

Configure Cisco Contact Center SIP Proxy (CCCSP)

- Configure Logical Networks, on page 1
- Configure Trigger Conditions, on page 2
- Configure Server Groups, on page 4
- Configure Route Tables, on page 6
- Configure Normalization Policies, on page 7
- Configure Lookup Policies, on page 8
- Configure Routing Triggers, on page 10
- Configure Normalization Triggers, on page 11
- Configure Listen and Record-Route Ports, on page 12
- Configure a Hostname, on page 13
- Configure Transport Layer Security (TLS), on page 14
- Update Web Session with an Imported Signed Certificate, on page 24
- Configure Lite Mode, on page 25
- Configure Call Rate Limit, on page 26
- Commit the Configuration, on page 27

Configure Logical Networks

Before you begin

Each interface on the Cisco Contact Center SIP Proxy (CCCSP) is associated with a logical network. Logical networks are used to organize server groups, listen points, and other properties. SIP messages are associated with the network on which they arrive.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **sip network** *network*
4. **end network**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP EXEC mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters Cisco Unified SIP Proxy configuration mode.
Step 3	**sip network** *network* **Example:** `Hostname(cusp-config)# sip network service-provider`	Creates a network and puts you into network command mode. In this case, the network that is being created is called "service provider".
Step 4	**end network** **Example:** `Hostname(cusp-config-network)# end network`	Exits network command mode.

Example

The following example creates a network called **service-provider**:

```
Hostname# cusp
Hostname(cusp)# configure
Hostname(cusp-config)# sip network service-provider
Hostname(cusp-config-network)# end network
```

Configure Trigger Conditions

Before you begin

You create trigger conditions to allow Cisco Contact Center SIP Proxy (CCCSP) to respond with the appropriate action for various call flows. In general, the more complex the call flow is, the more complex the trigger must be.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **trigger condition** *trigger-condition-name*

4. **sequence** *sequence-number*
5. **in-network** *network-name*
6. **mid-dialog**
7. **end sequence**
8. **end trigger condition**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters Cisco Unified SIP Proxy EXEC mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters Cisco Unified SIP Proxy configuration mode.
Step 3	**trigger condition** *trigger-condition-name* **Example:** `Hostname(cusp-config)# trigger condition call-from-service-provider`	Creates a trigger condition and puts you into trigger command mode. In this case, the trigger that is being created is called “call-from-service-provider”.
Step 4	**sequence** *sequence-number* **Example:** `Hostname(cusp-config-trigger)# sequence 1`	Creates a sequence with the specified number and puts you into trigger sequence command mode. The number indicates the order in which triggers are evaluated. In this case, the sequence that is being created is sequence number 1.
Step 5	**in-network** *network-name* **Example:** `Hostname(cusp-config-trigger-seq)# in-network service-provider`	Specifies the incoming network name for the trigger condition. In this case, the incoming network is the “service-provider” network.
Step 6	**mid-dialog** **Example:** `Hostname(cusp-config-trigger-seq)# mid-dialog`	A special trigger that bypasses routing policies on mid-dialog messages.
Step 7	**end sequence** **Example:** `Hostname(cusp-config-trigger-seq)# end sequence`	Exits the trigger sequence command mode.
Step 8	**end trigger condition** **Example:** `Hostname(cusp-config-trigger)# end trigger condition`	Exits the trigger command mode.

Example

In this example, Cisco Contact Center SIP Proxy (CCCSP) only reacts based on the network the call came in on, so the triggers are simple.

```
Hostname# cusp

Hostname(cusp)# configure
Hostname(cusp-config)# trigger condition call-from-service-provider
Hostname(cusp-config-trigger)# sequence 1
Hostname(cusp-config-trigger-seq)# in-network service-provider
Hostname(cusp-config-trigger-seq)# end sequence
Hostname(cusp-config-trigger)# end trigger condition

Hostname(cusp-config)# trigger condition mid-dialog
Hostname(cusp-config-trigger)# sequence 1
Hostname(cusp-config-trigger-seq)# mid-dialog
Hostname(cusp-config-trigger-seq)# end sequence
Hostname(cusp-config-trigger)# end trigger condition
```

Configure Server Groups

Server groups define the elements that Cisco Contact Center SIP Proxy (CCCSP) interacts with for each network. The server group name that is used is inserted into the SIP URI of the outgoing request. Some devices, such as Cisco Unified Communications Manager, validate the URI of requests before processing, which means that the end device might need to be configured with a Fully Qualified Domain Name (FQDN) to allow for this.

Two of the fields for each individual element, q-value and weight, are important to use to specify the priorities of elements, and also for load balancing. Calls are routed to specific elements based on q-value. The element with the highest q-value receives all traffic routed to that server group. If multiple elements have the same q-value, traffic is distributed between them based on the load-balancing option used. The default load-balancing is based on call-id, but weight can also be used. If weight is used, the percentage of traffic that an element receives is equal to its weight divided by the sum of up elements with the same q-value's weights. The sum of their weights does not need to equal 100. You can change the weights and q-values to configure a different priority or load-balancing scheme.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **server-group sip group** *server-group-name network*
4. **element ip-address** *ipaddress port* {**udp** | **tcp** | **tls**} [**q-value** *q-value*] [**weight** *weight*]
5. **lbtype {global | highest-q | request-uri | call-id | to-uri | weight}**
6. **end server-group**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP Management mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters CCCSP Configuration mode.
Step 3	**server-group sip group** *server-group-name network* **Example:** `Hostname(cusp-config)# server-group sip group sp.example.com service-provider`	Creates a SIP server group and enters server group command mode. In this case, the server group being created is called "sp.example.com" and it uses the network called "service-provider".
Step 4	**element ip-address** *ipaddress port* {**udp** | **tcp** | **tls**} [**q-value** *q-value*] [**weight** *weight*] **Example:** `Hostname(cusp-config-sg)# element ip-address 192.168.10.3 5060 tls q-value 1.0 weight 100`	Creates an IP element for a SIP server group and determines the characteristics of the SIP server group. **Note** You can enter this command multiple times.
Step 5	**lbtype {global | highest-q | request-uri | call-id | to-uri | weight}** **Example:** `Hostname(cusp-config-sg)# lbtype weight`	Configures the load-balancing algorithm for the SIP server group. In this example, it specifies that the element will be selected proportional to its weight relative to the weights of other elements of the same q-value.
Step 6	**end server-group** **Example:** `Hostname(cusp-config-sg)# end server-group`	Exits the server group command mode.

