AVAYA

MASTERING AVAYA ONE-X® DESKPHONE APPLICATION DEVELOPMENT

For Avaya SIP and H.323 Deskphones

An Introduction to Architectural Concepts, Capabilities and Developer Tools

ADVANCED
IP COMMUNICATIONS
APPLICATION
DEVELOPMENT

TABLE OF CONTENTS

INTRODUCTION

In today's complex and hyper-competitive marketplace, organizations have to employ the latest communications technologies to remain agile and establish advantage. This involves leveraging IP Telephony, Unified Communications, Presence and other communications capabilities. As a result, the onus is increasingly on application developers to integrate communications capabilities with business processes.

Avaya calls this "Intelligent Communications" – the ability to integrate communications capabilities seamlessly into the fabric of a business in ways that profoundly transform how it operates at all levels to improve efficiency, customer service and profitability.

Typically, the focus is on improving the productivity of existing business processes by adding communications capabilities. However, in many situations, an equally valid and effective approach is to enable access to business processes at existing communications endpoints, in particular at the telephones located on the users' desks.

This book is about developing applications for Avaya one-X® IP deskphones, with H.323 or SIP firmware installed. The book includes information on:

- The application capabilities provided by Avaya one-X IP deskphones.
- Ideas for, and examples of, the types of applications you can develop for Avaya one-X IP deskphones.
- What you need to prototype, develop, test and deploy deskphone applications.
- The features and capabilities of Avaya one-X IP deskphones that can be leveraged by applications.
- The technologies used in deskphone application development.
- How to create the different types of deskphone application.
- The tools and other resources available through the Avaya DevConnect program to help you develop deskphone applications.

Note: Throughout the book the terms "deskphone", "IP deskphone" and "IP telephone" refer to Avaya IP deskphones with H.323 or SIP firmware installed. Although the book is primarily about Avaya one-X deskphones (9600 Series) much of the information is relevant to other Avaya IP deskphone models, such as 4600 and 1600 Series deskphones. Not all of the capabilities

described in the book are supported by all deskphones models or firmware releases – see the product documentation for the deskphone models and installed firmware used in your target environment to confirm which capabilities are supported.

WHY YOU SHOULD READ THIS BOOK

If you are a business analyst, system designer or application developer, or if you just have a general interest, this book provides an ideal introduction to the world of IP deskphone application development.

After reading the book you will have a thorough understanding of what IP deskphones applications are and what they can be used for. In particular, you will learn about the additional roles that IP deskphones can perform over and above being an in-call communications device, including:

- Acting as an information hub for out-of-call, unsolicited audio and visual messages pushed to the deskphone by an application.
- Requesting and displaying web content generated by an application.
- Acting as the user interface to browser-based web applications.

In addition, developers can create applications that dynamically update the content and look-and-feel of the deskphone user interface, allowing it to be customized for specific users and events.

> One very important point to consider is that the capabilities of Avaya IP deskphones can readily be exploited by application developers at little or no additional cost, other than the cost of developing the application software itself. The deskphone implements functionality that can be accessed through its APIs, and provides a built-in web browser and user interface to facilitate user interaction. It is possible to develop applications that only need the deskphones themselves and a web server to host the application and content requested by, or pushed to, the deskphones. Furthermore, most of the resources developers need to create and test IP deskphone applications are available at no cost through the Avaya DevConnect program.

In this book you will find out how cost efficient, quick and straightforward it can be to add value to your organization by creating applications that leverage the full range of capabilities provided by Avaya IP telephony systems.

DESKPHONE AND FIRMWARE RELEASES

The information in this book is correct and up-to-date for the Avaya IP deskphone models and firmware releases available at the time of publication (November 2009). For information about deskphone models and firmware released after this date, please go the Avaya DevConnect web site.

ASSUMED KNOWLEDGE

To get the most out of this book it would be useful to have a good understanding of IP telephony and computer telephony integration. You will also find it advantageous, but not essential, to have a working knowledge of XML, WML, web services and Java.

ABOUT THE AVAYA DEVCONNECT PROGRAM

DevConnect is Avaya's developer and partner program, open to IP communications application developers, System Integrators, ISVs, IHVs and customers alike. Basic membership is free, offering no-charge access to technical education, product API documentation, Software Development Kits (SDKs), sample applications, technical support and much more.

A variety of tools and other resources are available to DevConnect members for IP deskphone application development, including the Avaya PushSDK, Avaya one-X® Deskphone XML Designer, XML validators and the Avaya one-X® Deskphone Emulator. See *Chapter 6* for full details of the available tools.

Companies that wish to extend their relationship with Avaya can apply to become an enhanced-level member, which offers greater technical support, discounted procurement of lab systems and co-marketing benefits based on proven compliance with Avaya Solutions.

For more information, see *Appendix C* at the end of this book or visit **www.avaya.com/devconnect** and sign-up now for free registered membership.

HOW THE BOOK IS ORGANIZED

The book comprises a number of chapters, each covering an aspect of IP deskphone application development:

CHAPTER 1: AVAYA IP DESKPHONE APPLICATION DEVELOPMENT OVERVIEW

Discusses the significance and potential uses of IP deskphone applications in various environments, provides an overview of the various types of applications that can be created and looks at the features of the available Avaya IP deskphone models and firmware releases.

CHAPTER 2: PUSHING MESSAGES AND OTHER CONTENT TO AVAYA IP DESKPHONES

Describes the development of applications that push various types of content, to IP deskphones, including audio messages and web pages. The chapter includes a description of the Push architecture as well as information about each of the available Push types and how they are implemented by applications.

CHAPTER 3: USING SUBSCRIPTION SERVICES AND GLOBAL VARIABLES

Describes the use of subscription services and global variables to allow applications to access information about the deskphones in the target environment.

CHAPTER 4: CUSTOMIZING THE AVAYA IP DESKPHONE USER INTERFACE

Describes how the display content and look-and-feel of the deskphone user interface can be customized by an application. Additionally, the chapter looks at how customization can enhance the user experience; the ways in which user interfaces can be customized; and how customization can be applied.

CHAPTER 5: DEVELOPING WEB APPLICATIONS FOR AVAYA IP DESKPHONES

Describes the technologies supported by the deskphone web browser, the capabilities of the browser, how web content and applications are created, and the ways in which users can access web pages.

CHAPTER 6: AVAYA IP DESKPHONE APPLICATION DEVELOPMENT TOOLS

Describes the Avaya PushSDK, Avaya one-X Deskphone Designer and other development tools provided by Avaya to help developers create IP deskphone applications.

CHAPTER 7: APPLICATION SCENARIOS

Provides examples of how the various capabilities described in the book can come together to provide practical solutions in real-life applications.

APPENDICES

Includes a glossary of terms used in the book, a table showing when deskphones are in a state to receive push messages, and a description of the Avaya DevConnect program.

REFERENCES

Lists documents and other resources referenced in this book.

ICONS USED IN THE BOOK

The following icons are displayed in the margin to highlight notable information:

Important information about this book or IP deskphone application development.

Helpful hints and tips for IP deskphone application developers.

CHAPTER 1
AVAYA IP DESKPHONE APPLICATION DEVELOPMENT OVERVIEW

IN THIS CHAPTER:

- Enterprise IP deskphone applications
- What types of applications can be created for IP deskphones?
- Avaya IP deskphone models
- Application supported by Avaya IP deskphones
- Deskphone application development features

In this chapter, we are going to investigate the importance of IP deskphone applications in various environments, including the workplace. We'll then go on to look at the different types of application that can be developed, and consider the various roles that the deskphone can play within those applications. Following a brief overview of the available Avaya IP deskphone models and firmware releases, we'll take a more detailed look at the features of Avaya IP deskphones that are relevant to application development.

ENTERPRISE IP DESKPHONE APPLICATIONS

Most deskphone users in the workplace also have a networked PC or similar devices on their desktop, specifically designed to allow them to access the applications they need to do their job. These devices generally have larger, richer user interfaces than deskphones, more memory, better text input support, and so on. This begs the question, why develop applications for the deskphone? We'll answer that question in this section and explain how deskphone applications can perform a key role in many situations and environments.

THE ROLE OF IP DESKPHONES IN APPLICATIONS

An IP deskphone can perform any or all of the following roles within an application:

- It can act as the user-interface to a thin-client application hosted on a web server.
- It can act as the endpoint for the delivery of unsolicited audio and visual information sent by an application.
- It can be the target of an application that controls the deskphone itself, for example, to dynamically customize the user interface, settings and logs.

Another possible role is to host applications on the deskphone itself. Currently, hosted applications are only available on the recently released Avaya one-X® 9670G deskphone. At the time of publication, the 9670G natively hosts Avaya-developed applications only; therefore this book does not explore the option further.

CHARACTERISTICS OF IP DESKPHONES

What are the characteristics of IP deskphones that can be exploited by communications application developers?

- Ability to receive and play high-quality, out-of-call audio messages, through the speakerphone, handset or a headset.
- Ability to capture and transmit high-quality, out-of-call audio through the built-in microphone, handset or a headset.
- Most Avaya one-X IP deskphones include a WML browser allowing them to display web content.
- Deskphones tend to be a in a known, fixed location allowing messages and applications to be targeted geographically. For example, an application can send an emergency evacuation message to all deskphones in a particular building, or on a particular floor.
- Although the location is usually fixed, the user may change regularly. For example, in a corporate drop-in cubicle different employees may share a physical location. In a hotel room, the user changes each time a new guest checks in. Applications can be used to remotely set custom options, available functionality and deskphone user interface content for each new user, without needing to reregister or reset the deskphone.
- In office environments most employees have a telephone on their desk. Using the deskphone as an application endpoint device entails no additional end-user hardware, installation or maintenance costs.
- The ubiquitous nature of deskphones makes it easy to provide personnel with access to key administrative and reporting applications from just about any location. This can be used to report closure of work items or to report observed issues that need attention, without the overhead of needing to provide secure, PC-based access to mobile capabilities.
- Unlike many devices, deskphones are nearly always on and available, and are therefore ideally suited for the reliable delivery of critical information.
- In many instances, personal computer display screens are cluttered with multiple application windows, making it possible to miss important information updates. In contrast, deskphone display screens are more readily available to receive and display broadcast messages, ensuring they won't be missed or overlooked by users.

- Support for the Wireless Telephony Applications Interface (WTAI) and the deskphone's built-in telephony features make it easy for applications to integrate communications capabilities with business processes.
- In some environments the deskphone may be the only available device for information and application delivery, for example, in hotel guest rooms or private hospital rooms.

These characteristics make the use of IP deskphone applications the ideal choice in many situations and environments.

WHAT TYPES OF APPLICATIONS CAN BE CREATED FOR IP DESKPHONES?

There are two main types of applications that can be developed for Avaya IP deskphones:

- Browser-based applications – sometimes known as Pull applications
- Push applications

Nearly all Avaya IP deskphone models expose and implement both a Web API and a Push API in support of these application types.

In addition, developers can create applications that offer customized user interface content, including bespoke menus, and a branded "look-and-feel", to provide a personalized user experience.

BROWSER-BASED APPLICATIONS

Most Avaya one-X deskphones include a web browser capable of displaying pages written in the Wireless Mark-up Language (WML). Deskphones can request, or "pull", WML content from an application web server, the company intranet or the World Wide Web. WML support for form controls, allied with text input capabilities, allows the deskphone display screen to act as the user interface for interaction with web-based applications. The creation of WML content and browser-based applications is described in *Chapter 5: Developing Web Applications for Avaya IP Deskphones.*

PUSH APPLICATIONS

The Push API is an XML-based programming interface that allows applications to push unsolicited messages and other content to the deskphones. The following types of Push content are supported by the API:

- **Display:** sends full-screen, WML pages to the deskphone browser.
- **Top Line:** sends a single line text message to the top line of the deskphone display screen.

- **Unicast Audio Receive:** a separate instance of an audio message is played at each deskphone to which it is sent.
- **Multicast Audio Receive:** a single instance of an audio message is played simultaneously at multiple deskphones.
- **Audio Transmit:** a message is sent to a deskphone prompting the user to use the deskphone to transmit an audio message. The transmitted message could then be recorded or used in conjunction with an Audio Receive Push to stream the message live to other deskphones.
- **Phonexml:** allows an application to dynamically:
  - ◊ Update a deskphone's user interface content, including customized menus
  - ◊ Disable hard button functionality
  - ◊ Clear a deskphone's call, web and contacts history
  - ◊ Reset user-defined preferences
  - ◊ Update deskphone options and settings, such as preferred language, ringtone patterns and the displayed time format.

Developing applications that take advantage of the various Push types is described in *Chapter 2: Pushing Content to Avaya IP Deskphones.*

In addition, a **Subscribe** Push can be used to instruct a deskphone to send its details to a subscription service. Subscribe Push allows an application to maintain an up-to-date database of deskphones on the network that it can subsequently use to access information about those deskphones. Use of Subscribe Push is described in *Chapter 3: Using Subscription Services and Global Variables.*

The deskphones' Push capabilities can be used to, for example:

- Broadcast company news
- Send meeting reminders with conference bridge numbers
- Stream audio alerts and music, such as wake-up alarms in hotel rooms
- Stream audio announcements
- Send stock news and other critical business information
- Broadcast emergency messages, such as severe weather warnings and evacuation alerts
- Build intelligent databases that can subsequently be used to target information to individual deskphones or groups of deskphones.
- Dynamically update the deskphone user interface and settings.

USER INTERFACE CUSTOMIZATION

Two aspects of a deskphone's native, non-browser user interface can be customized:

- **Content:** the menu options and functionality available via the user interface can be customized. This includes title line and prompt line text, application line functionality and labels, softkey assignments and hard button disablement. Content customization is defined in a Deskphone XML file.
- **Skin:** the look-and-feel of the user interface can be customized. This includes logos, text colors and menu option images for both selected and unselected options. Look-and-feel customization is defined in a Skin XML file.

Deskphone user interface customization is described in detail in *Chapter 4: Customizing the Avaya IP Deskphone User Interface.*

AVAYA IP DESKPHONE MODELS

Avaya offers a range of IP deskphone models to meet the full compass of business requirements.

- 9600 Series Avaya one-X Deskphones, support browser-based and Push application development
- 5600 Series IP Deskphones: for small and medium sized companies, support browser-based and Push applications on selected models
- 4600 Series IP Deskphones, also support browser-based and Push application development
- 1600 Series IP Deskphones: satisfy basic communications needs, with support for Push applications on some models

The information in this section is correct at the time of publication (November 2009). For up-to-date information about available Avaya IP deskphones visit the Avaya web site (http://www.avaya.com).

9600 SERIES AVAYA ONE-X DESKPHONES

The one-X 9600 Series represents Avaya's flagship deskphones for enterprise users, with models designed to meet a variety of needs and use cases:

- 9610 is for walk-up use in common areas.
- 9620, 9620Color and 9620Lite offer features such as status lights and buttons, and improved audio quality.
- 9630, 9630G, 9640 and 9640G provide advanced IP telephony features.
- 9650 and 9650Color are for receptionists and contact center agents.
- 9670G has a large touch screen and an onscreen keyboard for text input.

Note: Phone models with a "G" suffix, such as the 9670G, support GigE network connections.

APPLICATIONS SUPPORTED BY AVAYA IP DESKPHONES

The ability of a deskphone to support browser-based and Push applications depends on a number of factors:

- The features and capabilities of the deskphone model. For example 1600 Series deskphones do not support the Display Push type.
- The type of firmware installed, either H.323 or SIP. For example, the Phonexml Push type is supported by SIP firmware only.
- The firmware release. For example, the Multicast Audio Receive Push type is supported on H.323 firmware release 3.0 and higher only.
- System configuration. Various parameters defined in the system-wide Settings file determine browser behavior and permitted Push types.

In addition, how web and user interface content is rendered at a deskphone depends on factors such as the display screen size and color support.

This book describes the super-set of all available types of application functionality. However, because new deskphone models and firmware are released regularly, the book does not list the specific deskphone models and firmware releases that support each type of application functionality – for definitive information refer to the product documentation for the deskphones used in your target environment.

Unless stated otherwise, the images and examples in this book are based on a mid-range Avaya one-X 9640 Deskphone with SIP firmware release 2.5 installed. This model has a medium-sized graphical color display, supports browser-based applications, supports all Push types except Multicast Audio Receive, and supports user interface customization.

DESKPHONE APPLICATION DEVELOPMENT FEATURES

Figure 1-1 shows an Avaya one-X® 9640 deskphone. The features of the deskphone that are relevant to application development are labeled.

Figure 1-1: Avaya one-X 9640 Deskphone

- **Display Screen:** the graphical screen displays:
  - ◊ Native user interface pages, such as the Phone (see **Figure 1-2**), Contacts or Call Log page,
  - ◊ Customized user interface pages (see **Figure 1-3**), or
  - ◊ WML browser pages (see **Figure 1-4**).
- **Line Buttons:** used to select the adjacent menu option on the screen display.
- **Soft Keys:** Soft key functionality and labels can be defined on each deskphone user interface and WML page. The labels are displayed on the bottom line of the display screen immediately above the corresponding softkeys.
- **Navigation Buttons:** the **Up** and **Down** navigation buttons are used to give focus to, and highlight, menu options on the display screen. The **Left** and **Right** navigation keys are used to display the previous and next pages in a sequence.

- **OK Button:** used to select the currently highlighted menu option.
- **Hard Buttons:** the Phonexml Push type allows you to disable the labeled hard buttons, preventing users from accessing **Call Log**, **Contacts**, **Forward**, **Avaya Menu** (labeled **A Menu** on this model, but may be labeled differently or omitted on other models), **Message** and/or **Headset** functionality.

