Line-Side Support for Cisco Unified Communications Manager

Cisco Unified Communications Manager is an enterprise-class IP communications processing system. It extends enterprise telephony features and capabilities to IP phones, media processing devices, VoIP gateways, mobile devices, and multimedia applications. Cisco Unified Border Element (SP Edition) provides line-side support for Cisco Unified Communications Manager. This support enables phones used by remote users to communicate with phones on the organizational network.

Cisco Unified Border Element (SP Edition) was formerly known as Integrated Session Border Controller and may be commonly referred to in this document as the session border controller (SBC).

For a complete description of the commands used in this chapter, refer to *Cisco Unified Border Element (SP Edition) Command Reference: Unified Model* at:

http://www.cisco.com/en/US/docs/ios/sbc/command/reference/sbcu_book.html

For information about all the Cisco IOS commands, use the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or the Cisco IOS master commands list.

Note The Line-Side Support for Cisco Unified Communications Manager feature is supported in the unified model for Cisco IOS XE Release 3.5S and later releases.

Feature History for Line-Side Support for Cisco Unified Communications Manager

Release	Modification
Cisco IOS XE Release 3.5S	The Line-Side Support for Cisco Unified Communications Manager feature was introduced.

Contents

This chapter contains the following sections:

About the Line-Side Support for Cisco Unified Communications Manager Feature

Cisco Unified Communications Manager is used to manage a cluster of IP phones on the network of an organization. It can operate in one of the following modes:

- Nonsecure mode—In this mode, devices with nonsecure profiles and Real-Time Transport Protocol (RTP) media can connect to Cisco Unified Communications Manager.
- Mixed mode—In this mode, devices with nonsecure or secure profiles and RTP or Secure Real-Time Transport Protocol (SRTP) media can connect to Cisco Unified Communications Manager.

For more information about Cisco Unified Communications Manager, see the documentation for this product at the following location:

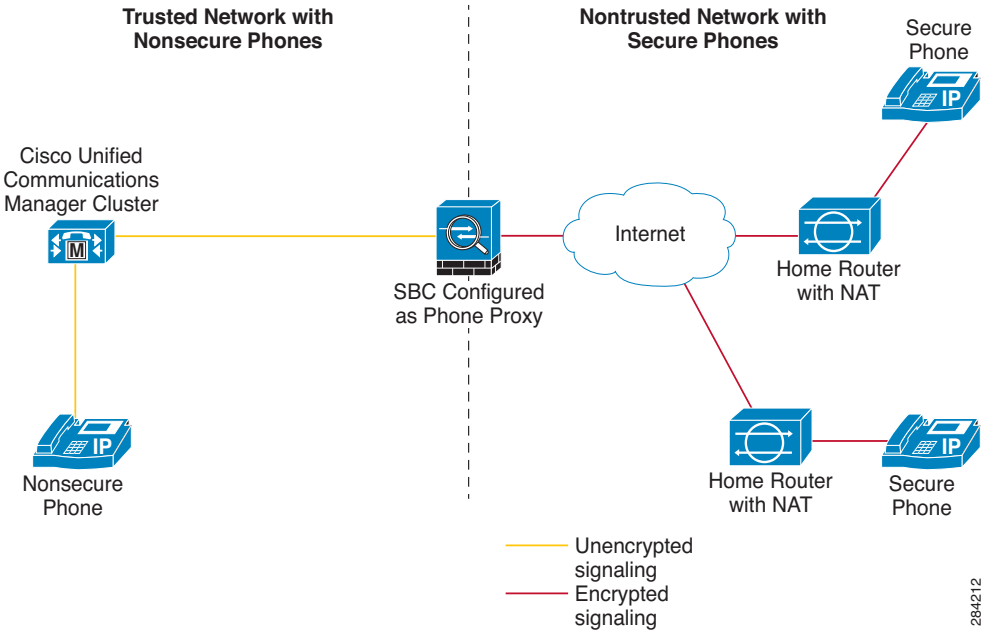
http://www.cisco.com/en/US/products/sw/voicesw/ps556/products_documentation_roadmaps_list.html

Note In this chapter, the organizational network is called the trusted network. All other networks, including the Internet, are called nontrusted networks. Communication secured through encryption is called secure communication, and communication that is not encrypted is called nonsecure communication.

Communication between phones on the trusted network is not encrypted because they are on the trusted network. The VPN used by remote computer users in an organization is an extension of the trusted network. In releases prior to Release 3.5.0, remote users could not use their phones to call phones on the trusted network because VPN support does not cover IP phones. From Release 3.5.0 onward, Cisco Unified Border Element (SP Edition) provides line-side support for Cisco Unified Communications Manager. This support enables the configuration of a phone proxy on the SBC through which phones used by remote users can communicate with phones on the organizational network.

The phone proxy on the SBC bridges IP telephony between the corporate IP telephony network and the Internet. You can secure this bridge by configuring the phone proxy to force the data coming from the phones on nontrusted networks to be encrypted. Alternatively, the phone proxy can be configured to support TCP and RTP coming from the phones on nontrusted networks. Telecommuters can connect their phones to the corporate IP telephony network over the Internet securely via the phone proxy without having to connect over a VPN tunnel. This is shown in Figure 35.

Figure 35 SBC Configured as a Phone Proxy



The Certificate Trust List (CTL) is at the center of the Cisco Unified Communications Manager server authentication process. The CTL file is one of the files downloaded by a phone during the phone registration process. The file contains a list of identities that are attested by a systems administrator using a security token. In this case, the identities are the phones managed by Cisco Unified Communications Manager. The security token contains a certificate that is rooted in a certificate authority (CA). This CA, in turn, is included in the trust anchor list of a phone. To validate the signature of the CTL, a phone validates the systems administrator's security token certificate and then validates the file signature using the public key contained in the certificate. After the phone validates the signature on the CTL file, the phone installs all the identities listed in the CTL file as trusted systems elements.

A phone proxy configured on the SBC can support a Cisco Unified Communications Manager cluster in both mixed mode and nonsecure mode. If you configure the phone proxy for secure communication, remote phones that are capable of encryption are always forced into encrypted mode, regardless of the cluster mode. In addition, Transport Layer Security (TLS), that is, signaling and SRTP (media), is always terminated on the SBC. The SBC can also perform NAT, open pinholes for the media, and apply inspection policies on incoming SIP streams.

A phone proxy that is configured on an adjacency acts as a TFTP proxy and listens at the signaling address of the adjacency. Therefore, the native TFTP service provided by Cisco IOS XE running on the Cisco ASR 1000 Series Router cannot use the same IP address.

If the phone proxy is configured for secure communication, the phone proxy terminates TLS connections coming from the external phones on the SBC and opens TCP connections from the SBC to Cisco Unified Communications Manager. If the phone proxy is configured for nonsecure communication, it terminates TCP connections coming from the external phones on the SBC and opens new TCP connections from the SBC to Cisco Unified Communications Manager.

The phone proxy performs the following additional functions:

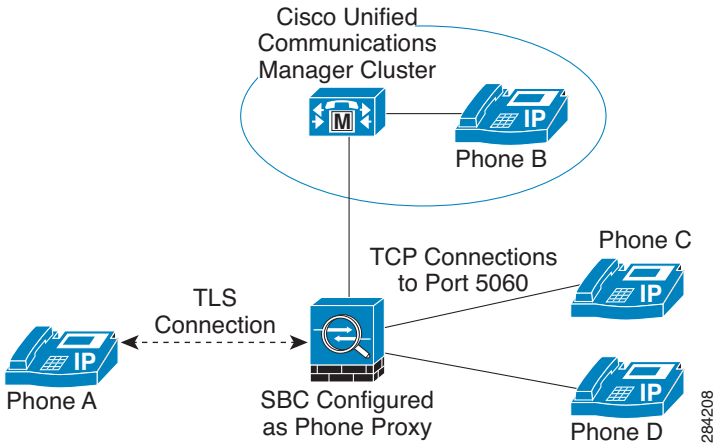
- Creates the CTL file that is used to perform certificate-based authentication with the phones on the nontrusted network.

- Modifies the IP phone configuration file when it is requested via TFTP, changes security fields from nonsecure to secure, and signs all the files sent to the phone. These modifications secure the phones on the nontrusted network by forcing the phones to perform encrypted signaling and media.

Signaling and Media Flows During Phone Calls

The signaling protocols and media flow protocols supported by the phone proxy during a call depend on the type of caller phone and callee phone that take part in the call. Figure 36 shows the types of phones that can communicate with Cisco Unified Communications Manager through a phone proxy.