Example

```
Hostname# cusp

Hostname(cusp)# configure
Hostname(cusp-config)# server-group sip group sp.example.com service-provider
Hostname(cusp-config-sg)# element ip-address 192.168.10.3 5060 tls q-value 1.0 weight 100
Hostname(cusp-config-sg)# element ip-address 192.168.10.4 5060 tls q-value 1.0 weight 50
Hostname(cusp-config-sg)# element ip-address 192.168.10.5 5060 tls q-value 1.0 weight 50
Hostname(cusp-config-sg)# lbtype weight
Hostname(cusp-config-sg)# end server-group
```

Configure Route Tables

You must configure route tables to direct SIP requests to their appropriate destinations. Each route table consists of a set of keys that are matched based on the lookup policy. For example, each key might represent the prefix of a phone number dialed.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **route table** *table-name*
4. **key** *key* **response** *reponse-code*
5. **key** *key* **target-destination** *target-destination netowrk*
6. **end route table**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP Management mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters CCCSP Configuration mode.
Step 3	**route table** *table-name* **Example:** `Hostname(cusp-config)# route table service-provider-table`	Creates a route table and enters route table command mode. In this case, it creates a route table called "service-provider-table".
Step 4	**key** *key* **response** *reponse-code* **Example:** `Hostname(cusp-config-rt)# key * response 404`	Assigns a response code to a lookup key. In this example, it returns a response of "404" to everything.
Step 5	**key** *key* **target-destination** *target-destination netowrk* **Example:** `Hostname(cusp-config-rt)# key 510 target-destination cube-sp.example.com cube-sp`	Replaces the key part of the target destination with a specified value. **Note** You can enter this command multiple times.
Step 6	**end route table** **Example:**	Exits the route table command mode.

	Command or Action	Purpose
	`Hostname(cusp-config-rt)# end route table`	

Example

```
Hostname# cusp

Hostname(cusp)# configure
Hostname(cusp-config)# route table service-provider-table
Hostname(cusp-config-rt)# key * response 404
Hostname(cusp-config-rt)# key 510 target-destination cube-sp.example.com cube-sp
Hostname(cusp-config-rt)# end route table
```

Configure Normalization Policies

Normalization policies modify SIP messages to account for incompatibilities between networks. In this case, the service provider cannot handle phone numbers with the escape sequence “91,” so the sequence must be removed from the request-uri and TO header.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **policy normalization** *policy-name*
4. **uri-component update request-uri {user | host | host-port | phone | uri}** {**all** | *match-string*} *replace-string*
5. **uri-component update header { first | last | all} {user | host | host-port | phone | uri }** {**all** | *match-string*} *replace-string*
6. **end policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP Management mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters CCCSP Configuration mode.

	Command or Action	Purpose
Step 3	**policy normalization** *policy-name* **Example:** `Hostname(cusp-config)# policy normalization outgoing-norm-policy`	Creates a normalization policy and enters policy normalization command mode. In this example, the normalization policy is called “outgoing-norm-policy”.
Step 4	**uri-component update request-uri {user | host | host-port | phone | uri}** {**all** | *match-string*} *replace-string* **Example:** `Hostname(cusp-config-norm)# uri-component update request-uri user ^91 ""`	Configures a normalization policy step that updates a URI component field within a request URI.
Step 5	**uri-component update header { first | last | all} {user | host | host-port | phone | uri }** {**all** | *match-string*} *replace-string* **Example:** `Hostname(cusp-config-norm)# uri-component update TO all user ^91 ""`	Configures a normalization policy step that updates a URI component field within a header of the source message.
Step 6	**end policy** **Example:** `Hostname(cusp-config-norm)# end policy`	Exits policy normalization command mode.

Example

```
Hostname# cusp

Hostname(cusp)# configure
Hostname(cusp-config)# policy normalization outgoing-norm-policy
Hostname(cusp-config-norm)# uri-component update request-uri user ^91 ""
Hostname(cusp-config-norm)# uri-component update TO all user ^91 ""
Hostname(cusp-config-norm)# end policy
```

Configure Lookup Policies

Lookup policies decide how the keys in the route tables are used. Each key represents the beginning of the phone number dialed because each policy states to match the user component of the request-uri against the keys in its route table. The user component of the request-uri is the phone number called. The rule used to match is prefix, which means that the longest prefix match in the route table is used. So if the dialed number is 510-1XX-XXXX, the call is sent to the cme.example.com server group. If the dialed number is 510-XXX-XXXX, the call is sent to the cucm.example.com server group. The four policies in the following example are identical, except that they each refer to their specific table.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **policy lookup** *policy-name*
4. **sequence** *sequence-number table-name* **field {in-network | local-ip-address | local-ip-port | remote-ip-address | remote-ip-port} | header {p-asserted identity| from | to | diversion | remote-party-id} | request uri [uri component {param| user | phone | host| host-port| uri}]**
5. **rule {exact | prefix | subdomain | subnet | fixed** *length}* [**case-insensitive**]
6. **end sequence**
7. **end policy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP Management mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters CCCSP Configuration mode.
Step 3	**policy lookup** *policy-name* **Example:** `Hostname(cusp-config)# policy lookup service-provider-policy`	Creates a policy with the specified name and enters policy lookup command mode. In this case, creates a policy called "service-provider-policy".
Step 4	**sequence** *sequence-number table-name* **field {in-network	local-ip-address	local-ip-port	remote-ip-address	remote-ip-port}	header {p-asserted identity| from	to	diversion	remote-party-id}	request uri [uri component {param| user	phone	host| host-port| uri}]** **Example:** `Hostname(cusp-config-lookup)# sequence 1`	Creates a sequence with the specified number and enters policy lookup sequence command mode. Sequences are performed according to the order of their number.
Step 5	**rule {exact	prefix	subdomain	subnet	fixed** *length}* [**case-insensitive**] **Example:** `Hostname(cusp-config-lookup-seq)# rule prefix`	Creates a rule that determines the routing algorithm for the lookup policy. In this case, it creates a rule that specifies that the lookup policy searches for the longest prefix match.

	Command or Action	Purpose
Step 6	**end sequence** **Example:** `Hostname(cusp-config-lookup-seq)# end sequence`	Exits policy lookup sequence command mode.
Step 7	**end policy** **Example:** `Hostname(cusp-config-lookup)# end policy`	Exits policy lookup command mode.