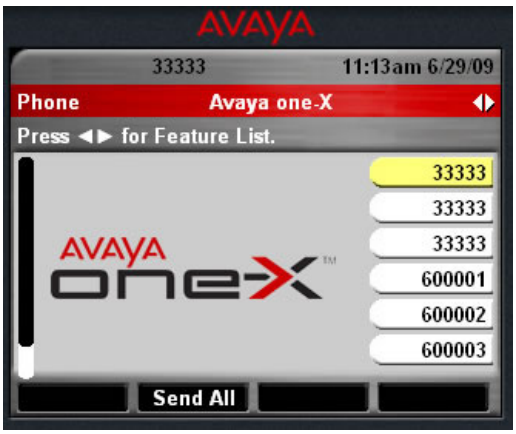

Figure 1-2: Native Phone page

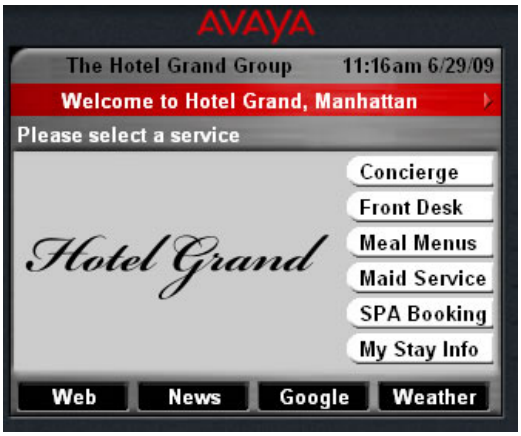

Figure 1-3: Customized page content

Figure 1-4: WML browser page

ROUND-UP

In this chapter you were introduced to the various types of applications supported on Avaya IP deskphones and how they can perform a key role in enterprise and other environments, providing practical solutions and value. Now that you have a basic understanding of what's possible, we will go on to look at the development of each type of IP deskphone application in more detail, starting with Push applications.

CHAPTER 2
PUSHING MESSAGES AND OTHER CONTENT TO AVAYA IP DESKPHONES

IN THIS CHAPTER:

- What is Push and what is it used for?
- How Push works
- Configuring systems to support Push
- The available Push types

In this chapter we will look at the Push API exposed by most Avaya IP deskphones and discover how it can be used to send unsolicited messages (audio and text) and customized user interface settings to Avaya IP deskphones. We'll start by considering what Push is and its possible applications, before going on to find out about the Push architecture, configuring a system to support Push and details of the different Push types.

WHAT IS PUSH AND WHAT IS IT USED FOR?

Push is the ability of an application to send unsolicited, out-of-call messages and other content to IP deskphones. This ability is supported by an XML-based programming interface, known as the Push API, exposed by, and implemented on, the deskphones themselves. Depending on the Push type, the pushed content is rendered at the deskphone as an audio or visual message, or as customized user interface content.

The types of content that can successfully be pushed to a deskphone depend partly on the deskphone model, installed firmware type (SIP or H.323) and firmware release. As you learned in *Chapter 1,* the following types of Push content may be sent by an application:

- **Display:** full-screen, WML pages.
- **Top Line:** a single line text message.
- **Unicast Audio Receive:** an audio message to individual deskphones.
- **Multicast Audio Receive:** a broadcast audio message to multiple deskphones.
- **Audio Transmit:** a prompt for the user to record or transmit an audio message from a deskphone.
- **Phonexml:** to dynamically update deskphone settings, logs and user interface content.
- **Subscribe:** to prompt the deskphone to send its details to a subscription service.

Each of the Push types is described in detail later in this chapter, in the section on *The available Push types.*

USES OF PUSH

Push can take advantage of many of the characteristics of IP deskphones to communicate important information to individuals, whole organizations or groups of users based on criteria such as their physical location, department, role or level. Some examples of uses for Push include:

- Broadcasting company news.
- Sending meeting reminders to attendees.
- Streaming audio alerts and announcements.
- Broadcasting emergency messages, such as severe weather warnings and evacuation alerts.
- Building intelligent databases to target information to individuals or groups of deskphone users.
- Remotely updating or resetting the deskphone user interface, settings, call history, etc. when the primary user of the deskphone changes.

There are many more possible applications for Push; too many to list here. *Chapter 7* describes some practical use cases to help illustrate the value of Push in your target environments.

HOW PUSH WORKS

In this section we'll look at the Push process, the message flow between an application and a deskphone, and the anatomy of a Push Initiation message.

THE PUSH PROCESS

Applications that exploit the Push capabilities of Avaya IP deskphones must be able to perform some or all of the following functions:

Processing Push application triggers:
Push applications must listen for or receive the events that trigger the Push process. For example, a Push could be triggered automatically by a business process, or manually by operator input.

Obtaining deskphone IP addresses:
To be able to perform a Push, an application needs to know the IP addresses of the deskphones that it is sending the Push Content to. For example, when an operator submits a request, via the application user interface, to send a Top Line Push message to selected extensions, the application performs a database query to obtain the IP addresses of the deskphones associated

with those extensions. In addition, the application can look-up the model type of each deskphone to ensure it uses a Push type supported by the model.

Chapter 3: Using Subscription Services and Global Variables, describes how subscription services can be used by client applications to maintain and access information about the deskphones in the network, including their IP addresses and models.

Creating and provisioning Push Content:
The message content to be pushed to the deskphones must be hosted on a web server. Depending on the Push type, the Push Content is in the form of an XML or WML file. Deskphones can only download Push Content from web servers that are defined as Trusted Push Servers in the system-wide Settings file: see *Configuring systems to support Push,* below.

In some scenarios, standard message content may already exist on a Trusted Push Server and be used by the Push application. For example, an emergency notification application can use a standardized WML file and pre-recorded announcement whenever there is a fire drill to push an audio message to all deskphones. In this case, the Push application needs to know where the Push Content is stored, but does not need to create these files or announcements in real time.

In other scenarios, the application may need to generate and provision the Push Content itself, on demand. For example, when an operator enters a message at the application user interface and submits a request for it to be sent to selected deskphones, the application generates a properly formatted XML file containing the message text and copies it to a Trusted Push Server, before initiating a Top Line Push.

Initiating the Push
All Push applications must include a web server so that it can send Push Initiation requests in HTTP POSTs addressed to the deskphones. In its role as Push Initiator, the application sends XML messages to the deskphones to initiate the Push process. The Push Initiation request contains information such as the Push type and the location of the Push Content.

The final stage of the Push process is performed by the deskphones, rather than the application. The deskphones use the information in the Push Initiation request to retrieve the Push Content from the Trusted Push Server. We'll look at how Push Initiation messages are formed and processed in much more detail later in this section.

Figure 2-1 is a schematic diagram of the Push process, illustrating the functionality of Push applications.

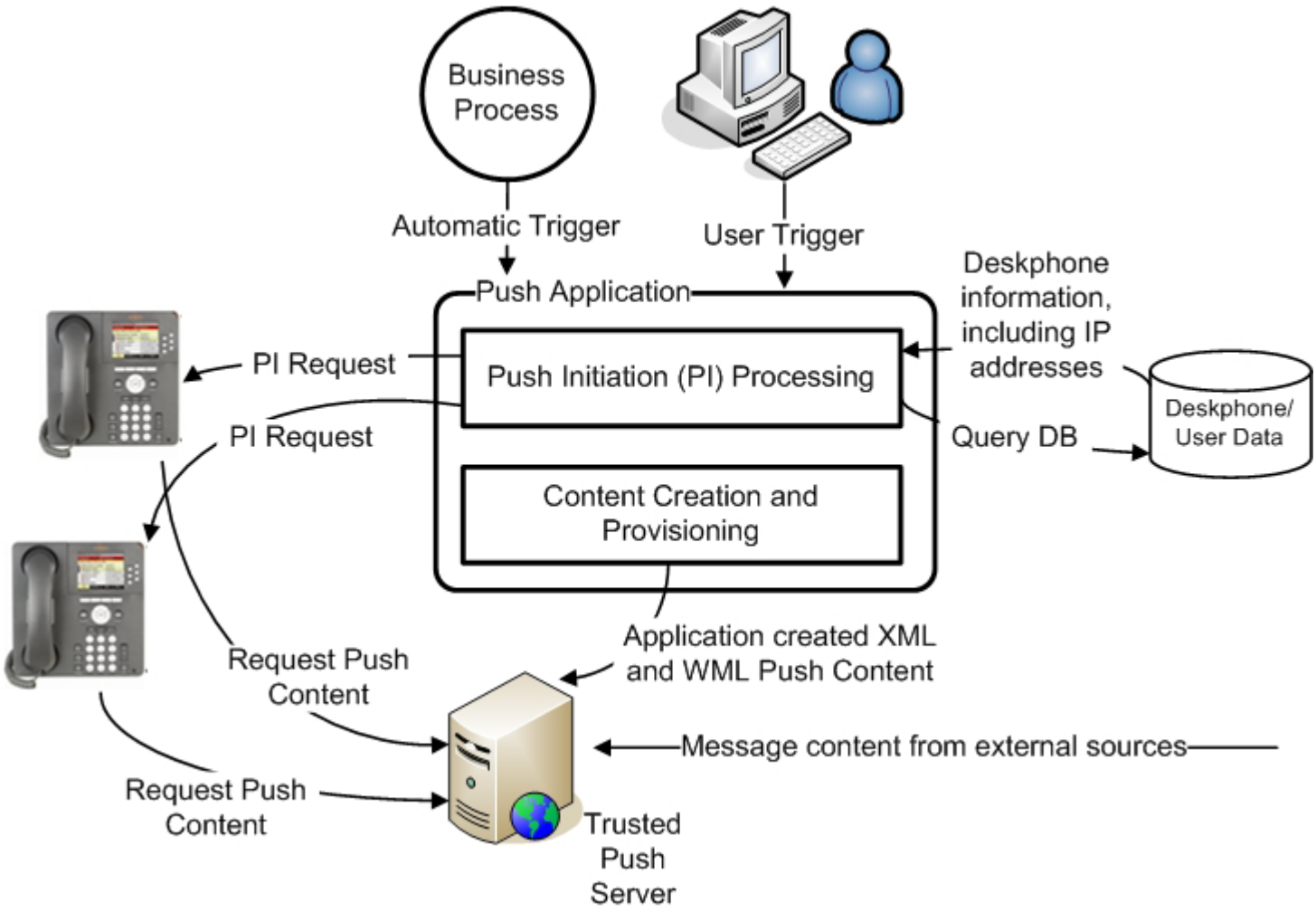

Figure 2-1: Schematic diagram of the Push process

In practice, the Push Content may be hosted on the same physical web server as the Push Initiator.

PUSH MESSAGE FLOW

Figure 2-2 shows the message flow between a Push Initiator, an Avaya IP deskphone and a Trusted Push Server in a basic Push transaction.

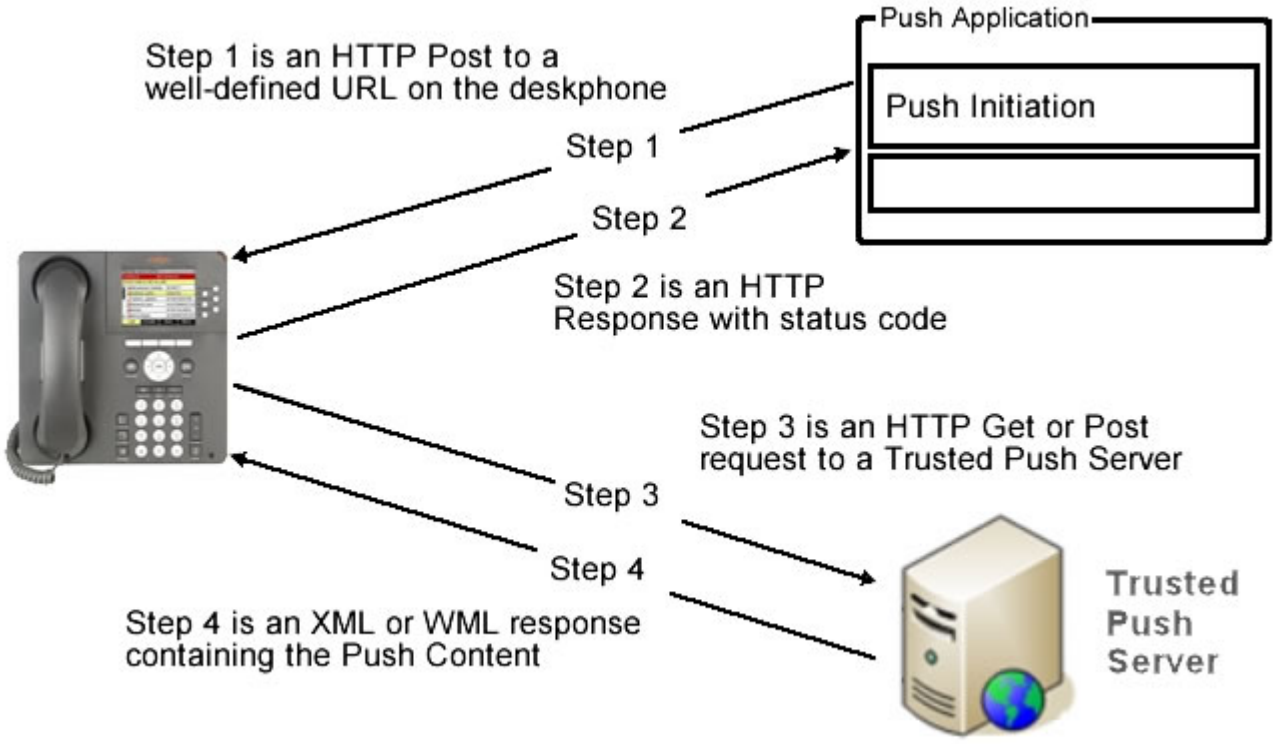

Figure 2-2: The Push message flow

Step 1:
The Push application sends a Push Initiation request in an HTTP POST addressed to the deskphone. The HTTP POST address of the deskphone has the format:

http://**<ip_address>**/forms/push

where **<ip_address>** is the IP address of the targeted deskphone. The deskphone port number is specified in the system-wide Settings file and does not need to be included in the address: see *Configuring systems to support Push,* below.

The Push Initiation request is an XML message containing information such as the Push type and the location of the Push Content. We'll look at Push Initiation requests in more detail in the next section, *The anatomy of a Push Initiation request.*

Step 2:
The Avaya IP deskphone includes a built-in HTTP server to process the incoming Push Initiation message. The deskphone receives the Push Initiation request and performs various checks to ensure the request is valid, and that the deskphone is able and permitted to render the Push Content. The checks include:

- That the XML message is valid and well-formed.
- That the deskphone is not currently in a state, or performing a task, that prevents it receiving the Push Content. What constitutes a "non-pushable" state depends on the Push type and the request priority. For example, most types of Push cannot be processed if the deskphone is restoring a back-up file or running a local procedure; similarly, a normal priority audio Push will not be processed if the deskphone is involved in a call. *Appendix B: Pushable and Non-pushable States* includes a table showing which states and request priorities prevent Push Initiation requests from being processed.
- The deskphone maintains a list of Trusted Push Servers from which it is allowed to retrieve Push Content. If the location of the Push Content, as specified in the Push Initiation request, is not on a Trusted Push Server the request will be rejected. We'll look at how the deskphone maintains a list of Trusted Push Servers later in this chapter, in the section on *Configuring systems to support Push*.
- That the appropriate system parameter has been set to allow the requested Push type. The customizable system parameters that affect Push are described later in this chapter in the section on *Configuring systems to support Push*.
- Obviously, the deskphone model and installed firmware must also support the requested Push type.

The deskphone responds to the Push Initiation request with an HTTP status code. The response also contains an HTTP header extension, called the "x-Avaya-Push-Status" code, which notifies the Push Initiator that the request has been accepted or the reason for it being rejected.

Step 3:
Provided the request is accepted, the deskphone sends an HTTP POST or GET to the Trusted Push Server requesting the Push Content.

Step 4:
The deskphone receives and parses the Push Content from the Trusted Push Server, then displays or streams it, as appropriate. Depending on the Push type, the Push Content is an XML or WML file.

In *Chapter 6* we will look at the tools Avaya provides to help you create and send Push Initiation messages, and to create and provision Push Content. These tools include a client-side Java interface known as the PushSDK API, the one-X® Deskphone XML Designer and XML validators.

THE ANATOMY OF A PUSH INITIATION REQUEST

In this section we will dissect a typical Push Initiation request message to investigate its structure and elements, to help better understand how Push works. **Figure 2-3** shows a typical Push Initiation message, used to push a WML page to a deskphone's browser.

```
<?xml version="1.0"?>
<Push alert="3" type="display" mode="barge">
      <go href="http://trusted_push_server/filename.wml" method="post">
            <postfield name="name1" value="value1"/>
            <postfield name="name2" value="value2"/>
      </go>
</Push>
```

Figure 2-3: A sample Display Push Initiation message

As you can see, the body of the request is contained within a **<Push>** tag, which has the following attributes:

- **type:** defines the type of Push Content to be sent to the deskphone. Possible values are "top line", "display", "audio", "multicast", "transmit", "phonexml" or "subscribe".
- **alert:** defines whether ring-pings are sounded at the deskphone to notify the user of the incoming message and, if so, how many. Possible values are "0", "1", "2" or "3".
- **mode:** defines the priority to be given to the Push, either "barge" or "normal". The mode determines whether the Push interrupts other activities being performed on the deskphone when the request is received. For example, an Audio Receive Push request sent in barge mode will interrupt an active call at a deskphone, whereas, if the request is sent in normal mode, the Push will be rejected. See *Appendix B: Pushable and Non-pushable States*, for information about how the priority affects whether and how a Push Initiation request is processed.

