CHAPTER 13

Configuring CWMP Service Security

This chapter describes security options for Cisco Broadband Access Center's (BAC) CWMP services, including authentication and encryption. It describes how to enable HTTP over SSL transport, called SSL in this chapter, for the CWMP services and the HTTP file services on the DPE, and use the certificate-management tool to create a certificate store on the DPE.

This chapter includes the following sections:

- Overview, page 13-1
- Key and Certificate Management in Cisco BAC, page 13-2
- Configuring SSL Service, page 13-3
- Configuring Security for DPE Services, page 13-11
- Configuring Security for RDU Services, page 13-18
- Signed Configuration for Devices, page 13-20

Overview

Cisco BAC supports secure provisioning of CWMP devices by using SSL, specifically SSL 3.0 and TLS 1.0, as defined in RFC 2246. It supports device authentication based on shared secrets with the DPE by using Basic and Digest authentication as defined in RFC 2617.

Cisco BAC provides multiple services at the DPE: two instances of the CWMP service and two instances of the HTTP file service. You enable and configure each service independently, with different security options providing the flexibility of handling various devices differently on the same DPE.

Cisco BAC also provides secure device authentication by using unique or generic client certificates.

- Generic—Enables device certificate authentication through SSL by using a generic certificate that is common to all customer premises equipment (CPE) or a large subset of CPE. For example, all VoIP devices in a given deployment may have the same generic certificate that the service provider issues.

  The client certificate is validated against a signing authority key, but does not establish identity of a given device. The device identifier is formed by using the data provided using HTTP Basic or Digest authentication; or by using the data in the CWMP Inform message.

- Unique—Enables client certificate authentication through SSL by using the unique certificate that each device provides. After the client certificate is validated by using the signing certificate authority's public key, the device's unique identifier is formed by using the CN field of the client certificate.

Cisco BAC supports a unique option of performing client-certificate authentication at a hardware load balancer and SSL accelerator such as Cisco ACE 4710.

In this scenario, the ACE performs certificate validation, extracts information about the SSL session, specifically client certificate fields, from the device certificate, and inserts that data into special HTTP headers. Cisco BAC then retrieves the CN field from the ACE header ClientCert-Subject-CN to form the unique device identifier.

Cisco BAC allows a mix of authentication options. For example, you can validate that the device has a generic certificate by using SSL and then perform additional HTTP Basic or Digest authentication once SSL connection is established.

Cisco BAC also provides an option of no-device authentication. This option is useful if the device is authenticated downstream from the DPE and the identity that the device presents can be trusted. If the Cisco BAC DPE is configured to perform no-authentication, it extracts the device identity from the Inform message and trusts it.

You can encrypt traffic between the device and a DPE by using SSL. Cisco BAC supports a variety of cipher suites, which determine the encryption algorithms to be used with the device and the encryption key length. You can configure acceptable cipher suites on the DPE by using the CLI. Another option is to terminate SSL at a hardware load balancer and accelerator for higher scalability.

Note Use the DPE CLI to configure security options for the CWMP service and the HTTP file service. For configuration instructions, see the *Cisco Broadband Access Center 3.8 DPE CLI Reference*.

Key and Certificate Management in Cisco BAC

The DPE stores the certificates that the SSL protocol requires for authentication in the keystore. This keystore is a database in the form of a file that contains private keys and their associated public key X.509 certificates.

There are two keystores on the DPE server. The keystores are the cacerts keystore and the server certificates keystore.

- The cacerts keystore contains public key certificates that the DPE trusts for authenticating devices' client certificates.
- The server certificates keystore contains the private key and the associated certificate chain for the server-side certificate, which is used to authenticate the DPE to the devices.

All DPE SSL services share a single cacerts keystore. This keystore can contain any number of signing authority certificates. The name of the cacerts keystore is fixed, and it must always reside in BPR_HOME/jre/lib/security directory. Cisco BAC ships with a default cacerts keystore, which can be manipulated by adding and removing signing authority certificates.

In contrast to the cacerts keystore, there can be multiple server certificate keystores and you can configure each SSL service in the DPE to use a different server certificate keystore. Each server certificate keystore should contain only one certificate chain. The server certificate keystores must reside in the BPR_HOME/dpe/conf directory.

If SSL is being terminated on the DPEs, and the provisioning group contains multiple DPEs, then all the DPEs must be configured with an identical keystore. An identical keystore is required since the same fully qualified domain name (FQDN) URL of the autoconfiguration server (ACS) is used to resolve to all DPEs in the provisioning group.

As defined in the TR-069 specification, the ACS URL must be identical to the Common Name (CN) value of the server certificate, which is imported into the keystore.

The DPE ships with a default sample server certificates keystore, called servercerts, which contains a self-signed server certificate.

However, because a CWMP device does not normally trust self-signed certificates, you cannot use the sample keystore to enable SSL for device provisioning. Instead, you must obtain a signed ACS certificate with private key and use it to create a new server certificate keystore (see Configuring DPE Keystore by Using the Keytool, page 13-3). You can use the default keystore to test the SSL service link before acquiring and configuring an ACS certificate.

Configuring SSL Service

To enable SSL on Cisco BAC:

Step 1 Create a server certificate keystore, which contains the private key and the associated ACS public key certificate. This process also requires updating the cacerts keystore to load the public certificate of the signing authority for the server certificate.

Step 2 Optionally, if you would like to configure CPE authentication to use client certificates, update the cacerts keystore with the public key of the CPE certificate authority root certificate, which can validate CPE certificates. For details, see Configuring DPE Keystore by Using the Keytool, page 13-3.

Step 3 Configure the DPE to use the new server certificate keystore from the DPE CLI. For details, see Configuring Security for DPE Services, page 13-11.

Step 4 Enable SSL transport for the CWMP service or the HTTP file service by using the DPE CLI. For more details, see Configuring Security for DPE Services, page 13-11.

Note To ensure that the changes you make to the keystore take effect, you must restart the DPE by using the **dpe reload** command from the CLI, or the **/etc/init.d/bprAgent restart dpe** command from the watchdog agent command line (see Using Cisco BAC Process Watchdog from the Command Line, page 9-2).

Configuring DPE Keystore by Using the Keytool

You use Cisco BAC to configure the server certificate keystore and the cacerts keystore by using the keytool utility. The keytool is a key and certificate-management utility, which you use to administer the certificates on the DPE server.

