Genesys Administrator Extension Deployment Guide

Configuring System Security

5/3/2025

Contents

Configuring System Security

GAX has many features that enhance your system security. This section discusses GAX security features and describes how to configure and/or use them.

Default Account Support

Genesys uses a default user account. This is a special account that always has full privileges to all objects and can perform any action. This account ensures that there is always at least one account that enables the administrator to correct permissions and access issues if other administrative accounts are deleted, disabled, or otherwise compromised.

GAX supports the default user account. The default user account always has full access to all the functions that are specified for the GAX role, even if this account does not have any role privileges or explicit permissions specified. When the default account is created during the installation of Configuration Server, it has full control over all configuration objects; however, this account might be deleted or its permissions on objects might be revoked. If this happens, GAX cannot work around the permissions. The default account must have the permissions set to write objects in the Configuration Server.

Use the **default_account_dbid** option to configure the actual account to be used, and that has all privileges assigned, in case the original default user account is disabled for security reasons or has been deleted.

Transport Layer Security (TLS)

GAX employs Transport Layer Security (TLS), a cryptographic protocol that provides security and data integrity for communications over networks such as the Internet. TLS encrypts the segments of network connections at the transport layer from end to end.

GAX supports TLS-enabled connections to the following Genesys servers:

- Configuration Server
- Solution Control Server
- Message Server
- Genesys Deployment Agent

GAX also supports TLS-enabled connections to the GAX database and the LRM database.

For the GAX database connection (either Oracle, Microsoft SQL Server, or PostgreSQL), the database driver and database must also support TLS. For information about configuring your GAX database, refer to the documentation that is specific to the database that you are using:

- Oracle: *Oracle Database Advanced Security Administrator's Guide*
- Microsoft SQL Server: Use the documentation that came with your database application.
- PostgreSQL: Use the documentation that came with your database application.

For information about TLS and detailed instructions about configuring secure connections, and creating and managing certificates, refer to the TLS section of the *Genesys Security Deployment Guide*.

Follow the instructions to create a certificate, assign that certificate to a Host object (which is required for Genesys Server to run in TLS mode), and configure the use of a secured port for the GAX application.

Next, import the server certificate to the trust storage for GAX to enable authentication for TLS connections.

Cipher Lists

Starting in GAX release 8.5.25x, you can customize Jetty SSL configuration to meet customer security requirements by setting supported ciphers for HTTPS. This is done by specifying either the cipher lists used or not used in the **gax.properties** file, adding new parameters to the file as necessary.

You must configure the parameters in the **gax.properties** file before startup of the GAX server. During the GAX startup, all these configurations are read from **gax.properties** and set in the embedded Jetty configuration. If they are not configured, default values will be passed to the Jetty server.

The new parameters are described in the following table:

Parameter	Description
setIncludeProtocols	Specifies one or more protocols to be supported. Valid values are TLSv1, TLSv1.1, and TLSv1.2. Note that JDK1.8 supports only TLSv1.2 protocol.
setIncludeCipherSuites	Specifies one or more cipher suites to be used for encoding data.
setExcludeProtocols	Specifies one or more protocols that are not to be supported. Valid values are TLSv1, TLSv1.1, and TLSv1.2. Note that JDK1.8 supports only TLSv1.2 protocol.
setExcludeCipherSuites	Specifies one or more cipher suites that are not to be used for encoding data.
addExcludeProtocols	Specifies additional protocols that are not to be supported.
addExcludeCipherSuites	Specifies additional cipher suites that are not to be used for encoding data.

Each parameter is a comma-separated list of values, using the same syntax conventions as the Jetty XML descriptor.

For example, the **gax.properties** file might contain something like this::

```
port=2020
host=ca-to-dove
app=GAX_Rob
...
addExcludeCipherSuites=SSL_RSA_WITH_NULL_MD5,SSL_RSA_EXPORT_WITH_RC4_40_MD5
...
```

Note: Regular expressions for ciphers and protocols are not currently supported because OpenSSL accepts only an exact match.

