# SIP TLS Support

## Overview

The Cisco Unified Border Element (CUBE) supports secure SIP calls with Transport Layer Security (TLS). CUBE uses TLS over TCP transport to provide privacy and data integrity of SIP signaling messages it exchanges with remote services. TLS can be configured at the global, tenant and dial peer levels to secure signaling sessions with remote endpoints.

### Feature Information

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

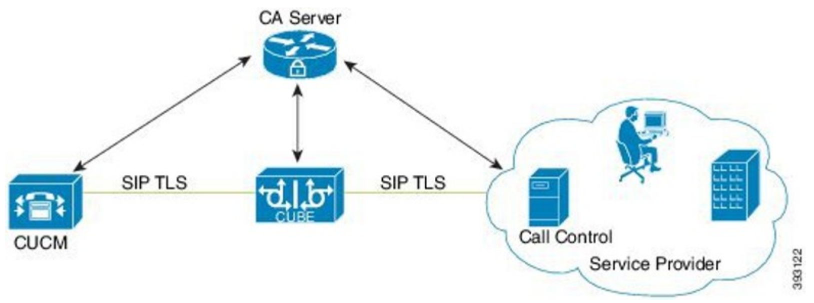
*Table 1: Feature Information*

| Feature Name | Releases | Feature Information |
|---|---|---|
| Client Identity Validation through CN-SAN Fields in a TLS Certificate | Cisco IOS XE Cupertino 17.8.1a | Support introduced for CN-SAN validation of client certificate. The following commands under **voice class tls-profile** *tag* were updated or introduced:<br>• **cn-san validate** {**client** \| **bidirectional** }<br>• **cn-san** *tag san-name* |
| Configurable SIP Trunk Listen Port | Cisco IOS XE Cupertino 17.8.1a | Incoming calls may now be associated with a trunk by destination IP and port number. |
| Trunk Specific TLS Policy | Cisco IOS XE Cupertino 17.8.1a | Trunk specific TLS security trustpoint may now be defined in a tenant configuration. The **voice class tls-cipher** *tag* command was introduced to configure preferred TLS cipher options. |
| Secured SIP with TLS version 1.3 Support | Cisco IOS XE 17.14.1a | Transport Layer Security (TLS) version 1.3 support is introduced to enhance the security of CUBE flows. The supported TLS version 1.3 cipher suites are:<br>• AES128_GCM_SHA256<br>• AES256_GCM_SHA384<br>• CHACHA20_POLY1305_SHA256<br>In addition, support for the minimum TLS version functionality with TLS version 1.2 is added.<br>The following commands were modified: **transport tcp tls, voice class tls-cipher,** and **show sip-ua connection tcp tls details.** |

# Deployment

The following figure illustrates an example of CUBE with SIP TLS connections.

*Figure 1: CUBE with SIP TLS connections*



In a typical deployment, CUBE is placed between an enterprise calling solution such as CUCM, and the PSTN. These devices are authenticated and enrolled with a Certificate Authority (CA) server that issues certificates.

When making a call, a TLS session is created based on mutual trust established through PKI (public key infrastructure), which involves the exchange and validation of certificates signed by a trusted certificate authority (CA). This secure session is then used to send and receive SIP messages, including the sharing of symmetric keys used to encrypt media in associated SRTP streams.

**Note** PKI requires clients to use the correct time. It is recommended that all devices in a solution are synchronized with a common source using NTP.

### TLS Versions

Starting from Cisco IOS XE 17.14.1a, TLS version 1.3 is supported in addition to TLS versions 1.0, 1.1 and 1.2. It is recommended that TLS version 1.2 or 1.3 is used wherever possible to ensure security or compliance.

- The TLS exclusivity functionality allows you to configure a particular version of TLS (1.0 or 1.1 or 1.2 or 1.3). It ensures that only the specified version is enabled for secure communication.
- The default behavior of the CLI command is enabled when none of the versions are specified. In the default form, all the TLS versions 1.1, 1.2, and 1.3 are supported. However, to configure TLS version 1.0, you must explicitly specify the TLS version.
- The minimum TLS version functionality is available only with TLS version 1.2. This minimum configuration enables TLS versions 1.2, and 1.3.

For the list of supported TLS cipher suites, see *TLS Cipher Suites*.

