APPX LIBRARY

PostgreSQL with APPX 5.5.1 on Windows

PostgreSQL 14.1 · Windows 11 · APPX 5.5.1a (32-bit engine)

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

APPX Software, Inc.

Made on November 18, 2026 from the pages of the APPX Library, https://wiki.appx.com/manuals/library/rdbms-postgresql-14-1-windows-11-appx-551-x86/. The pages are the current version.

Contents

Chapter 1 **What to install** 3

Chapter 2 **Create the database and login** 4

Chapter 3 **Set up the 32-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

Here's the combination used for this example on Windows 11. If PostgreSQL is already installed, check that its Windows service is running and use the port it listens on. This example runs both servers on the same machine.

Component	Download
APPX server	Install the 32-bit 5.5.1 server and apply the Windows 5.5.1a patch using the directions in the ZIP. Set up and start an APPX Login Manager for this installation. 5.5.1 server installer · 5.5.1a patch · Installation instructions
APPX Desktop Client	Use the Windows 5.5.1 client to sign in to the Login Manager. 5.5.1 Desktop Client installer
PostgreSQL	The test server ran 14.1 x64. The exact installer is here for reproducing this setup. For a new installation, use a maintained PostgreSQL release and test it with your application. 14.1 x64 installer · Windows downloads
psqlODBC	Install the x86 driver on the machine running the APPX engine. The engine is 32-bit, so it needs the 32-bit driver even with 64-bit PostgreSQL and Windows. 18.00.0001 x86 MSI · Driver documentation

CHAPTER 2

Create the database and login

Open **SQL Shell (psql)** and connect to `postgres` as a PostgreSQL administrator. Run the following at the `psql` prompt. `WIN1252` is the encoding for this single-byte example. Use your own password outside the lab.

```
CREATE ROLE appx_demo LOGIN PASSWORD 'AppxDemo551!';

CREATE DATABASE appx_demo OWNER appx_demo
    TEMPLATE template0 ENCODING 'WIN1252'
    LC_COLLATE 'C' LC_CTYPE 'C';

REVOKE CONNECT ON DATABASE appx_demo FROM PUBLIC;
\connect appx_demo
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
GRANT USAGE, CREATE ON SCHEMA public TO appx_demo;
```

The APPX login needs permission to connect to the database and create tables in the target schema. This example uses `public`. If your DBA has already prepared a database and a different schema, see how to select that schema. PostgreSQL CREATE DATABASE

CHAPTER 3

Set up the 32-bit System DSN

1. On the APPX server, run `%windir%\SysWOW64\odbcad32.exe` as a Windows administrator. The title should read **ODBC Data Source Administrator (32-bit)**. Microsoft's ODBC Administrator paths
2. On the **System DSN** tab, click **Add**, pick **PostgreSQL ANSI**, and click **Finish**.
3. Enter the values below. Under **Datasource**, clear **Bools as Char**, **LF <-> CR/LF conversion**, and **ReadOnly**. The first two affect how Boolean values and line endings are returned. psqlODBC options
4. Enter the example password and click **Test**. After a successful test, clear the password in the DSN and save it. APPX supplies the password from its own FMS identity.

DSN setting	Example
Data Source	`APPX-POSTGRES`
Server and Port	`127.0.0.1` and `5432`
Database	`appx_demo`
User Name for Test	`appx_demo`
SSL Mode	`prefer` for this local test