The **<go>** tag has two attributes that define where the Push Content is located and how it should be retrieved:

- **href:** defines the location of the Push Content, including the URL of the Trusted Push Server and the content's filename.
- **method:** defines whether the deskphone will use an HTTP POST or GET to request the Push Content from the Trusted Push Server. If not specified, the value defaults to POST if **<postfield>** tags are defined inside the **<go>** tag, or GET if not.

If the HTTP POST method is used, the client application can define **<postfield>** tags in the Push Initiation message. Postfield tags are used to specify name-value pairs that will be passed to the

Trusted Push Server in the request for the Push Content. If the HTTP GET method is to be used, these values can be included in the **href** attribute.

Push Initiation requests for other Push types support additional tags: for full details see the *Avaya one-X Deskphone Edition for 9600 Series IP Telephones Application Programmer Interface (API) Guide [2].*

PUSH CONTENT FILES

Depending on the Push type, Push Content may be in the form of a WML file for Display Push or of an XML file for other Push types. You will find summary information about Push Content files for each Push type later in this chapter, in the section on *The available Push types.* However, for detailed information about the format, specification and creation of Push Content files see the *Avaya one-X Deskphone Edition for 9600 Series IP Telephones Application Programmer Interface (API) Guide [2].*

CONFIGURING SYSTEMS TO SUPPORT PUSH

There are a number of customizable system parameters that affect whether and how Push is processed by the deskphones in a network. These parameters are defined in a system-wide Settings file, called `46xxsettings.txt`, hosted on an HTTP server. Whenever a deskphone is registered or reset, it loads and applies the settings defined in the system-wide Settings file. For more detailed information about the Settings file, see the *Administrator Guide [4]* for the deskphones used in your target environment.

The following parameters, defined in the system-wide Settings file, affect how Push is processed by the deskphones in a network:

- **PUSHCAP:** defines the types of content that can be pushed to the deskphones in the network, and the priority levels that are supported for each Push type ("barge-only" or "normal + barge"). If an application attempts to initiate a disallowed Push type or use a disallowed priority value, the Push Initiation request will be rejected.
- **PUSHPORT:** defines the Transmission Control Protocol (TCP) listening port used by all deskphones in the network to listen for incoming Push Initiation requests. The default value is "80".
- **SUBSCRIBELIST:** defines the subscription services to which deskphones in the network can subscribe. See *Chapter 3: Using Subscription Services and Global Variables*, for detailed information about subscription services.
- **TPSLIST:** defines the list of Trusted Push Servers from which deskphones in the network can download Push Content.

It is important that developers are aware of the values defined for these system parameters in their target environment so that deskphone applications behave as intended when they are deployed.

THE AVAILABLE PUSH TYPES

This section describes the various types of content that can be pushed to Avaya IP deskphones, namely:

- Top Line Push
- Display Push
- Receive Audio Push
- Transmit Audio Push
- Phonexml Push
- Subscribe Push

Although each Push type is described in turn, remember that they can be used in combination with each other, and with browser-based applications, to achieve the desired solution.

TOP LINE PUSH

Top Line Push is used to send a single-line text message to a deskphone. The message is displayed in the top line of the deskphone display screen, where the missed call icon, extension number and date/time information are usually shown.

User experience

Provided the deskphone is in a pushable state, up to three ring-pings are sounded to notifying the user of an incoming message. If system messages are being displayed on the top line, the pushed message is held until they complete. The message is then displayed on the top line of the deskphone display screen, as shown in **Figure 2-4.**

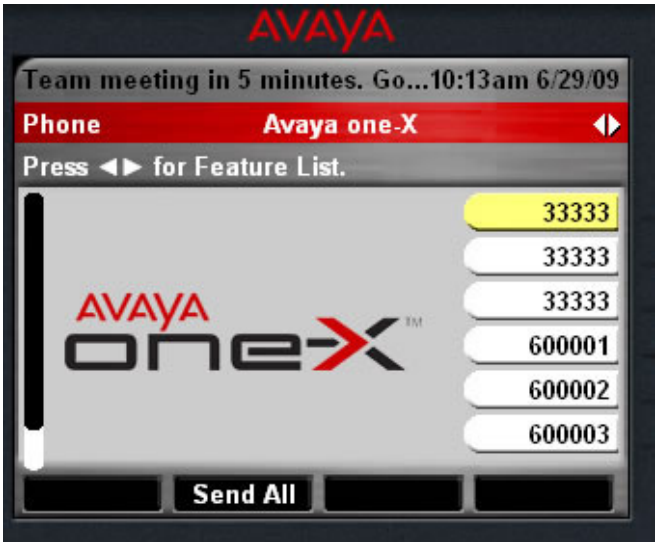

Figure 2-4: Top Line "Team Meeting" message pushed to the deskphone

If the message is longer than will fit on a single line (typically about 30 characters, but dependent on the deskphone model), its content alternates; if the message is more than twice as long as will fit, it is truncated. The message displays for a period determined by the Push API (in practice, for about 30 seconds) and is then cleared.

Push Content

Top Line Push Content is in the form of a simple XML file that is rendered at the deskphone. **Figure 2-5** shows the XML for the Team Meeting message displayed in **Figure 2-4.**

```
<?xml version="1.0"?>
<Response>
  <Topline>
    Team meeting in 5 minutes. Go to the auditorium now.
  </Topline>
</Response>
```

Figure 2-5: Sample Top Line Push Content

Like all XML Push Content, the root of the message is the **<Response>** tag.

DISPLAY PUSH

Display Push is used to send full-screen web content to a deskphone's WML web browser. The Push Initiation request sent to the deskphone includes the address of the WML file to be displayed.

User experience

Provided the deskphone is in a pushable state, up to three ring-pings are sounded. In most cases, the web browser automatically takes focus and the WML file is displayed, as shown in **Figure 2-6**. However, for a normal priority Push where the user is in text entry mode at a non-web screen, such as the Contacts screen, the web page is loaded in the background and displayed when the user selects the **Web** softkey.

Figure 2-6: Sample Display Push Content

The WML page is displayed until the user navigates to another page or exits the browser.

Push Content

Display Push Content is in the form of a WML file. The WML may:

- incorporate static, informational content,
- provide the user-interface to a web application, or
- enable access to native telephony functionality.

In *Chapter 5: Developing Web Applications for Avaya IP Deskphones,* we will take a look at the creation of WML pages, including the use of forms, scripts, telephony interfaces and images.

RECEIVE AUDIO PUSH

Receive Audio Push is used to stream unsolicited, out-of-call audio messages to deskphones. Two types of Receive Audio Push are supported:

- **Unicast** (Push Initiation type = "audio"): A separate RTP stream is played at each deskphone.
- **Multicast** (Push Initiation type = "multicast"): A single RTP stream is played at all of the deskphones to which it is pushed, preventing unnecessary packet duplication and allowing a message to be broadcast to a large number of deskphones simultaneously.

Multicast Receive Audio Push requires an IP network infrastructure capable and configured to support multicast.

Push Initiation requests include the location of the XML file that defines the Push Content. Multicast Push Initiation requests must additionally include the deskphone's RTP port on which the deskphones listen for the multicast message and the multicast IP address to which the deskphones must subscribe.

User experience

Provided the deskphone is in a pushable state, up to three ring-pings are sounded to notify the user that an audio message is about to be played. Unless a barge priority Display Push accompanies the Audio Push, an interrupt screen is displayed at the deskphone. The interrupt screen includes information about the audio stream, including instructions for stopping the message. A custom message may also be included in the Audio Push Content and displayed on the prompt line.

While the message is playing, the user can switch from the speaker to the handset or to a headset, as desired, and adjust the audio volume. The user can stop the message by going on hook, by selecting a call appearance, or by selecting a "click-to-call" link on a web page. If a call is made to the deskphone while the message is playing, the ring-tone is muted and the call either goes to cover or can be answered when the message ends.

Push Content

Receive Audio Push Content is in the form of a special XML file. To help understand how Receive Audio Push works it is worth analyzing a typical Audio Receive Push Content file, as shown in **Figure 2-7.**

```
<?xml version="1.0"?>
<Response>
  <Audio packetsize="40" codec = "PCMU">
    <AudioTimer value="30"/>
    <Url href="RTPRx://ip_address:port"/>
    <Promptline>
      Listen to this important security announcement
    </Promptline>
  </Audio>
</Response>
```

Figure 2-7: Sample Receive Audio Push Content

The **<Response>** tag must include a valid **<Audio>** tag. The **<Audio>** tag can have two attributes:

- **packetsize:** the length of the packets in the RTP stream, in milliseconds. Possible values are "10", "20", "30", "40" (default, if not specified), "50" or "60".
- **codec:** the codec used to encode the RTP stream. Possible values are "PCMU" (default, if not specified) or "PCMA".

The **<Audio>** tag contains the following elements:

- **<AudioTimer>:** an optional element that sets an inter-packet timer, specified in seconds, which is reset when each packet is received. If no packet is received within the period defined in the tag's **value** attribute, the RTP stream is terminated. The **value** must be in the range 5 to 30 seconds; the default is 20 seconds.
- **<Url>:** this mandatory element has one attribute, **href**, that contains the IP address of the endpoint transmitting the RTP.
- **<Promptline>:** optional element whose contents are displayed on the prompt line of the interrupt screen while the message is being played.

Stopping the Audio Stream Programmatically

A deskphone application can terminate the audio stream at any time by making use of a special stop file. The stop is a Receive Audio Push Content file with the **<Url>** tag's **href** attribute set to "RTPRx://STOP". The application sends a Push Initiation request containing the location of the special stop file in the **<go>** tag's **href** attribute. When the deskphone requests and receives the stop file from the Trusted Push Server, the audio is automatically terminated.

TRANSMIT AUDIO PUSH

Transmit Audio Push is used to prompt a user to record or broadcast an audio message from their deskphone. The Transmit Audio Push Initiation request invites the user to transmit the audio message. The client application could, for example, save the message to voicemail or use a subsequent Receive Audio Push to broadcast the message to other deskphones in the network.

Remember that if you use a Receive Audio Push to play a transmitted message at another user's deskphone, the application must go through the Push Initiation steps for the receiving deskphone independently to establish the connection to the Trusted Push Server from which the transmitted audio is being provided.

User experience

Provided the deskphone is in a pushable state, up to three ring-pings are sounded to notify the user that a Transmit Audio Push request has been received. Unless a barge priority Display Push accompanies the request, an interrupt screen is displayed at the deskphone, containing information about how to start and end the audio transmission or cancel the request.

The interrupt screen is displayed until the user ends the transmission, hangs up or cancels the request.

To decline an invitation to transmit a message, the user presses the **Cancel** softkey.

To start transmitting a message, the user presses the **Talk** softkey. The user can choose to talk into the speaker, handset or a headset and adjust the volume to the required level. To end the transmission, the user goes on hook, selects a call appearance, or presses the **Cancel** or **Exit** button on the interrupt screen.

Push Content

Push Content is in the form of an XML file. The file has the same format as a Receive Audio Push Content file, except that:

- The **<Url>** element's **href** attribute has the format RTPTx://ip_address:port, which identifies the server and port that the message is streamed to.
- The **<AudioTimer>** tag is not used.

Starting and Stopping Streaming Programmatically

A deskphone application can start and stop the audio stream from the transmitting deskphone by making use of Push Content files with special values in the **<Url>** tags **href**: RTPTx://START and RTPTx://STOP. The application sends a Transmit Audio Push Initiation request to the deskphone containing the location of the special start or stop file, as required, in the **<go>** tag's **href** attribute.

PHONEXML PUSH

The Phonexml Push type allows deskphone applications to dynamically:

- Customize the deskphone non-browser user interface, including title line, top line, menu options, softkey functionality and labels.
- Customize the deskphone user interface look-and-feel, including logos, background images, text colors and softkey shapes.
- Clear deskphone settings, including the call log, redial values, web history and user-defined contacts lists.
- Reset all preferences, set by the user via the deskphone's **Options and Settings** menus, to their default values.
- Configure select deskphone options and settings, such as the preferred language, ringtone and displayed time format.

Phonexml Push is particularly useful in situations where a deskphone's user regularly changes, for example, in a hotel room where each new guest becomes the deskphone's primary user for the period of their stay. Phonexml Push allows an application to dynamically clear the previous user's settings and to customize the deskphone for the new user, without having to reboot or reset.

User experience

The user will be able to see the results of the Push at the deskphone, but will otherwise be unaware that the Push has taken place: no alerts are sounded with a Phonexml Push.

Push Content

Push Content is in the form of a valid Phonexml file. **Figure 2-8** shows sample XML code, used to display the name of the user on the top line of the display in place of the extension number.

```
<?xml version="1.0"?>
<Response>
 <SetSettingsRequest>
 <data>
  <name>UserDisplayName</name>
  <value>
   <stringValue>Jack Francis</stringValue>
  </value>
 </data>
 </SetSettingsRequest>
</Response>
```

Figure 2-8: Sample Phonexml Push Content

The **<Response>** tag must contain one of the following tags, depending on the type of action to be performed:

- **<SetSettingsRequest>**: to configure or reset one or more user-defined settings on the deskphone.
- **<ClearPhoneHistory>:** to clear the deskphone's call and web histories.
- **<RefreshResourceRequest>**: to dynamically customize the display content, including menu options and language.

Each of these Phonexml Push types is described in more detail below.

Set Settings Requests

Requests to configure deskphone settings are contained within a **<SetSettingsRequest>** tag. Each setting is contained in a **<data>** tag and is defined by the setting name and its new value. **Table 2-1** lists the configurable settings, by **<name>** tag.

<NAME> TAG VALUE	DESCRIPTION OF SETTING
ButtonClickEnabled	Determines whether or not an audible click sounds when buttons are pressed on the deskphone.
CurrentLogo	Sets the logo to be used as the background for the deskphone display. The logo must have been listed in the LOGOS parameter in the system-wide Settings file loaded when the deskphone was last reset.
CurrentSkin	Sets the skin to be used to determine the look-and-feel of the deskphone's non-browser display. The skin must be listed under the SKINS parameter in the system-wide Settings file loaded when the deskphone was last reset.
DefaultAudioPath	Determines whether the speakerphone or headset goes off hook to handle the audio when the user makes an on-hook call. If the deskphone has been configured for auto-answer, incoming calls are answered on the same device.
DisplayBrightness	Sets the display brightness.
DisplayCallTimers	Determines whether or not a call timer is displayed during active calls.
EffectOfRedialButton	Determines whether the deskphone's **Redial** option automatically redials the last number called or presents a list of recently dialed numbers for selection.
ErrorToneEnabled	Determines whether or not an error tone sounds when the user makes a mistake or attempts a disallowed action.
PersonalRingTonePatterm	Sets the ringtone that sounds when a call is received.
ShowPhoneScreenOnAlert	Determines whether or not the Phone Screen is displayed when the telephone rings.
ShowPhoneScreenOnCall	Determines whether or not the Phone Screen is displayed when the user places a call.
TimeFormat	Determines whether the time is displayed on the deskphone using the 12- or 24-hour clock format.

<NAME> TAG VALUE	DESCRIPTION OF SETTING
UserDisplayName	Sets the text to be displayed in the top line of the deskphone display screen in place of the extension number.
UserPreferredLanguage	Sets the language to be used for text on the deskphone display. The language file must have been listed in the LANGUAGES parameter in the system-wide Settings file loaded when the deskphone was last reset.
UseVisualAlerting	Determines whether or not the voicemail message light, typically on the top-right corner of the deskphone, flashes when the telephone rings.
WMLBrowserIdleUrl	Sets the URL of the WML web page to be displayed when the deskphone remains idle for the period defined in the WMLIDLETIME setting in the system-wide Settings file. If set to an empty string, the feature is turned off.

Table 2-1: Deskphone settings that can be configured using Phonexml Push

Clear Telephone History Request

A clear telephone history request allows a client application to dynamically:

- Clear a deskphone's call history.
- Clear a deskphone's web history.
- Clear a deskphone's contacts history.
- Clear all user-defined options and settings on a deskphone, and resets them to the default values defined in the system-wide Settings file.

Refresh Resource Requests

A refresh resource request allows a client application to dynamically:

- Update the content of a deskphone's non-browser user interface, including labels, menu options, softkeys functionality and disabling hardkeys.
- Upload a language file to be used for text on the updated user interface display.

Refresh resource requests are described in detail in *Chapter 4: Customizing the Avaya IP Deskphone User Interface.*

SUBSCRIBE PUSH

Subscribe Push is used to prompt a deskphone to send its details to one or more registration applications. The registration application can use the details to update a database that can subsequently be used by the deskphone application to access information about the deskphones in the network and determine which type of content is pushed to which deskphones. The deskphone can only send its details to services listed under the SUBSCRIBELIST parameter in the system-wide Settings file – see the section on *Configuring systems to support Push* earlier in this chapter.

We will take a detailed look at the uses of Subscribe Push in the next chapter, *Using Subscription Services and Global Variables.*

User experience

Subscription Pushes are performed in the background, without the user being aware that they are happening. Normal and barge priorities do not apply to Subscribe Push Initiation requests because they don't interfere with any other activities at the deskphone.