## Peer Verification

When establishing a TLS connection, there are several options for verifying the peer.

### Certificate Verification

Every TLS session is verified by authenticating the certificate provided by the peer during the TLS exchange. Certificates may be self-signed, or signed by a mutually trusted Certificate Authority (CA). Provided that the certificate is found to be valid, the TLS session is established.

The limitation to this approach alone, is that it does not provide any assurance that the session is being established with the intended peer. For example, you may wish to connect with abc.com. In this case, CUBE would resolve the peer's fully qualified domain name using DNS and establish a connection with the resulting IP address. If this DNS resolution were compromised and the session established with a server at xyz.com with its own valid certificate, the session would still be established. To counter this possibility, the certified hostname of the peer should also be verified.

### Hostname Verification

To ensure that a TLS session is established with the intended peer, CUBE can be configured to validate that hostname information provided in the peer's certificate is as expected. Both the Common Name (CN) and Subject Alternative Name (SAN) certificate fields are used to validate the peer.

Prior to Cisco IOS XE Cupertino 17.8.1a, CUBE was able to verify peer identity for outbound connections only, using the **cn-san-validate server** option. Here, the TLS session is established only if the Fully Qualified

Domain Name (FQDN) of the intended destination can be matched with the CN or SAN fields provided in the peer's certificate.

From Cisco IOS XE Cupertino 17.8.1a, CUBE can also verify peer identity for inbound connections. In this case the client provided certificate CN or SAN fields are matched against a list of preconfigured, permitted FQDNs. The **cn-san validate** command has been extended to include **client** and **bidirectional** options to configure inbound and/or outbound verification. The **cn-san** command is also used to configure a permitted list of FQDNs.

When an inbound TLS session is established using CN-SAN verification, the trusted IP address check is bypassed. You don't, therefore, need to add trust list IP addresses for peers that will be verified using CN-SAN.

Also from Cisco IOS XE Cupertino 17.8.1a, it is possible to configure CN-SAN verification and a list of trusted FQDNs in a TLS profile assigned to a tenant configuration. This allows you to craft unique TLS policies for each SIP trunk.

## Remote Application Selection

To realise more efficient use of IP addresses, application service providers direct incoming TLS connections to different applications that share a common address using the TLS Server Name Indication (SNI) header. CUBE supports this concept, allowing the target application to be defined in a TLS policy. By associating a TLS policy that includes target SNI details with a tenant, all calls placed through the trunk are directed efficiently to the destination application.

## TLS Cipher Suites

Cisco IOS XE supports the following TLS cipher suites, which may be configured, in preference order using the **voice class tls cipher** command.

| **Ciphers** | **Descriptions** |
|---|---|
| AES128_GCM_SHA256 | Supported in TLS 1.3 |
| AES256_GCM_SHA384 | Supported in TLS 1.3 |
| CHACHA20_POLY1305_SHA256 | Supported in TLS 1.3 |
| DHE_RSA_AES128_GCM_SHA256 | Supported in TLS 1.2 |
| DHE_RSA_AES256_GCM_SHA384 | Supported in TLS 1.2 |
| DHE_RSA_WITH_AES_128_CBC_SHA | Supported in TLS 1.2 and below versions |
| DHE_RSA_WITH_AES_256_CBC_SHA | Supported in TLS 1.2 and below versions |
| ECDHE_RSA_AES128_GCM_SHA256 | Supported in TLS 1.2 |
| ECDHE_RSA_AES256_GCM_SHA384 | Supported in TLS 1.2 |
| ECDHE_ECDSA_AES128_GCM_SHA256 | Supported in TLS 1.2 |
| ECDHE_ECDSA_AES256_GCM_SHA384 | Supported in TLS 1.2 |
| RSA_WITH_AES_128_CBC_SHA | Supported in TLS 1.2 and below versions |

| Ciphers | Descriptions |
|---|---|
| RSA_WITH_AES_256_CBC_SHA | Supported in TLS 1.2 and below versions |

## Restrictions

- ECDSA ciphers are not supported with TLS version 1.0.
- WebSocket based media forking is not supported with TLS version 1.3.
- TLS handshake fails when CUBE in client mode (with RSA-based Trust point) initiates the TLS connection with Cisco Unified Communications Manager (Call Manager) that is configured with both Rivest, Shamir, Adleman (RSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) certificates.

