APPX LIBRARY

MariaDB 10.11 with APPX 6.2.1 on Rocky Linux

MariaDB 10.11.18 · Rocky Linux 9.8 · APPX 6.2.1 (64-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-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 uses the 64-bit APPX 6.2.1 engine on Rocky Linux 9.8, MariaDB Server 10.11.18, unixODBC 2.3.12, and 64-bit MariaDB Connector/ODBC 3.1.22. APPX and MariaDB ran on the same host. If MariaDB runs elsewhere, put unixODBC and the driver on the APPX server.

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
MariaDB Server	Use the Rocky Linux 10.11 AppStream packages. Rocky 9.8 MariaDB 10.11.18 server RPM · MariaDB downloads
MariaDB Connector/ ODBC	Install the 64-bit driver on the APPX server. Connector/ODBC 3.1.22 EL9 x86-64 RPM · 3.1.22 packages

On a new Rocky Linux 9.8 database server, use DNF to install the 10.11 stream and its dependencies. If MariaDB is already installed, check the version and listening port before proceeding.

```
sudo dnf module install mariadb:10.11/server
sudo systemctl enable --now mariadb.service
```

On the APPX server, install unixODBC and the downloaded Connector/ODBC package.

```
sudo dnf install unixODBC ./mariadb-connector-odbc-3.1.22-1.el9.x86_64.rpm
```

CHAPTER 2

Choose a database and create a login

Ask your DBA for the MariaDB host, TCP port, and database. The server can host both of these databases without changing its default character set.

Example database	Character set and collation	Use
appxmb	utf8mb4 · utf8mb4_unicode_ci	Multibyte database used in this walkthrough.
appxsb	latin1 · latin1_general_ci	Separate single-byte database for an application that needs that character set.

Connect as a DBA in the mariadb client. Run this for a new test database and login. Use a password of your own. The login is restricted to 127.0.0.1 in this example. If APPX runs on a separate machine, use the address of the APPX server in the MariaDB account.

```
CREATE DATABASE appxmb
    CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE DATABASE appxsb
    CHARACTER SET latin1 COLLATE latin1_general_ci;

CREATE USER 'appx_maria_demo'@'127.0.0.1'
    IDENTIFIED BY 'DemoMaria10_42';

GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, INDEX, DROP
    ON appxmb.* TO 'appx_maria_demo'@'127.0.0.1';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, INDEX, DROP
    ON appxsb.* TO 'appx_maria_demo'@'127.0.0.1';
```

These grants let APPX create and manage tables and indexes in the two test databases. The record test uses appxmb and ASCII text. MariaDB CREATE DATABASE

CHAPTER 3

Set up the 64-bit System DSN

Install unixODBC and the x86-64 MariaDB Connector/ODBC on the APPX server. Put `odbcinst.ini` and `odbc.ini` in a directory the APPX engine can read. This example uses `/opt/appx/mariadb-odbc`. Use the actual path to `libmaodbc.so`. Run `ldd` on it to check that the loader finds all its dependencies.

```
# odbcinst.ini
[MariaDB]
Driver=/usr/lib64/libmaodbc.so

# odbc.ini
[APPX-MARIADB]
Driver=MariaDB
SERVER=127.0.0.1
PORT=3306
DATABASE=appxmb
CHARSET=utf8mb4

[APPX-MARIADB-SB]
Driver=MariaDB
SERVER=127.0.0.1
PORT=3306
DATABASE=appxsb
CHARSET=latin1
```

Use the server's actual host and port. Run a connection test as the account that runs the APPX engine. Use a lab password for this test. For a remote server, set the driver's TLS options according to the server configuration. MariaDB unixODBC DSN setup

```
ODBCSYSINI=/opt/appx/mariadb-odbc \
  isql -v APPX-MARIADB appx_maria_demo DemoMaria10_42
```

At the `SQL>` prompt, run the following. Repeat for `APPX-MARIADB-SB` and check that both character sets read `latin1`.

```
SELECT DATABASE(), @@character_set_database, @@character_set_client;
```

Set these values in the APPX engine's `appx.env`. `APPX_ODBC_LIB` names the unixODBC driver manager. The driver manager loads MariaDB's `libmaodbc.so` from the driver definition.

