Cisco Unified IP Conference Phone 8831 for Third-Party Call Control Provisioning Guide, Release 9.3(3)

September 9, 2014

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide.
Addresses, phone numbers, and fax numbers
are listed on the Cisco website at
www.cisco.com/go/offices.

CONTENTS

About This Document

This guide describes the provisioning of the Cisco Unified IP Conference Phone 8831 for Third-Party Call Control:

Purpose

The Cisco Unified IP Conference Phone 8831 for Third-Party Call Control can be remotely provisioned or preprovisioned by using the information in this document.

Document Audience

This document is written for service providers who offer services by using the Cisco VoIP products and specifically for administrative staff responsible for remote provisioning and preprovisioning Cisco Telephony devices.

Organization

This document is divided into the following chapters and appendices.

Chapter	Contents
Chapter 1, "Deployment and Provisioning"	Introduces Cisco VoIP products.
Chapter 2, "Creating Provisioning Scripts for Configuration Profile"	Describes how to work with Cisco provisioning scripts and configuration profiles.

Chapter	Contents
Chapter 3, "In-House Preprovisioning and Provisioning Servers"	Describes the features and functionality available when preprovisioning Cisco IP Telephony devices in-house and provisioning them remotely.
Chapter 4, "Provisioning Examples"	Step-by-step procedures for using the scripting language to create a configuration profile.
Chapter 5, "Provisioning Parameters"	A systematic reference for each parameter on the Provisioning tab of the administration web server.
Appendix A, "Sample Configuration Profiles"	A sample profile that you might find helpful.
Appendix B, "Acronyms"	The expansions for the acronyms used in this document.
Appendix C, "Related Documentation"	Links to resources for information and support.

Document Conventions

The following typographic conventions are used in this document.

Typographic Element	Meaning
Boldface	An option on a menu or a literal value to be entered in a field.
<parameter>	Angle brackets (<>) identify parameters that appear on the configuration pages of the administration web server.
Italic	A variable that should be replaced with a literal value.
Monospaced Font	A code sample or system output.

Deployment and Provisioning

Cisco IP Telephony devices are intended for high-volume deployments by VoIP service providers to residential and small business customers. In business or enterprise environments, IP Telephony devices can serve as terminal nodes. These devices are widely distributed across the Internet, connected through routers and firewalls at the customer premises.

The IP Telephony device can be used as a remote extension of the service provider back-end equipment. Remote management and configuration ensures the proper operation of the IP Telephony device at the customer premises.

This customized, ongoing configuration is supported by the following features:

- Reliable remote control of the endpoint
- Encryption of the communication controlling the endpoint
- Streamlined endpoint account binding

This chapter describes the features and functionality available when provisioning the Cisco Small Business IP Telephony devices and explains the setup required:

- Deployment, page 1-1
- Provisioning Overview, page 1-3

Deployment

Cisco IP Telephony devices provide convenient mechanisms for provisioning, based on two deployment models:

- Bulk distribution—The service provider acquires IP Telephony devices in bulk quantity and either preprovisions them in-house or purchases RC units from Cisco. The devices are then issued to the customers as part of a VoIP service contract.
- Retail distribution—The customer purchases the IP Telephony device from a retail outlet and requests VoIP service from the service provider. The service provider must then support the secure remote configuration of the device.

Bulk Distribution

In this model, the service provider issues IP Telephony devices to its customers as part of a VoIP service contract. The devices are either RC units or preprovisioned in-house.

RC units are preprovisioned by Cisco to resynchronize with a Cisco server that downloads the device profile and firmware updates.

A service provider can preprovision IP Telephony devices with the desired parameters, including the parameters that control resynchronization, through various methods: in-house by using DHCP and TFTP; remotely by using TFTP, HTTP, or HTTPS; or a combination of in-house and remote provisioning.

RC Unit Deployment

RC units eliminate in-house preprovisioning of IP Telephony devices and reduce the need for the service provider to physically handle the devices prior to shipping them to end customers. This approach also discourages the use of IP Telephony devices with an inappropriate service provider.

A RC unit is preprovisioned by Cisco with the connection information for the provisioning servers. These servers are maintained by Cisco Systems, Inc. for the service provider that purchased the units. The MAC address of each RC unit is associated with a customizable profile on the Cisco provisioning servers. When the RC unit is connected to the broadband link, it contacts the Cisco provisioning server and downloads its customized profile.

The service provider works with a Cisco sales engineer to develop a simple provisioning profile. The profile contains minimal information that redirects the device to the service provider provisioning server. This profile is placed on the Cisco RC server by the Cisco Voice Team.

RC Unit Status

The status of an RC unit can be determined by viewing the Info > Product Information page, Customization section, on the administration web server. An RC unit that has not been provisioned displays **Pending**. An RC unit that has been provisioned displays the name of the company that owns the unit. If the unit is not an RC unit, the page displays **Open**.

Below is a sample template for an RC unit to be preprovisioned by Cisco with the connection information:

```
Restricted Access Domains "domain.com, domain1.com, domain2.com";
Primary_DNS               * "x.y.w.z";
Secondary_DNS             * "a.b.c.d";
Provision_Enable          * "Yes";
Resync_Periodic           * "30";
Resync_Error_Retry_Delay  * "30";
Profile_Rule  * "http://prov.domain.com/sipura/profile?id=$MA";
```

The `Restricted Access Domains` parameter is configured with the actual domain names of up to a maximum of five domains. The Primary_DNS and Secondary_DNS parameters are configured with the actual domain names or IP addresses of the DNS servers available to the RC unit.

Retail Distribution

In a retail distribution model, a customer purchases a Cisco IP Telephony device and subscribes to a particular service. The Internet Telephony Service Provider (ITSP) sets up and maintains a provisioning server, and preprovisions the phone to resynchronize with the service provider server. See the "In-House Device Preprovisioning" section on page 3-2 for more information.

The customer signs on to the service and establishes a VoIP account, possibly through an online portal, and binds the device to the assigned service account. When the device is powered up or a specified time elapses, the IP Telephony device resynchronizes, downloading the latest parameters. These parameters can address goals such as setting up a hunt group, setting speed dial numbers, and limiting the features that a user can modify.

Resynchronization Process

The firmware for each IP Telephony device includes an administration web server that accepts new configuration parameter values. The IP Telephony device is instructed to resync with a specified provisioning server through a resync URL command in the device profile. The URL command typically includes an account PIN number or alphanumeric code to associate the device with the new account. For example:

```
http://192.168.1.102/admin/resync?https://prov.supervoip.com/cisco-init/1234abcd
```

In this example, a device at the DHCP-assigned IP address 192.168.1.102 is instructed to provision itself to the SuperVoIP service at `prov.supervoip.com`. The PIN number for the new account is 1234abcd. The remote provisioning server is configured to associate the IP Telephony device that is performing the resync request with the new account, based on the URL and PIN.

Through this initial resync operation, the IP Telephony device is configured in a single step, and is automatically directed to resync thereafter to a permanent URL on the server.

For both initial and permanent access, the provisioning server relies on the client certificate for authentication and supplies configuration parameter values based on the associated service account.

Provisioning Overview

An IP Telephony device can be configured to resynchronize its internal configuration state to match a remote profile periodically and on power up by contacting a normal provisioning server (NPS) or an access control server (ACS).

By default, a profile resync is only attempted when the IP Telephony device is idle, because the upgrade might trigger a software reboot interrupting a call. If intermediate upgrades are required to reach a current upgrade state from an older release, the upgrade logic is capable of automating multi-stage upgrades.

NPS

The NPS can be a TFTP, HTTP, or HTTPS server. A remote firmware upgrade is achieved by using TFTP or HTTP, but not by using HTTPS because the firmware does not contain sensitive information.

Communication with the NPS does not require the use of a secure protocol because the updated profile can be encrypted by a shared secret key. Secure first-time provisioning is provided through a mechanism that uses SSL functionality. An unprovisioned IP Telephony device can receive a 256-bit symmetric key encrypted profile specifically targeted for that device.

Provisioning States

The provisioning process involves these provisioning states.

State	Description
MFG-RESET **Manufacturing Reset**	The device returns to a fully unprovisioned state; all configurable parameters regain their default values. On phones that do not support IVR, perform the factory reset through LCD **Setup** menu. Allowing the end user to perform a manufacturing reset guarantees that the device can always be returned to an accessible state.
SP-CUST **Service Provider Customization**	The Profile_Rule parameter points to a device-specific configuration profile by using a provisioning server that is specific to the service provider. The methods for initiating resynchronization are: • Auto-configuration by using a local DHCP server. A TFTP server name or IPv4 address is specified by DHCP. The TFTP server includes the Profile_Rule parameter in the configuration file. • Entering a resync URL. The URL starts a web browser and requests a resync to a specific TFTP server by entering the URL syntax: http://*x.x.x.x*/admin/resync?*prvserv*/*device.cfg*, where: *x.x.x.x* is the IP address of the IP Telephony device, *prvserv* is the target TFTP server, and *device.cfg* is the name of the configuration file on the server. • Editing the Profile_Rule parameter by opening the provisioning pane on the web interface and entering the TFTP URL in the Profile_Rule parameter. For example, *prserv*/spa962.cfg. • Modifying the configuration file Profile_Rule and to contact a specific TFTP server and request a configuration file identified by the MAC-address. For example, this entry contacts a provisioning server, requesting a profile unique to the device with a MAC address identified by the `$MA` parameter: `Profile_Rule tftp.callme.com/profile/$MA/spa962.cfg;`

State	Description

SEC-PRV-1 Secure Provisioning—Initial Configuration	An initial, device-unique CFG file is targeted to a IP Telephony device by compiling the CFG file with the SPC --target option. This provides an encryption that does not require the exchange of keys.
	
	The initial, device-unique CFG file reconfigures the device profile to enable stronger encryption by programming a 256-bit encryption key and pointing to a randomly-generated TFTP directory. For example, the CFG file might contain:
	
	`Profile_Rule [--key $A] tftp.callme.com/profile/$B/spa962.cfg;`
	`GPP_A 8e4ca259…;  # 256 bit key`
	`GPP_B Gp3sqLn…;  # random CFG file path directory`

SEC-PRV-2 Secure Provisioning—Full Configuration	Profile resync operations subsequent to the initial SEC-PRV-1 provisioning retrieve the 256-bit encrypted CFG files that maintain the IP Telephony device in a state synchronized to the provisioning server. The profile parameters are reconfigured and maintained through this strongly encrypted profile. The encryption key and random directory location in the SEC-PRV-2 configuration can be changed periodically for extra security.

Configuration Access Control

The IP Telephony device firmware provides mechanisms for restricting end-user access to some parameters. The firmware provides specific privileges for login to an **Admin** account or a **User** account. Each can be independently password protected.:

- Admin Account—Allows the service provider full access to all administration web server parameters.
- User Account—Allows the user to configure a subset of the administration web server parameters.

The service provider can restrict the user account in the provisioning profile in the following ways:

- Indicate which configuration parameters are available to the User account when creating the configuration. (Described in the "Element Tags" section on page 2-2.)
- Disable user access to the administration web server.
- Disable user access for LCD GUI. (Described in the "Access Control for LCD GUI" section on page 2-4.)
- Disable the factory reset control by using the IVR.
- Restrict the Internet domains accessed by the device for resync, upgrades, or SIP registration for Line 1.

Communication Encryption

The configuration parameters communicated to the device can contain authorization codes or other information that protect the system from unauthorized access. It is in the service provider's interest to prevent unauthorized activity by the customer, and it is in the customer's interest to prevent the unauthorized use of the account. The service provider can encrypt the configuration profile communication between the provisioning server and the device, in addition to restricting access to the administration web server.

Creating Provisioning Scripts for Configuration Profile

The Cisco Unified IP Conference Phone 8831 for Third-Party Call Control accepts a profile format, based on an open, published syntax. The Open format uses a simple XML-like syntax.

The examples in this document uses configuration profiles with Open format (XML-style) syntax. Sample profiles can be found in Appendix A, "Sample Configuration Profiles."

This chapter describes the provisioning script in the following sections:

- Configuration Profile Formats, page 2-1
- Open Profile (XML-style) Compression and Encryption, page 2-5
- Using Provisioning Parameters, page 2-11
- Data Types, page 2-16

For detailed information about your IP Telephony device, refer to the administration guide for that device. Each guide describes the parameters that can be configured through the administration web server.

Configuration Profile Formats

The configuration profile defines the parameter values for the IP Telephony device.

The configuration profile format for the IP Telephony device, Cisco Unified IP Conference Phone 8831 for Third-Party Call Control is described below:

- Open format uses standard XML authoring tools to compile the parameters and values. To protect confidential information in the configuration profile, this type of file is typically delivered from the provisioning server to the IP Telephony device over a secure channel provided by HTTPS. See the "Open Profile (XML-style) Compression and Encryption" section on page 2-5.

Note Only UTF-8 charset is supported. If you modify the profile in an editor, do not change the encoding format; otherwise, the IP Telephony device cannot recognize the file.

Each model of IP Telephony device has a different feature set and therefore a different set of parameters.

Open Format (XML-style) Profile

The Open format profile is a text file with XML-like syntax in a hierarchy of elements, with element attributes and values. This format lets you use standard tools to create the configuration file. A configuration file in this format can be sent from the provisioning server to the IP Telephony device during a resync operation without compiling the file as a binary object.

The IP Telephony device can accept configuration formats that are generated by standard tools. This feature eases the development of back-end provisioning server software to generate configuration profiles from existing databases.

To protect confidential information contained in the configuration profile, this file is generally delivered from the provisioning server to the IP Telephony device over a secure channel provided by HTTPS. Optionally, the file can be compressed by using the gzip deflate algorithm (RFC1951). In addition, the file can be encrypted by using 256-bit AES symmetric key encryption.

Example: Open Profile Format

```
<device> <flat-profile>
<Resync_On_Reset> Yes
  </Resync_On_Reset>
<Resync_Periodic> 7200
  </Resync_Periodic>
<Profile_Rule>
    tftp://prov.telco.com:6900/cisco/config/spa504.cfg
  </Profile_Rule>
</flat-profile> </device>
```

The <flat-profile> element tag encloses all parameter elements to be recognized by the IP Telephony device.

Note IP Telephony devices with firmware versions before 2.0.6 do not support Open format profiles.

Element Tags, Attributes, Parameters, and Formatting

A file can include element tags, attributes, parameters, and formatting features.

Element Tags

The properties of element tags are:

- The IP Telephony device recognizes elements with proper parameter names, when encapsulated in the special `<flat-profile>` element.
- The `<flat-profile>` element can be encapsulated within other arbitrary elements.
- Element names are enclosed in angle brackets.
- Most of the element names are similar to the field names in the administration web pages for the device, with the following modifications:
  - Element names may not include spaces or special characters. To derive the element name from the administration web field name, substitute an underscore for every space or the special characters `[`, `]`, `(`, `)`, or `/`.

    For example, the Resync On Reset field is represented by the element <Resync_On_Reset>.

  – Each element name must be unique. In the administration web pages, the same fields might appear on multiple web pages, such as the Line, User, and Extension pages. Append `[n]` to the element name to indicate the number that is shown in the page tab.