## Prerequisites

- Cisco 4000 series Integrated Services Routers (ISR4300 and ISR4400) require a security license to use TLS for SIP applications.
- For higher call volumes, you may also require a High Security (HSEC) license for your router.

## Configure SIP TLS

### Step 1: Create a certificate for CUBE to use

#### SUMMARY STEPS

1. **enable**
2. **configure terminal**

#### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |

## Step 1a: Create a private key

### SUMMARY STEPS

1. **crypto key generate rsa{general-keys | usage-keys}** [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *device*]

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **crypto key generate rsa{general-keys \| usage-keys}** [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *device*]<br><br>**Example:**<br><br>`Router(config)# crypto key generate rsa general-keys label kp1 exportable` | **Note** Note: Alternatively to create an elliptic curve key, use **crypto key generate ec keysize{256 \| 384 \| 512}** [**label** *key-label*] [**exportable**]<br><br>Generates RSA key pairs. Arguments and keywords are as follows:<br><br>• **general-keys**—Specifies that the general-purpose key pair should be generated.<br>• **usage-keys**—Specifies that two RSA special-usage key pairs should be generated (that is, one encryption pair and one signature pair) instead of one general-purpose key pair.<br>• **label key-label**—(Optional) Name that is used for an RSA key pair when they are being exported. If a key label is not specified, the fully qualified domain name (FQDN) of the router is used.<br>• **exportable**—(Optional) Specifies that the RSA key pair can be exported to another Cisco device, such as a router.<br>• **modulus modulus-size**—(Optional) IP size of the key modulus in a range from 512 to 4096. If you do not enter the modulus keyword and specify a size, you will be prompted.<br>• **storage device:**—(Optional) Specifies the key storage location. The name of the storage device is followed by a colon (:). |

## Step 1b: Create a trustpoint to hold the certificate

### SUMMARY STEPS

1. **crypto pki trustpoint** *name*
2. **rsakeypair** *key-label* [*key-size* [*encryption-key-size*]]
3. **fqdn** *CUBE FQDN*

4. (Optional) **serial-number** [**none**]
5. (Optional) **ip-address** [*ip-address* |*interface-name* |**none**]
6. **subject-name** *x.500-name*
7. **subject-alt-name** *fqdn*
8. **enrollment** [**mode ra** ][**retry period** *minutes*][**retry count** *number*][**terminal** [**pem**] ] **url** *url*[**pem**]]
9. **revocation-check** *method1*[*method2*[*method3*]]
10. **crl**
11. **password** *string*
12. **exit**