Trust Store

By default, trust storage is in the JRE folder at the following location:
C:\Program Files\Java\jre<Java version>\lib\security\cacerts

> **Important**
>
> JDK is mandatory to install GAX 9.0.000.xx. For more information on recommended JDK versions, see the *Supported Operating Environment Guide* for Genesys Administrator Extension.

The default password is "changeit".

Genesys recommends that you create a separate trust store for GAX.

[+] Create a trust store and import the certificates

Genesys recommends that you do not use the default keystores that are shipped with Java. To ensure a clean separation, you should create a separate storage. If you use a standard **cacert** file, you must re-import the certificates after each Java Virtual Machine (JVM) update.

The trust store should contain the CA certificates that GAX should trust (note that trust store may also contain a certificate used by GAX for HTTPS). If a server sends GAX its certificate during a TLS handshake, GAX will search in its keystore for a matching CA certificate (root and/or intermediate) that was used to sign the remote server certificate. If the CA certificate is found and the remote server certificate is validated, the connection is accepted; otherwise, the connection is rejected.

Prerequisites

- Your Keytool must be configured to your path.
- You have JRE or JDK installed. For more information on recommended JDK versions, see the *Supported Operating Environment Guide* for Genesys Administrator Extension.

Steps

To create a trust and key store that is separate from the default keystores that come with Java, do the following:

1. To create an empty keystore, execute the following command lines on your shell:

```
keytool -genkey -alias initKey -keystore trusted.keystore -storetype jks
 keytool -delete -alias initKey -keystore trusted.keystore
```

2. Make the **trusted.keystore** file readable for the user that owns the GAX process.
3. Set a strong password on your keystore.
4. Add a certificate to the trust store by executing the following command line:

   ```
   keytool -import -alias mssql -keystore trusted.keystore -file "cert/demosrc.cer"
   ```

   Note: It is recommended to use **-importcert** instead of **-import** if you are using Java 8 or above.

   Where:

   - `-alias` corresponds to the certificate being imported; it can be an address within the trust store.
   - `-keystore` specifies the keystore file.
   - `-file` specifies the certificate to be imported. It must be a PEM encoded certificate and the extension can be **.cer** or **.pem**.
5. To display the whole content of a keystore, execute the following command line:

   ```
   keytool -list -keystore trusted.keystore
   ```
6. To display a specific certificate, execute the following command line:

   ```
   keytool -list -v -alias mssql -keystore trusted.keystore
   ```
7. To delete a certificate from the keystore, execute the following command line:

   ```
   keytool -delete -alias mssql -keystore trusted.keystore
   ```

> ### Important
> Most systems have multiple trusted stores. You must always use the same store for GAX.

The following options must be set to configure the trust store location for GAX. The options also enable authentication on a global level for all connections that use a secured port. On Linux or Windows, set these options by adding the following lines to the `setenv.sh` or `setenv.bat` script, respectively.

```
set JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStore="D:\certificates\trusted.keystore"
set JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStorePassword=changeit
```

If you have configured GAX to start as a service, add the following arguments in the **JavaServerStarter.ini** file:

```
-Djavax.net.ssl.trustStore="D:\certificates\trusted.keystore"
-Djavax.net.ssl.trustStorePassword=changeit
```

> ### Important

> GAX does not support Client Authentication. GAX will not authenticate itself by sending a certificate to the server.

Secure Use of the Auto Detect Port

When GAX connects to the Auto Detect (Upgrade) port, the **trustStore** set by the **setenv.sh** or **setenv.bat** script is ignored. You must configure the trust store based on the settings of the Auto Detect port. In addition, you usually must import the GAX certificate into the trust store.