    For example, the Dial Plan for Line 1 is represented by the element <Dial_Plan[1]>

- Each opening element tag must be matched by a corresponding closing element tag. For example:

```
<device> <flat-profile>
<Resync_On_Reset> Yes
   </Resync_On_Reset>
<Resync_Periodic> 7200
   </Resync_Periodic>
<Profile_Rule>tftp://prov.telco.com: 6900/cisco/config/spa962.cfg
   </Profile_Rule>
</flat-profile> </device>
```

- Element tags are case sensitive.
- Empty element tags are allowed. Enter the opening element tag without a corresponding element tag, and insert a space and a forward slash before the less-than symbol. In this example, Profile Rule B is empty:

```
<Profile_Rule_B />
```

- Unrecognized element names are ignored.
- An empty element tag can be used to prevent the overwriting of any user-supplied values during a resync operation. In the following example, the user speed dial settings are unchanged:

```
<Speed_Dial_2_2_  ua=”rw”/>
  <Speed_Dial_3_2_  ua=”rw”/>
  <Speed_Dial_4_2_  ua=”rw”/>
  <Speed_Dial_5_2_  ua=”rw”/>
  <Speed_Dial_6_2_  ua=”rw”/>
  <Speed_Dial_7_2_  ua=”rw”/>
  <Speed_Dial_8_2_  ua=”rw”/>
  <Speed_Dial_9_2_  ua=”rw”/>
<device> </flat-profile> </device>
```

- An empty value can be used to set the corresponding parameter to an empty string. Enter an opening and closing element without any value between them. In the following example, the GPP_A parameter is set to an empty string.

```
<device> <flat-profile>
<GPP_A>
  </GPP_A>
<flat-profile> </device>
```

User Access

The user access (**ua**) attribute controls access by the User account for specific parameters. If the **ua** attribute is not specified in an element tag, the factory default user access is applied for the corresponding parameter applied. Access by the Admin account is unaffected by this attribute.

The **ua** attribute, if present, must have one of the following values:

- na—no access
- ro—read-only
- rw—read/write

The **ua** attribute is illustrated by the following example:

```
<device> <flat-profile>
  <SIP_TOS_DiffServ_Value_1_   ua="na"/>
  <Dial_Plan_1_   ua="ro"/>
  <Dial_Plan_2_   ua="rw"/>
<flat-profile> </device>
```

The value of the **ua** option must be enclosed by double quotes.

Access Control for LCD GUI

When the parameter <Phone-UI-User-Mode> is enabled, the phone UI honors the user access attribute of the relevant parameters when it comes to presenting a menu item.

For menu entries associated with a single configuration parameter:

- Provisioning the parameter with "ua=na" ("ua" stands for "user access") attribute makes the entry disappear.
- Provisioning the parameter with "ua=ro" attribute makes the entry read-only and non-editable.

For menu entries that are associated with multiple configuration parameters:

- Provisioning all concerned parameters with "ua=na" attribute makes the entries disappear.

Note For login as normal user or admin from LCD GUI, the default display for all setting page is "User Mode". After admin login, the mode switches to "Admin Mode", and the attribute is "ua=xx", where all parameters are ignored.

Parameter Properties

These properties apply to the parameters:

- Any parameters that are not specified by a profile are left unchanged in the IP Telephony device.
- Unrecognized parameters are ignored.
- The IP Telephony device recognizes arbitrary, configurable aliases for a limited number of parameter names.
- If the Open format profile contains multiple occurrences of the same parameter tag, the last such occurrence overrides any earlier ones. To avoid inadvertently overriding configuration values for a parameter, it is recommended that at most one instance of a parameter be specified in any one profile.

Formatting

These properties apply to the formatting of the strings:

- Comments are allowed by using standard XML syntax.

  ```
  <!-- My comment is typed here -->
  ```

- Leading and trailing white space is allowed for readability and will be removed from the parameter value.
- New lines within a value are converted to spaces.
- An XML header of the form <? . . . ?> is allowed, but is ignored by the IP Telephony device.

- To enter special characters, use basic XML character escapes, as shown in the following table.

Special Character	XML Escape Sequence
& (ampersand)	&amp;
< (less than)	&lt;
> (greater than)	&gt;
’ (apostrophe)	&apos;
” (double quote)	&quot;

  In the following example, character escapes are entered to represent the greater than and less than symbols that are required in a dial plan rule. This example defines an information hotline dial plan that sets the Dial_Plan[1] parameter equal to (S0 <:18005551212>).

```
<device> <flat-profile>
 <Dial_Plan_1_>
    (S0 &lt;:18005551212&gt;)
  </Dial_Plan_1_>
</flat-profile> </device>
```

- Numeric character escapes, using decimal and hexadecimal values (s.a. `&#40;` and `&#x2e;`), are translated.
- The firmware does not support the full Unicode character set, but only the ASCII subset.

Open Profile (XML-style) Compression and Encryption

The Open configuration profile can be compressed to reduce the network load on the provisioning server. It also can be encrypted to protect confidential information. Compression is not required, but it must precede encryption.

Open Profile Compression

The supported compression method is the gzip deflate algorithm (RFC1951). The gzip utility and the compression library that implements the same algorithm (zlib) are available from Internet sites.

To identify when compression is applied, the IP Telephony device expects the compressed file to contain a gzip compatible header, as generated by invoking the gzip utility on the original Open profile. The IP Telephony device inspects the downloaded file header to determine the format of the file.

For example, if `profile.xml` is a valid profile, the file `profile.xml.gz` is also accepted. This profile type can be generated with either of the following commands:

```
>gzip profile.xml
```
replaces original file with compressed file.

```
>cat profile.xml | gzip > profile.xml.gz
```
leaves original file in place, produces new compressed file.

A tutorial on compression is provided in the “Open Profile gzip Compression” section on page 4-12.

Open Profile Encryption by Using AES

An Open configuration profile can be encrypted by using symmetric key encryption, whether or not the file is compressed. The supported encryption algorithm is the American Encryption Standard (AES), using 256-bit keys, applied in cipher block chaining mode.

Note Compression must precede encryption for the IP Telephony device to recognize a compressed and encrypted Open format profile. A tutorial on encryption is provided in Profile Encryption by Using OpenSSL, page 4-13.

The OpenSSL encryption tool, available for download from various Internet sites, can be used to perform the encryption. Support for 256-bit AES encryption might require recompilation of the tool (to enable the AES code). The firmware has been tested against version openssl-0.9.7c.

If the file is encrypted, the profile expects the file to have the same format as generated by the following command:

```
# example encryption key = SecretPhrase1234

openssl enc –e –aes-256-cbc –k SecretPhrase1234 –in profile.xml –out profile.cfg

# analogous invocation for a compressed xml file

openssl enc –e –aes-256-cbc –k SecretPhrase1234 –in profile.xml.gz –out profile.cfg
```

A lower case -k precedes the secret key, which can be any plain text phrase and is used to generate a random 64-bit salt. Then, in combination with the secret specified with the -k argument, the encryption tool derives a random 128-bit initial vector, and the actual 256-bit encryption key.

When this form of encryption is used to encrypt a configuration profile, the IP Telephony device must be informed of the secret key value to decrypt the file. This value is specified as a qualifier in the profile URL. The syntax is as follows, using an explicit URL:

```
[--key “SecretPhrase1234”] http://prov.telco.com/path/profile.cfg
```

This value is programmed by using one of the Profile_Rule parameters. The key must be preprovisioned into the unit at an earlier time. This bootstrap of the secret key can be accomplished securely by using HTTPS.

Preencrypting configuration profiles off-line with symmetric key encryption allows the use of HTTP for resyncing profiles. The provisioning server uses HTTPS to handle initial provisioning of the IP Telephony device after deployment. This feature reduces the load on the HTTPS server in large scale deployments.

The final file name does not need to follow a specific format, but it is conventional to end the name with the .cfg extension to indicate that it is a configuration profile.

Comments

During development and scripting, it is often convenient to temporarily disable a provisioning parameter by entering a # character at the start of the parameter value. This effectively comments-out the remaining text in that parameter.

For example, a Profile_Rule with the value `“# http://192.168.1.200/sample.cfg”` is equivalent to an empty Profile_Rule. The # character comment-mechanism applies to the Profile_Rule*, Upgrade_Rule, and Resync_Trigger_* parameters.

Macro Expansion

Several provisioning parameters undergo macro expansion internally prior to being evaluated. This preevaluation step provides greater flexibility controlling the resync and upgrade activities of the IP Telephony device.

The parameter groups which undergo macro expansion before evaluation are as follows:

- Resync_Trigger_*
- Profile_Rule*
- Log_Resync_*
- Upgrade_Rule
- Log_Upgrade_*

Under certain conditions, some general purpose parameters (GPP_*) also undergo macro expansion, as explicitly indicated in the Optional Resync Arguments section.

During macro expansion, expressions of the form $NAME and $(NAME) are replaced by the contents of the named variables. These variables include general purpose parameters, several product identifiers, certain event timers, and provisioning state values. For a complete list, see the "Macro Expansion Variables" section on page 5-5.

In the following example, the expression $(MAU) is used to insert the MAC address 000E08012345.

The administrator enters: $(MAU)config.cfg
The resulting macro expansion for a device with MAC address 000E08012345 is:
000E08012345config.cfg

If a macro name is not recognized, it remains unexpanded. For example, the name STRANGE is not recognized as a valid macro name, while MAU is recognized as a valid macro name.

The administrator enters: $STRANGE$MAU.cfg
The resulting macro expansion for a device with MAC address 000E08012345 is:
$STRANGE000E08012345.cfg

Macro expansion is not applied recursively. For example, $$MAU" expands into $MAU" (the $$ is expanded), and does not result in the MAC address.

The special purpose parameters (GPP_SA through GPP_SD), whose contents are mapped to the macro expressions $SA through $SD, are only macro expanded as the argument of the **--key** option in a resync URL.

Also, the macro expression can qualify the expansion so that only a substring of the macro variable is used instead of its full value, such as a portion of the MAC address.

The syntax for substring macro expansion is $(NAME:p) and $(NAME:p:q), where p and q are non-negative integers. The resulting expansion results in the macro variable substring starting at character offset p, and of length q (or till end-of-string if q is not specified). Refer to the following examples.

The administrator enters: $(MAU:4)
The resulting macro expansion for a device with MAC address 000E08012345 is: 08012345

The administrator enters: $(MAU:8:2)
The resulting macro expansion for a device with MAC address 000E08012345 is: 23

Conditional Expressions

Conditional expressions can trigger resync events and select from alternative URLs for resync and upgrade operations.

Conditional expressions consist of a list of comparisons, separated by the **and** operator. All comparisons must be satisfied for the condition to be true.

Each comparison can relate one of three types of literals:

- Integer values
- Software or hardware version numbers
- Doubled-quoted strings

Note that version numbers take the form of three non-negative integers separated by periods (major, minor, and build numbers), plus an optional alphanumeric string in parentheses. No spaces are allowed.

The following are examples of valid version numbers:

```
1.0.31(b)
1.0.33
2.0.3(G)
2.0.3(0412s)
2.0.6
```

Quoted strings can be compared for equality or inequality. Integers and version numbers can also be compared arithmetically. The comparison operators can be expressed as symbols or as acronyms. Acronyms are particularly convenient when expressing the condition in an Open format profile.

Operator	Alternate Syntax	Description	Applicable to Integer and Version Operands	Applicable to Quoted String Operands
=	eq	equal to	Yes	Yes
!=	ne	not equal to	Yes	Yes
<	lt	less than	Yes	No
<=	le	less than or equal to	Yes	No
>	gt	greater than	Yes	No
>=	ge	greater than or equal to	Yes	No

For legacy support to firmware versions prior to 2.0.6, the not-equal-to operator can also be expressed as a single ! character (in place of the two-character != string).

Conditional expressions typically involve macro-expanded variables. For example:

```
$REGTMR1 gt 300 and $PRVTMR gt 1200 and "$EXTIP" ne ""

$SWVER ge 2.0.6 and "$CCERT" eq "Installed"
```

It is important to enclose macro variables in double quotes where a string literal is expected. Do not do so where a number or version number is expected.

For legacy support of firmware versions prior to 2.0.6, a relational expression with no left-hand-side operand assumes $SWVER as the implicit left-hand-side. For example, ! 1.0.33 is equivalent to: $SWVER != 1.0.33.

When used in the context of the Profile_Rule* and Upgrade_Rule parameters, conditional expressions must be enclosed within the syntax "(expr)?" as in the following upgrade rule example:

```
($SWVER ne 2.0.6)? http://ps.tell.com/sw/spa021024.bin
```

On the other hand, the syntax above using parentheses should not be used when configuring the Resync_Trigger_* parameters.

Assignment Expressions

Arbitrary parameters can be pre-assigned values within the context of Profile_Rule* and Upgrade_Rule parameter. This causes the assignment to be performed before the profile if retrieved.

The syntax for performing these assignments is a list of individual parameter assignments, enclosed within parentheses (assignments)!, with each assignment taking the form:

ParameterXMLName = "Value" ;

Note that the recognized parameter names correspond to the names as for XML-based profiles.

Any parameter can be assigned a new value in this way, and macro-expansion applies. For example, the following is a valid assignment expression:

```
(User_ID_1_ = "uid$B" ; GPP_C = "" ; GPP_D = "$MA" ;)!
```

For conciseness, the general purpose parameters GPP_A through GPP_P can also be referred to by the single lowercase letters a through p. The example above is equivalent to the following:

```
(User_ID_1_ = "uid$B" ; c = "" ; d = "$MA" ;)!
```

White space can be used for readability.

URL Syntax

Standard URL syntax is used to specify how to retrieve configuration files and firmware loads in Profile_Rule* and Upgrade_Rule parameters, respectively. The syntax is as follows:

```
[ scheme:// ] [ server [:port]] filepath
```

Where scheme is one of the following values:

- tftp
- http
- https

If scheme is omitted, tftp is assumed. The server can be a DNS-recognized host name or a numeric IP address. The port is the destination UDP or TCP port number. The filepath must begin with the root directory (/); it must be an absolute path.

If server is missing, then the tftp server specified through DHCP (option 66) is used.

If port is missing, then the standard port for the specified scheme is used (tftp uses UDP port 69, http uses TCP port 80, https uses TCP port 443).

A filepath must be present. It need not necessarily refer to a static file, but can indicate dynamic content obtained through CGI.

Macro expansion applies within URLs. The following are examples of valid URLs:

```
/$MA.cfg
/cisco/spa021025.bin
192.168.1.130/profiles/init.cfg
```

```
tftp://prov.call.com/cpe/cisco$MA.cfg
http://neptune.speak.net:8080/prov/$D/$E.cfg
https://secure.me.com/profile?Linksys
```

Optional Resync Argument

The URLs entered in Profile_Rule* parameters can be preceded by optional argument, collectively enclosed by square brackets. The option argument is key.

key

The **key** option is used to specify an encryption key. It is required to decrypt profiles that have been encrypted with an explicit key. The key itself is specified as a (possibly quoted) string following the term **--key**.

Some usage examples:

```
[--key VerySecretValue]
[--key "my secret phrase"]
[--key a37d2fb9055c1d04883a0745eb0917a4]
```

The bracketed optional arguments are macro expanded. In particular, note that the special purpose parameters GPP_SA through GPP_SD are only macro expanded into their macro variables $SA through $SD when used as arguments of the key option, as in the following examples:

```
[--key $SC]
[--key "$SD"]
```

In the case of Open format profiles, the argument to **--key** must be the same as the argument to the **-k** option given to **openssl**.

Applying a Profile to the IP Telephony Device

After you create an XML configuration script, it must be passed to the IP Telephony device for application. To apply the configuration, choose one of the following methods:

TFTP and the Resync URL

Complete the following steps to post the configuration file to a TFTP server application on your PC.

Step 1 Connect your PC to the ATA LAN.

Step 2 Run a TFTP server application on the PC and make sure that the configuration file is available in the TFTP root directory.

