



APPX LIBRARY

APPX ODBC 5.5.2 Cookbook (32-bit)

APPX 5.5.2, 32-bit clients

Updated October 20, 2026: SQL names, reserved words and views checked. Verified October 15, 2026 on Windows 11 Pro 25H2. Matching 5.5.2 engine/gateway/ASQL and supplied Dharma 9.01 x86 driver. WinSQL, Excel, enriched PowerShell, Base direct SQL and Calc passed. Crystal Reports 2016 passed an ODBC (RDO) SQL command, including save/export and fresh refresh. Repeated Access imports fail; the nearest verified Access success is documented in 5.5.3.

OFFLINE FILES

Using this PDF offline

Safari and Microsoft Edge: If the browser shows no Attachments panel, use the [complete offline package](#) or [scripts-only package](#). These ordinary download links require Internet access and provide the same files embedded in this PDF.

Extract directly from the PDF: Save the PDF and open it in desktop Adobe Acrobat or Acrobat Reader on Windows or macOS. Choose View > Show/Hide > Side panels > Attachments (under Menu on Windows), select the files, and choose Save Attachment. The browser PDF toolbar and the desktop attachment panel are different controls.

This PDF/A-3u edition contains its instructions, screenshots, fonts and a set of checked examples. Open the PDF viewer's Attachments panel to save the files. The complete set is also in `appx-odbc-552-offline.zip`; `appx-odbc-552-scripts.zip` contains the PowerShell helpers.

Alongside the two PowerShell helpers are the read-only COUNTRY SQL, seven expected records, nine column descriptions, and the saved Calc workbook and Base connection/query. Read `README-offline.txt` for the required 32-bit clients and database registration. The Base file requires a live ODBC connection; the Calc workbook contains a sample snapshot and can refresh through that connection. Neither file contains a saved password.

Download links in this PDF lead here. Save and read the required files before following their recipes. Attachments do not run automatically. Some browser viewers hide attachments; download the PDF and open it in a viewer with an Attachments panel. Use `SHA256SUMS.txt` to check the extracted examples.

Vendor installers and external reference manuals remain download links. APPX and gateway access are required to run a query or refresh live data.

Embedded file	SHA-256
APPX-552-COUNTRY.odb	72c1ef825605f07e1c5484d3d8c97ad44b87d0ffe950d6ebb474988742b1d57d
APPX-552-COUNTRY.ods	71d2702c6b464c2ceac3a0ef986296d0facf6f5d78bba35cc721ca92920e0d76
COUNTRY-columns.csv	3170dbb1d5cc7c5e9ebcd15eab3bc95332f5c031c4839c416407abc44d69d117
COUNTRY-expected.csv	3ebd56a425a722096e90724381919f2a7b535dac8ae17cd19102a64834896790
COUNTRY-report.sql	47f9348f0dd74d0b34599aee6b910289d6fcf8061cb176f030990a624e8b8af3
COUNTRY.sql	fb25dbf415bc4251bb57576b32e49e6fa292edd2d8953de12b6a6ebccbbe636a

Embedded file**SHA-256**

README-offline.txt	2b37c5a2af612914e80c9aa170912d8376686f8c440574324ccf742907a583b2
SHA256SUMS.txt	7312e46a964a8d6baf790dcdbfbf7db21048822c26d3fbb61e54476fdd3f573d
query-appx-country-32.ps1	ae11a07ef6a606f938ed5c1adcb40a99a9e04cbd738136f5ce7447d6d4231b6
read-appx-odbc-32.ps1	febe68779ff090f89ff683ee3b842d2d02d696c09c1a60bcff007d4b5c1440f6

Contents

Connection glossary: Service, sqlnw, DSNs and ports	9
Chapter 1 Before You Start	6
What You Need and Where to Begin	6
Check Versions and Choose 32-bit Programs	7
What Works Better in Later Releases.....	8
Identify the Server, Ports and Connection Names	9
Connection Glossary: Service, sqlnw, DSNs and Ports	9
Decide What to Install on Each Computer.....	14
Download the Tested Example Files	14
Chapter 2 Install and Check APPX ODBC	16
Install the 5.5.2 Gateway and ASQL	16
Start the Gateway and Check Required Microsoft Files.....	20
Install the Driver on Another Windows Computer	22
Check the Client-to-Gateway and Gateway-to-APPX Connections.....	22
Check the Saved AppxODBC Connection (DSN).....	23
Chapter 3 Make APPX Files Available to SQL	25
What ASQL Does and Where to Find It.....	25
Add the COUNTRY Table to SQL with ASQL	25
Check Every COUNTRY Record and Field.....	28
Show Only the APPX Application You Need.....	29
When to Create or Update SQL Table Definitions.....	30
How APPX Files and Fields Become SQL Tables and Columns	31
SQL Name Rules and Reserved Words.....	32
Review and Edit the SQL That ASQL Creates.....	35
Use a View for Friendly Names and Shorter Queries.....	37
Check Limits for Large Tables and Long SQL Statements.....	38
Chapter 4 Read APPX Data with WinSQL.....	41
Connect WinSQL and Read COUNTRY	41
Chapter 5 Read APPX Data with Excel, Access and LibreOffice.....	42
Choose 32-bit Excel and Access	42
Load COUNTRY into Excel and Refresh It.....	42
Access Import Fails with the Supplied Driver	44