Figure 36 Sample Scenario Showing Signaling and Media Flows During Phone Calls



In this figure:

- Phone A is on a nontrusted network. Communication between the SBC and Phone A is encrypted by TLS.
- Phone B is in a cluster managed by Cisco Unified Communications Manager, within the trusted network.
- Phone C and Phone D are on a nontrusted network. These two phones may or may not be behind NAT devices. Communication between these two phones and the SBC is not encrypted. Although communication between Phone C and Phone D is not covered in this chapter, note that the phone proxy also enables communication between Phone C and Phone D.

Using the elements shown in Figure 36, the following scenarios describe the signaling and media flows during calls in various scenarios:

- Scenario 1: A Secure Phone on A Nontrusted Network Calls A Nonsecure Phone on A Nontrusted Network, page 661
- Scenario 2: A Secure Phone on a Nontrusted Network Calls a Nonsecure Phone on the Trusted Network, page 661
- Scenario 3: A Nonsecure Phone on a Nontrusted Network Calls a Nonsecure Phone on the Trusted Network, page 662

Scenario 1: A Secure Phone on A Nontrusted Network Calls A Nonsecure Phone on A Nontrusted Network

In Figure 36, Phone A is a secure phone on a nontrusted network and Phone C is a nonsecure phone on a nontrusted network. Table 44 lists the protocols used in signaling and media flows during a call from Phone A to Phone C.

Table 44 Signaling and Media Flows Between Phone A and Phone C

Flow Direction	Signaling Protocol	Media Protocol
From Phone A to the SBC	SIP over TLS	SRTP
From the SBC to Cisco Unified Communications Manager	SIP over TCP	RTP
From Cisco Unified Communications Manager to the SBC	SIP over TCP	RTP
From the SBC to Phone C	SIP	RTP
From Phone C to the SBC	SIP	RTP
From the SBC to Cisco Unified Communications Manager	SIP over TCP	RTP
From Cisco Unified Communications Manager to the SBC	SIP over TCP	RTP
From the SBC to Phone A	SIP over TLS	SRTP

Scenario 2: A Secure Phone on a Nontrusted Network Calls a Nonsecure Phone on the Trusted Network

In Figure 36, Phone A is a secure phone on a nontrusted network and Phone B is a nonsecure phone on a trusted network. Table 45 lists the protocols used in signaling and media flows during a call from Phone A to Phone B.

Table 45 Signaling and Media Flows Between Phone A and Phone B

Flow Direction	Signaling Protocol	Media Protocol
From Phone A to the SBC	SIP over TLS	SRTP
From the SBC to Cisco Unified Communications Manager	SIP over TCP	RTP
From Cisco Unified Communications Manager to Phone B	SIP	RTP
From Phone B to Cisco Unified Communications Manager	SIP	RTP

Table 45 Signaling and Media Flows Between Phone A and Phone B (continued)

Flow Direction	Signaling Protocol	Media Protocol
From Cisco Unified Communications Manager to the SBC	SIP over TCP	RTP
From the SBC to Phone A	SIP over TLS	SRTP

Scenario 3: A Nonsecure Phone on a Nontrusted Network Calls a Nonsecure Phone on the Trusted Network

In Figure 36, Phone C is a nonsecure phone on a nontrusted network and Phone B is a nonsecure phone on a trusted network. Table 46 lists the protocols used in signaling and media flows during a call from Phone C to Phone B.

Table 46 Signaling and Media Flows Between Phone C and Phone B

Flow Direction	Signaling Protocol	Media Protocol
From Phone C to the SBC	SIP	RTP
From the SBC to Cisco Unified Communications Manager	SIP over TCP	RTP
From Cisco Unified Communications Manager to Phone B	SIP	RTP
From Phone B to Cisco Unified Communications Manager	SIP	RTP
From Cisco Unified Communications Manager to the SBC	SIP over TCP	RTP
From the SBC to Phone C	SIP	RTP

Restrictions for the Line-Side Support for Cisco Unified Communications Manager Feature

The following are restrictions pertaining to the Line-Side Support for Cisco Unified Communications Manager feature:

- A phone proxy that is configured on an adjacency acts as a TFTP proxy and listens at the signaling address of the adjacency. Therefore, the native TFTP service provided by Cisco IOS XE running on the Cisco ASR 1000 Series Router cannot use the same IP address.
- When the Cisco Unified Communications Manager TFTP server is configured as a domain name instead of an IP address, only the first IP address in the DNS response is used. In other words, if the first IP address is not reachable for any reason, alternative IP addresses are not tried and the connection attempt fails.

- Only the Configuration High Availability feature is supported. The High Availability feature of the TFTP session is not supported. Therefore, after a switchover takes place, all the TFTP connections are lost and the IP phones must re-establish these connections.
- Only the Intrachassis High Availability feature is supported. Because the Line-Side Support for Cisco Unified Communications Manager feature uses crypto Public Key Infrastructure (PKI), which does not support the Interchassis High Availability feature, the Interchassis High Availability feature is not supported.
- TLS mutual authentication is not supported.
- Only IPv4 is supported. IPv6 is not supported.
- Key Press Markup Language (KPML) on Cisco Unified Communications Manager is not supported. While configuring Cisco Unified Communications Manager, you must choose Dial Plan instead of KPML.
- Services such as overlap dialing, ad hoc conference, call pickup, call park, shared line, reset, and restart are not supported.

Configuring a Phone Proxy

The following sections provide information about configuring a phone proxy:

Configuration Prerequisites

Before configuring the phone proxy, ensure that the SBC meets the following configuration requirements:

- The SBC must have an IP address for media termination that fulfills the following criteria:
  - The IP address is a publicly routable address that is an unused IP address within an address range associated with the outside network interface on the SBC.
  - The IP address cannot have the same address as that of an interface on the SBC. This includes the IP address of the external interface on the SBC to which remote IP phones connect.
  - The IP address cannot overlap with existing static NAT pools or NAT rules.
  - The IP address cannot be the same as the IP address of the Cisco Unified Communications Manager server or the TFTP server.
- For IP phones behind a router or gateway, routes must be added to the media termination address on the router or gateway so that the phones can reach the media termination address.

Note If your organizational security policy requires that IP phones on internal networks must not have routes to external networks, we recommend that you use a NAT device that is compatible with Cisco Unified Communications Manager on the internal network. By representing the media termination address with an address within the internal network address range, you avoid having to expose the internal IP phones to external routes.

- The TFTP server must reside on the same interface as Cisco Unified Communications Manager.
- While configuring Cisco Unified Communications Manager, you must choose Dial Plan instead of KPML. This is because KPML on Cisco Unified Communications Manager is not supported by the phone proxy. For information about the procedure, see *Cisco Unified Communications Manager Administration Guide*.
- If you have a fully qualified domain name (FQDN) instead of an IP address configured for Cisco Unified Communications Manager, you must configure and enable DNS lookup on the SBC. After configuring DNS lookup, ensure that the SBC can ping Cisco Unified Communications Manager with the configured FQDN.
- If you have enabled the Certificate Authority Proxy Function (CAPF) service and Cisco Unified Communications Manager is not running on the Publisher, and if the Publisher is configured with an FQDN instead of an IP address, you must configure DNS lookup.
- You must configure the following access-list rules on the SBC to allow TFTP requests:
  - Address—TFTP Server
  - Port—69
  - Protocol—UDP
  - Description—Allow incoming TFTP
- If the phone proxy is deployed behind an existing firewall, you must configure access-list rules to permit signaling, TFTP, and media traffic to the phone proxy. However, if NAT is required for Cisco Unified Communications Manager, you must configure NAT on the SBC and not on the existing firewall.

  Table 47 lists the port configuration requirements.

Table 47 Port Configuration Requirements

Address	Port	Protocol	Description
Media Termination	1024-65535	UDP	Allow incoming SRTP
TFTP Server	69	UDP	Allow incoming TFTP
Cisco Unified Communications Manager	5061	TCP	Allow incoming secure SIP

Note All these ports are configurable on Cisco Unified Communications Manager, except for TFTP. The default values listed in this table must be modified to match the values set on Cisco Unified Communications Manager. If NAT is configured for the TFTP server or Cisco Unified Communications Manager, the translated global address must be used in the access lists.

Cisco Unified Communications Manager and IP Phone Versions Supported by the Phone Proxy

The Line-Side Support for Cisco Unified Communications Manager feature is supported from Cisco Unified Communications Manager Release 7.0 onward. This is because earlier releases of Cisco Unified Communications Manager do not support multiple phones registered at one IP address.

Cisco Unified Communications Manager supports multiple phones registered at one IP address only when the phones use SIP over TCP. Therefore, UDP-based phones are not supported. To determine whether a phone is UDP based, see the user documentation shipped with the phone.

