Recording Override: Disabled

Call recording format: wav

Recording Location:

Run after record:

Voicemail

Ringtime Default: 15

Direct Dial Voicemail Prefix: *

Direct Dial to Voicemail message type: Unavailable

Optional Voicemail Recording Gain:

Do Not Play "please leave message after tone" to caller □

Voicemail VmX Locator

Default Context & Pri: from-internal context 1 pri

Timeout/#-press default: context dovm exten 1 pri

Loop Exceed default: context dovm exten 1 pri

Timeout VM Msg: Std Instructions

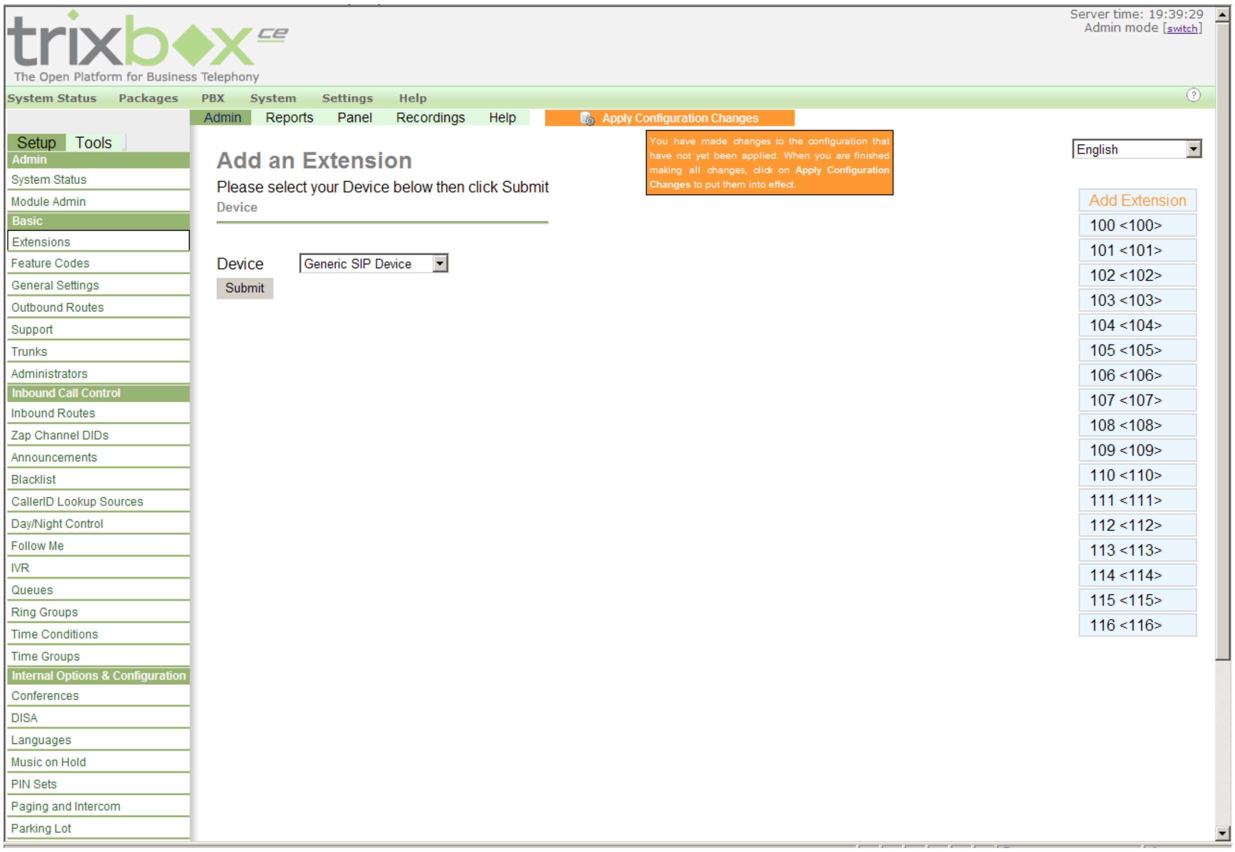
Max Loop VM Msg: Std Instructions

Direct VM Option Std Instructions

Msg Timeout: 2 seconds

Msg Play: 1 times

7 When all 18 extensions have been entered click the orange **Apply Configuration Changes** button to save the configuration.



Server time: 19:39:29
Admin mode [switch]

You have made changes to the configuration that have not yet been applied. When you are finished making all changes, click on **Apply Configuration Changes** to put them into effect.