Choose 32-bit LibreOffice Calc and Base	45
Connect LibreOffice Base and Save a COUNTRY Query	46
Load COUNTRY into LibreOffice Calc and Refresh It	50
Help LibreOffice Find a Saved Base Connection.....	52
Chapter 6 Create Reports with Crystal Reports 2016	54
Choose a Compatible Crystal Reports Version	54
Create a COUNTRY Report through ODBC.....	54
How This Differs from the 5.5.1 Report Procedure	56
Chapter 7 Read APPX Data with 32-bit PowerShell	57
Run a Query with a Secure Login Prompt.....	57
Check the Driver Architecture and Gateway Connection	58
Work with Query Results in PowerShell.....	59
Filter Records with SQL Parameters	60
Export COUNTRY to CSV and JSON.....	60
Summarize and Check the Returned Data.....	61
List Tables and Columns and Identify the Driver	62
Set Query Limits and Read Error Details	62
Chapter 8 Solve Connection and Import Problems.....	64
Set Connection and Diagnostic Options.....	64
Record a Diagnostic Log, Then Turn Logging Off	67
Check the Driver, Ports and Login Before Retrying	69
Check the dhsodbc.ini and appx.env Settings Files	69
Fix Driver Setup Error 126.....	70
Fix the ASQL System Directory Error	70
Fix Import Errors Caused by Field Names.....	70

CHAPTER 1

Before You Start

Check the tested setup, connection names and software needed for this edition.

What You Need and Where to Begin

Connect 32-bit Windows applications to APPX 5.5.2 using a single ODBC data source. Configure the gateway, import the DMO COUNTRY sample, then use the verified recipe for [WinSQL](#), [Excel](#) or [PowerShell](#).

Verified route: clean APPX 5.5.2, its matching gateway and ASQL package, and the supplied Dharma 9.01 driver. ASQL import, WinSQL, Excel and PowerShell pass the seven-record COUNTRY sample, including all nine field values. LibreOffice Base passes with its saved query in direct SQL mode; ordinary table browsing and parsed queries failed. Calc import and saved-file reopen/password/refresh pass all seven records and nine fields. Crystal Reports retrieves seven records and four fields through an ODBC (RDO) SQL command. Access import fails on repeated attempts with the supplied driver; the nearest verified Access success is the [5.5.3 configuration with Dharma 14.01 x86](#).

Use the driver and ASQL supplied in the **5.5.2 package**. The stock driver's catalog query passed in this Windows configuration. The 5.5.3 and 5.5.4 compatibility-driver instructions do not apply automatically to this edition.

[Check the tested versions and client bitness](#), [record the server names and ports](#), and [choose which components to install](#).

Requirements

- An installed and licensed APPX 5.5.2 system, an authorized APPX login, and its server name and login-manager port.
- The matching 5.5.2 ODBC gateway and ASQL packages, with the [runtime checks](#) if a component fails to start.
- Administrator access for installation and [System DSN](#) creation. Use an ordinary account for the client recipes.
- A 32-bit client: WinSQL32, Microsoft 365 Excel or Access, Crystal Reports 2016, LibreOffice Calc or Base, or Windows PowerShell.
- The DMO database and DMO application with the COUNTRY file, for the sample recipes.

Check Versions and Choose 32-bit Programs

Component	Tested version	Bitness
Windows	Windows 11 Pro, version 25H2, OS build 26200.6584	64-bit
APPX engine	5.5.2	32-bit
APPX ODBC gateway	5.5.2	32-bit
Client ODBC driver	Supplied Dharma 9.01.0000	32-bit
ASQL Dictionary Importer	From the APPX 5.5.2 package; executable version 5.4.7.0	32-bit
WinSQL, Excel, Access, Crystal Reports and PowerShell	Releases specified in their recipes	32-bit
LibreOffice Calc and Base	26.2.6.3, after an update from the original 25.8.7 x86 package	32-bit

Choose the driver and ODBC Administrator that match the **client application's bitness**. The engine and gateway are separate components connected over the network. The installation, ASQL import, WinSQL and Excel checks cover the combination above. Use the specified ODBC (RDO) SQL command for Crystal Reports and read the Access failure note before choosing that client. The LibreOffice recipes include profile registration and saved-file refresh. A different APPX release or bitness needs its corresponding installation and client recipe.

