Provisioning Formats

Configuration Profiles

The phone accepts configuration in an XML format.

The examples in this document use configuration profiles with an XML format (XML) syntax.

For detailed information about your phone, refer to the administration guide for your particular 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 phone.

The configuration profile XML format uses standard XML authoring tools to compile the parameters and values.

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

Each phone has a different feature set and therefore, a different set of parameters.

XML Format (XML) 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 phone during a resync operation. The file can be sent without compilation as a binary object.

The phone can accept configuration formats that standard tools generate. This feature eases the development of back-end provisioning server software that generates configuration profiles from existing databases.

To protect confidential information in the configuration profile, the provisioning server delivers this type of file to the phone over a channel secured by TLS. Optionally, the file can be compressed by using the gzip deflate algorithm (RFC1951).

The file can be encrypted with one of these encryption methods:

- AES-256-CBC encryption
- RFC-8188 based HTTP content encryption with AES-128-GCM ciphering

Example: Open Profile Format

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

The <flat-profile> element tag encloses all parameter elements that the phone recognizes.

Configuration File Components

A configuration file can include these components:

- Element tags
- Attributes
- Parameters
- Formatting features
- XML comments

Element Tag Properties

- The XML provisioning format and the Web UI allow the configuration of the same settings. The XML tag name and the field names in the Web UI are similar but vary due to XML element name restrictions. For example, underscores (_) instead of " ".
- The phone recognizes elements with proper parameter names that are encapsulated in the special <flat-profile> element.
- Element names are enclosed in angle brackets.
- Most 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 `/`.

  Example: The <Resync_On_Reset> element represents the **Resync On Reset** field.

- Each element name must be unique. In the administration web pages, the same fields can 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.

  Example: The <Dial_Plan_1_> element represents the **Dial Plan** for Line 1.

- Each opening element tag must have a matching closing element tag. For example:

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

- Element tags are case-sensitive.
- Empty element tags are allowed and will be interpreted as configuring the value to be empty. Enter the opening element tag without a corresponding element tag, and insert a space and a forward slash before the closing angle bracket (>). In this example, Profile Rule B is empty:

```
<Profile_Rule_B />
```

- 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:

```
<flat-profile>
<Speed_Dial_2_Name ua="rw"/>
<Speed_Dial_2_Number ua="rw"/>
<Speed_Dial_3_Name ua="rw"/>
<Speed_Dial_3_Number ua="rw"/>
<Speed_Dial_4_Name ua="rw"/>
<Speed_Dial_4_Number ua="rw"/>
<Speed_Dial_5_Name ua="rw"/>
<Speed_Dial_5_Number ua="rw"/>
<Speed_Dial_6_Name ua="rw"/>
<Speed_Dial_6_Number ua="rw"/>
<Speed_Dial_7_Name ua="rw"/>
<Speed_Dial_7_Number ua="rw"/>
<Speed_Dial_8_Name ua="rw"/>
<Speed_Dial_8_Number ua="rw"/>
<Speed_Dial_9_Name ua="rw"/>
<Speed_Dial_9_Number ua="rw"/>
</flat-profile>
```

- Use an empty value 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.

```
<flat-profile>
<GPP_A>
```

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

- Unrecognized element names are ignored.

Parameter Properties

These properties apply to the parameters:

- Any parameters that are not specified by a profile are left unchanged in the phone.
- Unrecognized parameters are ignored.
- If the Open format profile contains multiple occurrences of the same parameter tag, the last such occurrence overrides any earlier ones. To avoid inadvertent override of configuration values for a parameter, we recommend that each profile specify at most one instance of a parameter.
- The last profile processed takes precedence. If multiple profiles specify the same configuration parameter, the value of the latter profile takes precedence.

String Formats

These properties apply to the formatting of the strings:

- Comments are allowed through standard XML syntax.

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

- Leading and trailing white space is allowed for readability but is removed from the parameter value.
- New lines within a value are converted to spaces.
- An XML header of the form `<? ?>` is allowed, but the phone ignores it.
- 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 (**Admin Login** > **advanced** > **Voice** > **Ext (n)**) equal to (S0 <:18005551212>).

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

- Numeric character escapes, using decimal and hexadecimal values (s.a. `&#40;` and `&#x2e;`), are translated.
- The phone firmware only supports ASCII characters.