End-User Phone Provisioning

Registering a phone involves providing the phone with the information required to authenticate and communicate with other phones through the phone proxy and Cisco Unified Communications Manager. Registration is an automated process that takes place when you connect a phone to the trusted network. The following is a summary of the registration process:

1. The phone presents its manufacturer-installed certificate (MIC) for authentication to the CAPF service on the Cisco Unified Call Manager publisher.

   By default, the MIC is present in the phone.

2. After the CAPF authenticates the MIC, it deploys a locally significant certificate (LSC) on the phone.

   The LSC is used to establish a TLS connection between Cisco Unified Communications Manager and the phone during each call.

3. The phone downloads the CTL file.

   The phone sends a TFTP request for the CTL file to the SBC. As part of the configuration procedure, the SBC generates a CTL file. The SBC responds to the TFTP request from the phone by sending this file.

4. The phone downloads the Initial Trust List (ITL) file.

   The phone sends a TFTP request for the ITL file to the SBC. The SBC forwards this request to Cisco Unified Communications Manager, which responds by sending this file to the SBC. The SBC forwards the file to the phone.

5. The phone downloads the SEP file.

   The phone sends a TFTP request for the Selsius Ethernet Phone (SEP) file (that is, the SEP<mac>.cnf.xml file) to the SBC. The SBC forwards this request to Cisco Unified Communications Manager, which responds by sending the file to the SBC. The SBC updates the file to reflect the access permissions defined for the services supported by Cisco Unified Communications Manager. Note that you set these access permissions while configuring the phone proxy. The SBC then signs the file and forwards it to the phone.

6. Some other files, such as locale-specific files, are also downloaded by the SBC from Cisco Unified Communications Manager. The SBC signs these files and then forwards them to the phone.

Summary of the Procedure to Configure a Phone Proxy

The following are the high-level steps to configure a phone proxy:

1. Create the PKI trustpoints.
2. Create the CTL file.
3. Create the phone proxy.
4. Attach the phone proxy to an adjacency.

Creating the PKI Trustpoints

For the TLS proxy used by the phone proxy to complete a TLS handshake, the TLS proxy must verify the certificates from the IP phone. The TLS proxy must also verify the certificates from Cisco Unified Communications Manager if a TLS handshake is being performed with Cisco Unified Communications Manager. The TLS proxy requires a CA Manufacturer certificate to validate the IP phone certificate. To import the CA Manufacturer certificate from Cisco Unified Communications Manager to the SBC:

Step 1 To create the keypair and selfsigned trustpoint on the SBC, run the following commands:

```
Router(config)# crypto key generate rsa label pp_rsa modulus 1024 general-keys
Router(config)# crypto pki trustpoint self_trustpoint
Router(config-ca-trustpoint)# enrollment selfsigned
Router(config-ca-trustpoint)# subject-name CN=SBC-Phone-Proxy,OU=my_BU,O=my_company
Router(config-ca-trustpoint)# rsakeypair pp_rsa
Router(config-ca-trustpoint)# crypto pki enroll self_trustpoint
```

Step 2 Open the Cisco Unified Communications Manager Operating System Administration web page.

Step 3 Choose **Security > Certificate Management**.

Note Some releases of Cisco Unified Communications Manager may have a different UI and may require you to perform different steps to locate the certificates.

Step 4 Click **Find**. The list of certificates is displayed.

Step 5 Double-click the **Cisco_Manufacturing_CA.pem** file. The certificate information is displayed in a dialog box that also enables you to download the certificate.

Note If the certificate list contains more than one certificate with the file name Cisco_Manufacturing_CA, ensure that you select Cisco_Manufacturing_CA.pem, that is, the file with the .pem file extension.

Step 6 Click **Download**, and save the file as a text file.

Step 7 On the SBC, create a trustpoint for the Cisco Manufacturing CA and enroll via the terminal by entering the following commands. You must enroll via the terminal because you will be copying the certificate that you download in Step 6.

```
Router(config)# crypto ca trustpoint trustpoint_name
Router(config-ca-trustpoint)# enrollment terminal
```

Step 8 Authenticate the trustpoint by running the following command:

```
Router(config)# crypto ca authenticate trustpoint
```

Step 9 At the prompt to enter the Base-64 encoded CA certificate, copy the contents of the file that you download in Step 6 and, paste them at the prompt. Because the file is already in Base-64 encoding, no conversion is required. At the prompt to accept the file, enter **yes**.

Note When you copy the certificate, ensure that you also copy the BEGIN and END lines.

Step 10 Repeat Steps 2 through 9 for the next certificate. Table 48 shows the certificates that are required by the SBC.

Table 48 Certificates Required by the SBC for the Phone Proxy

Certificate Name	Purpose
Cisco_Manufacturing_CA	Authenticating IP phones with a MIC.
CAP-RTP-001	Authenticating IP phones with a MIC.
CAP-RTP-002	Authenticating IP phones with a MIC.
CAPF	Authenticating IP phones with an LSC.

Creating the CTL File

This task shows how to create the CTL file.

SUMMARY STEPS

1. **configure terminal**
2. **sbc ctl-file** *ctl-file-name*
3. **description** *description*
4. **record-entry [capf | selfsigned] trustpoint** *trustpoint-name*
5. **complete**
6. **end**
7. **show sbc ctl-file** [*ctl-file-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc ctl-file** *ctl-file-name* **Example:** `Router(config)# sbc ctl-file ctl-1`	Creates the CTL file. • *ctl-file-name*—Name of the CTL file.

	Command or Action	Purpose
Step 3	**description** *description* **Example:** `Router(config-ctl-file)# description ctl_101`	Sets a description for the CTL file. • *description*—Description of the CTL file.
Step 4	**record-entry [capf | selfsigned] trustpoint** *trustpoint-name* **Example:** `Router(config-ctl-file)# record-entry capf trustpoint trustpoint_1`	Specifies the trustpoints to be used for the creation of the CTL file. • **capf**—Specifies that the trustpoint is created using the CAPF certificate imported from Cisco Unified Communications Manager to the router. • **selfsigned**—Specifies that the trustpoint is self-signed by the router. • **trustpoint** *trustpoint-name*—Specifies the name of the trustpoint.
Step 5	**complete** **Example:** `Router(config-ctl-file)# complete`	Completes the creation of the CTL file.
Step 6	**end** **Example:** `Router(config-ctl-file)# end`	Exits the CTL file configuration mode, and enters the privileged EXEC mode.
Step 7	**show sbc ctl-file [***ctl-file-name***]** **Example:** `Router# show sbc ctl-file ctl-1`	Displays details of all the CTL files or the specified CTL file.

Configuring a Phone Proxy on the SBC

This task shows how to configure a phone proxy on the SBC.

SUMMARY STEPS

1. **configure terminal**
2. **sbc phone-proxy** *phone-proxy-name*
3. **description** *description*
4. **tftp-server address [ipv4** *server-ip-address* **|** *domain-name***] local-address ipv4** *local-ip-address* **vrf** *vrf-name*
5. **ctl-file** *ctl-file-name*
6. **access-secure**
7. **disable service-settings**
8. **capf-address ipv4** *ip-address*
9. **session-timeout** *timeout-interval*

10. **max-concurrent-sessions** *number-of-sessions*
11. **complete**
12. **end**
13. **show sbc phone-proxy** [*phone-proxy-name* **[sessions] | sessions]**
14. **show sbc** *sbc-name* **sbe adjacencies pp-sip detail**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc phone-proxy** *phone-proxy-name* **Example:** `Router(config)# sbc phone-proxy phone-proxy-1`	Configures the phone proxy. • *phone-proxy-name*—Name of the phone proxy.
Step 3	**description** *description* **Example:** `Router(config-phone-proxy)# description cluster-test`	Sets a description for the phone proxy. • *description*—Description for the phone proxy.
Step 4	**tftp-server address [ipv4** *server-ip-address*| **domain-name] local-address ipv4** *local-ip-address* **vrf** *vrf-name* **Example:** `Router(config-phone-proxy)# tftp-server address ipv4 198.51.100.101 local-address ipv4 192.168.0.109 vrf vrf1`	Specifies the address of the TFTP server.
Step 5	**ctl-file** *ctl-file-name* **Example:** `Router(config-phone-proxy)# ctl-file myctl`	Specifies the name of the CTL file. • *ctl-file-name*—Name of the CTL file.
Step 6	**access-secure** **Example:** `Router(config-phone-proxy)# access-secure`	Specifies that the secure (encrypted) mode is to be used for accessing the SBC. The default is that the nonsecure mode is to be used for communication with the SBC.
Step 7	**disable service-settings** **Example:** `Router(config-phone-proxy)# disable-service-settings`	Disables the service settings configured on Cisco Unified Communications Manager. PC Port, Gratuitous ARP, Voice VLAN access, Web access, and Span to PC Port are examples of the services enabled by default on Cisco Unified Communications Manager.

