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:

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: $F_1@L_1[,F_2@_{L2}[,F_3@L_3[,F_4@L_4[,F_5@L_5[,F_6@L_6]]]]]$, 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.