Framework Database Connectivity Guide

Microsoft SQL Server Databases

4/4/2026

Contents

Microsoft SQL Server Databases

You must install client 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.

Genesys uses ODBC client software to access all supported versions of MS SQL. On Windows, ODBC software is provided as part of the operating system setup. On the Linux operating system, a Genesys installation must use MS SQL ODBC 13.0 client software to access all supported versions of MS SQL. For the Windows platform, you can use the alternative version described in Using an Alternative Version of ODBC Client Software, below. Management Framework Components on Linux

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`

Management Framework Components on Windows

- The MS SQL connection is made using ODBC, by default. In legacy environments, the connection can be made using the MS SQL 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 64-bit platforms, Genesys provides an alternative version of the dbclient that can be used on Windows 2012. See Using an Alternative Version of ODBC Client Software.

Management Framework Components on Linux

On the Linux operating system, you must install and configure access to ODBC driver software

provided by Microsoft, and ensure that dbclient_msql_64 is available, as follows: On the hosts that will be using this component, install the database client software and make it available for the component. To support MS SQL 2017 Server on Linux, **odbcinst.ini** and **odbc.ini** files must be configured and copied to **/etc** directory regardless of their initial location. The sample configuration for ODBC driver version 13.1 is as follows:

odbcinst.ini

```
[SQL Server]
Description=Microsoft ODBC Driver 13 for SQL Server
Driver=/opt/microsoft/msodbcsql/lib64/libmsodbcsql-13.1.so.1.0
UsageCount=1
```

> ## Important
>
> - Genesys recommends that you use ODBC driver version 13.x. If you are not using version 13.1, replace 13.1.so.1.0 with the version you are using.
> - You must replace the default header that Microsoft puts during installation in **odbcinst.ini** to **[SQL Server]**.

odbc.ini

```
[MS SQL Server]
Driver=SQL Server
Description=<optional>My MS SQL Server
Trace=No
```

Example of the configuration of **odbcinst.ini** and **odbc.in**i files in case of MSSQL 2019:

odbcinst.ini

```
[SQL Server]
Description=Microsoft ODBC Driver 17 for SQL Server
Driver=/opt/microsoft/msodbcsql17/lib64/libmsodbcsql-17.5.so.2.1
UsageCount=1
```

odbc.ini

```
[ODBC Data Sources]
SQLSRV_DB01 = Local SQL instance
[SQLSRV_DB01]
Driver = SQL Server
Server = localhost
```

Starting from June 26, 2020:

- dbclient_msql in the default root folder will support loading "SQL Server Native Client 11.0" driver.
- dbclient_msql in the "dbclients_next" folder will support loading the "ODBC Driver 13 for SQL Server" driver.

Using an Alternative Version of ODBC 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 running on the Windows platform. Only 64-bit versions of alternate clients are provided. To use the alternative versions, take a backup of executables in the root installation folder and copy the provided executables into the root installation folder.

Secure Connections Using TLS v1.2

> **Important**
> This functionality is supported only on the Windows platform.

TLS version 1.2 is supported for MS SQL Server versions 2016, 2014, 2012, 2008, and 2008 R2. TLS v1.1 and TLS v1.2 are not supported for versions prior to Windows Server 2008 R2 and Windows 7.

To use TLS v1.2 for secure connections between MS SQL Server and clients, Microsoft provides updates that must be installed on both SQL server and client machines. Minimum requirements and required updates for installation are provided by Microsoft here.

TLS v1.2 is not enabled by default in Windows versions prior to Windows 8.1 and Windows Server 2012. Consult the table in the More Information section here to determine if TLS v1.2 is enabled by default in your system:

If TLS v.1.2 is not enabled by default in your Windows system, follow the instructions here to enable TLS v1.2 using registry entries.

After the SQL server and client machines are ready to support TLS v1.2, enable DB Server (if installed) to use TLS v1.2 as described in the following sections:

Using Windows Authentication

Genesys recommends that you upgrade to DB Server 8.1.301.13, which supports the Domain Source Name (DSN) for MS SQL connection using Windows Authentication. When creating DSN, select the installed driver that supports TLSv1.2.

For more information about creating and using DSN with Genesys components, see Configuring Applications to use Windows Authentication when Accessing MS SQL Server below.

Using SQL Authentication

As stated above, registry entries must be set, so that the drivers with TLS v1.2 support are used to connect with the MS SQL Server. DB Server supports only two drivers, SQL Native Client and SQL Server.

Using SQL Native Client Driver

SQL Native Client driver can be used only with legacy systems using MS SQL Server 2005 with the driver installed on it. Set the registry entries as follows:

1. Open the Windows command-line window (**Run...**) and enter `regedit`.
2. Go to **Computer\HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\SQL Native Client**.
3. Change the Key Driver path, which identifies the driver file used to connect with the MS SQL Server, to the Native Client driver installed for TLSv1.2 support. For example, **Driver=C:\Windows\system32\sqlncli11.dll**.

Using SQL Server Driver

Set the registry entries for the SQL Server Driver for connection with MS SQL Server, as follows:

1. Open the Windows command-line window (**Run...**) and enter `regedit`.
2. Go to `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\SQL Server`.
3. Change the Key Driver path, which identifies the driver file used to connect with the MS SQL Server, to the ODBC SQL Server driver installed to provide TLS v1.2 support. For example, **Driver=C:\Windows\system32\msodbcsql11.dll**.

Windows Authentication with MS SQL Server

> **Important**
>
> This functionality is available only on the Windows platform.

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* (DSN).

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.

SQL Server Authentication with MS SQL Server

Genesys applications support SQL Server Authentication when you connect to an SQL Server using the username and password entered in the DSN configuration.