ROUND-UP

In this chapter you learned about Push: what it is, how it works, what it is used for and the available Push types. In the next chapter, we are going to look in more detail at how applications can make use of subscription services and global variables to access information about the deskphones in a network.

CHAPTER 3
USING SUBSCRIPTION SERVICES AND GLOBAL VARIABLES

IN THIS CHAPTER:

- About targeted and personalized content
- Using subscription services
- Using global variables

ABOUT TARGETED AND PERSONALIZED CONTENT

As you learned in *Chapter 1,* IP deskphone applications can deliver content to deskphones, typically in the form of a WML file containing messages or menu option links. Users can either request the content by selecting a menu option from the deskphone browser or native display screen (a Pull), or the content can be sent unsolicited by an application (a Push).

The content may be generic, with the same message sent to all deskphones regardless of which it was requested from or pushed to; or the message could be tailored for a specified deskphone or group of deskphones. There are two ways in which content can be tailored for specific deskphones:

- Targeted content: the whole message is specific to a user or group of users. For example, all invitees are sent a message reminding them that a multi-media conference is about to start – the message is specific to, and targeted at, the deskphones of invitees.
- Personalized content: the message can include information about the specific deskphone or user it is sent to, such as the extension number or user name.

The ability to target or personalize content can be exploited by applications to provide solutions in many practical scenarios. For example:

- A Push application can target an evacuation alert message to all deskphones in a specific geographical location.
- A web application can deliver a WML form to the deskphone web browser, pre-filled with the user name and extension.

ACCESSING DESKPHONE INFORMATION

To target or personalize content, an application must be able to access information about the deskphones in the system, and their associated users. Armed with the information, applications can tailor content to provide personalized messages, services and options. So, how do applications obtain this information? There are two main methods:

1. **Subscription Service:** a registration application or script that collects information about the deskphones that subscribe to the service, and typically saves the information to a database or flat file. A deskphone application may include its own registration capability to maintain information about the deskphones that interact with the application, or it may make use of a database maintained by an external subscription service. Deskphone applications can optionally perform database queries on the information collected by a subscription service and use it to target or personalize content. Subscription services are particularly valuable for applications that target large numbers of deskphones.

2. **Global Variables:** global variables are used to pass specific information about an individual deskphone between the deskphone and the application, at run time. Global variables can be used in URLs so that the information is automatically included in requests from the deskphone to the application, or in the web content returned by the application to the deskphone.

We'll look at both of these methods in this chapter, and compare their relative merits and limitations. Obviously, it is possible, and sometimes desirable, for a deskphone application to combine both methods, each in different scenarios.

Global variables are supported by Avaya one-X® deskphones with H.323 firmware release 3.1 or higher installed.

USING SUBSCRIPTION SERVICES

A subscription service maintains information about all of the deskphones in a network that subscribe to that service. When a deskphone is subscribed to a service, it sends information about itself to the service whenever it is:

- Registered or reset,
- Prompted by an application that sends a Subscribe Push to the deskphone.

We'll look at these methods in more detail later in this section, under *How deskphones update subscription services.* Deskphones can subscribe to multiple services.

THE INFORMATION SENT TO SUBSCRIPTION SERVICES

When a deskphone is subscribed to a service, it sends the following information about itself to the subscription service:

- IP address
- MAC address
- Extension
- Model

Typically, this information is used to update the database maintained by the subscription service. The deskphone information collected by the subscription service can be combined with personnel data collected from other sources to allow applications to access information about deskphone users, such as their:

- Geographical location
- User name
- User department, role, level, position, etc.

This information can subsequently be retrieved from the database by a deskphone application and used in a number of ways. For example:

- Applications define the deskphones they want to push content to by their IP addresses. Using the database, the application can look up the IP address of a given extension, the IP address of a given user's deskphone, the IP addresses of all deskphones belonging to employees in a given office or department, or of all employees at a given level in the organization.
- Applications can look up deskphone and associated user information based on a known key, such as the telephone extension, and use it to personalize messages sent to that deskphone.
- Deskphone applications can use information about deskphone models to determine which types of Push they can support. For example, an application could implement logic to send a Display Push to all deskphones with a browser and a Top Line Push to other deskphones.

Note: As you learned in the previous chapter, just because a particular deskphone model **can** support a particular Push type, it will not necessarily accept Pushes of that type: the installed firmware type and release, and the system settings, also affect which types of Push will be accepted by a deskphone.

HOW DESKPHONES UPDATE SUBSCRIPTION SERVICES

There are two ways in which deskphones can send their details to subscription services:

- On registration or reset
- On receipt of a Subscribe Push Initiation request

Let's look at each of these in turn.

Registration Subscription

As we learned in the previous chapter, in the section *Configuring systems to support Push*, the system-wide Settings file includes the SUBSCRIBELIST parameter, which defines the URL locations of all the subscription services to which the deskphones in the network can subscribe. Whenever a deskphone is registered on the network or reset, it automatically sends its details to all of the subscription services defined in the list.

When an application provides its own subscription service, it is likely to be located on the Initiation Server with the application

Push Subscription

Client applications can prompt a deskphone to re-subscribe to one or more services by sending it a Subscribe Push Initiation request. This capability allows client applications to ensure that the information they use to push content to deskphones is kept up-to-date. **Figure 4-1** shows a typical Subscribe Push Initiation request.

```
<?xml version="1.0"?>
<Push type="subscribe">
 <go href="http://trusted_push_server_ip/subscribe.xml" method="get" />
</Push>
```

Figure 4-1: Sample Subscribe Push Initiation request

Notice that Subscribe Push requests do not have an **alert** or **mode** attribute: this is because subscriptions are processed in the background without the user being made aware and without being effected by other activities at the deskphone.

Figure 4-2 shows an example of the Push Content in the `subscribe.xml` file retrieved from the Trusted Push Server.

```
<?xml version="1.0"?>
<Response>
 <Subscribe type="me">
 <Url href="http://push_initiation_server/subscription_service.php" />
 </Subscribe>
</Response >
```

Figure 4-2: Sample Subscribe Push Content

The **<Response>** tag contains the **<Subscribe>** element, which has one attribute:

- **type:** if set to "all", the deskphone subscribes to all services listed in the SUBSCRIBELIST parameter. If set to "me", the deskphone subscribes only to the service whose location is defined in the **href** parameter of the inner **<Url>** element.

If the request is to subscribe to a specified service, that service must be included in the list of services defined in the SUBSCRIBELIST parameter. In the example, the application provides its own subscription service, using the `subscription_service.php` script located with the application on the Push Initiation server.

A deskphone application can send a Subscribe Push immediately before other Push types to ensure that a deskphone is still on the network and to obtain deskphone information without needing to perform a database look-up.

USING GLOBAL VARIABLES

Remember, global variables are only supported by Avaya one-X deskphones with H.323 firmware release 3.1 or higher installed.

Global variables allow applications to access information about deskphones without using a subscription service or performing database queries. The variables provide access to information same similar to that collected by subscription services, namely:

- $IPADD IP address
- $MACADDR MAC address
- $PHONEXT Extension
- $MODEL Model

WHY USE GLOBAL VARIABLES?

Using global variables has a number of advantages over subscription services:

- Applications can access information about deskphones without having to use a subscription service or perform database queries. This greatly simplifies the programming of the application.
- Database queries are based on the IP address of the deskphone making the request to the application. However, the true IP address of a deskphone can sometimes be masked by a proxy server, causing the query to fail or to return incorrect information.

Global variables do not allow applications to access information about the users of deskphones. To include user information in personalized messages requires the use of a database populated with personnel data obtained from other sources.

HOW GLOBAL VARIABLES CAN BE USED

Global variables can be used in a number of ways:

- In WML content sent to deskphones
- In request URLs or associated postfields
- In Display Push Initiation messages

WML Content

WML content sent to a deskphone by a Push or Pull application can contain global variables. When the content is rendered by the deskphone browser, the variables are replaced by the appropriate values for the deskphone.

Variables can be referenced anywhere in the WML content. For example, many deskphone applications that require the user to log in use the extension number as the User Id: using the $PHONEXT variable within the **<input>** tag on the log in page allows the User Id to be automatically pre-populated:

```
<?xml version="1.0"?>
<wml>
 <card id="login" title=" Please log in:">
 <p>
  User Id: <input type="text" name="userid" size="10" value="$PHONEXT" /><br />
     Password: <input type="password" name="passwd" size="10" />
 </p>
 …
 </card>
</wml >
```

Web Application URLs and postfields

When making a request to a web application, global variables can be referenced in URLs or associated postfields to include information about the deskphone. For example, to include the deskphone's extension in a request, the variable can be referenced in the **href** parameter of an **<a>** tag on a WML page:

```
<a href="http://135.8.63.61/script.php?var=$PHONEXT" />
```

or in a **<postfield>** tag:

```
<anchor>
 <go href="http://135.8.63.61/script.php">
 <postfield name="var" value="$PHONEXT" />
 </go>
</anchor>
```

In both cases, the variable is replaced by the actual extension number of the deskphone making the request. The application receiving the request (in this case script.php) can use the extension number to personalize its response.

Display Push Initiation Requests

Display Push Initiation messages can include global variables in the URL of the web content or in the associated postfields. The deskphone replaces the variables with the appropriate values when it subsequently requests the display content from the Trusted Push Server. For example:

```
<?xml version="1.0"?>
<Push alert="3" type="display" mode="barge">
  <go href=" http://135.8.63.61/script.php" method="post">
    <postfield name="var" value="$PHONEXT"/>
  </go>
</Push>
```

Again, the variable is replaced by the actual extension number of the deskphone making the request. The application receiving the request can use the extension number to personalize the display content that it returns to the deskphone browser.

ROUND-UP

In this chapter you learned how applications can access information about deskphones using subscription services or global variables, and how applications can use the information to target and personalize content sent to the deskphones. Next, you will learn how to customize the content and look-and-feel of the deskphone's native display.

CHAPTER 4
CUSTOMIZING THE AVAYA IP DESKPHONE USER INTERFACE

IN THIS CHAPTER:

- About deskphone user interface customization
- Reasons for customizing the deskphone user interface
- Applying user interface customization to deskphones
- Customizing the deskphone user interface content
- Customizing the deskphone user interface look-and-feel

At the time of publication, the information in this chapter applies only to Avaya one-X® Deskphones (9600 Series) models with SIP firmware release 2.2, 2.5 or higher installed. Note, however, that not all 9600 Series deskphones support SIP firmware or, by extension, deskphone user interface customization.

ABOUT DESKPHONE USER INTERFACE CUSTOMIZATION

There are two main facets of a deskphone's native, non-browser user interface that can be customized:

- **Content:** the menu options and functionality available via the deskphone user interface can be customized. This includes title line and prompt line text, menu options, labels, softkey assignments and hard button disablement. Content customization is defined in a Deskphone XML file.
- **Skin:** the look-and-feel of the deskphone user interface can be customized. This includes background images and colors, text colors and softkey label shapes. Look-and-feel customization is defined in a Skin XML file.

In this chapter we will start by looking at how customization can be used to enhance the user experience, the ways in which deskphone user interfaces can be customized and how customization can be applied. We will then go on to look at the individual elements of the deskphone user interface that can be customized, including the structure of the Content and Skin customization files and what they enable you to do.

REASONS FOR CUSTOMIZING THE DESKPHONE USER INTERFACE

Being able to customize the content and look-and-feel of a deskphone's user interface provides a number of important benefits:

- The options and features available from a deskphone can be tailored for a particular user, group or organization.
- New and updated menu options and features can be made available without the need to manually reset the deskphone.
- Time and event-specific options and features can be made available. For example, in a hotel, an option could be made available that allows a guest to pre-order their breakfast from a menu of available items. The option could be pushed to the deskphone with the guest's wake-up call and removed at the end of the breakfast period. Another example: during an arts festival, options could be made available to obtain information or order tickets for various events.
- A user's experience of the deskphone can be personalized with their own name, welcome message, company colors and logo, etc.

One of the use cases discussed in *Chapter 7*, which demonstrates the practical application of deskphone programming in the hotel industry, includes an example of deskphone user interface customization.

APPLYING USER INTERFACE CUSTOMIZATION TO DESKPHONES

Both content and skin customization can be applied in two ways:

- When the deskphone is registered or reset.
- When an application uses the Push API to send an appropriate Phonexml Push request to the deskphone.

REGISTRATION CUSTOMIZATION

Whenever a deskphone is registered or reset, it downloads the Content and Skin customization files defined in the system-wide Settings file. The following parameters in the Settings file determine which files are uploaded and applied:

- **CURRENT_CONTENT:** Defines the URL of the Deskphone XML file to be downloaded by the deskphone and applied to its user interface.
- **SKINS:** Defines the names and URLs of the Skin XML files to be downloaded by the deskphone. Each Skin file represents a different look-and-feel that can be applied to the deskphone's user interface.
- **CURRENT_SKIN:** Defines the name of the skin to be applied to the deskphone's user interface. This must be one of the skins defined in the SKINS parameter.

If these parameters are set, the files are downloaded from a web server and applied at the deskphone. If the parameters are not set, the deskphone's default content and skin are used.

Users can subsequently apply a different look-and-feel by selecting an alternative skin from the deskphone's **Settings and Options** menu. Only skins that have been downloaded to the deskphone are available for selection. Users cannot change the user interface content.

PUSH CUSTOMIZATION

Applications can use the Push API's Phonexml Push type to dynamically:

- Download and apply a different Deskphone XML file at a deskphone. A Phonexml Push can also be used to download additional language files to the deskphone, containing the text to be used for the content.
- Apply a different skin to a deskphone. The skin must already have been downloaded to the deskphone when it was registered or last reset.

Note that it is not possible to use Phonexml Push to push a new or updated skin to a deskphone: you can only change which of the skins, previously uploaded at registration, is applied.

Phonexml Push is described in Chapter 2, in the section *The available Push types*

CUSTOMIZING THE DESKPHONE USER INTERFACE CONTENT

In this section we will look at the aspects of the user interface that can be customized by downloading and applying a Deskphone XML file at a deskphone.

Deskphone XML files have a well-defined schema, the elements of which are described in detail in the *Avaya one-X Deskphone Edition for 9600 Series SIP IP Telephones Developer Guide [3].*

Avaya provides a comprehensive range of tools to help you create valid and well-formed Deskphone XML files, including the Avaya one-X Deskphone XML Designer, Emulator, Validator and Templates. Read Chapter 6 to find out about the available tools and how they are used to create Deskphone XML files: in this chapter we will concentrate on what Deskphone XML allows you do.

CUSTOMIZABLE DESKPHONE USER INTERFACE CONTENT ELEMENTS

To begin, let's look at the elements of a deskphone's user interface content that can be customized. **Figure 4-1** shows the locations of the customizable content elements on the display of an Avaya 9640 IP deskphone, and the elements are described below the figure.

Figure 4-1: Customizable content elements

Cards:

Customized user interface content can be displayed across multiple pages, known as cards. Each card is assigned a unique name to identify it. One of the cards must be defined as the "root": this is the card that acts as the user's home page. For each card, the name of the previous and next card in a sequence can be defined. Users navigate to the next and previous cards in the sequence using the **Left** and **Right** navigation keys.

Password Restricted Cards:

Passwords can be used to restrict access to hidden pages. A password is a fixed-length, four-digit number associated with a particular card: to access the card, the user presses Mute and enters the password followed by # on the deskphone keypad. For example, in a hotel, a card containing special menu options for room cleaners only could be assigned a password: the cleaners enter the password to access the menu options, but room guests are unaware of the card's existence and prevented from accessing it.

Top line, title line and prompt line:

The text displayed on the title line and prompt line of the deskphone display can be defined at content level across all cards or overridden for individual cards. In addition, the prompt line can be used to display context-sensitive, help text when a menu option has focus and is highlighted.

Applications lines:
Application lines display the menu options available on each card. If more menu options are defined than are viewable in the applications area, a scroll bar is automatically included. Each application line can have a label describing what the menu option is and an associated action that defines what happens when the menu option is selected. In addition, prompt line text can be defined for each menu option and be displayed when it has focus and is highlighted. Users can highlight a menu option using the **Up** and **Down** navigation buttons. Depending on the deskphone model, a menu option can be selected by highlighting it and pressing the **OK** button or by pressing the adjacent line button.

Softkey Labels:
The labels and actions associated with the deskphone's softkeys can be customized for each card. If more softkeys are defined on a card than there are physical softkeys on the deskphone, the right-hand softkey is automatically labeled **More** and used to access the additional softkey functionality.

Hard buttons:
A number of the deskphone's hard-wired, physical buttons can be selectively disabled, preventing users from accessing **Call Log, Contacts, Forward, A**(vaya) **Menu, Message** and/or **Headset** functionality. For example, a hotel may not want guests to be able to change the deskphone's default settings so would disable the **A Menu** button.

Note: Hard buttons are not shown on **Figure 4-1**; see **Figure 1-1** for the locations of hard buttons, navigation buttons and line buttons on an Avaya one-X 9640 Deskphone.

MULTIPLE LANGUAGE SUPPORT FOR CUSTOMIZED PAGES

It is possible to bind language files to Deskphone XML documents so that all text on the customized pages is displayed in the user's preferred language. So, how is it done?

