This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Framework Database Connectivity Guide

Management Framework 8.5.1

12/30/2021

Table of Contents

Framework Database Connectivity Reference Guide	**3**
Overview	**4**
Environment Settings	**6**
General Recommendations	7
Oracle Databases	9
PostgreSQL Databases	17
IBM DB2 Databases	27
Microsoft SQL Server Databases	29
Database Access Points	**39**
FAQs and Troubleshooting	**42**
Document Change History	**46**

Framework Database Connectivity Reference Guide

This guide describes the concepts and procedures relevant to how Genesys software connects to databases.

Overview

Databases in Genesys

Database connectivity prior to release 8.5

Database Access Points

Databases in multi-language environments

Failure of database access functionality

Environment Variable Settings

General recommendations

Oracle databases

PostgreSQL databases

IBM DB2 databases

Microsoft SQL Server databases

Database Access Points

Configuring a DAP

Using DAPs

FAQs and Troubleshooting

Frequently Asked Questions

Troubleshooting

Overview

In Genesys software, data is stored in databases. Server applications, such as Configuration Server, manage this data by connecting to the database.

Databases in Genesys

In any Genesys environment, there is only one database (or a database cluster, if configured on the DBMS level) that is intended for use by Configuration Server – this is the Configuration Database.

In addition to the Configuration Database, there can be a number of other databases that Genesys servers and GUI applications may need to access. Applications access these databases directly, using information from Database Access Points (DAPs) to locate the database and obtain access credentials. In Management Framework, for example, Message Server accesses the Log Database through a DAP.

Connectivity Prior to Release 8.5

Prior to release 8.5, a DB Server Application object was used to access one or more databases. Release 8.5 effectively streamlines database access by removing DB Server from the access path.

However, if you still want to use DB Server in your configuration, or if you have legacy applications that cannot access the database directly, do the following:

1. For new applications only: Use the configuration option that controls how a database is accessed by the application, either using DB Server or direct database access capability, and set it to the appropriate value. In Configuration Server, for example, this option is called **dbthread**. Refer to application-specific documentation for the name and description of the option, if there is one, used by the particular application.
2. Follow the instructions and information in Framework 8.1 documentation to deploy and use DB Server.

Important

If you choose to use DB Server, be aware that you will be unable to access any of the new database-related features and functionality introduced in release 8.5. In addition, you will be able to use DB Server with only those Database Management Systems supported in 8.5; you will not be able to use it with any DBMS that is no longer supported.

Database Access Points

To provide an interface between applications in the Genesys installation and databases, the Configuration Layer uses the concept of a Database Access Point (DAP). If, according to your configuration, a database can be accessed by multiple servers simultaneously, register one DAP for each potential connection.

See Database Access Points for detailed information about installing DAPs.

Databases in Multi-language Environments

To be used in a multi-language environment, a database must be able to support data that can be encoding in different, or a common, format. Normally, this is done by encoding data using UTF-8.

The DBMS-specific sections of Environment Settings contain information about using the DBMS in multi-language environments; refer to these sections for more information and instructions.

Failure of Database Access Functionality in Genesys Servers

The Management Layer can detect internal failure of the database access module within Genesys applications if you configure the unresponsive process detection feature, and specify that it should detect application thread failures. See the *Framework Management Layer User's Guide* and the related option descriptions in the *Framework Configuration Options Reference Manual* for more information about setting up this feature for the particular application for which you want to monitor the thread.

Warning

Use this functionality with great care. Failure to use it properly could result in unexpected behavior, from ignoring the options to an unexpected restart of the application.

Environment Settings

To work with a particular DBMS, an application (the Genesys DB Client application) requires particular environment settings, as listed in the following sections:

- General recommendations
- Oracle databases
- Postgre databases
- IBM DB2 databases
- Microsoft SQL Server databases

General Recommendations

The recommendations in this section apply regardless of the type of DBMS that you are using.

DBMS Versions

32-bit or 64-bit

Make sure that you are installing DBMS vendor client software that matches the 32-bit or 64-bit Genesys software that you want to enable for database access.

For example, if your Genesys application is 64-bit, then make sure that it can access 64-bit DBMS client software, as provided by your database vendor.

Client Software Version

Make sure you are using the correct version of DBMS client software, as given in the following sections, for each type of database. All Genesys applications of a particular release use the same version of DBMS client software. Even if you are accessing a database of different versions, you might still need to have another version of DBMS client software on the host where the Genesys application is installed.

Genesys supports multiple versions of a DBMS using the same version of DBMS client software. For example, you need Oracle 11g client software to access both Oracle 11 and Oracle 12 databases.

Restriction for Large Data

In MSSQL, Oracle, DB2 databases, data retrieval from columns of character and binary data types (for example, CLOB and BLOB data types) that can handle data that is greater than 10 MB in size, is limited to a maximum of 10 MB; larger sizes are not supported. Genesys strongly recommends that applications insert data smaller than 10 MB into columns of these data types.

DBMS Encoding

Make sure encoding on DBMS is set correctly to match the encoding being used in the Genesys environment. If you are using single-language (in addition to English-US), you must create your database with the respective encoding, as described in the following DBMS-specific pages of this section. If you are using a multi-language environment, UTF-8 encoding should be used on the DBMS, also as described in the DBMS-specific sections.

Important

You will not be able to use some Genesys applications in multi-language mode.

Database Failures

Starting in release 8.0, a database client process can detect a connection failure with the corresponding database and attempt to reconnect. To detect the failure, database clients monitor the responses they receive from the DBMS. If a response is not received within the interval specified by the configuration option **db-request-timeout**, the client process stops executing. This is understood to be a failure of the DBMS.

The option **db-request-timeout** is configured in the **Query Timeout** field of Database Access Point (DAP) Application objects and stored in the DAP's annex. The timeout set in the DAP overrides the timeout set in the database client application, but applies only to client processes that connect to the database through this DAP.

See Creating a DAP for more information about how to configure this timeout for Log Database access.

Oracle Databases

You must have Oracle Client software accessible in the environment where the Genesys application is running. Genesys uses Oracle 11g client software to access all supported versions of Oracle.

Using Full Installation of Oracle Client

Connectivity to an Oracle database relies on TCP/IP to work between an Oracle server and its client.