The tested configuration uses **Windows 11 Pro, version 25H2, OS build 26200.6584 (64-bit)**, verified on October 15, 2026. The clients are WinSQL 15, Microsoft 365 version 16.0.20430.20182, and Crystal Reports 2016 SP9 Patch 21. Every ODBC client and its driver in this recipe must be **32-bit**, even though Windows is 64-bit.

Which client routes worked

Client	Result for this 5.5.2 configuration	Procedure
WinSQL 32-bit	Seven records and all nine fields returned.	Query COUNTRY
Excel 32-bit	Fresh import and saved-workbook reopen/refresh passed; all seven records and nine fields agree.	Load and refresh
PowerShell 32-bit	Query objects, parameters, exports and diagnostics passed.	PowerShell recipes
LibreOffice Base 32-bit	Direct SQL saved query and fresh reopen passed; table browsing and parsed SQL failed -20217.	Base connection and query
LibreOffice Calc 32-bit	Native import, saved-file reopen/password/refresh and all seven records/nine fields passed.	Calc import and refresh
Crystal Reports 2016	ODBC (RDO) SQL command, saved report, PDF export and fresh reopen/refresh pass seven records and four fields.	Create a direct ODBC report
Access 32-bit	Import failed on repeated attempts. Nearest verified success: 5.5.3 with Dharma 14.01 x86 .	Read the Access failure

Matching and mixed versions

The **matching 5.5.2 gateway and supplied Dharma 9.01 driver** connected and read the SQL catalog in this environment. No client-driver replacement was needed for the verified ASQL, WinSQL and Excel route. Complete [ASQL import](#) and the [full-value sample comparison](#) before treating another combination as verified.

The tested official APPX engine contains the 5.5.2 build marker, compiled **June 22, 2024**. Its Windows file-resource version is the older 5,5,1,0 label; a folder name or that resource label alone does not identify the installed engine.

The ASQL executable in this 5.5.2 download reports 5.4.7.0. Keep the program delivered in the matching package; that file version does not mean that the wrong package was downloaded. Record the package hashes below when identifying an installation.

Use matching APPX engine and gateway releases as the starting point. A different driver or ASQL release needs its own [DSN](#) discovery, file listing, import and complete field-value checks. A newer ASQL program does not upgrade the gateway's field conversion or the APPX engine.

What Works Better in Later Releases

The later configurations below have passed their own tests. Follow the linked cookbook's complete engine, gateway, ASQL, driver and client setup before using its result. Replacing only the driver on this 5.5.2 gateway does not establish those improvements.

Capability or capacity	This edition	Verified later configuration and procedure
Access import	Repeated native imports failed with the supplied Dharma 9.01 x86 driver.	Import and all seven records/nine fields passed with matching 5.5.3 engine/gateway/ASQL, the separate official Dharma 14.01 x86 driver and Access 32-bit. Follow 5.5.3 Access import . The 5.5.4 Access recipe also passed its own configuration.
Reported table and name capacities	Supplied driver reports 500 table columns and 32-character column/identifier names.	The tested Dharma 14.01 driver with the matching 5.5.3 or 5.5.4 gateway reports 2,000 table columns and 64-character column/identifier names. Read 5.5.3 limits or 5.5.4 limits . These are reported capacities, not completed maximum-width import tests. Measured on 2026-10-20, the 5.5.3 and 5.5.4 gateways still refuse names over 32 characters with <code>-20248 Identifier too long</code> . The 6.2.1 64-bit gateway accepts names up to 64 characters with its 64-bit driver and 64-bit clients.
Column names that are SQL function names	Refused with <code>-20049 Keyword used for a name</code> , for example SUFFIX, PREFIX and LENGTH.	Accepted as plain names by the 5.5.3 and 5.5.4 engines and gateways with their tested Dharma 14.01 drivers, and by the 6.2.1 64-bit configuration, measured 2026-10-20. Those gateways refuse a few other words instead, such as DAY; compare 5.5.4 SQL name rules . A newer driver alone was not tested.

No SQL statement-length increase has been verified between these configurations. Each driver reports 35,000 characters, while execution probes fail below that advertised value. Use [this edition's limits and observed errors](#). Later 6.2.1 client configurations have not completed acceptance and are not presented as improvements here.

Identify the Server, Ports and Connection Names

Setting	Value in this cookbook	Value to use
APPX login-manager endpoint	<code>localhost:31053</code>	The APPX server and its configured login-manager port
APPX Windows interface	<code>Pipe</code>	The interface configured on the tested login manager
ODBC gateway endpoint	<code>localhost:1990</code>	The gateway server and its verified listening port
Gateway SQL dictionary	<code>appx</code>	The default dictionary created by the gateway installer
Windows System DSN	<code>AppxODBC</code>	The same DSN in every client recipe
APPX Database ID / Application ID	<code>DMO / DMO</code>	The installed DMO sample IDs
Imported SQL table	<code>DMO_DMO_COUNTRY</code>	The COUNTRY table created by ASQL

`localhost` means the computer running the program. On a separate Windows client, replace the DSN's Host with the gateway server's hostname. The gateway dictionary's APPX_DATA_SERVER identifies the APPX server, which can be a different computer.

