Framework Database Connectivity Guide

PostgreSQL Databases

12/13/2025

Contents

PostgreSQL Databases

You must make PostgreSQL client software accessible in the environment where the Genesys application is running.

Genesys can use PostgreSQL 9.0 or PostgreSQL 10.1 client software. By default, Genesys installation will utilize PostgreSQL 9.0 client software to access all supported versions of PostgreSQL. You can choose to use the alternative version as described in Using an Alternative Version of PostgreSQL Client Software, below.

Using PostgreSQL Client Software

The vendor client software must be in the folder specified in the environment variable **PATH** (for Windows), **LIBPATH** (for AIX), or **LD_LIBRARY_PATH** (for Linux and Solaris). Refer to PostgreSQL documentation for more information.

The following is an example of the configuration of Genesys Database Access Point parameters for PostgreSQL:

```
dbengine=postgre
dbserver=<postgresql server host>
dbname=<database name>
username=<user name>
password=<password>
```

Connectivity to PostgreSQL relies on TCP/IP between server and client. PostgreSQL client software uses operating system settings for the TCP/IP stack to determine how long to wait for a response form the PostgreSQL server after submitting a request to it. For example, on Linux it may take up to two hours to detect a disconnection, unless the Operating System parameter **tcp_keepalive_time** is adjusted. Refer to documentation for your operating system for more information.

Important

It is mandatory to use the PostgreSQL default port, 5432. Multiple cluster environment is not supported.

PostgreSQL Strings

When using PostgreSQL Server 9.2 or higher, make sure the following parameters are set in the **postgresql.conf** file, as follows:

```
bytea_output ='escape'
standard_conforming_strings='off'
```

These parameters enable the use of escape characters, namely backslashes in a string. By default, backslash characters in strings are treated as ordinary characters by the **standard_conforming_strings** parameter set to on, overriding the value of **bytea_output**. Setting **standard_conforming_strings** to `off` treats backslashes as escape characters, to work with **bytea_enum**. For more information about these parameters, refer to PostgreSQL documentation.

Using an Alternative Version of PostgreSQL Client Software

Genesys provides newer versions of some dbclients in the **dbclient_next** folder of the component's Installation Package (IP). Use these processes to enable support of new databases and features. Only 64-bit versions of alternate clients are provided.

To use the latest functionality with a component, you must do the following:

1. Replace the dbclients in the root installation folder with the dbclients from the **dbclient_next** folder.
2. On the hosts that will be using this component, install the database client software and make it available for the component.

Secure Connections with PostgreSQL DBMS

You can use TLS via OpenSSL to secure connections to PostgreSQL databases.

A secure connection to a PostgreSQL database is required to set up and use SSL. OpenSSL must be installed on both the client and the server systems. This section describes how to configure the connection and enable SSL on both the PostgreSQL server and the database client. Note that the configuration steps given in this section are supported only by PostgreSQL Server version 9.2 and above.

The default directories, from which the client and server obtain their certificates, can be overridden by setting appropriate configuration options. The default directories are:

- Client: (if necessary, create the folder)
    - Linux platform: **~/.postgresql/**
    - Windows platform: **%APPDATA%\postgresql\**
- Server:
    - Postgre SQL Server's data directory

On PostgreSQL Server

To enable SSL on the PostgreSQL server, set the parameter **ssl** to on in the **postgresql.conf** configuration file. This enables the server to listen for both normal and SSL connections on the same TCP port, based on the options specified by client.

On UNIX systems, the permissions of **server.key** must not allow any access to the world or group.

You can achieve this by issuing the following command:

```
chmod 0600 server.key
```

If the private key is protected by a passphrase, the server will prompt for the passphrase and will not start until the correct passphrase is entered.

To start in SSL mode, files containing the server certificate and private key must exist and be available. By default, these files are expected to be named **server.crt** and **server.key**, and are located in the PostgreSQL Server's data directory. If you are using other names and locations, specify them in **postgresql.conf** with the parameters **ssl_cert_file** and **ssl_key_file**, respectively. You can use relative or absolute paths, but all relative paths must be relative to the server's data directory.

If you are using mutual TLS, you need to verify the client's trusted certificate. Put all Certificate Authorities (CAs) in a single file, such as **root.crt**, and put that file in the PostgreSQL server's data directory. In the **postgresql.conf** file, set the parameter **ssl_ca_file** to the name, and path if appropriate, of the CA file.