Open Profile (XML) Compression and Encryption

The Open configuration profile can be compressed to reduce the network load on the provisioning server. The profile can also 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 compression, the phone expects the compressed file to contain a gzip compatible header. Invocation of the gzip utility on the original Open profile generates the header. The phone inspects the downloaded file header to determine the file format.

For example, if `profile.xml` is a valid profile, the file `profile.xml.gz` is also accepted. Either of the following commands can generate this profile type:

- `>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 Compress an Open Profile with Gzip section.

Open Profile Encryption

Symmetric key encryption can be used to encrypt an open configuration profile, whether the file is compressed or not. Compression, if applied, must be applied before encryption.

The provisioning server uses HTTPS to handle initial provisioning of the phone after deployment. Pre-encrypting configuration profiles offline allows the use of HTTP for resyncing profiles subsequently. This reduces the load on the HTTPS server in large-scale deployments.

The phone supports two methods of encryption for configuration files:

- AES-256-CBC encryption
- RFC 8188-based HTTP content encryption with AES-128-GCM ciphering

The key or Input Keying Material (IKM) must be preprovisioned into the unit at an earlier time. Bootstrap of the secret key can be accomplished securely by using HTTPS.

The configuration file name does not require a specific format, but a file name that ends with the `.cfg` extension normally indicates a configuration profile.

AES-256-CBC Encryption

The phone supports AES-256-CBC encryption for configuration files.

The OpenSSL encryption tool, available for download from various Internet sites, can perform the encryption. Support for 256-bit AES encryption may require recompilation of the tool to enable the AES code. The firmware has been tested against version openssl-1.1.1d.

Encrypt a Profile with OpenSSL provides a tutorial on encryption.

For an encrypted file, 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 lowercase -k precedes the secret key, which can be any plain text phrase, and which is used to generate a random 64-bit salt. With the secret specified by 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 on a configuration profile, the phone 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.

Macro Expansion

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

These parameter groups undergo macro expansion before evaluation:

- Resync_Trigger_*
- Profile_Rule*
- Log_xxx_Msg
- Upgrade_Rule

Under certain conditions, some general-purpose parameters (GPP_*) also undergo macro expansion, as explicitly indicated in Optional Resync Arguments, on page 10.

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

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 contents of the special purpose parameters, GPP_SA through GPP_SD, are mapped to the macro expressions $SA through $SD. These parameters are only macro expanded as the argument of the **--key** , **--uid**, and **--pwd** options in a resync URL.

Conditional Expressions

Conditional expressions can trigger resync events and select from alternate 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 to one of the following three types of literals:

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

Version Numbers

The software version for Cisco IP Phones with Multiplatform Firmware uses this format (where *BN* is the Build Number):

- For Firmware Release 11.3(1)SR1 and previous: sip*yyyy*.*11*-*0*-*1*MPP-*376*

  where *yyyy* indicates the phone model or phone series; *11* is the major version; *0* is the minor version; *1MPP* is the micro version; and *376* is the build number.

- For Firmware Release 11.3(2) and later: sip*yyyy*.*11*-*3*-*2*MPP*0001*-*609*

  where *yyyy* indicates the phone model or phone series; *11* is the major version; *3* is the minor version; *2MPP0001* is the micro version; and *609* is the build number.

The comparing string must use the same format. Otherwise, a format parsing error results.

When comparing the software version, the major version, minor version, and micro version are compared in sequence, and the leftmost digits take precedence over the latter ones. When version numbers are identical, the build number is compared.

Examples of Valid Version Number

- For Firmware Release 11.3(1)SR1 and previous:

```
sip68xx.11-0-1MPP-312
```

- For Firmware Release 11.3(2) and later:

```
sip68xx.11-3-2MPP0001-609
```

- For Firmware Release 11.3(1)SR1 and previous:

```
sip78xx.11-0-1MPP-312
```

- For Firmware Release 11.3(2) and later:

```
sip78xx.11-3-2MPP0001-609
```

- For Firmware Release 11.3(1)SR1 and previous:

```
sip88xx.11-0-1MPP-312
```

- For Firmware Release 11.3(2) and later:

```
sip88xx.11-3-2MPP0001-609
```

Comparison

- For Firmware Release 11.3(1)SR1 and previous:

```
sipyyyy.11-3-1MPP-110 > sipyyyy.11-2-3MPP-256
```

- For Firmware Release 11.3(2) and later:

```
sipyyyy.11-3-2MPP0002-256 > sipyyyy.11-3-2MPP0001-609
```

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 convenient for expressing the condition in an Open format profile.

