CHAPTER 49

Configuring Cisco Unified Presence

This chapter describes how to configure the adaptive security appliance for Cisco Unified Presence.

This chapter includes the following sections:

Information About Cisco Unified Presence

This section includes the following topics:

Architecture for Cisco Unified Presence

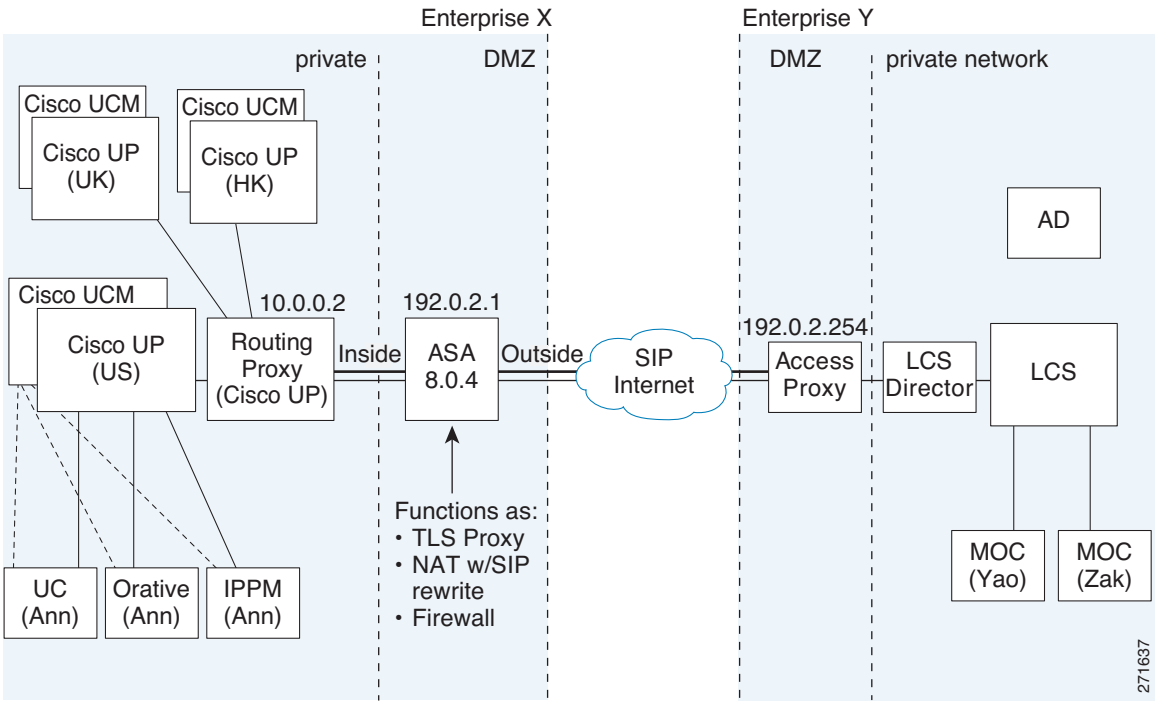
Figure 49-1 depicts a Cisco Unified Presence/LCS Federation scenario with the ASA as the presence federation proxy (implemented as a TLS proxy). The two entities with a TLS connection are the "Routing Proxy" (a dedicated Cisco UP) in Enterprise X and the Microsoft Access Proxy in Enterprise Y. However, the deployment is not limited to this scenario. Any Cisco UP or Cisco UP cluster could be deployed on the left side of the ASA; the remote entity could be any server (an LCS, an OCS, or another Cisco UP).

The following architecture is generic for two servers using SIP (or other ASA inspected protocols) with a TLS connection.

Entity X: Cisco UP/Routing Proxy in Enterprise X

Entity Y: Microsoft Access Proxy/Edge server for LCS/OCS in Enterprise Y

Figure 49-1 Typical Cisco Unified Presence/LCS Federation Scenario



In the above architecture, the ASA functions as a firewall, NAT, and TLS proxy, which is the recommended architecture. However, the ASA can also function as NAT and the TLS proxy alone, working with an existing firewall.

Either server can initiate the TLS handshake (unlike IP Telephony or Cisco Unified Mobility, where only the clients initiate the TLS handshake). There are by-directional TLS proxy rules and configuration. Each enterprise can have an ASA as the TLS proxy.

In Figure 49-1, NAT or PAT can be used to hide the private address of Entity X. In this situation, static NAT or PAT must be configured for foreign server (Entity Y) initiated connections or the TLS handshake (inbound). Typically, the public port should be 5061. The following static PAT command is required for the Cisco UP that accepts inbound connections:

```
hostname(config)# static (inside,outside) tcp 192.0.2.1 5061 10.0.0.2 5061 netmask
255.255.255.255
```

The following static PAT must be configured for each Cisco UP that could initiate a connection (by sending SIP SUBSCRIBE) to the foreign server.