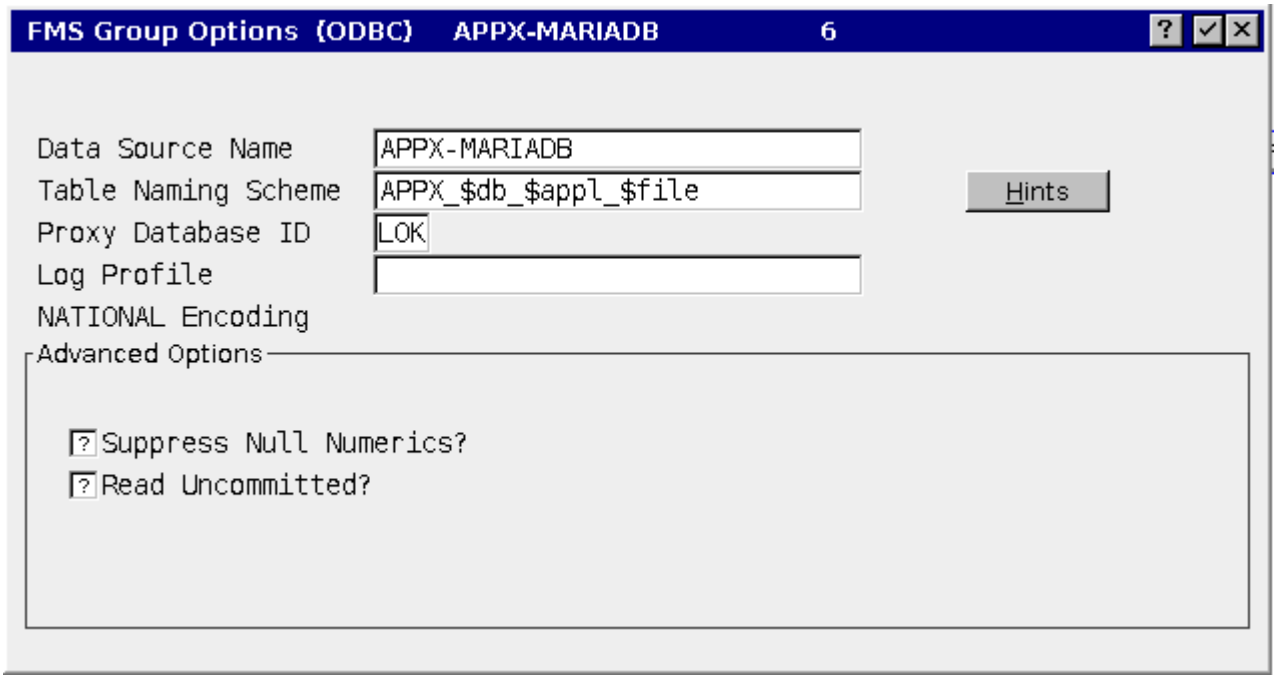
```
APPX_ODBC_LIB=/usr/lib64/libodbc.so.2
ODBCSYSINI=/opt/appx/mariadb-odbc
```