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **crypto pki trustpoint** *name*<br>**Example:**<br>`Router(config)# crypto pki trustpoint CUBE-TrustPoint` | Declares the trustpoint that your router should use. Argument is as follows:<br>• *name*—Creates a name for the trustpoint that you created.<br>• cube1—Represents the trustpoint name that the user specifies. |
| **Step 2** | **rsakeypair** *key-label* [*key-size* [*encryption-key-size*] ]<br>**Example:**<br>`Router(config)# rsakeypair kp1` | **Note** Alternatively use **eckeypair** *key-label* , if you have created an elliptic curve key.<br>Specifies which key pair to associate with the certificate. Arguments are as follows:<br>• *key-label*—Name of the key pair, which is generated during enrollment if it does not exists or if the **auto-enroll regenerate** command is configured.<br>• *key-size*—(Optional) Size of the desired RSA key. If not specified, the existing key size is used.<br>• *encryption-key-size*—(Optional) Size of the second key, which is used to request separate encryption, signature keys, and certificates. |
| **Step 3** | **fqdn** *CUBE FQDN*<br>**Example:**<br>`Router (ca-trustpoint)# fqdn cube.example.com` | Specifies Fully Qualified Domain Name (FQDN). |
| **Step 4** | (Optional) **serial-number** [**none**]<br>**Example:**<br>`Router(ca-trustpoint)# serial-number` | Specifies whether the router serial number should be included in the certificate request. Keyword is as follows:<br>• **none**—(Optional) Specifies that a serial number will not be included in the certificate request. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 5** | (Optional) **ip-address** [*ip-address* \|*interface-name* \|**none**]<br>**Example:**<br>`Router(ca-trustpoint)# ip-address 172.18.197.154 GigabitEthernet2` | Specifies a dotted IP address or an interface that will be included as "unstructuredAddress" in the certificate request. Arguments and keyword are as follows:<br>• ip-address—Specifies a dotted IP address that will be included as "unstructuredAddress" in the certificate request.<br>• interface—Specifies an interface, from which the router can get an IP address, that will be included as "unstructureAddress" in the certificate request.<br>• **none**—Specifies that an IP address is not to be included in the certificate request. |
| **Step 6** | **subject-name** *x.500-name*<br>**Example:**<br>`Router(ca-trustpoint)# subject-name CN=cube.example.com` | Specifies the subject name in the certificate request. Argument is as follows:<br>• **x.500-name**—(Optional) Specifies the subject name that is used in the certificate request. |
| **Step 7** | **subject-alt-name** *fqdn*<br>**Example:**<br>`Router (ca-trustpoint)# subject-alt-name cube.example.com` | Specifies the subject-alternate-name of the FQDN. |
| **Step 8** | **enrollment** [**mode ra** ][**retry period** *minutes*][**retry count** *number*][**terminal** [**pem**] ] **url** *url*[**pem**]]<br>**Example:**<br>`Router (ca-trustpoint)# enrollment url terminal` | Specifies the enrollment parameters of a certificate authority (CA). Arguments and keywords are as follows:<br>• **mode**—(Optional) Registration authority (RA) mode, if your CA system provides an RA. By default, RA mode is disabled.<br>• **retry period minutes**—(Optional) Specifies the period in which the router waits before sending the CA another certificate request. The default is 1 minute between retries. (Specify from 1 through 60 minutes.)<br>• **retry count number**—(Optional) Specifies the number of times a router resends a certificate request when it does not receive a response from the previous request. The default is 10 retries. (Specify from 1 through 100 retries.)<br>• **url url**—URL of the file system where your router should send certificate requests. For enrollment method options, see the enrollment url command.<br>• **terminal**—includes the Privacy Enhanced Mail (PEM) encapsulation boundaries. |

| | Command or Action | Purpose |
|---|---|---|
| | | • **pem**—(Optional) Adds privacy-enhanced mail (PEM) boundaries to the certificate request. |
| **Step 9** | **revocation-check** *method1*[*method2*[*method3*]]<br>**Example:**<br>`Router(ca-trustpoint)# revocation-check none` | This describes one specific behaviour - that is **revocation-check none**. The command more generally defines how the router should check to see if the certificate in the trustpoint has been revoked by the CA.<br>• *method1 [method2 [method3]]*—Method used by the router to check the revocation status of the certificate.<br>Available methods are as follows:<br>• **crl**—Certificate checking is performed by a certificate revocation list (CRL). This is the default behavior.<br>• **none**—Certificate checking is not required.<br>• **ocsp**—Certificate checking is performed by an Online Certificate Status Protocol (OCSP).<br>**Note** If the second and the third methods are specified, each method will be used only if the previous method returns an error, such as the server being down. |
| **Step 10** | **crl**<br>**Example:**<br>`Router(ca-trustpoint)# crl` | |
| **Step 11** | **password** *string*<br>**Example:**<br>`Router(ca-trustpoint)# password password` | (Optional) Specifies the revocation password for the certificate. Argument is as follows:<br>• *string*—Name of the password |
| **Step 12** | **exit**<br>**Example:**<br>`Router# exit` | Exists the current mode. |

## Step 1c: Create a certificate signing request

### SUMMARY STEPS

1. **crypto pki enroll** *trustpoint*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **crypto pki enroll** *trustpoint*<br>**Example:**<br>`Device(config)# crypto pki enroll CUBE-TrustPoint` | The Certificate Signing Request is displayed on the terminal. This should be sent to the Certificate Authority to generate a signed certificate. |

## Step 1d: Authenticate the trustpoint using the signing CA's certificate

### SUMMARY STEPS

1. **crypto pki authenticate** *trustpoint*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **crypto pki authenticate** *trustpoint*<br>**Example:**<br>`crypto pki authenticate CUBE-TrustPoint` | Paste the CA certificate when prompted so that the signed certificate can be authenticated when entered. |