Connection Glossary: Service, sqlnw, DSNs and Ports

Start here when a setup page uses an unfamiliar name. These definitions apply throughout this 5.5.2 32-bit cookbook. The port values below belong to this edition's tested configuration; use the site's actual server names and verified listeners.

Find a term: [Service field](#) · [sqlnw and the services file](#) · [Custom aliases](#) · [Check this computer](#) · [Host and endpoint](#) · [Database and DSN](#) · [Windows services and Services.msc](#) · [Gateway, Pipe and Socket](#) · [Ports](#).

Service field

The ODBC Service field is optional. Leaving it blank is a valid choice for the standard connection.

Blank does not disable networking or leave the port undecided. The tested Dharma driver first looks up the TCP service name `sqlnw` on the client computer. Entering `sqlnw` explicitly requests that same lookup. With the same mapping, blank and explicit `sqlnw` select the same port.

This is different from Services.msc. The ODBC Service field selects a network port by name. Services.msc manages Windows background programs. Leaving this ODBC field blank does not stop a Windows service, and entering `sqlnw` does not start one.

This cookbook's current connection recipe uses the **blank Service default** after its independent installer check. This is distinct from the default database name **appx**. Entering **appx** in Service requests a TCP service alias named **appx**; it does not select the APPX database. No **appx** TCP service alias is required for the standard recipe.

The shipped driver-install command creates the DSN with Host and Database but does not supply a Service value. An existing DSN can retain a previously configured value. Inspect the actual field rather than assuming that a screenshot of an existing installation shows an untouched installer default.

sqlnw and the Windows services file

sqlnw is a TCP service alias: a name used to look up a port number. It is the default network-service name used by the tested Dharma networking code. It is not an APPX user, SQL dictionary, Windows background-service name, executable or separate program to install.

Windows reads the name-to-port mapping from `C:\Windows\System32\drivers\etc\services` on the computer running the client. For this edition's verified gateway port, the relevant line is:

sqlnw	1990/tcp
-------	----------

This line means “when this computer looks up **sqlnw** for TCP, use port 1990.” It does not start the gateway, change its listening port, open a firewall or configure another computer. On a separate client and gateway, their services files are separate. Check each computer that performs a service-name lookup.

An existing wrong mapping can override the intended port even when Service is blank. Duplicate entries can select an unintended port; use one deliberate mapping for the selected alias. If no **sqlnw** mapping exists, the tested **Dharma 9.01 x86** client falls back to **TCP 1990**. That fallback is a measured property of this driver, not a promise that every APPX/ODBC release uses that port. Another edition can use a driver with a different fallback.

The matching gateway install script adds **sqlnw 1990/tcp** when its simple search finds no existing text containing **1990**. That check does not validate the existing service name or resolve duplicates. Verify the actual mapping and listener after installation.

Choosing a custom service name

Use a custom alias only when deliberately selecting a different gateway listener, for example when two gateways on the same host use different ports. An administrator can add **appx_reporting 21990/tcp** on the client and enter **appx_reporting** in the ODBC Service field. That example requires a gateway actually listening on TCP 21990 and the corresponding firewall access; adding the line alone does not create a working gateway.

Changing a mapping affects new connections that use that alias on this computer, including blank-Service connections that default to **sqlnw**. Changing one DSN to a distinct custom alias affects clients using that DSN. Preserve the original file and DSN before changing either. Close and reopen the client,

then verify the destination port and the sample query. Keep the field as a service name in these recipes; do not substitute a numeric port without separately checking that driver's handling of numeric Service values.

Check the Service setting on this computer

1. Check the application's connection settings to find the DSN it actually uses. The cookbook uses **AppxODBC**. A workbook, report or connection string can override saved DSN settings, so inspect its Host and Service overrides as well. For example, Excel can retain `SERVICE=sqlnw` after the DSN is cleared. Clear Service in the workbook's login dialog, complete a refresh and save the workbook to retain the empty setting. Do this on the computer running that client; one desktop's settings do not establish another's.
2. Open `C:\Windows\SysWOW64\odbcad32.exe` for a 32-bit application, or `C:\Windows\System32\odbcad32.exe` for a 64-bit application. Match the application architecture, even on 64-bit Windows.
3. Select **System DSN**, select the actual DSN and choose **Configure**. If the application uses a User DSN, check **User DSN** under that same Windows account instead. Read Host, Database and Service. Choose **Cancel** when finished inspecting so the check does not change the connection.
4. If Service is **blank**, the tested driver uses the `sqlnw` lookup. If it contains `sqlnw`, that alias was explicitly requested. If it contains **another name**, look up that name instead. Open `C:\Windows\System32\drivers\etc\services` read-only and inspect the matching active entry, including aliases after the port. Text following `#` is a comment. Check for conflicting or duplicate entries.
5. Compare the resulting port with the gateway's actual listener. If no matching `sqlnw` entry exists, this edition's measured client fallback is **1990**. Do not assume another custom name uses that same fallback. A valid port mapping alone does not prove that the intended gateway is running, reachable or able to authenticate.