CHAPTER 4

Set up the FMS group and identity

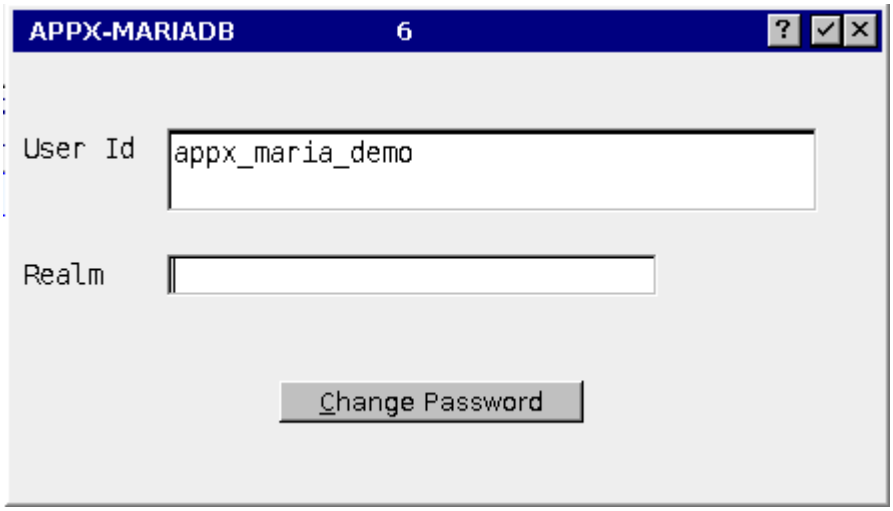
1. Open a new APPX Desktop Client session. From the main menu, choose **System Administration → Configuration → File System Groups** under **File Maintenance**.
2. Add `APPX-MARIADB` and enter FMS type **6**. Click **FMS Group Attributes**.
3. Enter these values and save. Reserve an unused three-character ID for proxy locks. We normally use `LOK`.

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



ODBC FMS options showing the DSN, naming scheme, and 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_maria_demo` in **User Id** and leave **Realm** blank. Click **Change Password**, enter the MariaDB password from the start of each field, and click **Change Password** again. When APPX reports **Password Changed, END to Exit**, close the 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_maria_demo.

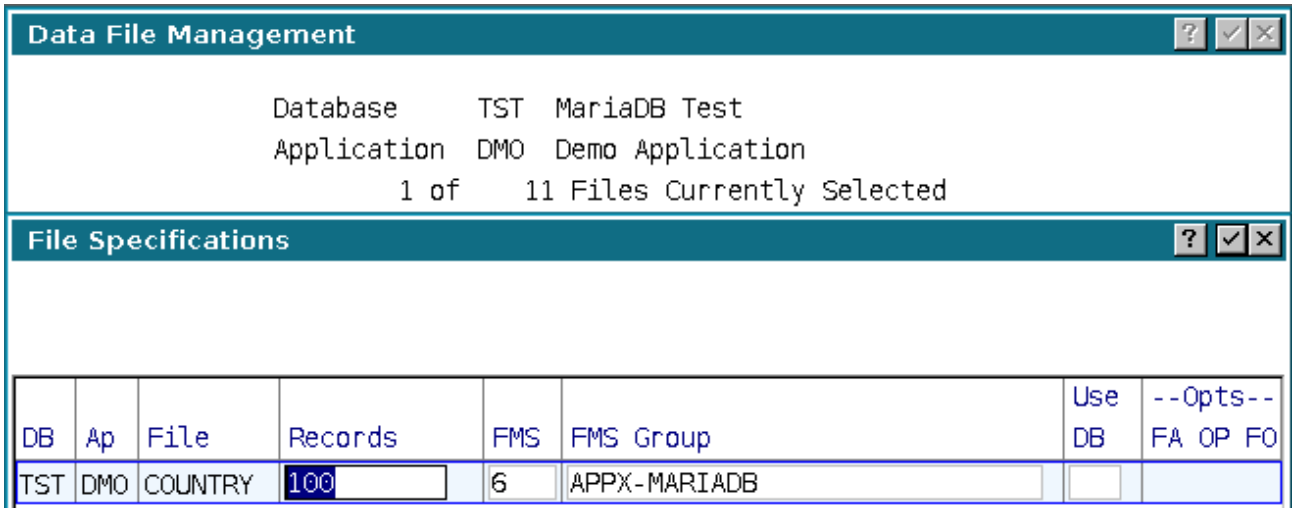
> **Use proxy locking for MariaDB with FMS type 6.** APPX uses an APPX/IO proxy file to coordinate record holds between APPX sessions. The engine's operating system account must be able to create and open the proxy files. Direct MariaDB clients can take their own locks and update the table, so account for that if another program writes to these records.