Example

```
Hostname# cusp

Hostname(cusp)# configure
Hostname(cusp-config)# policy lookup service-provider-policy
Hostname(cusp-config-lookup)# sequence 1 service-provider-table request-uri uri-component
user
Hostname(cusp-config-lookup-seq)# rule prefix
Hostname(cusp-config-lookup-seq)# end sequence
Hostname(cusp-config-lookup)# end policy
```

Configure Routing Triggers

Routing triggers correlate trigger conditions with lookup policies. A single policy is chosen based on which corresponding condition is matched. The conditions are evaluated in ascending order based on sequence number. The mid-dialog condition is the first one so that the policy step is skipped for mid-dialog messages. Based on the following configuration, after the INVITE message is successfully routed, all subsequent messages (which are mid-dialog) bypass routing policies.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **trigger routing sequence** *sequence-number* { **by-pass | policy** *policy*} [**condition** *trigger-condition*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP Management mode.

	Command or Action	Purpose
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters CCCSP Configuration mode.
Step 3	**trigger routing sequence** *sequence-number* { **by-pass** | **policy** *policy*} [**condition** *trigger-condition* **Example:** `Hostname(cusp-config)# trigger routing sequence 2 policy service-provider-policy condition call-from-service-provider`	Associates a routing policy with a trigger condition. In this example, the second sequence follows the previously-created policy called “service-provider-policy” and the previously-created trigger called “call-from-service-provider”.

Example

```
Hostname# cusp

Hostname(cusp)# configure
Hostname(cusp-config)# trigger routing sequence 1 by-pass condition mid-dialog
Hostname(cusp-config)# trigger routing sequence 2 policy service-provider-policy condition
 call-from-service-provider
Hostname(cusp-config)# trigger routing sequence 3 policy cube-sp-policy condition
call-from-cube-sp
Hostname(cusp-config)# trigger routing sequence 4 policy cube-es-policy condition
call-from-cube-es
Hostname(cusp-config)# trigger routing sequence 5 policy enterprise-policy condition
call-from-enterprise
```

Configure Normalization Triggers

Normalization triggers correlate trigger conditions with normalization policies. There are two types of triggers: pre-normalization, which occurs before routing, and post-normalization, which occurs after routing. Similar to routing policies, a special policy bypasses normalization on mid-dialog messages.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **trigger pre-normalization sequence** *sequence-number* { **by-pass | policy** *policy*} [**condition** *trigger-condition*]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP Management mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters CCCSP Configuration mode.
Step 3	**trigger pre-normalization sequence** *sequence-number* { **by-pass | policy** *policy*} [**condition** *trigger-condition*] **Example:** `Hostname(cusp-config)# trigger pre-normalization sequence 2 policy outgoing-norm-policy condition call-from-cube-sp`	Configures a pre-normalization algorithm for incoming SIP messages to a normalization policy. In this example, the second sequence follows the previously-created policy called "outgoing-norm-policy" and the previously-created trigger called "call-from-cube-sp".

Example

```
Hostname# cusp

Hostname(cusp)# configure
Hostname(cusp-config)# trigger pre-normalization sequence 1 by-pass condition mid-dialog
Hostname(cusp-config)# trigger pre-normalization sequence 2 policy outgoing-norm-policy
condition call-from-cube-sp
```

Configure Listen and Record-Route Ports

You must configure listen and record-route ports for each network. For the listen and record-route ports, the actual addresses of the Cisco Contact Center SIP Proxy (CCCSP) application are used. The **sip record-route** command inserts the record-route header into outgoing requests. The **sip listen** command allows for CCCSP to accept incoming requests on that port.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **sip record-route** *network_name* { **tcp | tls | udp** } *ip_addess* [*port*]
4. **sip listen** *network_name* { **tcp | tls | udp** } *ip_addess* [*port*]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP Management mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters CCCSP Configuration mode.
Step 3	**sip record-route** *network_name* { **tcp** | **tls** | **udp** } *ip_addess* [*port*] **Example:** `Hostname(cusp-config)# sip record-route service-provider udp 10.10.10.99 5060`	Enables record-routing for a SIP network. In this example, the "service-provider" network is associated with a record-route configuration and the IP address that populates the record-route header field is "10.10.10.99" and the port that populates the record-route header is 5060.
Step 4	**sip listen** *network_name* { **tcp** | **tls** | **udp** } *ip_addess* [*port*] **Example:** `Hostname(cusp-config)# sip listen service-provider udp 10.10.10.99 5060`	Creates a listener that listens for SIP traffic on a specific SIP network, host, and port.

Example

```
Hostname# cusp

Hostname(cusp)# configure
Hostname(cusp-config)# sip record-route service-provider udp 10.10.10.99 5060
Hostname(cusp-config)# sip listen service-provider udp 10.10.10.99 5060
```

Configure a Hostname

If the upstream element is using DNS SRV for routing to the two CCCSPs in a network, you must configure the two CCCSPs to have the same FQDN by entering the **sip alias** command in the configuration mode on both CCCSPs.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **sip alias** *hostname*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP Management mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters CCCSP Configuration mode.
Step 3	**sip alias** *hostname* **Example:** `Hostname(cusp-config)# sip alias myhost`	Configures the hostname of this instance.

Example

```
Hostname# cusp

Hostname(cusp)# configure
Hostname(cusp-config)# sip alias myhost
```

Configure Transport Layer Security (TLS)

Create and Import a Signed Certificate

Cisco Contact Center SIP Proxy (CCCSP) supports TLS, Transmission Control Protocol (TCP), and User Datagram Protocol (UDP). Establishing TLS connections requires some extra steps because the connections require authentication using signed certificates.

Before you begin

You need an SFTP server or HTTP to import certificate requests.

Use the **crypto key delete all** command and delete all the existing certificates before importing new certificates.



Note Use the **show crypto key all** command to verify the key generation. Note down the **Label** from the output displayed.

Note When CCCSP operates as a SIP TLS client, it prioritizes the use of RSA ciphers and signature algorithms. If CCCSP connects to applications such as CUCM that present both RSA and ECDSA certificates, these applications favors the RSA certificate.

Note SHA1 is deprecated and not secure. When handling externally generated certificates, ensure that SHA1 is not used for certificate generation or signing. We recommend you to use stronger algorithms such as SHA256 or higher.

SUMMARY STEPS

1. **configure terminal**
2. **crypto key generate** [**rsa** { **label** *label-name* | **modulus** *modulus-size* } | **default**]
3. **crypto key certreq label** *label-name* **url** { **sftp:** | **http:** }
4. Use your CA infrastructure or use a 3rd party CA to get the CSR signed (external) before importing it in the next step.
5. **crypto key import trustcacert label** *label-name* { **der url**{ **sftp: | http:** } | **pem { terminal | url {sftp: | http: }} [default]**
6. **crypto key import cer label** *label-name* { **der url**{ **sftp: | http:** } | **pem { terminal | url {sftp: | http: }} [default]**
7. **offline**
8. **reload**

DETAILED STEPS

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`Hostname# configure terminal`
	`Enter configuration commands, one per line. End with CNTL/Z.`
	`Hostname(config)#`
		Enters configuration mode.