You must set the following environment variables for DB client for Oracle:
ORACLE_HOME
ORACLE_SID

In addition, you must specify the full path to the **bin** of the Oracle home directory in the **PATH** variable, and either **LIBPATH** for AIX or **LD_LIBRARY_PATH** for Linux and Solaris, depending on the platform you are using. If the DBMS client for Oracle runs on a different host other than the Oracle server, you must also configure the **SQLNet** file on both hosts. Note that a TCP/IP Adapter is also required. For more information, refer to the Oracle documentation for your platform.

If you are using the UNIX/Linux platform, you might want to create a link to the DB client so that the server accessing the database can find the library using the default name. Enter the following on the operating system command line:

```
ln -s libclntsh.so.11.1 libclntsh.so
```

You must use the **tnsnames.ora** file to specify database access as defined in Oracle documentation. Genesys DB Client will load the **.tns** file.

The following is an example of TNS name content:

```
CMES =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST =<your oracle host>)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = cmes)
    )
  )
```

With this definition in the **.tns** file, the parameters of the Genesys Database Access Point can be set as follows:

```
dbengine=oracle
dbserver=CMES
dbname=
username=<oracle schema user>
password=<oracle schema user's password>
```

Using Oracle Instant Client

Instead of a full Oracle client installation, you can use the Instant Client package from Oracle, downloadable from here.

For core Genesys servers to work, you must have the Basic package. If you have to perform conversion of character set encodings for languages other than English, make sure that the Instant Client package contains all necessary encoding tables. If you cannot find the proper package of Instant Client, you may want to use a full Oracle installation.

Connectivity to an Oracle database relies on TCP/IP to work between the Oracle server and client. You do not need to set up the **ORACLE_HOME** or **ORACLE_SID** environment variables. However, you must specify the full path to the folder where you put the Oracle Instant Client in the appropriate environment variable—**PATH** for Windows, **LIBPATH** for AIX, or **LD_LIBRARY_PATH** for Linux and Solaris.

For example, the configuration of Genesys Database Access Points without using TNS should be as follows:

```
dbengine=oracle
dbserver=<oracle host>:1521/<oracle service name>
dbname=
username=<oracle schema user>
password=<oracle schema user's password>
```

You can still use TNS-based connection information with Instant Client if you set up the **TNS_ADMIN** environment variable, or you can use the SQL Connect URL string `<host>[:port]/<service name>`. Both are described in Oracle Instant Client documentation.

Service Name

Instant Client requires a Service Name, rather than the SID required by the full Oracle installation package. The Service Name can be found in the **tnsnames.ora** file, which is found in the **<oracle home>/network/admin** folder on the server. For example, in the following excerpt from the **tnsnames.ora** file, the SID appears below ORCL, and the Service Name is `orcl.us.int.genesyslab.com`.

```
ORCL =
 (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = <IP of a host>)(PORT = 1521))
  (CONNECT_DATA =
   (SERVER = DEDICATED)
   (SERVICE_NAME = orcl.us.int.genesyslab.com)
  )
 )
```

Secure Communications with Oracle DBMS

You can use Transport Layer Security (TLS) to secure connections with an Oracle database.

Warning

You must be using Oracle Full Client if you want to configure secure connections. Do not configure TLS/SSL support if you are using Oracle Instant Client.

Can Secure Connections be Set?

Before you can configure secure connections, you have to determine if the Oracle **tnsnames.ora** file can be set to enforce TLS/SSL. To do this, open the **tnsnames.ora** file and look for the **SECURITY** section and its accompanying parameter **SSL_SERVER_CERT_DN** under DESCRIPTION. It will look something like this:

```
<net_service_name>=
 (DESCRIPTION=
   (ADDRESS=...)
   (ADDRESS=...)
   (CONNECT_DATA=
    (SERVICE_NAME=...))
   (SECURITY=
    (SSL_SERVER_CERT_DN=...)))
```

If this section and its parameter is present, you can configure secure connections.

For more information about the **tnsnames.ora** file and its parameters, refer to the tnsnames.ora section of the *Oracle Database Net Services Reference*.

Configure the Secure Connections

Secure connections using TLS with the Oracle database are configured in the **.tns** file, not in or by the DAP or Configuration Server configuration files. However, these files contain the **net_service_name** used to locate the TNS listener definition in the client configuration.

To configure the secure connections, do the following:

1. In the Oracle server, open the **listener.ora** file and configure the following parameters:
   - **SID_LIST_LISTENER**
   - **WALLET_LOCATION**
   - **LISTENER**

   For example:

```
# listener.ora Network Configuration File: /opt/oracle/app/oracle/product/11.2.0/dbhome_1/
 network/admin/listener.ora
 # Generated by Oracle configuration tools.

 SID_LIST_LISTENER =
   (SID_LIST =
     (SID_DESC =
       (GLOBAL_DBNAME = db01)
       (ORACLE_HOME = /opt/oracle/app/oracle/product/11.2.0/dbhome_1)
       (SID_NAME = db01)
     )
```

```
    )

  SSL_CLIENT_AUTHENTICATION = FALSE

  WALLET_LOCATION =
    (SOURCE =
      (METHOD = FILE)
      (METHOD_DATA =
        (DIRECTORY = /opt/oracle/app/oracle/product/11.2.0/dbhome_1/owm/wallets/oracle)
      )
    )

  LISTENER =
    (DESCRIPTION_LIST =
      (DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCP)(HOST = vce-u0047.us.int.genesyslab.com)(PORT = 1521))
      )
      (DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCPS)(HOST = vce-u0047.us.int.genesyslab.com)(PORT = 2484))
      )
    )

  ADR_BASE_LISTENER = /opt/oracle/app/oracle
```

For more information about the **listener.ora** file and its parameters, refer to the Listener Control Utility section of the *Oracle Database Net Services Reference*.

2. In the Oracle client, open the **tnsnames.ora** file and configure the following:
   - The Oracle secure port **2484** to use the secure protocol **TCPS**.
   - The **SECURITY** section.