CHAPTER 5

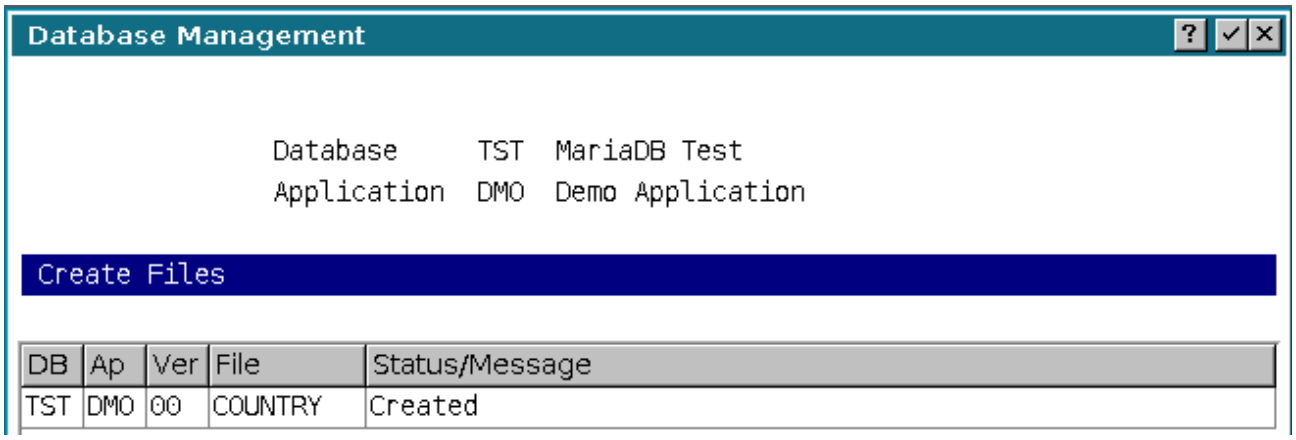
Assign one file and create the table

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**. 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-MARIADB` 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-MARIADB.



Create Files reports Created for COUNTRY.

The naming scheme expands to `APPX_TST_DMO_COUNTRY`. In this lab, MariaDB stores it as `appxmb.APPX_TST_DMO_COUNTRY`. 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 MariaDB 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 MariaDB

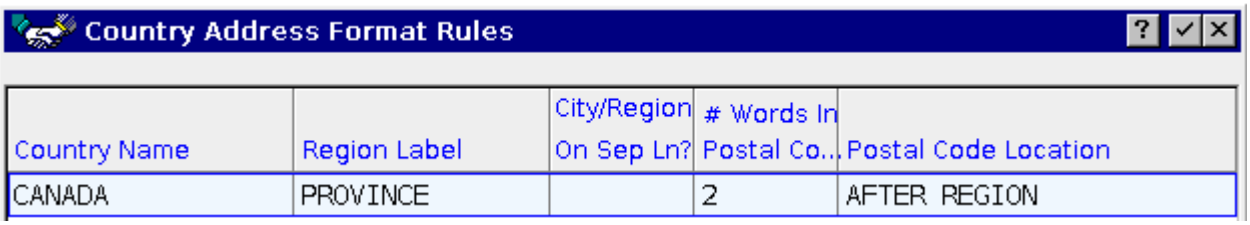
See APPX field type mappings for the column types APPX created. In the `mariadb` client, run `SHOW FULL COLUMNS FROM appxmb.APPX_TST_DMO_COUNTRY;` for the column types and collations. In this example, APPX created a `varchar(2)` column for the two-digit Numeric field. SQL sorts that column as text.

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.

Connect to `appxmb` in the `mariadb` client as `appx_maria_demo` and check the same key.

```
SELECT RTRIM(country_name) AS country_name,
       RTRIM(country_region_label) AS region_label, country_no_words
FROM appxmb.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 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 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 `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