The keytool utility resides in the Cisco BAC default installation directory, at /opt/CSCObac/jre/bin/keytool. Run the keytool utility by using a secure connection to the DPE server.

Note You must execute the keytool utility bundled with this Cisco BAC version, because the keystore file format varies between keytool releases.

To configure a server certificate:

Step 1 Create a new server keystore with a new private key by using keytool. See Generating Server Certificate Keystore and Private Key for a New Certificate, page 13-6, for details.

Step 2 Generate a certificate-signing request (CSR). See Generating a Certificate-Signing Request, page 13-7, for details.

Step 3 Request a public certificate from the signing authority by using CSR. See Generating a Certificate-Signing Request, page 13-7, for details.

Step 4 Load the public key of the signing authority into the cacerts keystore. See Importing Signing Authority Certificate into Cacerts Keystore, page 13-8, for details.

Step 5 Load the signed server certificate into the server keystore. See Importing the Signed Certificate into Server Certificate Keystore, page 13-9, for details.

Step 6 Put the new keystore file in the DPE *BPR_HOME/dpe/conf* directory.

Step 7 At the CLI, configure one of the DPE services to use the new keystore. See Configuring SSL Service, page 13-3, for details

Step 8 Restart the DPE by using the **dpe reload** command from the CLI, or the **/etc/init.d/bprAgent restart dpe** command from the watchdog agent command line (see Using Cisco BAC Process Watchdog from the Command Line, page 9-2).

Note To enable the use of client certificates for device authentication, ensure that the public certificate of the signing authority for device certificates is loaded into the cacerts keystore. Follow the procedure described in Importing Signing Authority Certificate into Cacerts Keystore, page 13-8, for details.

Importing an Existing Signed Server Certificate

If you already have the signed server certificate and you want to load it into the keystore, you must know the private key that is associated with the certificate. In this case, instead of following the procedure described above, follow the steps outlined in this section. Use the PCKS#12 file format, which combines both the private key and the signed certificate. You can load this file into a keystore by using the **keystore import-pkcs12** command.

To configure a server certificate with an existing signed server certificate:

Step 1 Load the existing private key and certificates into a DPE-compatible file, used in authenticating the DPE to SSL clients, by using the **keystore import-pkcs12** command.

When using this command, the syntax is:

```
keystore import-pkcs12 keystore-filename pkcs12-filename keystore-password key-password
export-password export-key-password
```

- *keystore-filename*—Identifies the keystore file to create. If it already exists, it will be overwritten.

  Note Remember to specify the full path of the keystore file.

- *pkcs12-filename*—Identifies the PKCS#12 file from which you intend to import the key and certificate.

- *keystore-password*—Identifies the private key password and the keystore password that you used when you created your keystore file. This password must be between 6 and 30 characters.
- *key-password*—Identifies the password used to access keys within DPE keystore. This password must be between 6 and 30 characters.
- *export-password*—Identifies the password used to decrypt the key in the PKCS#12 file. The export password must be between 6 and 30 characters.
- *export-key-password*—Identifies the password used to access keys within the PKCS#12 keystore. This password must be between 6 and 30 characters.

For example:

```
dpe# keystore import-pkcs12 example.keystore example.pkcs12 changeme changeme changeme
changeme
% Reading alias [1]

% Reading alias [1]: key with format [PKCS8] algorithm [RSA]

% Reading alias [1]: cert type [X.509]

% Created JKS keystore: example.keystore

% OK
```

Step 2 Copy the new keystore file into the DPE *BPR_HOME/dpe/conf* directory.

Step 3 At the CLI, configure one of the DPE services to use the new keystore. See Configuring SSL Service, page 13-3, for details.

Step 4 Restart the DPE by using the **dpe reload** command from the CLI, or the **/etc/init.d/bprAgent restart dpe** command from the watchdog agent command line (see Using Cisco BAC Process Watchdog from the Command Line, page 9-2).

Using the Keytool Commands

The keytool utility uses command arguments to configure a DPE keystore. Table 13-1 lists the keytool commands and their descriptions.

Table 13-1 Keytool Commands

-alias *alias*	Identifies the identity assigned to a keystore entry, which stores the certificate chain and the private key. Subsequent keytool commands must use the same alias to refer to the entity.
-dname *dname*	Identifies the X.500 Distinguished Names used to identify entities, such as those that the subject and the issuer named.
-file *csr_file*	Identifies the CSR file to be exported.
-file *cert_file*	Identifies the file from which the certificate is to be read.
-keyalg *keyalg*	Identifies the algorithm to be used for key-pair creation. The values are DSA (default) and RSA.
-keysize *keysize*	Specifies a keysize, whose values must be in multiples of 64 bits.
-keypass *keypass*	Identifies the password assigned to a key pair.
-keystore *keystore*	Customizes the name and location of a keystore.

Table 13-1 Keytool Commands (continued)

-noprompt	Specifies that no prompts are to be issued during an import operation.
-provider *provider_class_name*	Identifies the name of the cryptographic service provider's master class file when the service provider is not listed in the security properties file.
-rfc	Specifies that the output of the MD5 fingerprint of a certificate, which appears in printable-encoding format.
-storepass *storepass*	Identifies the password assigned to a keystore.
-sigalg *sigalg*	Specifies the algorithm to be used to sign the certificate.
-storetype *storetype*	Identifies the type assigned to a keystore or an entry into a keystore.
-trustcacerts	Specifies that additional certificates are considered for the chain of trust.
-v	Specifies that the output of the MD5 fingerprint of a certificate is printed in human-readable format.
-validity *valDays*	Identifies an expiration period. The default is 90 days.

Note For additional information on keytool and general certificate-management concepts, refer to Sun Microsystems documentation.

Generating Server Certificate Keystore and Private Key for a New Certificate

The **keytool -genkey** command generates a key pair (a public key and an associated private key), and wraps the public key into an X.509 self-signed certificate, which is stored as a single-element certificate chain. This certificate chain and the private key are stored in a new keystore entry identified by alias.

Note The name that you use for the alias is insignificant. Its purpose is to identify a key pair within the keystore if you had multiple key pairs. In the context of Cisco BAC, you should have only one key pair in the server certificate keystore, but you can have multiple keystores.