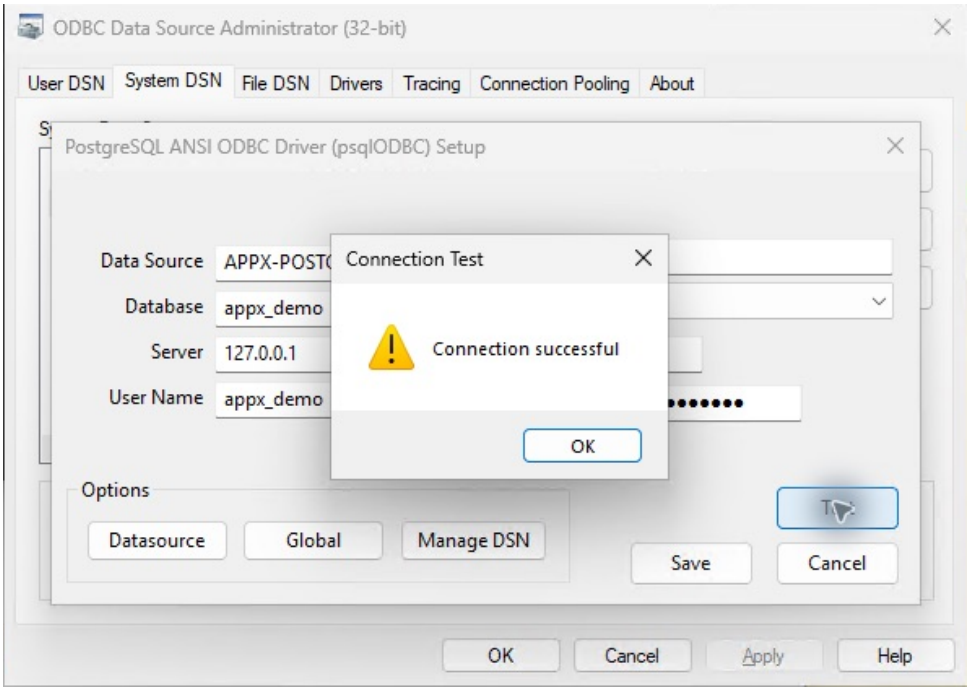


Figure 1. The `APPX-POSTGRES` System DSN passes the 32-bit connection test.

CHAPTER 4

Set up the FMS group and identity

1. Open the **APPX Desktop Client**, select **Remote**, and sign in to your 5.5.1 Login Manager with an APPX administrator account. Use the Login Manager port in this dialog. The lab used `127.0.0.1` and `31052`.
2. Go to **System Administration → Configuration → File System Groups**. Add `APPX-POSTGRES` with **FMS** set to **6**. Open **FMS Group Attributes**.
3. Enter the settings below and save. Reserve an unused three-character ID for proxy locks. We normally use `LOK`.

FMS group field	Value
Data Source Name	`APPX-POSTGRES`
Table Naming Scheme	`APPX_$db_$appl_$file`
Proxy Database ID	`LOK`