   For example:

```
DB_GENCHNGC3031_SSL=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS= (PROTOCOL = TCPS)(HOST = GEN-CHN-GC3-031)(PORT = 2484))
    (CONNECT_DATA=
      (SERVER= DEDICATED)
      (SERVICE_NAME= GENCHNGC3031.genesys.lab))
    (SECURITY =
      (SSL_SERVER_CERT_DN="CN=GENCHNGC3031.genesys.lab,C=US")))
```

   For more information about the **tnsnames.ora** file and its parameters, refer to the tnsnames.ora section of the *Oracle Database Net Services Reference*.

3. In the Oracle client, open the **sqlnet.ora** file and configure the following parameters:
   - **SQLNET.AUTHENTICATION_SERVICES**
   - **SSL_SERVER_DN_MATCH**
   - **WALLET_LOCATION**

   For example:

```
SQLNET.AUTHENTICATION_SERVICES= (BEQ, TCPS)

 SSL_SERVER_DN_MATCH= ON

 WALLET_LOCATION=
   (SOURCE =
        (METHOD = FILE)
        (METHOD_DATA =
          (DIRECTORY = C:\app\wallet)))
```

For more information about the **sqlnet.ora** file and its parameters, refer to the sqlnet.ora section of the *Oracle Database Net Services Reference*.

Additional Information

For additional information about using TLS with Oracle databases, refer to the *Oracle Database Advanced Security Administrator's Guide*, and *Configuring Secure Sockets Layer Authentication* in particular.

Using Oracle Database with National Languages

Single Language Deployment

You must create all Oracle databases using the same character set, such as WE8MSWIN1252. You must select an encoding that matches Microsoft Windows Operating System default encoding for a selected language, so that applications, like Interaction Routing Designer, display data correctly.

On every host that has a Genesys application accessing an Oracle database, make sure that the **NLS_LANG** environment variable is set to match the language and character encoding of data in the Oracle database, as defined in the following table . For example, **NLS_LANG=**AMERICAN_AMERICA.WE8MSWIN1252.

[+] Show table

Operating System Locale	NLS_LANG Environment Variable Value
Arabic (U.A.E.)	ARABIC_UNITED ARAB EMIRATES.AR8MSWIN1256
Bulgarian	BULGARIAN_BULGARIA.CL8MSWIN1251
Catalan	CATALAN_CATALONIA.WE8MSWIN1252
Chinese (PRC)	SIMPLIFIED CHINESE_CHINA.ZHS16GBK
Chinese (Taiwan)	TRADITIONAL CHINESE_TAIWAN.ZHT16MSWIN950
Croatian	CROATIAN_CROATIA.EE8MSWIN1250
Czech	CZECH_CZECH REPUBLIC.EE8MSWIN1250
Danish	DANISH_DENMARK.WE8MSWIN1252
Dutch (Netherlands)	DUTCH_THE NETHERLANDS.WE8MSWIN1252
English (United Kingdom)	ENGLISH_UNITED KINGDOM.WE8MSWIN1252
English (United States)	AMERICAN_AMERICA.WE8MSWIN1252
Estonian	ESTONIAN_ESTONIA.BLT8MSWIN1257
Finnish	FINNISH_FINLAND.WE8MSWIN1252
French (Canada)	CANADIAN FRENCH_CANADA.WE8MSWIN1252
French (France)	FRENCH_FRANCE.WE8MSWIN1252

Operating System Locale	NLS_LANG Environment Variable Value
German (Germany)	GERMAN_GERMANY.WE8MSWIN1252
Greek	GREEK_GREECE.EL8MSWIN1253
Hebrew	HEBREW_ISRAEL.IW8MSWIN1255
Hungarian	HUNGARIAN_HUNGARY.EE8MSWIN1250
Icelandic	ICELANDIC_ICELAND.WE8MSWIN1252
Indonesian	INDONESIAN_INDONESIA.WE8MSWIN1252
Italian (Italy)	ITALIAN_ITALY.WE8MSWIN1252
Japanese	JAPANESE_JAPAN.JA16SJIS
Korean	KOREAN_KOREA.KO16MSWIN949
Latvian	LATVIAN_LATVIA.BLT8MSWIN1257
Lithuanian	LITHUANIAN_LITHUANIA.BLT8MSWIN1257
Norwegian	NORWEGIAN_NORWAY.WE8MSWIN1252
Polish	POLISH_POLAND.EE8MSWIN1250
Portuguese (Brazil)	BRAZILIAN PORTUGUESE_BRAZIL.WE8MSWIN1252
Portuguese (Portugal)	PORTUGUESE_PORTUGAL.WE8MSWIN1252
Romanian	ROMANIAN_ROMANIA.EE8MSWIN1250
Russian	RUSSIAN_CIS.CL8MSWIN1251
Slovak	SLOVAK_SLOVAKIA.EE8MSWIN1250
Spanish (Spain)	SPANISH_SPAIN.WE8MSWIN1252
Swedish	SWEDISH_SWEDEN.WE8MSWIN1252
Thai	THAI_THAILAND.TH8TISASCII
Spanish (Mexico)	MEXICAN SPANISH_MEXICO.WE8MSWIN1252
Spanish (Venezuela)	LATIN AMERICAN SPANISH_VENEZUELA.WE8MSWIN1252
Turkish	TURKISH_TURKEY.TR8MSWIN1254
Ukrainian	UKRAINIAN_UKRAINE.CL8MSWIN1251
Vietnamese	VIETNAMESE_VIETNAM.VN8MSWIN1258

For more information, see the Oracle documentation here.

If you are unable to setup MS Windows compatible character encoding when creating the Oracle database, make sure that the Oracle client software on all hosts with Genesys applications has been set to use character encoding that matches the target as close as possible, by following these steps:

1. Set up **NLS_LANG** to use the closest compatible encoding. For example, `WE8ISO8858P1` to match `WE8MSWIN1252` if you are using Linux to host Genesys applications that should access Oracle Databases.
2. Make sure that Oracle client software contains NLS tables allowing conversion between character encoding of the database and the host. Refer to Oracle documentation for more information about supported character conversions.

With your 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.

Multi-Language Deployment

To enable storage and processing of data in multiple languages using Oracle Databases, you must create all your database instances using the AL32UTF8 character set. For example:

```
CREATE DATABASE orclutf8
...
  CHARACTER SET AL32UTF8
  NATIONAL CHARACTER SET AL16UTF16
```

On every host that has a Genesys application accessing an Oracle database, make sure the **NLS_LANG** environment variable is set to match the character encoding of data in the Oracle database; for example, **NLS_LANG=.**UTF8.

