Genesys Security Deployment Guide

Securing Connections Using TLS

5/6/2025

Contents

Securing Connections Using TLS

This section describes securing only those connections that are defined using objects in Genesys Configuration Server, using both simple and mutual TLS. Those are typically connections between Genesys applications that utilize proprietary protocols, such as connections to Message Server, StatServer, and T-Server. Some connections have additional considerations discussed later.

Where to Set TLS Properties

> **Important**
> Configuration section names, configuration option names, and predefined option values are case-sensitive. Type them in Genesys Administrator exactly as they are documented in this Guide.

> **Tip**
> You must configure all required settings at each level (or leave that level empty). Do not configure, for example, **certificate** and **certificate-key**, or **certificate** and **sec-protocol**, on different levels. If you have to configure TLS at a particular level, add all options that have no default values and/or are otherwise needed for this TLS connection to work properly.

Server Listening Ports

You must specify a port as secure by setting its port properties. The Transport parameters of a secure port contains **tls=1**, and optionally **tls-mutual=1**. You will typically use Genesys Administrator to enable secure mode on the server port. This is a mandatory step to make the server port available for TLS, and must be set only on this (port) level. You can configure additional parameters at the Port level, or you can do that at higher levels (Application or Host levels).

For secure listening ports on the server to work properly, each must be provisioned with **certificate**, and for native applications on UNIX with **certificate-key** and **trusted-ca**. Additional options can also be specified, as required.

> **Tip**
> Not all listening ports of an application are configured using Port configuration objects.

> Some ports (such as a SIP Server SIP port) are configured differently. Refer to descriptions of individual connections later in this guide and to specific product-specific documentation to confirm how to configure server ports for each particular case.

Client Connections

When you link your application object with a server object in your configuration, and specify the secure port as the target for the connection, then and only then is this connection considered secure. Additional configuration might be needed for this application to operate properly. You will typically use Genesys Administrator to define the secure connection by establishing this link.

For client connections to operate in native applications on UNIX you must configure **trusted-ca**. If the application also supports mutual TLS, then **certificate** and, for native UNIX applications, **certificate-key**, is mandatory. Specify these and other parameters at connection level, or you can use upper levels to do so.

> ### Tip
> Not all client connections of an application are configured using objects. Refer to descriptions of individual connections (such as secure SIP Server connections to a remote trunk) later in this guide, and to product-specific documentation for more information about configuring client connections in each particular case.

Application Objects

Parameters for all TLS connections and listening ports can be specified for each application object (*Application-level*). For native UNIX applications that are clients of other secure servers, you must configure **trusted-ca**. For applications that have server ports, you must also configure **certificate** and, for native UNIX applications, **certificate-key**. Additional options might also be specified, as necessary, and will apply to all connections and ports not yet provisioned with their own parameters.

Some applications will need to be restarted for changes made on this level to take effect. In addition, connections and ports of applications that do not use the relevant configuration objects might require that you specify additional parameters. Refer to each type of connection for more details.

Host Objects

Parameters for TLS connections and listening ports for all applications running on a particular machine can be specified on the host object (*Host-level*) if no such parameters are already set on the Application or Connection/Port level. If any of the host's applications act as a server (that is, a listening port is in secure mode), then **certificate**, and, for native UNIX applications, **certificate-key** and **trusted-ca** are required. If only client applications (that is, applications that open client connections) are deployed on the host, and if mutual TLS connections are supported , **certificate** (and **certificate-key** and **trusted-ca** for native UNIX applications) are required. **trusted-ca** is mandatory for hosts on which native UNIX applications are running. Additional options can also be specified, as necessary, and apply to all connections/ports of all applications, where the options are

supported but the applications are not yet provisioned with their own parameters at lower levels.

Applications must be restarted to pick up changes made in the configuration at the host level, unless specified otherwise for each individual application. Connections and Ports of applications that do not use relevant configuration objects typically do not support configuring parameters on host level; refer to each type of connection for more details.

Using Genesys Administrator for Configuring Secure Parameters

In Genesys Administrator, configure settings in the **Network Security** section of the **Configuration** tab, noting the following:

- This is done most easily at the Host level, but there are mixed cases because components are often both clients and servers. When set at the Host level, you must specify that Host settings at to be used at the Application level.
- Settings made at the Connection/Port level are the most precise, but can be a lot of work in a large configuration.
- Because of different implementations (for example, native applications vs. PSDK Java-based), you might have to configure settings for specific cases at lower levels, even if default settings are done at a higher level.