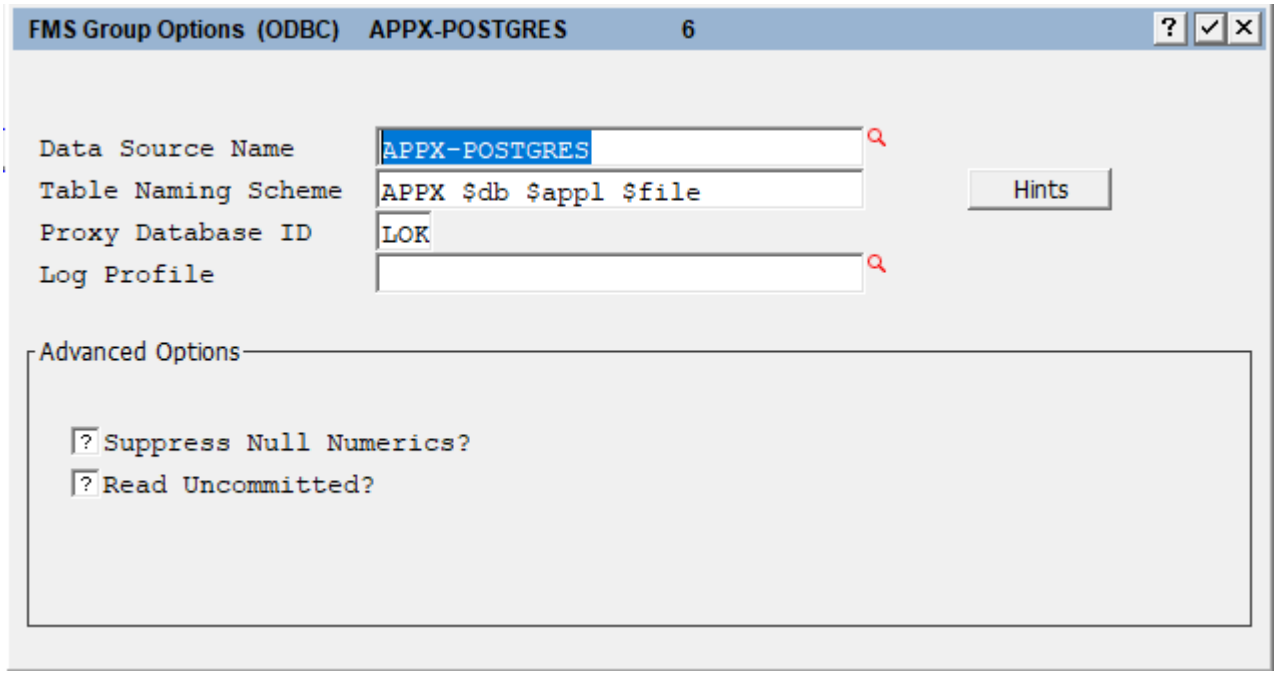


Figure 2. The saved DSN, table naming scheme, and proxy database ID.

> **APPX 5.5.1 requires a proxy database ID for PostgreSQL.** When an APPX session holds a record for change, it takes a lock in an APPX/IO proxy file. Other APPX sessions check that file. Direct PostgreSQL clients can still update the row, so account for that if another program writes to these tables. APPX FMS group options

4. On the group row, click **Identities**. Leave **APPX User** blank for the default identity, or enter the APPX user ID for a specific mapping. For the default, press `Enter` to accept the blank key and again to open the details.
5. Set **User Id** to `appx_demo` and leave **Realm** blank. Click **Change Password**, type the PostgreSQL password from the start of each box, and click **Change Password** again. APPX keeps the characters hidden as you type. When the status says **Password Changed, END to Exit**, close the password window and save the identity.
6. Use **File → New Session** before testing the saved FMS settings. APPX can keep the earlier settings for the life of a session.

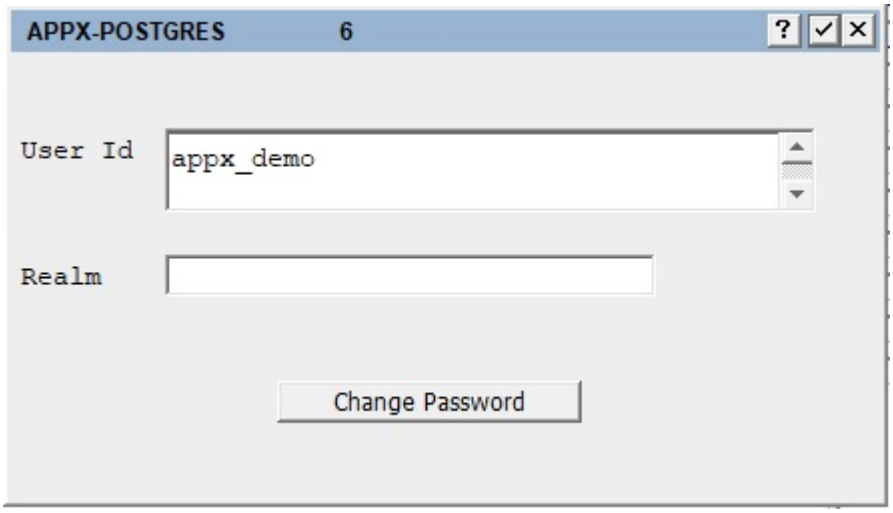


Figure 3. Use the PostgreSQL login in **User Id** and set its password here.

CHAPTER 5

Assign one file and create the table

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

1. From **System Administration → Database/Applications**, open **Databases / Applications** under **File Maintenance**. Add database `TST` with the company name **PostgreSQL Test**. Press `Enter` to get to the **Startup** fields, enter application `DMO` and version `00`, then save.
2. For `TST`, open **Related Applications**. Add `DMO` in **Live Operations**, then click **Database Management** on that row.
3. Open **File Selection**. Click **Unselect All**, then **Select**. Put `COUNTRY` in **Name**, accept the selection, and return to **Data File Management**. Check that it says **1 of 11 Files Currently Selected**.
4. Open **File Specifications**. For `COUNTRY`, set **FMS** to **6** and **FMS Group** to `APPX-POSTGRES`. Save and return.
5. Choose **Create Files**. The result for `COUNTRY` should say **Created**.