Using Oracle TAF

Genesys supports using the Oracle Real Application Cluster to provide redundant access to database storage. You must use the TNS file to define cluster access, as specified in Oracle documentation.

For example:

```
LORAC =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST =<oracle RAC listener> )(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME =<oracle service name>)
      (FAILOVER_MODE =
        (TYPE = SELECT)
        (METHOD=BASIC)
        (RETRIES = 2)
        (DELAY = 1)
      )
    )
  )
```

The **RETRIES** and **DELAY** parameters might affect how long the Genesys application will wait for Oracle to respond before it attempts to reconnect. Refer to the Oracle TAF guide for more details.

Failure of an Oracle 12g RAC Database

The Oracle 12g Real Application Cluster (RAC) DBMS includes a client-side feature called Transparent Application Failover (TAF). If an instance of a database fails, TAF automatically reconnects to a surviving database instance (node).

However, TAF only restores the connection; it is the responsibility of the application to restart on the new node any operations that were in process on the failed node. These operations could be any of the following:

- Individual Data Manipulation Language (DML) statements, such as INSERT, UPDATE, and DELETE.

- Active transaction involving DML statements, issuing ROLLBACK instructions to these transactions first.
- Active binding packages.

To support Oracle 12g RAC in TAF mode, Configuration Server can optionally resubmit DML statements (DML transactions or binding package execution) when the appropriate error messages are received from the DBMS. This is implemented using the configuration option **dml-retry**. Refer to the *Framework Configuration Options Reference Manual* for the full description of this option.

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.

IBM DB2 Databases

You must make DB2 client software accessible in the environment where the Genesys application is running. Genesys uses IBM DB2 9.7 client software to access all supported versions of DB2.

Using IBM DB2 Client Software

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).

You must set the **DB2INSTANCE** and **INSTHOME** environment variables for the DB client for DB2. For more information, refer to DB2 documentation.

Using DB2 Databases with National Languages

Single Language Deployment

You must create all DB2 databases using the same character set, such as `ISO-8859-1`, as provided in the following table. You must select an encoding that matches Microsoft Windows Operating System default encoding for the selected language so applications, like Interaction Routing Designer, display data correctly.

[+] Show table

Category	Encoding	Codepage
ASCII	iso-8859-1	819
	ibm-1252	1252
	iso-8859-2	912
	iso-8859-5	915
	iso-8859-6	1089
	iso-8859-7	813
	iso-8859-8	916
	iso-8859-9	920
MBCS	gb2312	1386
	ibm-932, shift_jis78	932
	Shift_JIS	943
	IBM-eucCN	1383
	ibm-1388	1388
	IBM-eucJP, EUC-JP	954, 33722

Category	Encoding	Codepage
	ibm-930	930
	ibm-939	939
	ibm-1390	1390
	ibm-1399	1399
	ibm-5026	5026
	ibm-5035	5035
	euc-tw, IBM-eucTW	964
	ibm-937	937
	euc-kr, IBM-eucKR	970
	big5	950

For more information about encoding and DB2 codepages, see IBM documentation here.

Multiple Languages Deployment

For a DB2 database to store multiple languages, it must be created using the UTF-8 codeset. In addition, the DB2 server must be configured with a bufferpool of at least 8KB page size, and a tablespace that uses that bufferpool. Use the following commands:

```
db2 create bufferpool bp8k pagesize 8K
db2 create tablespace data pagesize 8K bufferpool bp8K
```

On every host running a Genesys application that accesses a DB2 multi-language database, set the environment variable **DB2CODEPAGE=**1208.

Microsoft SQL Server Databases

You must install software to access the version of Microsoft SQL Server you are using. Refer to Microsoft documentation for details. You can use any edition of Microsoft SQL Server, including Express.

Using Microsoft Client Software

Genesys uses TCP/IP as a way to access Microsoft SQL Server. When installing Microsoft SQL Server and/ or Microsoft client software, make sure that Server and Client are using TCP/IP. Dynamic ports are not supported; you must configure the server to listen on a fixed port (1433).

You can access default instances or named instance (including Express) of Microsoft SQL Server. To use a default instance, set the following parameters of the Database Access Point:

```
dbengine = mssql
dbserver = <sql server host>
dbname = <database name>
username = <user>
password = <password>
```

If a named (non-default) instance is used, the **dbserver** parameter must be specified in the format:
`dbserver = <sql server host>\<named instance>`

Or for the Microsoft SQL Express edition:
`dbserver = <sql server host>\sqlexpress`

Notes for Management Framework Components

- The MSSQL connection is made using ODBC, by default. In legacy environments, the connection can be made using the MSSQL 2005 Server Native Client driver, if it is installed.
- To work with MS SQL databases, Configuration Server and Message Server require Microsoft Data Access Components (MDAC) version 2.8 or later.
- For MS SQL databases, DB Server did not correctly read international characters that were written to the database if both of the following conditions existed:
  - The records were originally written using DB Server 7.2 or earlier.
  - On the host on which DB Server was running, the option **SQL Server Client Network Utility > DB-Library Options > Automatic ANSI to OEM** conversion was turned on.

Windows Authentication with MS SQL Server

Windows Authentication provides a more secure way for an Application to access an MS SQL database without storing the database password in the Genesys configuration. Windows Authentication uses the Kerberos security protocol, enforces password policies to ensure strong passwords, and supports account lockout and password expiration. A connection made using Windows Authentication is sometimes called a *trusted* connection, because SQL Server trusts the credentials provided by Windows.

This section describes how to enable and configure Windows Authentication with MS SQL Server for Genesys applications that support it.

Enabling a Windows Process to Utilize Windows Authentication on the MS SQL Server

For an application to use Windows Authentication to access an MS SQL database, the Windows account under which the application runs must have both of the following:

- Login access to the MS SQL Server.
- Appropriate access to the MS SQL database that the application will use.