Certificates for Windows Native Applications and Applications with .NET dependency

For Windows native applications, and applications that depend on the .NET framework, Genesys implementation of secure connections rely on the presence of certificates (an individual host certificate and a trusted certificate authority certificate) stored in Windows Certificate Storage on every host where Genesys applications are installed. With these certificates properly installed and managed (as described in TLS Certificates), the only configuration that is mandatory on the Genesys side for applications that are configured to open secure listening ports is the **certificate** option set to the Host certificate thumbprint, on the server at the Port, Application, or Host level. There is no mandatory provisioning of certificate-related options for an application to establish secure client connections when working in such environment. An application that does not use Port/Connection configuration objects might have different requirements.

Enabling a Secure Port on a Server

On Linux

1. In the **Listening Ports** field of the **Configuration** tab of the server Application object, select the port to be configured as secured, and click edit. The **Port Info** window opens.
2. On the **General** tab, choose Secured in the **Select Listening Mode** field. This automatically enters `tls=1` in the **Transport Parameters** field of the **Advanced** tab.
3. If you are setting up Mutual TLS, also add `tls-mutual=1` to the **Transport Parameters** field. All

parameters in this field must be separated by semi-colons (`;`).

4. Configure the secure port parameters at the appropriate level, as follows:
   - **Host level**:
     a. In the Host object on which the server is running, in the **Network Security** section of the **Configuration** tab, enter the absolute paths to the certificate, certificate key, and Trusted CA in the corresponding fields.
     b. In the server Application object, in the **Network Security** section of the **Configuration** tab, select *Host* in the **Certificate Source** field.
     c. Restart the server.
5. **Application level**: In the server Application object, in the **Network Security** section of the **Configuration** tab, do the following:
   a. Select *Application* in the **Certificate Source** field.
   b. Enter the absolute paths to the certificate, certificate key, and Trusted CA in the corresponding fields.
6. **Port level:** In the **Network Security** tab of the **Port Info** window, enter the absolute paths to the certificate, certificate key, and Trusted CA in the corresponding fields.

On Windows

1. Import the server Host certificate and the Trusted CA certificate into Windows Certificate Storage.
2. In the **Listening Ports** field of the **Configuration** tab of the Message Server Application object, select the port to be configured as secured, and click edit. The **Port Info** window opens.
3. On the **General** tab, choose **Secured** in the **Select Listening Mode** field. This automatically enters `tls=1` in the **Transport Parameters** field of the **Advanced** tab.
4. If you are setting up Mutual TLS, also add `tls-mutual=1` to the **Transport Parameters** field. All parameters in this field must be separated by semi-colons (`;`).
5. Configure the secure port parameters at the appropriate level, as follows:
   - **Host level:** In the Host object on which Message Server is running, in the **Network Security** section of the **Configuration** tab :
     a. Enter the thumbprint of the certificate in the **Certificate** field.
     b. In the server Application object, in the **Network Security** section of the **Configuration** tab, select *Host* in the `Certificate Source` field.
     c. Restart the server.
   - **Application level:** In the Message Server Application object, in the **Network Security** section of the **Configuration** tab:
     - Select *Application* in the `Certificate Source` field.
     - Enter the thumbprint of the certificate in the **Certificate** field.
   - At the **Port level**: On the **Network Security** tab of the **Port Info** window, enter the thumbprint of the certificate in the **Certificate** field.

Configuring Client side of Secure Connections

On Linux

1. In the **Connections** field of the **Configuration** tab of the Client Application object, select the connection to the server. The **Connections Info** window opens.
2. On the **General** tab, in the **ID** field, select the ID/number of the secured port on the server from the drop-down list.
3. Configure the parameters of the secure connection at the appropriate level, as follows:
   - **Host level**: In the Host object on which client is running, in the **Network Security** section of the **Configuration** tab, do the following:
     a. Enter the absolute path to the Trusted CA in the corresponding field.
     b. If you are configuring Mutual TLS, enter the absolute path to the certificate in the corresponding field.
     c. In the Client Application object, in the **Network Security** section of the **Configuration** tab, select `Host` in the `Certificate Source` field.
   - **Application level**: In the Server Application object, in the **Network Security** section of the **Configuration** tab, do the following:
     a. Select `Application` in the `Certificate Source` field.
     b. Enter the absolute path to the Trusted CA in the corresponding field.
     c. If you are configuring Mutual TLS, enter the absolute path to the certificate in the corresponding field.
   - **Connection level:** In the **Network Security** tab of the **Connection Info** window, do the following:
     a. Enter the absolute path to the Trusted CA in the corresponding field.
     b. If you are configuring Mutual TLS, enter the absolute path to the certificate in the corresponding field.