1. Instead of specifying text strings in the Deskphone XML file, you use placeholder variables. For example, to specify the prompt line text associated with a menu option, instead of:

   ```
   <label>Select to dial</label>
   ```

   you would use:

   ```
   <label>${SELECT_TEXT}<label>
   ```

2. Create a separate language resource file for each language that you want to support, and define the text to be displayed for each placeholder variable.
3. Copy the language files to a web server.

4. Bind the language files to the Deskphone XML file, which involves simply listing the language file URLs in the document.
5. Download the language files to the deskphone. This can be done in one of two ways: either by listing the language files under the LANGUAGES parameter in the system-wide Settings file so that they are automatically downloaded when the deskphone is registered or reset; or by using a Phonexml Push to dynamically upload the language files to the deskphone.

By default, text is displayed in the language defined in the SYSTEM_LANGUAGE parameter in the system-wide Settings file. Users can subsequently select a different language from the deskphone's **Options and Settings** menu, or the language can be changed dynamically by a client application using a Phonexml Push.

CUSTOMIZED MENU OPTION AND SOFTKEY ACTIONS

As stated above, it is possible to customize the actions associated with menu options and softkeys. So, what actions can be invoked and what do they do?

Navigate to another card:
This action displays another card defined in the Deskphone XML file. The option allows you to define a hierarchy of menus and sub-menus that the user can navigate. For each card, it is also possible to define the next and previous card in a linear sequence. Users navigate to the next and previous cards in the sequence using the **Left** and **Right** navigation keys.

Launch an application
This action launches one of the deskphones onboard applications:

- **Call Log:** launches the Call Log, listing the user's outgoing, answered and missed calls.
- **Contacts:** allows the user to initiate a directory search or find a contact in their personal Contacts list.
- **Feature:** launches the Features menu from which the user can access Speed Dial buttons and advanced telephony features, such as Call Forwarding.
- **Options:** launches the Avaya menu from which the user can adjust and customize their deskphone settings.
- **Web:** launches the deskphone's web browser.

Open a WML web page
This action launches the deskphone's web browser and displays the specified URL. This could be a static intranet or internet WML page, or the user interface for a web-based application.

Dial a number
This action automatically dials the specified telephone number.

In addition, when an action is associated with a menu option, it is possible to specify prompt line help text to be displayed when that option has focus and is highlighted. For example, you could display the message "Click OK to call reception:" on the prompt line when a menu option with a dial action associated with it has focus.

CUSTOMIZING THE DESKPHONE USER INTERFACE LOOK-AND-FEEL

Skin XML files have a well-defined schema, the elements of which are described in detail in the *Avaya one-X Deskphone Edition for 9600 Series SIP Telephones Developer Guide [3].* In this section we will look at the aspects of the user interface that can be customized by downloading and applying a Skin XML file at a deskphone.

CUSTOMIZABLE SKIN ELEMENTS

The following elements of the user interface look-and-feel can be individually customized in the Skin XML file:

- Background
- Logo
- Top line
- Title line
- Prompt line
- Application lines (with and without focus)
- Web lines (with and without focus)
- Softkeys
- Pop-ups
- Information area
- Scroll bar

For each of these elements the following properties can be defined:

- Image (including the image URL, size and position) or shape (rectangle, line or rounded rectangle, including position, size, color, border color and border size)
- Text color
- Background color

All images must be in the JPEG format.

To achieve the highlight effect when an application line or web line has focus, it is necessary to define separate properties for the elements with and without focus, including using color-reversed or other visually differentiated image files.

Figure 4-2 and **Figure 4-3** show the effect of applying different skins to the default display.

Figure 4-2: Default skin

Figure 4-3: Custom skins

ROUND-UP

In this chapter we have looked at the various options available for customizing the content and look-and-feel of Avaya IP deskphones, and at the ways in which that customization is defined and applied. In the next chapter, we will go on to investigate the creation of web content and browser-based applications for IP deskphones.

CHAPTER 5
DEVELOPING WEB APPLICATIONS FOR AVAYA IP DESKPHONES

IN THIS CHAPTER:

- About deskphone browsers and web applications
- Deskphone web browser capabilities
- The structure and content of WML pages
- Creating web pages and web applications
- Accessing web pages
- Latest Avaya one-X® deskphone browser features

ABOUT DESKPHONE BROWSERS AND WEB APPLICATIONS

Many Avaya IP deskphone models incorporate a browser, capable of displaying web pages written in Wireless Markup Language (WML). This opens up a whole range of possibilities for extending the capabilities of the deskphone, from being just a conventional telephony end point to being an interactive information hub and an interface to practical business and leisure applications.

The capabilities and features of a deskphone browser vary depending on the deskphone model, particularly at the very bottom and top ends of the range. However, most Avaya IP deskphones have a browser with features similar to those described here.

In this chapter, we will look at the technologies supported by the deskphone browser, the capabilities of the browser itself, how web content and applications are created and the ways in which users can access web pages. At the end of the chapter we will take a quick look at the advanced browser capabilities available with the latest top-of-the-range Avaya one-X deskphones.

DESKPHONE WEB BROWSER CAPABILITIES

The browser provided with Avaya IP deskphones can display and/or support the following technologies used to create web content:

- WML
- WML forms, to support user input
- Images

- WTAI functions
- WMLScript

We'll look at each of these technologies below.

Note: The latest top-of-the-range Avaya one-X deskphones support additional browser capabilities. These discussed separately at the end of this chapter.

WML

Wireless Markup Language was originally designed to create web pages that can be displayed on the small WAP browsers often found on mobile devices. WML is similar to HTML in that it provides navigational support, data input, hyperlinks, text and image presentation, and forms. However, unlike HTML, WML is a strict XML language and therefore must always be valid and well-formed.

Another difference between WML and HTML is that a WML file can contain multiple pages. A WML page is known as a "card" and a WML file can incorporate a "deck" of cards. This has the advantage that users can navigate between multiple pages without each having to be loaded individually.

Note that the terminology and concept of WML cards and decks is similar to that for Deskphone XML used to customize the deskphone user interface, as described in the previous chapter. However, WML and Deskphone XML are distinct and different languages used to define the browser and non-browser display, respectively.

Avaya IP deskphones support the WML 1.3 specification.

WML FORMS

One aspect of WML that is particularly valuable to developers is the ability to create forms that support user input. Users can enter text into forms and select options using controls such as selection boxes, radio buttons and check boxes. Thus a WML page can act as a user interface to a thin-client web application, allowing users to interact with the application. This opens up a whole raft of opportunities for adding value to desktop telephones and extending the ways in which they can be used.

IMAGES

Avaya IP deskphone web browsers can display images in two formats:

- **JPEG:** the JPEG format is ideal for displaying full-color or grey scale images, such as photographs and detailed diagrams. Although the JPEG format uses image compression, the file size is much larger than for WBMPs.

- **Wireless Bitmap (WBMP):** WBMP was originally designed specifically for displaying images on mobile devices. The images are monochrome, comprising only black and white pixels, thereby minimizing the file size.

Figures 5-1 and **5-2** show the same image saved as a JPEG and a WBMP file and rendered in a deskphone browser.

Figure 5-1: JPEG image

Figure 5-2: WBMP image

Images are selectable and can be used as navigational hyperlinks or to call actions.

WIRELESS TELEPHONY APPLICATIONS INTERFACE (WTAI)

The Avaya IP deskphone browser supports two WTAI functions:*

- **Click-to-dial:** when a user clicks on a **Click-to-dial** WTAI link, the telephone autodials the number specified in the link. **Click-to-dial** links are indicated by a telephone handset icon.
- **Add to phonebook:** when a user clicks on an **Add to phonebook** WTAI link, the name and number specified in the link are added to the user's Contacts list. **Add to phonebook** links are indicated by a plus-sign icon.

These functions can be assigned to application lines or to softkeys.

*The only exception is the 9610 deskphone, which does not support the Add to phonebook WTAI function.

WMLSCRIPT

WMLScript is a lightweight version of JavaScript used to validate user input, generate dialog boxes, display error message and provide other dynamic features on WML pages. WMLScript is to WML what JavaScript is to HTML. For example, WMLScript can be used to check that all mandatory input fields on a page have been completed before the form data is submitted to the client application. For an introduction to WMLScript, go to http://www.w3schools.com/wmlscript/default.asp.

THE STRUCTURE AND CONTENT OF WML PAGES

Like HTML, WML is an XML-based language defined using nested tags and attributes. However, WML pages have a more formal structure than standard HTML pages. The structure of a web page is similar to that of a native user interface page, as described in *Chapter 4*, although there are some differences. **Figure 5-3** shows the elements of a web page; the elements are described below the figure.

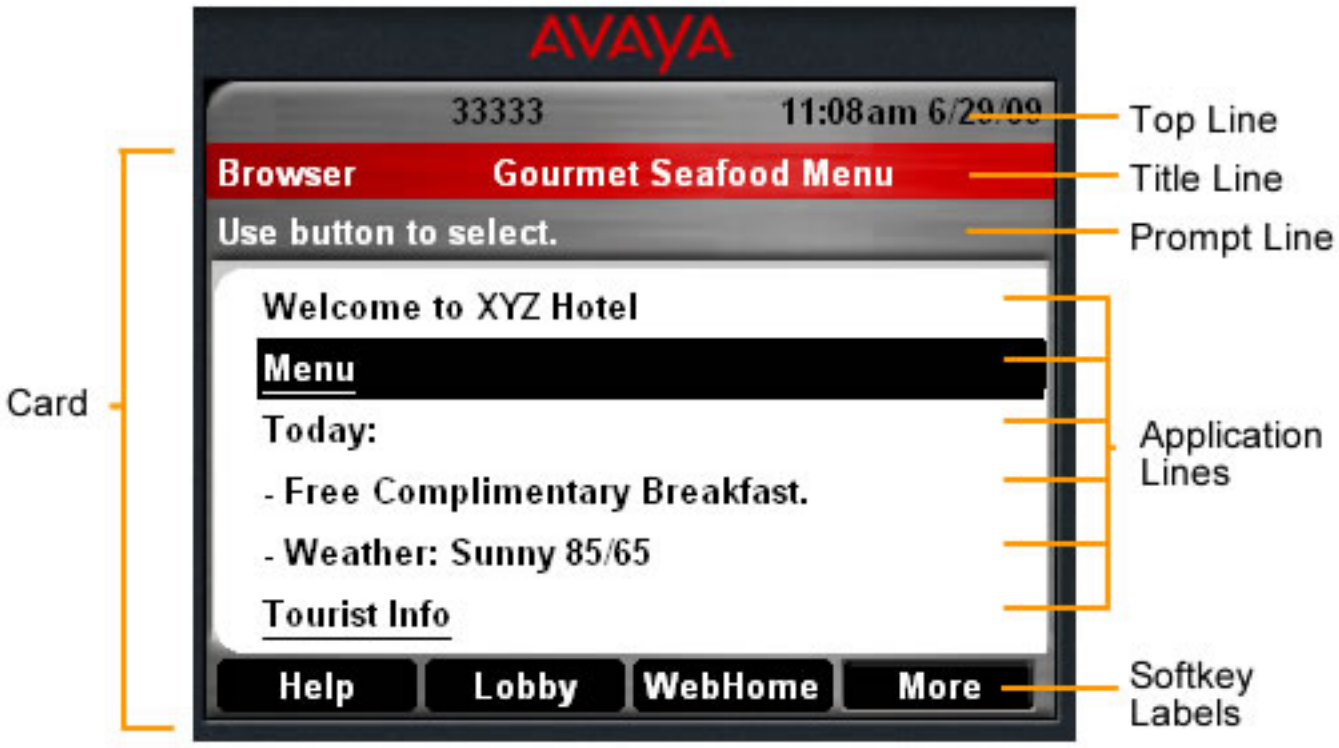

Figure 5-3: WML browser display elements

Cards:
Web content can be displayed across multiple pages, known as cards. Each card is assigned a title and a unique Id, which identifies the card and acts like an anchor name tag for navigation between pages in the deck.

Title line:
The title line shows the title of the currently displayed card, as defined in the **<card>** tag's "title" attribute. The default value is "Browser".

Prompt line:
Displays help text for the currently highlighted application line, as defined in the corresponding **<anchor>**, **<a>**, **<input>** or **<option>** tags' "title" attribute.

Applications lines:
Application lines can be used to display text (in a **<p>** tag), URL links (in an **<a>** tag), action links (in an **<anchor>** tag), images and form controls.

Softkey labels:
The labels and actions associated with the deskphone's softkeys on each card can be defined within **<do>** tags.

Top line: Although part of the display screen, the top line is not within the browser display area and therefore cannot be defined in WML files. This means the current top line message is still displayed when the browser is activated. It also means that an application can push separate Top Line and Display messages to the deskphone at the same time. Applications can take advantage of this by sending a Top Line Push with a Display Push to provide an alternative communication method for deskphones in the network that do not have a browser.

For a full list of supported WML tags, and information on how they are used, see the *Avaya one-X Deskphone Edition for 9600 Series IP Telephones Application Programmer Interface (API) Guide [2].*

BROWSER CAPABILITIES IN DIFFERENT DESKPHONE MODELS

The browser provided with all Avaya IP deskphones will display correctly formed WML files in a consistent and predictable way. However, there are physical and functional differences between the browser in different models, particularly the size of the display area and whether it supports color or grey-scale only. For example, in some models the browser can display up to six Application Lines whereas in others it can display only three.

There are also differences in the way users can interact with WML pages displayed in the browser. For example, some deskphones have line buttons associated with application lines; some top-end models provide a touch screen for selecting options and advance text input features. For a detailed description of the browser provided with each deskphone model, see the *Avaya one-X Deskphone Edition for 9600 Series IP Telephones Application Programmer Interface (API) Guide [2].*

CREATING WEB PAGES AND WEB APPLICATIONS

Static web pages can obviously be authored directly in WML and uploaded to a web server that is accessible from the network. There are a large number of WML editors available to help you, including plug-ins for high-end HTML editors such as Dreamweaver.

Web pages served by applications are more likely to be created dynamically using a server-side scripting language such as PHP, Perl, ASP, ASP.NET, JSP or ColdFusion. Avaya provides a number of sample applications to help you. For example, the sample application *Integrating Twitter as an Avaya IP Telephone Application [6]*, available from the DevConnect web portal, demonstrates how to use PHP to write an application that interfaces with Twitter using its public APIs and displays recent tweets to which a user is subscribed.

ACCESSING WEB PAGES

There are a number of ways in which a particular web page may be displayed in the deskphone browser:

Navigation from the home page:
The browser's home page can be defined using the WMLHOME parameter in the system-wide Settings file. The home page is displayed whenever the user selects the browser and presses the **Home** softkey. The user can navigate to other pages using links from the home page.

Navigation from the "telephone idle" web page:
The WMLIDLEURI parameter in the system-wide Settings file can be used to specify a web page to be displayed when the deskphone has been idle for the period defined in the WMLIDLETIME parameter. The user can navigate to other pages using links from the telephone idle page.

Timer activated:
A timer can be set on a web page so that the next page in a sequence is automatically displayed after a defined period. This capability is useful if, for example, you wish to display rotating advertisements or informational pages at a deskphone that is otherwise idle.

URL input:
Although the deskphone browser does not have a dedicated location field, it is possible to program an application line to allow a user to enter the URL of a WML page they want to display.

Unsolicited push:
The Display Push type can be used to send a web page to the deskphone. Note that a Display Push can be sent even if the WMLHOME parameter is not defined; however, the user cannot access any other web pages. See Chapter 2 for more information about Display Push.

LATEST AVAYA ONE-X® DESKPHONE WEB BROWSER FEATURES

The recently released Avaya one-X 9670G IP deskphone represents a significant advance when it comes to providing users with a richer, user-friendly interface. The 9670G, as shown in **Figure 5-4**, incorporates a screen with additional features that are leveraged by the web browser, including:

- Large 640x480 pixel screen
- Full VGA color screen to provide sharper image display
- Touch-sensitive screen for selecting options
- Touch-sensitive, on-screen QWERTY keyboard for text input (see **Figure 5-5**)
- Touch-sensitive, on-screen softkeys
- Hosts on-board applications including a World Clock, Weather and Calculator application

Figure 5-4: Avaya one-X 9670G IP deskphone

Figure 5-5: Avaya one-X 9670G deskphone showing touch-sensitive, on-screen keyboard

In addition, with H.323 firmware release 3.1 or higher installed, the 9670G supports image map functionality allowing hot touch areas to be added to images, as shown in **Figure 5-6**. Users can interact with the browser and select options by touching the appropriate areas of the image on the screen.

Figure 5-6: Image map on Avaya one-X 9670G IP deskphone

ROUND-UP

In this chapter you learned how applications can leverage the web browser built-in to most Avaya IP deskphones, to deliver content and provide an interactive user interface. In the next chapter we will look at the tools Avaya provides to help you create both browser-based and Push applications.

CHAPTER 6
AVAYA IP DESKPHONE APPLICATION DEVELOPMENT TOOLS

IN THIS CHAPTER:

- About the Avaya IP deskphone application development tools
- Using the PushSDK and PushSDK API
- Using the Deskphone XML Designer and other deskphone user interface customization tools
- Additional DevConnect resources

ABOUT AVAYA IP DESKPHONE APPLICATION DEVELOPMENT TOOLS

Avaya provides an SDK and a number of other development tools to help you create Push applications and customize Avaya one-X® Deskphone user interfaces. These tools are available for download from the DevConnect web portal (**http://www.avaya.com/devconnect** - registration and log in required).