Example 13-1 Keytool -genkey

The following example uses *train-1.keystore* as the name of the keystore file. You can use any file name, but using *servercerts* is not recommended because this conflicts with the sample keystore that Cisco BAC provides.

```
dpe# ./keytool -keystore train-1.keystore -alias train-1 -genkey -keyalg RSA
Enter keystore password:  changeme
What is your first and last name?
  [Unknown]:  train-1.bac.test
What is the name of your organizational unit?
  [Unknown]:  BAC Training
What is the name of your organization?
  [Unknown]:  Acme Device, Inc.
What is the name of your City or Locality?
  [Unknown]:  Boxborough
What is the name of your State or Province?
```

```
  [Unknown]:  MA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=train-1.bac.test, OU=BAC Training, O="Acme Device, Inc.", L=Boxborough, ST=MA, C=US
correct?
  [no]:  yes

Enter key password for <train-1>
        (RETURN if same as keystore password):
```

In this example, *train-1.bac.test* goes into the CN field of the certificate and represents the FQDN that the ACS URL contains. According to the TR-069 specification, the device validates the certificate signature and ensures that the CN field in the certificate matches that of the URL it contacts.

Displaying Self-Signed Certificate

The **keytool -list** argument displays the contents of the keystore entry identified by alias. If you do not specify an alias, the entire contents of the keystore appear.

If you combine **-list** with **-v**, the certificate chain associated with the alias appears. The following **keytool -list** sample output displays the keystore containing a single self-signed certificate.

Example 13-2 Keytool -list

```
# ./keytool -keystore train-1.keystore -list -v
Enter keystore password:  changeme

Keystore type: jks
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: train-1
Creation date: Nov 8, 2005
Entry type: keyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=train-1.bac.test, OU=BAC Training, O="Acme Device, Inc.", L=Boxborough, ST=MA,
C=US
Issuer: CN=train-1.bac.test, OU=BAC Training, O="Acme Device, Inc.", L=Boxborough, ST=MA,
C=US
Serial number: 43714f22
Valid from: Tue Nov 08 20:21:38 EST 2005 until: Mon Feb 06 20:21:38 EST 2006
Certificate fingerprints:
         MD5:  CF:D4:CB:D1:20:6F:8C:12:ED:EA:2F:21:53:57:E5:1D
         SHA1: DD:AE:96:02:71:55:F8:1F:14:4F:D7:64:9C:FE:91:DE:65:C9:BB:49
```

Generating a Certificate-Signing Request

At this point in the procedure, the keystore contains a private key and a X.509 self-signed certificate. If the DPE ACS tries to respond with this certificate to a device's initial handshake, the device will reject the certificate with a TLS alert `bad CA`, indicating that the certificate authority that the CPE trusted did not sign the certificate. Therefore, the signing authority that the CPE trusts must sign the certificate.

Note To support SSL, the device must have a list of preconfigured public certificates of signing authorities that it trusts. One of the authorities that the device trusts must sign the ACS certificate.

The **keytool -certreq** command parameter generates a certificate-signing request (CSR). This command generates the CRS in the industry standard PKCS#10 format.

Example 13-3 Keytool -certreq

The following example uses a keystore with a preexisting self-signed certificate under **alias** *train-1* to generate a certificate-signing request and output the request into the `train-1.csr` file.

```
dpe# ./keytool -keystore train-1.keystore -alias train-1 -certreq -file train-1.csr
Enter keystore password:  changeme
```

The next step is to submit the CSR file to your signing authority. Your signing authority or your administrator, who is in possession of the private key for the signing authority, will generate a signed certificate based on this request. From the administrator, you must also obtain the public certificate of the signing authority.

Verifying the Signed Certificate

After you have received the signed certificate, use the **keytool -printcert** command to verify if the self-signed certificate is in the correct file format and uses the correct owner and issuer fields. The command reads the certificate from the **-file** *cert_file* parameter, and prints its contents in a human-readable format.

Example 13-4 Keytool -printcert

The *train-1.crt* file in this example identifies the signed certificate that the administrator provides.

```
dpe# ./keytool -printcert -file train-1.crt
Owner: CN=train-1.bac.test, OU=BAC Training, O="Acme Device, Inc.", ST=MA, C=US
Issuer: EMAILADDRESS=linksys-certadmin@cisco.com, CN=Acme Device Provisioning Root
Authority 1, OU=Acme Device Certificate Authority, O="Acme Device, Inc.", L=Irvine,
ST=California, C=US
Serial number: 1
Valid from: Tue Nov 08 12:40:28 EST 2005 until: Thu Nov 08 12:40:28 EST 2007
Certificate fingerprints:
         MD5:  25:8E:98:C5:5C:23:5C:A0:4D:51:CF:2A:AA:2A:FC:42
         SHA1: 05:C1:2D:C6:94:78:D1:40:88:6A:55:67:43:27:68:D3:AC:43:C6:A5
```

Note The keytool can print X.509 v1, v2, and v3 certificates, and PKCS#7-formatted certificate chains comprising certificates of that type. The data to be printed must be provided in binary-encoding format, or in printable-encoding format (also known as Base64 encoding) as defined by the RFC 1421.

Importing Signing Authority Certificate into Cacerts Keystore

Before importing the certificate into the server certificate keystore, you must import the public certificate of the signing authority into the cacerts keystore; because when a certificate is being imported into the keystore, the keytool checks if a chain of trust can be established between the certificate and its signing authority. If a chain of trust cannot be established, an error message appears.

Note The cacerts file bundled with Cisco BAC ships with several root certificate common third-party signing authorities. You can manage the cacerts keystore by using the keytool utility. The default cacerts keystore password is **changeit**. The cacerts database file resides in the *BPR_HOME/jre/lib/security* directory. The cacerts keystore does not need to be copied anywhere. The DPE will use the new keystore as soon as it is restarted.