Step 2	**crypto key generate** [**rsa** { **label** *label-name* | **modulus** *modulus-size* } | **default**]
	Example:
	`Hostname(config)# crypto key generate rsa label mykey modulus 2048 default`
	`Key generation in progress. Please wait...`
	`The label name for the key is Hostname`
	`Hostname(config)#`
		Creates an 2048-bit RSA private key.

	Command or Action	Purpose
	Hostname(config)# **exit** Hostname# **write**	
Step 3	**crypto key certreq label** *label-name* **url** { **sftp:** | **http:** } **Example:** Hostname(config)# **crypto key certreq label mykey url sftp:** Address or name of remote host? 10.64.x.y Username (ENTER if none)? user Password (not shown)? Destination path? /tmp/ Uploading CSR file succeed	Creates a certificate request to be signed.
Step 4	Use your CA infrastructure or use a 3rd party CA to get the CSR signed (external) before importing it in the next step.	
Step 5	**crypto key import trustcacert label** *label-name* { **der url**{ **sftp: | http:** } | **pem { terminal | url {sftp: | http: }} [default]** **Example:** Hostname(config)# **crypto key import trustcacert label rootCA terminal** Enter certificate... End with a blank line or "quit" on a line by itself -----BEGIN CERTIFICATE----- MIIFszCCA5ugAwIBAgIUNh1Jsm5FO6QkWXoAHzQQJBJ6eT4wDQYJKoZIhvcNAQEL BQAwaTELMAkGA1UEBhMCSU4xEjAQBgNVBAgMCUthcm5hdGFrYTESMBAGA1UEBwwJ QmFuZ2Fsb3JlMRQwEgYDVQQKDAtDb2xsYWJhdGlvbjEcMBoGA1UEAwwTQ29sbGFi YXRpb24gUm9vdCBDQTAeFw0yNTA3MTYwOTA4MTJaFw0zNTA3MTQwOTA4MTJaMGkx CzAJBgNVBAYTAklOMRIwEAYDVQQIDAlLYXJuYXRha2ExEjAQBgNVBAcMCUJhbmdh bG9yZTEUMBIGA1UECgwLQ29sbGFiYXRpb24xHDAaBgNVBAMME0NvbGxhYmF0aW9u IFJvb3QgQ0EwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQDB1d/DJi9q KFjG0McBCWsPQIojQE6j61R2pqgqZdKGfuoi68B/zWctspMck6DvvQ/KNCQ8LYoa Cr3QL6dJAxRkLGBu2FBV5NattAybaEoJNZRlwnCKfPTksUNnL/U/PkooQU+Bn8WN v+dh8zTL+ffN9uRuDJPq7rptn2fZfk+2IT3x2XNQMYOZuwIcvG/urB6sSbUFJu0x JUl9Kv//HX9COyOrnRW3g7tq/CT0eObAoIddqf12H7woCL3PiR/YwsmtfwGXL/sk xLONfbP1wxq/WdjgX/QBc07vZF9uUjsIupzEKLqvjBK1CsE4bYxKkgrreik320wK ySvSgdFfmnhA+G2V7FqlHAV9s1LBrhaqnCWDQFMOzEDqkKn+UAcATD3u9b7NdcPn NKKxKMtvC7A+JDbI2naMPASPI99shTE0Pr0ZwaYL1TVN+UlDCRYYqtTDHm1PEi+R GNgeW99/LeW49n0qHtqw8xb5OtQbGUAEQaOztnKWUR+nQ19pCHuNflBDtuKGSvzk iE+vHFSHo9L66kU+oVIMziiqJ1jRnnWMaYiQy077XWB1/jkC5593t1cmai6DUEeP 91INDbXiIcYNnJGK/JKSIp0JJ1N5gyyPAoXWv7sRG2Cu0VNaXtkAk10KHnnTIHJp fP1oCiJtZ8RdJmRthztLUs13MBHy1+fEmwIDAQABo1MwUTAdBgNVHQ4EFgQUksR4 KNEr9C+AeTd4z5sagT2Y+NAwHwYDVR0jBBgwFoAUksR4KNEr9C+AeTd4z5sagT2Y +NAwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAgEACu84XDsAO3EU tyhVIvSKSyOXJen1iqw6vK1idnMEbre00a4hinOPIVbi+O9+o1vL4KIstoPBkM7+ YffWXhRewX2HyHu/W4d7aNowVaXB8S2IGjisQf3vdrA2IAIU65DdsZwDcNCsg8HA BDE6XkrA0FOD1oW3ge6z5anIThwqp3mofi3jcT6RacmibRAf8cKPi3xfKAxdgSNM 0ZOmoR8ahhINszo6puLoYdV1GDBR4UMPl+qhktd4t/mW5nXfcHfIKbCUecjFP4td 5rOIpJOxHJQBlksxUPul7FXKfOPkiDyDRVnJzz+yPpqukmimxx2KQUyEkX/z24SR uRVr77VIKi6ZQBi5Lq3+rPstrTdNtet4o1InJl3yjJqlottDvmaSAsbqN9z85/Dz MjsIHHGA63aevJo7/3R7v1o9qYigwMd1Tz2Qhz7rXVLFZikHn2ZKEYeIzA3sCxgI p08qIvwyERrAoAYrSBmtMyDy7MtESplty3gJ7jcmSnm6uAF3JV8j7vjooKVIT7Ne Kng+7R7GqnnJYeezRPAgwyEcHt50ng4B98I98q+AHFaWyMtpW9zo4kYkd1G/Nntx nMqoDo6QwICy5GbvAaXbzsGfQm8ULDeS9Zk0AdmSpZalRsplym0mRTfnTcsRkwjk	After the certificate request is signed, imports the trusted certificate authority (CA) certificate that you used to sign the request. **Note** Trusted CA validity must be between 5 to 30 years, else the import will fail.

