APPX LIBRARY

SQL Server 2022 with APPX 6.2.1 on Rocky Linux

SQL Server 2022 (16.0.4305.5) · Rocky Linux 9.8 · APPX 6.2.1 (64-bit engine)

> Set up SQL Server as an APPX file system using FMS type 6. Start with one file, then save a record in APPX and check it in SQL Server.

APPX Software, Inc.

Made on November 18, 2026 from the pages of the APPX Library, https://wiki.appx.com/manuals/library/rdbms-sqlserver-2022-rocky-9-8-appx-621-x64/. The pages are the current version.

Contents

Chapter 1 **What to install** 3

Chapter 2 **Choose a database and create a login** 4

Chapter 3 **Set up the 64-bit System DSN** 5

Chapter 4 **Set up the FMS group and identity** 6

Chapter 5 **Assign one file and create the table** 8

Chapter 6 **Save and check a record** 10

Chapter 7 **Troubleshooting** 11

CHAPTER 1

What to install

This example was tested with the 64-bit APPX 6.2.1 engine on Rocky Linux 9.8, SQL Server 2022 Developer Edition build `16.0.4305.5`, unixODBC `2.3.12`, and Microsoft ODBC Driver `18.7.1.1`. APPX and SQL Server ran on the same host. If SQL Server runs elsewhere, install unixODBC and the ODBC driver on the APPX server and use the SQL Server host and port in the DSN.

Component	Download
APPX server	Install the 64-bit Linux engine. APPX 6.2.1 Linux server installer · Installation instructions
APPX Desktop Client	Use the 6.2.1 Desktop Client to open the APPX session. Linux Desktop Client installer · Other desktop platforms
SQL Server	Use SQL Server 2022. Developer Edition is for development and testing. If you already have an instance, check the engine version with `SERVERPROPERTY('ProductVersion')`. SQL Server 2022 Linux installation · SQL Server 2022 builds and updates
Microsoft ODBC Driver	Install the x86-64 unixODBC driver manager and the x86-64 driver on the APPX server. Match both to the APPX engine's bitness. Microsoft ODBC Driver 18 for Linux

On a new Rocky Linux 9 x86-64 database server, use Microsoft's SQL Server 2022 repository for RHEL 9. Run setup, select Developer for this test, and set the `sa` password when prompted. If SQL Server is already installed, check the version and listening port before proceeding.

```
sudo curl -fsSLo /etc/yum.repos.d/mssql-server-2022.repo \
  https://packages.microsoft.com/config/rhel/9/mssql-server-2022.repo
sudo dnf install mssql-server
sudo /opt/mssql/bin/mssql-conf setup
systemctl is-active mssql-server
```

On the APPX server, use Microsoft's RHEL 9 package repository to install the ODBC driver and command-line tools. Accept the Microsoft license when prompted. Install `mssql-tools18` on the machine where you run `sqlcmd`.

```
sudo curl -fsSLo /etc/yum.repos.d/mssql-tools.repo \
  https://packages.microsoft.com/config/rhel/9/prod.repo
sudo dnf install unixODBC.x86_64 msodbcsql18.x86_64 mssql-tools18.x86_64
```

Check that the APPX system has the APPX/ODBC product registration for SQL Server. If the APPX message names `ODB`, check that registration before troubleshooting the DSN. APPX/ODBC connection requirements

CHAPTER 2

Choose a database and create a login

Ask your DBA for the SQL Server host, TCP port, and database. The walkthrough uses the multibyte database and ASCII text.

Example database	Collation	Use
`appxmb`	`Latin1_General_100_CI_AS_SC_UTF8`	Multibyte setup used in this walkthrough.
`appxsb`	`SQL_Latin1_General_CP1_CI_AS`	Separate code page 1252 setup for a single-byte application.