	Command or Action	Purpose
Step 8	**capf-address ipv4** *ip-address* **Example:** `Router(config-phone-proxy)# capf-address ipv4 198.51.100.102`	Sets a dummy IP address as the local address for the CAPF service. This address is used for LSC updates. • *ip-address*—Dummy IP address for the CAPF service. **Note** The dummy IP address that you specify must not be used by other services.
Step 9	**session-timeout** *timeout-interval* **Example:** `Router(config-phone-proxy)# session-timer 200`	Specifies the maximum amount of time, in seconds, for which a TFTP session can remain open. The range is from 60 to 6000. The default is 180. • *timeout-interval*—Maximum length of a session.
Step 10	**max-concurrent-sessions** *number-of-sessions* **Example:** `Router(config-phone-proxy)# max-concurrent-sessions 4`	Specifies the maximum number of concurrent sessions. • *number-of-sessions*—Maximum number of concurrent sessions. The default is 30 sessions.
Step 11	**complete** **Example:** `Router(config-phone-proxy)# complete`	Completes the configuration of the phone proxy.
Step 12	**end** **Example:** `Router(config-phone-proxy)# end`	Exits the phone proxy configuration mode, and enters the privileged EXEC mode.
Step 13	**show sbc phone-proxy [***phone-proxy-name* **[sessions] | sessions]** **Example:** `Router# show sbc phone-proxy phone-proxy-1`	Displays the details of the sessions being conducted through the specified phone proxy or all the phone proxies. • *phone-proxy-name*—Name of the phone proxy.
Step 14	**show sbc** *sbc-name* **sbe adjacencies pp-sip detail** **Example:** `Router# show sbc mysbc sbe adjacencies pp-sip detail`	Shows the configuration details of the specified adjacency.

Configuring the TFTP Port Range for a Phone Proxy

This task shows how to configure the TFTP port range for a phone proxy.

SUMMARY STEPS

1. **configure terminal**
2. **sbc phone-proxy tftp-address ipv4** *ip-address* **[vrf** *vrf-name***]**
3. **port-range** *min-port max-port*
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc phone-proxy tftp-address ipv4** *ip-address* **[vrf** *vrf-name***]** **Example:** `Router(config)# sbc phone-proxy tftp-address ipv4 192.168.0.109 vrf vrf1`	Specifies the IP address and VRF name of the TFTP server. • *ip-address*—IP address of the TFTP server. • *vrf-name*—Name of the TFTP server's VRF.
Step 3	**port-range** *min-port max-port* **Example:** `Router(config-pp-pr)# port-range 30000 40000`	Specifies a port range for the TFTP server. • *min-port*—First port number of the port range. • *max-port*—Last port number of the port range.
Step 4	**end** **Example:** `Router(config-pp-pr)# end`	Exits the phone proxy configuration mode, and enters the privileged EXEC mode.

Associating a Phone Proxy with an Adjacency

This task shows how to associate a phone proxy with an adjacency.

SUMMARY STEPS

1. **configure terminal**
2. **sbc** *sbc-name*
3. **sbe**
4. **adjacency sip** *adjacency-name*
5. **phone-proxy** *phone-proxy-name*
6. **attach**
7. **end**
8. **show sbc** *sbc-name* **sbe adjacencies** *adjacency-name* **detail**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enables the global configuration mode.
Step 2	**sbc** *sbc-name* **Example:** `Router(config)# sbc mysbc`	Enters the SBC service mode. • *sbc-name*—Name of the SBC.
Step 3	**sbe** **Example:** `Router(config)# sbe`	Enters the SBE entity mode within the SBC service.
Step 4	**adjacency sip** *adjacency-name* **Example:** `Router(config-sbc-sbe)# adjacency sip pp-adj`	Enters the mode of an SBE SIP adjacency. • *adjacency-name*—Name of the adjacency.
Step 5	**phone-proxy** *phone-proxy-name* **Example:** `Router(config-sbc-sbe-adj-sip)# phone-proxy phone-proxy-1`	Specifies the name of the phone proxy that you want to associate with the adjacency. • *phone-proxy-name*—Name of the phone proxy.
Step 6	**attach** **Example:** `Router(config-sbc-sbe-adj-sip)# attach`	Associates the phone proxy with the adjacency.
Step 7	**end** **Example:** `Router(config-sbc-sbe-adj-sip)# end`	Exits the SBE SIP adjacency mode, and enters the privileged EXEC mode.
Step 8	**show sbc** *sbc-name* **sbe adjacencies** *adjacency-name* **detail** **Example:** `Router# show sbc mysbc sbe adjacencies pp-adj detail`	Shows the configuration details of the specified adjacency.

Viewing Information Pertaining to the Phone Proxy

You can use the following commands to view information about and troubleshoot issues pertaining to the phone proxy:

- Use the following command to enable debug logging of data pertaining to phone proxy connections:

  debug sbc phone-proxy [all | cli| | detail | error | event]

- Use the following command to enable debug logging of data pertaining to the High Availability feature of the phone proxy:

  debug sbc *sbc-name* **high-availability phone-proxy**

Configuration Examples

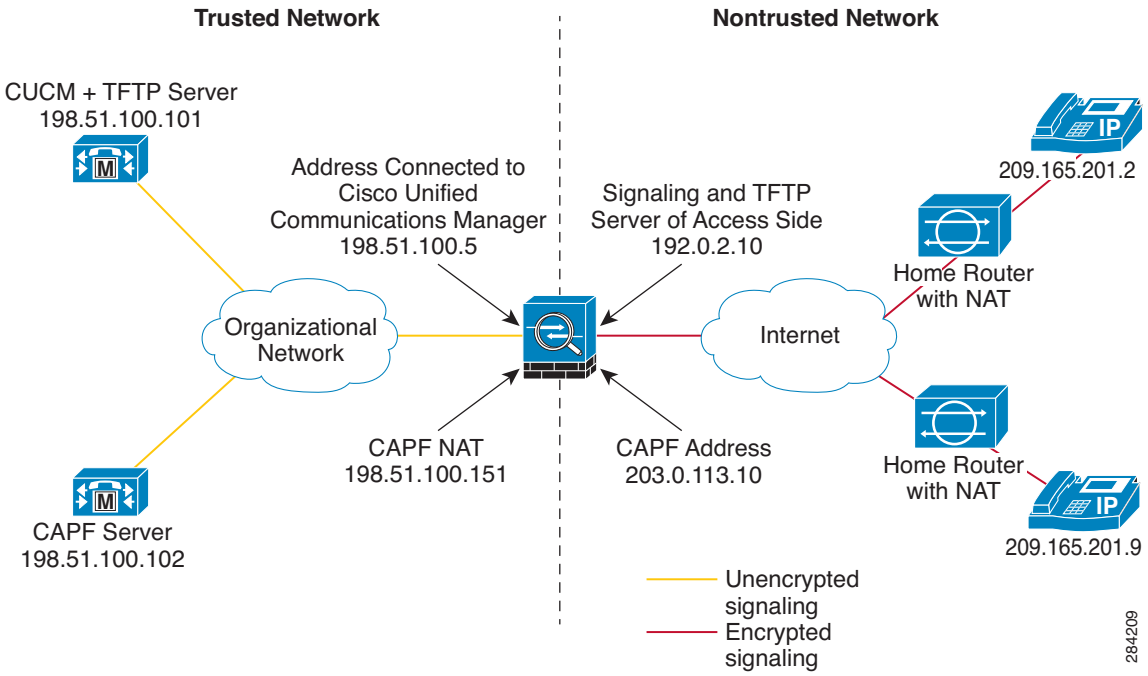
The examples described in the following sections show how to configure a phone proxy in different scenarios:

- Configuring a Phone Proxy for Secure Access with LSC Installed for a Single Cluster, page 673
- Configuring a Phone Proxy for Secure Access Without LSC Installed for a Single Cluster, page 677
- Configuring a Phone Proxy for Nonsecure Access for a Single Cluster, page 680
- Removing a Phone Proxy, page 683

Configuring a Phone Proxy for Secure Access with LSC Installed for a Single Cluster

Figure 37 shows an operating environment in which secure phones on the nontrusted network access phones on the trusted network. The LSC is installed and the CAPF server is configured in this sample scenario.

Figure 37 Secure Phones on a Nontrusted Network Access Phones on the Trusted Network with LSC Installed



Perform the following steps to configure a phone proxy and the other entities in this operating environment. Note that some commands have been omitted from these steps for the sake of brevity.

