Genesys Security Deployment Guide

TLS Implementations in Genesys

12/19/2025

Contents

TLS Implementations in Genesys

TLS is a protocol with an agreed-upon standard definition. To utilize TLS in real applications, the protocol must be implemented in source code. Many different TLS implementations exist, some developing and patching newly found security issues, and some of them not. Refer to Comparison of TLS implementations to see what TLS implementations exist and how they differ.

All TLS implementations differ in the list of supported features, aspects, protocol versions and cipher lists. However, all TLS implementations still implement the same TLS, and therefore should be able to communicate with each other, given that a compatible set of features is requested at each end of the connection.

Even Genesys implementations vary. For example:

- Genesys uses different technologies (C++, Java, and so on)
- Genesys operates on different infrastructures (Windows, Unix)

Summary

Genesys products use the following TLS implementations for their proprietary protocols, depending on the component platform.

Environment	Platform	TLS Implementation Used	Keystore
Genesys native applications (with no dependency on .NET or Java)	Microsoft Windows	Microsoft SChannel (built into the operating system)	Windows Certificate Services
	*nix (Linux, AIX, Solaris)	OpenSSL (was RSA Bsafe)	File system
Genesys applications with dependency on .NET	Microsoft Windows	Microsoft SChannel (built into the operating system)	Windows Certificate Services
Genesys applications with dependency on Java	All	Java Secure Socket Extensions from Oracle JRE with configured Provider	File system; JKS; or Windows Certificate Services (on Windows only)
Selected applications (such as Genesys Voice Gateway and so on) or individual connections of a native application (such as GVP Media or Configuration LDAP)	Microsoft Windows, *nix (Linux, AIX, Solaris)	Built-in OpenSSL (review the documentation for each application)	File system (review the documentation for each application)
HTTPs (web interfaces)	Typically based on Application Server, such as J2EE, Microsoft IIS		

All Genesys TLS implementations are compatible and can communicate with each other. However, configuration details for components using different TLS implementations differ significantly.

Genesys components may also utilize open standard application protocols, such as LDAP, which, if secured, may have third-party implementations that are different (based on different versions of OpenSSL for example) than the standard implementation of Genesys TLS on a selected platform. More details can be found in sections dedicated to each specific application protocol.

Standard TLS Implementation for an Application with no .NET or Java Dependencies

Native Genesys components use an internal Genesys common library to facilitate network communication using proprietary Genesys protocols, and selected open standard protocols, like HTTP. The Genesys common library encapsulates the actual underlying TLS implementation from the component code, allowing same applications to run on different platforms while using the same API. The Genesys common library is used only by native Genesys components, not components that are written in managed code such as Java and .NET. Any exceptions to this are noted in the documentation for individual applications.

As shown in the table above, Genesys utilizes different TLS implementations to facilitate secure connections, depending on the underlying operating system.

Genesys Native Applications on Windows

When running on Windows, the Genesys common library uses Microsoft SChannel TLS implementation, technically a part of the host Windows operating system itself. All TLS operations are delegated to the operating system (SChannel) level, and all configuration is passed to the operating system level. Genesys components have very little control over TLS operations.

Because of the built-in nature of Windows' SChannel module, no specific installation of the Genesys Security Pack is required.

TLS certificates (including private keys) and CA certificates (trusted or not) are stored in Windows certificate storage. Refer to Managing Certificates using MMC on Windows for details about accessing and managing the certificate storage. Two types of certificate storage are available: user and system level. The Genesys common library first looks in user-level storage for the configured certificate, then in system-level storage.

Windows implementations do not allow any wildcard symbols to be used in the certificate Subject Alternate Name (SAN) or Common Name (CN).

In mutual TLS mode, whenever a server requests a client's certificate, it presents a list of server trusted CAs from which the client selects a certificate to present. Regardless of the client certificate configuration, Windows will lookup a certificate issued by one of the CAs provided and send it to the server.

> **Important**
>
> To avoid confusion and the presentation of wrong certificates, Genesys strongly recommends that you import only certificates intended for actual usage.

Available protocol versions and cipher lists may differ dependent on the version of the Windows operating system, since the SChannel module is an essential part of the operating system. Your operating system documentation provides information about the availability of particular protocol versions and ciphers. In addition, you may also want to consult the following:

- Information about the availability of protocol versions: Support for SSL/TLS protocols on Windows
- Information about availability of ciphers is available in the Cipher Suites in SChannel

Policies for CRL verification and protocol version availability are controlled on the operating system level by registry settings and system policies. A detailed description of Windows operating system security administration is outside the scope of this document, but for information about setting available TLS versions on Windows, see TLS/SSL Settings.

For information about configuring the availability of cipher lists, see How to restrict the use of certain cryptographic algorithms and protocols in Schannel.dll.

For information about how to configure the selection of ciphers with Genesys TLS, see Tuning Available Cipher Lists. To understand how protocol selection works in Genesys TLS, see Tuning Protocol Version Availablity.

Genesys Native Applications – Genesys Security Pack on UNIX

When running on Unix-like operating systems (Linux, AIX, Solaris, and so on) Genesys common library loads and uses the Genesys Security Pack on UNIX component (referred to in this document as *Security Pack*) to facilitate secure connections. Security Pack encapsulates the OpenSSL library, and provides most of the features of the OpenSSL library.