On Windows

1. Import the trusted CA certificate and, if using Mutual TLS, the client host certificate into Windows certificate storage.
2. In the **Connections** field of the **Configuration** tab of the Client Application object, select the connection to the server. The **Connections Info** window opens.
3. On the **General** tab, in the **ID** field, select the ID or number of the secured port on the server from the drop-down list.
4. If you are setting up Mutual TLS, configure the parameters of the secure connection at the appropriate level, as follows:
   - **Host level:** In the Host object on which the client Application is running, in the **Network Security**

section of the **Configuration** tab, do the following:

a. Enter the thumbprint of the certificate, imported in step 1, in the **Certificate** field.

b. In the client Application object, in the **Network Security** section of the **Configuration** tab, select `Host` in the `Certificate Source` field.

- **Application level:** In the client Application object, in the **Network Security** section of the **Configuration** tab, do the following:

  a. Select `Application` in the `Certificate Source` field.

  b. Enter the thumbprint of the certificate, imported in step 1, in the **Certificate** field.

- **Connection level:** On the **Network Security** tab of the **Connection Info** window, do the following :

  a. Enter the thumbprint of the certificate, imported in step 1, in the **Certificate** field.

Configuring Certificate Chains

Starting with release 8.1.3, Genesys Security Pack on UNIX supports security certificate chains, sending out the intermediate certificates along with the root certificate. The certificate container PEM format used to load certificates allows storing multiple certificates in a single PEM file.

However, note that the OpenSSL Security Pack requires the full path of local certificate chains to be specified in the configuration, starting with the root CA, even if some of the intermediate certificates are explicitly listed as trusted in the **CA** field.

PEM file Containing Multiple Certificates
A PEM file can contain multiple certificates, listed in order. Certificate info is stored in a **.pem** file, starting with `-----BEGIN CERTIFICATE-----` and ending with `-----END CERTIFICATE-----`. A PEM file can contain multiple certificates, each starting and ending with these tags.

To generate a multi-certificate PEM file from many simple PEM files, all source PEM files must be concatenated into the multi-certificate PEM file, for example:

```
cat cert_1.pem cert_2.pem cert_3.pem > cert_result.pem
```

Certificates should be listed in order from the end entity certificate, then intermediate certificates, up to the root CA certificate. Security Pack attempts to form a correct certification chain from the provided certificates by rearranging them, but it is best to provide the certificate chain already arranged.

Supplying Multiple Certificate PEM Files
The Security Pack enables configuration of multiple certificate storage PEM files that are to be loaded when forming its security credentials. PEM files must be listed in a comma-separated list. Each of the provided files can contain one or many PEM certificates. All the provided certificates will be loaded by Security Pack. The general guideline—to keep the certificate chain in order—still applies.

To enable this support, specify the multiple certificates in a comma-delimited list in the **Certificate**

field when configuring TLS. The certificates are sent in the order in which they are specified.

Configuring Certificate Revocation Lists (CRLs)

CRL on Windows

The Microsoft SChannel security provider retrieves Certificate Revocation List (CRL) information for the certificate being verified using a CRL distribution point (CDP) mechanism. If the CDP URL specified in the certificate is not reachable from the current host (or is blocked by a firewall or other network policy), the process of certificate verification might pause for a time interval specified in system settings (default is 15 seconds). This might be because the CDP URL is accessible but resides on a slow network resource, or because connection quality is very low, resulting in significant delays when retrieving the CRL. This in turn might lead to various undesired consequences. To avoid these problems, Genesys strongly recommends that you use a local CDP in certificates and make sure all CDPs are accessible without any significant delay. You can also turn off CRL verification using Windows tools; refer to the documentation for your version of Windows.

CRL with Security Pack on Unix

Set the **crl** configuration option (in the **[security]** section) at the same level (host, application, or connection) as the certificates it will contain, to allow the supporting Genesys component to verify certificates against a CRL. Specify the name of a .PEM file that contains one or more certificates defining the Certificate Revocation List. Refer to the *Framework Configuration Options Reference Manual* for a full description of this option.

Configuring Multiple Trusted CAs

Starting with release 8.0.0, Genesys Security Pack on UNIX supports multiple Trusted CA certificates for TLS connections. To enable this support, you must create a PEM file listing all of the certificates issued by the Trusted CAs, as follows.