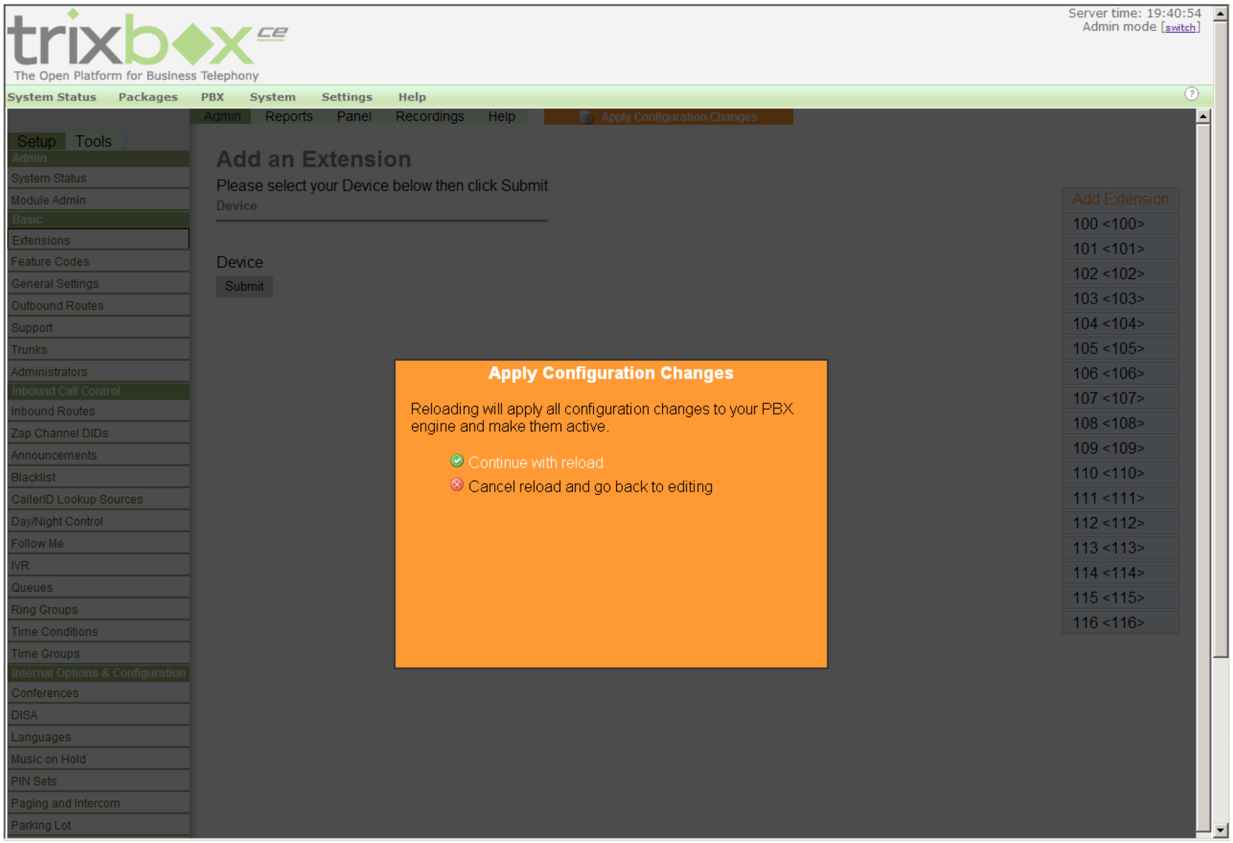
Add an Extension

Please select your Device below then click Submit

Device

Device: Generic SIP Device

8 Then select the **Continue with reload** button.



Add an Extension

Please select your Device below then click Submit

Apply Configuration Changes

Reloading will apply all configuration changes to your PBX engine and make them active.

Continue with reload

Cancel reload and go back to editing

Appendix C: Products Mentioned in this Document

Android is a registered trademark owned by Google, Inc.

Apache and Tomcat are trademarks of the Apache Software Foundation.

Balsamiq is a registered trademark of Balsamiq Studios, LLC.

Chrome browser is a trademark owned by Google, Inc.

Django is a registered trademark of the Django Software Foundation.

Eclipse is a trademark of Eclipse Foundation, Inc.

Firefox is a registered trademark of the Mozilla Foundation.

Fonality and trixbox are registered trademarks of NetFortris, Inc.

JavaScript is a registered trademark owned by Oracle Corporation.

PowerPoint, Visual Studio and Visio are registered trademarks of Microsoft Corporation.

Python is a registered trademark of Python Software Foundation.

Safari is a registered trademark owned by Apple Inc.

W3C, World Wide Web Consortium is a registered trademark of the Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, or Keio University on behalf of the World Wide Web Consortium.

Appendix D: Terms

1.0 XML API

Versity Web API

Activities

activity

Alerts

Android Notification

Android Notification message

Android Widget

custom embedded tone

Document Object Model (DOM)

Event notifications

HTTP Digest Challenge

Internal Uniform Resource Identifiers (URIs)

JavaScript

Notification bar

Phone state polling

post dialing (postd)

Push requests

Server root URL

Spectralink Alertview

Spectralink Configuration Management Server

Spectralink App URLs

Status Bar

web application shortcut widget

App URLs

XHTML

*****END OF DOCUMENT*****