Step 3 In a web browser, and enter the LAN IP address of the IP Telephony device, the IP address of the computer, the filename, and the login credentials, in this format:

http://<*WAN_IP_Address*>/admin/resync?tftp://<*PC_IP_Address*>/<*file_name*>&xuser=admin&xpassword=<*password*>

Example:

```
http://192.168.15.1/admin/resync?tftp://192.168.15.100/my_config.xml&xuser=admin&xpassword
=admin
```

Direct HTTP Post Using cURL

Complete the following steps to post the configuration to the IP Telephony device by using cURL. This command line tool is used to transfer data with a URL syntax. To download cURL, see:

http://curl.haxx.se/download.html

Step 1 Connect your PC to the LAN port of the IP Telephony device.

Step 2 Post the configuration file to the IP Telephony device by entering the following cURL command:

```
curl -d @my_config.xml
"http://192.168.15.1/admin/config.xml&xuser=admin&xpassword=admin"
```

Using Provisioning Parameters

This section describes the provisioning parameters broadly organized according to function:

- General Purpose Parameters
- Enables
- Triggers
- Configurable Schedules
- Profile Rules
- Upgrade Rule

General Purpose Parameters

The general purpose parameters GPP_* are used as free string registers when configuring the IP Telephony device to interact with a particular provisioning server solution. The GPP_* parameters are empty by default. They can be configured to contain diverse values, including the following:

- Encryption keys
- URLs
- Multi-stage provisioning status information
- Post request templates
- Parameter name alias maps
- Partial string values, eventually combined into complete parameter values.

The GPP_* parameters are available for macro expansion within other provisioning parameters. For this purpose, single-letter upper-case macro names (A through P) are sufficient to identify the contents of GPP_A through GPP_P. Also, the two-letter upper-case macro names SA through SD identify GPP_SA through GPP_SD as a special case when used as arguments of the key URL option.

For example, if GPP_A contains the string ABC, and GPP_B contains 123, the expression AB macro expands into ABC123.

Enables

All profile resync and firmware upgrade operations are controlled by the Provision_Enable and Upgrade_Enable parameters. These parameters control resyncs and upgrades independently of each other. These parameters also control resync and upgrade URL commands issued through the administration web server. Both of these parameters are set to yes by default.

In addition, the Resync_From_SIP parameter controls requests for resync operations via a SIP NOTIFY event sent from the service provider proxy server to the IP Telephony device. If enabled, the proxy can request a resync by sending a SIP NOTIFY message containing the Event: resync header to the device.

The device challenges the request with a 401 response (authorization refused for used credentials), and expects an authenticated subsequent request before honoring the resync request from the proxy. The Event: reboot_now and Event: restart_now headers perform cold and warm restarts, respectively, are also challenged.

The two remaining enables are Resync_On_Reset and Resync_After_Upgrade_Attempt. These determine if the device performs a resync operation after power-up software reboots and after each upgrade attempt.

When enabling Resync_On_Reset, the device introduces a random delay following the boot-up sequence before actually performing the reset. The delay is a random time up to the value specified in Resync_Random_Delay (in seconds). In a pool of IP Telephony devices, all of which are simultaneously powered up, this introduces a spread in the times at which each unit initiates a resync request to the provisioning server. This feature can be useful in a large residential deployment, in the case of a regional power failures.

Triggers

The IP Telephony device allows you to resync at specific intervals or at a specific time.

Resyncing at Specific Intervals

The IP Telephony device is designed to resync with the provisioning server periodically. The resync interval is configured in Resync_Periodic (seconds). If this value is left empty, the device does not resync periodically.

The resync typically takes place when the voice lines are idle. In case a voice line is active when a resync is due, the IP Telephony device delays the resync procedure until the line becomes idle again. However, it waits no longer than Forced_Resync_Delay (seconds). A resync might cause configuration parameter values to change. This, in turn, causes a firmware reboot and terminates any voice connection active at the time of the resync.

If a resync operation fails because the IP Telephony device was unable to retrieve a profile from the server, if the downloaded file is corrupt, or an internal error occurs, the device tries to resync again after a time specified in Resync_Error_Retry_Delay (seconds). If Resync_Error_Retry_Delay is set to 0, the device does not try to resync again following a failed resync attempt.

When upgrading, if an upgrade fails, a retry is performed after Upgrade_Error_Retry_Delay seconds.

Two configurable parameters are available to conditionally trigger a resync: Resync_Trigger_1 and Resync_Trigger_2. Each of these parameters can be programmed with a conditional expression (which undergoes macro expansion). If the condition in any of these parameters evaluates to true, a resync operation is triggered, as though the periodic resync timer had expired.

The following example condition triggers a resync if Line 1 failed to register for more than 5 minutes (300 seconds), and at least 10 minutes (600 seconds) have elapsed since the last resync attempt.

```
$REGTMR1 gt 300 and $PRVTMR ge 600
```

Resyncing at a Specific Time

The Resync_At parameter allows the phone to resync at a specific time. This parameter uses the 24-hour format (hhmm) to specify the time.

Another parameter Resync_At_Random_Delay allows the phone to resync at an unspecified delay in time. This parameter uses a positive integer format to specify the time.

To avoid simultaneously flooding the server with resync requests from multiple phones set to resync at the same time, the phone triggers the resync up to 10 minutes after the specified time.

For example, if you set the resync time to 1000 (10 a.m.), the phone triggers the resync anytime between 10:00 a.m. and 10:10 a.m.

By default, this feature is disabled. If the Resync_At parameter is provisioned, the Resync_Periodic parameter is ignored.

Configurable Schedules

You can configure schedules for periodic resyncs, and you can specify the retry intervals for resync and upgrade failures by using these provisioning parameters:

- Resync_Periodic
- Resync_Error_Retry_Delay
- Upgrade_Error_Retry_Delay

Each parameter accepts a single delay value (seconds). The new extended syntax allows for a comma-separated list of consecutive delay elements. The last element in the sequence is implicitly repeated forever. Below is an example:

```
Resync_Periodic=7200
Resync_Error_Retry_Delay=1800,3600,7200,14400
```

In the above example, the IP Telephony device periodically resyncs every two hours. In case of a resync failure, the device retries at these intervals: 30 minutes, 1 hour, 2 hours, 4 hours. It continues trying at 4-hour intervals until it successfully resyncs.

Optionally, you can use a plus sign to specify an additional numeric value that appends a random extra delay, as shown in this example:

```
Resync_Periodic=3600+600
Resync_Error_Retry_Delay=1800+300,3600+600,7200+900
```

In the above example, the device periodically resyncs every hour (plus an additional random delay of up to 10 minutes). In case of a resync failure, the device retries at these intervals: 30 minutes (plus up to 5 minutes). 1 hour (plus up to 10 minutes), 2 hours (plus up to 15 minutes). It continues trying at 2-hour intervals (plus up to 15 minutes) until it successfully resyncs.

Below is another example:

```
Upgrade_Error_Retry_Delay  =  1800,3600,7200,14400+3600
```

In this example, if a remote upgrade attempt fails, the device retries the upgrade in 30 minutes, then again after one more hour, then in two hours. If it still fails, it subsequently retries every four to five hours, until it succeeds.

Profile Rules

The IP Telephony device provides multiple remote configuration profile parameters (Profile_Rule*). This means that each resync operation can retrieve multiple files, potentially managed by different servers.

In the simplest scenario, the device resyncs periodically to a single profile on a central server, which updates all pertinent internal parameters. Alternatively, the profile can be split between different files. One file is common for all the IP Telephony devices in a deployment, while a separate file is provided that is unique for each account. Encryption keys and certificate information could be supplied by still another profile, stored on a separate server.

Whenever a resync operation is due, the IP Telephony device evaluates the four Profile_Rule* parameters in sequence:

1. Profile_Rule
2. Profile_Rule_B
3. Profile_Rule_C
4. Profile_Rule_D

Each evaluation can result in a profile being retrieved from a remote provisioning server, possibly updating some number of internal parameters. If an evaluation fails, the resync sequence is interrupted, and is retried again from the beginning specified by the Resync_Error_Retry_Delay parameter (seconds). If all evaluations succeed, the device waits for the second specified by the Resync_Periodic parameter, and then performs a resync again.

The contents of each Profile_Rule* parameter consist of a set of alternatives. The alternatives are separated by the | (pipe) character. Each alternative consists of a conditional expression, an assignment expression, a profile URL, and any associated URL options. All these components are optional within each alternative. The following are the valid combinations, and the order in which they must appear, if present:

```
[ conditional-expr ] [ assignment-expr ] [[ options ] URL ]
```

Within each Profile_Rule* parameter, all of the alternatives except the last one must provide a conditional expression. This expression is evaluated and processed as follows:

1. Conditions are evaluated from left to right, until one is found that evaluates as true (or until one alternative is found with no conditional expression)
2. Any accompanying assignment expression is evaluated, if present
3. If a URL is specified as part of that alternative, an attempt is made to download the profile located at the specified URL, and update the internal parameters accordingly.

If all alternatives have conditional expressions, and none evaluates to true (or if the whole profile rule is empty), then the entire Profile_Rule* parameter is skipped, and the next profile rule parameter in the sequence is evaluated.

The following are some examples of valid programming for a single Profile_Rule* parameter.

The following example resyncs unconditionally to the profile at the specified URL, performing an HTTP GET request to the remote provisioning server.

```
http://remote.server.com/cisco/$MA.cfg
```

In the following example, the device resyncs to two different URLs, depending on the registration state of Line 1. In case of lost registration, the device performs an HTTP POST to a CGI script, transmitting the contents of the macro expanded GPP_A (which may provide additional information on the state of the device).

```
($REGTMR1 eq 0)? http://p.tel.com/has-reg.cfg
| [--post a] http://p.tel.com/lost-reg?
```

In the following example, the device resyncs to the same server, but provides additional information if a certificate is not installed in the unit (for legacy pre-2.0 units).

```
("$CCERT" eq "Installed")? https://p.tel.com/config?
| https://p.tel.com/config?cisco$MAU
```

In the following example, Line 1 is disabled until GPP_A is set equal to Provisioned through the first URL. Afterwards, it resyncs to the second URL.

```
("$A" ne "Provisioned")? (Line_Enable_1_ = "No";)! https://p.tel.com/init-prov
| https://p.tel.com/configs
```

In the following example, the profile returned by the server is assumed to contain XML element tags that need to be remapped to proper parameter names by the aliases map stored in GPP_B.

```
[--alias b] https://p.tel.com/account/spa$MA.xml
```

A resync is typically considered unsuccessful if a requested profile is not received from the server. This default behavior can be overridden by the parameter Resync_Fails_On_FNF. If Resync_Fails_On_FNF is set to No, then the device accepts a file-not-found response from the server as a successful resync. The default value for Resync_Fails_On_FNF is Yes.

Upgrade Rule

The IP Telephony device provides one configurable remote upgrade parameter, Upgrade_Rule. This parameter accepts a syntax similar to the profile rule parameters. URL options not supported for upgrades, but conditional expressions and assignment expressions can be used. If conditional expressions are used, the parameter can be populated with multiple alternatives, separated by the | character. The syntax for each alternative is as follows:

```
    [ conditional-expr ] [ assignment-expr ] URL
```

As in the case of Profile_Rule* parameters, the Upgrade_Rule parameter evaluates each alternative until a conditional expression is satisfied or an alternative has no conditional expression. The accompanying assignment expression is evaluated, if specified. Then, an upgrade to the specified URL is attempted.

If the Upgrade_Rule contains a URL without a conditional expression, the device upgrades to the firmware image specified by the URL. Subsequently, it does not attempt to upgrade again until either the rule itself is modified or the effective combination of scheme + server + port + filepath is changed, following macro expansion and evaluation of the rule.

In order to attempt a firmware upgrade, the device disables audio at the start of the procedure, and reboots at the end of the procedure. For this reason, an upgrade driven by the contents of Upgrade_Rule is only automatically initiated by the device if any voice line is currently inactive.

For example,

```
    http://p.tel.com/firmware/spa021025.bin
```

In this example, the Upgrade_Rule upgrades the firmware to the image stored at the indicated URL. The following is another example:

```
("$F" ne "beta-customer")? http://p.tel.com/firmware/spa021025.bin
| http://p.tel.com/firmware/spa-test-0527s.bin
```

This example directs the unit to load one of two images, based on the contents of a general purpose parameter, GPP_F.

The device can enforce a downgrade limit with respect to firmware revision number. This can be useful as a customization option. If a valid firmware revision number is configured in the parameter Downgrade_Rev_Limit, the device rejects upgrade attempts for firmware versions earlier than the specified limit.

Data Types

The data types used with configuration profile parameters are as follows:

- Uns<n>—Unsigned n-bit value, where n = 8, 16, or 32. It can be specified in decimal or hex format such as 12 or 0x18 as long as the value can fit into n bits.
- Sig<n>—Signed n-bit value. It can be specified in decimal or hex format. Negative values must be preceded by a "-" sign. A + sign before positive value is optional.
- Str<n>—A generic string with up to n non-reserved characters.
- Float<n>—A floating point value with up to n decimal places.
- Time<n>—Time duration in seconds, with up to n decimal places. Extra decimal places specified are ignored.
- PwrLevel—Power level expressed in dBm with 1 decimal place, such as –13.5 or 1.5 (dBm).
- Bool—Boolean value of either "yes" or "no."
- {a,b,c,…}—A choice among a, b, c, …
- IP—IP Address in the form of x.x.x.x, where x between 0 and 255. For example 10.1.2.100.
- Port—TCP/UDP Port number (0-65535). It can be specified in decimal of hex format.
- UserID—User ID as appeared in a URL; up to 63 characters.
- FQDN—Fully Qualified Domain Name, such as "sip.Cisco.com:5060", or "109.12.14.12:12345". It can contain up to 63 characters.
- Phone—A phone number string, such as 14081234567, *69, *72, 345678, or a generic URL such as 1234@10.10.10.100:5068, or jsmith@Cisco.com. It can contain up to 39 characters.
- ActCode—Activation code for a supplementary service, such as *69. It can contain up to 7 characters.
- PhTmplt—A phone number template. Each template may contain one or more patterns separated by a ",". White space at the beginning of each pattern is ignored. "?" and "*" represent wildcard characters. To represent literally use %xx. For example, %2a represents *. It can contain up to 39 characters. Examples: "1408*, 1510*", "1408123????, 555?1.".
- RscTmplt—A template of SIP Response Status Code, such as "404, 5*", "61?", "407, 408, 487, 481". It can contain up to 39 characters.
- CadScript—A mini-script that specifies the cadence parameters of a signal. Up to 127 characters. Syntax: S_1[;S_2], where:
  S_i=D_i($on_{i,1}$/$off_{i,1}$[,$on_{i,2}$/$off_{i,2}$[,$on_{i,3}$/$off_{i,3}$[,$on_{i,4}$/$off_{i,4}$[,$on_{i,5}$/$off_{i,5}$[,$on_{i,6}$/$off_{i,6}$]]]]]) and is known as a

section, $on_{i,j}$ and $off_{i,j}$ are the on/off duration in seconds of a *segment* and i = 1 or 2, and j = 1 to 6. D_i is the total duration of the section in seconds. All durations can have up to three decimal places to provide 1 ms resolution. The wildcard character "*" stands for infinite duration. The segments within a section are played in order and repeated until the total duration is played.