For this edition's **32-bit System DSN on 64-bit Windows**, run the following in **64-bit PowerShell**. Replace AppxODBC in the registry path if the application uses another DSN. The commands read settings and active mapping lines; they do not change the DSN, services file, listener or firewall, and they display no password fields.

```
$dsn = Get-ItemProperty 'HKLM:\SOFTWARE\WOW6432Node\ODBC\ODBC.INI\AppxODBC'
$dsn | Select-Object Host, Database, Service
$alias = if ([string]::IsNullOrEmpty($dsn.Service)) { 'sqlnw' } else { $dsn.Service }
"TCP alias selected: $alias"
Get-Content "$env:WINDIR\System32\drivers\etc\services" |
  Where-Object { ($_ -split '#', 2)[0] -match '^(^|\s)' + [regex]::Escape($alias) + '(\s|
$)') }
```

An empty Service output cell means the DSN has no nonblank override. For the standard setup the displayed alias should be `sqlnw`, and its active line should select `1990/tcp` for this gateway. No output from the last command means no matching active mapping was found; use the separately verified fallback information above. A populated Service field containing another name is a custom choice, not an APPX database name.

For a **User DSN**, the Windows-user registry location is **HKCU**: `\SOFTWARE\ODBC\ODBC.INI\AppxODBC`; inspect it while signed in as the application user. For a **64-bit System DSN**, the location is **HKLM**: `\SOFTWARE\ODBC\ODBC.INI\AppxODBC`, without `WOW6432Node`. Microsoft documents the [ODBC Administrator and registry distinctions](#). A DSN-less connection has no saved DSN to inspect: check the application's connection string while keeping any credentials private.

Host, localhost and endpoint

Host names the computer running the ODBC gateway. A hostname is a name that DNS or the local computer resolves to an IP address. **localhost** means the computer running the program, not automatically the APPX server. Use localhost only when the required gateway is on that same computer.

An **endpoint** is a host plus its listening TCP port. Here the client contacts the gateway at `localhost:1990`. The gateway separately contacts the APPX login manager at `localhost:31053`, as recorded in `APPX_DATA_SERVER` inside `appx.dbs\appx.env`. A remote client changes its DSN Host to the gateway host; the gateway's backend endpoint can remain on another computer.

Database, dictionary, DSN and APPX IDs

Database appx selects the SQL dictionary created on the gateway as `appx.dbs`. A dictionary stores SQL definitions describing APPX files; it is not the TCP service alias and is not a copy of the business records. **ASQL Dictionary Importer** reads the APPX file definitions and creates the shared SQL metadata in that dictionary.

DSN means Data Source Name: the saved ODBC connection settings on the client. **AppxODBC** is the default DSN name used by these recipes. A **System DSN** is available to authorized applications across Windows user accounts; a **User DSN** belongs to one Windows user. Its driver architecture must match the client program. In Windows, SysWOW64 contains the 32-bit ODBC Administrator and System32 contains the 64-bit administrator.

DMO / DMO are the Database ID and Application ID inside APPX. **COUNTRY** is the APPX file. ASQL exposes it as `DMO_DMO_COUNTRY` in SQL. These identifiers have different purposes from the gateway dictionary `appx`, the DSN `AppxODBC`, and the TCP alias `sqlnw`.

Windows services and service accounts

A **Windows service** is a background program managed by Windows Service Control Manager. The gateway's installed Windows service is **Appx ODBC Data Gateway Server for Windows**. `sc.exe query` and `sc.exe start` take this Windows service name, not the ODBC Service field's `sqlnw` alias. Running means the background program started; verify its actual listening socket as well.

Services.msc belongs to this Windows-service context, not the ODBC Service field. Open it on the gateway computer to inspect or manage the gateway background program, and on the APPX server to inspect the login manager. Do not search Services.msc for `sqlnw` as the way to check a port alias. The alias belongs in the plain-text `services` file described above. Microsoft's [Service Control Manager reference](#) describes the Windows component that controls background services.

The APPX login manager is another Windows service. This cookbook's test name is **Appx-552-31053-OSAuth**. Its name identifies the managed login-manager process, while **31053** is the separate TCP port on which that process accepts gateway connections. Use the site's actual Windows service name in service commands and in the `HKLM\SYSTEM\CurrentControlSet\Services` registry path.