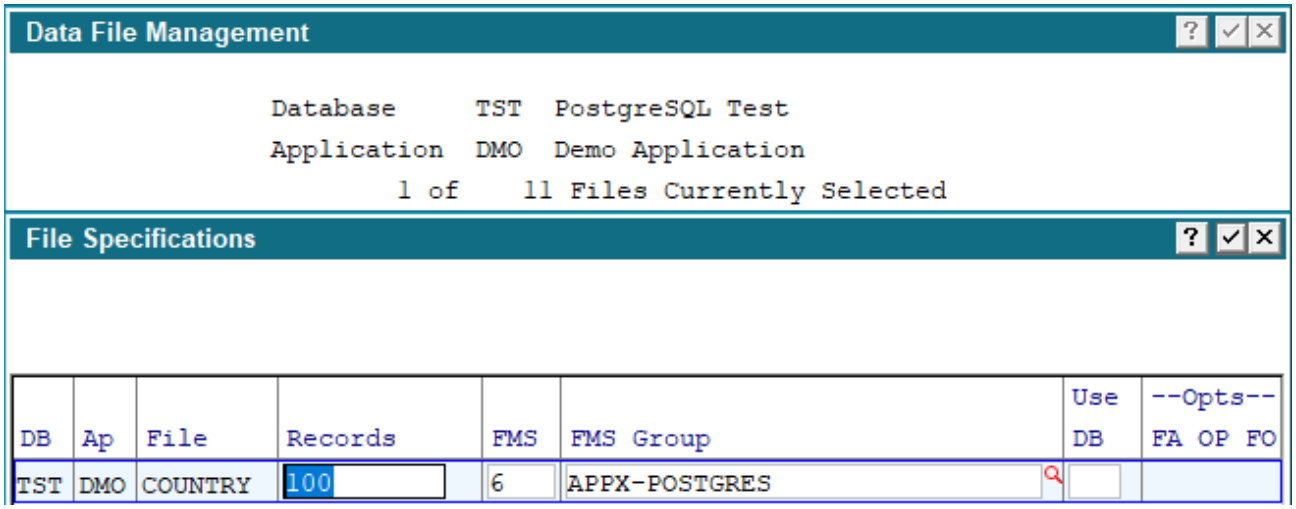


Figure 4. The FMS group is assigned to `COUNTRY` in the `TST` database.

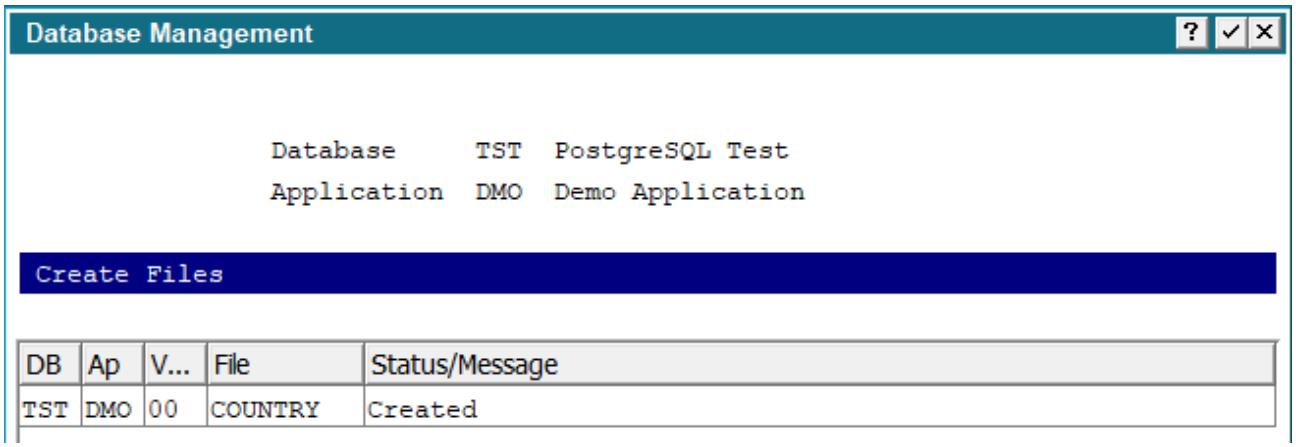


Figure 5. Create Files reports **Created** for the selected file.