Example 1:

```
60(2/4)

Number of Cadence Sections = 1
Cadence Section 1: Section Length = 60 s
Number of Segments = 1
Segment 1: On=2s, Off=4s

Total Ring Length = 60s
```

Example 2—Distinctive ring (short,short,short,long):

```
60(.2/.2,.2/.2,.2/.2,1/4)

Number of Cadence Sections = 1
Cadence Section 1: Section Length = 60s
Number of Segments = 4
Segment 1: On=0.2s, Off=0.2s
Segment 2: On=0.2s, Off=0.2s
Segment 3: On=0.2s, Off=0.2s
Segment 4: On=1.0s, Off=4.0s

Total Ring Length = 60s
```

- FreqScript—A mini-script that specifics the frequency and level parameters of a tone. Up to 127 characters. Syntax: F_1@L_1[,F_2@L_{L2}[,F_3@L_3[,F_4@L_4[,F_5@L_5[,F_6@L_6]]]]], where F_1–F_6 are frequency in Hz (unsigned integers only) and L_1–L_6 are corresponding levels in dBm (with up to 1 decimal places). White spaces before and after the comma are allowed (but not recommended).

  Example 1—Call Waiting Tone:

```
     440@-10

  Number of Frequencies = 1
     Frequency 2 = 440 Hz at -10 dBm
```

  Example 2—Dial Tone:

```
      350@-19,440@-19

  Number of Frequencies = 2
     Frequency 1 = 350 Hz at -19 dBm
     Frequency 2 = 440 Hz at -19 dBm
```

- ToneScript—A mini-script that specifies the frequency, level and cadence parameters of a call progress tone. May contain up to 127 characters. Syntax: FreqScript;Z_1[;Z_2]. The section Z_1 is similar to the S_1 section in a CadScript except that each on/off segment is followed by a frequency components parameter: $Z_1 = D_1(on_{i,1}/off_{i,1}/f_{i,1}[,on_{i,2}/off_{i,2}/f_{i,2}$ $[,on_{i,3}/off_{i,3}/f_{i,3}$ $[,on_{i,4}/off_{i,4}/f_{i,4}$ $[,on_{i,5}/off_{i,5}/f_{i,5}$ $[,on_{i,6}/off_{i,6}/f_{i,6}]]]]])$, where fi,j = $n_1[+n_2]+n_3[+n_4[+n_5[+n_6]]]]]$ and $1 < n_k < 6$ indicates which of the frequency components given in the FreqScript are used in that segment; if more than one frequency component is used in a segment, the components are summed together.

  Example 1—Dial tone:

```
350@-19,440@-19;10(*/0/1+2)

Number of Frequencies = 2
Frequency 1 = 350 Hz at -19 dBm
Frequency 2 = 440 Hz at -19 dBm
Number of Cadence Sections = 1
Cadence Section 1: Section Length = 10 s
Number of Segments = 1
Segment 1: On=forever, with Frequencies 1 and 2

Total Tone Length = 10s
```

Example 2—Stutter tone:

```
350@-19,440@-19;2(.1/.1/1+2);10(*/0/1+2)

Number of Frequencies = 2
Frequency 1 = 350 Hz at -19 dBm
Frequency 2 = 440 Hz at -19 dBm
Number of Cadence Sections = 2
Cadence Section 1: Section Length = 2s
Number of Segments = 1
Segment 1: On=0.1s, Off=0.1s with Frequencies 1 and 2
Cadence Section 2: Section Length = 10s
Number of Segments = 1
Segment 1: On=forever, with Frequencies 1 and 2

Total Tone Length = 12s
```

Example 3—SIT tone:

```
985@-16,1428@-16,1777@-16;20(.380/0/1,.380/0/2,.380/0/3,0/4/0)
Number of Frequencies = 3
Frequency 1 = 985 Hz at -16 dBm
Frequency 2 = 1428 Hz at -16 dBm
Frequency 3 = 1777 Hz at -16 dBm
Number of Cadence Sections = 1
Cadence Section 1: Section Length = 20s
Number of Segments = 4
Segment 1: On=0.38s, Off=0s, with Frequency 1
Segment 2: On=0.38s, Off=0s, with Frequency 2
Segment 3: On=0.38s, Off=0s, with Frequency 3
Segment 4: On=0s, Off=4s, with no frequency components
Total Tone Length = 20s
```

- ProvisioningRuleSyntax—Scripting syntax used to define configuration resync and firmware upgrade rules.
- DialPlanScript—Scripting syntax used to specify Line 1 and Line 2 dial plans.

Note

- <Par Name> represents a configuration parameter name. In a profile, the corresponding tag is formed by replacing the space with an underscore "_", such as **Par_Name**.
- An empty default value field implies an empty string < **""** >.
- The IP Telephony device continues to use the last configured values for tags that are not present in a given profile.
- Templates are compared in the order given. The first, *not the closest*, match is selected. The parameter name must match exactly.

- If more than one definition for a parameter is given in a profile, the last such definition in the file is the one that takes effect in the IP Telephony device.
- A parameter specification with an empty parameter value forces the parameter back to its default value. To specify an empty string instead, use the empty string "" as the parameter value.

In-House Preprovisioning and Provisioning Servers

Cisco IP Telephony devices, other than RC units, are preprovisioned by the service provider with a profile. That pre-provision profile can range from a a limited set of parameters that resynchronizes the IP Telephony device to another profile with a complete set of parameters delivered by remote server. Or, it can be a complete set of parameters. By default, the IP Telephony device resynchronizes on power up and at intervals configured in the profile. When the user connects the IP Telephony device at the customer premises, the device downloads the updated profile and any firmware updates.

This process of preprovisioning, deployment, and remote provisioning can be accomplished many ways. This chapter describes the features and functionality available when preprovisioning the Cisco IP Telephony devices in-house and provisioning them remotely:

- Server Preparation and Software Tools, page 3-1
- In-House Device Preprovisioning, page 3-2
- Provisioning Server Setup, page 3-2

Server Preparation and Software Tools

The examples presented in this chapter require the availability of one or more servers. These servers can be installed and run on a local PC:

- TFTP (UDP port 69)
- syslog (UDP port 514)
- HTTP (TCP port 80)
- HTTPS (TCP port 443).

To troubleshoot server configuration, it is helpful to install clients for each type of server on a separate server machine. This establishes proper server operation, independent of the interaction with the Cisco IP Telephony devices.

Cisco also recommends the installation of the following software tools:

- To generate configuration profiles, it is useful to install the open source gzip compression utility.
- For profile encryption and HTTPS operations, install the open source OpenSSL software package.

- To test the dynamic generation of profiles and one-step remote provisioning using HTTPS, a scripting language with CGI scripting support, such as open source Perl language tools, is recommended.
- To verify secure exchanges between provisioning servers and the Cisco IP Telephony devices, install an Ethernet packet sniffer (such as the freely downloadable Ethereal/Wireshark). Capture an Ethernet packet trace of the interaction between the IP Telephony device and the provisioning server by running the packet sniffer on a PC that is connected to a switch with port mirroring enabled. For HTTPS transactions, you can use the ssldump utility.

In-House Device Preprovisioning

With the Cisco factory default configuration, an IP Telephony device automatically tries to resync to a profile on a TFTP server. The information regarding the profile and TFTP server configured for preprovisioning is delivered to the device by a managed DHCP server on a LAN. The service provider connects each new IP Telephony device that LAN and the IP Telephony device automatically resyncs to the local TFTP server, initializing its internal state in preparation for deployment. This preprovisioning profile typically includes the URL of a remote provisioning server that will keep the device updated after it is deployed and connected to the customer network.

The preprovisioned device bar code can be scanned to record its MAC address or serial number before the IP Telephony device is shipped to the customer. This information can be used to create the profile to which the IP Telephony device will resynchronize.

Upon receiving the IP Telephony device, the customer connects it to the broadband link. On power-up the IP Telephony device contacts the provisioning server through the URL configured through preprovisioning to for its resync and updates the profile and firmware as necessary.

Provisioning Server Setup

This section describes setup requirements for provisioning an IP Telephony device by using various servers and different scenarios. For testing purposes and for the purposes of this document, provisioning servers are installed and run on a local PC. Also, generally available software tools are useful for provisioning the Cisco IP Telephony devices.

TFTP Provisioning

Cisco IP Telephony devices support TFTP for both provisioning resync and firmware upgrade operations. When devices are deployed remotely, HTTP is recommended for provisioning as it offers greater reliability, given NAT and router protection mechanisms. TFTP is useful for the in-house preprovisioning of a large number of unprovisioned devices. See the "In-House Device Preprovisioning" section on page 3-2 for more information.

The IP Telephony device is able to obtain a TFTP server IP address directly from the DHCP server through DHCP option 66. If a Profile_Rule is configured with the filepath of that TFTP server, the device downloads its profile from the TFTP server when it is connected to a LAN and powered up.

The Profile_Rule provided with the factory default configuration is /*device*.cfg. For example, on a CP-8831-3PCC the filename is CP-8831-3PCC.cfg. If the device has the factory default profile, when powered up it resyncs to this file on the local TFTP server specified by DHCP option 66. (The filepath is relative to the TFTP server virtual root directory.)

Remote Endpoint Control and NAT

The IP Telephony device accesses the Internet through a router by using network address translation (NAT). For enhanced security, the router might attempt to block unauthorized incoming packets by implementing symmetric NAT (a packet filtering strategy that severely restricts the packets that are allowed to enter the protected network from the Internet). For this reason, remote provisioning by using TFTP is not recommended.

Voice over IP can co-exist with NAT only when some form of NAT traversal is provided. Configure Simple Traversal of UDP through NAT (STUN). This option requires that the user have (1) a dynamic external (public) IP address from your service, (2) a computer running STUN server software, and (3) an edge device with an asymmetric NAT mechanism.

HTTP Provisioning

The IP Telephony device behaves like a browser requesting web pages from a remote Internet site. This provides a reliable means of reaching the provisioning server, even when a customer router implements symmetric NAT or other protection mechanisms. HTTP and HTTPS work more reliably than TFTP in remote deployments, especially when the deployed units are connected behind residential firewalls or NAT-enabled routers.

Basic HTTP-based provisioning relies on the HTTP GET method for retrieving configuration profiles. Typically, a configuration file is created for each deployed IP Telephony device, and these files are stored within a HTTP server directory. When the server receives the GET request, it simply returns the file specified in the GET request header.

Alternatively, the requested URL can invoke a CGI script (using the GET method). The configuration profile is generated dynamically by querying a customer database and producing the profile on-the-fly.

When CGI handles resync requests, the IP Telephony device can use the HTTP POST method to request the resync configuration data. The device can be configured to convey certain status and identification information to the server within the body of the HTTP POST request. The server uses this information to generate a desired response configuration profile, or store the status information for later analysis and tracking.

As part of both GET and POST requests, the IP Telephony device automatically includes basic identifying information in the request header, in the User-Agent field. This information conveys the manufacturer, product name, current firmware version, and product serial number of the device.

For example, the following example is the User-Agent request field from a CP-8831-3PCC :

```
User-Agent: cisco/CP-8831-3PCC (88012BA01234)
```

When the IP Telephony device is configured to resync to a configuration profile by using HTTP, it is recommended that the profile be encrypted to protect confidential information. The IP Telephony device supports 256-bit AES in CBC mode to decrypt profiles. Encrypted profiles downloaded by the IP Telephony device by using HTTP avoid the danger of exposing confidential information contained in the configuration profile. This resync mode produces a lower computational load on the provisioning server when compared to using HTTPS.

Note The Cisco Unified IP Conference Phone 8831 for Third-Party Call Control supports HTTP Version 1.0, HTTP Version1.1, and Chunk Encoding when HTTP Version 1.1 is the negotiated transport protocol.

HTTP Status Code Handling on Resync and Upgrade

The phone supports improved HTTP response for remote provisioning (Resync). Current phone behavior is categorized in 3 ways:

- A—Success, where the subsequent requests are determined by "Resync Periodic" and "Resync Random Delay" values.
- B—Failure when File Not Found or corrupt profile. Subsequent requests are determined by "Resync Error Retry Delay" value.
- C—Other failure when there is a connection error caused by bad URL or IP address. Subsequent requests are determined by "Resync Error Retry Delay" value. When the request times out:”

Table 3-1 Phone Behavior for HTTP Responses

HTTP Status Code	Description	Phone Behavior
301 Moved Permanently	This and future requests should be directed to a new location.	Retry request immediately with new location.
302 Found	Known as Temporarily Moved.	Retry request immediately with new location.
3xx	Other 3xx responses not processed.	C
400 Bad Request	The request cannot be fulfilled due to bad syntax.	C
401 Unauthorized	Basic or digest access authentication challenge.	Immediate retry request with authentication credentials. Maximum 2 retries. Upon failure, the phone’s behavior is C.
403 Forbidden	Server refuses to respond.	C
404 Not Found	Requested resource not found. Subsequent requests by client are permissible.	B
407 Proxy Authentication Required	Basic or digest access authentication challenge.	Immediate retry request with authentication credentials. Maximum 2 retries. Upon failure, the phone’s behavior is C.
4xx	Other client error status codes are not processed.	C
500 Internal Server Error	Generic error message.	The IP conference phone 8831 behavior is C.
501 Not Implemented	The server does not recognize the request method, or it lacks the ability to fulfill the request.	The IP conference phone 8831 behavior is C.
502 Bad Gateway	The server is acting as a gateway or proxy and receives an invalid response from the upstream server.	The IP Conference Phone 8831 behavior is C.
503 Service Unavailable	The server is currently unavailable (overloaded or down for maintenance). This is a temporary state.	The IP Conference Phone 8831 behavior is C.

Table 3-1 Phone Behavior for HTTP Responses (continued)

HTTP Status Code	Description	Phone Behavior
504 Gateway Timeout	The server behaves as a gateway or proxy and does not receive timely response from the upstream server.	C
5xx	Other server error	C

HTTPS Provisioning

For increased security managing remotely deployed units, the IP Telephony device supports HTTPS for provisioning. Each IP Telephony device carries a unique SLL Client Certificate (and associated private key), in addition to a Sipura CA server root certificate. The latter allows the IP Telephony device to recognize authorized provisioning servers, and reject non-authorized servers. On the other hand, the client certificate allows the provisioning server to identify the individual device that issues the request.

For a service provider to manage deployment by using HTTPS, a server certificate must be generated for each provisioning server to which an IP Telephony device resyncs by using HTTPS. The server certificate must be signed by the Cisco Server CA Root Key, whose certificate is carried by all deployed units. To obtain a signed server certificate, the service provider must forward a certificate signing request to Cisco, which signs and returns the server certificate for installation on the provisioning server.

The provisioning server certificate must contain the Common Name (CN) field, and the FQDN of the host running the server in the subject. It might optionally contain information following the host FQDN, separated by a slash (/) character. The following examples are of CN entries that are accepted as valid by the IP Telephony device:

```
CN=sprov.callme.com
CN=pv.telco.net/mailto:admin@telco.net
CN=prof.voice.com/info@voice.com
```

In addition to verifying the server certificate, the IP Telephony device tests the server IP address against a DNS lookup of the server name specified in the server certificate.

A certificate signing request can be generated by using the OpenSSL utility. The following example shows the **openssl** command that produces a 1024-bit RSA public/private key pair and a certificate signing request:

```
openssl req –new –out provserver.csr
```

This command generates the server private key in **privkey.pem** and a corresponding certificate signing request in **provserver.csr**. The service provider keeps the **privkey.pem** secret and submits **provserver.csr** to Cisco for signing. Upon receiving the **provserver.csr** file Cisco generates **provserver.crt**, the signed server certificate.

Cisco also provides a Sipura CA Client Root Certificate to the service provider. This root certificate certifies the authenticity of the client certificate carried by each IP Telephony device. The Cisco Unified IP Conference Phone 8831 for Third-Party Call Control also supports third-party signed certificates such as those provided by Verisign, Cybertrust, and so forth.