A **service account** is the Windows identity under which a background service runs. It needs the appropriate file and log-directory permissions. It is separate from the APPX user entered in an ODBC login. An ODBC Service field value does not select either identity.

Gateway, driver, listener, Pipe and Socket

The **ODBC driver** is the client DLL loaded by Excel, Access, WinSQL, PowerShell or another ODBC application. The **gateway** receives its SQL requests and obtains records from the APPX engine. A **listener** is the program accepting TCP connections on a specific port. A firewall rule can permit traffic to that port; it does not prove that the right program is listening.

Pipe and **Socket** in the APPX Windows login-manager configuration describe its interface to the APPX engine. This edition was tested with **Pipe**. That setting is separate from the desktop-to-gateway TCP connection and its ODBC Service field. Do not change it merely to match another edition's recipe.

Port reference for this edition

Port or name	Meaning	What to check
TCP 1990	Tested ODBC gateway listener for APPX 5.5.2	Match the gateway process PID to its listener, then check the client's <code>sqlnw</code> mapping or chosen custom alias.
TCP 1990	Fallback in this edition's tested Dharma 9.01 x86 client when <code>sqlnw</code> has no mapping	A fallback is driver-specific; it does not override a conflicting named mapping.
TCP 31053	APPX login-manager endpoint in this cookbook	Match <code>APPX_DATA_SERVER</code> to the running login-manager service and intended APPX engine.

Port or name	Meaning	What to check
TCP 1990 / TCP 1992	Different ports used by the tested driver/gateway generations	Do not assume they are interchangeable or open both as a substitute for checking the actual configuration.
TCP 8060	Conventional APPX login-manager port in APPX installation references	This cookbook uses 31053 instead; retain the site's configured port.
sqlnw	Default TCP service alias used by the tested Dharma client	It has meaning through a local mapping or the driver's fallback; it is not a Windows service name.
appx	Default SQL dictionary name	Enter it in Database, not Service, for the standard setup.

Use the [gateway Windows-service and listener check](#), the [client DSN check](#), and the [complete COUNTRY result](#) to verify the connection. A successful TCP probe alone does not verify authentication, imported metadata or correct field values.

Decide What to Install on Each Computer

Role	Work to complete
Gateway administrator	Install and configure the gateway, then maintain its shared SQL dictionary.
APPX administrator or developer	Use ASQL to make the required APPX file definitions available through that dictionary.
Excel, Access, WinSQL or report user	Install the client driver and configure <code>AppxODBC</code> ; follow the relevant client recipe after the administrator has imported the definitions.

Routine users do **not** import the same definitions again on every desktop. [Install the gateway and ASQL](#), then read [what ASQL does and who needs it](#) before [importing COUNTRY](#).

Complete the [gateway setup](#) and [data-source check](#), then [check the sample](#) before moving to a client recipe. Each recipe includes its expected result.

Download the Tested Example Files

The [offline examples ZIP](#) includes both [PowerShell helpers](#), the [read-only COUNTRY SQL](#), [seven expected records](#), [nine column descriptions](#), the checked [Base connection and saved query](#), and the [Calc workbook](#). Read the [offline README](#) and use [SHA-256 checksums](#) to check the extracted files.

These are working examples for this edition, not installers. The Calc workbook contains saved sample values. Refreshing it requires the matching driver, `AppxODBC`, a live gateway, your APPX login, and the [registered Base connection](#). The Base file stores its ODBC connection and query; it does not contain an embedded copy of the APPX database. No saved password is included.

The PDF edition uses **PDF/A-3u** and embeds these examples individually and in ZIP form, along with its screenshots and fonts. In a PDF viewer with attachment support, open the **Attachments** panel and save the required files. A browser's PDF preview may hide that panel; download the PDF and use a viewer that exposes attachments. Files do not run automatically. The web download links above provide the same examples.

CHAPTER 2

Install and Check APPX ODBC

Install the gateway and driver, then check the connection settings.

Install the 5.5.2 Gateway and ASQL

1. Confirm that APPX 5.5.2 is installed and licensed and that an authorized user can open its DMO application. Check the engine's actual version if an existing folder has been upgraded. This example uses a clean official 32-bit 5.5.2 installation at `C:\appx\Appx.552` and its login manager on `localhost:31053`. The usual APPX port is 8060; use the configured port at your site.
2. Download the [official 5.5.2 gateway and 5.5.2 driver and ASQL package](#). The APPX download archive also provides the matching server installation.
3. Verify the downloads from Command Prompt, one command at a time:

```
certutil.exe -hashfile "AppxODBCserver_windows.exe" SHA256
certutil.exe -hashfile "AppxODBCdriver+ASQL_windows.exe" SHA256
```

Download	SHA-256
AppxODBCserver_windows.exe	12f7a0d0f06cd0a457356458d4282a45f46cecd8aeb0202e34b5fe55a1e91a58
AppxODBCdriver+ASQL_windows.exe	c378ff56bb39b6745adbae2efef15038b5f0fa9e5cb309d4691d1c71cb767947

4. Right-click the gateway installer and select **Run as administrator**. Select **Next** and keep `C:\Program Files (x86)\AppxODBCserver` and the default Start menu folder.