## Step 1e: Import signed certificate

### SUMMARY STEPS

1. **crypto pki import <trustpoint> certificate**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **crypto pki import <trustpoint> certificate**<br>**Example:**<br>`Router(config)# crypto pki import  CUBE-TrustPoint certificate` | Imports the certificate given by the CA using the method configured for the trustpoint. |

## Step 2: Configure preferred TLS cipher options

### SUMMARY STEPS

1. **voice class tls-cipher** *tag*
2. **cipher** *preference cipher-name*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **voice class tls-cipher** *tag*<br>**Example:**<br>`Router(config)# voice class tls-cipher 100` | Creates a cipher list with the provided tag number. |
| **Step 2** | **cipher** *preference cipher-name*<br>**Example:**<br>`Router(config-class)# cipher 1 AES128_GCM_SHA256` | Add up to 13 ciphers in order of preference. |

## Step 3: Configure TLS preferences with a TLS profile

### SUMMARY STEPS

1. **voice class tls-profile** *tag*
2. **cipher** *cipher-list-tag*
3. **cn-san-validate** [**server** | **client** | **bidirectional** ]
4. **trustpoint trustpoint-name**
5. **sni send**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **voice class tls-profile** *tag*<br>**Example:**<br>`Router(config)# voice class tls-profile 100` | Creates a TLS profile with the provided tag number. |
| **Step 2** | **cipher** *cipher-list-tag*<br>**Example:**<br>`Router(config-class)# cipher 100` | The trustpoint label refers to the CUBE's certificate that is generated with the Cisco IOS PKI commands as part of the enrollment process. **cipher** *100* command argument, avoids changes to the configuration if SIP should mandate newer ciphers. The SSL layer in Cisco IOS does not support TLS_RSA_WITH_3DES_EDE_CBC_SHA. Therefore, CUBE actively uses only the TLS_RSA_WITH_AES_128_CBC_SHA suite in strict mode. |
| **Step 3** | **cn-san-validate** [**server** | **client** | **bidirectional** ]<br>**Example:**<br>`Router(config-class)# cn-san validate bidirectonal` | **cn-san-validate** {**server** | **client** | **bidirectional**}—Enables server identity validation through Common Name (CN) and Subject Alternate Name (SAN) fields in the server certificate during client initiated SIP/TLS connections.<br>CUBE will only permit a TLS connection to be established if the domain name configured in the SIP session target is |

| | Command or Action | Purpose |
|---|---|---|
| | | included in either the CN or SAN field of the certificate received from the server, client or both. Once you configure **cn-san-validate**{**server** \| **client** \| **both**}, the specified domain name check is carried out for every new TLS session. |
| **Step 4** | **trustpoint trustpoint-name**<br>**Example:**<br>`Router(config-class)# trustpoint CUBE-TrustPoint` | • **trustpoint** *string*—Refers to the trustpoint for the enrolled certificate. |
| **Step 5** | **sni send**<br>**Example:**<br>`Router(config-class)# sni send` | The **sni send** command enables Server Name Indication (SNI), a TLS extension that allows a TLS client to indicate the name of the server that it is trying connect during the initial TLS handshake process. Only the fully qualified DNS hostname of the server is sent in the client hello. SNI does not support IPv4 and IPv6 addresses in the client hello extension. After receiving a "hello" with the server name from the TLS client, the server uses appropriate certificate in the subsequent TLS handshake process. SNI is supported from TLS 1.2. |

## Step 4: Configure trunk or Tenant for TLS

### SUMMARY STEPS

1. **voice class tenant** *tag*
2. **tls-profile** *tag*
3. **session transport tcp tls**
4. **listen-port secure** *port-number*
5. **url** {*sip* | *sips* | *tel*}
6. **end**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **voice class tenant** *tag*<br>**Example:**<br>`Device(config)# voice class tenant 100` | Enables tenant configuration mode. |
| **Step 2** | **tls-profile** *tag*<br>**Example:**<br>`Device(config-class)# tls-profile 100` | Associates the voice class TLS profile with the tenant. Tag range is 1—10000. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 3** | **session transport tcp tls**<br>**Example:**<br>`Router(voice class)# session transport tcp tls` | Enable the session transport tcp tls. |
| **Step 4** | **listen-port secure** *port-number*<br>**Example:**<br>`Device(config-class)# listen-port secure 5062` | Configures the TLS listen port **secure** *port-number* in voice class tenant. |
| **Step 5** | **url** {*sip*\| *sips* \|*tel*}<br>**Example:**<br>`Router(config-class)# url sips` | Configures URLs to either the SIP, SIPS, or TEL format for your VoIP SIP calls. Keywords are as follows:<br>• **sip**—Generate URLs in SIP format for VoIP calls.<br>• **sips**—Generate URLs in SIPS format for VoIP calls.<br>• **tel** —Generate URLs in TEL format for VoIP calls. This SIP gateway is now configured to use TLS with endpoints sharing the same CA.<br>**Note** This SIP gateway is now configured to use TLS<br>with endpoints sharing the same CA. |
| **Step 6** | **end**<br>**Example:**<br>`Router(config-class)# end` | Ends the current mode. |