Connect as a DBA with `sqlcmd`. This example uses SQL Server authentication. The command prompts for the SQL password. For a loopback lab using a self-signed certificate, add `-C`. Substitute a password for your test login. The two `CREATE DATABASE` statements are for new databases.

```
/opt/mssql-tools18/bin/sqlcmd -S tcp:127.0.0.1,1433 -U sa -Nm
```

```
USE master;
GO
CREATE DATABASE appxmb COLLATE Latin1_General_100_CI_AS_SC_UTF8;
GO
CREATE DATABASE appxsb COLLATE SQL_Latin1_General_CP1_CI_AS;
GO
CREATE LOGIN appx_sql_demo WITH PASSWORD = N'DemoSql22_42',
    DEFAULT_DATABASE = appxmb;
GO
USE appxmb;
GO
CREATE USER appx_sql_demo FOR LOGIN appx_sql_demo
    WITH DEFAULT_SCHEMA = dbo;
GRANT CREATE TABLE TO appx_sql_demo;
GRANT ALTER, SELECT, INSERT, UPDATE, DELETE
    ON SCHEMA::dbo TO appx_sql_demo;
GO
```

For `appxsb`, switch to that database and repeat `CREATE USER` and the grants there. The grants above are for a dedicated test database. `ALTER` on `dbo` lets APPX create and manage tables and indexes in that schema. Microsoft collation and Unicode reference

CHAPTER 3

Set up the 64-bit System DSN

On the SQL Server host, use `sudo ss -ltnp` to check the listening address and port. The default port is `1433`. If you change it with `mssql-conf`, restart `mssql-server`. Find or set the TCP port

1. On the APPX server, find the System DSN file with `odbcinst -j` (normally `/etc/odbc.ini`).
2. Add the following entry to that file. Use the driver name shown by `odbcinst -q -d`.

```
[APPX-SQLSERVER]
Driver=ODBC Driver 18 for SQL Server
Server=tcp:127.0.0.1,1433
Database=appxmb
Encrypt=yes
TrustServerCertificate=yes
```

3. If the host or port differs, set `Server` to the address and port of your instance.
4. Use SQL Server authentication. The `Database` setting selects `appxmb`.
5. Keep `Encrypt=yes`. For a remote server, set `TrustServerCertificate=no` and use a server name and certificate the Linux host can validate. Microsoft ODBC DSN setup

Test as the APPX engine's account. APPX supplies its password from its identity.

```
read -rsp 'SQL Server test password: ' sql_test_password; printf '\n'
isql -v APPX-SQLSERVER appx_sql_demo "$sql_test_password"
unset sql_test_password
```

At the `SQL>` prompt, check both the database and its collation against the table above.

```
SELECT DB_NAME() AS db,
       CONVERT(varchar(128), DATABASEPROPERTYEX(DB_NAME(), 'Collation')) AS collation;
```

For the single-byte database, copy the entry to `APPX-SQLSERVER-SB`, set `Database=appxsb`, and repeat the test. The remaining steps use `APPX-SQLSERVER`.

In the engine's `appx.env`, point `APPX_ODBC_LIB` to the unixODBC driver manager. If the engine uses `ODBCSYSINI`, test with that same setting.

```
APPX_ODBC_LIB=/usr/lib64/libodbc.so.2
```