The following is an example of settings in the **postgresql.conf** file.

```
ssl = on
#ssl_ciphers = 'DEFAULT:!LOW:!EXP:!MD5:@STRENGTH'    # allowed SSL ciphers
#ssl_renegotiation_limit = 512MB          # amount of data between renegotiations
ssl_cert_file = 'WIN-VI3D4A69M8O.crt'
ssl_key_file = 'WIN-VI3D4A69M8O.key'
ssl_ca_file = 'cert_authority.crt'
#ssl_crl_file =
```

Notes:

- Hash symbols (#) indicate comments in the file, or lines that are currently not read by the system.
- A change in any of the parameters except `ssl_renegotiation_limit` requires a restart to take effect.

The parameters in the file are described in the following table.

Option	Default Value [a]	Valid Values	Contents [b]	Description
ssl_cert_file	Linux: $PGDATA/ server.crt Windows: %PGDATA%\server.crt	Absolute path, or path relative to data directory, of server certificate file	Server certificate	Sent to client to identify server.
ssl_key_file	Linux : $PGDATA/ server.key Windows: %PGDATA%\server.key	Absolute path, or path relative to data directory, of server private key file	Server private key	Verifies that the server certificate was sent by the owner; it does not indicate that the certificate owner is trustworthy.
ssl_ca_file	No value (empty)	Absolute path, or path relative to data directory, of server CA file	Trusted Certificate Authorities (CAs)	List of trusted CAs, against which the client certificate is compared to ensure it was

Option	Default Value [a]	Valid Values	Contents [b]	Description
				signed by a trusted CA.
ssl_crl_file	No value (empty)	Absolute path or path relative to data directory, of revoked certificate list file	Certificates revoked by CAs	List of revoked certificates; the client certificate must not be on this list.

a. PGDATA is the server's data directory, and is located by default as follows:

- Linux: **/var/lib/pgsql/<version>/data**
- Windows: **C:\Program Files\PostgreSQL\<version>\data**

b. If any existing certificates are replaced by new ones, you must restart the PostgreSQL server.

In the authentication configuration file (**pg_hba.conf** located in the server's data directory), configure records that support SSL, using one of the following methods:

- `host database user address auth-method [auth-options]`
  This record matches connection attempts made using TCP/IP, and match either SSL or non-SSL connection attempts.
- `hostssl database user address auth-method [auth-options]`
  This record matches connection attempts made using TCP/IP, but only when the connection is made with SSL encryption. If you are using mutual TLS, add `clientcert=1` at the end of this line to require the clinet to supply a trusted certificate.

The **clientcert** option in **pg_hba.conf** is available for all authentication methods, but only for rows specified as `hostssl`. When **clientcert** is not specified or is set to `0`, the PostgreSQL server will still verify any client certificates presented to it against its own CA or CA list, but it will not insist or expect a client certificate to be presented.

The following is an example of the settings in the **pg_hba.conf** file:

```
# TYPE DATABASE USER ADDRESS METHODs
# IPv4 local connections:
  host all all 127.0.0.1/32 trust
# IPv6 local connections:
  host all all ::1/128 md5
  host replication postgres 127.0.0.1/32 md5
  host replication postgres ::1/128 md5
  host all all 172.24.128.47/32 md5
  hostssl configdb postgres 10.31.0.50/32 password clientcert=1
```

Warning

The **ssl_ca_file**, **ssl_cert_file**, **ssl_crl_file**, and **ssl_key_file** options are not supported by Postgre SQL server earlier than version 9.2.

On PostgreSQL Client

To configure a secure connection on a PostgreSQL client, you must first enable the SSL connection by setting the **sslmode** option on the client. Set this option to one of the following values:

Value	Description
prefer	(Default) First try an SSL connection; if that fails, try a non-SSL connection. Furthermore, use this value only where the record uses *hostssl*. Note that this value is provided only for backward compatibility, and is not recommended in secure deployments.
allow	First try a non-SSL connection; if that fails, try an SSL connection. Furthermore, use this value only where the record uses *hostssl*.
disable	Try only a non-SSL connection.
verify-ca	Try only an SSL connection, and verify that the server certificate is issued by a trusted certificate authority (CA).
verify-full	Try only an SSL connection, verify that the server certificate is issued by a trusted CA, and verify that the server host name matches the SSL Name/ Common Name (CN) of the certificate.
require	Try only an SSL connection. If a root CA file is present, verify that the server certificate is issued by a trusted CA.

If the option is set to `verify-ca` or `verify-full`, a root CA certificate must be available.

For a mutual TLS connection, the server requests a trusted client certificate, which must be signed by a CA trusted by the server. A matching private key file must also be available.

Using Secure Connections to PostgreSQL Databases

If the Postgre SQL server is enabled for SSL connection, a Genesys Application can enable or disable it by using the appropriate environment variables described in this section.

Important

The default values will be used only if SSL is enabled in the PostgreSQL server, otherwise the environment variables will be empty if not set. The environment variable names and corresponding values are case sensitive.