## Configure SIP TLS (sip-ua)

### Before you begin

Starting from Cisco IOS XE 17.14.1a, TLS version 1.3 is supported along with the existing TLS versions 1.2, 1.1, and 1.0. In addition, the support for **minimum** keyword configuration with TLS version 1.2 is introduced.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **sip-ua**
4. **transport {tcp [tls [v1.0 | v1.1 | v1.2 [minimum] | v1.3 ]] | udp}**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **sip-ua**<br>**Example:**<br>`Device(config-voi-serv)# sip-ua` | Enters voice service SIP user-agent configuration mode. |
| **Step 4** | **transport {tcp [tls [v1.0 \| v1.1 \| v1.2 [minimum] \| v1.3 ]] \| udp}**<br>**Example:**<br>`Device(config-sip-ua)# transport tcp tls` | Configures the specified TLS version. Enables SIP user agent in TLS over TCP mode. The default configuration supports all TLS versions with fallback.<br>• Default configuration:<br>Configured without specifying any specific TLS version. The TLS versions 1.1, 1.2, and 1.3 are supported except version 1.0.<br>`Device(config-sip-ua)# transport tcp tls`<br>• Exclusive version:<br>Only the configured TLS version is enabled.<br>`Device(config-sip-ua)# transport tcp tls v1.3`<br>• Minimum TLS version:<br>Only the configured TLS version ciphers or above version ciphers are supported. Fallback to previous version is not allowed.<br>**Note** Supports the **minimum** keyword configuration only with TLS version 1.2.<br>`Device(config-sip-ua)# transport tcp tls v1.2 minimum` |

# Verify SIP TLS Configuration

After a call is made, the following commands may be used to verify details of the TLS connection.

- **show sip-ua connections tcp tls brief**
- **show sip-ua connections tcp tls detail**
- The brief command displays the associated tenant (trunk) and listen port only.

### Example brief Output

```
Router#show sip-ua connections tcp tls brief
Total active connections      : 0
No. of send failures          : 0
No. of remote closures        : 0
No. of conn. failures         : 0
No. of inactive conn. ageouts : 0
TLS client handshake failures : 0
TLS server handshake failures : 0

-------------- SIP Transport Layer Listen Sockets ---------------
  Conn-Id             Local-Address            Tenant
 ==========    ==========================   ============
  0             [0.0.0.0]:5061:                0
  3             8.43.21.8:8888:                200
  4             8.43.21.8:6061:                400
  5             10.64.100.145:5091:vrf         44

Router#
```