For Cisco UP with the address 10.0.0.2, enter the following command:

```
hostname(config)# static (inside,outside) tcp 192.0.2.1 5062 10.0.0.2 5062 netmask
255.255.255.255
hostname(config)# static (inside,outside) udp 192.0.2.1 5070 10.0.0.2 5070 netmask
255.255.255.255
hostname(config)# static (inside,outside) tcp 192.0.2.1 5060 10.0.0.2 5060 netmask
255.255.255.255
```

For another Cisco UP with the address 10.0.0.3, you must use a different set of PAT ports, such as 45062 or 45070:

```
hostname(config)# static (inside,outside) tcp 192.0.2.1 45061 10.0.0.3 5061 netmask
255.255.255.255
```

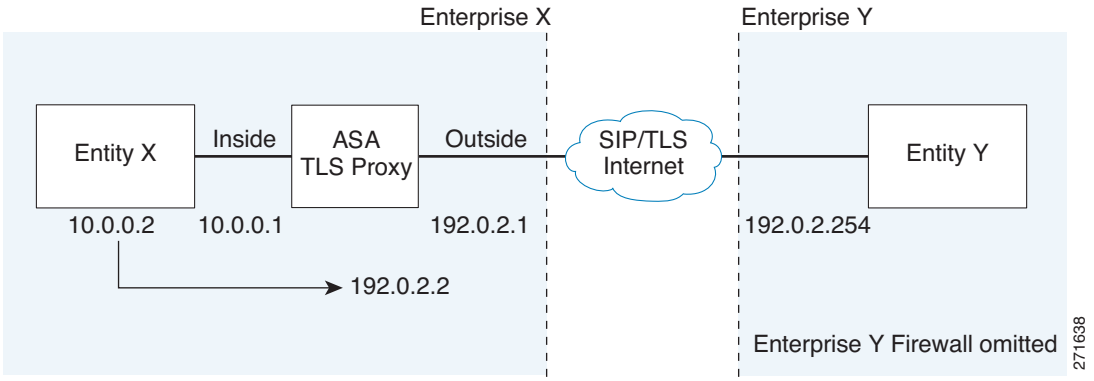
```
hostname(config)# static (inside,outside) tcp 192.0.2.1 45062 10.0.0.3 5062 netmask
255.255.255.255
hostname(config)# static (inside,outside) udp 192.0.2.1 45070 10.0.0.3 5070 netmask
255.255.255.255
hostname(config)# static (inside,outside) tcp 192.0.2.1 5070 10.0.0.2 5070 netmask
255.255.255.255
hostname(config)# static (inside,outside) tcp 192.0.2.1 45060 10.0.0.3 5060 netmask
255.255.255.255
```

Dynamic NAT or PAT can be used for the rest of the outbound connections or the TLS handshake. The ASA SIP inspection engine takes care of the necessary translation (fixup).

```
hostname(config)# global (outside) 102 192.0.2.1 netmask 255.255.255.255
hostname(config)# nat (inside) 102 0.0.0.0 0.0.0.0
```

Figure 49-2 illustrates an abstracted scenario with Entity X connected to Entity Y through the presence federation proxy on the ASA. The proxy is in the same administrative domain as Entity X. Entity Y could have another ASA as the proxy but this is omitted for simplicity.

Figure 49-2 Abstracted Presence Federation Proxy Scenario between Two Server Entities



For the Entity X domain name to be resolved correctly when the ASA holds its credential, the ASA could be configured to perform NAT for Entity X, and the domain name is resolved as the Entity X public address for which the ASA provides proxy service.

Trust Relationship in the Presence Federation

Within an enterprise, setting up a trust relationship is achievable by using self-signed certificates or you can set it up on an internal CA.