The naming scheme expands to `APPX_TST_DMO_COUNTRY`. PostgreSQL folds the unquoted name to lowercase, so the table is `public.appx_tst_dmo_country`. APPX also creates a unique index for the `COUNTRY` key. PostgreSQL identifier rules

To move an existing APPX/IO file, rehearse on a copy. Back up the file and export it while its FMS type is still **1**. Change the file specification to FMS type **6** and assign the PostgreSQL group. Import the export, then compare the record count and check a few keys in APPX. Keep the original backup until those checks pass. Convert between APPX/IO and ODBC

Field types in PostgreSQL

See APPX field type mappings for the column types this setup produced. In `psql`, run `\d public.appx_tst_dmo_country` to see all the columns and the key index.

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 new Desktop Client session and go back to **Setup → Countries** in `TST`. In **Inquire** mode, check that the saved record comes back.

In a test database with only `COUNTRY` created, the demo may report **Cannot Access Required File** for `PROSPECT`. Dismiss that message with **Cancel** to continue with this Add and Inquire test. Use a complete demo database for the rest of the demo.

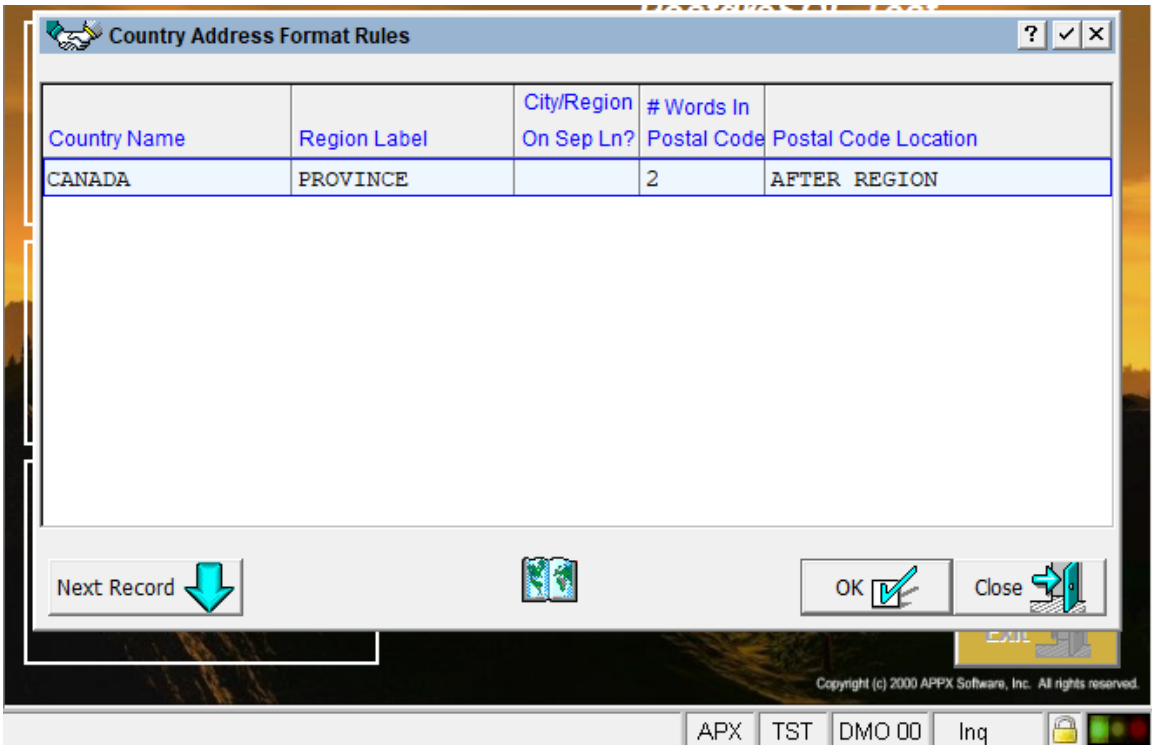


Figure 6. A fresh APPX session reads the saved record in **Inquire** mode.