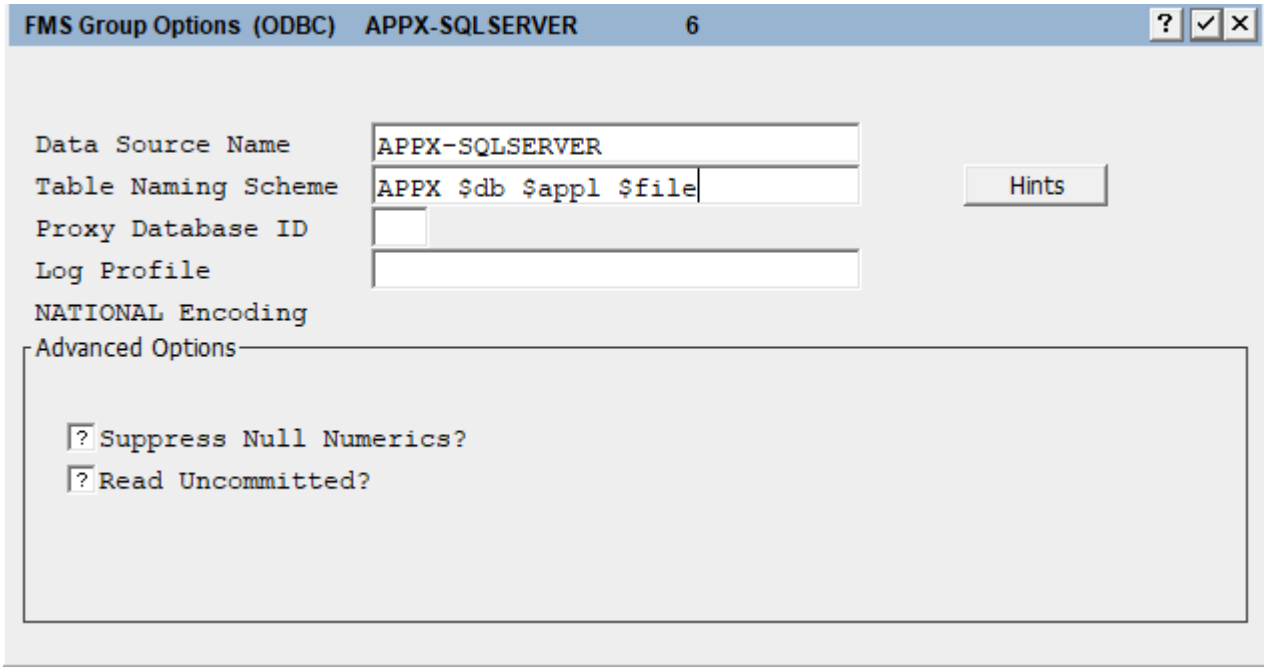
CHAPTER 4

Set up the FMS group and identity

1. Open a new APPX Desktop Client session. Choose **System Administration → Configuration → File System Groups** under **File Maintenance**.
2. Add `APPX-SQLSERVER` and enter FMS type **6**. Click **FMS Group Attributes**.
3. Enter these values and save. Leave the other options at their defaults for this example.

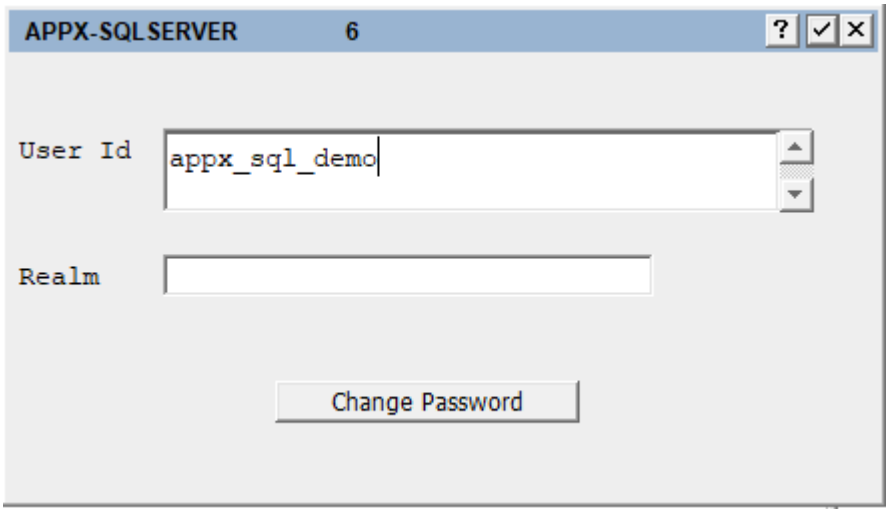
FMS option	Value
Data Source Name	`APPX-SQLSERVER`
Table Naming Scheme	`APPX_$db_$appl_$file`
Proxy Database ID	Leave blank

The APPX Desktop Client screenshots are from the Windows cookbook.



ODBC FMS options showing the DSN, naming scheme, and blank proxy database ID.

4. On the group row, click **Identities**. Leave **APPX User** blank for the default identity. Press `Enter` to accept the blank key and again to open the details. To set credentials for just one APPX user, enter that user's ID instead.
5. Enter `appx_sql_demo` in **User Id** and leave **Realm** blank. Click **Change Password**, enter the SQL Server password from the start of both fields, and click **Change Password** again. When APPX reports **Password Changed, END to Exit**, close the password dialog and save the identity.
6. Open a new APPX session before testing. APPX can keep the earlier settings for the life of a session.



Default ODBC identity for appx_sql_demo.