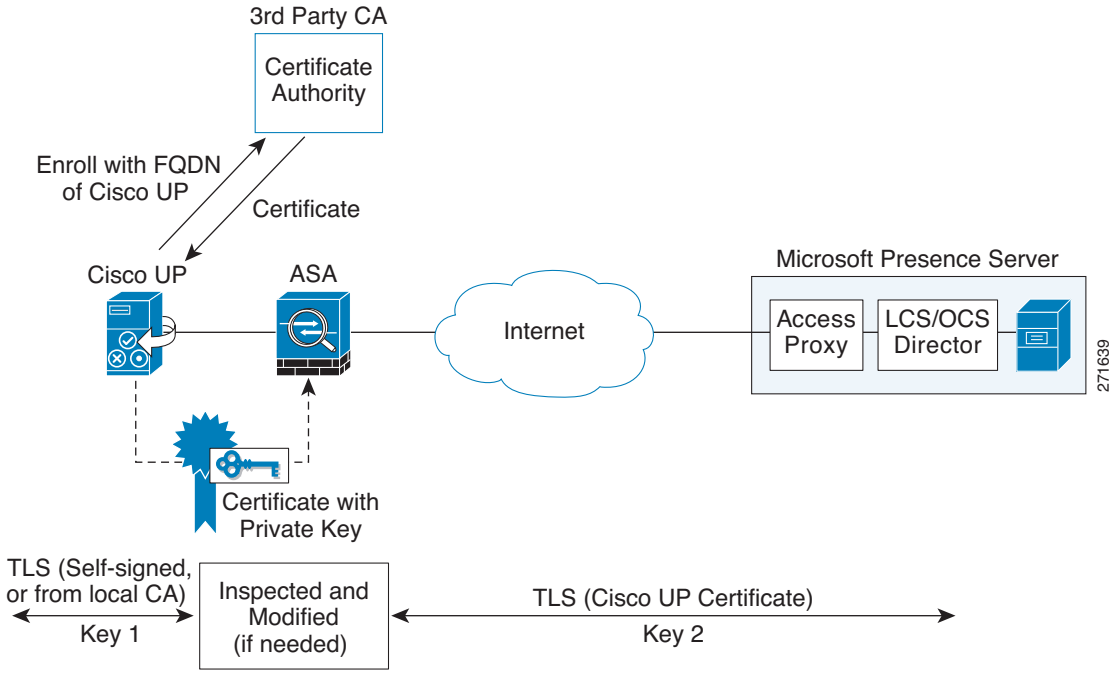
Establishing a trust relationship cross enterprises or across administrative domains is key for federation. Cross enterprises you must use a trusted third-party CA (such as, VeriSign). The ASA obtains a certificate with the FQDN of the Cisco UP (certificate impersonation).

For the TLS handshake, the two entities could validate the peer certificate via a certificate chain to trusted third-party certificate authorities. Both entities enroll with the CAs. The ASA as the TLS proxy must be trusted by both entities. The ASA is always associated with one of the enterprises. Within that enterprise (Enterprise X in Figure 49-1), the entity and the ASA could authenticate each other via a local CA, or by using self-signed certificates.

To establish a trusted relationship between the ASA and the remote entity (Entity Y), the ASA can enroll with the CA on behalf of Entity X (Cisco UP). In the enrollment request, the Entity X identity (domain name) is used.

Figure 49-3 shows the way to establish the trust relationship. The ASA enrolls with the third party CA by using the Cisco UP FQDN as if the ASA is the Cisco UP.

Figure 49-3 How the Security Appliance Represents Cisco Unified Presence – Certificate Impersonate



Security Certificate Exchange Between Cisco UP and the Security Appliance

You need to generate the keypair for the certificate (such as `cup_proxy_key`) used by the ASA, and configure a trustpoint to identify the self-signed certificate sent by the ASA to Cisco UP (such as `cup_proxy`) in the TLS handshake.

For the ASA to trust the Cisco UP certificate, you need to create a trustpoint to identify the certificate from the Cisco UP (such as `cert_from_cup`), and specify the enrollment type as terminal to indicate that you will paste the certificate received from the Cisco UP into the terminal.

Licensing for Cisco Unified Presence

The Cisco Unified Presence feature supported by the ASA require a Unified Communications Proxy license.

The Cisco Unified Presence feature is licensed by TLS session. For the federation proxy, each endpoint utilizes one Unified Communications Proxy session.

Table 49-1 shows the Unified Communications Proxy license details by platform.

Table 49-1 License Requirements for the Security Appliance

Security Appliance Platform	Max UC Proxy Licenses	Tiers for UC Proxy Licenses
ASA 5505	24	24
ASA 5510	100	24, 50, 100
ASA 5520	1,000	24, 50, 100, 250, 500, 750, 1000
ASA 5540	2,000	24, 50, 100, 250, 500, 750, 1000, 2000
ASA 5550	3,000	24, 50, 100, 250, 500, 750, 1000, 2000, 3000
ASA 5580	10,000	24, 50, 100, 250, 500, 750, 1000, 2000, 3000, 5000, 10000

A Unified Communications Proxy license is applied the same way as other licensed features (such as, SSL VPN), via the **activation-key** command. For more information about licensing, see Chapter 3, "Managing Feature Licenses."