[+] Set up secure connection to Auto Detect port on Windows

On the GAX host:

1. Import the certificate into the Windows certificate store in Microsoft Management Console (MMC), under the same user that starts the GAX processes. If GAX runs under a local system account, there are several ways to import certificates to this local account. This is the most common:
   - Enter `psexec.exe –i –s mmc.exe` on the command line, and then import the certificate for the local system account user.
   - Enter `psexec.exe -i -s certutil -f -user -p [password] -importpfx [path to the certificate]`

   > **Important**
   >
   > - `psexec.exe` with the `-s` flag executes the specified program under the system account.
   > - **psexec** is part of PSTools that you can download from http://technet.microsoft.com/en-US/sysinternals.

2. On the **Options** tab of the GAX application object, in the **[security]** section, create a new option **certificate** and set its value to the thumbprint of the certificate that you imported in step 1.
3. Try to connect to the Configuration Server Auto Detect port and see if it works.

Secure Socket Layer (SSL) Security

Genesys Administrator Extension supports Secure Socket Layer (SSL) communications between the GAX server and client-side connections using the web browser interface.

GAX can support connections through HTTP or HTTPS simultaneously. This is defined through configuration of the **supported_protocol** parameter in the **gax.properties** file, which can be found in the **conf** directory of your GAX installation.

> **Tip**
>
> You must enable X-Forwarded-Proto on the F5 load balancer when GAX is accessed via HTTPS through F5 load balancer on High Availability setup.

GAX is HTTP Strict Transport Security (HSTS) compliant starting from the release 8.5.260.11. HSTS is disabled by default and you can enable it by configuring `enable_hsts=true` in the **gax.properties** file. This is a static change. Once HSTS is enabled, GAX prevents downgrading of encrypted HTTPS connection to unencrypted HTTP. It is implemented by sending a response header record from the server indicating that compliant Web browsers or other HTTP client programs must use HTTPS and they must display the appropriate confirmation message or error message in the browser console.

[+] Set up HTTPS for use with GAX

To set up HTTPS for use with GAX, do the following:

1. Create a keystore file to store the private key and certificate for the GAX server.
   - To create a self-signed certificate, execute the following command:
     ```
     keytool -keystore keystore -alias gax -genkey -keyalg RSA
     ```
     As prompted, enter the information required. Note that it is mandatory to specify the value of **-alias** as **gax**.
2. Rename **gax.properties** to **gax.properties.tmp**. A new **gax.properties** file will be created. At the end of the procedure, when https is set up successfully, the other values in **gax.properties.tmp** must be copied again to **gax.properties**.

> **Warning**
>
> - If you are trying to configure GAX with plug-ins installed, remove the **webapp** and **plug-ins** folders from the GAX installation directory. When GAX configuration is complete, you can rollback all the changes and use the keystore password in your environment.
> - Make sure that GAX **gax_startup.bat** does not contain `GAX_CMD_LINE_ARGS`. Otherwise, GAX cannot enter setup mode.

3. As a local user (whether in person or via a remote desktop connection), log in to GAX as `root` user (`localhost:8080/gax` in GAX installed machine).
4. In another tab, generate gax encryption key using: `http://localhost:8080/gax/api/system/generategaxkey`
   This generates a new encryption key and stores it in a newly created `gax_store.txt` file in the `conf` folder.

> **Important**
>
> Provide read permission only for GAX user for the <GAX_installation_directory>/conf/gax_store.txt file.

> **Important**
>
> For releases from 9.0.100.56 to 9.0.100.66, GAX automatically generates the crypto_code which is used for encryption/decryption while calling the setkeystorepassword webservice API mentioned in step 5. Ignore Step 4 for those releases.

5. Call the webservice API by entering the following in the address bar of your web browser (for example: http://localhost:8080/gax/api/system/setkeystorepassword?password={password}). The password is stored in an obscured fashion in the **gax.properties** file. The password specified must be the same as the password specified for the keystore (see Step 1, above).
   Alternately, use the following commands without a web browser:

```
curl -i --cookie-jar "./cookie.txt" -H "Content-Type: application/json" -d "
{\"username\":\"root\",\"password\":\"\",\"isPasswordEncrypted\":\"false\"}"
http://localhost:8080/gax/api/session/login
curl --cookie "./cookie.txt" http://localhost:8080/gax/api/system/
setkeystorepassword?password=password
```

> **Important**
>
> Password can contain special characters except #, %, &, +, and `.