Operator	Alternate Syntax	Description	Applicable to Integer and Version Operands	Applicable to 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
AND		and	Yes	Yes

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

When used in the context of the Profile_Rule* and Upgrade_Rule parameters, conditional expressions must be enclosed within the syntax "(expr)?" as in this upgrade rule example. Remember to replace *BN* with the build number of your firmware load to upgrade to.

- For Firmware Release 11.3(1)SR1 and previous

```
($SWVER ne sip68xx.11-0-0MPP-256)? http://ps.tell.com/sw/sip68xx.11-0-0MPP-BN.loads
```

- For Firmware Release 11.3(2) and later

```
($SWVER ne sip68xx.11-3-2MPP0001-609)?
http://ps.tell.com/sw/sip68xx.11-3-2MPP0001-BN.loads
```

- For Firmware Release 11.3(1)SR1 and previous

```
($SWVER ne sip78xx.11-0-0MPP-256)? http://ps.tell.com/sw/sip78xx.11-0-0MPP-BN.loads
```

- For Firmware Release 11.3(2) and later

```
($SWVER ne sip78xx.11-3-2MPP0001-609)?
http://ps.tell.com/sw/sip78xx.11-3-2MPP0001-BN.loads
```

- For Firmware Release 11.3(1)SR1 and previous

```
($SWVER ne sip88xx.11-0-0MPP-256)? http://ps.tell.com/sw/sip88xx.11-0-0MPP-BN.loads
```

- For Firmware Release 11.3(2) and later

```
($SWVER ne sip88xx.11-3-2MPP0001-609)?
http://ps.tell.com/sw/sip88xx.11-3-2MPP0001-BN.loads
```

Do not use the preceding syntax with parentheses to configure the Resync_Trigger_* parameters.

URL Syntax

Use the Standard URL syntax 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 these values:

- tftp
- http
- https

If **`scheme`** is omitted, tftp is assumed. The server can be a DNS-recognized hostname 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, the tftp server specified through DHCP (option 66) is used.

Note For upgrade rules, the server must be specified.

If **`port`** is missing, 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:

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

When using DHCP option 66, the empty syntax is not supported by upgrade rules. It is only applicable for Profile Rule*.

RFC 8188-Based HTTP Content Encryption

The phone supports RFC 8188-based HTTP content encryption with AES-128-GCM ciphering for configuration files. With this encryption method, any entity can read the HTTP message headers. However, only the entities that know the Input Keying Material (IKM) can read the payload. When the phone is provisioned with the IKM, the phone and the provisioning server can exchange configuration files securely, while allowing third-party network elements to use the message headers for analytic and monitoring purposes.

The XML configuration parameter **IKM_HTTP_Encrypt_Content** holds the IKM on the phone. For security reasons, this parameter is not accessible on the phone administration web page. It is also not visible in the phone's configuration file, which you can access from the phone's IP address or from the phone's configuration reports sent to the provisioning server.

If you want to use the RFC 8188-based encryption, ensure the following:

- Provision the phone with the IKM by specifying the IKM with the XML parameter **IKM_HTTP_Encrypt_Content** in the configuration file that is sent from the provisioning server to the phone.
- If this encryption is applied to the configuration files sent from the provisioning server to the phone, ensure that the *Content-Encoding* HTTP header in the configuration file has “aes128gcm”.

  In the absence of this header, the AES-256-CBC method is given precedence. The phone applies AES-256-CBC decryption if a AES-256-CBC key is present in a profile rule, regardless of IKM.
- If you want the phone to apply this encryption to the configuration reports that it sends to the provisioning server, ensure that there is no AES-256-CBC key specified in the report rule.

Optional Resync Arguments

Optional arguments, **key**, **uid**, and **pwd**, can precede the URLs entered in Profile_Rule* parameters, collectively enclosed by square brackets.

key

The **--key** option tells the phone that the configuration file that it receives from the provisioning server is encrypted with AES-256-CBC encryption, unless the *Content-Encoding* header in the file indicates “aes128gcm” encryption. The key itself is specified as a string following the term **--key**. The key can be enclosed in double-quotes (") optionally. The phone uses the key to decrypt the configuration file.