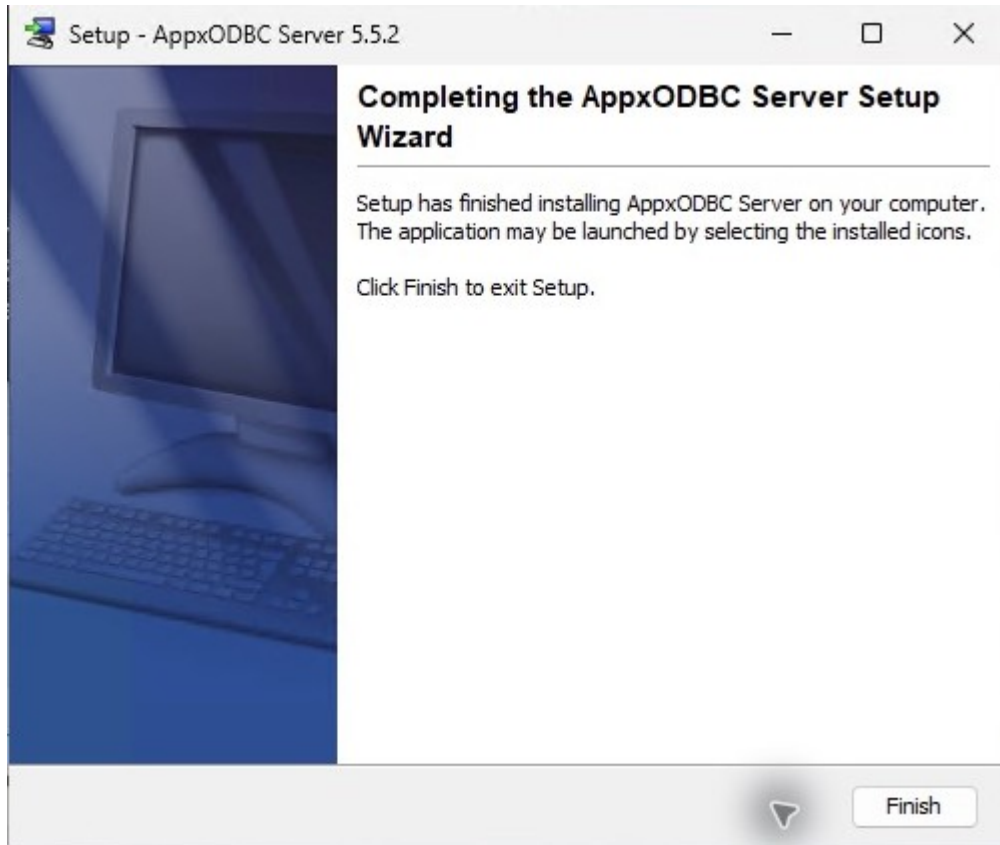


Figure 1. The matching APPX ODBC 5.5.2 gateway installer completes.

5. Enter the APPX server's hostname and login-manager port. The example uses `localhost` and `31053`. These fields identify the APPX backend, not the ODBC gateway port. Complete the wizard.