The following tools are available:

- **PushSDK:** includes a client-side Java interface (the PushSDK API) that can be used to create Push applications.
- **Avaya one-X Deskphone XML Designer:** an intuitive, user-friendly editor to help you create valid and well-formed Deskphone XML and Skin XML files.
- **Avaya one-X Deskphone Emulator:** provides a SIP firmware deskphone user interface on your desk-top, on which you can test your Push applications, Deskphone XML and Skin XML files.
- **Avaya one-X Deskphone and Skin XML Validator:** allows you to verify that your Deskphone XML and Skin XML files are valid and well-formed.
- **Avaya one-X Deskphone and Skin XML Templates:** sample files that can be modified to meet your own requirements.

In addition, DevConnect members can order a copy of the Avaya IP Communications Development Environment (IPCoDE). IPCoDE includes desktop installations of Communication Manager and SIP Enablement Services, allowing you to provision the Emulator or other IP deskphones on which you can test and demonstrate your applications under development

In the following sections, we will take a closer look at the use of the PushSDK and the user interface customization tools.

USING THE PUSHSDK AND PUSHSDK API

WHAT IS THE PUSHSDK?

The PushSDK provides an API that can be used by Java applications to generate and send Push Initiation messages to deskphones. In some cases, the API can also be used to create and upload the Push Content that is rendered at the deskphones. The client-side PushSDK API allows Java applications to generate the XML files that can be processed by the native Push API on the deskphones.

The PushSDK download includes:

- The PushSDK WAR file that must be deployed on the Push Initiation web server.
- The PushSDK library file that defines the PushSDK API and which must be included in your PushSDK projects.
- A skeleton `pushproperties.xml` file. This file specifies where log files will be created, the maximum log file size, the logging level, etc. The file also specifies the IP address and port number of a Trusted Push Server to which any Push Content created by the PushSDK will be uploaded and from where it will subsequently be downloaded by the deskphones.
- Javadoc for the PushSDK API.
- PushSDK installation and developer guides.
- Sample Push applications.

To help understand what the PushSDK does, and how it is used, let's look at an example.

EXAMPLE: USING THE PUSHSDK TO PUSH WEB CONTENT TO DESKPHONE BROWSERS

In *Chapter 2* we investigated the anatomy of a Push Initiation message. In this section we will use an example to look at how the PushSDK can be used to create a Push Initiation message and send it to a number of deskphones.

In the example, we will use the PushSDK to create the Display Push Initiation message, shown in **Figure 6-1**, and send it to three deskphones with IP addresses 192.168.152.2, 192.168.152.3 and 192.168.152.4.

```
<?xml version="1.0"?>
<Push alert="3" type="display" mode="barge">
      <go href="http://192.168.152.1:8080/display.wml" method="post">
            <postfield name="date" value="June 30, 2009"/>
      </go>
</Push>
```

Figure 6-1: Example Display Push Initiation message

The PushSDK API provides two methods for creating and sending this message. Which method you use depends on whether the WML file exists or whether you want to create it as part of the Push request. We'll look at both methods in a moment, but first there are some preliminary steps that the application needs to perform:

- Define the location of the `pushproperties.xml` file.
- Apply the settings in the `pushproperties.xml` file.
- Create a Push object.

Figure 6-2 shows the code snippet for performing these preliminary steps.

```
try {
      ChangePropertyFile.path="C:/AvayaIPPhonePushSDK/PushSDKLib/pushproperties.xml";
      Push.LoadProperties();
      Push pushObject = new Push();
      …
}
catch(Exception e) {
      …
}
```

Figure 6-2: Instantiating a Push object

The same Push object can be used for all the different types of Push you want your application to be able to perform; simply call the appropriate push method on the object. For the example, there are two possible push methods that could be used, both called `pushDisplay()`, but with different signatures.

Pushing an existing WML file

To push an existing WML file to the deskphones, you use the `pushDisplay()` method with the signature:

```
pushDisplay(String[] phoneIp, String wmlFile, String alert, String mode,
      HashMap<String,String> postfields, int connectionTimeOut)
```

where:

- **phoneIp** is an array of deskphone IP addresses to which the WML file is to be pushed.
- **wmlFile** is the absolute location (IP address, port number and file name) of the existing WML file on a Trusted Push Server.
- **alert** is the number of ring-pings to be sounded at the deskphone to notify the user of the incoming message, i.e. “0”, “1”, “2” or “3”.
- **mode** is the Push priority, i.e. “normal” or “barge”.
- **postfields** are key value pairs to be sent in the HTTP POST requests from the deskphones to the Trusted Push Server. If set to null the deskphones will use an HTTP GET to request the Push Content.
- **connectionTimeout** is the maximum time, in seconds, that the Servlet will attempt to connect to the deskphone. The value should be between five and ten seconds.

Figure 6-3 shows the code that generates the example Push Initiation message and sends it to the deskphones.

```
String[] phoneIps = {“192.168.152.2”,”192.168.152.3”,”192.168.152.3”};

HashMap<String,String> keyValuePairs = new HashMap<String,String>();

keyValuePairs.put(“date”,”June 30, 2009”);

PushStatus[] pushStatus = pushObject.pushDisplay(phoneIps,
      ”http://192.168.152.1:8080/display.wml”,
      ”3”,”barge”,keyValuePairs,5);
      ...
```

Figure 6-3: Pushing an existing WML file

Notice that the method returns an array of PushStatus objects containing the x-Avaya-Push-Status code returned by each deskphone, and which can be inspected to determine whether the pushes were successful.

Creating and pushing a new WML file

The second `pushDisplay()` method allows you specify a text message, which is converted to a WML file and copied to the location on the Trusted Push Server specified in the `pushproperties.xml` file. This method also allows you to specify a message that is sent as an accompanying Top Line Push. The method has the following signature:

```
pushDisplay(String[] phoneIp, String txtMsg, String title, String alert,
      String mode, HashMap<String,String> postfields, int connectionTimeOut)
```

Notice that the `wmlFile` argument is omitted, but that there are two additional arguments:

- **txtMsg** is the message that will be converted to WML.
- **title** is the message that will be pushed to the deskphone display's top line.

Figure 6-4 shows the code that generates the example Push Initiation message and sends it to the deskphones.

```
String[] phoneIps = {"192.168.152.2","192.168.152.3","192.168.152.3"};

HashMap<String,String> keyValuePairs = new HashMap<String,String>();

keyValuePairs.put("date","June 30, 2009");

PushStatus[] pushStatus = pushObject.pushDisplay(phoneIps,
      "Here is the message to be displayed in the browser","Here is the top
      line message","3","barge",keyValuePairs,5);
      ...
```

Figure 6-4: Creating and pushing a WML file

So what happens when this method is called?

1. The text message is converted to a temporary WML file and copied to the Trusted Push Server.
2. The title message is converted to an XML Top Line Push Content file and copied to the Trusted Push Server.
3. A Display Push Initiation message and a Top Line Push Initiation message are created and sent to each of the three deskphones.
4. The deskphones retrieve the WML file and Top Line XML file from the Trusted Push Server.

Using the PushSDK in a production setting

In practice many of the arguments passed to the push methods would be sourced from user input at a user interface or from a database. For example, the application could provide an interface that allows the user to select a location, department or series of extension numbers. The application would use a database, maintained via a subscription service, to look up the selected values and convert them to the deskphone IP addresses passed to the selected push method. The interface would also allow the user to enter message text, content file locations, etc.

The sample applications provided with the PushSDK download include JSP, HTML and JavaScript code used to create the user interfaces. You can use this code to help you create your own applications.

For detailed instructions on how to install the PushSDK and use the PushSDK API, see the documentation included in the PushSDK download *[5].*

USING THE DESKPHONE XML DESIGNER AND OTHER DESKPHONE USER INTERFACE CUSTOMIZATION TOOLS

Avaya provides a number of tools to help you customize Avaya one-X deskphone user interfaces. These tools can be installed on you desktop machine to provide a complete development environment for creating, testing and demonstrating your deskphone user interface projects.

AVAYA ONE-X® DESKPHONE XML DESIGNER

The Avaya one-X Deskphone XML Designer provides an intuitive, user-friendly editor to help you create valid and well-formed Deskphone XML and Skin XML files. **Figure 6-5** and **Figure 6-6** show the Designer's main editing screens for creating Deskphone and Skin XML files, respectively.

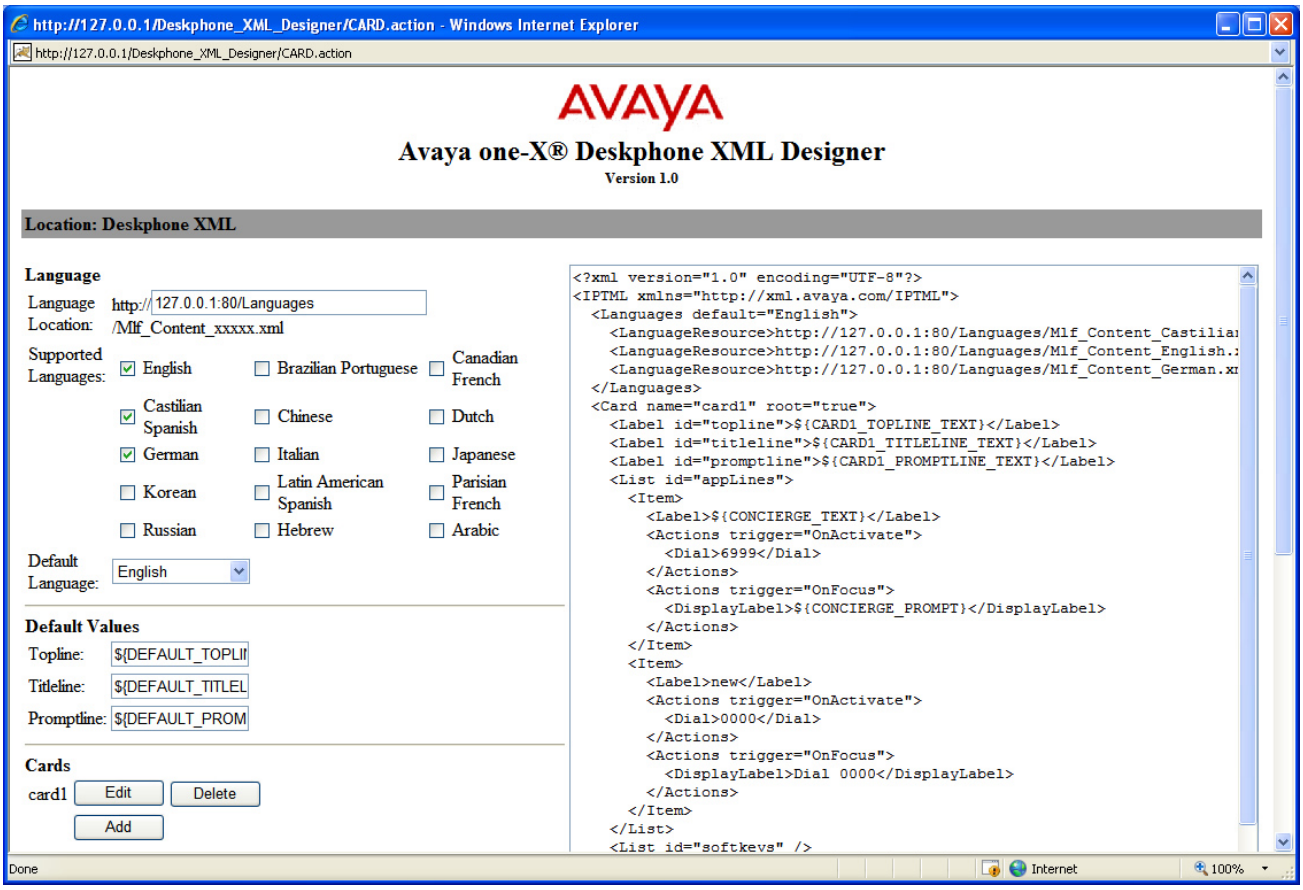

Figure 6-5: The Avaya one-X Deskphone XML Designer: Create Deskphone XML

As you enter required values in input fields on the left, the generated Deskphone XML code is displayed on the right.

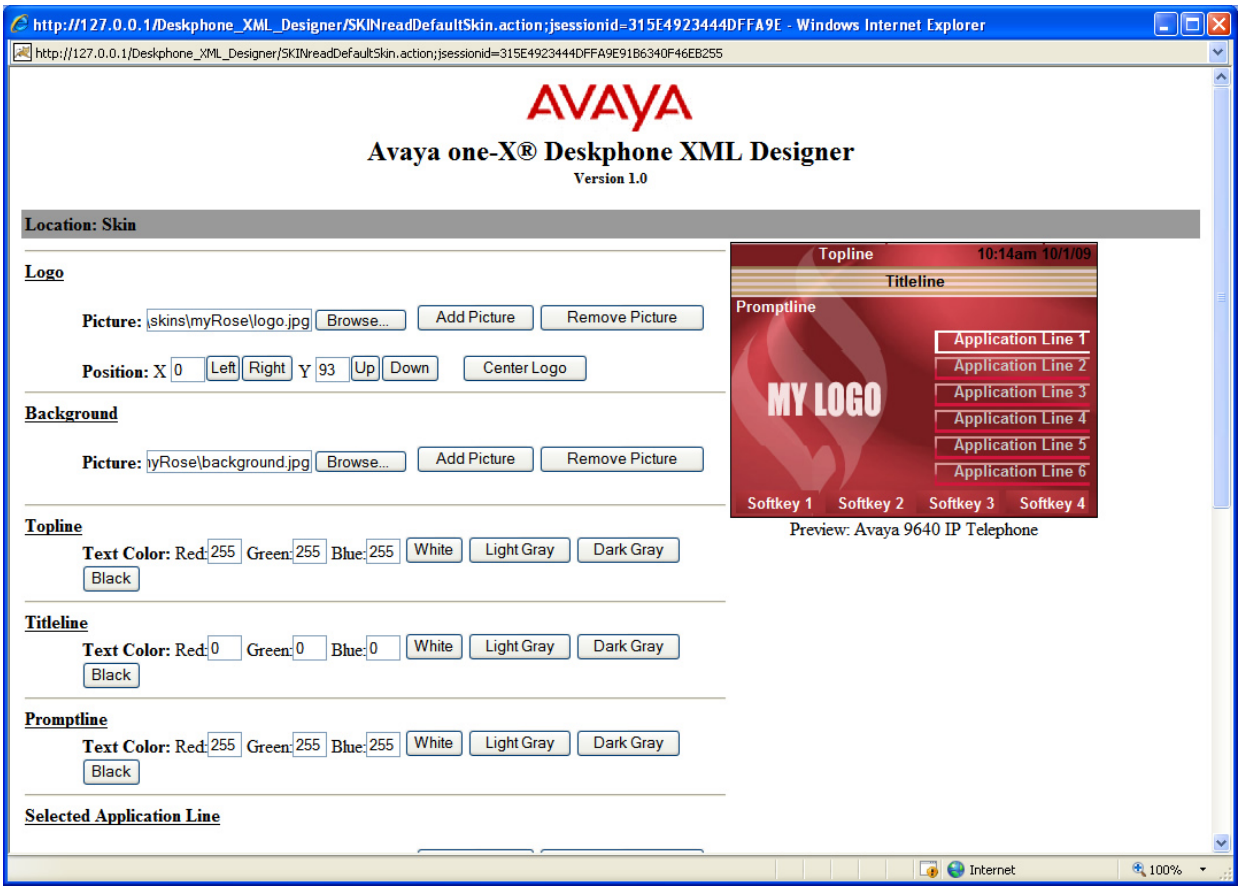

Figure 6-6: The Avaya one-X Deskphone XML Designer: Create Skin XML

As you enter values in input fields on the left, the affect on the look-and-feel of the display is shown on the right.

AVAYA ONE-X® DESKPHONE EMULATOR

The Avaya one-X Deskphone Emulator provides an emulation of the user interface on a range of Avaya one-X IP deskphones with SIP firmware installed. The emulator allows you to test and demonstrate your Deskphone XML and Skin XML files.

Note: The Emulator requires an installation of Avaya Communication Manager and Avaya SIP Enablement Services, so that it can log into a provisioned SIP station. If you don't have a production system available, Communication Manager and SIP Enablement Services installations can be provided on the desktop by using Avaya IPCoDE.

Figure 6-7 shows the Emulator with a Deskphone XML and Skin XML file loaded.

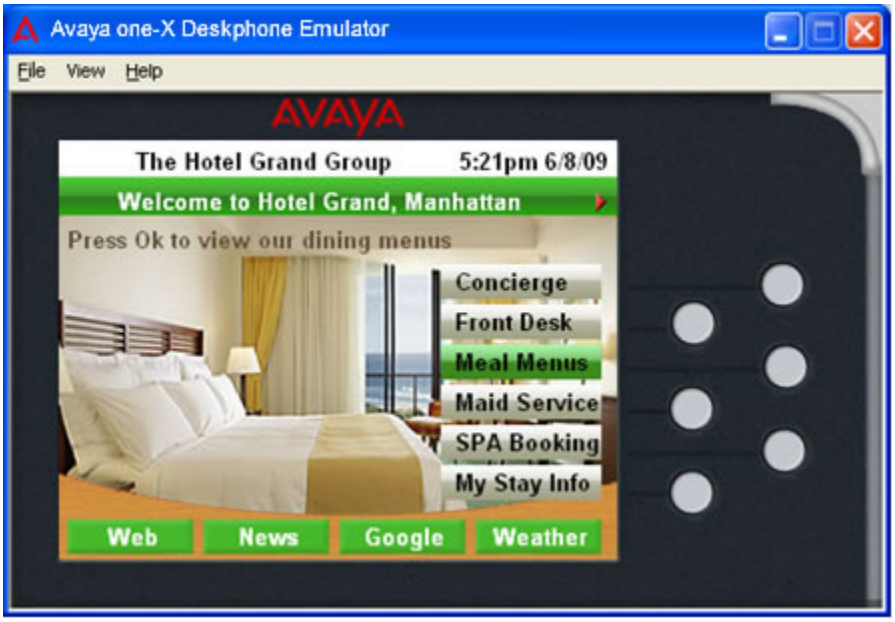

Figure 6-7: The Avaya one-X Deskphone Emulator

The Emulator allows you to perform most of the tasks you can perform on the equivalent deskphone models, including making calls, creating a contacts list and personalizing settings.