	Command or Action	Purpose
	KOmRkBwUNMu2DdByY9fcuVZu+RcXgkw= -----END CERTIFICATE----- Certificate info ********** Owner: CN=Collabation Root CA,O=Collabation,L=Bangalore,ST=Karnataka,C=IN Issuer: CN=Collabation Root CA,O=Collabation,L=Bangalore,ST=Karnataka,C=IN Valid from: Wed Jul 16 02:08:12 PDT 2025 until: Sat Jul 14 02:08:12 PDT 2035 Certificate fingerprint (MD5): 9D:D4:F6:0C:08:12:2A:7E:EE:64:16:93:D9:28:AF:55 Do you want to continue to import this certificate, additional validation will be perfomed? [y/n]: y Hostname(config)#	
Step 6	**crypto key import cer label** *label-name* { **der url**{ **sftp: | http:** } | **pem { terminal | url {sftp: | http: }} [default]** **Example:** Hostname(config)# **crypto key import cer label mykey terminal** Enter certificate... End with a blank line or "quit" on a line by itself -----BEGIN CERTIFICATE----- MIIESjCCAjICFC9ZC6cO5IXQSmjDeP7z3uWMgmLPMA0GCSqGSIb3DQEBCwUAMGkx CzAJBgNVBAYTAklOMRIwEAYDVQQIDAlLYXJuYXRha2ExEjAQBgNVBAcMCUJhbmdh bG9yZTEUMBIGA1UECgwLQ29sbGFiYXRpb24xHDAaBgNVBAMME0NvbGxhYmF0aW9u IFJvb3QgQ0EwHhcNMjUwNzI5MDcyMDIxWhcNMjYwNzI5MDcyMDIxWjBaMQswCQYD VQQGEwInJzELMAkGA1UECAwCJycxCzAJBgNVBAcMAicnMQswCQYDVQQKDAInJzEL MAkGA1UECwwCJycxFzAVBgNVBAMMDnZtNTYuY2lzY28uY29tMIIBIjANBgkqhkiG 9w0BAQEFAAOCAQ8AMIIBCgKCAQEAoyi9yh/+/W3F+SyJNebEDeouCItlba+mSv9n ztADtMKsxDRPd+4gO4xPFeoVGPtuixuVtlxIGXB9lzfrie0Byk3v/kmwkCPDLaBY OWXQAzPqf/4EDBQZRQ9+7yGqtfFnn5aiLojFefGvkniR2Z7YXzCMHDdjDic9ygUm /Hg6qvpOwRrBCCOrLGOb+giG0c2vvjtSq9fYqI6VyJftjRqE5CTODWpXMxlpxGS8 XUATJcXaJkdqDWbeHFOlqhNCBYSQEiSsP/gJOrg06lIAEApnrBHUrXRf+fKNGPmJ en54KqR37TiONPBy0i6YNnXbFDXexETJZxx6/dNcQSZVoDId8QIDAQABMA0GCSqG SIb3DQEBCwUAA4ICAQB2g2i43xaFgBtYxeATxY+rqkYuRUex0x5ojzZsEdlCWV9b wxyRQ5G7p3a81/bm8tnzlN2sUWb86Xjf37mVP7xY0bYM1A/kPNEwpkPVivGHycXQ cX9G98Nl3c1vHB5/v94tb8tcvnpVWs5uemSWDsRbh9x729YcWGQ+ZO7D9sEXk1aF xuvN+5ufx6V7ARu7RVDC0d5Ti5du37gB6731R/E68hN2XHdGMSvCVfnTSnNUbyr8 qhwaArRVTCMqPqSKv0PkhQHgINZQ2p1hXPf38NBdrVnh/2wE0ib0zP/fnY7AsxuW jBJyDcdciXkbDBXzMHGsH/vf0yvj5DAgh33mN/YI9q0f3b9WZUyh/NrFUpeV6Jn/ 5570WJ2XEDFb39L3ok590yHY0NqewPfXYfnJvo2doHltm88V4KKvPzGWAvJ6+8MY IO/8mlA3E2Or5IFjUkOTr/yvX67vlqSzFwArC9Hjxw4VG23mi4fZQ0Qnpj2Mnwrs YEDQ/oP7uRkHjtPwkq35Pyug4ieyWIumKbBvMr+iCdS4sFmKlmKmYteT6s4SIDfQ ryZv6MgqhOhDfEvu/F4lbdmW9SjFxSWd8FDHPgkS8iVmDfbM/sOyVfC6cDVdL0ba JOOmr/uzFBk5smVepZg1fALYgsGeiFR9tItDmouk9fYYbq9jcjHEFqCLWryu6Q== -----END CERTIFICATE----- Import succeeded Hostname(config)#	After the root CA is imported, imports the signed certificate.
Step 7	**offline** **Example:**	Initiates CCCSP offline mode.

	Command or Action	Purpose

	`Hostname# offline`
	
	`!!!WARNING!!!: Putting the system offline will terminate all active calls.`
	
	`Do you wish to continue[n]?: y`	

Step 8	**reload** **Example:** `Hostname(offline)# reload`	Restarts the CCCSP system and enables CCCSP to verify the imported trusted certificate.

Example

Example for creating a signed certificate:

```
Hostname# configure terminal
Hostname(config)# crypto key generate rsa label mykey modulus 2048 default
Key generation in progress. Please wait...
The label name for the key is mykey

Hostname(config)# crypto key certreq label mykey url sftp:
Address or name of remote host? test:test123@192.168.202.216
Username (ENTER if none)? anonymous
Password (not shown)?
Destination path? netmod/mykey.csr
Uploading CSR file succeed

Hostname(config)# crypto key import trustcacert label rootCA url sftp:
Import certificate file...
Address or name of remote host? test:test123@192.168.202.216
Source filename? netmod/rootCA/cacert.pem
1212 bytes received.

Hostname(config)# crypto key import cer label mykey url sftp:
Import certificate file...
Address or name of remote host? test:test123@192.168.202.216
Source filename? netmod/mycert.cer
952 bytes received.
Import succeeded
```

What to do next

- Import the trusted CA certificates for any of the TLS peer elements.

Create and Import a Self-Signed Certificate

Self signed certificates are generated using SSL utilities on a Linux or Windows machine and are imported to CCCSP.

SUMMARY STEPS

1. **vim** *filename*

2. **openssl req -new -newkey rsa:2048 -nodes -keyout** *key* **-out** *csr* **-config** *configuration file name*
3. **openssl x509 -req -days** <*days*> **-in** <*csr*>**-signkey** <*keys*> **-out** <*certificate*>
4. **configure terminal**
5. **crypto key import trustcacert label** <*label_name*> **terminal**
6. **crypto key import rsa label** *label-name* **pem terminal**
7. **offline**
8. **reload**

DETAILED STEPS

Procedure

	Command or Action	Purpose

Step 1	**vim** *filename*
	Example:
	This is an example only. You can use any text editor.
	