Configuring Cisco Unified Presence

This section contains the following topics:

Task Flow for Configuring Cisco Unified Presence

To configure a Cisco Unified Presence/LCS Federation scenario with the ASA as the TLS proxy where there is a single Cisco UP that is in the local domain and self-signed certificates are used between the Cisco UP and the ASA (like the scenario shown in Figure 49-1), perform the following tasks.

Step 1 Create the following static NAT for the local domain containing the Cisco UP.

For the inbound connection to the local domain containing the Cisco UP, create static PAT by entering the following command:

```
hostname(config)# static (real_ifc,mapped_ifc) tcp mapped_ip mapped_port netmask mask
```

Note For each Cisco UP that could initiate a connection (by sending SIP SUBSCRIBE) to the foreign server, you must also configure static PAT by using a different set of PAT ports.

For outbound connections or the TLS handshake, use dynamic NAT or PAT. The ASA SIP inspection engine takes care of the necessary translation (fixup).

```
hostname(config)# global (mapped_ifc) nat_id mapped_ip netmask mask
```

```
hostname(config)# nat (real_ifc) nat_id real_ip mask
```

Step 2 Create the necessary RSA keypairs and proxy certificate, which is a self-signed certificate, for the remote entity. See Creating Trustpoints and Generating Certificates, page 49-6.

Step 3 Install the certificates. See Installing Certificates, page 49-7.

Step 4 Create the TLS proxy instance for the Cisco UP clients connecting to the Cisco UP server. See Creating the TLS Proxy Instance, page 49-8.

Step 5 Enable the TLS proxy for SIP inspection. See Enabling the TLS Proxy for SIP Inspection, page 49-9.

Creating Trustpoints and Generating Certificates

You need to generate the keypair for the certificate (such as `cup_proxy_key`) used by the ASA, and configure a trustpoint to identify the self-signed certificate sent by the ASA to Cisco UP (such as `cup_proxy`) in the TLS handshake.

	Command	Purpose

Step 1	`hostname(config)# crypto key generate rsa label key-pair-label modulus size`
	Example:
	`crypto key generate rsa label ent_y_proxy_key modulus 1024`
	`INFO: The name for the keys will be: ent_y_proxy_key`
	`Keypair generation process begin. Please wait...`
	`hostname(config)#`
		Creates the RSA keypair that can be used for the trustpoints.
		The keypair is used by the self-signed certificate presented to the local domain containing the Cisco UP (proxy for the remote entity).

Step 2	`hostname(config)# crypto ca trustpoint trustpoint_name`
	Example:
	`hostname(config)# crypto ca trustpoint ent_y_proxy`
		Enters the trustpoint configuration mode for the specified trustpoint so that you can create the trustpoint for the remote entity.
		A trustpoint represents a CA identity and possibly a device identity, based on a certificate issued by the CA.

Step 3	`hostname(config-ca-trustpoint)# enrollment self`	Generates a self-signed certificate.
Step 4	`hostname(config-ca-trustpoint)# fqdn none`	Specifies not to include a fully qualified domain name (FQDN) in the Subject Alternative Name extension of the certificate during enrollment.

Step 5	`hostname(config-ca-trustpoint)# subject-name X.500_name`
	Example:
	`hostname(config-ca-trustpoint)# subject-name cn=Ent-Y-Proxy`
		Includes the indicated subject DN in the certificate during enrollment

Step 6	`hostname(config-ca-trustpoint)# keypair keyname`
	Example:
	`hostname(config-ca-trustpoint)# keypair ent_y_proxy_key`
		Specifies the key pair whose public key is to be certified.

Step 7	`hostname(config-ca-trustpoint)# exit`	Exits from the CA Trustpoint configuration mode.

Step 8	`hostname(config)# crypto ca enroll trustpoint`
	Example:
	`hostname(config)# crypto ca enroll ent_y_proxy`
		Starts the enrollment process with the CA and specifies the name of the trustpoint to enroll with.

What to Do Next

Install the certificate on the local entity truststore. You could also enroll the certificate with a local CA trusted by the local entity. See Installing Certificates, page 49-7.

Installing Certificates

Export the self-signed certificate for the ASA created in Creating Trustpoints and Generating Certificates, page 49-6 and install it as a trusted certificate on the local entity. This task is necessary for local entity to authenticate the ASA.

Prerequisites

To create a proxy certificate on the ASA that is trusted by the remote entity, obtain a certificate from a trusted CA. For information about obtaining a certificate from a trusted CA, see Configuring Digital Certificates, page 73-8.