The unique client certificate offered by each device during an HTTPS session carries identifying information embedded in its subject field. This information can be made available by the HTTPS server to a CGI script invoked to handle secure requests. In particular, the certificate subject indicates the unit product name (OU element), MAC address (S element), and serial number (L element). The following example from the Cisco Unified IP Conference Phone 8831 for Third-Party Call Control client certificate subject field shows these elements:

```
OU=CP-8831-3PCC, L=88012BA01234, S=000e08abcdef
```

Units manufactured before firmware 2.0.x do not contain individual SSL client certificates. When these units are upgraded to a firmware release in the 2.0.x tree, they become capable of connecting to a secure server using HTTPS, but are only able to supply a generic client certificate if requested to do so by the server. This generic certificate contains the following information in the identifying fields:

```
OU=cisco.com, L=ciscogeneric, S=ciscogeneric
```

To determine if an IP Telephony device carries an individualized certificate, use the $CCERT provisioning macro variable. The variable value expands to either Installed or Not Installed, according to the presence or absence of a unique client certificate. In the case of a generic certificate, it is possible to obtain the serial number of the unit from the HTTP request header in the User-Agent field.

HTTPS servers can be configured to request SSL certificates from connecting clients. If enabled, the server can verify the client certificate by using the Sipura CA Client Root Certificate supplied by Cisco. It can then provide the certificate information to a CGI for further processing.

The location for storing certificates might vary. For example, on an Apache installation the file paths for storing the provisioning server–signed certificate, its associated private key, and the Sipura CA client root certificate are as follows:

```
# Server Certificate:
SSLCertificateFile /etc/httpd/conf/provserver.crt

# Server Private Key:
SSLCertificateKeyFile /etc/httpd/conf/provserver.key

# Certificate Authority (CA):
SSLCACertificateFile /etc/httpd/conf/spacroot.crt
```

Refer to the documentation provided for a HTTPS server for specific information.

Firmware release 2.0.6 and higher supports the following cipher suites for SSL connection to a server by using HTTPS.

Table 3-2 Cipher Suites Supported for Connecting to an HTTPS Server

Numeric Code	Cipher Suite
0x0039	TLS_DHE_RSA_WITH_AES_256_CBC_SHA
0x0035	TLS_RSA_WITH_AES_256_CBC_SHA
0x0033	TLS_DHE_RSA_WITH_AES_128_CBC_SHA
0x002f	TLS_RSA_WITH_AES_128_CBC_SHA
0x0005	TLS_RSA_WITH_RC4_128_SHA
0x0004	TLS_RSA_WITH_RC4_128_MD5
0x0062	TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
0x0060	TLS_RSA_EXPORT1024_WITH_RC4_56_MD5
0x0003	TLS_RSA_EXPORT_WITH_RC4_40_MD5

Redundant Provisioning Servers

The provisioning server can be specified as an IP address or as a fully qualified domain name (FQDN). The use of a FQDN facilitates the deployment of redundant provisioning servers. When the provisioning server is identified through a FQDN, the IP Telephony device attempts to resolve the FQDN to an IP address through DNS. Only DNS A-records are supported for provisioning; DNS SRV address resolution is not available for provisioning. The IP Telephony device continues to process A-records until a server responds. If no server associated with the A-records responds, the IP Telephony device logs an error to the syslog server.

Syslog Server

If a syslog server is configured on the IP Telephony device (using the <Syslog_Server> parameters), the resync and upgrade operations log messages to the syslog server. A message can be generated at the start of a remote file request (configuration profile or firmware load), and at the conclusion of the operation (indicating either success or failure).

The logged messages are configured in the following parameters and macro expanded into the actual syslog messages:

- Log_Request_Msg
- Log_Success_Msg
- Log_Failure_Msg

The Cisco Client Certificate Root Authority signs each unique certificate. The corresponding root certificate is made available to service providers for client authentication purposes.

Provisioning Examples

This chapter provides example procedures for transferring configuration profiles between the IP Telephony device and the provisioning server:

- Basic Resync, page 4-1
- Secure HTTPS Resync, page 4-6
- Profile Management, page 4-12

For information about creating configuration profiles, refer to Chapter 2, "Creating Provisioning Scripts for Configuration Profile."

Basic Resync

This section demonstrates the basic resync functionality of the Cisco IP Telephony devices.

TFTP Resync

The IP Telephony device supports multiple network protocols for retrieving configuration profiles. The most basic profile transfer protocol is TFTP (RFC1350). TFTP, widely used for the provisioning of network devices within private LAN networks. Although not recommended for the deployment of remote endpoints across the Internet, it can be convenient for deployment within small organizations, for in-house preprovisioning, and for development and testing. See the "In-House Device Preprovisioning" section on page 3-2 for more information on in-house preprovisioning. In this exercise, a profile is modified after downloading a file from a TFTP server.

Exercise

Step 1 Within a LAN environment connect a PC and an IP Telephony device to a hub, switch, or small router.

Step 2 Connect an analog phone to the Phone 1 port of the IP Telephony device.

Step 3 On the PC, install and activate a TFTP server.

Step 4 Using a text editor, create a configuration profile that sets the value for GPP_A to 12345678 as shown in the example.

```
<device> <flat-profile>
  <GPP_A> 12345678
  </GPP_A>
</flat-profile> </device>
```

Step 5 Save the profile with the name basic.txt in the root directory of the TFTP server.

You can verify that the TFTP server is properly configured by requesting the basic.txt file by using a TFTP client other than the IP Telephony device. Preferably, use a TFTP client that is running on a separate host from the provisioning server.

Step 6 Open the PC web browser on the admin/advanced configuration page. For example, if the IP address of the phone is 192.168.1.100:

```
http://192.168.1.100/admin/advanced
```

Step 7 Select the Provisioning tab, and inspect the values of the general purpose parameters GPP_A through GPP_P. These should be empty.

Step 8 Resync the test IP Telephony device to the basic.txt configuration profile by opening the resync URL in a web browser window.

Assuming the IP address of the TFTP server is 192.168.1.200 the command should be similar to this example:

```
http://192.168.1.100/admin/resync?tftp://192.168.1.200/basic.txt
```

When IP Telephony device receives this command, the device at address 192.168.1.100 requests the file basic.txt from the TFTP server at IP address 192.168.1.200. It then parses the downloaded file and updates the GPP_A parameter with the value 12345678.

Step 9 Verify that the parameter was correctly updated by refreshing the admin/advanced page on the PC web browser and selecting the Provisioning tab on that page.

The GPP_A parameter should now contain the value 12345678.

Logging with syslog

The IP Telephony device sends a syslog message to the designated syslog server when the device is about to resync to a provisioning server and after the resync has either completed or failed. This server is identified in the web server administration (admin/advanced, System tab, Syslog_Server parameter). Configure the syslog server IP address into the device and observe the messages generated during the remaining exercises.

Exercise

Step 1 Install and activate a syslog server on the local PC.

Step 2 Program the PC IP address into the Syslog_Server parameter of the profile and submit the change:

```
<Syslog_Server ua="na">192.168.1.210</Syslog_Server>
```

Step 3 Click the **System** tab and enter the value of your local syslog server into the Syslog_Server parameter.

Step 4 Repeat the resync operation as described in the TFTP Resync exercise.

The device generates two syslog messages during the resync. The first indicates that a request is in progress. The second marks success or failure of the resync.

Step 5 Verify that your syslog server received messages similar to the following:

```
CP-8831-3PCC 00:0e:08:ab:cd:ef -- Requesting resync tftp://192.168.1.200/basic.txt
CP-8831-3PCC 00:0e:08:ab:cd:ef -- Successful resync tftp://192.168.1.200/basic.txt
```

Detailed messages are available by programming a Debug_Server parameter (instead of the Syslog_Server parameter) with the IP address of the syslog server, and setting the Debug_Level to a value between 0 and 3 (3 being the most verbose):

```
<Debug_Server ua="na">192.168.1.210</Debug_Server>
<Debug_Level ua="na">3</Debug_Level>
```

The contents of these messages can be configured by using the following parameters:

- Log_Request_Msg
- Log_Success_Msg
- Log_Failure_Msg.

If any of these parameters are cleared, the corresponding syslog message is not generated.

Automatic Device Resync

A device can resync periodically to the provisioning server to ensure that any profile changes made on the server are propagated to the endpoint device (as opposed to sending an explicit resync request to the endpoint).

To cause the IP Telephony device to periodically resync to a server, a configuration profile URL is defined by using the Profile_Rule parameter, and a resync period is defined by using the Resync_Periodic parameter.

Exercise

Step 1 Using a web browser, open the admin/advanced page Provisioning tab.

Step 2 Define the Profile_Rule parameter. The example assumes a TFTP server IP address of 192.168.1.200:

```
<Profile_Rule ua="na">tftp://192.168.1.200/basic.txt</Profile_Rule>
```

Step 3 In the Resync_Periodic parameter enter a small value for testing, such as **30** seconds:

```
<Resync_Periodic ua="na">30</Resync_Periodic>
```

Step 4 Click **Submit all Changes**.

With the new parameter settings, the IP Telephony device resyncs to the configuration file specified by the URL twice a minute.

Step 5 Observe the resulting messages in the syslog trace (as described in the Logging with syslog section).

Step 6 Ensure that the Resync_On_Reset parameter is set to **yes**:

```
<Resync_On_Reset ua="na">Yes</Resync_On_Reset>
```

Step 7 Power cycle the IP Telephony device to force it to resync to the provisioning server.

If the resync operation fails for any reason, such as if the server is not responding, the unit waits the number of seconds configured in Resync_Error_Retry_Delay before attempting to resync again. If Resync_Error_Retry_Delay is zero, the IP Telephony device does not try to resync after a failed resync attempt.

Step 8 (Optional) Set the value of Resync_Error_Retry_Delay is set to a small number, such as **30**:

```
<Resync_Error_Retry_Delay ua="na">30</Resync_Error_Retry_Delay>
```

Step 9 Disable the TFTP server, and observe the results in the syslog output.

Unique Profiles, Macro Expansion, and HTTP

In a deployment where each IP Telephony device must be configured with distinct values for some parameters, such as User_ID or Display_Name, the service provider can create a unique profile for each deployed device and host those profiles on a provisioning server. Each IP Telephony device, in turn, must be configured to resync to its own profile according a predetermined profile naming convention.

The profile URL syntax can include identifying information specific to each IP Telephony device, such as MAC address or serial number, by using the macro expansion of built-in variables. Macro expansion eliminates the need to specify these values in multiple locations within each profile.

A profile rule undergoes macro expansion before being applied to the IP Telephony device. The macro expansion controls a number of values, for example:

- $MA expands to the unit 12-digit MAC address (using lower case hex digits). For example, 000e08abcdef.
- $SN expands to the unit serial number. For example, 88012BA01234.

Other values can be macro expanded in this way, including all the general purpose parameters, (GPP_A through GPP_P). An example of this process can be seen in the "TFTP Resync" section. Macro expansion is not limited to the URL file name, but can also be applied to any portion of the profile rule parameter. These parameters are referenced as $A through $P. For a complete list of variables available for macro expansion, see the "Macro Expansion Variables" section on page 5-5.

In this exercise, a profile specific to an IP Telephony device is provisioned on a TFTP server.

Exercise

Step 1 Obtain the MAC address of the IP Telephony device from its product label. (The MAC address is the number, using numbers and lower–case hex digits, such as 000e08aabbcc.

Step 2 Copy the `basic.txt` configuration file (described in the "TFTP Resync" exercise) to a new file named **`CP-8831-3PCC_`***`macaddress`***`.cfg`** (replacing *`macaddress`* with the MAC address of the IP Telephony device). For example:

```
CP-8831-3PCC_000e08abcdef.cfg
```

Step 3 Move the new file in the virtual root directory of the TFTP server.

Step 4 Open the admin/advanced page Provisioning tab.

Step 5 Enter `tftp://192.168.1.200/CP-8831-3PCC$MA.cfg` in the Profile_Rule parameter:

```
<Profile_Rule ua="na">
  tftp://192.168.1.200/CP-8831-3PCC$MA.cfg
</Profile_Rule>
```

Step 6 Click **Submit All Changes**. This causes an immediate reboot and resync.

When the next resync occurs, the IP Telephony device retrieves the new file by expanding the $MA macro expression into its MAC address.

HTTP GET Resync

HTTP provides a more reliable resync mechanism than TFTP because HTTP establishes a TCP connection and TFTP uses the less reliable UDP. In addition, HTTP servers offer improved filtering and logging features compared to TFTP servers.

On the client side, the IP Telephony device does not require any special configuration setting on the server to be able to resync by using HTTP. The Profile_Rule parameter syntax for using HTTP with the GET method is similar to the syntax used for TFTP. If a standard web browser can retrieve a profile from a your HTTP server, the IP Telephony device should be able to do so as well.

Exercise

Step 1 Install an HTTP server on the local PC or other accessible host. (The open source Apache server can be downloaded from the Internet.)

Step 2 Copy the `basic.txt` configuration profile (described in the TFTP Resync exercise) onto the virtual root directory of the installed server.

Step 3 Verify proper server installation (and file access to basic.txt) by accessing the profile by using a web browser.

Step 4 Modify the Profile_Rule of the test IP Telephony device to point to the HTTP server in place of the TFTP server, so as to download its profile periodically.

For example, assuming the HTTP server is at 192.168.1.300, enter the following value:

```
<Profile_Rule ua="na">
http://192.168.1.200/basic.txt
</Profile_Rule>
```

Step 5 Click **Submit All Changes**. This causes an immediate reboot and resync.

Step 6 Observe the syslog messages sent by the IP Telephony device. The periodic resyncs should now be obtaining the profile from the HTTP server.

Step 7 In the HTTP server logs, observe how information identifying the test IP Telephony device appears in the log of user agents.

This should include the manufacturer, product name, current firmware version, and serial number.

Provisioning Through Cisco XML

The CP-8831-3PCC extends Cisco XML feature to support provisioning via XML object:

```
<CP-8831-3PCCExecute>
        <ExecuteItem URL=Resync:[profile-rule]/>
</CP-8831-3PCCExecute>
```

After receiving the XML object, the CP-8831-3PCC downloads the provisioning file from [profile-rule]. Macros are used in this rule to simplify the development of XML services application.

URL Resolution by Using Macro Expansion

Subdirectories with multiple profiles on the server is a convenient method for managing a large number of deployed devices. The profile URL can contain:

- A provisioning server name or an explicit IP address. If the profile identifies the provisioning server by name, the IP Telephony device performs a DNS lookup to resolve the name.
- A non-standard server port specified in the URL by using the standard syntax`:`*`port`* following the server name.
- The subdirectory of the server virtual root directory where the profile is stored, specified by using standard URL notation and managed by macro expansion.

For example, the following Profile_Rule requests the profile spa962.cfg, in the server subdirectory /cisco/config, from the TFTP server running on host prov.telco.com listening for a connection on port 6900:

```
<Profile_Rule ua="na">
/tftp://prov.telco.com:6900/cisco/config/CP-8831-3PCC.cfg
</Profile_Rule>
```

A profile for each IP Telephony device can be identified in a general purpose parameter, with its value referred within a common profile rule by using macro expansion.

For example, assume GPP_B is defined as `Dj6Lmp23Q`.

The Profile_Rule has the value:

```
tftp://prov.telco.com/cisco/$B/$MA.cfg
```

When the device resyncs and the macros are expanded, the IP Telephony device with a MAC address of 000e08012345 requests the profile with the name that contains the device MAC address at the following URL:

```
tftp://prov.telco.com/cisco/Dj6Lmp23Q/000e08012345.cfg
```

Secure HTTPS Resync

This section demonstrates the mechanisms available on the IP Telephony device for resyncing by using a secure communication process. It includes the following topics:

- Basic HTTPS Resync, page 4-6
- HTTPS With Client Certificate Authentication, page 4-8
- HTTPS Client Filtering and Dynamic Content, page 4-9

Basic HTTPS Resync

HTTPS adds SSL to HTTP for remote provisioning so that the:

- IP Telephony device can authenticate the provisioning server
- Provisioning server can authenticate the IP Telephony device
- Confidentiality of information exchanged between the IP Telephony device and the provisioning server is ensured.