	`Linux-server-test$ vim abc`
	`distinguished_name = req_distinguished_name`
	`[ req_distinguished_name ]`
	`countryName = Country Name (2 letter code)`
	`countryName_default = US`
	`countryName_min = 2`
	`countryName_max = 2`
	
	`stateOrProvinceName = State or Province Name (full name)`
	`stateOrProvinceName_default = California`
	
	`localityName = Locality Name (eg, city)`
	`localityName_default = San Jose`
	
	`organizationName = Organization Name (eg, company)`
	`organizationName_default = Cisco Systems, Inc.`
	
	`organizationalUnitName = Organizational Unit Name (eg, section)`
	`organizationalUnitName_default = Cisco Webex`
	
	`commonName = Common Name (eg, YOUR name)`
	`commonName_max = 64`
	`emailAddress = Email Address`
	`emailAddress_default = csg-avops@cisco.com`
	`emailAddress_max = 40`
		On a Linux server, create a configuration file.

Step 2	**openssl req -new -newkey rsa:2048 -nodes -keyout** *key* **-out** *csr* **-config** *configuration file name* **Example:** `openssl req -new -newkey rsa:2048 -nodes -keyout me90sjvce001.webex.com.key -out me90sjvce001.webex.com.csr -config abc`	Generate self signed certificate and csr pair on Linux server. **Note** Use the **crypto key delete all** command to delete all the existing certificates before importing new certificates.

	Command or Action	Purpose
Step 3	**openssl x509 -req -days** <*days*> **-in** <*csr*>**-signkey** <*keys*> **-out** <*certificate*> **Example:** `openssl x509 -req -days 720 -in me90sjvce001.webex.com.csr -signkey me90sjvce001.webex.com.key -out me90sjvce001.webex.com.cer`	Sign the CSR file with your own key.
Step 4	**configure terminal** **Example:** `Hostname# configure terminal`	Log in to CCCSP and enter the configuration mode.
Step 5	**crypto key import trustcacert label** <*label_name*> **terminal** **Example:** `Hostname(config)# crypto key import trustcacert label sample_cert terminal` `% Self-signed CA certificate:` `-----BEGIN CERTIFICATE-----` `MIIDLjCCAhagAwIBAgIBATANBgkqhkiG9w0BAQUFADAwMS4wLAYDVQQDEyVJT1Mt U2VsZi1TaWduZWQtQ2VydGlmaWNhdGUtMzc4ODk3MTYzMB4XDTE3MDExODA2MzYy N1oXDTIwMDEwMTAwMDAwMFowMDEuMCwGA1UEAxMlSU9TLVNlbGYtU2lnbmVkLUNl cnRpZmljYXRlLTM3ODg5NzE2MzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC ggEBAI/k+J1/RdXkUu3aBp8qIMVA7ifpRehG9AXJKlqOafc9Ly92hwNxeLGV/U8k Xlo/fuoyaNyLIu9GwS1BfvM3yHOthhX+T5RHgcj3s1Yctl6HUW93M/EJYluo5RDE NAXJ2UXa/Utl9ZGjCvat8h3N4QduP2ulIsK1IqyYLDRwD1fiSNFrdZB2zzIE1M7g eeitn4n1INHiVtH0jQmO4En/FjUa3YPCFEyB1/U17YGWN/GOHguCsZluL8WywAT5 PqluaipVxWoCzXCb74BSxTJiHs/tmPGkIHi57RvLKxgqr5vHXCOsWsQ6/C9z6My3 tvE6dtLHuP2RgR6r+3xOhKdqcHECAwEAAaNIMEEwDwYDVR0TAQH/BAUwAwEB/zAf BgNVHSMEGDAWgBSIzQOOrJrnxzR8LEQ2VIIfVFpO2DAdBgNVHQ4EFgQUiM0Djqya 58c0fCxENlSCH1RaTtgwDQYJKoZIhvcNAQEFBQADggEBABhYrhWv9DZ0sZZt7Smc o5pgIIFFOtGQYc+ei7H6QNzW5iNSZbSPBAIpmVMQWHVS6cOvJ/N63ayQ+1TN3rZm wmOU9tFExBzjge0nX+Go+0KdWNNQG4XO8SU7BKwM8iWTsM1jT1j6cb9Bv1kMgXW0 5K5AzVYIbaTP/OMoMCsuOJts+GI/Q82H7tlIbdJFbbu3iVEN+gf3coUrHa4X2jLr K3EVLniCLedkcXdy5TppTvQM9j1FzkGMIrWAlFlp/Vh2CTigJy8GZ4pWt5QzjO6m KuP6FZxGPNe8F5BsFCWNM5aHPa8MUqlFKZMuUb50w43SZRT3xfI2WLv1yd49f65T mBA=` `-----END CERTIFICATE-----` `Certificate info` `**********` `Owner: CN=cusp.test.cisco.com,OU=Cisco,O=Example Company,L=San Francisco,ST=Cal ifornia,C=US` `Issuer: CN=cusp.test.cisco.com,OU=Cisco,O=Example Company,L=San Francisco,ST=Ca lifornia,C=US` `Valid from: Wed Jul 16 06:02:54 PDT 2025 until: Sat Jul 14 06:02:54 PDT 2035` `Certificate fingerprint (MD5): 8F:62:94:9C:87:09:5E:B7:91:15:8B:94:63:FD:17:B5`	Import the self-signed CA certificate to CCCSP. Copy the content of the certificate created and paste it when prompted. **Note** Trusted CA validity must be between 5 to 30 years.