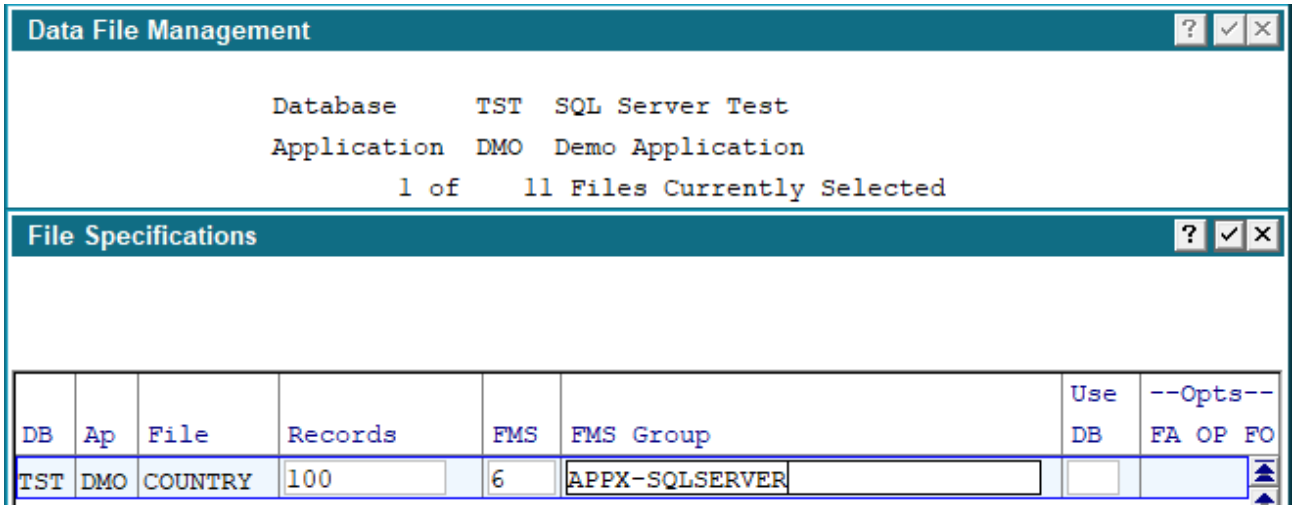
> **This SQL Server setup uses database locking.** A second SQL Server session could read the Canada row while APPX held it in Change mode. A request for an update lock with a zero lock timeout returned error `1222`. The same request succeeded after APPX released the row.

CHAPTER 5

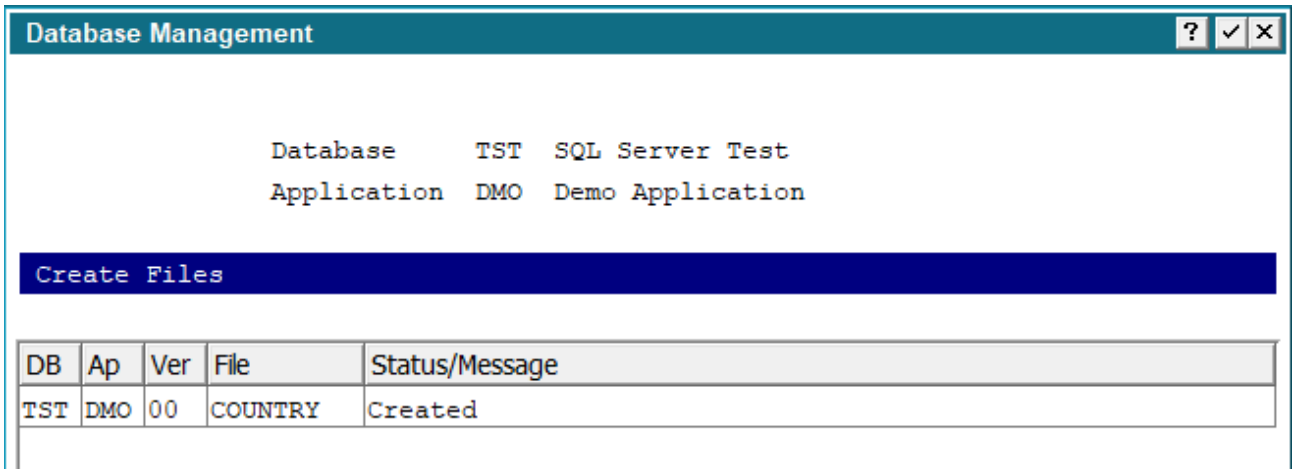
Assign one file and create the table

Use a test database and a file you can create empty. This example uses the Demo Application's `COUNTRY` file.