**Note** The RSA or ECDSA key types in the detailed output are displayed only with TLS version 1.3.

The following is a sample output for the **show sip-ua connections tcp tls detail** command that displays RSA key type along with TLS version 1.3 ciphers:

```
Router#show sip-ua connections tcp tls detail
Total active connections      : 2
No. of send failures          : 0
No. of remote closures        : 0
No. of conn. failures         : 0
No. of inactive conn. ageouts : 0
Max. tls send msg queue size of 1, recorded for 10.64.100.152:5061
TLS client handshake failures : 0
TLS server handshake failures : 0

---------Printing Detailed Connection Report---------
Note:
 ** Tuples with no matching socket entry
    - Do 'clear sip <tcp[tls]/udp> conn t ipv4:<addr>:<port>'
      to overcome this error condition
 ++ Tuples with mismatched address/port entry
    - Do 'clear sip <tcp[tls]/udp> conn t ipv4:<addr>:<port> id <connid>'
      to overcome this error condition
 * Connections with SIP OAuth ports
```

```
Remote-Agent:10.64.100.150, Connections-Count:1
  Remote-Port Conn-Id Conn-State  WriteQ-Size            Local-Address
TLS-Version                Cipher              Curve Tenant
  =========== ======= =========== =========== =====================================
=========== ================================= ===== ======
        22943       7 Established           0 10.64.100.151:5061
TLSv1.3         TLS_AES_256_GCM_SHA384:RSA P-521     0

Remote-Agent:10.64.100.152, Connections-Count:1
  Remote-Port Conn-Id Conn-State  WriteQ-Size            Local-Address
TLS-Version                Cipher              Curve Tenant
  =========== ======= =========== =========== =====================================
=========== ================================= ===== ======
         5061       8 Established           0 10.64.100.151:47687
TLSv1.3         TLS_AES_256_GCM_SHA384:RSA P-521     0

-------------- SIP Transport Layer Listen Sockets ---------------
  Conn-Id             Local-Address                      Tenant
 ==========    ==========================              ========
  0             [0.0.0.0]:5061:                             0
  6             [10.64.100.151]:5061:                       0
```

The following is a sample output for the **show sip-ua connections tcp tls detail** command that displays ECDSA key type along with TLS version 1.3 ciphers:

```
Router#show sip-ua connections tcp tls detail
Total active connections      : 2
No. of send failures          : 0
No. of remote closures        : 0
No. of conn. failures         : 0
No. of inactive conn. ageouts : 0
Max. tls send msg queue size of 1, recorded for 10.1.10.50:5061
TLS client handshake failures : 0
TLS server handshake failures : 0

---------Printing Detailed Connection Report---------
Note:
 ** Tuples with no matching socket entry
    - Do 'clear sip <tcp[tls]/udp> conn t ipv4:<addr>:<port>'
      to overcome this error condition
 ++ Tuples with mismatched address/port entry
    - Do 'clear sip <tcp[tls]/udp> conn t ipv4:<addr>:<port> id <connid>'
      to overcome this error condition
 * Connections with SIP OAuth ports

Remote-Agent:10.1.10.50, Connections-Count:2
  Remote-Port Conn-Id Conn-State  WriteQ-Size            Local-Address
TLS-Version               Cipher               Curve Tenant
  =========== ======= =========== =========== =====================================
=========== ================================= ===== ======
         5061       9 Established           0 10.1.20.155:37081
TLSv1.3  ECDHE-RSA-AES256-GCM-SHA384:ECDSA P-521     0
        41635       8 Established           0 10.1.20.155:5061
TLSv1.3       TLS_AES_256_GCM_SHA384:ECDSA P-256     0

  Remote-Port Conn-Id Conn-State  WriteQ-Size            Local-Address
TLS-Version Cipher                         Curve Tenant
  =========== ======= =========== =========== =====================================
=========== ============================= ===== ======
        53516     102 Established           0 10.64.100.150:5061
TLSv1.2    ECDHE-RSA-AES256-GCM-SHA384 P-521     0

-------------- SIP Transport Layer Listen Sockets ---------------
  Conn-Id             Local-Address                      Tenant
 ==========    ===========================            ========
  0             [0.0.0.0]:5061:                            0
  1             [::]:5061:                                 0
  6             [10.1.20.155]:5061:                        0
  7             [2001:10:1:20::135]:5061:                  0
```

Alternatively, the debug ccsip messages command can be used to verify the "Via:" header for TLS is included. This output is a sample INVITE request of a call that uses SIP TLS and the "sips:" URI scheme:

```
INVITE sips:777@172.18.203.181 SIP/2.0
Via: SIP/2.0/TLS 172.18.201.173:5060;branch=z9hG4bK2C419
From: <sips:333@172.18.201.173>;tag=581BB98-1663
To: <sips:5555555@172.18.197.154>
Date: Wed, 28 Dec 2005 18:31:38 GMT
Call-ID: EB5B1948-770611DA-804F9736-BFA4AC35@172.18.201.173
Remote-Party-ID: "Bob" <sips:+14085559999@1.2.3.4>
Contact: <sips:123@host>
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, COMET, REFER, SUBSCRIBE, NOTIFY, INFO
Max-Forwards: 70
Cseq: 104 INVITE
Expires: 60
Timestamp: 730947404
Content-Length: 298
Content-Type: application/sdp

v=0
o=CiscoSystemsSIP-GW-UserAgent 8437 1929 IN IP4 172.18.201.173
s=SIP Call
c=IN IP4 1.1.1.1
t=0 0
m=audio 18378 RTP/AVP 0 19
c=IN IP4 1.1.1.1
a=rtpmap:0 PCMU/8000
a=rtpmap:19 CN/8000
a=ptime:20
```