SSL generates and exchanges secret (symmetric) keys for each connection between the IP Telephony device and the server, using public/private key pairs preinstalled in the IP Telephony device and the provisioning server.

On the client side, the IP Telephony device does not require any special configuration setting on the server to be able to resync using HTTPS. The Profile_Rule parameter syntax for using HTTPS with the GET method is similar to the syntax used for HTTP or TFTP. If a standard web browser can retrieve a profile from a your HTTPS server, the IP Telephony device should be able to do so as well.

In addition to installing a HTTPS server, a SSL server certificate signed by Cisco must be installed on the provisioning server. The devices cannot resync to a server using HTTPS unless the server supplies a Cisco-signed server certificate. Instructions for creating signed SSL Certificates for Voice products can be found at https://supportforums.cisco.com/docs/DOC-9852.

Exercise

Step 1 Install an HTTPS server on a host whose IP address is known to the network DNS server through normal hostname translation.

The open source Apache server can be configured to operate as an HTTPS server when installed with the open source mod_ssl package.

Step 2 Generate a server Certificate Signing Request for the server. For this step, you might need to install the open source OpenSSL package or equivalent software. If using OpenSSL, the command to generate the basic CSR file is as follows:

```
openssl req –new -out provserver.csr
```

This command generates a public/private key pair, which is saved in the privkey.pem file.

Step 3 Submit the CSR file (provserver.csr) to Cisco for signing. (See https://supportforums.cisco.com/docs/DOC-9852 for more information.) A signed server certificate is returned (provserver.cert) along with a Sipura CA Client Root Certificate, spacroot.cert.

Step 4 Store the signed server certificate, the private key pair file, and the client root certificate in the appropriate locations on the server.

In the case of an Apache installation on Linux, these locations are typically as follows:

```
# Server Certificate:
SSLCertificateFile /etc/httpd/conf/provserver.cert
# Server Private Key:
SSLCertificateKeyFile /etc/httpd/conf/pivkey.pem
# Certificate Authority:
SSLCACertificateFile /etc/httpd/conf/spacroot.cert
```

Step 5 Restart the server.

Step 6 Copy the `basic.txt` configuration file (described in the "TFTP Resync" exercise) onto the virtual root directory of the HTTPS server.

Step 7 Verify proper server operation by downloading basic.txt from the HTTPS server by using a standard browser from the local PC.

Step 8 Inspect the server certificate supplied by the server.

The browser probably does not recognize it as valid unless the browser has been preconfigured to accept Cisco as a root CA. However, the IP Telephony devices expect the certificate to be signed this way.

Modify the Profile_Rule of the test device to contain a reference to the HTTPS server, for example:

```
<Profile_Rule ua="na">
https://my.server.com/basic.txt
</Profile_Rule>
```

This example assumes the name of the HTTPS server is `my.server.com`.

Step 9 Click **Submit All Changes**.

Step 10 Observe the syslog trace sent by the IP Telephony device.

The syslog message should indicate that the resync obtained the profile from the HTTPS server.

Step 11 (Optional) Use an Ethernet protocol analyzer on the IP Telephony device subnet to verify that the packets are encrypted.

In this exercise, client certificate verification was not enabled. The connection between IP Telephony device and server is encrypted. However, the transfer is not secure because any client can connect to the server and request the file, given knowledge of the file name and directory location. For secure resync, the server must also authenticate the client, as demonstrated in the exercise described in the "HTTPS With Client Certificate Authentication" section.

HTTPS With Client Certificate Authentication

In the factory default configuration, the server does not request a SSL client certificate from a client. Transfer of the profile is not secure because any client can connect to the server and request the profile. You can edit the configuration to enable client authentication; the server requires a client certificate to authenticate the IP Telephony device before accepting a connection request.

Because of this, the resync operation cannot be independently tested by using a browser lacking the proper credentials. The SSL key exchange within the HTTPS connection between the test IP Telephony device and the server can be observed using the ssldump utility. The utility trace shows the interaction between client and server.

Exercise

Step 1 Enable client certificate authentication on the HTTPS server.

Step 2 In Apache (v.2), set the following in the server configuration file:

```
SSLVerifyClient  require
```

Also ensure that the spacroot.cert has been stored as shown in the "Basic HTTPS Resync" exercise.

Step 3 Restart the HTTPS server and observe the syslog trace from the IP Telephony device.

Each resync to the server now performs symmetric authentication, so that both the server certificate and the client certificate are verified before the profile is transferred.

Step 4 Use ssldump to capture a resync connection between the IP Telephony device and the HTTPS server.

If client certificate verification is properly enabled on the server, the ssldump trace shows the symmetric exchange of certificates (first server-to-client, then client-to-server) before the encrypted packets containing the profile.

With client authentication enabled, only an IP Telephony device with a MAC address matching a valid client certificate can request the profile from the provisioning server. A request from an ordinary browser or other unauthorized device is rejected by the server.

HTTPS Client Filtering and Dynamic Content

If the HTTPS server is configured to require a client certificate, then the information in the certificate identifies the resyncing IP Telephony device and supplies it with the correct configuration information.

The HTTPS server makes the certificate information available to CGI scripts (or compiled CGI programs) invoked as part of the resync request. For the purpose of illustration, this exercise uses the open source Perl scripting language, and assumes that Apache (v.2) is used as the HTTPS server.

Exercise

Step 1 Install Perl on the host running the HTTPS server.

Step 2 Generate the following Perl reflector script:

```
#!/usr/bin/perl -wT
use strict;
print "Content-Type: text/plain\n\n";
print "<flat-profile><GPP_D>";

print "OU=$ENV{'SSL_CLIENT_I_DN_OU'},\n";
print "L=$ENV{'SSL_CLIENT_I_DN_L'},\n";
print "S=$ENV{'SSL_CLIENT_I_DN_S'}\n";
print "</GPP_D></flat-profile>";
```

Step 3 Save this file with the file name reflect.pl, with executable permission (chmod 755 on Linux), in the CGI scripts directory of the HTTPS server.

Step 4 Verify accessibility of CGI scripts on the server (as in /cgi-bin/…).

Step 5 Modify the Profile_Rule on the test device to resync to the reflector script, as in the following example:

```
https://prov.server.com/cgi-bin/reflect.pl?
```

Step 6 Click **Submit All Changes**.

Step 7 Observe the syslog trace to ensure a successful resync.

Step 8 Open the admin/advanced page, Provisioning tab.

Step 9 Verify that the GPP_D parameter contains the information captured by the script.

This information contains the product name, MAC address, and serial number if the test device carries a unique certificate from the manufacturer, or else generic strings if it is a unit manufactured before firmware release 2.0.

A similar script could be used to determine information about the resyncing device and then provide it with appropriate configuration parameter values.

HTTPS Certificates

The IP Telephony device provides a reliable and secure provisioning strategy based on HTTPS requests from the device to the provisioning server. Both a server certificate and a client certificate are used to authenticate the IP Telephony device to the server and the server to the IP Telephony device.

To use HTTPS, you must generate a Certificate Signing Request (CSR) and submit it to Cisco. Cisco generates a certificate for installation on the provisioning server. The IP Telephony device accepts the certificate when it seeks to establish an HTTPS connection with the provisioning server.

How HTTPS Works

HTTPS encrypts the communication between a client and a server, protecting the message contents from other network devices. The encryption method for the body of the communication between a client and a server is based on symmetric key cryptography. With symmetric key cryptography, a single secret key is shared by a client and a server over a secure channel protected by Public/Private key encryption.

Messages encrypted by the secret key can only be decrypted using the same key. HTTPS supports a wide range of symmetric encryption algorithms. The IP Telephony device implements up to 256-bit symmetric encryption, using the American Encryption Standard (AES), in addition to 128-bit RC4.

HTTPS also provides for the authentication of a server and a client engaged in a secure transaction. This feature ensures that a provisioning server and an individual client cannot be spoofed by other devices on the network. This is an essential capability in the context of remote endpoint provisioning.

Server and client authentication is performed by using public/private key encryption with a certificate that contains the public key. Text that is encrypted with a public key can be decrypted only by its corresponding private key (and vice versa). The IP Telephony device supports the RSA algorithm for public/private key cryptography.

SSL Server Certificates

Each secure provisioning server is issued a SSL server certificate, directly signed by Cisco. The firmware running on the IP Telephony device recognizes only a Cisco certificate as valid. When a client connects to a server by using HTTPS, it rejects any server certificate that is not signed by Cisco.

This mechanism protects the service provider from unauthorized access to the IP Telephony device, or any attempt to spoof the provisioning server. Without such protection, an attacker might be able to reprovision the IP Telephony device, to gain configuration information, or to use a different VoIP service.

Client Certificates

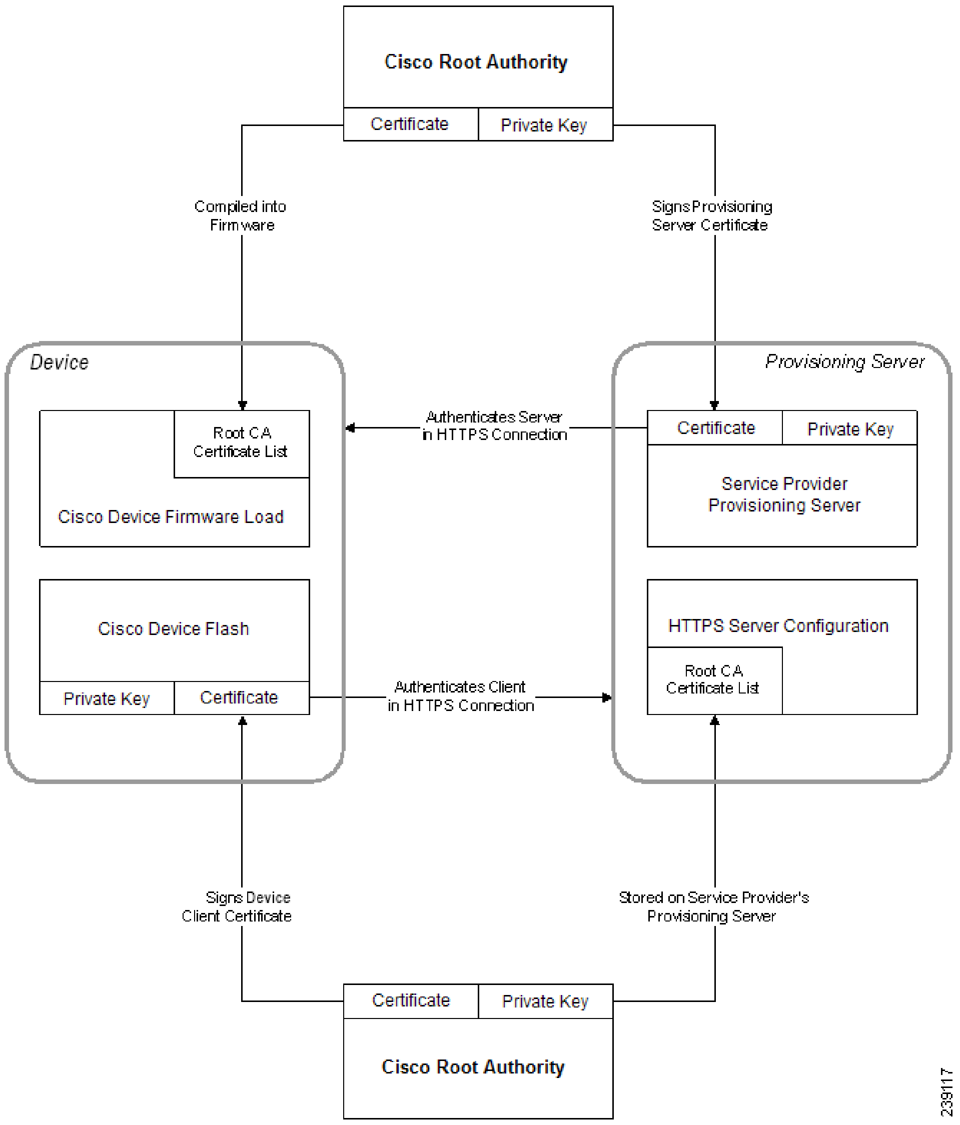
In addition to a direct attack on an IP Telephony device, an attacker might attempt to contact a provisioning server by using a standard web browser or another HTTPS client to obtain the configuration profile from the provisioning server. To prevent this kind of attack, each IP Telephony device also carries a unique client certificate, signed by Cisco, including identifying information about each individual endpoint. A certificate authority root certificate capable of authenticating the device client certificate is given to each service provider. This authentication path allows the provisioning server to reject unauthorized requests for configuration profiles.

Certificate Structure

The combination of a server certificate and a client certificate ensures secure communication between a remote IP Telephony device and its provisioning server. Figure 4-1 illustrates the relationship and placement of certificates, public/private key pairs, and signing root authorities, among the Cisco client, the provisioning server, and the certification authority.

The upper half of the diagram shows the Provisioning Server Root Authority that is used to sign the individual provisioning server certificate. The corresponding root certificate is compiled into the firmware, allowing the IP Telephony device to authenticate authorized provisioning servers.

Figure 4-1 Certificate Authority Flow


Cisco Root Authority
Certificate
Private Key
Compiled into Firmware
Signs Provisioning Server Certificate
Device
Root CA Certificate List
Cisco Device Firmware Load
Authenticates Server in HTTPS Connection
Provisioning Server
Certificate
Private Key
Service Provider Provisioning Server
Cisco Device Flash
Private Key
Certificate
Authenticates Client in HTTPS Connection
HTTPS Server Configuration
Root CA Certificate List
Signs Device Client Certificate
Stored on Service Provider's Provisioning Server
Certificate
Private Key
Cisco Root Authority
239117

Profile Management

This section demonstrates the formation of configuration profiles in preparation for downloading. To explain the functionality, TFTP from a local PC is used as the resync method, although HTTP or HTTPS can be used as well.

Open Profile gzip Compression

A configuration profile in XML format can become quite large if all parameters are individually specified by the profile. To reduce the load on the provisioning server, the IP Telephony device supports compression of the XML file, by using the deflate compression format supported by the gzip utility (RFC 1951).

Note Compression must precede encryption for the IP Telephony device to recognize a compressed and encrypted XML profile.

For integration into customized back-end provisioning server solutions, the open source zlib compression library can be used in place of the standalone gzip utility to perform the profile compression. However, the IP Telephony device expects the file to contain a valid gzip header.

Additional information on compression is provided in the "Open Profile Compression" section on page 2-5.

Exercise

Step 1 Install gzip on the local PC.

Step 2 Compress the `basic.txt` configuration profile (described in the "TFTP Resync" exercise) by invoking gzip from the command line:

```
gzip basic.txt
```

This generates the deflated file `basic.txt.gz`.

Step 3 Save the `basic.txt.gz` file in the TFTP server virtual root directory.

Step 4 Modify the Profile_Rule on the test device to resync to the deflated file in place of the original XML file, as shown in the following example:

```
tftp://192.168.1.200/basic.txt.gz
```

Step 5 Click **Submit All Changes**.

Step 6 Observe the syslog trace from the IP Telephony device.

Upon resync, the new file is downloaded by the IP Telephony device and used to update its parameters.

Profile Encryption by Using OpenSSL

A compressed or uncompressed profile can be encrypted (however, a file must be compressed before it is encrypted). This is useful when the confidentiality of the profile information is of particular concern, such as when using TFTP or HTTP for communication between the IP Telephony device and the provisioning server.