For more detailed information about the Designer and Emulator see the *Avaya one-X Deskphone Designer and Emulator Installation and Configuration Guide [7] and Avaya one-X Deskphone Designer and Emulator User Guide [8].*

AVAYA ONE-X® DESKPHONE AND SKIN XML VALIDATOR

The Avaya one-X Deskphone and Skin XML Validator allows you to check that the Deskphone and Skin XML files you create are valid and well-formed before you use them in your applications. The Validator also allows you to view the schema for both file types. **Figure 6-8** shows the Validator being used to validate a Deskphone XML file that has errors.

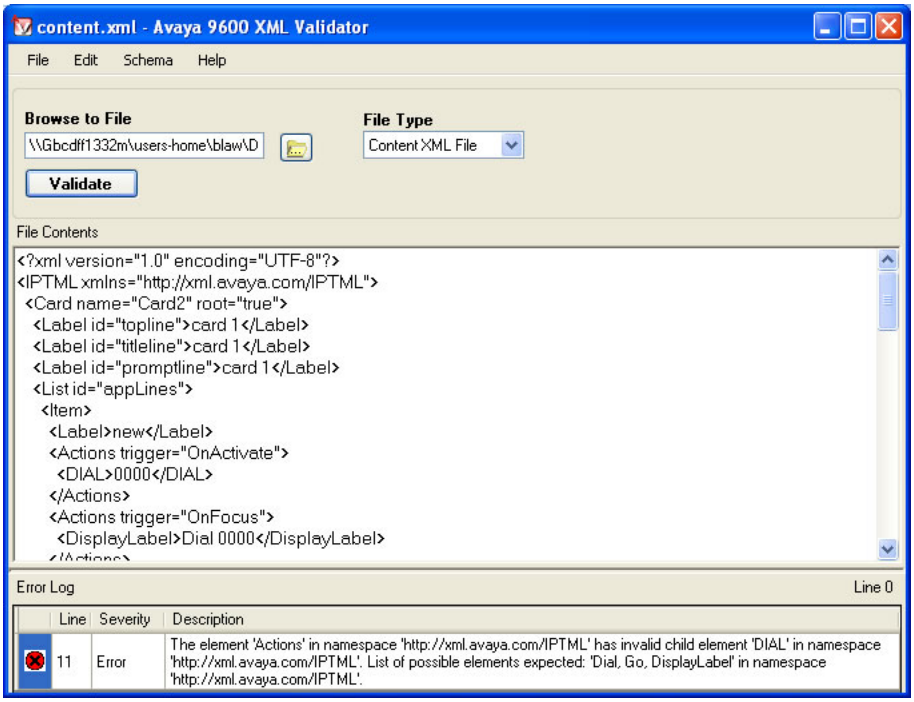

Figure 6-8: The Avaya one-X Deskphone and Skin XML Validator

Avaya one-X® Deskphone and Skin XML Templates

The Avaya one-X Deskphone XML and Skin XML templates comprise a number of valid and well-formed Deskphone and Skin XML files that can be adapted to create your own customization files. The Deskphone and Skin XML files used in **Figure 6-6** are provided as templates.

OTHER DEVCONNECT RESOURCES

DevConnect makes a wide variety of resources available to its members to help them through the IP deskphone application development process. These resources include:

- **Avaya IP Deskphones - Application Interface Capabilities Utility:** allows you to easily find out which Push types are supported on selected deskphone models and firmware releases. The utility also allows you to find out which deskphone models and firmware releases support selected Push types.
- **Sample applications:** including the AvayaTwitter Deskphone Application, Emergency Broadcast System, Rotating Ads Application, Screensaver Application and Webcam Application.
- **Tutorials:** including tutorials on creating browser-based and Push applications.
- **Forums:** allow members to raise and discuss issues relating to Avaya IP deskphone application development and use of the Deskphone XML Designer.
- **FAQs:** answers to frequently asked questions about Avaya IP deskphone application development.
- **On-demand presentations:** including a presentation on developing applications for Avaya IP deskphones.
- **On-line, Flash-based training course**

ROUND-UP

In this chapter we have looked at the tools Avaya provides to help you create Push applications and customize Avaya one-X Deskphone user interfaces. Remember, all of these tools are available to DevConnect members as no-cost downloads - see *Appendix C* for more information regarding DevConnect membership.

CHAPTER 7
APPLICATION SCENARIOS

IN THIS CHAPTER:

- Exploiting Avaya IP deskphone application capabilities
- Avaya IP deskphone application scenarios

EXPLOITING AVAYA IP DESKPHONE APPLICATION CAPABILITIES

In this chapter we will look at examples of how the Avaya IP deskphone application capabilities described previously in the book can be exploited in diverse situations and production environments. The examples cover a variety of industries, including education, hospitality and corporate, and various utilities, including conferences, emergency broadcasts and feedback surveys.

The high level descriptions of the applications are intended to give you a feel for what is possible and to act as springboard for your own ideas and solutions. If you want more detailed examples, with source code and comprehensive documentation, your next step should be to investigate the sample applications developed by Avaya, including those provided with the PushSDK (see *Chapter 6*) and those available for download from the DevConnect web portal (see *Appendix C*).

The following application scenarios are described in this chapter:

- Hotel Check-in and Guest Services
- Conferencing Services
- Podcast Record, Broadcast and Feedback Services
- Meeting/Seminar Room Scheduling
- Legal Advice Real-time Charging
- High Security Password Maintenance
- Webcam Door Security
- Social Network Integration
- Emergency Message Broadcasting

The following information is given for each of the scenarios:

- **Title**
- **Environment:** the industry and utility
- **Functionality:** what the application does and the user experience
- **Implementation:** how the Push and/or browser capabilities are exploited by the application

Remember, the scenarios described here represent a tiny sample of what is possible. The ways in which the capabilities of Avaya IP deskphones can be exploited by application developers are almost limitless.

Important: the scenarios in this chapter are intended to provide insight into the potential types of deskphone application that you can create; they are not specific offers from Avaya. Avaya Professional Services and many Avaya BusinessPartners can help you implement deskphone applications like these if you don't wish to undertake the development yourself.

AVAYA IP DESKPHONE APPLICATION SCENARIOS

HOTEL CHECK-IN AND GUEST SERVICES

Environment: Hospitality; Hotel guest services

Functionality: When a new guest checks in, the deskphone in their room is reset and a personalized welcome message displayed. Resetting involves deleting the call history and returning user-configurable options to their default values. During their stay guests can use the deskphone to request and receive audio wake-up calls, receive restaurant menus and pre-order meals, and access information about local events.

Implementation: When the guest checks-in, the application uses the Phonexml Push type to send a `ClearPhoneHistory` and `SetSettingRequest` to the deskphone in the guest's room. This could be done automatically, if the deskphone application functionality is integrated with the check-in system or be triggered by the receptionist via a dedicated user interface.

The deskphone displays a menu option that links to a WML page that allows the guest to request a wake-up call. The WML page includes a form in which the guest enters the required call time and wake-up method (for example, classic or rock music, spoken time check or alarm) and snooze properties. The guest submits the request to the application, which then sends a Receive Audio Push to the deskphone at the appropriate time.

The wake-up can be accompanied by a Display Push, causing the breakfast menu to be displayed in the deskphone's browser. The guest can select items from the menu and submit a request for them to be delivered to the room at a specified time.

Periodically, the application uses a Phonexml Push to send a `RefreshResourceRequest` to update the deskphone user interface with a menu of links to web-based information about local events. The administrator generates the `content.xml` file containing the new menu options, and the linked WML pages, via the application user interface. The `content.xml` file is Pushed to the deskphone and the WML pages hosted on a web server.

CONFERENCING SERVICES

Environment: Corporate Enterprise; Web or telephone conferencing

Functionality: A fixed time before the conference is due to start, invitees receive a reminder on their deskphones informing them that the conference is imminent.

At the start time, invitees receive information about the conference, such as the attendee list, scheduled duration and instructions for joining. The message also includes a click-to-dial option that invitees can select to join the conference.

During the conference, a countdown of time remaining and other conference information is displayed at the invitees' deskphones.

Implementation: The deskphone application integrates with the Avaya Meeting Exchange conference scheduler to access information about conference times, durations and invitees. Before the due start time the application obtains the list of invitees and performs a database query to look up the IP addresses of their deskphones.

The application uses a normal Display Push to send a WML file to invitees' deskphones, accompanied by one ring ping. The web page contains a reminder that the conference is due to start at the scheduled time.

At the start time, the application uses a barge Display Push to send another WML file to the invitees' deskphones, this time accompanied by three ring pings. The web page includes a WTAI click-to-dial function that autodials the appropriate number to join the conference.

When the deskphone joins the conference, the application uses a third Display Push to display a WML page containing information about other attendees and a countdown to the scheduled end times.

PODCAST SERVICES

Environment: Corporate Enterprise; communication, training

Functionality: The podcaster is prompted to present the podcast as the scheduled time and uses their deskphone to record the podcast.

The podcast is broadcast live or at a scheduled time to all employees in an organization.

The podcast can subsequently be played back by employees, on request.

When the podcast ends, the employee is presented with a feedback form that prompts them to rate and comment on the podcast.

Implementation: The application sends a Transmit Audio Push to the podcaster's deskphone at the scheduled time. The podcaster transmits the podcast from their deskphone and it is recorded and saved by the application.

To broadcast the podcast, the application uses a Multicast Receive Audio Push to play the podcast at the deskphones of all employees in the organization. The Receive Audio Push can either be dovetailed with the Transmit Audio Push for live broadcast, or be scheduled to broadcast the podcast recording at a future date.

The application uses a Phonexml Push to update the deskphone user interface with a menu option that allows them to request the recording of the podcast be played at their deskphone.

At the end of the podcast, the application uses a Display Push to send a WML-coded feedback form to the deskphone's browser. The employee enters their comments and submits the completed form to the application for analysis.

ROOM SCHEDULING

Environment: Corporate Enterprise, Education; meeting/tutorial room administration

Functionality: When the meeting organizer or tutor enters the room they can check their booking details on the deskphone display. If the meeting is in danger of overrunning, the organizer can see whether the room is booked for the next session and, if not, extend the booking. When looking for a room for an impromptu meeting, staff can check the display on the deskphone in a currently free room to see how long it is available for.

Implementation: The deskphone application integrates with the room booking system. At the end of each booked meeting, the application gets information about the next two bookings from the

booking system. The application renders the information in a WML file and uses a Display Push to send the information to the deskphone located in the meeting room. The displayed browser pages also include an option that links to the booking system and allows the organizer to extend the meeting if the room is free at the end of the scheduled meeting time.

LEGAL FEE SERVICES

Environment: Legal; charging for telephone consultations

Functionality: When a solicitor receives a telephone call from a client, the client's details are popped at the deskphone display screen. The solicitor can select an option on the deskphone display to begin charging the client for time spent on the call. At the end of the call, the charging timer is automatically stopped. The solicitor can manually set the charge rate, adjust the amount of time to be charged and select options to record other information about the call before submitting the details.

Implementation: The deskphone application integrates with the company's call recognition, client details and charging systems. When a call is made, the call recognition system automatically attempts to match the caller's number to a client record and, if found, retrieves their details. The deskphone application renders the client details in a WML file and uses a Display Push to display the details to the solicitor's deskphone. The pushed web pages also include an option to start a call charge meter and form fields to record additional information about the call. At the end of the call, the solicitor updates the call details and selects an option to submit the details to the system.

ADVANCED SYSTEM SECURITY

Environment: Government, Corporate Enterprise; system security

Functionality: This application adds an extra layer of authentication of users who request that their password be reset to allow them to access a secure system or network. The application requires that users be located at their usual workstation to be able to reset their password, thereby helping prevent hackers using identity theft to obtain passwords illicitly.

The user forgets or locks their password and submits a request from their PC to have it reset. A unique PIN is sent to the user's deskphone. The user enters the PIN number in the password reset form on their PC. The user receives the new password in an email.

Implementation: When the application receives the password reset request from the user it looks up the IP address of the user's deskphone and generates a unique PIN for the request. The application uses a Display Push to send the unique PIN to the user's deskphone browser. When

the user submits the PIN from their PC, the application checks that it is valid, generates the new password and sends it to the user in an email.

WEBCAM DOOR SECURITY

Environment: Domestic, Education, Corporate Enterprise; access security

Functionality: Whenever someone approaches a door in a secure area, the receptionist's or security guard's deskphone sounds an alert and an image of the vicinity is displayed in the deskphone's web browser. The image automatically updates every 10 seconds. The receptionist or security guard may use a separate intercom system to talk with the visitor or, if they recognize them, use an automatic door-release system to grant entry.

Implementation: Every 10 seconds the application captures a JPEG image from a webcam located near the door and uploads the image to a web server (Trusted Push Server). The Trusted Push Server also hosts a WML file that includes a WML page that references and displays the most recent captured image. A motion-detector triggers the application to use a Display Push to send the WML file to the deskphone browser. The Push Initiation request is sent in barge mode with three ring ping alerts. The WML page automatically refreshes every 10 seconds while displayed in the browser, to be updated with the latest captured image.

Note: A more advanced and fully documented version of this example is available on the Avaya DevConnect web portal: see the *Webcam sample application [9].*

SOCIAL NETWORK INTEGRATION

Environment: Corporate Enterprise; social networking

Functionality: Many of the Social Networking sites, such as Facebook, Twitter and Yammer, provide public APIs that third-party developers can use to include functionality provided by those sites in their own applications. In this example, the user is able to request that recent messages ("tweets") posted on Twitter, by friends and colleagues who they are following, are displayed at their deskphone. The user must previously have set up an account on the Twitter web site and configured it to receive tweets from other Twitter users.

The deskphone user selects a "View Tweets" option on the deskphone user interface: a WML card is displayed in the deskphone browser, incorporating a form in which the user enters and submits their Twitter account credentials. The three most recent tweets posted by the user's contacts are displayed in the deskphone's browser.

Implementation: A Phonexml Push is used to add the "View Tweets" menu option to the deskphone user interface of users in the organization who are permitted to use this service. The user selects the option and submits their Twitter account credentials to the application. The application uses the credentials to log into the user's Twitter account, and then uses the Twitter API method `friends_timeline` to retrieve the three most recent tweets posted by the user's Twitter contacts. The application generates a WML file containing the messages that is displayed in the deskphone's browser.

Note: A more advanced and fully documented version of this example is available on the Avaya DevConnect web portal: see the sample application *Integrating Twitter as an Avaya IP Telephone Application [6].*

EMERGENCY MESSAGE BROADCASTING SERVICE

Environment: Education, Corporate Enterprise; emergency broadcast

Functionality: In the event of an emergency requiring evacuation from all or part of a building or complex, an emergency message is broadcast to all deskphones in the affected locations.

Implementation: The administrator uses the application's user interface to select previously created audio, text and WML messages from a menu, and the locations to which the messages are to be sent. The application performs a database query to obtain the IP addresses of the deskphones in the selected locations, and uses a Top Line Push, Display Push and Receive Audio Push to send the messages to the deskphones. The Push Initiation requests are all sent in barge mode with three ring ping alerts. Using three separate pushes increases the likelihood of all deskphones in the locations being able to accept at least one Push Type, regardless of model or installed firmware.

ROUND-UP

The brief examples given in this chapter have illustrated how the characteristics of deskphones and their application capabilities can be exploited to provide solutions in a wide variety of environments and scenarios. Hopefully, this has whetted you appetite to explore further and stimulated your imagination to come up with your own applications and solutions.

This is the final chapter of the book. To learn more, your next step should be to visit the Avaya DevConnect web portal and register as DevConnect member: see *Appendix C* for information about the educational and technical resources that DevConnect offers.

APPENDIXES

The following information is provided in the appendixes:

- Appendix A: Glossary of Terms
- Appendix B: Pushable and Non-pushable States
- Appendix C: The Avaya DevConnect Developer and Partner Program

APPENDIX A
GLOSSARY OF TERMS

The following abbreviations and technical terms are used in this book:

46xxsettings.txt — See **System-wide Settings File**.

Application Area — The usable display area between the Prompt Line and Softkey labels.

Application Line — The display area line that indicates application-specific messages.

Card (content) — Customized user interface content is displayed across a series of pages, known as cards. Each card is assigned a unique name to identify it. One of the cards must be defined as the "root": this is the card that acts as the user's home page. For each card, the name of the previous and next card in a sequence can be defined.

Card (WML) — A WML card is similar to an HTML page, but WML delivers a set (deck) of closely related cards in a single file. The complete WML file comprises a deck of cards, of which only one is visible in the browser at one time. As each of the cards is labeled by a name and ID, they can be linked together without difficulty. The WML card author determines the content of the card. The browser determines how this card is displayed (rendered).