> **Important**
>
> For releases from 9.0.100.72, use the following commands without a web browser:

```
curl -i --cookie-jar "./cookie.txt" -H "Content-Type: application/json" -d "
{\"username\":\"root\",\"password\":\"\",\"isPasswordEncrypted\":\"false\"}"
http://localhost:8080/gax/api/session/login
curl --cookie "./cookie.txt" http://localhost:8080/gax/api/system/generategaxkey
curl --cookie "./cookie.txt" http://localhost:8080/gax/api/system/
setkeystorepassword?password=password
```

> **Important**
> root mentioned in the above commands is the GAX root user.

- Define the parameter https_port in gax.properties with the secured port number according to your setup. The default HTTPS port for Tomcat is 8443. See Configuring GAX Properties for more information.
  - https_port=8443
  - supported_protocol=https or both.
  - keystore_path=full path of the location of keystore
- Restart GAX in HTTPS mode.
  For example gax.properties in conf folder contains the following:

```
keystore_password=***/SGo*********moXg\=\=
backup port=2020
port=2020
backup host=
https_port=8443
supported_protocol=https
keystore_path=C:\\OpenSSL-Win32\\bin\\keystore
host=xxx.xx.xxx.xx
app=gax_https1
```

> **Important**
> Sample keystore_path for Linux is keystore_path=/opt/genesys/gax/conf/keystore

TLS: Preparing Genesys Management Framework

To enable GAX to connect securely to Genesys servers, you must configure the Genesys Framework as described in the *Genesys Security Deployment Guide*. Follow the instructions in this guide to create and manage certificates and make them usable within Genesys Framework.

Configuration Server

You must meet the following conditions to create a secure connection to Configuration Server:

1. Create a an Auto Detect listening port for your Configuration Server with a certificate configured.
2. Configure the GAX Server to connect when it starts up to the Configuration Server Auto Detect port by setting the GAX Server -port property. In the **Start Info** tab of the GAX_Server Properties dialog box, enter the following settings:

- Working Directory: `/path/gax`
- Command Line: `./startup.sh`
- Command Line Arguments: `-host <host name> -port <auto detect port number> -app GAX_Server`

Message Server and Solution Control Server

Both Message Server and Solution Control Server are configured the same way.

1. Create a `Secured` port for Message Server and Solution Control Server.
2. Configure the GAX Server to connect to Message Server and Solution Control Server by using the *specific* `Secured` ports that you have created. In the `Properties` dialog box for the server and in the `Connections` tab of the `GAX_Server` dialog box, secured ports are displayed with a key symbol icon.
3. Restart GAX Server to connect over an encrypted session by using the secure ports.

Genesys Deployment Agent

> **Important**
>
> Genesys Deployment Agent (GDA) is no longer installed and supported as part of Management Framework 8.5.1 (starting from Local Control Agent *O8.5.100.31*), and therefore all functionality using GDA (including the installation of IPs) is deprecated.

Configuring mutual TLS between GAX and Configuration Server

To support mutual TLS connection between GAX and Configuration Server, you must do the following:

1. **Download OpenSSL**
2. **Add environmental variable**
   1. Set `path = C:\OpenSSL-0.9.8\bin\`
   2. Set `OPENSSL_CONF=C:\openssl-0.9.8\OpenSSL.cnf`
3. **Prepare certificate**
   1. Go to OpenSSL location.
   2. Run `Openssl>req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 5000 -config "c:\openssl-0.9.8\openssl.cnf`
   3. Set `passphrase = changeit` (password that will be used later)
   4. `Openssl> x509 -outform der -in cert.pem -out cert.der`
   5. Copy `cert.der cert.cer` in a folder, which will be used later (optional)
   6. Remove password:
      `OpenSSL rsa -in key.pem -out keynopass.pem`