The IP Telephony device supports symmetric key encryption by using the 256-bit AES algorithm. This encryption can be performed by using the open source OpenSSL package. Additional information on encryption is provided in Open Profile Encryption by Using AES, page 2-6.

Exercise

Step 1 Install OpenSSL on a local PC. This might require that the OpenSSL application be recompiled to enable AES.

Step 2 Using the `basic.txt` configuration file (described in the TFTP Resync exercise), generate an encrypted file with the following command:

```
>openssl enc -aes-256-cbc -k MyOwnSecret -in basic.txt -out basic.cfg
```

The compressed basic.txt.gz file created in Open Profile gzip Compression also can be used, because the XML profile can be both compressed and encrypted.

Step 3 Store the encrypted basic.cfg file in the TFTP server virtual root directory.

Step 4 Modify the Profile_Rule on the test device to resync to the encrypted file in place of the original XML file. The encryption key is made known to the IP Telephony device with the following URL option:

```
[--key MyOwnSecret ] tftp://192.168.1.200/basic.cfg
```

Step 5 Click **Submit All Changes**.

Step 6 Observe the syslog trace from the IP Telephony device.

On resync, the new file is downloaded by the IP Telephony device and is used to update its parameters.

Partitioned Profiles

An IP Telephony device downloads multiple separate profiles during each resync. This allows managing different kinds of profile information on separate servers and maintaining common configuration parameter values separate from account specific values.

Exercise

Step 1 Create a new XML profile, basic2.txt, that specifies a value for a parameter that makes it distinct from the earlier exercises. For instance, to the basic.txt profile you can add the following:

```
<GPP_B>ABCD</GPP_B>
```

Step 2 Store the basic2.txt profile in the virtual root directory of the TFTP server.

Step 3 Leave the first profile rule from the earlier exercises in the folder, but configure the second profile rule (Profile_Rule_B) to point to the new file:

```
<Profile_Rule_B ua="na">tftp://192.168.1.200/basic2.txt
</Profile_Rule_B>
```

Step 4 Click **Submit All Changes**.

The IP Telephony device now resyncs to both the first and second profiles, in that order, whenever a resync operation is due.

Step 5 Observe the syslog trace to confirm the expected behavior.

Provisioning Parameters

This chapter describes the provisioning parameters that can be used in configuration profile scripts. It includes the following sections:

Configuration Profile Parameters

The following table defines the function and usage of each parameter in the Configuration Profile Parameters section under the Provisioning tab.

Parameter Name	Description and Default Value
Provision_Enable	Controls all resync actions independently of firmware upgrade actions. Set to Yes to enable remote provisioning. The default value is Yes.
Resync_On_Reset	Triggers a resync after every reboot except for reboots caused by parameter updates and firmware upgrades. The default value is Yes.
Resync_At	The hour and minutes (HHmm) that the device resynchronizes with the provisioning server. The default value is empty. If the value is invalid, the parameter is ignored. If this parameter is set with a valid value, the Resync_Periodic parameter is ignored.

Parameter Name	Description and Default Value
Resync_At_Random_Delay	Prevents an overload of the provisioning server when a large number of devices power-on simultaneously. To avoid flooding resync requests to the server from multiple phones, the phone resynchronizes in the range between the hours and minutes, and the hours and minutes plus the random delay (hhmm, hhmm+random_delay). For example, if the random delay = (Resync_At_Random_Delay + 30)/60 minutes. The input value in seconds is converted to minutes, rounding up to the next minute to calculate the final random_delay interval. This feature is disabled when this parameter is set to zero. The default value is 600 seconds (10 minutes). If the parameter value is set to less than 600, the default value is used.
Resync_Periodic	The time interval between periodic resynchronizes with the provisioning server. The associated resync timer is active only after the first successful sync with the server. Set this parameter to zero to disable periodic resynchronization. The default value is 3600 seconds.
Forced_Resync_Delay	Maximum delay (in seconds) the IP Telephony deviceATA waits before performing a resynchronization. The device does not resync while one of its phone lines is active. Because a resync can take several seconds, it is desirable to wait until the device has been idle for an extended period before resynchronizing. This allows a user to make calls in succession without interruption. The device has a timer that begins counting down when all of its lines become idle. This parameter is the initial value of the counter. Resync events are delayed until this counter decrements to zero. The default value is 14,400 seconds.

Parameter Name	**Description and Default Value**
Resync_From_SIP	Enables a resync to be triggered via a SIP NOTIFY message. The default value is Yes.
Resync_After_Upgrade_Attempt	Triggers a resync after every firmware upgrade attempt. The default value is Yes.
Resync_Trigger_1, Resync_Trigger_2	Configurable resync trigger conditions. A resync is triggered when the logic equation in these parameters evaluates to TRUE. The default value is (empty).
Log_Request_Msg	This parameter contains the message that is sent to the syslog server at the start of a resync attempt. The default value is $PN $MAC – Requesting resync $SCHEME://$SERVIP:$PORT$PATH.
Log_Success_Msg	The syslog message that is issued upon successful completion of a resync attempt. The default value is $PN $MAC – Successful resync $SCHEME://$SERVIP:$PORT$PATH -- $ERR.
Log_Failure_Msg	The syslog message that is issued after a failed resync attempt. The default value is $PN $MAC – Resync failed: $ERR.

Firmware Upgrade Parameters

The following table defines the function and usage of each parameter in the Firmware Upgrade section of the Provisioning tab.

Parameter Name	**Description and Default Value**
Upgrade_Enable	Enables firmware upgrade operations independently of resync actions. The default value is Yes.
Upgrade_Error_Retry_Delay	The upgrade retry interval (in seconds) applied in case of upgrade failure. The device has a firmware upgrade error timer that activates after a failed firmware upgrade attempt. The timer is initialized with the value in this parameter. The next firmware upgrade attempt occurs when this timer counts down to zero. The default value is 3600 seconds.
Upgrade_Rule	This parameter is a firmware upgrade script with the same syntax as Profile_Rule. Defines upgrade conditions and associated firmware URLs. The default value is (empty).
Log_Request_Msg	The syslog message that is issued at the start of a firmware upgrade attempt. The default value is $PN $MAC -- Requesting upgrade $SCHEME://$SERVIP:$PORT$PATH.
Log_Success_Msg	The syslog message that is issued after a firmware upgrade attempt completes successfully. The default value is $PN $MAC -- Successful upgrade $SCHEME://$SERVIP:$PORT$PATH -- $ERR.
Log_Failure_Msg	The syslog message that is issued after a failed firmware upgrade attempt. The default value is $PN $MAC -- Upgrade failed: $ERR.

General Purpose Parameters

The following table defines the function and usage of each parameter in the General Purpose Parameters section of the Provisioning tab.

Parameter Name	Description and Default Value
GPP_SA, GPP_SB, GPP_SC, GPP_SD	Special purpose provisioning parameters, designed to hold encryption keys and passwords. To ensure the integrity of the encryption mechanism, these parameters must be kept secret. Therefore these parameters are not displayed on the device configuration web page, and they are not included in the configuration report sent in response to a SIP NOTIFY command. Note that these parameters are not available on the SPA500 Series phones. The default value is (empty).
GPP_A through GPP_P	General purpose provisioning parameters. These parameters can be used as variables in provisioning and upgrade rules. They are referenced by prepending the variable name with a '$' character, such as $GPP_A. The default value is (empty).

Macro Expansion Variables

Certain macro variables are recognized within the following provisioning parameters:

- Profile_Rule
- Profile_Rule_*
- Resync_Trigger_*
- Upgrade_Rule
- Log_*
- GPP_* (under specific conditions)

Within these parameters, syntax types, such as $NAME or $(NAME), are recognized and expanded.

Macro variable substrings can be specified with the notation $(NAME:p) and $(NAME:p:q), where p and q are non-negative integers (available in revision 2.0.11 and above). The resulting macro expansion is the substring starting at character offset p, with length q (or else till end-of-string if q is not specified). For example, if GPP_A contains ABCDEF, then $(A:2) expands to CDEF, and $(A:2:3) expands to CDE.

An unrecognized name is not translated, and the $NAME or $(NAME) form remains unchanged in the parameter value after expansion.

Parameter Name	Description and Default Value
$	The form $$ expands to a single $ character.
A through P	Replaced by the contents of the general purpose parameters GPP_A through GPP_P.
MA	MAC address using lower case hex digits, for example, 000e08aabbcc.
MAU	MAC address using upper case hex digits, for example 000E08AABBCC.
MAC	MAC address using lower case hex digits, and colons to separate hex digit pairs, for example 00:0e:08:aa:bb:cc.
PN	Product Name, for example CP-8831-3PCC.
PSN	Product Series Number, for example 514.
SN	Serial Number string, for example 88012BA01234.
CCERT	SSL Client Certificate status: Installed or Not Installed.
IP	IP address of the IP Telephony device within its local subnet, for example 192.168.1.100.
EXTIP	External IP of the IP Telephony device, as seen on the Internet, for example 66.43.16.52.
SWVER	Software version string, for example 7.5.1(a).
HWVER	Hardware version string, for example 2.0.1
SCHEME	File access scheme, one of TFTP, HTTP, or HTTPS, as obtained after parsing resync or upgrade URL.
SERV	Request target server host name, as obtained after parsing resync or upgrade URL.
SERVIP	Request target server IP address, as obtained after parsing resync or upgrade URL, possibly following DNS lookup.
PORT	Request target UDP/TCP port, as obtained after parsing resync or upgrade URL.
PATH	Request target file path, as obtained after parsing resync or upgrade URL.
ERR	Result message of resync or upgrade attempt. Only useful in generating result syslog messages. The value is preserved in the UPGERR variable in the case of upgrade attempts.
ISCUST	Value=1 if unit is customized, 0 otherwise; customization status viewable on WebUI Info page.
SA through SD	Replaced by the contents of the parameters GPP_SA through GPP_SD, which are for secure key string.

Internal Error Codes

The IP Telephony device defines a number of internal error codes (X00–X99) to facilitate configuration in providing finer control over the behavior of the unit under certain error conditions.

Parameter Name	Description and Default Value
X00	Transport layer (or ICMP) error when sending a SIP request.
X20	SIP request times out while waiting for a response.
X40	General SIP protocol error (for example, unacceptable codec in SDP in 200 and ACK messages, or times out while waiting for ACK).
X60	Dialed number invalid according to given dial plan.

