Provisioning Examples

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

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

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

Basic Resync

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

TFTP Resync

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

Exercise

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The GPP_A parameter should now contain the value 12345678.

Logging with syslog

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

Exercise

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

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

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

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

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

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

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

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

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

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

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

- Log_Request_Msg
- Log_Success_Msg
- Log_Failure_Msg.

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

Automatic Device Resync

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

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

Exercise

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

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

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

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

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

Step 4 Click **Submit all Changes**.

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

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

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

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

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

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

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

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

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

Unique Profiles, Macro Expansion, and HTTP

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

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

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

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

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

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

Exercise

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

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

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

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

Step 4 Open the admin/advanced page Provisioning tab.

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

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

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

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

HTTP GET Resync

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

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

Exercise

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

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

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

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

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

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

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

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

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

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

Provisioning Through Cisco XML

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

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

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

URL Resolution by Using Macro Expansion

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

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

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

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

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

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

The Profile_Rule has the value:

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

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

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

Secure HTTPS Resync

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

Basic HTTPS Resync

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

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

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

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

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

Exercise

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

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

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

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

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

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

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

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

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

Step 5 Restart the server.

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

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

Step 8 Inspect the server certificate supplied by the server.

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

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

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

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

Step 9 Click **Submit All Changes**.

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

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

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

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

HTTPS With Client Certificate Authentication

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

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

Exercise

Step 1 Enable client certificate authentication on the HTTPS server.

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

```
SSLVerifyClient  require
```

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

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

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

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

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

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

HTTPS Client Filtering and Dynamic Content

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

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

Exercise

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

Step 2 Generate the following Perl reflector script:

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

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

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

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

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

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

Step 6 Click **Submit All Changes**.

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

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

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

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

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

HTTPS Certificates

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

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

How HTTPS Works

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

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

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

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

SSL Server Certificates

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

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

Client Certificates

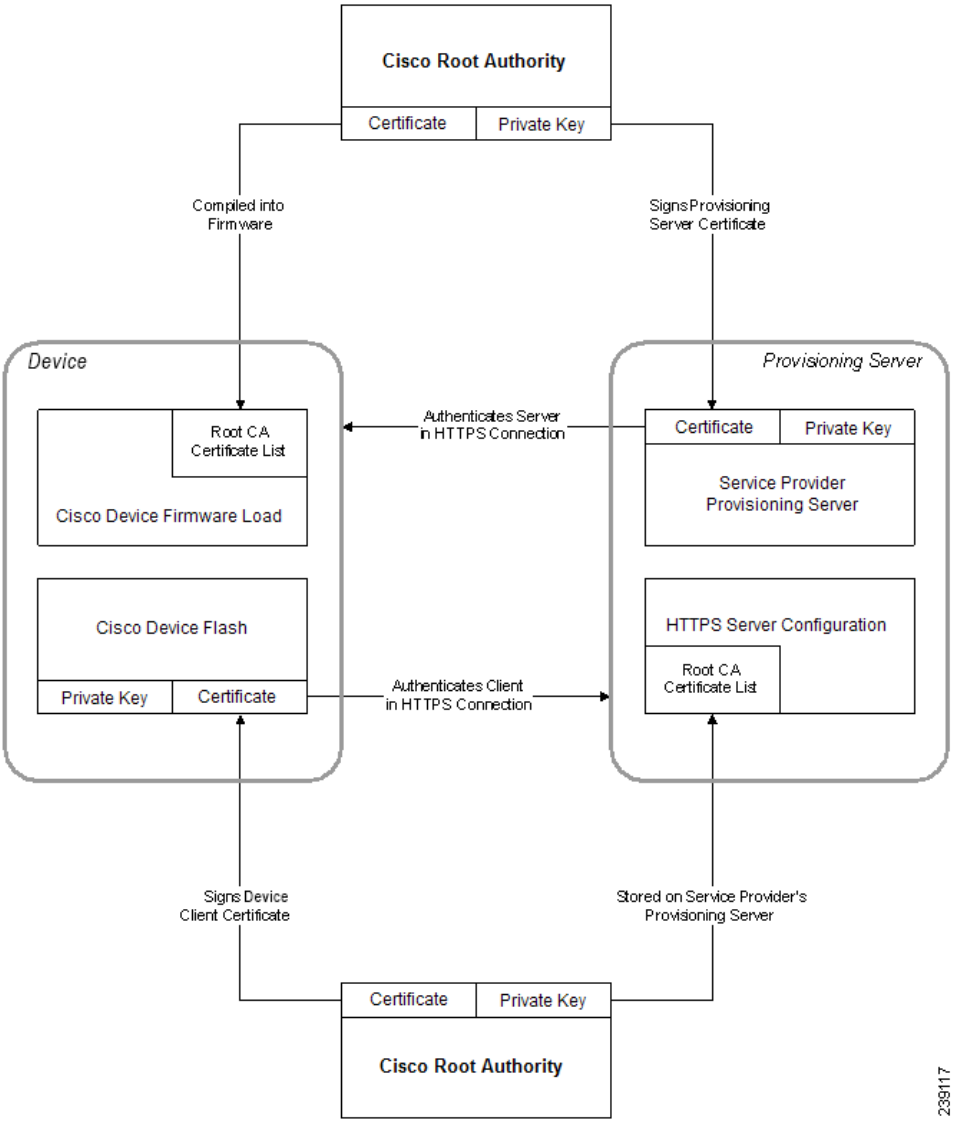
In addition to a direct attack on an IP Telephony device, an attacker might attempt to contact a provisioning server by using a standard web browser or another HTTPS client to obtain the configuration profile from the provisioning server. To prevent this kind of attack, each IP Telephony device also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

Certificate Structure

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

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

Figure 4-1 Certificate Authority Flow



Profile Management

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

Open Profile gzip Compression

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

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

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

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

Exercise

Step 1 Install gzip on the local PC.

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

```
gzip basic.txt
```

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

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

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

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

Step 5 Click **Submit All Changes**.

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

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

Profile Encryption by Using OpenSSL

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

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

Exercise

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

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

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

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

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

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

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

Step 5 Click **Submit All Changes**.

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

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

Partitioned Profiles

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

Exercise

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

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

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

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

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

Step 4 Click **Submit All Changes**.

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

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