Deck — A deck is a stack of cards, or pages, defined in a single WML or Deskphone XML file. Only one card in the deck is visible in the deskphone display at any one time.

Deskphone XML — An XML language designed specifically for defining Avaya deskphone custom user interface content.

Focus — An application line is given focus using the deskphone's **Up** and **Down** navigation buttons. When an application line has focus the text is highlighted, the associated help message displayed on the prompt line and the associated action is performed when the user presses the deskphone's **OK** button. When defining a customized skin, it is necessary to define the appearance of application lines with and without focus to achieve the highlight effect.

H.323 — H.323 is a set of protocols that defines audio-visual communication sessions on packet networks. The H.323 standard addresses call signaling and control, multimedia transport and control, and bandwidth control for point-to-point and multi-point conferences. Along with SIP, H.323 is one of the two protocols supported by Avaya IP deskphones.

HTML — *Hyper Text Markup Language* is a text-based way of describing data for transmission over the Internet HTML is usually used with larger, color displays.

Interrupt Screen — A screen that automatically accompanies a standalone Audio Receive or Audio Transmit Push. Interrupt screens provide the user with specific information about terminating the audio push.

IP Address — *Internet Protocol Address* – an address in the format *nnn.nnn.nnn.nnn* that uniquely identifies any device on the network, such as a web server or IP deskphone.

IPCoDE — The Avaya IP Communications Development Environment comprises software-only, developer oriented editions of Communication Manager, SIP Enablement Services and Application Enablement Services.

IP Deskphone — For the purposes of this book, and IP deskphone is defined as an Avaya one-X® (9600-series), 5600-series, 4600-series or 1600-series deskphone with H.323 or SIP firmware installed.

JPEG — Image format supported by most Avaya IP deskphone browsers. The JPEG format is ideal for displaying full-color or grey scale images, such as photographic images.

Multicast — A technique developed to send packets from one location in the Internet to many other locations without any unnecessary packet duplication. In multicasting, one packet is sent from a source and is replicated as needed in the network to reach as many end users as necessary.

Phonexml — Push type that allows an application to change a deskphone's user interface content and skin, clear its call history and web logs, and set or clear user defined settings such as preferred language and ringtone.

Priority — Determines whether the priority the deskphone must give to a request; either "normal" or, for important messages, "barge". The priority is defined by the value of the mode attribute in the Push Initiation message.

Prompt Line — The third line of a deskphone screen's top display area. The Prompt Line is used to display a context-specific help message for the currently selected application line.

Push Content — A valid XML or a WML file that contains a **<Response>** or **<Wml>** tag as the root. The file carries the actual information to be displayed or streamed on to a deskphone.

Push Initiation Message — An XML message that contains a **<Push>** tag as the root. The Push Initiation message uses a **<go>** tag to specify a URI from which a deskphone can request Push Content.

Push Initiation Server — The server that hosts the Push Initiator and from which Push Initiation messages are sent to the deskphone.

Push Initiator — The client application that creates and sends Push Initiation messages to deskphones.

PushSDK	A software development kit that includes a Java Application Programming Interface (API) that can be used by an application to create and send Push Initiation messages to Avaya IP deskphones and, in some cases, create and upload Push Content. The kit also includes sample applications, javadoc and user documentation, and Push Initiation and Trusted Push Server elements.
Push API	XML-based interface exposed and implemented by Avaya IP deskphones. The interface allows applications to spontaneously push messages to Avaya IP deskphones without involving the user.
Push State	Indicates whether the deskphone is involved in an activity, such as restoring a back-up file, that prevents it accepting a Push request. Possible states are pushable or non-pushable. See *Appendix B: Pushable and Non-pushable States* for details.
RTP Audio	An audio stream received from an application outside the context of a telephone call.
Settings File	See **System-wide Settings File.**
SIP	*Session initiation Protocol* is a standards-based signaling protocol, widely used for setting up and tearing down multimedia communication sessions over Internet Protocol (IP). Along with H.323, SIP is one of the two protocols supported by Avaya IP deskphones.
Softkeys	Most deskphone models have a row of softkeys located immediately below the display screen. Applications can program the labels displayed against the softkeys and their associated functionality.
Subscription Server	A server or a database that stores information about Push-enabled deskphones, such as their IP addresses, extensions, deskphone models, etc.
System-wide Settings	A text file (46xxsettings.txt) that defines the options and settings for all Avaya IP deskphones in an enterprise. The file is hosted on a web server and downloaded to each deskphone whenever it is registered or reset. The file includes parameters that affect deskphone application development, including the PUSHCAP parameter that defines which types of Push are supported within the enterprise.
Title Line	The second line of a deskphone screen's top display area. The title line can be used to show the title and sub-title of the current page or a custom message. **Previous** and **Next** page indicators are also displayed on the Title Line, if applicable.
Top Line	The first line of a deskphone screen's top display area. The Top Line usually displays the telephone extension number and status information, but can display a custom message. The Top Line does not form part of the deskphone browser. It is therefore possible to process and display simultaneous Top Line and Display Pushes.

Trusted Push Server	A server from which Push Content can be downloaded by the deskphone. The server's IP address must be listed against the TPSLIST parameter in the **System-wide Settings File**; otherwise, the Push request will be rejected.
Trusted Receive Server	The server to which the RTP audio generated by a Transmit Audio Push request is sent by the deskphone. The server's IP address must be listed against the TPSLIST parameter in the system-wide Settings file; otherwise, the Push request will be rejected.
WBMP	*Wireless Bitmap* is an image format support by most Avaya IP deskphone browsers. WBMP was originally designed specifically for displaying images on mobile devices. The images are monochrome, comprising only black and white pixels, thereby minimizing the file size.
WML	*Wireless Markup Language* is a subset of XML, used by the Avaya IP deskphone web browser to communicate with WML Servers.
WMLScript	A scripting language specifically designed for programming mobile devices. It is based on ECMAScript, but has been optimized for low bandwidth communication and limited processing power and memory.
WTAI	*Wireless Telephony Applications Interface* is a set of interfaces that extend the Wireless Application Environment to include telephony applications. Most Avaya IP deskphone web browsers support two WTAI functions: click-to-dial and add-to-phonebook.
x-Push-Status	A HTTP extension header designed specifically for the Push process. The extension is used by the deskphone to send the Push Initiation message status (either success or reason for failure) to the Push Initiator.
XML	*eXtensible Markup Language.* W3C's standard for Internet Markup Languages. WML, Deskphone XML and Skin XML are all examples of these languages.

APPENDIX B
PUSHABLE AND NON-PUSHABLE STATES

The table below shows which deskphone states and activities will cause a Push Initiation message to be rejected for each Push type and priority.

	PUSH TYPE AND PRIORITY									
State of phone when push received	Top Line		Browser Display		Audio Receive		Audio Transmit		Subscribe	Phonexml
	Normal	Barge	Normal	Barge	Normal	Barge	Normal	Barge	n/a	Barge
Restoring back-up file	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
Local procedure running	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Broadcasting Transmit Audio	✗	✗	✗	•	✗	✓	✗	✗	✓	✓
In text-entry mode (browser)	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓
In text entry mode (UI)	✗	✓	•	✓	✓	✓	✓	✓	✓	✓
Call message on top line	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓
System message on top line	•	✓	✗	✓	✓	✓	✓	✓	✓	✓
Call incoming or active	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓

Legend:

✓ The phone is in a pushable state and the Push can be accepted.

• The telephone is in a pushable state, but the Push Content is loaded in the background and not displayed until current activity ends.

✗ The telephone is in a non-pushable state. The Push **cannot** be accepted.

APPENDIX C
THE AVAYA DEVCONNECT DEVELOPER AND PARTNER PROGRAM

DEVCONNECT DEVELOPER & PARTNER PROGRAM

As a communications application developer, you've already made an investment in the future of intelligent communications solutions for your business. We'd like to help you get more out of that investment, by providing you with the tools and resources to take your business to the next level.

Whether you are simply exploring how Avaya technology can create opportunities for your company and customers, or seeking additional technical know-how and information to aid in your current development efforts, a free DevConnect membership offers:

- Downloadable software development kits (SDKs) and client-side libraries aid in application development and integration.
- Step-by-step tutorials, in-depth Flash-based training courses, and on-demand webinars build skills within your technical community.
- Technical Frequently Asked Questions (FAQs) and community-supported forums assist in knowledge sharing.
- Sample Applications jump start your application development efforts by demonstrating how to use key APIs.
- Developer conferences, podcasts, newsletters and other developer communications provide ongoing technical awareness.
- Simulators* and remote labs aid in prototyping and proof-of-concept implementations with minimal up-front investments.
- Evaluation kits and developer editions for emerging Avaya products, allow you to explore new solutions that can give you a competitive advantage or help you deliver superior customer service.
- Interoperability Notes and DevConnect Member Application Notes created by our Solution and Interoperability Test Lab assist in deploying solutions.
- Opportunities to participate in technical beta programs give you a leg up on next generation application development activities.
- And much more…

* Additional media & support charges may apply

A SAMPLE LISTING OF AVAILABLE DEVCONNECT RESOURCES:

SDK & API DOWNLOADS AND DOCUMENTATION

Information on over 30 product interfaces, APIs and SDKs, including:

- AE Services, including: Telephony & System Management Web Services; DMCC SDKs for Java, .NET and XML; and JTAPI/TSAPI
- IP Telephone Push API and PushSDK
- SIP Enablement Services Personal Profile Manager web service
- Proactive Contact Agent & Event Services APIs
- Meeting Exchange Bridge Control API
- Interaction Center 7.1 Client SDK

SIMULATORS & REMOTE LABS

- Avaya IP Communication Development Environment (IPCoDE), featuring AE Services, Communication Manager & SIP Enablement Services
- Self Service Remote Lab, featuring Avaya Voice Portal
- AE Services & Communication Manager Remote Lab

TRAINING COURSES, TUTORIALS & TECHNICAL WEBINARS

- Over 25 hours of in-depth training courses on Avaya product interfaces, including Avaya Distributed Office, SIP, Interaction Center, Dialog Designer and Avaya IP Telephone APIs
- Step-by-step tutorials, including:
  - ◊ Setup, Application Initialization, and Event Monitoring using the AE Services DMCC Java SDK
  - ◊ Designing a Microsoft SQL database connector in Dialog Designer
  - ◊ Avaya IP Telephones Push and Browser Applications Setup
  - ◊ Over 15 on-demand Technical Webinars, including An Introduction to the Voice Portal and Dialog Designer

TOOLS, EVALUATION KITS & DEVELOPER EDITIONS

- Avaya Dialog Designer
- Avaya AE Services DMCC Dashboard
- Avaya one-X Deskphone XML Designer

SAMPLE APPLICATIONS

More than 25 sample applications (in addition to those provided with individual SDKs) for IP telephone applications, AE Services, Dialog Designer, Meeting Exchange, Proactive Contact, and SIP Enablement Services (SES).

REGISTER TODAY

Register today for your free DevConnect membership at **www.avaya.com/devconnect**

AVAYA IP COMMUNICATIONS DEVELOPMENT ENVIRONMENT

The Avaya IP Communications Development Environment (Avaya IPCoDE) enables developers to test and debug IP communications applications under development, cost-efficiently and easily. Avaya IPCoDE comprises software-only, developer-oriented editions of Avaya Application Enablement (AE) Services, Avaya Communication Manager and Avaya SIP Enablement Services. The complete environment can be installed and run on a single machine.

BENEFITS OF AVAYA IPCODE

The Avaya IP Communications Development Environment allows DevConnect members to:

- Reduce their up-front investment in Avaya products for debugging and unit testing IP communications applications under development.
- Speed up application development by providing an easily accessible debugging and unit testing environment on the developer's desktop.
- Validate IP communications applications in preparation for DevConnect compliance testing in the Avaya Solution and Interoperability Test Lab.
- Run sample applications, included in the appropriate SDKs and available for download from the DevConnect web portal, to gain an appreciation of the capabilities of Avaya solutions.
- Use tools such as the AE Services DMCC Dashboard, TSAPI Exerciser and JTAPI Exerciser to expedite and learn about the capabilities of the corresponding services, and aid application development. These tools are included in the appropriate SDKs.
- Set up an environment for other Avaya products with a dependency on Avaya Communication Manager, such as Avaya Voice Portal or the Avaya one-X Deskphone Emulator.

DEBUGGING AND TESTING APPLICATIONS

Avaya IPCoDE is ideally suited for debugging and testing applications under development, including:

- AE Services applications developed against the AE Services APIs, protocol descriptions and web services that enable access to the capabilities of Communication Manager. These include applications that use the Device, Media, and Call Control (DMCC) service, TSAPI service, JTAPI service, Telephony Web Service and System Management Service (SMS) Web Service.
- Avaya Communication Manager operations, administration, maintenance and provisioning applications.
- Avaya SIP-enabled applications, including:
  - ◊ Applications that expose the capabilities of Communication Manager to Avaya SIP telephones.
  - ◊ Applications that incorporate SIP-based Presence and Instant Messaging with Avaya IP Softphone, Avaya IP Agent, Avaya one-X® Desktop, and Avaya one-X® Deskphones.
- Avaya SIP Personal Profile Manager (PPM)-based applications, utilizing web services exposed by Avaya SIP Enablement Services for user-specific information.
- Avaya IP telephone applications that leverage the capabilities exposed by the Avaya IP telephone APIs, such as the Push API.

In addition, the environment can be used to debug and test applications that support various types of phones, including SIP and H.323 IP telephones.

- **Note:** While Avaya IPCoDE provides fully-featured editions of the software, certain functional limitations exist, including some media processing capabilities. See the DevConnect web portal for additional details.

OBTAINING THE AVAYA IP COMMUNICATIONS DEVELOPMENT ENVIRONMENT

Information on how to request your copy of Avaya IPCoDE, including details of the release-cycle, licensing and pricing, is available on the DevConnect web portal (registration required). Procurement discounts are available to DevConnect Gold and Platinum members.

TECHNICAL SUMMARY

IP Phone Connectivity

The Avaya IP Communications Development Environment provides connectivity for up to 10 IP desktop telephones or soft phones.

Network Options

To provide flexibility of use, Avaya IPCoDE can be configured so that it can only be accessed from the host machine, or bridged to a private or public network.

Installation Requirements

To install and use Avaya IPCoDE, developers need:

- A desktop PC or server running Microsoft Windows 2000, 2003 or XP operating system. See the DevConnect web portal for details of the minimum requirements for the machine processor, RAM and hard disk space.
- A VMware environment installed on the PC or server.
- A DVD drive from which to install the Avaya IPCoDE environment.
- Adobe Acrobat Reader.

A dual monitor system is recommended as an effective method for monitoring the many simultaneous points of interaction and information exchange between the Avaya IPCoDE elements and the applications under development.

ADDITIONAL RESOURCES FOR DEVELOPERS

DevConnect members who require access to media gateway and other IP Communication features not supported by Avaya IPCoDE can also use the free AE Services & Communication Manager Remote Lab in support of development, testing and pre-compliance activities.

REFERENCES

All documentation and resources referenced in this book are available for download from the Avaya DevConnect web site (http://www.avaya.com/devconnect). The following documents are referenced:

1. **Avaya IP Deskphones - Application Interface Capabilities Utility:** allows users to determine which Push types are supported on selected deskphone models and firmware releases. The utility also allows users to determine which deskphone models and firmware releases support selected Push types.

2. **Avaya one-X® Deskphone Edition for 9600 Series IP Telephones Application Programmer Interface (API) Guide:** describes the set up and use of the Push and Web browser interfaces to Avaya 9600 Series H.323 and SIP deskphones.

3. **Avaya one-X® Deskphone Edition for 9600 Series SIP Telephones Developer Guide:** describes deskphone user interface customization using Deskphone XML and Skin XML files. The guide includes instructions for installing and using the Deskphone and Skin XML Validator.

4. **Avaya one-X® Deskphone Edition for 9600 [4600, 1600] Series IP Telephones Administrator Guide:** describes how to set up and administer an Avaya IP telephone system.

5. **PushSDK:** includes PushSDK API Javadoc, plus PushSDK installation and developer guides.

6. **Integrating Twitter as an Avaya IP Telephone Application:** documents the DevConnect sample application that enables users to access and view messages posted on Twitter (twitter.com) using Avaya 4600 and 9600 Series IP deskphones.

7. **Avaya one-X® Deskphone XML Designer and Emulator Installation and Configuration Guide:** describes how to set up a development environment for creating and testing user interface content customization files.

8. **Avaya one-X® Deskphone XML Designer and Emulator User Guide:** describes how to create and test user interface content customization files using the Designer and Emulator.

9. **Webcam sample application** demonstrates how the Avaya IP telephone web browser can be used to display near real-time images captured from a PC based web camera.

Note: In addition, SIP and H.323 firmware releases, and product documentation for all Avaya IP deskphones, are available for download from the Avaya Support site (http://support.avaya.com).

MASTERING AVAYA ONE-X™ DESKPHONE APPLICATION DEVELOPMENT

PUBLICATION OF THIS BOOK WAS SPONSORED BY AVAYA DEVCONNECT

About Avaya

Avaya is a global leader in enterprise communications systems. The company provides unified communications, contact centers, and related services directly and through its channel partners to leading businesses and organizations around the world. Enterprises of all sizes depend on Avaya for state-of-the-art communications that improve efficiency, collaboration, customer service and competitiveness. For more information please visit www.avaya.com.

avaya.com