Sample Configuration Profiles

XML Open Format Sample

```
<?xml version="1.0" encoding="UTF-8"?>
<device xsi:type="axl:XIPPhone" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <flat-profile>



  <!-- System Configuration -->

  <Restricted_Access_Domains ua="na"/>
  <Enable_Web_Server ua="na">Yes</Enable_Web_Server>
  <Web_Server_Port ua="na">80</Web_Server_Port>
  <Enable_Web_Admin_Access ua="na">Yes</Enable_Web_Admin_Access>
  <Admin_Password ua="na"/>
  <User_Password ua="rw"/>
  <Phone-UI-user-mode ua="na">No</Phone-UI-user-mode>


  <!-- Internet Connection Type  -->

  <Connection_Type ua="rw">DHCP</Connection_Type>


  <!--  Static IP Settings -->

  <Static_IP ua="rw"/>
  <NetMask ua="rw"/>
  <Gateway ua="rw"/>


  <!-- Optional Network Configuration -->

  <HostName ua="rw"/>
  <Domain ua="rw"/>
  <Primary_DNS ua="rw"/>
  <Secondary_DNS ua="rw"/>
  <Syslog_Server ua="na"/>
  <Debug_Level ua="na"/>0</Debug_Level>
  <Layer_2_Logging ua="na">No</Layer_2_Logging>
  <Primary_NTP_Server ua="na">us.pool.ntp.org</Primary_NTP_Server>
  <Secondary_NTP_Server ua="na"/>
  <SSH_Access ua="na">No</SSH_Access>
  <DNS_Cache_TTL_Ignore ua="na">Yes</DNS_Cache_TTL_Ignore>
  <SSH_User_ID ua="na"/>
```

```
<SSH_Password ua="na"/>


<!-- VLAN Settings -->

<Enable_CDP ua="na">Yes</Enable_CDP>
<Enable_LLDP-MED ua="na">Yes</Enable_LLDP-MED>
<Network_Startup_Delay ua="na">3</Network_Startup_Delay>
<VLAN_ID ua="rw">4095</VLAN_ID>


<!-- Inventory Settings -->

<Asset_ID ua="na"/>


<!--  SIP Parameters   -->

<Max_Forward ua="na">70</Max_Forward>
<Max_Redirection ua="na">5</Max_Redirection>
<SIP_User_Agent_Name ua="na">$VERSION</SIP_User_Agent_Name>
<SIP_Server_Name ua="na">$VERSION</SIP_Server_Name>
<SIP_Reg_User_Agent_Name ua="na" />
<SIP_Accept_Language ua="na" />
<RFC_2543_Call_Hold ua="na">No</RFC_2543_Call_Hold>
<SIP_TCP_Port_Min ua="na">5060</SIP_TCP_Port_Min>
<SIP_TCP_Port_Max ua="na">5080</SIP_TCP_Port_Max>
<Caller_ID_Header ua="na">PAID-RPID-FROM</Caller_ID_Header>
<Max_INVITE_Retry_Attempts ua="na">3</Max_INVITE_Retry_Attempts>
<Max_NON-INVITE_Retry_Attempts ua="na">3</Max_NON-INVITE_Retry_Attempts>

<!--  SIP Timer Values   -->

<SIP_T1 ua="na">0.5</SIP_T1>
<SIP_T2 ua="na">4</SIP_T2>
<INVITE_Expires ua="na">240</INVITE_Expires>
<ReINVITE_Expires ua="na">30</ReINVITE_Expires>
<Reg_Retry_Intvl ua="na">30</Reg_Retry_Intvl>
<Reg_Retry_Long_Intvl ua="na">1200</Reg_Retry_Long_Intvl>
<Reg_Retry_Random_Delay ua="na">0</Reg_Retry_Random_Delay>
<Reg_Retry_Long_Random_Delay ua="na">0</Reg_Retry_Long_Random_Delay>
<Reg_Retry_Intvl_Cap ua="na">0</Reg_Retry_Intvl_Cap>


<!--  Response Status Code Handling  -->

<Try_Backup_RSC ua="na">5??, 6??</Try_Backup_RSC>
<Retry_Reg_RSC ua="na">5??, 6??</Retry_Reg_RSC>


<!--  RTP Parameters   -->

<RTP_Port_Min ua="na">16384</RTP_Port_Min>
<RTP_Port_Max ua="na">16538</RTP_Port_Max>
<RTP_Packet_Size ua="na">0.02</RTP_Packet_Size>
<RTCP_Tx_Enable ua="na">No</RTCP_Tx_Enable>


<!--  SDP Payload Types   -->

<AVT_Dynamic_Payload ua="na">101</AVT_Dynamic_Payload>


<!--  NAT Support Parameters  -->
```

```
  <NAT_Keep_Alive_Intvl ua="na">15</NAT_Keep_Alive_Intvl>


<!--  Configuration Profile  -->

   <Provision_Enable ua="na">Yes</Provision_Enable>
   <Resync_On_Reset ua="na">Yes</Resync_On_Reset>
   <Resync_Random_Delay ua="na">2</Resync_Random_Delay>
   <Resync_At__HHmm_ ua="na">0100</Resync_At__HHmm_>
   <Resync_At_Random_Delay ua="na">600</Resync_At_Random_Delay>
   <Resync_Periodic ua="na">3600</Resync_Periodic>
   <Resync_Error_Retry_Delay ua="na">3600</Resync_Error_Retry_Delay>
   <Forced_Resync_Delay ua="na">14400</Forced_Resync_Delay>
   <Resync_Fails_On_FNF ua="na">Yes</Resync_Fails_On_FNF>
   <Profile_Rule ua="na">http://10.74.10.225/yxue2/ut/8831System_wSBC.xml</Profile_Rule>
   <Profile_Rule_B ua="na" />
   <Profile_Rule_C ua="na">/ut/Register2.xml</Profile_Rule_C>
   <Profile_Rule_D ua="na" />
   <Resync_DHCP_Option_To_Use ua="na">160,159,66,150</Resync_DHCP_Option_To_Use>
   <Log_Request_Msg ua="na">$PN $MAC -- Requesting %s
$SCHEME://$SERVIP:$PORT$PATH</Log_Request_Msg>
   <Log_Success_Msg ua="na">$PN $MAC -- Successful %s $SCHEME://$SERVIP:$PORT$PATH --
$ERR</Log_Success_Msg>
   <Log_Failure_Msg ua="na">$PN $MAC -- %s failed: $ERR</Log_Failure_Msg>
   <User_Configurable_Resync ua="na">Yes</User_Configurable_Resync>
   <!--  Firmware Upgrade   -->

   <Upgrade_Enable ua="na">Yes</Upgrade_Enable>
   <Upgrade_Error_Retry_Delay ua="na">3600</Upgrade_Error_Retry_Delay>
   <Upgrade_Rule ua="na" />
   <Enable_Enterprise_Image_Upgrade ua="na">No</Enable_Enterprise_Image_Upgrade>


   <!--  CA Settings   -->

   <Custom_CA_Rule ua="na" />


   <!--  General Purpose Parameters  -->
   <GPP_A ua="na" />
   <GPP_B ua="na" />
   <GPP_C ua="na" />
   <GPP_D ua="na" />
   <GPP_E ua="na" />
   <GPP_F ua="na" />
   <GPP_G ua="na" />
   <GPP_H ua="na" />
   <GPP_I ua="na" />
   <GPP_J ua="na" />
   <GPP_K ua="na" />
   <GPP_L ua="na" />
   <GPP_M ua="na" />
   <GPP_N ua="na" />
   <GPP_O ua="na" />
   <GPP_P ua="na" />


   <!--  Control Timer Values (sec)  -->

   <Interdigit_Long_Timer ua="na">10</Interdigit_Long_Timer>
   <Interdigit_Short_Timer ua="na">3</Interdigit_Short_Timer>
  <!--  Time  -->

  <Time_Zone ua="na">GMT-8:00</Time_Zone>
  <Time_Offset__HH_mm_ ua="na">00/00</Time_Offset__HH_mm_>
  <Ignore_DHCP_Time_Offset ua="na">No</Ignore_DHCP_Time_Offset>
  <Daylight_Saving_Time_Rule
ua="na">start=3/-1/7/2;end=10/-1/7/2;save=1</Daylight_Saving_Time_Rule>
  <Daylight_Saving_Time_Enable ua="na">Yes</Daylight_Saving_Time_Enable>


  <!--  Localization  -->

  <Dictionary_Server_Script ua="na" />
  <Language_Selection ua="na" />
  <Locale ua="na">en-US</Locale>

- <!--  QoS Settings  -->

  <SIP_TOS_Value ua="na">0x60</SIP_TOS_Value>
  <RTP_TOS_Value ua="na">0xb8</RTP_TOS_Value>


  <!--  General  -->

  <Station_Display_Name ua="na" />
  <Text_Logo ua="na" />
  <PNG_Picture_Download_URL ua="na" />
  <Select_Logo ua="na">Default</Select_Logo>
  <Select_Background_Picture ua="na">None</Select_Background_Picture>
  <Screen_Saver_Enable ua="rw">No</Screen_Saver_Enable>
  <Screen_Saver_Wait ua="rw">300</Screen_Saver_Wait>
  <Screen_Saver_Icon ua="rw">Background Picture</Screen_Saver_Icon>
  <Co-branding_Banner_Picture_Download_URL ua="na" />


  <!--  Miscellaneous Line Key Settings  -->

  <Call_Appearances_Per_Line ua="na">2</Call_Appearances_Per_Line>


  <!--  Supplementary Services  -->

  <Conference_Serv ua="na">Yes</Conference_Serv>
  <Attn_Transfer_Serv ua="na">Yes</Attn_Transfer_Serv>
  <Blind_Transfer_Serv ua="na">Yes</Blind_Transfer_Serv>
  <Cfwd_All_Serv ua="na">Yes</Cfwd_All_Serv>
  <Cfwd_Busy_Serv ua="na">Yes</Cfwd_Busy_Serv>
  <Cfwd_No_Ans_Serv ua="na">Yes</Cfwd_No_Ans_Serv>


  <!--  Broadsoft Settings  -->

  <Directory_Enable ua="na">Yes</Directory_Enable>
  <XSI_Host_Server ua="na">xsi.iop1.broadworks.net</XSI_Host_Server>
  <Directory_Name ua="na">IOP1</Directory_Name>
  <Directory_Type ua="na">Enterprise</Directory_Type>
  <Directory_User_ID ua="na">broadsoft_user</Directory_User_ID>
  <Directory_Password ua="na" />


<!--  LDAP Corporate Directory Search  -->

  <LDAP_Dir_Enable ua="na">No</LDAP_Dir_Enable>
  <LDAP_Corp_Dir_Name ua="na" />
```

```
<LDAP_Server ua="na" />
<LDAP_Auth_Method ua="na">None</LDAP_Auth_Method>
<LDAP_Client_DN ua="na" />
<LDAP_Username ua="na" />
<LDAP_Password ua="na" />
<LDAP_Search_Base ua="na" />
<LDAP_Last_Name_Filter ua="na" />
<LDAP_First_Name_Filter ua="na" />
<LDAP_Search_Item_3 ua="na" />
<LDAP_Item_3_Filter ua="na" />
<LDAP_Search_Item_4 ua="na" />
<LDAP_Item_4_Filter ua="na" />
<LDAP_Display_Attrs ua="na" />
<LDAP_Number_Mapping ua="na" />


<!--  XML Service  -->

<XML_Directory_Service_Name ua="na" />
<XML_Directory_Service_URL ua="na" />
<XML_Application_Service_Name ua="na" />
<XML_Application_Service_URL ua="na" />
<XML_User_Name ua="na" />
<XML_Password ua="na" />


<!--  Call Forward  -->

<Cfwd_All_Dest ua="rw" />
<Cfwd_Busy_Dest ua="rw" />
<Cfwd_No_Ans_Dest ua="rw" />
<Cfwd_No_Ans_Delay ua="rw">20</Cfwd_No_Ans_Delay>


<!--  Speed Dial  -->

<Speed_Dial_2 ua="na" />
<Speed_Dial_3 ua="na" />
<Speed_Dial_4 ua="na" />
<Speed_Dial_5 ua="na" />
<Speed_Dial_6 ua="na" />
<Speed_Dial_7 ua="na" />
<Speed_Dial_8 ua="na" />
<Speed_Dial_9 ua="na" />


<!--  Supplementary Services  -->

<Time_Format ua="rw">12hr</Time_Format>
<Date_Format ua="rw">month/day</Date_Format>


<!--  Audio  -->

<Ringer_Volume ua="rw">8</Ringer_Volume>
<Speaker_Volume ua="rw">8</Speaker_Volume>


<!--  LCD  -->

<LCD_Contrast ua="rw">16</LCD_Contrast>
<Back_Light_Timer ua="na">10s</Back_Light_Timer>
<!--  General  -->
```

```
<Line_Enable_1_ ua="na">Yes</Line_Enable_1_>


<!--  NAT Settings  -->

<NAT_Keep_Alive_Enable_1_ ua="na">No</NAT_Keep_Alive_Enable_1_>
<NAT_Keep_Alive_Msg_1_ ua="na">$NOTIFY</NAT_Keep_Alive_Msg_1_>


<!--  SIP Settings  -->

<SIP_Transport_1_ ua="na">UDP</SIP_Transport_1_>
<SIP_Port_1_ ua="na">5060</SIP_Port_1_>
<SIP_100REL_Enable_1_ ua="na">Yes</SIP_100REL_Enable_1_>
<Auth_Resync-Reboot_1_ ua="na">Yes</Auth_Resync-Reboot_1_>
<SIP_Remote-Party-ID_1_ ua="na">Yes</SIP_Remote-Party-ID_1_>
<Refer-To_Target_Contact_1_ ua="na">No</Refer-To_Target_Contact_1_>
<SIP_Debug_Option_1_ ua="na">None</SIP_Debug_Option_1_>
<Sticky_183_1_ ua="na">No</Sticky_183_1_>
<Auth_INVITE_1_ ua="na">No</Auth_INVITE_1_>
<User_Equal_Phone_1_ ua="na">No</User_Equal_Phone_1_>


<!--  Call Feature Settings  -->

<Default_Ring_1_ ua="rw">1</Default_Ring_1_>
<Conference_Bridge_URL_1_ ua="na" />


<!--  Proxy and Registration  -->
<Proxy_1_ ua="na" />
<Outbound_Proxy_1_ ua="na">199.19.193.9</Outbound_Proxy_1_>
<Alternate_Proxy_1_ ua="na" />
<Alternate_Outbound_Proxy_1_ ua="na" />
<Register_1_ ua="na">Yes</Register_1_>
<Make_Call_Without_Reg_1_ ua="na">No</Make_Call_Without_Reg_1_>
<Register_Expires_1_ ua="na">3600</Register_Expires_1_>
<Use_DNS_SRV_1_ ua="na">Yes</Use_DNS_SRV_1_>
<Proxy_Fallback_Intvl_1_ ua="na">3600</Proxy_Fallback_Intvl_1_>
<Dual_Registration_1_ ua="na">No</Dual_Registration_1_>


<!--  Subscriber Information  -->

<Display_Name_1_ ua="na" />
<User_ID_1_ ua="na" />
<Password_1_ ua="na" />
<Auth_ID_1_ ua="na" />
<Reversed_Auth_Realm_1_ ua="na" />


<!--  Audio Configuration  -->

<Preferred_Codec_1_ ua="na">G722</Preferred_Codec_1_>
<Use_Pref_Codec_Only_1_ ua="na">No</Use_Pref_Codec_Only_1_>
<Second_Preferred_Codec_1_ ua="na">Unspecified</Second_Preferred_Codec_1_>
<Third_Preferred_Codec_1_ ua="na">Unspecified</Third_Preferred_Codec_1_>
<G711u_Enable_1_ ua="na">Yes</G711u_Enable_1_>
<G711a_Enable_1_ ua="na">Yes</G711a_Enable_1_>
<G729a_Enable_1_ ua="na">Yes</G729a_Enable_1_>
<G729ab_Enable_1_ ua="na">Yes</G729ab_Enable_1_>
<G722_Enable_1_ ua="na">Yes</G722_Enable_1_>
<iLBC_Enable_1_ ua="na">Yes</iLBC_Enable_1_>
<Silence_Supp_Enable_1_ ua="na">No</Silence_Supp_Enable_1_>
```

```
  <DTMF_Tx_Method_1_ ua="na">Auto</DTMF_Tx_Method_1_>


  <!--  Dial Plan  -->
  <Dial_Plan_1_ ua="na">( <8:>x. | [*#]xx[*x] | [*#]xx. | [2-9]11S0 | 00 | 011x. |
[0-1][2-9]xxxxxxxxx | 0 | [2-9]xxxxxxxxx | xxxx )</Dial_Plan_1_>

  </flat-profile>
  </device>
```

Acronyms

A/D	Analog To Digital Converter
ANC	Anonymous Call
B2BUA	Back to Back User Agent
Bool	Boolean Values. Specified as yes and no, or 1 and 0 in the profile
CA	Certificate Authority
CAS	CPE Alert Signal
CDR	Call Detail Record
CID	Caller ID
CIDCW	Call Waiting Caller ID
CNG	Comfort Noise Generation
CPC	Calling Party Control
CPE	Customer Premises Equipment
CWCID	Call Waiting Caller ID
CWT	Call Waiting Tone
D/A	Digital to Analog Converter
dB	decibel
dBm	dB with respect to 1 milliwatt
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DRAM	Dynamic Random Access Memory
DSL	Digital Subscriber Loop
DSP	Digital Signal Processor
DTAS	Data Terminal Alert Signal (same as CAS)
DTMF	Dual Tone Multiple Frequency
FQDN	Fully Qualified Domain Name
FSK	Frequency Shift Keying
FXS	Foreign eXchange Station

GW	Gateway
ITU	International Telecommunication Union
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SSL
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
ILEC	Incumbent Local Exchange Carrier
IP	Internet Protocol
ISP	Internet Service Provider
ITSP	Internet Telephony Service Provider
IVR	Interactive Voice Response
LAN	Local Area Network
LBR	Low Bit Rate
LBRC	Low Bit Rate Codec
MC	Mini-Certificate
MGCP	Media Gateway Control Protocol
MOH	Music On Hold
MOS	Mean Opinion Score (1-5, the higher the better)
ms	Millisecond
MSA	Music Source Adaptor
MWI	Message Waiting Indication
OSI	Open Switching Interval
PCB	Printed Circuit Board
PR	Polarity Reversal
PS	Provisioning Server
PSQM	Perceptual Speech Quality Measurement (1-5, the lower the better)
PSTN	Public Switched Telephone Network
NAT	Network Address Translation
OOB	Out-of-band
REQT	(SIP) Request Message
RESP	(SIP) Response Message
RSC	(SIP) Response Status Code, such as 404, 302, 600
RTP	Real Time Protocol
RTT	Round Trip Time
SAS	Streaming Audio Server
SDP	Session Description Protocol

SDRAM	Synchronous DRAM
sec	seconds
SIP	Session Initiation Protocol
SLA	Shared line appearance
SLIC	Subscriber Line Interface Circuit
SP	Service Provider
SSL	Secure Socket Layer
TFTP	Trivial File Transfer Protocol
TCP	Transmission Control Protocol
UA	User Agent
uC	Micro-controller
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VM	Voicemail
VMWI	Visual Message Waiting Indication/Indicator
VQ	Voice Quality
WAN	Wide Area Network
XML	Extensible Markup Language

APPENDIX C

Related Documentation

Cisco provides a wide range of resources to help you and your customer obtain the full benefits of the IP Telephony device.

Use the following sections to obtain related information.

Cisco IP Phone 8800 Series Documentation

Refer to publications that are specific to your language, phone model, and Cisco Unified Communications Manager release. Navigate from the following documentation URL:

http://www.cisco.com/c/en/us/support/collaboration-endpoints/unified-ip-phone-8800-series/tsd-products-support-series-home.html

Cisco Unified Communications Manager Documentation

See the Cisco Unified Communications Manager Documentation Guide and other publications that are specific to your Cisco Unified Communications Manager release. Navigate from the following documentation URL:

http://www.cisco.com/c/en/us/support/unified-communications/unified-communications-manager-callmanager/tsd-products-support-series-home.html

Cisco Business Edition 6000 Documentation

Refer to the Cisco Business Edition 6000 Documentation Guide and other publications that are specific to your Cisco Business Edition 6000 release. Navigate from the following URL:

http://www.cisco.com/c/en/us/support/unified-communications/business-edition-6000/tsd-products-support-series-home.html

Cisco IP Phone Firmware Support Policy

For information on the support policy for Cisco IP Phones, see http://www.cisco.com/c/en/us/support/docs/collaboration-endpoints/unified-ip-phone-7900-series/116684-technote-ipphone-00.html.

Documentation, Service Requests, and Additional Information

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/c/en/us/td/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS Version 2.0.