7. Combine certificate and key in 1 file:
   `openssl pkcs12 -inkey keynopass.pem -in cert.pem -export -out cert.pfx`

4. **Configure certificate**
   1. In the Configuration Server machine, configure Configuration Server certificates by using the mmc utility.
   2. Import your **cert.pfx** from CME-certs directory into **Personal** > **Certificates**.
   3. Import **cert.der** (or .cer or .pem, as per UI) into **Trusted Root certification** > **certificates**.
   4. In the GAX machine, create a directory **c:\install\openssl\GAX-certs**.
   5. Copy the **Certificates** folder from the Configuration Server machine to **c:\install\openssl\GAX-certs**.
   6. Type `changeit` if prompted for password.
   7. Repeat step 1 and 2.
   8. Navigate to the location **c:\install\OpenSSL\GAX-certs** and open OpenSSL.
   9. `Openssl>pkcs12 -export -in cert.pem -inkey keynopass.pem -certfile cert.pem -out cert-and-key.p12`.
   10. Open cmd and navigate to **c:\install\openssl\GAX-certs in cmd**.
   11. Run `keytool -importkeystore -srckeystore cert-and-key.p12 -srcstoretype pkcs12 -destkeystore gax.jks -deststoretype JKS`.
   12. Run `keytool -list -v -keystore gax.jks`.
   13. Run `keytool -importcert -alias CMEcert -file c:\install\openssl\CME-certs\cert.der -keystore gax.jks -storepass changeit`.

14. **Configure confserv application in Configuration Server**
   1. Select **Confserv** and create port `3040` as auto detect.
   2. Open **Port** > **Certificate** tab and click **Import Certificate** and then choose self-signed certificate for Configuration Server machine (the machine that you configured and not the existing machine).
   3. Add thumb print copied from mmc.
   4. In the **Advanced** tab of port, add `tls-mutual=1`, so that the output is similar to: `upgrade=1;tls-mutual=1;certificate=01 11 A2 B0 42 D6 99 C8 C7 F8 C6 21 58 DD AF E4 2D FF 51 FD`.

5. **Configure in GAX application**
   1. Go to GAX installed path.
   2. Edit gax start up file with the following options:

```
set JAVA_OPTS=-server -XX:MaxPermSize=512m -XX:+CMSClassUnloadingEnabled
 -XX:+UseConcMarkSweepGC -XX:+HeapDumpOnOutOfMemoryError
 set JAVA_OPTS=%JAVA_OPTS%
 -Dcom.genesyslab.platform.commons.log.loggerFactory=com.genesyslab.platform.commons.log.Log4JLoggerF
 set JAVA_OPTS=%JAVA_OPTS% -Djavax.net.debug=all
 set JAVA_OPTS=%JAVA_OPTS% -Djavax.net.debug=ssl:handshake
```

```
set DEBUG_OPTS=-Xdebug
-Xrunjdwp:transport=dt_socket,address=9002,server=y,suspend=n
set SECURITY_OPTS=-Djavax.net.ssl.trustStorePassword=changeit
-Djavax.net.ssl.trustStore=C:\install\openssl\GAX-certs\gax.jks
set JAVA_OPTS=%JAVA_OPTS% %SECURITY_OPTS%
java %JAVA_OPTS% %DEBUG_OPTS% -jar gax.war
```

3. Add line `mf_tls_mutual=true` in the **gax.properties** file.
4. Change port from `2020` to `3040` in **gax.properties** file.
5. Restart Configuration Server and GAX.