Step 1 If there is an existing self-signed trustpoint and certificate, skip this step and directly proceed to the next step. Otherwise, create an RSA key-pair and a self-signed trustpoint and generate the certificate by running the following commands:

```
Router(config)# crypto key generate rsa label pp_rsa modulus 1024 general-keys
Router(config)# crypto pki trustpoint self_trustpoint
Router(config-ca-trustpoint)# enrollment selfsigned
Router(config-ca-trustpoint)# subject-name CN=SBC-Phone-Proxy,OU=my_BU,O=my_company
Router(config-ca-trustpoint)# rsakeypair pp_rsa
Router(config-ca-trustpoint)# crypto pki enroll self_trustpoint
```

Step 2 Import the CAPF certificate from Cisco Unified Communications Manager and create the trustpoint by running the following commands:

```
Router(config)# crypto pki trustpoint capf_trustpoint
Router(config-ca-trustpoint)# enrollment terminal
Router(config-ca-trustpoint)# crypto pki authenticate capf_trustpoint
```

Step 3 Create the CTL file by running the following commands:

```
Router(config)# sbc ctl-file myctl
Router(config-ctl-file)# record-entry capf trustpoint capf_trustpoint
Router(config-ctl-file)# record-entry selfsigned trustpoint self_trustpoint
Router(config-ctl-file)# complete
```

Step 4 Create the phone proxy by running the following commands:

```
Router(config)# sbc phone-proxy mypp_1
Router(config-phone-proxy)# tftp-server address ipv4 198.51.100.101 local-address ipv4
192.0.2.10 vrf vrf1
Router(config-phone-proxy)# ctl-file myctl
```

```
Router(config-phone-proxy)# access-secure
Router(config-phone-proxy)# capf-address ipv4 203.0.113.10
Router(config-phone-proxy)# complete
```

Note The capf-address value can be any unused IP address.

Step 5 Define the TFTP port range by running the following commands:

```
Router(config)# sbc phone-proxy tftp-address ipv4 192.0.2.10 vrf vrf1
Router(config-pp-pr)# port-range 8192 30000
Router(config)# sbc phone-proxy tftp-address ipv4 192.0.2.10
```

Note Create a TFTP address entry for each TFTP client address that is in use and the local TFTP server address (which is the same as the signaling address of an access side adjacency). It is optional to specify the port range of each TFTP address. If the port range is not specified, then 32768 to 65535 is used as the default port range.

Step 6 Configure the SIP adjacency by running the following commands:

```
Router(config)# sbc sbc1
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip option-editor oe-all
Router(config-sbc-sbe)# blacklist
Router(config-sbc-sbe)# sip option-profile op-all
Router(config-sbc-sbe)# blacklist
Router(config-sbc-sbe)# sip header-profile h1
Router(config-sbc-sbe)# blacklist
Router(config-sbc-sbe)# sip method-profile mp1
Router(config-sbc-sbe-sip-mth)# blacklist
Router(config-sbc-sbe-sip-mth)# pass-body
Router(config-sbc-sbe)# sip header-editor acc-in
Router(config-sbc-sbe-sip-hdr)# blacklist
Router(config-sbc-sbe-sip-hdr)# store-rule entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-name from header-value
regex-match "^.*sips*:\([^@][^@]*\)@.*$" store-as fu
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and is-request eq true
Router(config-sbc-sbe-sip-hdr)# store-rule entry 2
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-name Contact header-value
regex-match "^\(.*sips*:\)\([^@][^@]*\)\(@[^;>][^;>]*\)\(.*\)$" store-as bcu cu ch ach
Router(config-sbc-sbe-sip-hdr)# store-rule entry 3
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-name to header-value regex-match
"^.*sips*:\([^@][^@]*\)@.*$" store-as tu
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and is-request eq false
Router(config-sbc-sbe-mep-hdr-ele-act)# store-rule entry 4
Router(config-sbc-sbe-mep-hdr-ele-act)# condition header-name Event header-value
regex-match "\(.*\)call-id=\(^[\";]*\)\(.*\)" store-as b_callid callid a_callid
Router(config-sbc-sbe-mep-hdr-ele-act)# condition and is-request eq true
Router(config-sbc-sbe-mep-hdr-ele-act)# header event entry 1
Router(config-sbc-sbe-mep-hdr-ele)# action replace-value value
"${b_callid}call-id=${callid}${a_callid}"
Router(config-sbc-sbe-mep-hdr-ele-act)# condition is-request eq true
Router(config-sbc-sbe-mep-hdr-ele-act)# condition and variable callid is-defined eq true
Router(config-sbc-sbe-sip-hdr)# header contact entry 1
Router(config-sbc-sbe-sip-hdr-ele)# action replace-value value
"${bcu}${fu}${ch};origUser=${cu}${ach}"
Router(config-sbc-sbe-sip-hdr-ele)# condition variable cu is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele)# condition and is-request eq true
Router(config-sbc-sbe-sip-hdr)# header contact entry 2
```

```
Router(config-sbc-sbe-sip-hdr-ele)# action replace-value value
"${bcu}${tu}${ch};origUser=${cu}${ach}"
Router(config-sbc-sbe-sip-hdr-ele)# condition variable cu is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele)# condition and is-request eq false
Router(config-sbc-sbe)# sip header-editor acc-out
Router(config-sbc-sbe-sip-hdr)# blacklist
Router(config-sbc-sbe-sip-hdr)# store-rule entry 1
Router(config-sbc-sbe-sip-hdr-ele)# condition header-name contact header-value regex-match
"^\([[:space:]<]*sips*\):\([^@]*\)@\(.*\);origUser=\([^;>]*\)\(.*\)" store-as cbu cu ch
cou cr
Router(config-sbc-sbe-sip-hdr-ele)# condition and is-request eq false
Router(config-sbc-sbe-sip-hdr)# store-rule entry 2
Router(config-sbc-sbe-sip-hdr)# header contact entry 1
Router(config-sbc-sbe-sip-hdr-ele)# action replace-value value "${cbu}:${cou}@${ch}${cr}"
Router(config-sbc-sbe-sip-hdr-ele)# condition is-request eq false
Router(config-sbc-sbe-sip-hdr-ele)# condition and variable cbu is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele)# condition and variable cou is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele)# condition and variable ch is-defined eq true
Router(config-sbc-sbe)# adjacency sip access-1
Router(config-sbc-sbe-adj-sip)# editor-type editor
Router(config-sbc-sbe-adj-sip)# header-editor inbound acc-in
Router(config-sbc-sbe-adj-sip)# header-editor outbound acc-out
Router(config-sbc-sbe-adj-sip)# option-editor ua inbound oe-all
Router(config-sbc-sbe-adj-sip)# option-editor ua outbound oe-all
Router(config-sbc-sbe-adj-sip)# fast-register disable
Router(config-sbe-adj-sip)# inherit profile preset-access
Router(config-sbc-sbe-adj-sip)# security untrusted-encrypted
Router(config-sbc-sbe-adj-sip)# signaling-address ipv4 192.0.2.10
Router(config-sbc-sbe-adj-sip)# remote-address ipv4 209.165.201.2 255.255.255.255
Router(config-sbc-sbe-adj-sip)# signaling-peer 209.165.201.2
Router(config-sbc-sbe-adj-sip)# phone-proxy mypp_1
Router(config-sbc-sbe-adj-sip)# registration rewrite-register
Router(config-sbc-sbe-adj-sip)# attach
Router(config-sbc-sbe)# adjacency sip core-1
Router(config-sbc-sbe-adj-sip)# vrf vrf1
Router(config-sbc-sbe-adj-sip)# force-signaling-peer
Router(config-sbc-sbe-adj-sip)# header-profile inbound h1
Router(config-sbc-sbe-adj-sip)# header-profile outbound h1
Router(config-sbc-sbe-adj-sip)# method-profile inbound mp1
Router(config-sbc-sbe-adj-sip)# method-profile outbound mp1
Router(config-sbc-sbe-adj-sip)# option-profile ua inbound op-all
Router(config-sbc-sbe-adj-sip)# option-profile ua outbound op-all
Router(config-sbc-sbe-adj-sip)# inherit profile preset-core
Router(config-sbc-sbe-adj-sip)# signaling-address ipv4 198.51.100.5
Router(config-sbc-sbe-adj-sip)# remote-address ipv4 198.51.100.101 255.255.255.255
Router(config-sbc-sbe-adj-sip)# signaling-peer 198.51.100.101
Router(config-sbc-sbe-adj-sip)# registration target address 198.51.100.101
Router(config-sbc-sbe-adj-sip)# registration contact username passthrough
Router(config-sbc-sbe-adj-sip)# attach
```

Note The registration target address is the IP address of Cisco Unified Communications Manager.

