APPX LIBRARY

MariaDB 10.11 with APPX 5.5.4 on Rocky Linux

MariaDB 10.11.18 · Rocky Linux 9.8 · APPX 5.5.4 (32-bit engine)

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

APPX Software, Inc.

Made on November 18, 2026 from the pages of the APPX Library, https://wiki.appx.com/manuals/library/rdbms-mariadb-10-11-18-rocky-9-8-appx-554-x86/. 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 32-bit System DSN** 5

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

Chapter 5 **Assign one file and create the table** 7

Chapter 6 **Save and check a record** 8

Chapter 7 **Troubleshooting** 9

CHAPTER 1

What to install

The example uses a 32-bit APPX 5.5.4 engine on Rocky Linux 9.8 and a 64-bit MariaDB 10.11.18 database. Install the 32-bit MariaDB Connector/ODBC client on the APPX server.

Component	Download
APPX server	Install the 32-bit APPX 5.5.4 engine. On x86-64 Rocky Linux, use the installer for 64-bit Linux. It installs the 32-bit APPX engine. APPX 5.5.4 Linux server installer · Installation instructions
APPX Desktop Client	Use the 5.5.4 Desktop Client to open an APPX session. Desktop Client installer
MariaDB	Install MariaDB 10.11 or use an existing server with a dedicated test database. Database installation
ODBC driver	Match the client to the APPX engine's 32-bit architecture. MariaDB Connector/ODBC 3.1.22

Install `unixODBC.i686`. The lab used unixODBC `2.3.12` and a matching 32-bit `isql` under `/usr/local/unixodbc32`. Build Connector/ODBC against the 32-bit Connector/C and unixODBC libraries and headers. Verify `libmaodbc.so` and its dependencies with `file` and `ldd`.

CHAPTER 2

Choose a database and create a login

Ask your DBA for the host, port, and database name. This single-byte example uses `appxsb` with `latin1_general_ci`. Use a dedicated schema or database to rehearse the setup. Replace the example password with a generated value.

```
CREATE DATABASE appxsb
    CHARACTER SET latin1 COLLATE latin1_general_ci;
CREATE USER 'appx_maria_demo'@'127.0.0.1'
    IDENTIFIED BY 'use_your_test_password';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, INDEX, DROP
    ON appxsb.* TO 'appx_maria_demo'@'127.0.0.1';
```

CHAPTER 3

Set up the 32-bit System DSN

Register the 32-bit driver in `odbcinst.ini` and create the System DSN in `odbc.ini`. Run `odbcinst -j` to find these files. Point the engine's `appx.env` to the same directory and 32-bit driver manager.

```
# odbcinst.ini
[MariaDB]
Driver=/opt/mariadb-odbc32/lib/mariadb/libmaodbc.so

# odbc.ini
[APPX-MARIADB]
Driver=MariaDB
Server=127.0.0.1
Port=3306
Database=appxsb
CHARSET=latin1
```

```
APPX_ODBC_LIB=/usr/lib/libodbc.so.2
ODBCSYSINI=/etc
```

Test as the APPX engine's account using the same DSN directory and the 32-bit `isql`. Enter the password at the prompt.

```
read -rsp 'Test password: ' test_password; printf '\n'
ODBCSYSINI=/etc /usr/local/unixodbc32/bin/isql -v APPX-MARIADB appx_maria_demo "$test_password"
unset test_password
```

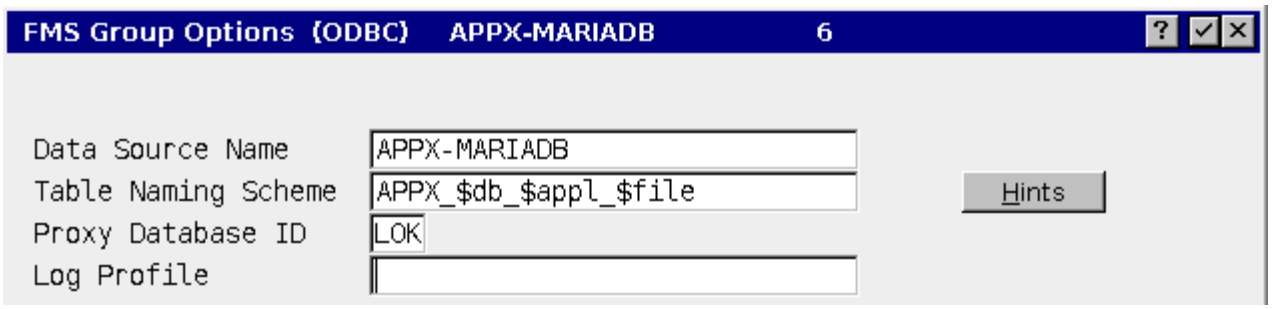
CHAPTER 4

Set up the FMS group and identity

Open a new APPX Desktop Client session. Choose **System Administration → Configuration → File System Groups** under **File Maintenance**.