Connect to `appx_demo` in `psql` as `appx_demo` and query the same key.

```
SELECT country_name, country_region_label, country_no_words
FROM public.appx_tst_dmo_country
WHERE country_name = 'CANADA';
```

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

CHAPTER 7

Troubleshooting

Check the APPX engine

This guide uses **FMS type 6 (ODBC)** through the PostgreSQL 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, open the 32-bit ODBC Administrator (`%windir%\SysWOW64\odbcad32.exe` on 64-bit Windows). Under **System DSN**, open the DSN from **FMS Group Attributes**. Confirm the driver, server, port, and database. Run **Test** with the same login and password as APPX. Close the setup dialog with **Cancel** if you entered credentials just for the test.

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.

A `no pg_hba.conf entry` message points to the client address, database, user, or connection type in the authentication rules. A `password authentication failed` message means PostgreSQL selected an authentication rule and rejected the login. Look for the matching `FATAL` and `DETAIL` lines in the PostgreSQL server log. PostgreSQL authentication messages

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 its length and punctuation locally. APPX 5.5.1 and 5.5.1a put the password in the ODBC connection string without escaping it. psqlODBC can interpret an unescaped `+` as a space, and `;` separates attributes. The Identity uses a 30-byte password field. To isolate a parsing or length problem, use a dedicated test login with a newly generated alphanumeric password shorter than 30 bytes, then repeat both tests with that login. psqlODBC connection string rules

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.

The DSN's **Database** names the database. PostgreSQL creates an unqualified table in the current schema. A `permission denied for schema public` error means the statement reached `public`. If the DBA assigned a schema such as `appx_data`, select that schema for the APPX connection.

In the 32-bit ODBC Administrator, open the PostgreSQL System DSN. Under **Datasource > Advanced Options 2/3 > Connect Settings**, enter `SET search_path TO appx_data` and save.

Keep the DSN's database name. Use a separate DSN for APPX if other applications use the existing DSN. If a grant is missing, ask the DBA to connect to the target database and grant `USAGE, CREATE` on the assigned schema to the APPX login. Apply the setup example's `REVOKE` commands only when preparing a new database you administer.

```
SELECT current_database(), current_user, current_schema();
SELECT has_schema_privilege(current_user, 'appx_data', 'USAGE') AS can_use,
       has_schema_privilege(current_user, 'appx_data', 'CREATE') AS can_create;
SELECT to_regclass('appx_data.appx_tst_dmo_country');
```

Run the first query through the same DSN to verify the connection's search path. Substitute your schema and table in these queries. After **Create Files** reports **Created**, confirm the table and its key index in the assigned schema. PostgreSQL schema search path · psqlODBC Connect Settings

For a proxy file error, check the FMS group's **Proxy Database ID** (`LOK` in this example) and the engine account's access to the proxy files under `APPXPATH`. If APPX reports that a record is held, check whether another APPX session has it open in Change mode.

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=C:\APPX\Logs\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, open **Tracing** in the same 32-bit ODBC Administrator. Set **Log File Path** to a private file the engine can write, such as `C:\APPX\Logs\odbc.log`. Click **Apply**, then **Start Tracing Now**. If tracing must include an engine running under another Windows identity, use **Machine-Wide tracing for all user identities** for this attempt. Windows ODBC tracing
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 ODBC tracing, click **Stop Tracing Now** and restore the previous tracing options.

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.

In the APPX 5.5.1 Windows lab, an invalid Identity produced `SQL_ERROR` and `08001` with PostgreSQL's password authentication error. The corrected Identity returned `SQL_SUCCESS_WITH_INFO` and created the table and index. Both attempts reported `IM006`. Use the connection return code and the database message to identify the failure.

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