Step 7 Configure NAT for the CAPF by running the following commands:

Note The address range that you specify must be the remote address range of the adjacency.

```
Router(config)# access-list 1 permit 209.165.201.2 0.0.0.255
Router(config)# ip nat pool CAPF_NAT 198.51.100.151 198.51.100.151 netmask 255.255.255.0
Router(config)# ip nat inside source list 1 pool CAPF_NAT overload
```

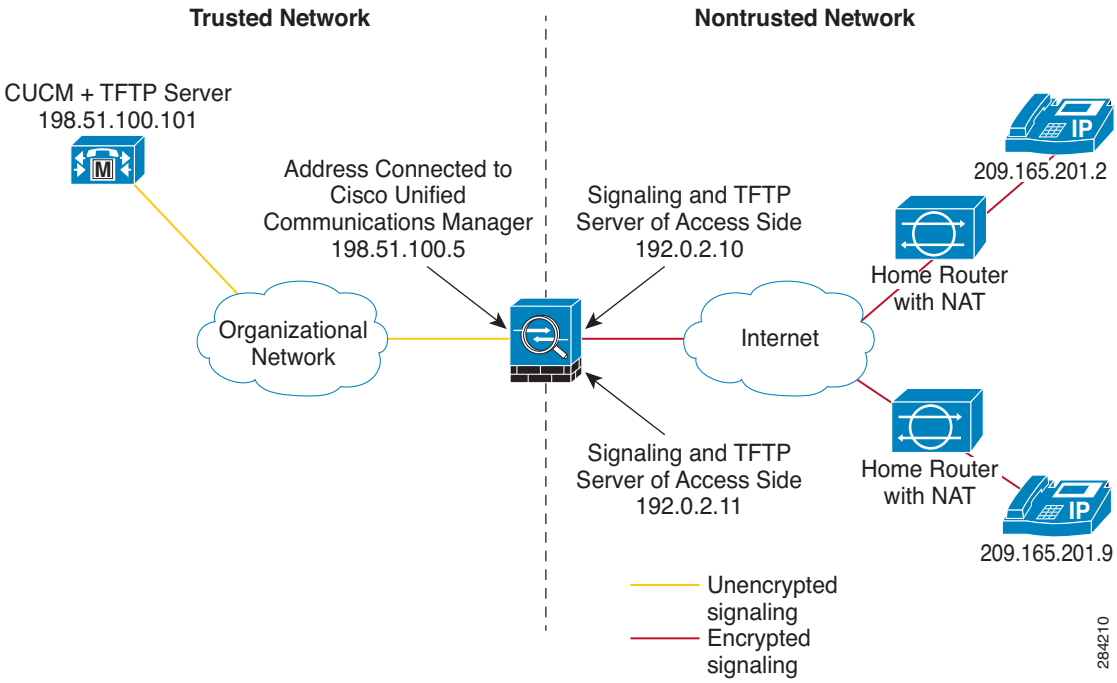
```
Router(config)# ip nat outside source static 198.51.100.102 203.0.113.10 add-route
```

Note 203.0.113.10 is the dummy CAPF IP address configured in the phone proxy. The dummy IP address must not be used for any other service.

Configuring a Phone Proxy for Secure Access Without LSC Installed for a Single Cluster

Figure 38 shows an operating environment in which secure phones on a nontrusted network access phones on the trusted network. The LSC is not installed in this sample scenario.

Figure 38 Secure Phones on a Nontrusted Network Access Phones on the Trusted Network Without LSC



Perform the following steps to configure the phone proxy and the other entities in this operating environment. Note that some commands have been omitted from these steps for the sake of brevity.

Step 1 If there is an existing self-signed trustpoint and certificate, skip this step and directly proceed to the next step. Otherwise, create an RSA key-pair and a self-signed trustpoint and generate the certificate by running the following commands:

```
Router(config)# crypto key generate rsa label pp_rsa modulus 1024 general-keys
Router(config)# crypto pki trustpoint self_trustpoint
Router(config-ca-trustpoint)# enrollment selfsigned
Router(config-ca-trustpoint)# subject-name CN=SBC-Phone-Proxy,OU=my_BU,O=my_company
Router(config-ca-trustpoint)# rsakeypair pp_rsa
Router(config-ca-trustpoint)# crypto pki enroll self_trustpoint
```

Step 2 Import the CAPF certificate from Cisco Unified Communications Manager and create the trustpoint by running the following commands:

```
Router(config)# crypto pki trustpoint CAP-RTP-001_trustpoint
Router(config-ca-trustpoint)# enrollment terminal
Router(config)# crypto pki authenticate CAP-RTP-001_trustpoint
Router(config)# crypto pki trustpoint CAP-RTP-002_trustpoint
Router(config-ca-trustpoint)# enrollment terminal
Router(config)# crypto pki authenticate CAP-RTP-002_trustpoint
Router(config)# crypto pki trustpoint Cisco_Manufacturing_CA_trustpoint
Router(config-ca-trustpoint)# enrollment terminal
Router(config)# crypto pki authenticate Cisco_Manufacturing_CA_trustpoint
```

Step 3 Create the CTL file by running the following commands:

```
Router(config)# sbc ctl-file myctl
Router(config-ctl-file)# record-entry selfsigned trustpoint self_trustpoint
Router(config-ctl-file)# complete
```

Step 4 Create the phone proxy by running the following commands:

```
Router(config)# sbc phone-proxy mypp_1
Router(config-phone-proxy)# tftp-server address ipv4 198.51.100.101 local-address ipv4
192.0.2.10 vrf vrf1
Router(config-phone-proxy)# ctl-file myctl
Router(config-phone-proxy)# access-secure
Router(config-phone-proxy)# disable-service-settings
Router(config-phone-proxy)# complete
```

Step 5 Define the TFTP port range by running the following commands:

```
Router(config)# sbc phone-proxy tftp-address ipv4 192.0.2.10 vrf vrf1
Router(config-pp-pr)# port-range 30000-60000
Router(config)# sbc phone-proxy tftp-address ipv4 192.0.2.10
Router(config)# sbc phone-proxy tftp-address ipv4 10.10.2.6
Router(config-pp-pr)# port-range 35000 55000
```

Step 6 Configure the SIP adjacency by running the following commands:

```
Router(config)# sbc sbc1
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip option-editor oe-all
Router(config-sbc-sbe)# blacklist
Router(config-sbc-sbe)# sip option-profile op-all
Router(config-sbc-sbe)# blacklist
Router(config-sbc-sbe)# sip header-profile h1
Router(config-sbc-sbe)# blacklist
Router(config-sbc-sbe)# sip method-profile mp1
Router(config-sbc-sbe-sip-mth)# blacklist
Router(config-sbc-sbe-sip-mth)# pass-body
Router(config-sbc-sbe)# sip header-editor acc-in
Router(config-sbc-sbe-sip-hdr)# blacklist
Router(config-sbc-sbe-sip-hdr)# store-rule entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-name from header-value
regex-match "^.*sips*:\([^@][^@]*\)@.*$" store-as fu
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and is-request eq true
Router(config-sbc-sbe-sip-hdr)# store-rule entry 2
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-name Contact header-value
regex-match "^\(.*sips*:\)\([^@][^@]*\)\(@[^;>][^;>]*\)\(.*\)$" store-as bcu cu ch ach
Router(config-sbc-sbe-sip-hdr)# store-rule entry 3
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-name to header-value regex-match
"^.*sips*:\([^@][^@]*\)@.*$" store-as tu
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and is-request eq false
Router(config-sbc-sbe-mep-hdr-ele-act)# store-rule entry 4
```