	Command	Purpose

Step 1	`hostname(config)# crypto ca export` *trustpoint* `identity-certificate`
	Example:
	`hostname(config)# crypto ca export ent_y_proxy identity-certificate`
		Export the ASA self-signed (identity) certificate.

Step 2	`hostname(config)# crypto ca trustpoint` *trustpoint_name*
	Example:
	`hostname(config)# crypto ca trustpoint ent_x_cert`
	`! for Entity X's self-signed certificate`
		Enters the trustpoint configuration mode for the specified trustpoint so that you can create the trustpoint for the local entity.
		A trustpoint represents a CA identity and possibly a device identity, based on a certificate issued by the CA.

Step 3	`hostname(config-ca-trustpoint)# enrollment terminal`	Specifies cut and paste enrollment with this trustpoint (also known as manual enrollment). If the local entity uses a self-signed certificate, the self-signed certificate must be installed; if the local entity uses a CA-issued certificate, the CA certificate needs to be installed. This configuration shows the commands for using a self-signed certificate.
Step 4	`hostname(config-ca-trustpoint)# exit`	Exits from the CA Trustpoint configuration mode.

Step 5	`hostname(config)# crypto ca authenticate` *trustpoint*
	Example:
	`hostname(config)# crypto ca authenticate ent_x_cert`
	`Enter the base 64 encoded CA certificate.`
	`End with a blank line or the word "quit" on a line by itself`
	`[ certificate data omitted ]`
	`Certificate has the following attributes:`
	`Fingerprint: 21B598D5 4A81F3E5 0B24D12E 3F89C2E4`
	`% Do you accept this certificate? [yes/no]: yes`
	`Trustpoint CA certificate accepted.`
	`% Certificate successfully imported`
		Installs and authenticates the CA certificates associated with a trustpoint created for the local entity.
		Where *trustpoint* specifies the trustpoint from which to obtain the CA certificate. Maximum name length is 128 characters.
		The ASA prompts you to paste the base-64 formatted CA certificate onto the terminal.

	Command	Purpose
Step 6	hostname(config)# **crypto ca trustpoint** *trustpoint_name* **Example:** hostname(config)# crypto ca trustpoint ent_y_ca ! for Entity Y's CA certificate	Install the CA certificate that signs the remote entity certificate on the ASA by entering the following commands. This step is necessary for the ASA to authenticate the remote entity.
Step 7	hostname(config-ca-trustpoint)# **enrollment terminal**	Specifies cut and paste enrollment with this trustpoint (also known as manual enrollment).
Step 8	hostname(config-ca-trustpoint)# **exit**	Exits from the CA Trustpoint configuration mode.
Step 9	hostname(config)# **crypto ca authenticate** *trustpoint* **Example:** hostname(config)# crypto ca authenticate ent_y_ca Enter the base 64 encoded CA certificate. End with a blank line or the word "quit" on a line by itself MIIDRTCCAu+gAwIBAgIQKVcqP/KW74VP0NZzL+JbRTANBgkqhkiG9w0BAQUFADCB [certificate data omitted] /7QEM8izy0EOTSErKu7Nd76jwf5e4qttkQ==	Installs and authenticates the CA certificates associated with a trustpoint created for the local entity. The ASA prompts you to paste the base-64 formatted CA certificate onto the terminal.

What to Do Next

Once you have created the trustpoints and installed the certificates for the local and remote entities on the ASA, create the TLS proxy instance. See Creating the TLS Proxy Instance, page 49-8.

Creating the TLS Proxy Instance

Because either server can initiate the TLS handshake (unlike IP Telephony or Cisco Unified Mobility, where only the clients initiate the TLS handshake), you must configure by-directional TLS proxy rules. Each enterprise can have an ASA as the TLS proxy.

Create TLS proxy instances for the local and remote entity initiated connections respectively. The entity that initiates the TLS connection is in the role of "TLS client". Because the TLS proxy has a strict definition of "client" and "server" proxy, two TLS proxy instances must be defined if either of the entities could initiate the connection.

	Command	Purpose
Step 1	! Local entity to remote entity hostname(config)# **tls-proxy** proxy_name **Example:** hostname(config)# tls-proxy ent_x_to_y	Creates the TLS proxy instance.
Step 2	hostname(config-tlsp)# **server trust-point** *proxy_name* **Example:** hostname(config-tlsp)# server trust-point ent_y_proxy	Specifies the proxy trustpoint certificate presented during TLS handshake. The certificate must be owned by the ASA (identity certificate). Where the *proxy_name* for the **server trust-point** command is the remote entity proxy name.