To verify that both exist:

1. Start MS SQL Server Management Studio.
2. In the **Connect to Server** dialog box, specify an MS SQL Server name and administrator credentials to connect to the MS SQL server.
3. In **Object Explorer**, expand the entry for the MS SQL Server identified in the previous step, then **Security**, then **Logins**. The Logins folder should contain an entry for either the Windows account itself, or the group to which that account belongs; for example, **<Domain name>\Administrators**.
4. To determine if the Windows account is either directly mapped to the database, or has administrative access to all databases, right-click the user's Login to open the **Properties** dialog box and select **Server Roles**. Then do one of the following:
    - If **sysadmin** is checked in the **Server Roles** list, this Windows account has access to all databases.
    - If **sysadmin** is not checked, click **User Mapping** to see if this Windows account is mapped to the appropriate database as **db_owner**.

If an appropriate Login does not exist, and/or the Login does not have access to the database, do the following steps, as appropriate:

1. Start MS SQL Server Management Studio.
2. In the **Connect to Server** dialog box, specify an MS SQL Server name and administrator credentials to connect to the MS SQL server.
3. In **Object Explorer**, expand the entry for the MS SQL Server identified in the previous step, then **Security**, then **Logins**.
4. Click **New Login**. The **Login-New** dialog box opens.
5. In the **Login name** field, enter the user name of the Windows account in the format `<domain>\<username>`.This creates the new Login.
6. In **Object Explorer**, configure access to the appropriate database, as follows:

   a. Select **User Mapping** in the left panel.
   b. In the upper half of the right side, select the appropriate database. The name of the Login you just created appears in the **User** field.
   c. In the **Database role membership for:** list, select **db_owner**.
7. Click **OK**.

After a Windows account has a Login and is associated with a database, anyone using that account can log in to the MS SQL Server without specifying a username or password.

If the application that connects with the database has been installed as a Windows Service, by default it is started under a Local System account with a user name of **NT AUTHORITY\SYSTEM**, in the group **BUILTIN\Administrators**. But this user account has no access permissions to the database, so the user account from which the service gets started needs to be changed.

Do one of the following to change the user account of the service:

- In the Computer Management/Services console, right-click the service and navigate to **Properties** > **Log On** tab > **This Account**, and enter a Windows username and password that has permission to connect to the MS SQL Server and access the database.
- Run the following command to change the user account:

```
sc.exe config <service name> obj= <.\user account name> password= <user account password>
```

You must also change the user account of the **Local Control Agent** service, so that it is able to start the application under a non-default Windows account.

Configuring Applications to use Windows Authentication when Accessing MS SQL Server

If an Application is using DB Server to access a database, it must be using DB Client 8.5.1 or higher. In addition, a Windows process for DB Server must be set up as described above. DB Server must then be set in the DAP.

If an Application is accessing the database directly (without DB Server), a DAP is required without access to DB Server. A Windows process for the Application itself must also be set up as described above.

After a Windows process and MS SQL Server have both been enabled to use Windows Authentication, you can force Genesys applications to connect to the database using Windows Authentication by using either a *Trusted User* or a *Data Source Name*.

Trusted User

For an application to use Windows Authentication, it must be provisioned with `username=trusted` in its configuration, where `trusted` is a keyword. The password field is not used in this case, and can be left empty.
Refer to the configuration options of the particular application to determine if it supports Windows

Authentication and where, in its configuration, to enter the database user name.
Example: Message Server with Direct Connection to Database To configure Message Server that connects directly to the Log Database (the default configuration), configure the options that describe the Log Database in the Database Access Point of the MS SQL Log Database, as follows:

- In the **DB Info** section of the **Configuration** tab, enter `trusted` in the **User Name** field.

DSN

To set up Windows Authentication using a DSN, you must first open and configure an ODBC Data Source using Microsoft Windows NT Authentication, as follows:

1. Open a Data Source Administrator by following the steps here for your particular version of Windows.

> **Tip**
> There might be two **Data Sources (ODBC)** available, one for 32-bit and another for 64-bit. Select one according to the type of Genesys application.

2. On the System **DSN** tab, click **Add** to add a system data source.
3. Select one of the following drivers, as appropriate:
   - MS SQL Server (recommended)
   - MS SQL Native Client
   - MS SQL Server Native Client
4. Click **Finish**.
5. Follow the instructions here to configure the DSN to be used to connect to the database.

> **Important**
> Use the System DSN to configure only Windows Authentication.

6. Click **Finish**. The application displays a summary page.
7. Click `Test Data Source` to run a test connection and ensure that your configuration is valid. If the test is successful, the `Test Results` are displayed.

After the Data Source is set up, you can then enable the MS SQL DB Client to support it.

For an application to use Windows Authentication using DSN, it must be provisioned with `DBMS Name=dsn` and `Database Name=<dsn name>` in its configuration, where `dsn` is a keyword and `<dsn name>` is the name of DSN you configured in the previous step. The `username` option is not required, and can be set to any name or password, or can be left empty.

Refer to the configuration options of the particular application to determine if it supports Windows Authentication and where, in its configuration, to enter the name of the database and DBMS.

Example: Message Server with Direct Connection to Database

To configure Message Server that connects directly to the Log Database (the default configuration), configure the options that describe the Log Database in the Database Access Point of the MS SQL Log Database, as follows:

- In the **DB Info** section of the **Configuration** tab, enter `dsn` and the name of the DSN in the **DBMS Name** and **Database Name** fields, respectively.

Example: Message Server using DB Server 8.1.3 to Connect to Database

To configure Message Server that connects to the Log Database using DB Server 8.1.3, configure the option that disables the direct connection, as follows:

```
...
[messages]
...
dbthread=false
...
```

Configure the options that describe the DB Server connection, and the Log Database, in the Database Access Point of the MS SQL Log Database, as follows:

- In the **DB Info** section of the **Configuration** tab, enter the following information:
  - **DBMS Name**—`dsn`
  - **Database Name**—DSN Name

Configure DB Server 8.1.3 using `dbclient_851`. If DB Server is started as a service, the user account of the service must be modified so it has access to the Configuration Database.

Encrypting Communications with Microsoft SQL DBMS

In addition to using Windows Authentication with an MS SQL Server you can also force Genesys components to use a secure connection to MSSQL by configuring MS SQL server to accept only encrypted connections, based on the certificate added to the server.