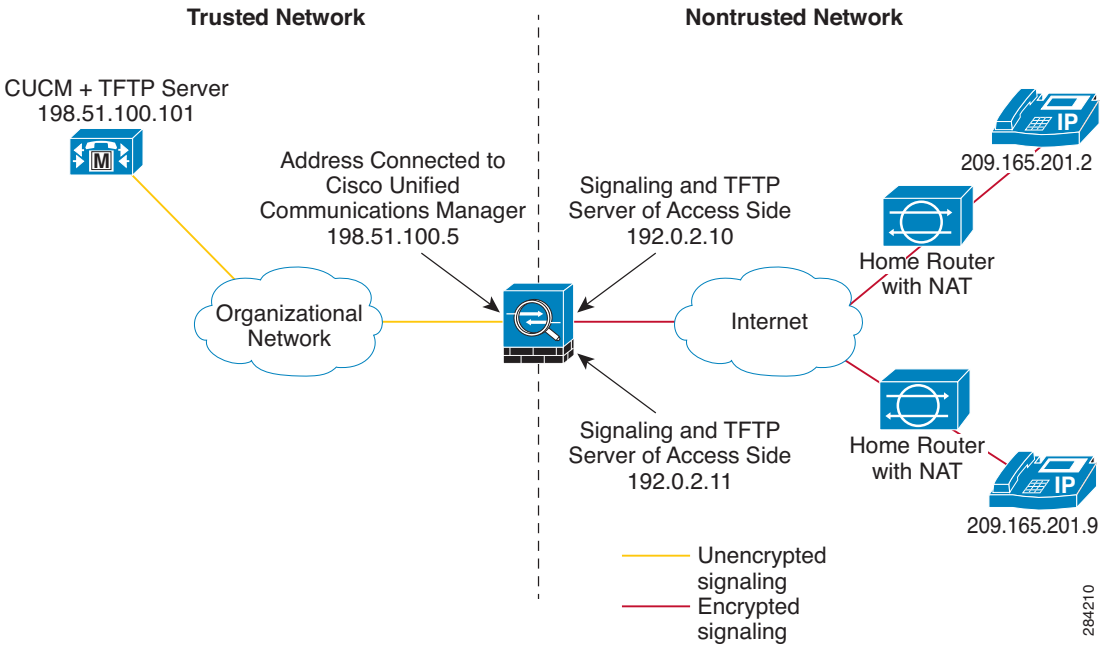
```
Router(config-sbc-sbe-mep-hdr-ele-act)# condition header-name Event header-value
regex-match "\(.*\)call-id=\(^[\";]*\)\(.*\)" store-as b_callid callid a_callid
Router(config-sbc-sbe-mep-hdr-ele-act)# condition and is-request eq true
Router(config-sbc-sbe-mep-hdr-ele-act)# header event entry 1
Router(config-sbc-sbe-mep-hdr-ele)# action replace-value value
"${b_callid}call-id=${callid}${a_callid}"
Router(config-sbc-sbe-mep-hdr-ele-act)# condition is-request eq true
Router(config-sbc-sbe-mep-hdr-ele-act)# condition and variable callid is-defined eq true
Router(config-sbc-sbe-sip-hdr)# header contact entry 1
Router(config-sbc-sbe-sip-hdr-ele)# action replace-value value
"${bcu}${fu}${ch};origUser=${cu}${ach}"
Router(config-sbc-sbe-sip-hdr-ele)# condition variable cu is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele)# condition and is-request eq true
Router(config-sbc-sbe-sip-hdr)# header contact entry 2
Router(config-sbc-sbe-sip-hdr-ele)# action replace-value value
"${bcu}${tu}${ch};origUser=${cu}${ach}"
Router(config-sbc-sbe-sip-hdr-ele)# condition variable cu is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele)# condition and is-request eq false
Router(config-sbc-sbe)# sip header-editor acc-out
Router(config-sbc-sbe-sip-hdr)# blacklist
Router(config-sbc-sbe-sip-hdr)# store-rule entry 1
Router(config-sbc-sbe-sip-hdr-ele)# condition header-name contact header-value regex-match
"^\([[:space:]<]*sips*\):\([^@]*\)@\(.*\);origUser=\([^;>]*\)\(.*\)" store-as cbu cu ch
cou cr
Router(config-sbc-sbe-sip-hdr-ele)# condition and is-request eq false
Router(config-sbc-sbe-sip-hdr)# store-rule entry 2
Router(config-sbc-sbe-sip-hdr)# header contact entry 1
Router(config-sbc-sbe-sip-hdr-ele)# action replace-value value "${cbu}:${cou}@${ch}${cr}"
Router(config-sbc-sbe-sip-hdr-ele)# condition is-request eq false
Router(config-sbc-sbe-sip-hdr-ele)# condition and variable cbu is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele)# condition and variable cou is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele)# condition and variable ch is-defined eq true
Router(config-sbc-sbe)# adjacency sip access-1
Router(config-sbc-sbe-adj-sip)# editor-type editor
Router(config-sbc-sbe-adj-sip)# header-editor inbound acc-in
Router(config-sbc-sbe-adj-sip)# header-editor outbound acc-out
Router(config-sbc-sbe-adj-sip)# option-editor ua inbound oe-all
Router(config-sbc-sbe-adj-sip)# option-editor ua outbound oe-all
Router(config-sbe-adj-sip)# inherit profile preset-access
Router(config-sbc-sbe-adj-sip)# security untrusted-encrypted
Router(config-sbc-sbe-adj-sip)# signaling-address ipv4 192.0.2.10
Router(config-sbc-sbe-adj-sip)# remote-address ipv4 209.165.201.2 255.255.255.255
Router(config-sbc-sbe-adj-sip)# signaling-peer 209.165.201.2
Router(config-sbc-sbe-adj-sip)# phone-proxy mypp_1
Router(config-sbc-sbe-adj-sip)# registration rewrite-register
Router(config-sbc-sbe-adj-sip)# attach
Router(config-sbc-sbe)# adjacency sip access-2
Router(config-sbc-sbe-adj-sip)# editor-type editor
Router(config-sbc-sbe-adj-sip)# header-editor inbound acc-in
Router(config-sbc-sbe-adj-sip)# header-editor outbound acc-out
Router(config-sbc-sbe-adj-sip)# option-editor ua inbound oe-all
Router(config-sbc-sbe-adj-sip)# option-editor ua outbound oe-all
Router(config-sbc-sbe-adj-sip)# security untrusted-encrypted
Router(config-sbc-sbe-adj-sip)# signaling-address ipv4 10.10.2.6
Router(config-sbc-sbe-adj-sip)# remote-address ipv4 69.118.130.9 255.255.255.255
Router(config-sbc-sbe-adj-sip)# signaling-peer 69.118.122.6
Router(config-sbc-sbe-adj-sip)# phone-proxy mypp_1
Router(config-sbc-sbe-adj-sip)# registration rewrite-register
Router(config-sbc-sbe-adj-sip)# attach
Router(config-sbc-sbe)# adjacency sip core-1
Router(config-sbc-sbe-adj-sip)# vrf vrf1
Router(config-sbc-sbe-adj-sip)# force-signaling-peer
Router(config-sbc-sbe-adj-sip)# header-profile inbound h1
Router(config-sbc-sbe-adj-sip)# header-profile outbound h1
```

```
Router(config-sbc-sbe-adj-sip)# method-profile inbound mp1
Router(config-sbc-sbe-adj-sip)# method-profile outbound mp1
Router(config-sbc-sbe-adj-sip)# option-profile ua inbound op-all
Router(config-sbc-sbe-adj-sip)# option-profile ua outbound op-all
Router(config-sbc-sbe-adj-sip)# inherit profile preset-core
Router(config-sbc-sbe-adj-sip)# signaling-address ipv4 198.51.100.5
Router(config-sbc-sbe-adj-sip)# remote-address ipv4 198.51.100.101 255.255.255.255
Router(config-sbc-sbe-adj-sip)# signaling-peer 198.51.100.101
Router(config-sbc-sbe-adj-sip)# registration target address 198.51.100.101
Router(config-sbc-sbe-adj-sip)# attach
```

Configuring a Phone Proxy for Nonsecure Access for a Single Cluster

Figure 39 shows a setup in which phones outside the cluster but within the trusted network access phones within the cluster.

Figure 39 Noncluster Phones Access Phones Within the Cluster



Perform the following steps to configure the phone proxy and the other entities in this operating environment. Note that some commands have been omitted from these steps for the sake of brevity.