	Command	Purpose
Step 3	hostname(config-tlsp)# **client trust-point** *proxy_trustpoint* **Example:** hostname(config-tlsp)# client trust-point ent_x_cert	Specifies the trustpoint and associated certificate that the ASA uses in the TLS handshake when the ASA assumes the role of the TLS client. The certificate must be owned by the ASA (identity certificate). Where the *proxy_trustpoint* for the **client trust-point** command is the local entity proxy.
Step 4	hostname(config-tlsp)# **client cipher-suite** *cipher_suite* **Example:** hostname(config-tlsp)# client cipher-suite aes128-sha1 aes256-sha1 3des-sha1 null-sha1	Specifies cipher suite configuration. For client proxy (the proxy acts as a TLS client to the server), the user-defined cipher suite replaces the default cipher suite.
Step 5	! Remote entity to local entity hostname(config)# **tls-proxy** *proxy_name* **Example:** tls-proxy ent_y_to_x	Creates the TLS proxy instance.
Step 6	hostname(config-tlsp)# **server trust-point** *proxy_name* **Example:** hostname(config-tlsp)# server trust-point ent_x_cert	Specifies the proxy trustpoint certificate presented during TLS handshake. Where the *proxy_name* for the **server trust-point** command is the local entity proxy name
Step 7	hostname(config-tlsp)# **client trust-point** *proxy_trustpoint* **Example:** hostname(config-tlsp)# client trust-point ent_y_proxy	Specifies the trustpoint and associated certificate that the ASA uses in the TLS handshake when the ASA assumes the role of the TLS client. Where the *proxy_trustpoint* for the **client trust-point** command is the remote entity proxy.
Step 8	hostname(config-tlsp)# **client cipher-suite** *cipher_suite* **Example:** hostname(config-tlsp)# client cipher-suite aes128-sha1 aes256-sha1 3des-sha1 null-sha1	Specifies cipher suite configuration.

What to Do Next

Once you have created the TLS proxy instance, enable it for SIP inspection. See Enabling the TLS Proxy for SIP Inspection, page 49-9.

Enabling the TLS Proxy for SIP Inspection

Enable the TLS proxy for SIP inspection and define policies for both entities that could initiate the connection. For more information about SIP application inspection,

	Command	Purpose
Step 1	hostname(config)# **access-list** *id* **extended permit tcp host** *src_ip* **host** *dest_ip* **eq** *port* **Examples:** access-list ent_x_to_y extended permit tcp host 10.0.0.2 host 192.0.2.254 eq 5061 access-list ent_y_to_x extended permit tcp host 192.0.2.254 host 192.0.2.1 eq 5061	Adds an Access Control Entry. The access list is used to specify the class of traffic to inspect.
Step 2	hostname(config)# **class-map** *class_map_name* **Example:** hostname(config)# class-map ent_x_to_y	Configures the secure SIP class of traffic to inspect. Where *class_map_name* is the name of the SIP class map.
Step 3	hostname(config-cmap)# **match access-list** *access_list_name* **Example:** hostname(config-cmap)# match access-list ent_x_to_y	Identifies the traffic to inspect.
Step 4	hostname(config-cmap)# **exit**	Exits from Class Map configuration mode.
Step 5	hostname(config)# **policy-map type inspect sip** *policy_map_name* **Example:** hostname(config)# policy-map type inspect sip sip_inspect	Defines special actions for SIP inspection application traffic.
Step 6	hostname(config-pmap)# **parameters** ! SIP inspection parameters	Specifies the parameters for SIP inspection. Parameters affect the behavior of the inspection engine. The commands available in parameters configuration mode depend on the application.
Step 7	hostname(config-pmap)# **exit**	Exits from Policy Map configuration mode.
Step 8	hostname(config)# **policy-map** name **Example:** hostname(config)# policy-map global_policy	Configure the policy map and attach the action to the class of traffic.
Step 9	hostname(config-pmap)# **class** classmap_*name* **Example:** hostname(config-pmap)# class ent_x_to_y	Assigns a class map to the policy map so that you can assign actions to the class map traffic. Where *classmap_name* is the name of the SIP class map.
Step 10	hostname(config-pmap)# **inspect sip** *sip_map* **tls-proxy** *proxy_name* hostname(config-pmap)# inspect sip sip_inspect tls-proxy ent_x_to_y	Enables TLS proxy for the specified SIP inspection session.
Step 11	hostname(config-pmap)# **exit**	Exits from Policy Map configuration mode.
Step 12	hostname(config)# **service-policy** *policy_map_name* **global** **Example:** hostname(config)# service-policy global_policy global	Enables the service policy for SIP inspection for all interfaces.. Where name for the policy-map command is the name of the global policy map.