Example 13-5 Keytool -import Signing Authority Certificate

```
# ./keytool -import -alias DeviceProvRoot -file rootCA4.crt -keystore
/opt/CSCObac/jre/lib/security/cacerts
Enter keystore password:  changeit
Owner: EMAILADDRESS=linksys-certadmin@cisco.com, CN=Acme Device Provisioning Root
Authority 1, OU=Acme Device Certificate Authority, O="Acme Device, Inc.", L=Irvine,
ST=California, C=US
Issuer: EMAILADDRESS=linksys-certadmin@cisco.com, CN=Acme Device Provisioning Root
Authority 1, OU=Acme Device Certificate Authority, O="Acme Device, Inc.", L=Irvine,
ST=California, C=US
Serial number: 8bcbc07a0768c1eb78e6c5c93c0c2ff0
Valid from: Fri Jul 01 21:22:12 EDT 2005 until: Mon Jun 29 21:22:12 EDT 2015
Certificate fingerprints:
         MD5:  C4:D4:09:6A:60:34:A0:00:96:4F:4D:47:23:86:8C:FA
         SHA1: B0:CC:6D:CD:BB:62:1B:A1:15:D3:2D:68:7E:D0:4A:0C:91:C2:A5:FD
Trust this certificate? [no]:  yes
Certificate was added to keystore.
```

Note The keytool can import X.509 v1, v2, and v3 certificates, and PKCS#7-formatted certificate chains comprising certificates of that type. The data to be imported must be provided in binary-encoding format, or in printable-encoding format (also known as Base64 encoding) as defined by the RFC 1421.

Importing the Signed Certificate into Server Certificate Keystore

Once you import the public certificate of the signing authority into the cacerts keystore, you must import the signed server certificate into the DPE server certificate keystore. You will already have a keystore with private key and corresponding self-signed certificate (public key).

By importing the certificate reply (signed certificate), the keystore is modified to associate the signed certificate with the existing private key in the server certificate keystore.

When importing the certificate reply into the keystore, you must use the **-trustcacerts** flag with the **-import** command for certificates in the *cacerts* file to be used to establish chains of trust with the certificate reply in the subject's keystore.

Example 13-6 Keytool -import (Signed Server Certificate)

```
dpe# ./keytool -import -trustcacerts -file train-1.crt -keystore train-1.keystore -alias
train-1
Enter key password: changeme2
Enter keystore password:  changeme
Certificate reply was installed in keystore.
Certificate was added to keystore.
```

After you import the signed server certificate into the DPE server certificate keystore, use the keytool **-printcert** command to verify the keystore contents, as outlined in Verifying the Signed Certificate, page 13-8. The **-printcert** output should now show the issuer to be the signing certificate authority, and that a chain of trust has been established using the signing authority with the root trusted certificate.

Importing Certificates for Client Authentication

This step is required only if you have configured client authentication by using client certificates on the DPE. If you have enabled client authentication by using client certificates, the cacerts keystore must contain the public certificate of the signing authority that signed the CPE client certificates. This certificate must be present to enable the DPE to validate certificates presented to it.

Example 13-7 Keytool -import (Signing Authority Certificate)

```
# ./keytool -import -alias DeviceClientRoot -file rootCA3.crt -keystore
/opt/CSCObac/jre/lib/security/cacerts
Enter keystore password:  changeit
Owner: EMAILADDRESS=linksys-certadmin@cisco.com, CN=Acme Device Client Root Authority 1,
OU=Acme Device Certificate Authority, O=Acme Device LLC., L=Irvine, ST=California, C=US
Issuer: EMAILADDRESS=linksys-certadmin@cisco.com, CN=Acme Device Client Root Authority 1,
OU=Acme Device Certificate Authority, O=Acme Device LLC., L=Irvine, ST=California, C=US
Serial number: d07d8a7badba7cb6446998b1ea89879f
Valid from: Fri Jul 01 21:19:50 EDT 2005 until: Mon Jun 29 21:19:50 EDT 2015
Certificate fingerprints:
         MD5:  40:B0:40:49:37:3A:51:1F:0D:78:B6:B3:E2:2C:1A:E8
         SHA1: 96:F5:84:71:84:CC:0A:A2:1E:7B:44:A2:B6:F5:B7:3D:C4:9F:81:3B
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Note This procedure is exactly the same as the one described in Importing Signing Authority Certificate into Cacerts Keystore, page 13-8. In both cases, you are loading the public certificate of the signing authority. If the signing authority of the server certificates is the same as the signing authority for the device certificates, you must add the certificate only once.

Providing the DPE with the service-provider keystore

Once you have a new service certificate keystore, which contains the signed public key certificate, you must move the keystore file to the DPE. The file must be moved to the *BPR_HOME/dpe/conf* directory.

Example 13-8 Move Keystore to DPE Configuration Directory

```
# mv train-1.keystore /opt/CSCObac/dpe/conf
```

Once you complete this step, you can configure the DPE services to use the new keystore by using the DPE CLI.

Note You do not need to copy the cacerts keystore anywhere. The DPE will use the new keystore as soon as it is restarted.

For more information, see Configuring Security for DPE Services, page 13-11. For additional information on DPE configuration commands, see the *Cisco Broadband Access Center 3.8 DPE CLI Reference*.

Configuring Security for DPE Services

This section describes how to configure authentication options and configure SSL on the DPE services.

You can configure DPE security options from the DPE CLI. For more information, see the *Cisco Broadband Access Center 3.8 DPE CLI Reference*.

The DPE supports running two CWMP and two HTTP file services concurrently. Each service can have a different configuration of security options and runs on a different port. By default, only one CWMP and one HTTP service are enabled, and they are configured without SSL. Two additional services are configured for SSL, but are disabled by default.

Table 13-2 specifies the defaults settings for each instance of the CWMP service and the HTTP file service:

Table 13-2 Default Settings for CWMP Technology

	CWMP Service		HTTP File Service	
	Service 1	**Service 2**	**File Service 1**	**File Service 2**
Default Mode	Enabled	Disabled	Enabled	Disabled
Authentication	Digest	Digest	Digest	Digest
Port Number	7547	7548	7549	7550
SSL Protocol	Disabled	Enabled	Disabled	Enabled

Configuring SSL on a DPE

To enable SSL on any given service:

Step 1 Configure HTTP client authentication. You can enable authentication in the Basic or Digest mode, or disable HTTP authentication.

Step 2 Enable the SSL protocol for the service.

Step 3 Configure the port through which the device contacts the service on the DPE.

Step 4 Set the keystore filename, keystore password, and key password.

Step 5 Configure client certificate authentication by using SSL. You can configure client authentication to use generic or unique client certificates.

Step 6 Optionally, disable other instances of the CWMP service or HTTP file service.

Step 7 Enable an instance of a service, which could be the CWMP service or the HTTP file service.

Step 8 Restart the DPE by using the **dpe reload** command to ensure that the changes take effect.

Enabling SSL for the CWMP Service

The following example describes the commands that you use to enable SSL for an instance of the CWMP service. In this example, double authentication is enabled for the SSL clients by using client certificates and HTTP authentication in Basic mode.