**clear sip-ua tcp tls connection**

Clear command to close a SIP TLS connection.

```
Router#clear sip-ua tcp tls connection ?
  id      Connection ID for the connection that needs to be closed in the sip tcp/udp process

  target  Target address for the connection that needs to be closed in the transport layer

Router#clear sip-ua tcp tls connection id ?
  <1-16380>  Value of the Connection ID that needs to be closed
```

## Example: SIP TLS Configuration

The following example provides excerpts from a typical running configuration.

```
show running-config  brief
ip domain name example.com
!
crypto pki trustpoint CUBE-TrustPoint
 enrollment terminal
 fqdn cube.example.com
 subject-name cn=cube.example.com
 subject-alt-name cube.example.com
 revocation-check crl
 rsakeypair cube kp1
!
crypto pki certificate chain CUBE-TrustPoint
 certificate 07
!
voice class tenant 100
 tls-profile 100
 listen-port secure 5080
 sip-server dns:sip.acme.co:5080
 srtp-crypto 100
 session transport tcp tls
 url sips
 bind control source-interface GigabitEthernet0/0/0
 bind media source-interface GigabitEthernet0/0/0
!
voice class srtp-crypto 100
 crypto 1 AEAD_AES_256_GCM
 crypto 2 AES_CM_128_HMAC_SHA1_80
!
voice class tls-cipher 100
 cipher 1 DHE_RSA_AES128_GCM_SHA256
!
voice class tls-profile 100
 cipher 100
 cn-san validate bidirectional
 cn-san 1 sip.acme.co
 trustpoint CUBE-TrustPoint
!
dial-peer voice 100 voip
 description Towards sip.acme.co
 destination-pattern 1000
 session protocol sipv2
 session target sip-server
 voice-class codec 1
 voice-class sip tenant 100
 dtmf-relay rtp-nte
 no vad
!
```

## Syslog Messages

- From Cisco IOS XE Cupertino 17.8.1a, when CUBE fails to validate the peer certificate identity check through CN-SAN validation, a syslog message is generated in the following format:

```
Sep 26 07:15:57.949: %SIP-2-TLS_HANDSHAKE_FAILED :Peer certificate identity
check failed - remote_addr=198.51.100.254, remote_port=36233,
local_addr=203.0.113.254, local_port=3000, vrf=, tenant=1
```

- When there is a failure to open listen-socket for the associated tenant, a syslog message is generated in the following format:

```
Sep 24 05:32:27.838: %SIP-2-LISTEN_SOCKET: Failed to open listen socket for
ip_addr=198.51.100.255, port=8888, vrf=, transport=TLS, tenant=200
```

- From Cisco IOS XE 17.14.1a, when TLS handshake fails due to certificate mismatch, a syslog message is generated in the following format:

```
Dec 15 08:59:17.625: %SIP-2-TLS_HANDSHAKE_FAILED: Reason: certificate verify
 failed, Connection ID: 17, local address : [2001:10:2:10::10]:24399 and remote
 address : [2001:10:1:10::50]:5061
```

- From Cisco IOS XE 17.14.1a, when TLS handshake fails due to cipher mismatch, a syslog message is generated in the following format:

```
Dec 13 14:56:42.556: %SIP-2-TLS_HANDSHAKE_FAILED: Reason: no shared cipher,
Connection ID: 12, local address : 10.04.200.100:5061 and remote address :
10.04.200.101:2009
```

- From Cisco IOS XE 17.14.1a, when TLS handshake fails due to version mismatch, a syslog message is generated in the following format:

```
Dec 8 04:02:47.096: %SIP-2-TLS_HANDSHAKE_FAILED: TLS handshake failure, Reason:
 tlsv1 alert protocol version, Connection ID: 8, local address :
10.04.200.101:5331 and remote address : 10.04.200.102:5061
```