Configuring mutual TLS between GAX and Solution Control Server

To support mutual TLS connection between GAX and Solution Control Server (SCS), you must do the following:

1. **Download OpenSSL**
2. **Add environmental variable**
   1. Set `path = C:\OpenSSL-0.9.8\bin\`
   2. Set `OPENSSL_CONF=C:\openssl-0.9.8\OpenSSL.cnf`
3. **Prepare certificate**
   1. Go to OpenSSL location.
   2. Run `Openssl>req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 5000 -config "c:\openssl-0.9.8\openssl.cnf`
   3. Set `passphrase = changeit` (password that will be used later)
   4. `Openssl> x509 -outform der -in cert.pem -out cert.der`
   5. Copy `cert.der cert.cer` in a folder, which will be used later (optional)
   6. Remove password:
      `OpenSSL rsa -in key.pem -out keynopass.pem`
   7. Combine cert and key in 1 file:
      `openssl pkcs12 -inkey keynopass.pem -in cert.pem -export -out cert.pfx`
4. **Configure certificate**
   1. In SCS machine, configure certificates by using the mmc utility.
   2. Import your **cert.pfx** from CME-certs directory into **Personal** > **Certificates**.
   3. Import **cert.der** (or .cer or .pem, as per UI) into **Trusted Root certification** > **certificates**.
   4. In the GAX machine, create a directory **c:\install\openssl\GAX-certs**.
   5. Copy the **Certificates** folder from SCS machine to **c:\install\openssl\GAX-certs**.
   6. Type `changeit` if prompted for password.

7. Repeat step 1 and 2.
8. Navigate to the location **c:\install\OpenSSL\GAX-certs** and open OpenSSL.
9. Run `Openssl>pkcs12 -export -in cert.pem -inkey keynopass.pem -certfile cert.pem -out cert-and-key.p12`.
10. Open cmd and navigate to **c:\install\openssl\GAX-certs in cmd**.
11. Run `keytool -importkeystore -srckeystore cert-and-key.p12 -srcstoretype pkcs12 -destkeystore gax.jks -deststoretype JKS`.
12. Run `keytool -list -v -keystore gax.jks`.
13. Run `keytool -importcert -alias CMEcert -file c:\install\openssl\CME-certs\cert.der -keystore gax.jks -storepass changeit`.

14. **Configure SCS application in Configuration Server**
    1. Select **Confserv** and create port 3045 as auto.
    2. Open **Port** > **Certificate** tab and click **Import Certificate** and then choose self-signed certificate for SCS machine (the machine that you configured and not the existing machine).
    3. Add thumb print copied from mmc.
    4. In the **Advanced** tab of port, add `tls-mutual=1`, so that the output is similar to: `upgrade=1;tls-mutual=1;certificate=01 11 A2 B0 42 D6 99 C8 C7 F8 C6 21 58 DD AF E4 2D FF 51 FD`.

5. **Configure in GAX application**
    1. Go to GAX installed path.
    2. Edit gax start up file with the following options:

```
set JAVA_OPTS=-server -XX:MaxPermSize=512m -XX:+CMSClassUnloadingEnabled
 -XX:+UseConcMarkSweepGC -XX:+HeapDumpOnOutOfMemoryError
 set JAVA_OPTS=%JAVA_OPTS%
 -Dcom.genesyslab.platform.commons.log.loggerFactory=com.genesyslab.platform.commons.log.Log4JLoggerF
 set JAVA_OPTS=%JAVA_OPTS% -Djavax.net.debug=all
 set JAVA_OPTS=%JAVA_OPTS% -Djavax.net.debug=ssl:handshake
 set DEBUG_OPTS=-Xdebug
 -Xrunjdwp:transport=dt_socket,address=9002,server=y,suspend=n
 set SECURITY_OPTS=-Djavax.net.ssl.trustStorePassword=changeit
 -Djavax.net.ssl.trustStore=C:\install\openssl\GAX-certs\gax.jks
 set JAVA_OPTS=%JAVA_OPTS% %SECURITY_OPTS%
 java %JAVA_OPTS% %DEBUG_OPTS% -jar gax.war
```

    3. Add line `mf_tls_mutual=true` in the **gax.properties** file.
    4. Go to Configuration Server, select GAX, and then in the **Connections** tab, add SCS.
    5. Go to Port and change port to the secured port (auto).
    6. Restart SCS and GAX.