PGMODE
Default value: `prefer`
Valid values: `disable`, `allow`, `prefer`, `require`, `verify-ca`, `verify-full`

Sets the client negotiation mode with the server. See the description of the values here.

PGROOTCA
Default value: `~/.postgresql/root.crt` (on Linux) or `%APPDATA%\postgresql\root.crt` (on Windows)
Valid values: Absolute path, or relative path from default directory, of Certificate authority file. For example: `D:\\MFWK\\cert\\Automated_cert_gen\\cert_authority.crt` or `/home/genesys/certs/cert_authority.crt`

PostgreSQL client will use this file to verify the server certificate.

PGCERT
Default value: `~/.postgresql/postgresql.crt` (on Linux) or `%APPDATA%\postgresql\postgresql.crt` (on Windows)
Valid values: Absolute path, or relative path from default directory, of client certificate file. For example: `D:\\MFWK\\cert\\Automated_cert_gen\\server_cert.crt` or `/home/genesys/certs/server_cert.crt`

PostgreSQL client passes this file to the server which will be verified by the server if a CA is present.

PGKEY
Default value: `~/.postgresql/postgresql.key` (on Linux) or `%APPDATA%\postgresql\postgresql.key` (on Windows)
Valid values: Absolute path, or relative path from default directory, of client private key file. For example: `D:\\MFWK\\cert\\Automated_cert_gen\\server_cert.key` or `/home/genesys/certs/server_cert.key`

PostgreSQL client passes this file with the certificate to the server for verification.

PGCRL
Default value: `~/.postgresql/postgresql.crl` (on Linux) or `%APPDATA%\postgresql\postgresql.crl` (on Windows)
Valid values: Absolute path, or relative path from default directory, of certificate revocation list file. For example: `D:\\MFWK\\cert\\Automated_cert_gen\\revoked.crl` or `/home/genesys/certs/revoked.crl`

PostgreSQL client checks if the server certificate is present in the list in this file; if the certificate is listed, the connection is not allowed.

PGAUTH
Default value: `password`
Valid values: `password`, `cert`

Specifies authentication method to be used, as follows:

- `password` - Password authentication; works for SSL and non-SSL connections.
- `cert` - Certificate authentication; works for only SSL connections.

Error Messages:

The following table lists the various error messages that you might encounter when setting up and using a secure connection to a PostgreSQL database.

Error Message	Meaning
SSL error : tlsv1 alert unknown ca	Client certificate exists but is not trusted by server CA.
root certificate file "%APPDATA%/postgresql/root.crt" does not exist	Client root CA is not present in default directory or not set in environment, when **sslmode=verify-ca or verify-full**.
FATAL: connection requires a valid client certificate	Client Certificate is not present in default directory or not set in environment, when **clientcert=1**.
Certificate present, but no private key file	Client private key file is not present in default directory or not set in environment, when **clientcert=1**.
FATAL: could not load pg_hba.conf	Server CA certificate is not set or configured properly, when clientcert=1.
FATAL: could not load pg_hba.conf	Configuration error in **pg_hba.conf** file.
FATAL: connection requires a valid client certificate	Server authentication mode is "cert" and no client certificate is provided.
Certificate present, but no private key file	Server authentication mode is "cert" and no client private key is provided.

Advanced Authentication methods with Postgre SQL DBMS

Genesys software supports the following PostgreSQL authentication methods:

- Password authentication over a plain TCP or TLS connection
- Certificate authentication This method is supported only when the connection is secured using SSL.

Password Authentication Through TCP/IP

Password authentication can be carried out over a plain (unsecured) TCP connection, or a secured TLS connection.

When SSL is enabled on both the server and client side, the encrypted or unencrypted password will be sent over a secure connection (TLS). When establishing the secure connection, you must follow basic rules of secure connection, such as:

- Server and Client machines must each have a CA that is able to verify certificates from the other machine.
- Certificate CNs should match the Host Name of the machine on which each certificate is installed.

Passwords can be authenticated using one of the following methods in PostgreSQL server:

- **md5**—An encrypted password is passed over the network.
- **password**—An unencrypted password is passed over the network.

Another related method, **trust**, does not pass any password over the network. In other words, it allows the client to log in without any authentication.

Methods are specified in the **pg_hba.conf** file. For example:

```
host all all 172.24.128.47/32 md5
hostssl configdb postgres 10.31.0.50/32 password
host all all 127.0.0.1/32 trust
```

Certificate Authentication Through Secured TCP/IP connection

Instead of passing a password over the network, use this method to perform authentication over the secure connection with the same certificates.This authentication method uses SSL client certificates to perform authentication, and is therefore available only for SSL connections. When using this method, the server requires that the client provide a valid certificate. No password prompt is sent to the client.