To configure the MS SQL Server to accept encrypted connections, you must add a certificate with a fully qualified computer domain name to MS Certificate Storage on the server side. Add the certificate to the **Personal** folder and the Trusted CA to the **Trusted Root Certification Authorities** folder, both in the Local Computer account. Use Microsoft Management Console (mmc) to manage certificates.

To configure the server:

1. In MS SQL Server Configuration Manager, expand **SQL Server Network Configuration**, right-click **Protocols for <server instance>**, and select **Properties** from the drop-down menu.

2. On the **Certificate** tab, select the desired certificate from the **Certificate** drop-down menu, and click **OK**.
3. On the **Flags** tab, select Yes in the **ForceEncryption** box, and click **OK** to close the dialog box.
4. Restart the SQL Server service.

After you add the certificate, all client connections with this server will be encrypted.

Using Microsoft SQL Server Databases with National Languages

Single Language Deployment

No special configuration or other preparations are needed to use Genesys applications in single language mode with Microsoft SQL Server databases. The databases themselves must be created with target language and default encoding, as given in the following table:

[+] Show table

Sort Order ID	SQL Server Collation Came
30	SQL_Latin1_General_Cp437_BIN
31	SQL_Latin1_General_Cp437_CS_AS
32	SQL_Latin1_General_Cp437_CI_AS
33	SQL_Latin1_General_Pref_CP437_CI_AS
34	SQL_Latin1_General_Cp437_CI_AI
40	SQL_Latin1_General_Cp850_BIN
41	SQL_Latin1_General_Cp850_CS_AS
42	SQL_Latin1_General_Cp850_CI_AS
43	SQL_Latin1_General_Pref_CP850_CI_AS
44	SQL_Latin1_General_Cp850_CI_AI
49	SQL_1Xcompat_CP850_CI_AS
50	Latin1_General_BIN
51	SQL_Latin1_General_Cp1_CS_AS
52	SQL_Latin1_General_Cp1_CI_AS
53	SQL_Latin1_General_Pref_CP1_CI_AS
54	SQL_Latin1_General_Cp1_CI_AI
55	SQL_AltDiction_Cp850_CS_AS
56	SQL_AltDiction_Pref_CP850_CI_AS
57	SQL_AltDiction_Cp850_CI_AI
58	SQL_Scandinavian_Pref_Cp850_CI_AS
59	SQL_Scandinavian_Cp850_CS_AS
60	SQL_Scandinavian_Cp850_CI_AS

Sort Order ID	SQL Server Collation Came
61	SQL_AltDiction_Cp850_CI_AS
71	Latin1_General_CS_AS
72	Latin1_General_CI_AS
73	Danish_Norwegian_CS_AS
74	Finnish_Swedish_CS_AS
75	Icelandic_CS_AS
80	Hungarian_BIN (or Albanian_BIN, Czech_BIN, and so on) See Note
81	SQL_Latin1_General_Cp1250_CS_AS
82	SQL_Latin1_General_Cp1250_CI_AS
83	SQL_Czech_Cp1250_CS_AS
84	SQL_Czech_Cp1250_CI_AS
85	SQL_Hungarian_Cp1250_CS_AS
86	SQL_Hungarian_Cp1250_CI_AS
87	SQL_Polish_Cp1250_CS_AS
88	SQL_Polish_Cp1250_CI_AS
89	SQL_Romanian_Cp1250_CS_AS
90	SQL_Romanian_Cp1250_CI_AS
91	SQL_Croatian_Cp1250_CS_AS
92	SQL_Croatian_Cp1250_CI_AS
93	SQL_Slovak_Cp1250_CS_AS
94	SQL_Slovak_Cp1250_CI_AS
95	SQL_Slovenian_Cp1250_CS_AS
96	SQL_Slovenian_Cp1250_CI_AS
104	Cyrillic_General_BIN (or Ukrainian_BIN, Macedonian_FYROM_90_BIN)
105	SQL_Latin1_General_Cp1251_CS_AS
106	SQL_Latin1_General_Cp1251_CI_AS
107	SQL_Ukrainian_Cp1251_CS_AS
108	SQL_Ukrainian_Cp1251_CI_AS
112	Greek_BIN
113	SQL_Latin1_General_Cp1253_CS_AS
114	SQL_Latin1_General_Cp1253_CI_AS
120	SQL_MixDiction_Cp1253_CS_AS
121	SQL_AltDiction_Cp1253_CS_AS
124	SQL_Latin1_General_Cp1253_CI_AI

Sort Order ID	SQL Server Collation Came
128	Turkish_BIN
129	SQL_Latin1_General_Cp1254_CS_AS
130	SQL_Latin1_General_Cp1254_CI_AS
136	Hebrew_BIN
137	SQL_Latin1_General_Cp1255_CS_AS
138	SQL_Latin1_General_Cp1255_CI_AS
144	Arabic_BIN
145	SQL_Latin1_General_Cp1256_CS_AS
146	SQL_Latin1_General_Cp1256_CI_AS
153	SQL_Latin1_General_Cp1257_CS_AS
154	SQL_Latin1_General_Cp1257_CI_AS
155	SQL_Estonian_Cp1257_CS_AS
156	SQL_Estonian_Cp1257_CI_AS
157	SQL_Latvian_Cp1257_CS_AS
158	SQL_Latvian_Cp1257_CI_AS
159	SQL_Lithuanian_Cp1257_CS_AS
160	SQL_Lithuanian_Cp1257_CI_AS
183	SQL_Danish_Pref_Cp1_CI_AS
184	SQL_SwedishPhone_Pref_Cp1_CI_AS
185	SQL_SwedishStd_Pref_Cp1_CI_AS
186	SQL_Icelandic_Pref_Cp1_CI_AS
192	Japanese_BIN
193	Japanese_CI_AS
194	Korean_Wansung_BIN
195	Korean_Wansung_CI_AS
196	Chinese_Taiwan_Stroke_BIN
197	Chinese_Taiwan_Stroke_CI_AS
198	Chinese_PRC_BIN
199	Chinese_PRC_CI_AS
200	Japanese_CS_AS
201	Korean_Wansung_CS_AS
202	Chinese_Taiwan_Stroke_CS_AS
203	Chinese_PRC_CS_AS
204	Thai_BIN
205	Thai_CI_AS
206	Thai_CS_AS
210	SQL_EBCDIC037_CP1_CS_AS

Sort Order ID	SQL Server Collation Came
211	SQL_EBCDIC273_CP1_CS_AS
212	SQL_EBCDIC277_CP1_CS_AS
213	SQL_EBCDIC278_CP1_CS_AS
214	SQL_EBCDIC280_CP1_CS_AS
215	SQL_EBCDIC284_CP1_CS_AS
216	SQL_EBCDIC285_CP1_CS_AS
217	SQL_EBCDIC297_CP1_CS_AS

Important

For Sort Order ID 80, use any of the Window collations with the code page of 1250, and binary order. For example: Albanian_BIN, Croatian_BIN, Czech_BIN, Romanian_BIN, Slovak_BIN, Slovenian_BIN.