> **Important**
>
> Genesys Security Pack does not currently support the Mac operating system. When running on Mac OS X, Genesys Common Library loads and uses the **libgsecurity_openssl.dylib** module that is included with the SIP endpoint SDK installation package (and/or bundled with endpoint executable).

Prior to release 8.5.1, Genesys used the RSA BSAFE SSL-C implementation of secure protocols. Starting with release 8.5.1, the default implementation is replaced with the OpenSSL library. The RSA BSAFE-based implementation is still provided as an alternative in cases when a higher level of backward compatibility and interoperability with legacy applications is required.

OpenSSL was chosen as the underlying TLS implementation because it is a de-facto industry standard

implementation. OpenSSL is constantly under observation by the development and cryptoanalyst community, so all security issues found are resolved promptly. The OpenSSL version used by the Genesys Security Pack is updated whenever it is needed from a security point of view.

> **Important**
>
> If you are using a pre-8.5 release of a Genesys product and want to utilize the latest version of the Security Pack (because you need the latest protocols or security fixes), refer to individual product documentation to determine if you can use it with the version of your product. There is limited interoperability of the latest versions of the Security Pack with Genesys releases before 8.5.

For information about installing and using Genesys Security Pack, see Installing Genesys Security Pack.

Genesys Security Pack is loaded as a shared (**.so**) library whenever the application requires a secure connection for the first time in its lifecycle. Genesys Security Pack is linked to OpenSSL statically, so distribution of additional shared modules is not required.

Genesys Security Pack is designed as a drop-in replacement library. Updating to a newer version (or rolling back to a previous version) of Genesys Security Pack is trivial, and requires only a restart of the application.

Refer to the Genesys Security Pack on UNIX Release Note to determine the OpenSSL version used, and recent modifications.

Backward Compatibility of Genesys Security Pack

The new Security Pack is a drop-in replacement of an existing Security Pack. To upgrade to the OpenSSL version, replace the binary modules. You do not have to make any change to the configuration of existing deployments.

> **Important**
>
> You must restart those applications using secured connections after upgrading to the new Security Pack.

To ensure backward compatibility, the new Security Pack includes a new mode (referred to as *compatibility mode*) that restores some behavior of the old Security Pack. This mode is disabled by default.

> **Warning**
>
> - Compatibility mode should be enabled only as a last resort if the new Security Pack is

encountering compatibility errors in your environment.

- If you are in compatibility mode, Genesys strongly recommends that you take the necessary actions to avoid long-term usage of this mode.

To enable the Security Pack compatibility mode, set the environment variable GCTI_SECPACK_COMPAT_MODE to 1 before starting the application. After starting it, you cannot disable the mode during application runtime.

The following compatibility issue workarounds are enabled by compatibility mode:

- When verifying a peer certificate chain, a chain entry certificate revocation status will be ignored if the certificate is explicitly trusted as a CA in the local configuration (that is, listed in the ca certificate list).
- The peer certificate chain verification process will ignore any non-compatible **Key usage** extension value. For example, a peer certificate without authentication usage will be accepted in compatibility mode. RSA did not verify the **Key usage** extension values; OpenSSL does.

If you want to continue using the RSA BSafe implementation instead of OpenSSL, make sure you set up your environment so that shared modules from the **<Security Pack root>/legacy** folder have been loaded instead of the default modules (located in **<Security Pack root>**).

Java PSDK Implementation

The following provides information about the behaviour of TLS as implemented by the specified options in Java 8 and Java 7. Refer to Genesys Platform SDK documentation and the appropriate Oracle website (given below) for more information.

JAVA 8

No default value for **sec-protocol**
Possible values for **sec-protocol**: `SSLv23 SSLv3 TLSv1 TLSv11 TLSv12`
Default value for **tls-version**: `TLSv1.2` (the real default value is TLS)
Default value for **protocol-list**: `SSLv2Hello SSLv3 TLSv1 TLSv1.1 TLSv1.2`
Other values for **tls-version**: `SSLv3 TLSv1 TLSv1.1 TLSv1.2`
See Oracle JDK 8 for more details.

JAVA 7

No default value for **sec-protocol**
Possible values for **sec-protocol**: `SSLv23 SSLv3 TLSv1 TLSv11 TLSv12`
Default value for **tls-version**: `TLSv1` (the real default value is TLS)
Default value for **protocol-list**: `SSLv2Hello SSLv3 TLSv1 TLSv1.1 TLSv1.2`
Other values for **tls-version**: `SSLv3 TLSv1 TLSv1.1 TLSv1.2`
See Oracle JSE 7 for more details.

> **Warning**
> Java 7 supports TLS 1.1 and TLS 1.2. For client connections, neither version is enabled by default. For server connections, TLS 1.1 and TLS 1.2 are enabled by default.

.NET PSDK Implementation

Platform SDK-based applications that utilize the .NET Framework are configured similarly to what has been described for core applications running on the Windows operating system. Refer to the Release Notes of particular application to see if there are any special considerations.