Step 1 If there is an existing self-signed trustpoint and certificate, skip this step and directly proceed to the next step. Otherwise, create an RSA key-pair and a self-signed trustpoint and generate the certificate by running the following commands:

```
Router(config)# crypto key generate rsa label pp_rsa modulus 1024 general-keys
Router(config)# crypto pki trustpoint self_trustpoint
Router(config-ca-trustpoint)# enrollment selfsigned
Router(config-ca-trustpoint)# subject-name CN=SBC-Phone-Proxy,OU=my_BU,O=my_company
Router(config-ca-trustpoint)# rsakeypair pp_rsa
Router(config-ca-trustpoint)# crypto pki enroll self_trustpoint
```

Step 2 Create the CTL file by running the following commands:

```
Router(config)# sbc ctl-file myctl
Router(config-ctl-file)# record-entry selfsigned trustpoint self_trustpoint
Router(config-ctl-file)# complete
```

Step 3 Create the phone proxy by running the following commands:

```
Router(config)# sbc phone-proxy mypp_1
Router(config-phone-proxy)# tftp-server address ipv4 198.51.100.101 local-address ipv4
192.168.0.109 vrf vrf1
Router(config-phone-proxy)# ctl-file myctl
Router(config-phone-proxy)# no access-secure
Router(config-phone-proxy)# complete
```

Step 4 Define the TFTP port range by running the following commands:

```
Router(config)# sbc phone-proxy tftp-address ipv4 192.0.2.10 vrf vrf1
Router(config-pp-pr)# port-range 30000 60000
Router(config)# sbc phone-proxy tftp-address ipv4 192.0.2.10
Router(config)# sbc phone-proxy tftp-address ipv4 10.10.2.6
Router(config-pp-pr)# port-range 35000 55000
```

Step 5 Configure the SIP adjacency by running the following commands:

```
Router(config)# sbc sbc-name
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip option-editor oe-all
Router(config-sbc-sbe)# blacklist
Router(config-sbc-sbe)# sip option-profile op-all
Router(config-sbc-sbe)# blacklist
Router(config-sbc-sbe)# sip header-profile h1
Router(config-sbc-sbe)# blacklist
Router(config-sbc-sbe)# sip method-profile mp1
Router(config-sbc-sbe-sip-mth)# blacklist
Router(config-sbc-sbe-sip-mth)# pass-body
Router(config-sbc-sbe)# sip header-editor acc-in
Router(config-sbc-sbe-sip-hdr)# blacklist
Router(config-sbc-sbe-sip-hdr)# store-rule entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-name from header-value
regex-match "^.*sips*:\([^@][^@]*\)@.*$" store-as fu
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and is-request eq true
Router(config-sbc-sbe-sip-hdr)# store-rule entry 2
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-name Contact header-value
regex-match "^\(.*sips*:\)\([^@][^@]*\)\(@[^;>][^;>]*\)\(.*\)$" store-as bcu cu ch ach
Router(config-sbc-sbe-sip-hdr)# store-rule entry 3
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-name to header-value regex-match
"^.*sips*:\([^@][^@]*\)@.*$" store-as tu
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and is-request eq false
Router(config-sbc-sbe-mep-hdr-ele-act)# store-rule entry 4
Router(config-sbc-sbe-mep-hdr-ele-act)# condition header-name Event header-value
regex-match "\(.*\)call-id=\(^[\";]*\)\(.*\)" store-as b_callid callid a_callid
Router(config-sbc-sbe-mep-hdr-ele-act)# condition and is-request eq true
Router(config-sbc-sbe-mep-hdr-ele-act)# header event entry 1
Router(config-sbc-sbe-mep-hdr-ele)# action replace-value value
"${b_callid}call-id=${callid}${a_callid}"
Router(config-sbc-sbe-mep-hdr-ele-act)# condition is-request eq true
Router(config-sbc-sbe-mep-hdr-ele-act)# condition and variable callid is-defined eq true
Router(config-sbc-sbe-sip-hdr)# header contact entry 1
Router(config-sbc-sbe-sip-hdr-ele)# action replace-value value
"${bcu}${fu}${ch};origUser=${cu}${ach}"
Router(config-sbc-sbe-sip-hdr-ele)# condition variable cu is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele)# condition and is-request eq true
Router(config-sbc-sbe-sip-hdr)# header contact entry 2
Router(config-sbc-sbe-sip-hdr-ele)# action replace-value value
"${bcu}${tu}${ch};origUser=${cu}${ach}"
Router(config-sbc-sbe-sip-hdr-ele)# condition variable cu is-defined eq true
```

```
Router(config-sbc-sbe-sip-hdr-ele)# condition and is-request eq false
Router(config-sbc-sbe)# sip header-editor acc-out
Router(config-sbc-sbe-sip-hdr)# blacklist
Router(config-sbc-sbe-sip-hdr)# store-rule entry 1
Router(config-sbc-sbe-sip-hdr-ele)# condition header-name contact header-value regex-match
"^\([[:space:]<]*sips*\):\([^@]*\)@\(.*\);origUser=\([^;>]*\)\(.*\)" store-as cbu cu ch
cou cr
Router(config-sbc-sbe-sip-hdr-ele)# condition and is-request eq false
Router(config-sbc-sbe-sip-hdr)# store-rule entry 2
Router(config-sbc-sbe-sip-hdr)# header contact entry 1
Router(config-sbc-sbe-sip-hdr-ele)# action replace-value value "${cbu}:${cou}@${ch}${cr}"
Router(config-sbc-sbe-sip-hdr-ele)# condition is-request eq false
Router(config-sbc-sbe-sip-hdr-ele)# condition and variable cbu is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele)# condition and variable cou is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele)# condition and variable ch is-defined eq true
Router(config-sbc-sbe)# adjacency sip access-1
Router(config-sbc-sbe-adj-sip)# editor-type editor
Router(config-sbc-sbe-adj-sip)# header-editor inbound acc-in
Router(config-sbc-sbe-adj-sip)# header-editor outbound acc-out
Router(config-sbc-sbe-adj-sip)# option-editor ua inbound oe-all
Router(config-sbc-sbe-adj-sip)# option-editor ua outbound oe-all
Router(config-sbc-sbe-adj-sip)# inherit profile preset-access
Router(config-sbc-sbe-adj-sip)# signaling-address ipv4 192.0.2.10
Router(config-sbc-sbe-adj-sip)# remote-address ipv4 69.118.122.5 255.255.255.255
Router(config-sbc-sbe-adj-sip)# signaling-peer 69.118.122.5
Router(config-sbc-sbe-adj-sip)# phone-proxy mypp_1
Router(config-sbc-sbe-adj-sip)# registration rewrite-register
Router(config-sbc-sbe-adj-sip)# attach
Router(config-sbc-sbe)# adjacency sip access-2
Router(config-sbc-sbe-adj-sip)# editor-type editor
Router(config-sbc-sbe-adj-sip)# header-editor inbound acc-in
Router(config-sbc-sbe-adj-sip)# header-editor outbound acc-out
Router(config-sbc-sbe-adj-sip)# option-editor ua inbound oe-all
Router(config-sbc-sbe-adj-sip)# option-editor ua outbound oe-all
Router(config-sbc-sbe-adj-sip)# inherit profile preset-access
Router(config-sbc-sbe-adj-sip)# signaling-address ipv4 10.10.2.6
Router(config-sbc-sbe-adj-sip)# remote-address ipv4 69.118.130.9 255.255.255.255
Router(config-sbc-sbe-adj-sip)# signaling-peer 209.165.201.9
Router(config-sbc-sbe-adj-sip)# phone-proxy mypp_1
Router(config-sbc-sbe-adj-sip)# registration rewrite-register
Router(config-sbc-sbe-adj-sip)# attach
Router(config-sbc-sbe)# adjacency sip core-1
Router(config-sbc-sbe-adj-sip)# vrf vrf1
Router(config-sbc-sbe-adj-sip)# force-signaling-peer
Router(config-sbc-sbe-adj-sip)# header-profile inbound h1
Router(config-sbc-sbe-adj-sip)# header-profile outbound h1
Router(config-sbc-sbe-adj-sip)# method-profile inbound mp1
Router(config-sbc-sbe-adj-sip)# method-profile outbound mp1
Router(config-sbc-sbe-adj-sip)# option-profile ua inbound op-all
Router(config-sbc-sbe-adj-sip)# option-profile ua outbound op-all
Router(config-sbc-sbe-adj-sip)# inherit profile preset-core
Router(config-sbc-sbe-adj-sip)# signaling-address ipv4 198.51.100.5
Router(config-sbc-sbe-adj-sip)# remote-address ipv4 198.51.100.101 255.255.255.255
Router(config-sbc-sbe-adj-sip)# signaling-peer 198.51.100.101
Router(config-sbc-sbe-adj-sip)# registration target address 198.51.100.101
Router(config-sbc-sbe-adj-sip)# attach
```

Removing a Phone Proxy

The following example shows how to remove a phone proxy:

```
Router# configure terminal
Router(config)# sbc mysbc
Router(config)# sbe
Router(config-sbc-sbe)# adjacency sip pp-adj
Router(config-sbc-sbe-adj-sip)# no attach
Router(config-sbc-sbe-adj-sip)# no phone-proxy pp
Router(config-sbc-sbe-adj-sip)# exit
Router(config-sbc-sbe)# exit
Router(config-sbc)# exit
Router(config)# sbc phone-proxy pp
Router(config-phone-proxy)# no complete
Router(config-phone-proxy)# exit
Router(config)# no sbc phone-proxy pp
Router(config)# sbc ctl-file ctl-1
Router(config-ctl-file)# no complete
Router(config-ctl-file)# exit
Router(config)# no sbc ctl-file ctl-1
```