Usage Examples

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

The bracketed optional arguments are macro expanded. Special purpose parameters, GPP_SA through GPP_SD, are macro expanded into macro variables, $SA through $SD, only when they are used as key option arguments. See these examples:

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

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

uid and pwd

The **uid** and **pwd** options can be used to specify the userID and password that will be sent in response to HTTP Basic and Digest authentication challenges when the specified URL is requested. The bracketed optional arguments are macro expanded. Special purpose parameters, GPP_SA through GPP_SD, are macro expanded into macro variables, $SA through $SD, only when they are used as key option arguments. See these examples:

```
GPP_SA = MyUserID
GPP_SB = MySecretPassword
```

[--uid $SA --pwd $SB] https://provisioning_server_url/path_to_your_config/your_config.xml

would then expand to:

[--uid MyUserID --pwdMySecretPassword]
https://provisioning_server_url/path_to_your_config/your_config.xml

Application of a Profile to the Phone

After you create an XML configuration script, it must be passed to the phone for application. To apply the configuration, you can either download the configuration file to the phone from a TFTP, HTTP, or HTTPS server using a web browser or by using cURL command line utility.

Download the Configuration File to the Phone from a TFTP Server

Complete these steps to download the configuration file to a TFTP server application on your PC.

Procedure

Step 1 Connect your PC to the phone LAN.

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

Step 3 In a web browser, enter the phone LAN IP address, the IP address of the computer, the filename, and the login credentials. Use 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
```

Download the Configuration File to the Phone with cURL

Complete these steps to download the configuration to the phone by using cURL. This command-line tool is used to transfer data with a URL syntax. To download cURL, visit:

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

Note We recommend that you do not use cURL to post the configuration to the phone because the username and password might get captured while using cURL.

Procedure

Step 1 Connect your PC to the LAN port of the phone.

Step 2 Download the configuration file to the phone by entering the following cURL command:

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

Provisioning Parameter Types

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

These provisioning parameter types exist:

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

General Purpose Parameters

The general-purpose parameters GPP_* (**Admin Login** > **advanced** > **Voice** > **Provisioning**) are used as free string registers when configuring the phone 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
- Multistage 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 uppercase macro names (A through P) suffice to identify the contents of GPP_A through GPP_P. Also, the two-letter uppercase macro names SA through SD identify GPP_SA through GPP_SD as a special case when used as arguments of the following URL options:

key, **uid**, and **pwd**

These parameters can be used as variables in provisioning and upgrade rules. They are referenced by prefixing the variable name with a '$' character, such as $GPP_A.

Use General Purpose Parameters

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

Before you begin

Access the phone administration web page. See Access the Phone Web Interface.

Procedure

Step 1 Select **Voice** > **Provisioning**.

Step 2 Scroll to the **General Purpose Parameters** section.

Step 3 Enter valid values in the fields, GPP A through GPP P.

Step 4 Click **Submit All Changes**.

Enable Parameters

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

The Resync_From_SIP parameter controls requests for resync operations. A SIP NOTIFY event is sent from the service provider proxy server to the phone. If enabled, the proxy can request a resync. To do so, the proxy sends a SIP NOTIFY message that contains the Event: resync header to the device.

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

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

When Resync_On_Reset is enabled, the device introduces a random delay that follows the boot-up sequence before the reset is performed. The delay is a random time up to the value that the Resync_Random_Delay (in seconds) specifies. In a pool of phones that power up simultaneously, this delay spreads out the start times of the resync requests from each unit. This feature can be useful in a large residential deployment, in the case of a regional power failure.

Triggers

The phone allows you to resync at specific intervals or at a specific time.

Resync at Specific Intervals

The phone 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. If a voice line is active when a resync is due, the phone delays the resync procedure until the line becomes idle again. A resync can cause configuration parameter values to change.

A resync operation can fail because the phone is unable to retrieve a profile from the server, the downloaded file is corrupt, or an internal error occurred. The device tries to resync again after a time that is specified in Resync_Error_Retry_Delay (seconds). If Resync_Error_Retry_Delay is set to 0, the device does not try to resync again after a failed resync attempt.

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 parameter can be programmed with a conditional expression that undergoes macro expansion. When the resync interval expires (time for the next resync) the triggers, if set, will prevent resync unless one or more triggers evaluates to true.

The following example condition triggers a resync. In the example, the last phone upgrade attempt has elapsed more than 5 minutes (300 seconds), and at least 10 minutes (600 seconds) have elapsed since the last resync attempt.

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

Resync 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.

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

Flooding the server with resync requests from multiple phones that are set to resync at the same time should be avoided. To do so, 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.

Optionally, you can use a plus sign to specify another numeric value that appends a random extra delay.

Example 1

In this example, the phone periodically resyncs every 2 hours. If a resync failure occurs, the device retries at these intervals: 30 minutes, 1 hour, 2 hours, 4 hours. The device continues to try at 4-hour intervals until it resyncs successfully.

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

Example 2

In this example, the device periodically resyncs every hour (plus an extra random delay of up to 10 minutes). In the 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). The device continues to try at 2-hour intervals (plus up to 15 minutes) until it successfully resyncs.

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

Example 3

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 the upgrade still fails, the device retries every four to five hours until the upgrade succeeds.

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

Profile Rules

The phone provides multiple remote configuration profile parameters (Profile_Rule*). Thus, each resync operation can retrieve multiple files that different servers manage.

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 phones in a deployment. A separate, unique file is provided for each account. Encryption keys and certificate information can be supplied by still another profile, stored on a separate server.

Whenever a resync operation is due, the phone 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 retrieval from a remote provisioning server, with a possible update of 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 another resync.

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 alternatives except the last one must provide a conditional expression. This expression is evaluated and is 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 that is located at the specified URL. The system attempts to update the internal parameters accordingly.

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

Example 1

This example resyncs unconditionally to the profile at the specified URL, and performs an HTTP GET request to the remote provisioning server:

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

Example 2

In this 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. The device sends the contents of the macro expanded GPP_A, which may provide additional information on the device state:

```
($PRVTMR ge 600)? http://p.tel.com/has-reg.cfg
| [--post a] http://p.tel.com/lost-reg?
```

Example 3

In this example, the device resyncs to the same server. The device 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
```

Example 4

In this 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
```

Example 5

In this example, the profile that the server returns is assumed to contain XML element tags. These tags must be remapped to proper parameter names by the aliases map stored in GPP_B:

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

A resync is typically considered unsuccessful if a requested profile is not received from the server. The Resync_Fails_On_FNF parameter can override this default behavior. If Resync_Fails_On_FNF is set to No, 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

Upgrade rule is to tell the device to activate to a new load and from where to get the load, if necessary. If the load is already on the device, it will not try to get the load. So, validity of the load location does not matter when the desired load is in the inactive partition.

The Upgrade_Rule specifies a firmware load which, if different from the current load, will be downloaded and applied unless limited by a conditional expression or Upgrade_Enable is set to **No**.

The phone provides one configurable remote upgrade parameter, Upgrade_Rule. This parameter accepts syntax similar to the profile rule parameters. URL options are 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 that the URL specifies. After macro expansion and evaluation of the rule, the device does not reattempt to upgrade until the rule is modified or the effective combination of scheme + server + port + filepath is changed.

To attempt a firmware upgrade, the device disables audio at the start of the procedure and reboots at the end of the procedure. The device automatically begins an upgrade that is driven by the contents of Upgrade_Rule only if all voice lines are currently inactive.

For example,

```
https://10.73.10.223/firmware/sip78xx.11-3-1MPP-678.loads
```

```
http://p.tel.com/firmware/sip88xx.11-3-1MPP-678.loads
```

- For Cisco IP Phone 6821:

  ```
  http://p.tel.com/firmware/sip6821.11-3-1MPP-678.loads
  ```

- For the other phones in Cisco IP 6800 Series:

  ```
  http://p.tel.com/firmware/sip68xx.11-3-1MPP-678.loads
  ```

In this example, the Upgrade_Rule upgrades the firmware to the image that is stored at the indicated URL.

Here is another example:

```
("$F" ne "beta-customer")? http://p.tel.com/firmware/sip78xx.11-3-1MPP-678.loads
| http://p.tel.com/firmware/sip78xx.11-3-1MPP-678.loads
```

Here is another example:

```
("$F" ne "beta-customer")? http://p.tel.com/firmware/sip88xx.11-3-1MPP-678.loads
| http://p.tel.com/firmware/sip88xx.11-3-1MPP-678.loads
```

Here is another example:

- For Cisco IP Phone 6821:

  ```
  ("$F" ne "beta-customer")? http://p.tel.com/firmware/sip6821.11-3-1MPP-678.loads
  | http://p.tel.com/firmware/sip6821.11-3-1MPP-678.loads
  ```

- For the other models in Cisco IP Phone 6800 Series:

```
("$F" ne "beta-customer")? http://p.tel.com/firmware/sip68xx.11-3-1MPP-678.loads
| http://p.tel.com/firmware/sip68xx.11-3-1MPP-678.loads
```

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 regarding firmware revision number, which can be a useful customization option. If a valid firmware revision number is configured in the Downgrade_Rev_Limit parameter, the device rejects upgrade attempts for firmware versions earlier than the specified limit.

Data Types

These data types are used with configuration profile parameters:

- {a,b,c,…}—A choice among a, b, c, …
- Bool—Boolean value of either "yes" or "no."
- CadScript—A miniscript 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*. 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
```

- DialPlanScript—Scripting syntax that is used to specify Line 1 and Line 2 dial plans.
- Float<n>—A floating point value with up to n decimal places.
- FQDN—Fully Qualified Domain Name. It can contain up to 63 characters. Examples are as follows:
  - sip.Cisco.com:5060 or 109.12.14.12:12345
  - sip.Cisco.com or 109.12.14.12
- FreqScript—A miniscript that specifics the frequency and level parameters of a tone. Contains up to 127 characters.

  Syntax: $F_1@L_1[,F_2@L_2[,F_3@L_3[,F_4@L_4[,F_5@L_5[,F_6@L_6]]]]]$, where:
  - F_1–F_6 are frequency in Hz (unsigned integers only).
  - L_1–L_6 are corresponding levels in dBm (with up to one decimal place).

  White spaces before and after the comma are allowed but not recommended.

  Example 1—Call Waiting Tone:

```
440@-10

Number of Frequencies = 1
Frequency 1 = 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
```

- IP— Valid IPv4 Address in the form of x.x.x.x, where x is between 0 and 255. Example: 10.1.2.100.
- UserID—User ID as it appears in a URL; 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. The string can contain up to 39 characters.
- PhTmplt—A phone number template. Each template may contain one or more patterns that are separated by a comma (,). White space at the beginning of each pattern is ignored. “?” and “*” represent wildcard characters. To represent literally, use %xx. For example, %2a represents *. The template can contain up to 39 characters. Examples: “1408*, 1510*”, “1408123????, 555?1.”.
- Port—TCP/UDP Port number (0-65535). It can be specified in decimal or hex format.
- ProvisioningRuleSyntax—Scripting syntax that is used to define configuration resync and firmware upgrade rules.
- PwrLevel—Power level expressed in dBm with one decimal place, such as –13.5 or 1.5 (dBm).

- RscTmplt—A template of SIP Response Status Code, such as “404, 5*”, “61?”, “407, 408, 487, 481”. It can contain up to 39 characters.
- Sig<n>—Signed n-bit value. It can be specified in decimal or hex format. A “-” sign must precede negative values. A + sign before positive values is optional.
- Star Codes—Activation code for a supplementary service, such as *69. The code can contain up to 7 characters.
- Str<n>—A generic string with up to n nonreserved characters.
- Time<n>—Time duration in seconds, with up to n decimal places. Extra specified decimal places are ignored.
- ToneScript—A miniscript that specifies the frequency, level, and cadence parameters of a call progress tone. Script 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:

  - $f_{i,j} = n_1[+n_2]+n_3[+n_4[+n_5[+n_6]]]]$.
  - $1 < n_k < 6$ specifies the frequency components in the FreqScript that 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
```

- 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.

Note Keep these under consideration:

- <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 phone 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 phone.
- 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.

Profile Updates and Firmware Upgrades

The phone supports secure remote provisioning (configuration) and firmware upgrades. An unprovisioned phone can receive an encrypted profile targeted for that device. The phone does not require an explicit key due to a secure first-time provisioning mechanism that uses SSL functionality.

User intervention is not required to either start or complete a profile update, or firmware upgrade, or if intermediate upgrades are required to reach a future upgrade state from an older release. A profile resync is only attempted when the phone is idle, because a resync can trigger a software reboot and disconnect a call.

General-purpose parameters manage the provisioning process. Each phone can be configured to periodically contact a normal provisioning server (NPS). Communication with the NPS does not require the use of a secure protocol because the updated profile is encrypted by a shared secret key. The NPS can be a standard TFTP, HTTP, or HTTPS server with client certificates.

The administrator can upgrade, reboot, restart, or resync phones by using the phone web user interface. The administrator can also perform these tasks by using a SIP notify message.

Configuration profiles are generated by using common, open-source tools that integrate with service provider provisioning systems.

Allow Profile Updates

Profile updates can be allowed at specified intervals. Updated profiles are sent from a server to the phone by using TFTP, HTTP, or HTTPS.

You can also configure the parameters in the phone configuration file with XML(cfg.xml) code.

Before you begin

Access the phone administration web page. See Access the Phone Web Interface.

Procedure

Step 1 Select **Voice** > **Provisioning**.

Step 2 In the **Configuration Profile** section, choose **Yes** from the **Provision Enable** parameter.

You can configure this parameter in the phone configuration XML file (cfg.xml) by entering a string in this format:

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

Default: Yes

Step 3 Set the parameters as described in the Profile Resync Parameters table.

Step 4 Click **Submit All Changes**.

Allow and Configure Firmware Upgrades

Firmware updates can be allowed at specified intervals. Updated firmware is sent from a server to the phone by using TFTP or HTTP. Security is less of an issue with a firmware upgrade, because firmware does not contain personal information.

You can also configure the parameters in the phone configuration file with XML(cfg.xml) code.

Before you begin

Access the phone administration web page. See Access the Phone Web Interface.

Procedure

Step 1 Select **Voice** > **Provisioning**.

Step 2 In the **Firmware Upgrade** section, choose **Yes** from the **Upgrade Enable** parameter.

You can configure this parameter in the phone configuration XML file (cfg.xml) by entering a string in this format:

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

Options: Yes and No

Default: Yes

Step 3 Set the **Upgrade Error Retry Delay** parameter in seconds.

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.

You can configure this parameter in the phone configuration XML file (cfg.xml) by entering a string in this format:

```
<Upgrade_Error_Retry_Delay ua="na">3600</Upgrade_Error_Retry_Delay>
```

Default: 3600

:

```
<tftp|http|https>://<ip address>/image/<load name>
```

Step 4 Set the **Upgrade Rule** parameter by entering a firmware upgrade script that defines upgrade conditions and associated firmware URLs. It uses the same syntax as Profile Rule. Enter a script and use the following format to enter the upgrade rule:

```
<tftp|http|https>://<ipaddress>/image/<load name>
```

For example:

tftp://192.168.1.5/image/sip88xx.11-0-0MPP-BN.loads

tftp://192.168.1.5/image/sip78xx.11-0-1MPP-BN.loads

You can configure this parameter in the phone configuration XML file (cfg.xml) by entering a string in this format:

```
<Upgrade_Rule ua="na">http://10.74.10.205:6970/sip8845_65.0104-MPP-9875dev.loads
</Upgrade_Rule>
```

Step 5 Click **Submit All Changes**.

Upgrade Firmware by TFTP, HTTP, or HTTPS

The phone supports firmware upgrade by TFTP, HTTP, or HTTPS.

Note Downgrades to earlier releases may not be available for all devices. For more information, see the release notes for your phone and firmware version.

Before you begin

The firmware load file must be downloaded to an accessible server.

Procedure

Step 1 Rename the image as follows:

`cp-x8xx-sip.aa-b-cMPP.cop` to **`cp-x8xx-sip.aa-b-cMPP.tar.gz`**

where:

`x8xx` is the phone series, such as 7811 or 7832.

`x8xx` is the phone series, such as 8811.

`x8xx` is the phone series, such as 6841.

`aa-b-c` is the release number, such as 10-4-1

Step 2 Use the **`tar -xzvf`** command to untar the tar ball.

Step 3 Copy the folder to a TFTP, HTTP, or HTTPS download directory.

Step 4 Access the phone administration web page. See Access the Phone Web Interface.

Step 5 Select **Voice** > **Provisioning**.

Step 6 Find the load filename which ends in **.loads** and append it to the valid URL.

Step 7 Click **Submit All Changes**.

Upgrade Firmware With a Browser Command

An upgrade command entered into the browser address bar can be used to upgrade firmware on a phone. The phone updates only when it is idle. The update is attempted automatically after the call is complete.

Procedure

To upgrade the phone with a URL in a web browser, enter this command:

```
http://<phone_ip>/admin/upgrade?<schema>://<serv_ip[:port]>/filepath
```