Monitoring Cisco Unified Presence

Debugging is similar to debugging TLS proxy for IP Telephony. You can enable TLS proxy debug flags along with SSL syslogs to debug TLS proxy connection problems.

For example, use the following commands to enable TLS proxy-related debug and syslog output only:

```
hostname(config)# debug inspect tls-proxy events
hostname(config)# debug inspect tls-proxy errors
hostname(config)# logging enable
hostname(config)# logging timestamp
hostname(config)# logging list loglist message 711001
hostname(config)# logging list loglist message 725001-725014
hostname(config)# logging list loglist message 717001-717038
hostname(config)# logging buffer-size 1000000
hostname(config)# logging buffered loglist
hostname(config)# logging debug-trace
```

For information about TLS proxy debugging techniques and sample output, see Monitoring the TLS Proxy, page 47-15.

Enable the **debug sip** command for SIP inspection engine debugging. See the *Cisco ASA 5500 Series Command Reference*.

Additionally, you can capture the raw and decrypted data by the TLS proxy by entering the following commands:

```
hostname# capture mycap interface outside (capturing raw packets)
hostname# capture mycap-dec type tls-proxy interface outside (capturing decrypted data)
hostname# show capture capture_name
hostname# copy /pcap capture:capture_name tftp://tftp_location
```

Configuration Example for Cisco Unified Presence

The following sample illustrates the necessary configuration for the ASA to perform TLS proxy for Cisco Unified Presence as shown in Figure 49-4. It is assumed that a single Cisco UP (Entity X) is in the local domain and self-signed certificates are used between Entity X and the ASA.

For each Cisco UP that could initiate a connection (by sending SIP SUBSCRIBE) to the foreign server, you must also configure static PAT and if you have another Cisco UP with the address (10.0.0.3 in this sample), it must use a different set of PAT ports (such as 45062 or 45070). Dynamic NAT or PAT can be used for outbound connections or TLS handshake. The ASA SIP inspection engine takes care of the necessary translation (fixup).

When you create the necessary RSA key pairs, a key pair is used by the self-signed certificate presented to Entity X (proxy for Entity Y). When you create a proxy certificate for Entity Y, the certificate is installed on the Entity X truststore. It could also be enrolled with a local CA trusted by Entity X.

Exporting the ASA self-signed certificate (ent_y_proxy) and installing it as a trusted certificate on Entity X is necessary for Entity X to authenticate the ASA. Exporting the Entity X certificate and installing it on the ASA is needed for the ASA to authenticate Entity X during handshake with X. If Entity X uses a self-signed certificate, the self-signed certificate must be installed; if Entity X uses a CA issued the certificate, the CA's certificated needs to be installed.

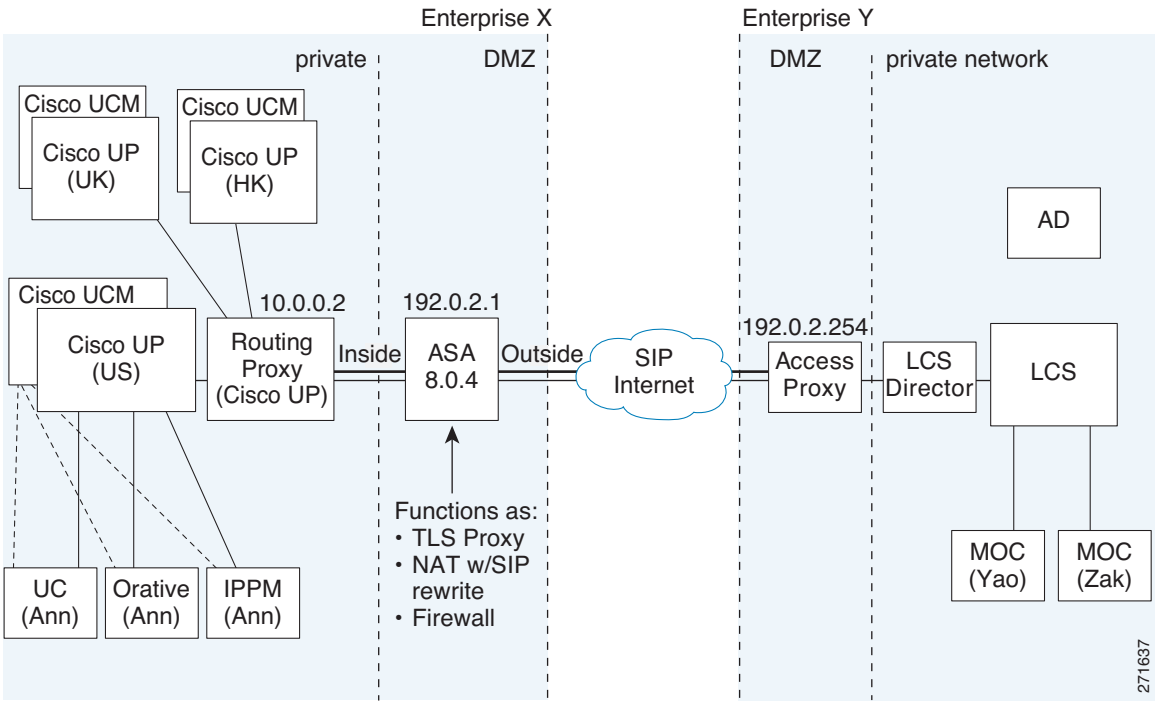
For about obtaining a certificate from a trusted CA, see Configuring Digital Certificates, page 73-8.