	Command or Action	Purpose
	Do you want to continue to import this certificate, additional validation will be perfomed? [y/n]: y	
Step 6	**crypto key import rsa label** *label-name* **pem terminal** **Example:** Hostname# **configure terminal** Hostname(config)# **crypto key import rsa label testtls pem terminal** Enter certificate... End with a blank line or "quit" on a line by itself -----BEGIN CERTIFICATE----- MIIDjTCCAnUCFFg2NWDvvdpXKLwV4hZZ/H9JeAqjMA0GCSqGSIb3DQEBCwUAMIGC MQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwNU2Fu IEZyYW5jaXNjbzEYMBYGA1UECgwPRXhhbXBsZSBDb21wYW55MQ4wDAYDVQQLDAVD aXNjbzEcMBoGA1UEAwwIY3VzcC50ZXN0LmNpc2NvLmNvbTAeFw0yNTA3MTYxMzAy NTRaFw0zNTA3MTQxMzAyNTRaMIGCMQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2Fs aWZvcm5pYTEWMBQGA1UEBwwNU2FuIEZyYW5jaXNjbzEYMBYGA1UECgwPRXhhbXBs ZSBDb21wYW55MQ4wDAYDVQQLDAVDaXNjbzEcMBoGA1UEAwwIY3VzcC50ZXN0LmNp c2NvLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMXIJCYqLZEd ZNwFKmvtxNCUgjLIQ9HBa+ivfqqv2fQPIGNv0+y++Io+PDjrXbUhC53EwZ0OtJ/A TSsuVb1/pXZgoes9eHSK7gA8ZujDi1Oj66EONcmDN9JF1ujUzHm5IUQG5SHntK1H WrGQRWPRSPEJTxrvt2H0Vp/3Iwb/7wKGpd/Gdor9v6c60bALNN8XoNs+u5UnC1eM YP4EzkFiou6zE4pI79vmVgHtEaWpCEMAdIbccToaOU/VmVu0C4pfmDdwmX1Yj2eQ tfWupUEdEbYb0wgvaBTIJ1xeL0Zd+38b8uwPuljjcsSQbPS6lgGMRN+6CNb3IIkS QB2rFUUs3XECAwEAATANBgkqhkiG9w0BAQsFAAOCAQEAYDw0t81LSNpIDJfJDVPU Bvc1qe7g08dT+eoWFN6m8s1IZoBs0y3yk6P75IKWCLOY+wbCC2NII2Gg04Odr7b9 3OhsIsDESa/sqOHIA0V+67sXpV5TimpJqUzcimHhvHSI764SSLU8Wf98oKlUVp7Z OHCFRYnh6brnGyrslIqR5YiNYuy7e1cH6OXAG706D14Ck63qvzF7rOttfeGVTfcf Hv3HzS+8elNtKX0TON7ijAWv3Jlntu74T9ZaqZZv15qe8i36dIM7j6g5bjOej2Ki a5xPEqahkYZhpvXzrRD3swhcB7vfsjdvM9NfSBEjagxO8yZb2GfdJ44wSFcNjf/V Zw== -----END CERTIFICATE----- Enter private key... Private key passphrase? End with a blank line or "quit" on a line by itself -----BEGIN PRIVATE KEY----- MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDFyCQmKi2RHWIc BSpr7cTQlIIyyEPRwWvor36qr9n0DyBjb9PsvviKPjw46121IQudxMGdDrSfwE0r LlW9f6V2YHHrPXh0iu4APGbow4tTo+uhDjXJgzfSRdbolMx5uSFEBuUh57StR1qx kEVj0UjxCU8a77dh9Faf9yMG/+8ChqXfxnaK/b+nOtGwCzTfF6DbPruVJwtXjGD+ BM5BYqIusxOKSO/b5lYB7RGlqQhDAHU23HE6GjlP1ZlbtAuKX5g3cJ19WI9nkIX1 rqVBHRG2G9MIL2gU0ydcXi9GXft/G/LsD7pY43LEkGz0upYBjETfugjW9yCJEkAd qxVFLNlxAgMBAAECggEACoTN81+r47th+Vem/W18WbGESSogfT/hutLWlEJc/59s VA2HAmDXvBZ6Qna2GaODyCTGPVGi2pqmIAVLVxQ6WxTfCilPRjd7V/TebkO0HdMq jXrBfpGlft/rAaALre+cJs+NESQPFmYkmyv2nMvtKrCZje+w25tQnhpZOQ6UihhC CdFIzNDd/Sx6N+HOoE/+NU+ERUb/yZQi0a2OyJm5cDvCOBLUWV35VRBxM5j6yJ+r zu/LWTq3duuwGg6nAzqoAfvs34bIYcLk6cqSIrYHIq0455czdog0EWJpfMG5E077 9kMQTQwO00d0MAZsu5aio08L3BFfT3rToniaRk7ZqQKBgQD/rHYZ9hYLuVHwKV0P BTLHSIoreKcUtgOS/G1xyw/RUljNdSZhYbYfmv5qHs2/8JeFw+/GBjItVC4+LZR/ pNFJHtiMMg3po6LAYHw3jNc0xfPaDUpYDFhU+704IzOb4A7/XZTedxHzqJ2bn4gA kSEf6uvOGDKpbHSzviO4lQW9JQKBgQDGCMOqPqEPoAV/HgQnYn8ykZZL967aye+Z +4p5Wh6Fhco1778wMYZ/jXjJ+HdK/sHGT4Py7BIB8oj78B18HUDOCgUFcFWPp48E HJkIhvGqy6Brf3WHek+EcY9PjimdWe61QVvJQU9biTej3t2yVvErNBEXOaChy6E/ vvS08P9bXQKBgH1vhk28XxqHZDzYflBlM3F0FF25hvdxSRWfBK70eiEvImQ2mori XnDpIPK9hCIsDprRA1z7Azkb+MgGorlxCkC+gm1dzVeYMawCak7yqU6546ZSKOxD uj6Pd0kIdDSAW7tLlVMDxenuCs0kGpx0l3TKU8A8PPbEQAlDOtE/Q56FAoGBAJ0w j78uf0Gbz0EbXki00iodjaM3Mr/P0jPzifLRKbDqhZtSdfuXFB9i+CG461fH55+1 thK4d4K8NSdlbMH7pljlcx7iD8g7RDYLITSZG0sSzZN5vCadD2qJSUWLggP+5RtTw 0B5rjMYnoNaeLHNGt5+HcJly0KIctB2ZQTvun4KxAoGAfoiJThJpklsIL/PbFBJE	Import the self-signed certificate. For each prompt provide the required details.

	Command or Action	Purpose
	wvKI6qJa7UdErubEuSedrarD72C2OOm9GBKgWxeJR3Jdzi6n9nSQoKBLJ/qV+AIV xb8HzNFwHPElsQR1Q2TPWoOx1SRseEtzueAZAvuHhgDXN/cLs5fb4tT88qPGxtKp w1z9CxJT96OQFn6OireQSp8= -----END PRIVATE KEY----- Import succeeded.	
Step 7	**offline** **Example:** Hostname# **offline** !!!WARNING!!!: Putting the system offline will terminate all active calls. Do you wish to continue[n]?: y	Initiates CCCSP offline mode.
Step 8	**reload** **Example:** Hostname(offline)# **reload**	Restarts the CCCSP system and enables CCCSP to verify the imported trusted certificate.