1. Add `APPX-MARIADB` and enter FMS type `6`.
2. Click **FMS Group Attributes.** Enter the values below and save.

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



The FMS group options for this connection.

On the group row, click **Identities.** Enter your APPX user ID to give that user a database login. Leave **APPX User** blank to set a default identity for the group. Press Enter to accept the key and again to open the details.

Enter `appx_maria_demo` in **User Id** and leave **Realm** blank. Save the identity and reopen it in **Change** mode. Click **Change Password**, enter the database password, and press `Tab` to move to **Confirm Password**. Enter it again and click **Change Password.** When APPX reports **Password Changed, END to Exit**, close the password dialog and save the identity. Open a fresh APPX session before testing.

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 **MariaDB 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** if it is absent. Select the `DMO` row and click **Database Management**. If APPX prompts for a **Language ID**, set `ENG` and reopen **Database Management**.
3. In **File Selection**, click **Unselect All**. Highlight `COUNTRY` and press `F3`. If you use the **Select** button, enter `COUNTRY` in **Name** and confirm. Return to **Data File Management** and confirm that one file is selected.
4. Open **File Specifications**. Set **FMS** to `6` and **FMS Group** to `APPX-MARIADB` for `COUNTRY`. Save and return.
5. Choose **Create Files**. The result for `COUNTRY` should say **Created**.

Create Files

DB	Ap	V...	File	Status/Message
TST	DMO	00	COUNTRY	Created

Create Files reports Created for COUNTRY.

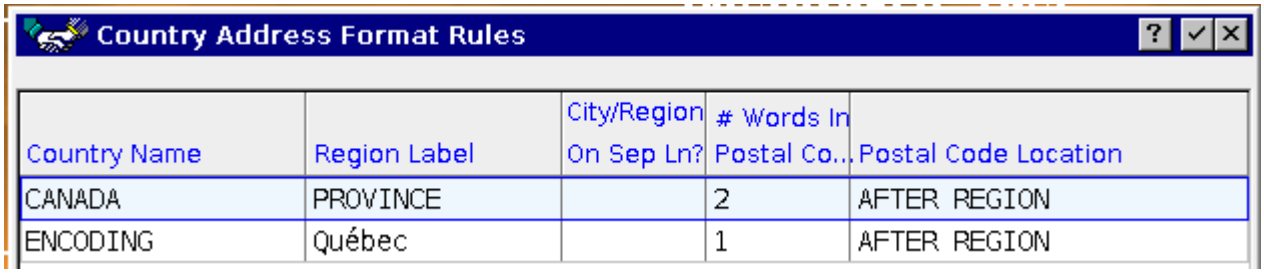
This naming scheme creates `APPX_TST_DMO_COUNTRY` in the selected database or schema. For an existing file, follow the APPX/IO migration and rollback guide.

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. Repeat with `ENCODING` for **Country Name**, `Québec` for **Region Label**, and `1` for **# Words In Postal Code**. Save.
4. Open a fresh Desktop Client session, run `TST`, and go back to **Setup → Countries**. Choose **Inquire** and click **First Record** if the key field is empty. Find the two keys and confirm the saved values, including the accent in `Québec`.

Cannot Access Required File for PROSPECT. In a test database with only `COUNTRY` created, click **Cancel** on this message to continue the Countries test. Create the full demo database to run the rest of the application.



Both records in a fresh Desktop Client session.

Connect independently in mariadb as `appx_maria_demo`. Check the records, encoded value, and unique index on `country_name`.

```
SELECT RTRIM(country_name), RTRIM(country_region_label), country_no_words,
       HEX(RTRIM(country_region_label)) AS stored_bytes
FROM APPX_TST_DMO_COUNTRY ORDER BY country_name;

SHOW INDEX FROM APPX_TST_DMO_COUNTRY;
```

Expect `CANADA / PROVINCE / 2` and `ENCODING / Québec / 1`. In the Windows-1252 text, `é` is byte `E9`.

CHAPTER 7

Troubleshooting

Check the APPX engine

This guide uses **FMS type 6 (ODBC)** through the MariaDB 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 32-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 `Access denied`, check the user and client host in the error. MariaDB matches accounts by user and host.

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.

In the MariaDB client, run `SELECT DATABASE(), CURRENT_USER();` and `SHOW GRANTS;`. Check `CREATE` on the selected database. Use `SHOW TABLES;` to check the exact spelling since table name case can differ by platform.

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=/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

If the SQL query returns a different result from the APPX screen, verify the effective FMS group, database, schema, and table name. Restart the APPX session after changing any of them.