Multiple Trusted CA PEM file

A multiple trusted CA PEM file is a file that contains information about two or more certificates, like this:

```
----BEGIN CERTIFICATE----
<encoded certificate-1 data>
----END CERTIFICATE----
----BEGIN CERTIFICATE----
<encoded certificate-2 data>
----END CERTIFICATE----
----BEGIN CERTIFICATE----
<encoded certificate-3 data>
----END CERTIFICATE----
```

To view the contents of a certificate PEM file, use the **openssl** utility, as follows:

```
openssl x509 -text -noout -in TrustedCA.pem
```

To create a multiple trusted CA certificate file, concatenate the individual certificate files, like this:

```
cat TrustedCA1.pem TrustedCA2.pem > multipleCAs.pem
```

Specify the full path to the multiple trusted CA certificate file in the **trusted-ca** field when configuring TLS. As security circumstances and requirements change, you can modify the file by adding and removing certificates, or completely replace it by specifying a single Trusted CA. Refer to the *Framework Configuration Options Reference Manual* for a full description of the **trusted-ca** option.

Check Certificate Host Matching

The **tls-target-name-check** option, set in the **[security]** section, enables a case-insensitive comparison of the TLS host name and the certificate's subject field during the authentication process. This option is transferred to a third-party library and describes whether it is necessary or not to check the names. To ensure a correct match, the client must be connecting to the server in exactly the same manner as stated in the certificate. The client must use the same FQDN, not an IP address.

Important

- Configure this option only on the client's side, at the same level where the certificate is configured.
- Security Pack 8.1 and earlier supported only a case-sensitive check of host names.

Refer to Validation (Authentication) by Receivers of Certificates for details on authentication of TLS-Server and TLS-Client identity, which includes a step to check for certificate-host matching.

If **tls-target-name-check** is set to `Host`, Genesys Security Pack uses the certificate SAN (if present, otherwise, the subject CN). Genesys Security Pack does not accept any wildcard symbols in the certificate SAN (or CN) field.

If the supporting Genesys component has a TLS-Client role for outbound connection and **tls-target-name-check=no**, then comparison of TLS-Server host name and the certificate's subject field is not made. This is used in cases when some phone devices or programs have the certificate without the host name in subject field, but have a MAC-address or other information.

By default, a comparison is not made, and the connection is allowed. Refer to the *Framework Configuration Options Reference Manual* for a full description of the **tls-target-name-check** option.

Sample Basic Configurations

This section contains examples of TLS configurations, both simple and mutual TLS.

Simple TLS on UNIX

Setting	Server Side	Client Side
Port set to:	**Listening Mode =** Secured	
Application to use:	Host TLS settings	Host TLS settings

Host set with:	`[security]`
	`trusted-ca=/etc/sec/ca/certauth.pem`
	`certificate=/etc/sec/certs/hostcert.pem`
	`certificate-key=/etc/sec/certs/privkey.pem`
		`[security]`
		`trusted-ca=/etc/sec/ca/certauth.pem`

Simple TLS on Windows

Setting	Server Side	Client Side
Port set to:	**Listening Mode =** Secured	
Application to use:	Host TLS settings	
Host set with:	`[security]` `certificate=89 A0 C1 D4 67 01 93 5D …`	

Simple TLS on Mixed Operating Systems

Setting	Server Side (*nix)	Client Side (Windows)
Port set to:	**Listening Mode =** Secured	
Application to use:	Host TLS settings	Host TLS settings

Host set with:	`[security]`
	`trusted-ca=/etc/sec/ca/certauth.pem`
	`certificate=/etc/sec/certs/hostcert.pem`
	`certificate-key=/etc/sec/certs/privkey.pem`	

Mutual TLS on UNIX

Setting	Server Side	Client Side
Port set to:	**Listening Mode =** Secured Advanced Transport Parameters for port	

	include **tls-mutual=1**	
Application to use:	Host TLS settings	Host TLS settings

Host set with:	`[security]`
	
	`certificate=/etc/sec/certs/servhostcert.pem`
	`certificate-key=/etc/sec/certs/servprivkey.pem`
	`trusted-ca=/etc/sec/ca/certauth.pem`
	`tls-mutual=1`
		`[security]`
		
		`certificate=/etc/sec/certs/clthostcert.pem`
		`certificate-key=/etc/sec/certs/cltprivkey.pem`
		`trusted-ca=/etc/sec/ca/certauth.pem`
		`tls-crl=/etc/sec/crl/crllist.pem`