1. Open **System Administration → Database/Applications → Databases / Applications** under **File Maintenance**. Add `TST` with the company name **SQL Server Test**. Set the startup application to `DMO`, version `00`. Save and accept the report defaults.
2. For `TST`, open **Related Applications**. Add `DMO` in **Live Operations**. Select the `DMO` row and click **Database Management**.
3. In **File Selection**, click **Unselect All**. Highlight `COUNTRY` and press `F3` to select it. Return to **Data File Management** and check that it says **1 of 11 Files Currently Selected**.
4. Open **File Specifications** and set **FMS** to **6** and **FMS Group** to `APPX-SQLSERVER` for `COUNTRY`. Save and return.
5. Choose **Create Files**. The result for `COUNTRY` should say **Created**.



File Specifications for COUNTRY with FMS type 6 and group APPX-SQLSERVER.



Create Files reports Created for COUNTRY.

SQL Server stores the table as `dbo.APPX_TST_DMO_COUNTRY` in the database selected by the DSN. APPX also created a unique index on `country_name`.

To move an existing APPX/IO file, back it up and export it before changing its FMS type. Rehearse the move on a copy. Assign the SQL Server FMS group, create the target table and import the export. Compare the record count and check a few keys in APPX before using the converted file. APPX file conversion procedure

Field types in SQL Server

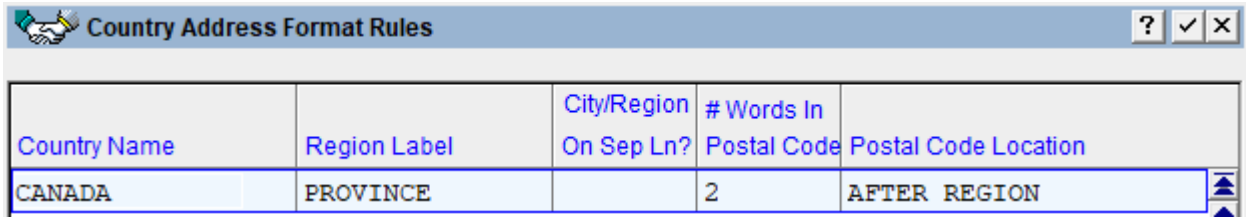
See APPX field type mappings for the column types APPX created. In SQL Server, run `EXEC sp_help N'dbo.APPX_TST_DMO_COUNTRY';` for a quick check of column types and indexes.

CHAPTER 6

Save and check a record

1. From the APPX main menu, choose **Run an Application** and execute database `TST`. In the demo's top menu, choose **Setup → Countries**.
2. In **Country Address Format Rules**, choose **Add**. Enter `CANADA` for **Country Name**, `PROVINCE` for **Region Label**, and `2` for **# Words In Postal Code**. Leave **Postal Code Location** at **AFTER REGION**. Click **Save**.
3. Open a fresh Desktop Client session, run `TST`, and go back to **Setup → Countries**. In **Inquire** mode, look up `CANADA` and check that the saved values come back.

With only `COUNTRY` created, the demo may report **Cannot Access Required File** for `PROSPECT`. Click **Cancel** on that message to continue the Countries test. Create the full demo database if you want to run the rest of the application.



CANADA and the saved values in a fresh Desktop Client session.

In `sqlcmd`, connect as `appx_sql_demo`, select `appxmb`, and check the same key. Add `-C` only for the loopback lab certificate.

```
/opt/mssql-tools18/bin/sqlcmd -S tcp:127.0.0.1,1433 \
  -U appx_sql_demo -d appxmb -Nm
```

Enter the query, then enter `GO` on its own line.

```
SELECT RTRIM(country_name) AS country_name,
       RTRIM(country_region_label) AS region_label, country_no_words
FROM dbo.APPX_TST_DMO_COUNTRY
WHERE country_name = N'CANADA';
```

The lab returned one row with `CANADA`, `PROVINCE`, and `2`.

To repeat the test in `appxsb`, set the test FMS group's **Data Source Name** to `APPX-SQLSERVER-SB`. Open a fresh APPX session, repeat **Create Files** and the record test, and use `-d appxsb` in `sqlcmd`. The single-byte test returned the same row.

CHAPTER 7

Troubleshooting

Check the APPX engine

This guide uses **FMS type 6 (ODBC)** through the SQL Server ODBC driver.

In APPX **Release Information**, record **Engine Release** and **Compile Date**. The application release numbers on the same screen identify the APPX system applications. Match the database client or ODBC driver to the bitness of the engine that runs the operation. Check the settings on the APPX server and under the Windows or Linux account that runs the engine.