For more information, refer to Microsoft SQL documentation here.

Multiple Languages Deployment

To use Microsoft SQL to store data in multiple languages, the database tables must be able to store UNICODE characters (UCS-2 encoding).

When configuring a Database Access Point to access a multi-language database, you must specify **utf8-ucs2=**`true` in the **[dbclient]** section of the annex of the DAP.

Using MSSQL 2012 Always On Failover Cluster Instances (SQL Server)

Genesys supports MSSQL 2012 Always On Failover Cluster Instances (FCI), that uses Windows Server Failover Clustering (WSFC) to provide local high availability (HA) of redundant MSSQL databases at the server-instance level. Resources (databases) are grouped into a WSFC resource group, which is owned by a single WSFC node. Each FCI is an instance of an SQL Server and contains a set of WSFC nodes. When a failure occurs, the ownership of that resource group is switched to another WSFC node within the FCI. The switchover is done automatically and without any impact to the user.

For more information about FCI and WSFC, refer to Microsoft documentation at https://msdn.microsoft.com/en-us/library/ms189134(v=sql.110).aspx.

Failure of an MSSQL 2012 Cluster Database

There is no automatic resubmission for MSSQL. If the database fails, you must manually resubmit all failed write operations.

Database Access Points

To provide an interface between applications in the Genesys installation and databases to which the applications require access, the Configuration Layer uses the concept of a Database Access Point.

A Database Access Point (DAP) is an object of the Application type that describes both the parameters required for communication with a particular database and the parameters of the database itself. If, according to your configuration, a database can be accessed by multiple applications simultaneously, register one DAP for each possible connection.

Creating a DAP

To create a DAP, you do not have to install a DAP; you only need to configure it.

[+] Show steps

Prerequisites

- You are logged in to Genesys Administrator.
- The database to which the DAP is to provide access exists.

Start of procedure

1. In Genesys Administrator, create a new Application Template for the Database Access Point. Refer to the Additional Information > Generic Deployment and Login Procedures section of the *Framework Deployment Guide* for instructions.
2. Go to **Provisioning > Environment > Applications**, click **New**, and import the DAP template you just created.
3. In the **Browse** dialog box, select the DAP template file. The **Configuration** tab for the new DAP Application object appears in the Details panel.
4. In the **General** section, enter a descriptive name in the **Name** field; for example, `MyDAP`.
   A DAP can have the same name as the database itself. However, it is recommended that you make their names unique if you are using multiple access points for the same database.
5. In the **DB Info** section, provide the following information about the Database:
   - **Connection Type**—The type of connection to the DBMS.
   - **Query Timeout**—The period of time for which a database client process using this DAP expect a response from the DBMS. If the client process does not receive a response within this period, it stops executing. This is interpreted as a failure of the DBMS. The timeout set in this DAP overrides that set in the database client, but applies only to database client processes using this DAP. For more information about how DB Server uses this value, see Database Failure.
   - **DBMS Name**—The name or alias identifying the DBMS that handles the database, as follows:

    - For DB2, set this value to the name or alias-name of the database specified in the DB2 client configuration.
    - For Microsoft SQL, set this value to one of the following:
        - If you are not using Windows Authentication, the SQL server name (usually the same as the host name of the computer on which Microsoft SQL runs).

          Important

          For named instances of MS SQL server, it must be specified in the format:
          `<computer name>\<instance name>`

        - `dsn` if you are using Windows Authentication.
    - For Oracle, set this value to the name of your listener service as specified in the TNS file, or, if you are using Oracle Instant Client and do not have a TNS file, use the format `<oracle host>/<service name>`.
    - For PostgreSQL, set this value to the SQL server name (usually the same as the host name of the computer on which PostgreSQL runs).
- **DBMS Type**—The type of DBMS that handles the database. You must set a value for this property.
- **Database Name**—The name of the database to be accessed, as it is specified in the DBMS that handles this database. You must set a value for this property unless `oracle` or db2 is specified as the **DBMS Type**. For Microsoft SQL and PostgreSQL, this value is the name of the database where the client will connect.
- **User Name**—Set this to one of the following:
    - If you are not using Windows Authentication, the user name established in the SQL server to access the database.
    - If you are using Windows Authentication, set this to `trusted` or, if **DBMS Name** is set to `dsn`, set this to the name of the DSN.

  You must set a value for this property.
- **User Password**—Set this to one of the following:
    - If you are not using Windows Authentication, the password established in the SQL server to access the database.
    - If you are using Windows Authentication, you can leave this field blank or use a dummy password.
- **Re-enter Password**—Confirmation of the value entered for **User Password**.
- **Case Conversion**—Case conversion method for key names of key-value lists coming from the database client process. This value specifies whether and how a client application converts the field names of a database table when receiving data from the database client process. If you select `upper`, field names are converted into uppercase; if you select `lower`, field names are converted into lowercase; and if you select any, field names are not converted. This setting does not affect the values of key-value lists coming from the database client process. That is, actual data is being presented exactly as in the database tables.

Important

For the **Case Conversion** option, use the default value (any) unless directed to do otherwise by Genesys Customer Care.

- If the Log Database is an MS SQL database, and has been initialized for use in multi-language environments, select `UTF-8 for MSSQL`.

6. Click **Save** or **Apply** in the toolbar to save the new object. The new object will appear in the list of Applications.

End of procedure

Using DAPs

To interface an Application object with a database through a certain DAP, add the DAP to the list of the Application's connections.