Disabling Authentication for Certain Connections

The configuring steps outlined above engage authentication for Configuration Server, Message Server, and Solution Control Server. If GAX uses the secure ports to connect to Message Server and Solution Control Server, both server-side certificates will automatically be validated against the trust storage.

In certain rare cases, you might want to disable authentication for one of the connections. To do this, add the following line to the **Advanced** tab of the **Properties** dialog box in the Transport parameters section:

```
"disableAuthentication=1"
```

Do not use white spaces. To separate this option from other options, use a semi-colon.

To disable TLS authentication for Configuration Server, add the following line to the following files:

- (Linux) `setenv.sh`:

```
JAVA_OPTS="$JAVA_OPTS -Dgax.configserver.validate.cert=off"
```

- (Windows) `setenv.bat`:

```
set JAVA_OPTS=%JAVA_OPTS% -Dgax.configserver.validate.cert=off
```

> **Important**
>
> - Connections to Message Server and Solution Control Server fail if GAX does not find the received certificate in the trust store, or if Message Server and Solution Control Server do not send a certificate.
> - Connections also fail to Configuration Server and databases if they are configured for authentication and the certificate is not in the trust store.

TLS: Configuring the GAX Database

You must configure your Oracle, Microsoft SQL, or PostgreSQL server to use TLS. In addition to the appropriate procedure below, refer to the documentation that came with your database for information on how to use TLS security.

[+] Configuring the GAX Oracle Database for TLS

Prerequisite:

- Setting up the Genesys Administrator database (for Oracle)

Steps

1. Configure Oracle as described in the related database guides, and configure a TCPS listener.
2. Set the level of TLS control on the DAP.
   - In the GAX section of the DAP, create an option that is named `tls_mode`.
   - Specify one of the following values for the `tls_mode` option:
     - `off`—No TLS will be used.
     - `required`—If a server does not support TLS, revoke the connection.
     - `authentication`—GAX will validate the server send-certificate with the local trust store.
     - `<option not set>`—Same as `off`.

[+] Configuring the GAX MS SQL Database for TLS

Procedure: Configuring the GAX Database for TLS (Microsoft SQL Server)

Prerequisites

- Set up the Genesys Administrator database for Microsoft SQL Server.
- Ensure that you are using the latest JTDS driver (1.2.5 or later).

Steps

1. Configure Microsoft SQL Server as described in the related database guides.
2. Set the level of TLS control on the DAP.
   - In the GAX section of the DAP, create an option that is named `tls_mode`.
   - Specify one of the following values for the `tls_mode` option:
     - `off`—Do not use TLS.
     - `request`—If the server supports TLS, it is used.
     - `required`—If the server does not support TLS, the connection is revoked.
     - `authentication`—GAX validates the server-send certificate against the local trust store.
     - `<option not set>`—Same as `off`.
3. Verify that the configured port is identical to the TLS listener port of Microsoft SQL Server
4. Due to an incompatibility between newer versions of Java and the Microsoft SQL Server driver, disable CBC Protection to enable GAX to connect to a Microsoft SQL Server database.
   - For Windows, add the following line to the **setenv.bat** file:

```
set JAVA_OPTS=%JAVA_OPTS% -Djsse.enableCBCProtection=false
```

- For Linux, add the following line to the **setenv.sh** file:
```
JAVA_OPTS="$JAVA_OPTS -Djsse.enableCBCProtection=false"
```

[+] Configuring the GAX PostgreSQL Database for TLS

Procedure: Configuring the GAX Database for TLS (PostgreSQL)

Prerequisites

- Set up the Genesys Administrator database for PostgreSQL.