```
dpe# service cwmp 2 client-auth basic
```

```
% OK (Basic authentication was enabled. Digest authentication was disabled. Requires DPE
restart "> dpe reload")

dpe# service cwmp 2 port 7548
% OK (Requires DPE restart "> dpe reload")

dpe# service cwmp 2 ssl enable true
% OK (Requires DPE restart "> dpe reload")

dpe# service cwmp 2 ssl keystore train-1.keystore changeme changeme2
% OK (Requires DPE restart "> dpe reload")

dpe# service cwmp 2 ssl client-auth client-cert-unique
% OK (Requires DPE restart "> dpe reload")

dpe# service cwmp 1 enabled false
% OK (Requires DPE restart "> dpe reload")

dpe# service cwmp 2 enable true
% OK (Requires DPE restart "> dpe reload")

dpe# dpe reload
Process dpe has been restarted.

% OK
```

Note This example configures SSL transport on CWMP instance 2. The example assumes that the `train-1.keystore` is preloaded with the signed public key certificate for the server and that the keystore file was moved to the *BPR_HOME/dpe/conf* directory on the DPE.

Enabling SSL for the HTTP File Service

The following example describes the commands that you use to enable the SSL protocol for an instance of the HTTP file service. In this example, client authentication is disabled; thus allowing access without an authentication challenge.

```
dpe# service http 2 client-auth none
% OK (Requires DPE restart "> dpe reload")

dpe# service http 2 port 7550
% OK (Requires DPE restart "> dpe reload")

dpe# service http 2 ssl enable true
% OK (Requires DPE restart "> dpe reload")

dpe# service http 2 ssl keystore train-1.keystore changeme changeme2
% OK (Requires DPE restart "> dpe reload")

dpe# service http 2 ssl client-auth none
% OK (Requires DPE restart "> dpe reload")

dpe# service http 1 enable false
% OK (Requires DPE restart "> dpe reload")

dpe# service http 2 enable true
% OK (Requires DPE restart "> dpe reload")
```

```
dpe# dpe reload
Process dpe has been restarted.

% OK
```

Note For detailed information, see the *Cisco Broadband Access Center 3.8 DPE CLI Reference*.

Configuring CPE Authentication

CPE authentication is supported through:

- Shared secret by using HTTP Basic or Digest authentication.
- Client certificates by using SSL. You can use certificate-based client authentication in lieu of or in addition to shared secret HTTP-based authentication.
- External client certificate authentication. In this scenario, the SSL connection is terminated at the hardware load balancer, which also authenticates the client.

The objective of authentication is to establish a trusted device ID, which is used to look up device instructions in the DPE cache. This device ID correlates to the device identifier preprovisioned for the device record at the Cisco BAC RDU database. In case of shared secret HTTP-based authentication, the username serves to establish the identity of the device and is treated as a device ID.

In case of authentication using unique client certificates, the device ID is obtained from the device certificate's CN field. With client certificate authentication by an external entity (such as a Cisco ACE 4710 load balancer), the certificate data, including the CN field, is passed to the DPE in the HTTP headers.

You can also choose the option of having no client authentication if clients are trusted. For example, the clients can already be authenticated for network access based on the subscriber physical line. In this case, you can configure Cisco BAC with no-authentication and it will derive the trusted device ID from the Inform message.

You can configure CPE authentication options from the DPE CLI.

Shared Secret Authentication

Cisco BAC supports HTTP authentication based on a shared password between the CPE and the DPE. This authentication is available in two modes: Basic and Digest.

Note To limit security risks during client authentication, Cisco recommends using the Digest mode (the default configuration). You should not allow client authentication in the Basic mode.

To configure authentication in the Basic or Digest mode from the DPE CLI, use:

```
# service {cwmp | http} num client-auth mode
```

- *num*—Identifies an instance the service, which could be 1 or 2.
- *mode*—Identifies the client authentication mode for the service. The client authentication mode could be:
  - basic—Enables Basic HTTP authentication.
  - digest—Enables Digest HTTP authentication. This is the default configuration.
  - none—Disables Basic and Digest authentication.

For detailed information, see the *Cisco Broadband Access Center 3.8 DPE CLI Reference*.

Changing a Device Password

You can configure the shared secret for the device in Cisco BAC. The shared secret is stored on the device record by using the `IPDeviceKeys.CPE_PASSWORD` property. The CPE must prove knowledge of this password during HTTP-based authentication.

In the Basic mode, the password is transmitted as encoded clear text, while in the Digest mode, the device is allowed to prove knowledge of the password (shared secret) without transmitting it.

You can configure the password:

- On the API, using the property `IPDeviceKeys.CPE_PASSWORD`.
- On the administrator user interface, using the CPE Password field in the **Devices > Add Device** or **Modify Device** pages.

Note You cannot change the CPE password if the DPE connection to the RDU is not available.

The CPE password is optional if you have enabled client authentication by using SSL and with client certificates.

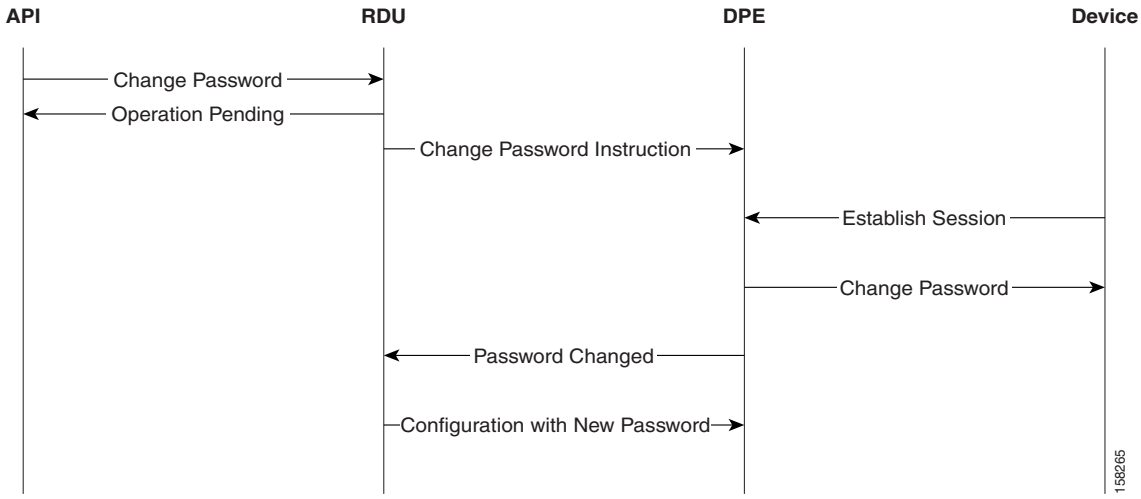
A distinction should be drawn between changing the password used by Cisco BAC and the password used by the device. When you configure a password on the device record in Cisco BAC, the outcome differs depending on the previous value of the password.

If the password was already set on the device record, Cisco BAC changes it and initiates the process of changing the password on the actual device. However, if the prior password value on the device record did not exist (or was an empty string), Cisco BAC sets the new password on the device record, but does not initiate the change of password on the actual device.

Hence, if you wish to change the password from the existing value to another value only in Cisco BAC, you must first reset the value in Cisco BAC (set it to an empty string) and then set it to a new value.

Figure 13-1 describes the process that Cisco BAC uses to change the password on the actual device. This process is complicated by the fact that Cisco BAC needs to use the old password to authenticate the device first, and then set the new password. Only after the password change is acknowledged can Cisco BAC forget the previous password and remove it from its database.

Figure 13-1 Change Password Flow



After the password is changed on the device object in the RDU, the change password instruction is included in the device configuration. At this point, the old password is still used to authenticate the device. The new configuration instruction is then forwarded to all the DPEs in the device's provisioning group.

When the device creates its next session with Cisco BAC, the device is authenticated with its old password; then, its new password is set using the SetParameterValues RPC. Once the DPE notifies the RDU of the change, the RDU changes the device password and generates a new configuration instruction, which includes the new password. Thereafter, the old password is no longer stored in Cisco BAC.

It is possible to change the new password up until it is set on the device. For example, if the original password of a device (*password*) has been changed to *newpassword*. But with the device yet to connect with Cisco BAC in order to set the new password, you can still change the password; for this purpose of this example, assume that the password is now changed to *n3wpa550d*.

The device continues to be authenticated with the old password (*password*) until the device connects to Cisco BAC. The new password (*n3wpa550d*) is set during the device's next session with Cisco BAC.

Correcting a Device Password

You can also change an incorrectly spelled password, in the RDU. For example, if you entered *passwrod* instead of *password*, first remove the password and then submit it. At this point, the device object does not have a password associated with it. Then, add the password on the device.

Note Use this method to change the password in Cisco BAC, but not the one on the device.

To correct a device password from the administrator user interface:

Step 1 From **Devices > Manage Devices**, click the Identifier link corresponding to the correct device.

The Modify Device page appears.

Step 2 Delete the device password in the CPE Password field.

Step 3 Click **Submit**.

Step 4 Return to the **Modify Device** page and enter the correct password.

Step 5 Click **Submit**.

The password is now changed on the device record in the RDU; no change of password on the actual device is initiated after this procedure.

Client Certificate Authentication

You can configure the DPE to require a validated device-provided certificate for authentication. These certificates could be:

- Generic—Enables device certificate authentication through SSL by using a generic certificate common to all CPE or a large subset of CPE.
- Unique—Enables client certificate authentication through SSL by using the unique certificate that each CPE provides.

If the device certificates are unique, it is not necessary to use HTTP authentication. But if the same certificate is used for all devices (that is, a single device certificate that a service provider uses), you should configure an additional HTTP authentication.

To configure client certification authentication from the DPE CLI, use:

```
# service {cwmp | http} num ssl client-auth mode
```

- *num*—Identifies the instance of the service, which could be 1 or 2. By default, client certificate authentication with SSL is:
  - Disabled for service 1.
  - Disabled for service 2.
- *mode*—Identifies the mode of client certificate authentication. Cisco BAC supports:
  - client-cert-generic—Uses a certificate common to a set of devices.
  - client-cert-unique—Uses a certificate unique to a device.
  - client-cert-ext—Uses a load balancer, such as ACE, to validate the client certificate.
  - none—Disables client certificate authentication.

External Client Certificate Authentication

An SSL connection can be terminated outside of the DPE, such as at a load balancer. In this case, the DPE may be configured without SSL service.

If a load balancer such as ACE, is used to terminate the SSL connections, the DPE does not receive the unique certificate, making it impossible to identify the device without HTTP authentication (Basic or Digest). To overcome this, the load balancer inserts additional HTTP headers that include the certificate information.

If a single session includes multiple TCP connections, each of the connections will be authenticated (if enabled). The session cookie binds the device to the existing session state.

Authentication Options in Cisco BAC

This section provides a summary of possible combinations of client-authentication options available in Cisco BAC for the CWMP and the HTTP file services. You can configure each instance of these services separately from the DPE CLI to suit your requirements.

Cisco BAC supports HTTP authentication in the Basic and Digest modes based on a shared password between the CPE and the DPE. If HTTP-based authentication is used, Cisco recommends use of the Digest mode.

You can also configure CPE authentication by using certificates unique to each CPE. In this case, HTTP authentication is not necessary. However, if you configure CPE authentication using generic certificates that are common to all CPE or a large subset of CPE, it is recommended that you configure the DPE to require an additional HTTP authentication.

Table 13-3 lists the various options that Cisco BAC supports and the commands you use to configure authentication from the DPE CLI. For details on each command, see the *Cisco Broadband Access Center 3.8 DPE CLI Reference*.

Table 13-3 Authentication Options in Cisco BAC

Option	Refer to ...
Using HTTP	
Enable device authentication by using HTTP in the Basic or Digest mode	**service** {**cwmp** | **http**} *num* **client-auth** {**basic** | **digest**}
Disable device authentication using HTTP	**service** {**cwmp** | **http**} *num* **client-auth none**
Note If device authentication using HTTP is disabled, trusted device identity is formed using the values in the Inform message from the device.	
Using SSL	
Enable device authentication in HTTP Basic or Digest mode over SSL connection but without client certificate authentication	**service** {**cwmp** | **http**} *num* **client-auth** {**basic** | **digest**} **service** {**cwmp** | **http**} *num* **ssl client-auth none**
Enable device authentication in HTTP Basic or Digest mode over SSL connection based on unique client certificates	**service** {**cwmp** | **http**} *num* **client-auth** {**basic** | **digest**} **service** {**cwmp** | **http**} *num* **ssl client-auth client-cert-unique**
Note When you configure the DPE to require HTTP-based (Basic or Digest) and authentication by using unique certificates, the device is authenticated by using both mechanisms. In this double authentication scenario, the device's unique identifier is formed by using the CN field of the client certificate; thus establishing trusted device ID.	
Enable device authentication in HTTP Basic or Digest mode over SSL connection based on generic client certificates	**service** {**cwmp** | **http**} *num* **client-auth** {**basic** | **digest**} **service** {**cwmp** | **http**} *num* **ssl client-auth client-cert-generic**

Table 13-3 Authentication Options in Cisco BAC (continued)

Option	Refer to ...
Enable device authentication by using HTTP Basic or Digest authentication and client certificate authentication by ACE	**service** {**cwmp**| **http**} *num* **client-auth** {**basic**	**digest**} **service** {**cwmp**	**http**} *num* **ssl client-auth client-cert-ext**
Note In this double authentication scenario, the trusted device ID is established based on authentication by ACE and communicated to the DPE using the HTTP headers.	
Enable device authentication based on client certificates validated by ACE	**service** {**cwmp**	**http**} *num* **ssl client-auth client-cert-ext**
Disable device authentication and client certificate authentication.	**service** {**cwmp**	**http**} *num* **client-auth none** **service** {**cwmp**	**http**} *num* **ssl client-auth none**
Note If device authentication and client-certificate authentication is disabled, devices are considered to be trusted or pre-authenticated, and the DPE uses data from the CWMP Inform message to establish trusted device identity.	
Disable HTTP-based device authentication and enable client authentication through SSL by using the unique certificate provided by each CPE	**service** {**cwmp**	**http**} *num* **client-auth none** **service** {**cwmp**	**http**} *num* **ssl client-auth client-cert-unique**
Disable HTTP-based device authentication and enable client authentication through SSL by using a generic certificate	**service** {**cwmp**	**http**} *num* **client-auth none** **service** {**cwmp**	**http**} *num* **ssl client-auth client-cert-generic**
Note If HTTP-based device authentication is disabled and client-certificate authentication is enabled to use generic certificates, devices are considered to be trusted or pre-authenticated and the DPE uses data from the CWMP Inform message to establish trusted device identity.	
Disable HTTP-based device authentication and enable authentication based on client certificates validated by ACE	**service** {**cwmp**	**http**} *num* **client-auth none** **service** {**cwmp**	**http**} *num* **ssl client-auth client-cert-ext**

Configuring Security for RDU Services

Cisco BAC supports local authentication as well as TACACS+ authentication. Local authentication is managed locally at the:

- DPE server, for DPE CLI
- RDU server, for the
  - Administrator user interface
  - Cisco BAC APIs

TACACS+ authentication relies on the TACACS+ protocol to support centrally managed user authentication using the TACACS+ server. The TACACS+ username, login password, and enable password are configurable at the TACACS+ server.

You can select one of the authentication modes by using the administrator user interface. Local authentication option is the default.

RDU Authentication Mode Settings

You can select either local or TACACS+ authentication by using the administrator user interface. To enable TACACS+ authentication:

Step 1 Choose **Configuration** on either the Primary Navigation bar or Main Menu page.

Step 2 Choose **Defaults** from the Secondary Navigation bar.

The Configure Defaults page appears.

Step 3 Click **TACACS+ Defaults** link on the left pane.

The TACACS+ Defaults page appears.

Step 4 Check the **TACACS+ Authentication** check box.

Step 5 Set the TACACS+ server and encryption key.

You can specify up to maximum of 5 TACACS+ servers. The order of the entries determines the order in which the TACACS+ servers are tried.

Step 6 Set the TACACS+ Client Read/Write timeout.

This is the time that the TACACS+ client waits for a TACACS+ server to reply to TACACS+ protocol requests. The range is from 1 to 300 seconds. The default is 5 seconds and applies to all TACACS+ servers.

Step 7 Set the TACACS+ Client Maximum retries.

This is the number of times that the TACACS+ client attempts a valid TACACS+ protocol exchange with a TACACS+ server if it fails to respond to initial request. The range is from 0 to 10. The default is 1 and applies to all TACACS+ server defined.

Step 8 Click **Submit**.

TACACS+ Authentication and Authorization in RDU

When TACACS+ authentication is enabled, the client attempts user login authentication to each server sequentially in the list until a successful authentication exchange is executed, or the list is exhausted. If the list is exhausted, the client automatically falls back into the local authentication mode (using the local system password).

After the TACACS+ authentication is done, the user authorization (i.e, user role as Admin, read-write user or read-only user) is retrieved from the RDU database. You can specify the user role (read-write or a read-only) in the Add Users page in the administrator user interface.

Signed Configuration for Devices

Cisco BAC uses the Signed Configuration feature to sign a portion of the CPE configuration that is targeted to be passed by the CPE to a third-party, such as an aggregation device. For example, a CPE with Femtocell functionality passes the access control entries to the Femtocell Gateway.

The configuration is signed using a secret key that is shared between Cisco BAC and the Gateway. This Signed Configuration eliminates the need to separately configure the Gateway for each individual CPE. The signature provides proof to the Gateway that the configuration:

- Was generated by Cisco BAC.
- Was not falsified during transmit.
- Is targeted for a specific CPE.
- Is targeted for a specific Gateway.
- Is current.

To prevent replay attacks, the time of the signature generation and its validity period are also incorporated in the signature.

Signature Expiration

Cisco BAC is configured with a signature validity period that determines the window during which the device communicates the signed data and signature to the Gateway. If the device communicates the data to the gateway beyond that validity period, the gateway rejects the signature.

The gateway can also be configured to cache configurations beyond their signature validity period, provided the device reconnects to the gateway within a certain interval. This behavior reduces the load of signature regenerations for active devices and allows the device to not persist a security-sensitive signature across reboots.

Signature Regeneration

The DPE generates a new signature and sets it on a device only if:

- The Signed Configuration parameters at the RDU are changed in the data model or in the configuration template.
- The device reports to the DPE that the Gateway rejected the signature for reasons such as invalidity or expiration.
- Device reports to the DPE that it does not have a signature.

Configuration Interfaces

You can enable the Signed Configuration feature in Cisco BAC by configuring the following properties:

- Signature Validity—Specifies the number of seconds for which the signature is considered valid
- Signature Key Name— Indicates the name of the key that is used by the gateway to look up the shared secret key. You must change the signature key name when the secret key is changed.
- Secret Key—Specifies the secret that is used to compute the signature.

These properties can be configured using the CWMP Defaults page in the administrator user interface or by using the *IPDevice.changeDefaults* API.

You must also create a configuration template for the device. To generate the Signed Configuration, at least one parameter in the template must be flagged to be signed. You can flag any TR069 parameter to be signed.

For example:

```
<Parameter>
<Name>MyParameter</Name>
<Value>Sample Value</Value>
<ToBeSigned>true</ToBeSigned>
</Parameter>
```

Depending on the target Gateway, there can be a minimum set of required parameters that must be specified as signed. Here is a sample CWMP configuration template:

```
<tc:Template
...
<configuration>
<ParameterDictionaries>
   <ParameterDictionary>femto-cwmp-dictionary.xml</ParameterDictionary>
</ParameterDictionaries>
<ObjectInstance name="Device">
  <ObjectInstance name="Services">
     <ObjectInstance name="X_00000C_FAPService">

     <ObjectInstance name="AccessControl">
        <Parameter>
           <Name>ACL</Name>
           <Value>VAR(name=FC-ACL, defaultValue="")</Value>
           <ToBeSigned>true</ToBeSigned>
        </Parameter>
       </ObjectInstance>

 <ObjectInstance name="FGW">
        <Parameter>
           <Name>Fqdn</Name>
           <Value>VAR(name=FC-FGW-FQDN, defaultValue="")</Value>
           <ToBeSigned>true</ToBeSigned>
        </Parameter>
      </ObjectInstance>
    </ObjectInstance>
  </ObjectInstance>
```

Monitoring the Signed Configuration Feature

You can monitor the Signed Configuration feature in Cisco BAC from the administrator user interface or DPE CLI. You have the following options:

- Choose **Configuration > Files**. Click the View icon () corresponding to the Configuration Template to view the configuration parameters that are designated to be signed.
- From the **Devices > Manage Devices** page, click the **View Details** icon () corresponding to the device. The following device faults are reported in the Device Details page:
  - Signature rejected by the gateway because the validity period has expired.
  - Signature rejected by the gateway because of an unknown secret key name
  - Signed data rejected by the gateway because it is missing required parameters

  - Incompatible device. Template contains parameters designated for signing, but the device does not report any expected support for the Signed Configuration feature

- To get basic information about the Signed Configuration process, enable the INFO logging level on the RDU and the DPEs. At this level of logging, the following are logged:
  - Changes in the Signed Configuration (in *rdu.log)*
  - Details on the generation and setting of a new signature on a device (in *dpe.log*)
  - Indications of signature rejection by the device (in *dpe.log*)
- You can use the **debug on** and **debug service cwmp** *num* **cpe-signed-config-sync** commands to obtain the low-level tracing information in addition to the INFO level logging.

Troubleshooting Signed Configuration Feature

When a device is placed in troubleshooting mode, all logging and debugging level details associated with Signature Generation are incorporated into the device troubleshooting log file. To view the device troubleshooting log file, see Viewing Device Troubleshooting Log, page 8-12.

You can use the *perfstat.log* to view statistics on the number of times that the DPE:

- Generates and sets the signature on a device.
- Receives signature faults from a device.

Custom Vendor Specific Attribute (VSA) in RADIUS Response

If custom vendor specific attribute (VSA) - in the form of VSA Index and VSA Name - is present in the Cisco Prime Access Registrar (PAR) server, this can be included in the RADIUS response. This is included in the authenticate requests by BAC DPE, which is also pushed to the Home-Node B (HNB) gateway as a response.

Adding Custom VSA

Note The process of adding custom VSA in PAR server should be done as part of maintenance activity and this should not be part of BAC running system. This is because adding custom VSA requires restart of the PAR server.

To add custom VSA, run the script **addCustomVSA.sh** in the server where PAR and PAR_EP is installed:

1. In the PAR (and PAR_EP) server, go to /opt/CSCObac/car_ep/bin
2. Run the script <*BPR_HOME*>**/car_ep/bin/addCustomVSA.sh**, and enter the corresponding values at the following prompts:
   - Enter VSA Name – Unique name for the VSA, and only letters [A-Z] and dashes [-] are allowed
   - Enter VSA Index – Unique index for the VSA
   - Enter Property Format – Depends on the VSA type also (STRING, INTEGER, or BYTE_ARRAY). See Property Format Based on Type, page 13-23.
   - Enter VSA Type – STRING, INTEGER, or BYTE_ARRAY

3. Restart the PAR server:

   /etc/init.d/arserver restart

Property Format Based on Type

Any device property can be included to construct the property of the VSA. But, the format to construct the property of the VSA depends on the data type of the VSA.

STRING

For STRING type, the device property format should contain the property name within '**${**' and '**}**' and can have any other string literals outside '${' and '}'. The value of this data type ranges between 0 and 253.

For example, if the property format is specified as **${FC-MNC}-${FC-MCC}**, and if FC-MNC = sid09869 and FC-MCC = etr123, then the value of corresponding VSA will be:

VSA Name = sid09869-etr123

INTEGER

For INTEGER type, the device property format should contain only one property. The value of this data type ranges between 0 and 4294967295.

For example, if the property format is specified as **FC-MNC**, and if FC-MNC = 253, then the value of corresponding VSA will be:

VSA Name = 253

BYTE_ARRAY

For BYTE_ARRAY type, the device property format should contain only one property. The value of this data type ranges between 0 and 253.

For example, if the property format is specified as **FC-MNC**, and if FC-MNC = f8:08:13:40:01:23:04:08:01:f6:08:13:40:01:22:04:18:21:f7:08:13:40, then the value of corresponding VSA will be:

VSA Name = f8:08:13:40:01:23:04:08:01:f6:08:13:40:01:22:04:18:21:f7:08:13:40

Note If the VSA Name is defined with an invalid data type or if the data type is empty, then the data type is defaulted to STRING.

Note If invalid values are configured for the Property Format, or if an empty string is configured for the Property Format, the Custom VSA value will be 0 (zero) for INTEGER type, and empty for STRING and BYTE_ARRAY types.