Test the connection outside APPX

On the APPX server, use the 64-bit `isql` and the DSN from **FMS Group Attributes**. Check the driver, server, port, and database. Use the same login and password as APPX. Run `odbcinst -j` from the matching unixODBC installation to see its configuration paths. Check `ODBCSYSINI` and `ODBCINI` in the engine's environment. If the driver fails to load, check its bitness with `file` and its dependencies with `ldd`.

For a timeout or connection refusal, check the database service, listening address and port, and firewall. A database authentication error means the request reached the server. Check the matching server log entry for the reason it rejected the login.

For a TLS error, check the certificate name and the Linux host's trusted certificates. For a SQL login error, check the account and `/var/opt/mssql/log/errorlog`. Confirm that `APPX_ODBC_LIB` points to the 64-bit unixODBC library.

Check the FMS group and Identity

If APPX reports **Can't connect to server (or driver error)** after a successful DSN test, confirm that the FMS group's **Data Source Name** matches the tested System DSN. The DSN test uses the credentials supplied to the test. APPX uses the credentials from its FMS Identity.

Under **Identities**, look for an active, unexpired entry for the APPX user running the operation. That entry takes precedence over the blank default. Check **User Id** and re-enter the database password with **Change Password**. Save the Identity and open a fresh APPX session. APPX can cache connection and Identity settings for the life of the session.

If the same password works in the DSN test and fails in APPX, check the driver's connection string rules and the password length supported by the APPX release. A dedicated test login with a newly generated alphanumeric password can help isolate a parsing or length problem.

Check where APPX creates the table

If the connection succeeds, read the **Create Files** result or open **View Log** for the database error. Check the database, login, and target schema or database with the same connection settings as APPX. Compare the APPX database ID, application ID, file name, and **Table Naming Scheme** with the SQL table name.

Check the DSN's database and the database user's default schema. The database user needs `CREATE TABLE` in the database and `ALTER` on the target schema. SQL Server table creation permissions

For an APPX error that names the `ODB` license, check the APPX/ODBC registration on the engine that opened the file. A driver connection test does not exercise the APPX registration check.

Capture one failing operation

1. Make a copy of the `appx.env` under the effective `APPXPATH` of the engine that runs the process. Create a private log directory that the engine account can write to, then temporarily set the variables below. Keep `$(pid)` as written. The SQL logger replaces it with the engine process ID.

   ```
   APPX_SQL_CMD=/var/tmp/appx-trace/appx-rdbms-$(pid).log
   APPX_DBG_CODE=0x201F
   ```

   The mask includes generated SQL, return statuses, fetch attempts, and engine and connection operations, plus proxy lock operations where applicable. Use `0xFFFFFFFF` for one attempt if support needs more detail. The engine reads these settings on startup. `APPX_ODBC_LOG` and `APPX_ODBC_CODE` apply to the driver used by external programs to read APPX data.

2. If the APPX SQL log lacks the driver error, add `Trace=Yes` and `TraceFile=/var/tmp/appx-trace/odbc.log` under `[ODBC]` in the engine's `odbcinst.ini`. This can trace other applications using that configuration. Limit the capture to one attempt. unixODBC tracing settings
3. Open a fresh APPX session and reproduce the failure once. For **Create Files**, select only the intended test file. For a failed `READ` with `HOLD`, note the APPX error and flags before the next `WRITE`. Try a plain `READ` of the same key in another fresh session. If the log is missing, check `APPXPATH`, the effective variables, and the engine account's access to the log directory.
4. Exit the test sessions and restore the previous `appx.env` settings. If you enabled unixODBC tracing, restore the previous `odbcinst.ini` settings.

If an insert follows a failed held read, keep the first error. A later duplicate-key error indicates that the key existed when the insert ran. Use the insert branch only when the read explicitly reports that the record was absent. Handle lock and database errors before trying another write.

In the ODBC trace, find the first failed call and its diagnostic records. For `SQLDriverConnect` (which may appear as `SQLDriverConnectW`), capture the return code, SQLSTATE, and full driver message. `IM002` points to DSN or driver lookup. `08001` covers several connection failures, so read the message that accompanies it.

Share the SQL statement, fetch or lock result, and the first SQLSTATE, native error number, and message when available. Include the APPX flags, file and operation, timestamp and time zone, APPX engine release, build date and bitness, operating system, driver name and version, and database versions before and after an upgrade. Keep the SQL structure and error codes intact and replace passwords and record values with placeholders. The APPX SQL log can contain the engine environment and full connection strings. APPX environment settings