Installing the CA certificate that signs the Entity Y certificate on the ASA is necessary for the ASA to authenticate Entity Y.

When creating TLS proxy instances for Entity X and Entity Y, the entity that initiates the TLS connection is in the role of "TLS client". Because the TLS proxy has strict definition of "client" and "server" proxy, two TLS proxy instances must be defined if either of the entities could initiate the connection.

When enabling the TLS proxy for SIP inspection, policies must be defined for both entities that could initiate the connection.

Figure 49-4 Typical Cisco Unified Presence/LCS Federation Scenario




```
static (inside,outside) tcp 192.0.2.1 5061 10.0.0.2 5061 netmask 255.255.255.255
static (inside,outside) tcp 192.0.2.1 5062 10.0.0.2 5062 netmask 255.255.255.255
static (inside,outside) udp 192.0.2.1 5070 10.0.0.2 5070 netmask 255.255.255.255
static (inside,outside) tcp 192.0.2.1 45062 10.0.0.3 5062 netmask 255.255.255.255
static (inside,outside) udp 192.0.2.1 45070 10.0.0.3 5070 netmask 255.255.255.255
global (outside) 102 192.0.2.1 netmask 255.255.255.255
nat (inside) 102 0.0.0.0 0.0.0.0
crypto key generate rsa label ent_y_proxy_key modulus 1024
    ! for self-signed Entity Y proxy certificate
crypto ca trustpoint ent_y_proxy
    enrollment self
    fqdn none
    subject-name cn=Ent-Y-Proxy
    keypair ent_y_proxy_key
crypto ca enroll ent_y_proxy
crypto ca export ent_y_proxy identity-certificate
! for Entity X's self-signed certificate
crypto ca trustpoint ent_x_cert
    enrollment terminal
crypto ca authenticate ent_x_cert
Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself
    [ certificate data omitted ]
quit
! for Entity Y's CA certificate
crypto ca trustpoint ent_y_ca
    enrollment terminal
crypto ca authenticate ent_y_ca
Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself
MIIDRTCCAu+gAwIBAgIQKVcqP/KW74VP0NZzL+JbRTANBgkqhkiG9w0BAQUFADCB
    [ certificate data omitted ]
/7QEM8izy0EOTSErKu7Nd76jwf5e4qttkQ==
quit
```

```
! Entity X to Entity Y
tls-proxy ent_x_to_y
    server trust-point ent_y_proxy
    client trust-point ent_x_cert
    client cipher-suite aes128-sha1 aes256-sha1 3des-sha1 null-sha1
! Entity Y to Entity X
tls-proxy ent_y_to_x
    server trust-point ent_x_cert
    client trust-point ent_y_proxy
    client cipher-suite aes128-sha1 aes256-sha1 3des-sha1 null-sha1
access-list ent_x_to_y extended permit tcp host 10.0.0.2 host 192.0.2.254 eq 5061
access-list ent_y_to_x extended permit tcp host 192.0.2.254 host 192.0.2.1 eq 5061
class-map ent_x_to_y
    match access-list ent_x_to_y
class-map ent_y_to_x
    match access-list ent_y_to_x
policy-map type inspect sip sip_inspect
    parameters
        ! SIP inspection parameters
policy-map global_policy
    class ent_x_to_y
        inspect sip sip_inspect tls-proxy ent_x_to_y
    class ent_y_to_x
        inspect sip sip_inspect tls-proxy ent_y_to_x
service-policy global_policy global
```

Feature History for Cisco Unified Presence

Table 49-2 lists the release history for this feature.

Table 49-2 Feature History for Cisco Phone Proxy

Feature Name	Releases	Feature Information
Cisco Unified Presence	8.0(4)	The Presence Federation Proxy feature was introduced.