This section describes how to enable and configure SQL Server Authentication with MS SQL Server.

Configuring Applications to use SQL Server Authentication when Accessing MS SQL Server

You can force Genesys applications to connect to the database using SQL Server Authentication by using a Data Source Name (DSN).

Creating DSN on Windows

To set up SQL Server Authentication using a DSN, you must first open and configure an ODBC Data Source using SQL Server 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 the **SQL Server** driver and click **Finish**.
4. Follow the instructions here to configure the DSN to be used to connect to the database.
   Make sure that **With SQL Server authentication using a login ID and password entered by the user** and **Connect to SQL Server to obtain default settings for the additional configurations options** options are selected on the second screen of the wizard.
5. Click **Finish**. The application displays a summary page.
6. 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.

You can now use the DSN in an application to connect to the MS SQL database.

Creating DSN on Linux

When connecting to the MS SQL database using a DSN, you only need to configure the **/etc/odbc.ini** file. Set the data source parameters in the **odbc.ini** file to create a DSN that specifies the connection information for the MS SQL database.

In the **odbc.ini** file, create a section that has a name for the DSN, and then specify the following parameters in the section.

```
[localdsn]  /* DSN Name */
Driver = SQL Server  /* As configured in /etc/odbcinst.ini */
Description = <description>
Server = <IP address of the host>
Database = <database name>
```

You can now use the DSN in an application to connect to the MS SQL database.

For an application to use SQL Server Authentication using DSN, it must be provisioned with the configuration options that describe the DB Server connection.

- For Configuration Server, update the following options in the Configuration Server configuration file.

  ```
  [dbserver]
  dbengine = mssql
  dbserver = dsn
  dbname = <DSN Name>
  username = <user>
  password = <password>
  ```

- For Message Server, configure the following options in the Database Access Point of the MS SQL Log Database.
  In the **DB Info** section of the **Configuration** tab, enter the following information:
  - **DBMS Name**—`dsn`
  - **DBMS Type**—`mssql`
  - **Database Name**—DSN Name
  - **User Name**—Username

- **Password**—Password

The username and password options are required and must map to a valid SQL Server user name and password that you configured in the previous step.

Encrypting Communications with Microsoft SQL DBMS

> **Important**
>
> This functionality is supported only on the Windows platform.

In addition to using Windows Authentication with an MS SQL Server you can also force Genesys components to use a secure connection to MS SQL 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:

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
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

Sort Order ID	SQL Server Collation Came
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
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

Sort Order ID	SQL Server Collation Came
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
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).

The following SQL Server collation name must be set for Multi-language support:

- Latin1_General_100_CI_AS_KS_WS_SC

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 MS SQL Always On Failover Cluster Instances (SQL Server)

MS SQL Always On with Single Subnet / Stretched LAN Listener

Genesys supports MS SQL 2012 (and later) Always On Failover Cluster Instances (FCI), that use Windows Server Failover Clustering (WSFC) to provide local high availability (HA) of redundant MS SQL 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.

MS SQL Always On with MultiSubnet Listener

This feature provides support for SQL Server 2016 (and later) Always On - MultiSubnetFailover for DBlibrary and DB Server. It is enabled by the Always On Availability Groups feature in SQL Server 2016 and later. Resources (databases) connect to an Availability Group Listener that maintains a list of the IP addresses associated with the host. If a connection is lost, the Availability Group Listener attempts to establish connections to all IP addresses in parallel. If a connection attempt succeeds, the driver discards any pending connection attempts.

This feature might cause some downtime during failover. The length of this downtime can vary, depending on the environment, network delay, and the time taken for switching between nodes.

Prerequisites

Genesys provides this functionality for components that operate on Windows and Linux. On Windows, MS SQL Server Native Client 11.0 driver or newer is required; on Linux, ODBC 13.1 and 17.1 are supported.

To utilize this feature, you must use versions of Genesys components that support this functionality when connecting to the database directly or using DB Server. If you are using DB Server, you must be

using version 8.1.302.02 or later; refer to the DB Server Release Notes for information about configuring DB Server to enable multisubnet failover. In all other cases, refer to the product documentation to confirm support and additional configuration requirements.

Basic Configuration

To configure this feature, use the **mssql-multisubnet** option in Configuration Server or in the DAP providing access to the Log Database. For detailed instructions, refer to MS SQL Cluster Database with MultiSubnet Listener for Framework Applications in the *Management Framework Deployment Guide*.

More information about the **mssql-multisubnet** option:

- This option can be set to Yes in only a multisubnet environment. It configures the SQL Server Native Client to provide faster detection and connection to a currently active server.
- Do not set this option to No in a single- or multisubnet environment. It will result in unexpected behavior, such as being unable to connect before or after failover (in a multisubnet environment). In either case, this is not recommended.
- When set to Yes, you must connect to an Availability Group Listener. Connecting to something other than an Availability Group Listener will result in a negative performance impact, and is not recommended.

> **Tip**
>
> When provisioning Genesys components that access a database directly, you might find out that there are several versions of **dbclient_msql** (the Genesys dbclient for MS SQL), with alternative versions contained in the **dbclient_next** subfolder, when installing an application. You must choose the version of **dbclient_msql** that is capable of using the MS SQL drivers listed in Prerequisites, and this might require additional configuration and/or installation steps for a component. Typically, you must back up the default version of **dbclient_msql** from the default installation folder and replace it with the **dbclient_msql** file from the **dbclient_next** folder and restart the application. Refer to component documentation for detailed instructions.

More information

For more information about this feature, refer to Microsoft documentation.

Failure of an MS SQL 2012 Cluster Database

There is no automatic resubmission for MS SQL 2012. If the database fails, you must manually resubmit all failed write operations. This does not apply to MS SQL 2014 and later.