Steps

1. Configure PostgreSQL as described in the related database guides.
2. Set the level of TLS control on the DAP.
   - In the GAX section of the DAP, create an option that is named `tls_mode`.
   - Specify one of the following values for the `tls_mode` option:
     - `off`—Do not use TLS.
     - `required`—If the server does not support TLS, the connection is revoked.
     - `authentication`—GAX validates the server-send certificate with the local trust store.
     - `<option not set>`—Same as `off`.

Cross-site Scripting and Cookies

You can configure your system to improve the protection of Genesys Administrator Extension against Cross-site Scripting (XSS) attacks by configuring the `HttpOnly` and `Secure` flags on your HTTP server to further enhance the existing GAX security. These flags tell browsers how to handle cookies.

Server-side cookies can be tagged with `HttpOnly` and `Secure` flags to tell the browser how to deal with them. To achieve a maximum level of security, administrators must make this configuration on the Application Server.

Securing Server-side Cookies

HttpOnly Flag

Setting the `HttpOnly` flag on cookies forces the browser to prevent (disallow) scripts from accessing the cookies. This prevents JavaScript that might be introduced through an XSS attack into a browser page to access cookie data and send it to a different person. Stolen cookie data can also be used to hijack a browser session.

When GAX is running on an embedded Jetty system, GAX automatically sets the `HttpOnly` flag on the JSESSIONID session cookie. You can turn off this flag by adding the following line in the **gax.properties** file:

```
session_httponly = false
```

If GAX is using an external Tomcat web server, open and edit the `$CATALINA_HOME/conf/context.xml` file.

To set the `HttpOnly` flag, add the following attribute:

```
useHttpOnly="true"
```

The main tag should be:

```
<Context useHttpOnly="true">
```

Secure Flag

With the Secure flag set, cookies are transmitted only from the browser to the server when the connection is secured by using the HTTPS protocol. This setting is applicable to HTTPS connections only. Therefore, you must configure GAX to use an HTTPS connector, not an HTTP connector.

When GAX is running on an embedded Jetty system and **supported_protocol** is set to `https` in **gax.properties**, GAX automatically sets the `Secure` flag on the JSESSIONID session cookie. You can turn off this flag by adding the following line to the **gax.properties** file:

```
session_securecookies = false
```

If GAX is using an external Tomcat web server, open and edit the `$CATALINA_HOME/conf/server.xml` file.

To set the `Secure` flag, add the following attribute to the HTTPS connector:

```
secure="true"
```

The flag must not be applied to any non-HTTPS connectors. If you apply the flag to an HTTP connection, it will become unusable for Genesys Administrator Extension.

The following is an example of a valid connector:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
keystoreFile="/home/gcti/keystore.key" keystorePass="genesys"
clientAuth="false" sslProtocol="TLS" />
```

Content Security Policy

GAX supports Content Security Policy (CSP) starting from the release 8.5.260.11. CSP helps users in mitigating Cross-Site Scripting, man-in-the-middle attacks, and data ex-filtration. You can limit the trusted external resources using your CSP.

You can frame your own CSP and configure the CSP headers for GAX using the `content_security_policy` option in the **gax.properties** file. This is a static change. Once CSP is configured, GAX adds the **Content-Security-Policy** header in the response with the configured policy.

> **Important**
>
> Micorsoft Internet Explorer does not support Content Security Policy.

Inactivity Timeout

For security purposes, GAX can be configured to lock the application if an administrator has not used the keyboard or mouse for a period that you specify. All user input is blocked until the administrator provides login information to unlock the application. This feature ensures that no unauthorized user can access an unattended terminal that is running GAX.

Use the `inactivity_timeout` option to specify the amount of time in seconds of administrator inactivity (no mouse or keyboard usage) that triggers application locking. If the administrator has been inactive longer than the number of seconds that is specified by the `inactivity_timeout` option, the administrator must re-authenticate to be able to use the GAX application. A negative value disables this functionality.

GAX employs a keep-alive strategy to prevent *session* timeout; this ensures that GAX maintains your session even if the inactivity timeout feature locks the application and requires you to log in.