Additional steps required to provide Windows Authentication are discussed in Windows Authentication with MS SQL Server.

DAP Configuration Options

Configuration options for DAP are set by values that you enter on the Configuration tab when creating a DAP object in Genesys Administrator. For more information about the options, refer to the *Framework Configuration Options Reference Manual*.

FAQs and Troubleshooting

This section contains answers to frequently asked questions about your database setup as well as some information about troubleshooting common problems. Look through this information before contacting Genesys Customer Care with your questions and problems.

Frequently Asked Questions

Question

If I change the port of my MS SQL server, how do I connect to my database?

Answer:
If you change the port of the MS SQL database, then you need to make the following changes in both the Client Network Utility and the Server Network Utility programs in MS SQL:

- On the **General** tab on the Client Network Utility, ensure that TCP/IP is enabled and click **Properties**. Enter the new port number and repeat this procedure on the Server Network Utility.
- Save the changes and your DB Client.

Question

What is the character limit for the stored procedure parameter?

Answer:
The maximum number of characters for processes are as follows:

- Parameter name: 64 bytes
- Number of parameters: 255
- Input/Output parameter limits: 2 KB
- The SQL statement passed from router to DB Server: 10240 bytes

---

Question

What causes the error Unresolved symbol: `getrlimit64(code)` in the DB Client's log?

Answer:
Generally, this error message is caused either by a mismatch between the DB Client and the OS version (for instance, running a 32-bit DB Client on a 64-bit OS), or by using an outdated DB Client patch.

Question

What should I do if the DB Client cannot connect to Oracle database and logs the error **ORA-12154** or **TNS-12154**?

Answer:
This error indicates that the Listener may be incorrectly configured in the **tnsnames.ora** file. Carefully check to ensure that the Listener is entered correctly in **tnsnames.ora**.

Question

What should I do if the DB Client cannot connect to an Oracle database and logs the error **ORA-12560**?

Answer:
To correct this problem, check that the **DBMS Name** parameter specified in the DAP Application object configured in the Configuration Layer is correct-it must match the name of the Listener configured in the **tnsnames.ora** file. Note that the **DBMS Name** parameter, although mandatory for all other database types, is not required for Oracle and thus can be left blank.

Question

What should I do if the DB Client cannot connect to Oracle database and records the error **ORA-12203** or **TNS-12203**?

Answer:
To correct this problem, verify that the option **dbserver** in the DB Client's configuration matches the setting of the **ORACLE_SID** environment variable on the computer on which the DB Client is running. It might also be helpful to analyze the **sqlnet.log** log file in the DB Client installation folder. This file is created by the Oracle Client in case there is a need to troubleshoot a network configuration problem.

Troubleshooting

This section discusses the solutions to several problems commonly encountered with DB Server.

Important

Set the value of the **verbose** option to 3 when you are troubleshooting a DB Client. This setting provides a detailed log that may indicate the source of a particular problem. After you resolve the problem, reset the verbose option to its previous level to increase performance.

Increase the Processes Allowed Limit in Windows

By default, Microsoft Windows limits each local system account to approximately 120 processes that are running as-or are started by-services. When the number of running processes reaches this limit, any attempt to start an additional process will fail. Because Windows counts DBClient processes, 120 can be too low a limit. One example of when this may happen is during Outbound deployment with a high number of calling lists.

To change the number of processes allowed, follow the procedure described in the Microsoft Knowledge Base article located here, which describes changing the registry value:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\SubSystems\
Windows
```

In this procedure, each additional 512 kilobytes that you specify in the third **SharedSection** parameter will allow approximately 120 additional processes/services to be started. The default **SharedSection** setting is highlighted in the default registry value data:

```
%SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows
SharedSection=1024,3072,512
Windows=On
SubSystemType=Windows
ServerDll=basesrv,1
ServerDll=winsrv:UserServerDllInitialization,3
ServerDll=winsrv:ConServerDllInitialization,2
ProfileControl=Off
MaxRequestThreads=16
```

Estimate the number of processes/services and apply it to the above calculation. For example:

- 65 services are running under the local system account.
- You plan to start DB Server as a service under the local system account.
- You plan to use 60 connections from Genesys DB Client applications to the DBMS.

These estimates require 125 services, which exceeds the default limit of 120.

In this example, Genesys recommends that you set the third **SharedSection** parameter to 1024, which increases the limit to 240. Use the Registry Editor to change the setting, as follows:

```
SharedSection=1024,3072,1024
```

Warning

Within the procedure, Microsoft presents a disclaimer about the Registry Editor: `If you use Registry Editor incorrectly, you may cause serious problems that may require you to reinstall your operating system. Microsoft cannot guarantee that you can solve problems that result from using Registry Editor incorrectly. Use Registry Editor at your own risk.`

DB Client Cannot Open the SQL Server Database

- Make sure the SQL Server client is properly installed and configured on the machine on which the DB Client is running (see the *Framework Deployment Guide* for more information).
- When DB Server tries to open the database, the following message appears in the log:

```
DBClient 96 - Server: username = 'user', password = '***********', database = 'test', DB Server = 'cti'
```

  Try to open the database manually (with the SQL Server client) using the values from the log message. If you can't do this, the problem lies with your SQL Server client configuration. In this case, contact your Database Administrator.

Request to Execute SQL Statement Fails

Check the log file for error messages. You may need to contact your Database Administrator for information on database-specific error codes. Check the log for the text of the SQL command that fails. Try to execute the same command manually with the SQL Server client, using the same username, password, and so on as specified in the Configuration Layer (configuration file).

Document Change History

This page provides a summary of changes to this document in release 8.5.1

New Pages

- This page, Document Change History

Updated Pages

- The section Secure Communications with PostgreSQL DBMS provides information and instructions for using TLS to configure secure connections to a PostgreSQL database.
- The section Windows Authentication with MS SQL Server provides information and instructions for configuring Applications to access MS SQL databases using Windows Authentication.
- The section Secure Communications with Oracle DBMS provides information and instructions for using TLS to configure secure connections to an Oracle database.
- The section Secure Communications with Microsoft SQL DBMS provides information and instructions for configuring the SQL server to accept encrypted connections, based on the certificate added to the server.