Example

On a Linux server, execute the following commands:

```
Linux-server-test$ vim abc
distinguished_name = req_distinguished_name
[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = US
countryName_min = 2
countryName_max = 2

stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = California

localityName = Locality Name (eg, city)
localityName_default = San Jose

organizationName = Organization Name (eg, company)
organizationName_default = Cisco Systems, Inc.

organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = Cisco Webex

commonName = Common Name (eg, YOUR name)
commonName_max = 64
emailAddress = Email Address
emailAddress_default = csg-avops@cisco.com
emailAddress_max = 40

openssl req -new -newkey rsa:2048 -nodes -keyout me90sjvce001.webex.com.key -out
me90sjvce001.webex.com.csr -config abc
openssl x509 -req -days 720 -in me90sjvce001.webex.com.csr -signkey me90sjvce001.webex.com.key
 -out me90sjvce001.webex.com.cer
```

Configure TLS on Cisco Contact Center SIP Proxy (CCCSP)

After you import the certificates, you must enable TLS connections. If you want more security, you can create a list of trusted peers. If you create such a list, only connections from those peers are accepted. The peer's hostname entry must be the peer's subjectAltName in its certificate. If subjectAltName is not used in the certificate, the peer's hostname entry must be CN.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **sip tls**
4. **sip tls trusted-peer** { *peer's hostname* }
5. **sip tls connection-setup-timeout** { *value in seconds* }
6. **sip tls** {*v1.1* | *v1.2* | *v1.3*}

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP Management mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters CCCSP Configuration mode.
Step 3	**sip tls** **Example:** `Hostname(cusp-config)# sip tls`	Enables the use of SIP TLS connections with other SIP entities, providing secure communication over the Internet.
Step 4	**sip tls trusted-peer** { *peer's hostname* } **Example:** `Hostname(cusp-config)# sip tls trusted-peer example.com`	Creates a list of trusted peers.
Step 5	**sip tls connection-setup-timeout** { *value in seconds* } **Example:** `Hostname(cusp-config)# sip tls connection-setup-timeout <1-60>`	The user specifies the time in SIP Proxy to establish connection with the trusted peer. The default value is 1 second. The range of values is 1 to 60 seconds.
Step 6	**sip tls** {*v1.1* | *v1.2* | *v1.3*} **Example:** `Hostname(cusp-config)# sip tls v1.2`	Enables SIP TLS versions (supports both TLS v1.2, and TLS v1.3). By default, TLS is not enabled for SIP networks. When TLS is enabled for a given network, the connection between the endpoints is established using the highest

	Command or Action	Purpose
		common protocol version negotiated during handshake. For instance, configuring **sip tls** *v1.2* sets "Min TLS 1.2", then the connection is established using TLS v1.2 or TLS v1.3. **Note** Starting from CCCSP-15.0(1) release, CCCSP supports minimum TLS 1.1, 1.2, and 1.3 versions. CCCSP no longer supports TLS v1.0.

Example

Example for TLS configuration:

```
Hostname# cusp
Hostname(cusp)# configure
Hostname(cusp-config)# sip tls
Hostname(cusp-config)# sip tls trusted-peer example.com
Hostname(cusp-config)# sip tls connection-setup-timeout <1-60>
Hostname(cusp-config)# sip tls v1.3
```

Update Web Session with an Imported Signed Certificate

HTTPS is enabled by default on CCCSP for web services. You need not manually generate a crypto key and pass it to the web session security to enable HTTPS. However, you should be able to import a signed certificate that you generated externally, and update the web session with this new key label.

SUMMARY STEPS

1. **configure**
2. **crypto key import rsa label** *label-name* { **der url {sftp: | http: } | pem { terminal | url {sftp: | http:**}} [**default**]
3. **web session security keylabel** *labelname*
4. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `Hostname# configure terminal`	Enters configuration mode.
Step 2	**crypto key import rsa label** *label-name* { **der url {sftp: | http: } | pem { terminal | url {sftp: | http:**}} [**default**]	Imports the signed certificate.

	Command or Action	Purpose
	Example: `Hostname(config)# crypto key import cer label mykey url sftp:`	
Step 3	**web session security keylabel** *labelname* **Example:** `Hostname(config)# web session security keylabel mykey`	Associates a security key for HTTPS.
Step 4	**end** **Example:** `Hostname(config)# end`	Exits to privileged EXEC mode.

Example

Updating web session with an imported signed certificate:

```
Hostname# configure terminal
Hostname(config)# crypto key import cer label mykey url sftp:
Import certificate file...
Address or name of remote host? 192.0.2.2
Source filename? netmod/mycert.cer
952 bytes received.
Import succeeded
Hostname(config)# web session security keylabel mykey
Hostname(config)# end
```

Note HTTPS is enabled by default on CCCSP for web services. You need not manually generate a crypto key and pass it to the web session security to enable HTTPS. The application supports only TLS v1.2 for HTTPS. If you delete the certificate from the web session security and try to log in through HTTP, you will be redirected to HTTPS. Only the latest connection is retained and the remaining connections are logged out.

Note CCCSP supports Min TLS v1.2 (supports both TLS v1.2, and TLS v1.3) for HTTP protocol.

Configure Lite Mode

You can to switch to lite mode to improve the performance of the Cisco Contact Center SIP Proxy (CCCSP) application. In Lite Mode, which requires you to disable record-route, the performance of the application is boosted. In standard mode, the application processes calls up to the licensed limit.

By default, the application is in standard mode.

For information on the performance difference when using Lite Mode versus standard mode, see *Release Notes for Cisco Contact Center SIP Proxy (CCCSP)*.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **lite-mode**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP Management mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters CCCSP Configuration mode.
Step 3	**lite-mode** **Example:** `Hostname(cusp-config)# lite-mode`	The CCCSP application is switched to Lite mode.

Example

The following example shows how to configure to switch to Lite mode:

```
Hostname# cusp
Hostname(cusp)# configure
Hostname(cusp-config)# lite-mode
```

Configure Call Rate Limit

One of the ways you can control the calls on CCCSP is to restrict the number of calls that the CCCSP can handle.

SUMMARY STEPS

1. **cusp**
2. **configure**
3. **call-rate-limit** *limit*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**cusp** **Example:** `Hostname# cusp`	Enters CCCSP Management mode.
Step 2	**configure** **Example:** `Hostname(cusp)# configure`	Enters CCCSP Configuration mode.
Step 3	**call-rate-limit** *limit* **Example:** `Hostname(cusp-config)# call-rate-limit 50`	Sets the maximum call rate that the CCCSP can handle.

Example

The following example limits the number of calls that the system can process to 50:

```
Hostname# cusp

Hostname(cusp)# configure
Hostname(cusp-config)# call-rate-limit 50
```

Commit the Configuration

Now you must commit the configuration. Committing the configuration serves two purposes: the configuration becomes active, and is persisted.

- To see the current active configuration, enter the **show configuration active** command.
- To see what the active configuration will be after you commit your changes, enter the **show configuration candidate** command.
- To commit the configuration for this example, enter the following command:

  `Hostname(cusp-config)# commit`