Before using this method, you must set **PGAUTH=cert**.

This method compares the CN attribute of the certificate to the requested database user name, and if they match, the login is allowed. For example:

```
hostssl configdb postgres 10.31.0.50/32 cert
```

If you are using mutual TLS, you must provide a client certificate and a key file. The username and the CN of the certificate should match. However, if you want to allow the situation in which the certificate's CN is different from the requested database user name, you can use what is called *username mapping*, with which you map the CN name to the requested PostgreSQL username using the **pg_ident.conf** file, as follows:

In the **pg_hba.conf** file, use the `map` option to set a map name for the client connection, such as `map=<mapname>`. For example:

```
hostssl configdb postgres 10.31.0.50/32 cert map=dbname
```

Then, in the **pg_ident.conf**, add the mapping information. For example:

```
# MAPNAME SYSTEM-USERNAME PG-USERNAME
dbname localuser postgres
```

where:

- MAPNAME is the value of the map option in the **pg_hba.conf**.
- SYSTEM-USERNAME is the detected user name of the client from CN
- PG-USERNAME is the requested PostgreSQL user name.

Using PostgreSQL Databases with National Languages

Single Language Deployment

You must create all PostgreSQL databases using the same character set, for example `WIN1252`. You must select encoding that matches Microsoft Windows Operating System default encoding for the selected language so applications, like Interaction Routing Designer, display data correctly.

On every host that has Genesys applications accessing PostgreSQL databases, make sure that the language and encoding environment variable (or the settings for non-Unicode applications, if you are using Windows) is set to match character encoding of data in the PostgreSQL database. If there is a discrepancy between the encoding that the database and the local client are using, set the environment variable **PGCLIENTENCODING** on the host where the client software is running to match the database (for example, **PGCLIENTENCODING=**`Win1252`), based on the following table:

Name	Description	Language	Server?	Bytes/Char	Aliases
GBK	Extended National Standard	Simplified Chinese	No	1-2	WIN936, Windows936
ISO_8859_5	ISO 8859-5, ECMA 113	Latin/Cyrillic	Yes	1	
ISO_8859_6	ISO 8859-6, ECMA 114	Latin/Arabic	Yes	1	
ISO_8859_7	ISO 8859-7, ECMA 118	Latin/Greek	Yes	1	
ISO_8859_8	ISO 8859-8, ECMA 121	Latin/Hebrew	Yes	1	
JOHAB	JOHAB	Korean (Hangul)	Yes	1-3	
LATIN1	ISO 8859-1, ECMA 94	Western European	Yes	1	ISO88591
LATIN2	ISO 8859-2, ECMA 94	Central European	Yes	1	ISO88592
LATIN3	ISO 8859-3, ECMA 94	South European	Yes	1	ISO88593
LATIN4	ISO 8859-4, ECMA 94	North European	Yes	1	ISO88594
LATIN5	ISO 8859-9, ECMA 128	Turkish	Yes	1	ISO88599
LATIN6	ISO 8859-10, ECMA 144	Nordic	Yes	1	ISO885910
LATIN7	ISO 8859-13	Baltic	Yes	1	ISO885913
LATIN8	ISO 8859-14	Celtic	Yes	1	ISO885914
LATIN9	ISO 8859-15	LATIN1 with Euro and accents	Yes	1	ISO885915

Name	Description	Language	Server?	Bytes/Char	Aliases
LATIN10	ISO 8859-16, ASRO SR 14111	Romanian	Yes	1	ISO885916
SJIS	Shift JIS	Japanese	No	1-2	Mskanji, ShiftJIS, WIN932, Windows932
SQL_ASCII	unspecified	any	Yes	1	
UHC	Unified Hangul Code	Korean	No	1-2	WIN949, Windows949
UTF8	Unicode, 8-bit	all	Yes	1-4	Unicode
WIN866	Windows CP866	Cyrillic	Yes	1	ALT
WIN874	Windows CP874	Thai	Yes	1	
WIN1250	Windows CP1250	Central European	Yes	1	
WIN1251	Windows CP1251	Cyrillic	Yes	1	WIN
WIN1252	Windows CP1252	Western European	Yes	1	
WIN1256	Windows CP1256	Arabic	Yes	1	
WIN1258	Windows CP1258	Vietnamese	Yes	1	ABC, TCVN, TCVN5712, VSCII

For more information, refer to PostgreSQL documentation here.

With the environment set up this way, you can use character data in a single language (such as French) for all information stored and transmitted between Genesys applications.

Multiple Languages Deployment

To enable storage and processing of data in multiple languages using a PostgreSQL database, you must create all your database instances using the UTF8 character set.