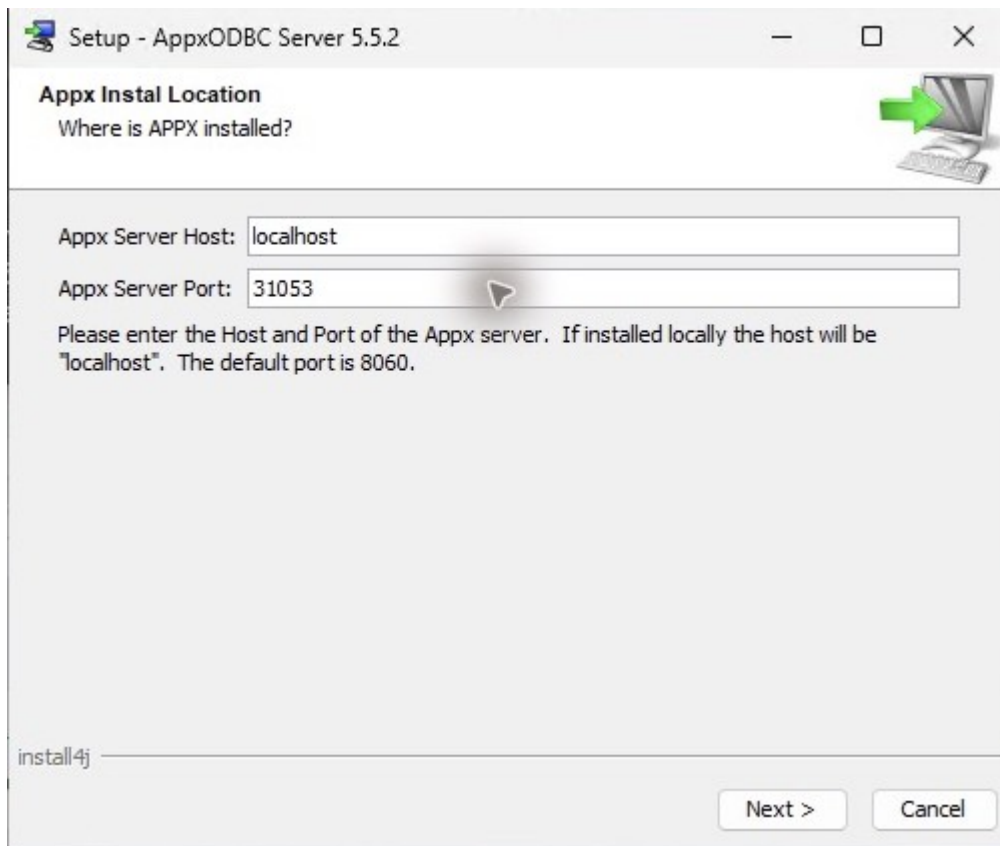


Figure 2. Point the gateway to the intended APPX login manager.

6. On the ASQL administration computer, run the matching driver and ASQL installer as administrator. Keep `C:\Program Files (x86)\AppxODBCDriver` and the default Start menu folder.

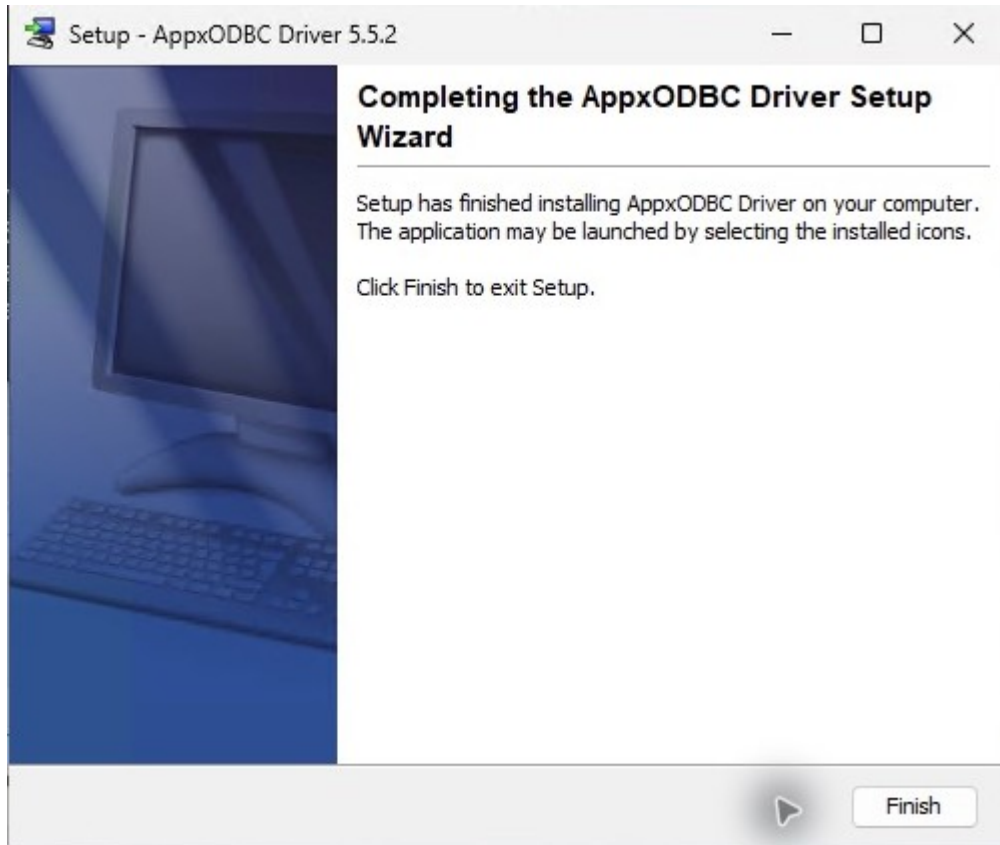


Figure 3. The matching 5.5.2 driver and ASQL installer completes.

7. Keep **Create System DSN** selected. Enter the gateway hostname, `localhost` when it is on this computer. Keep Data Source Name **AppxODBC** and Database **appx**. Complete setup, then [check the gateway Windows service and DSN](#).

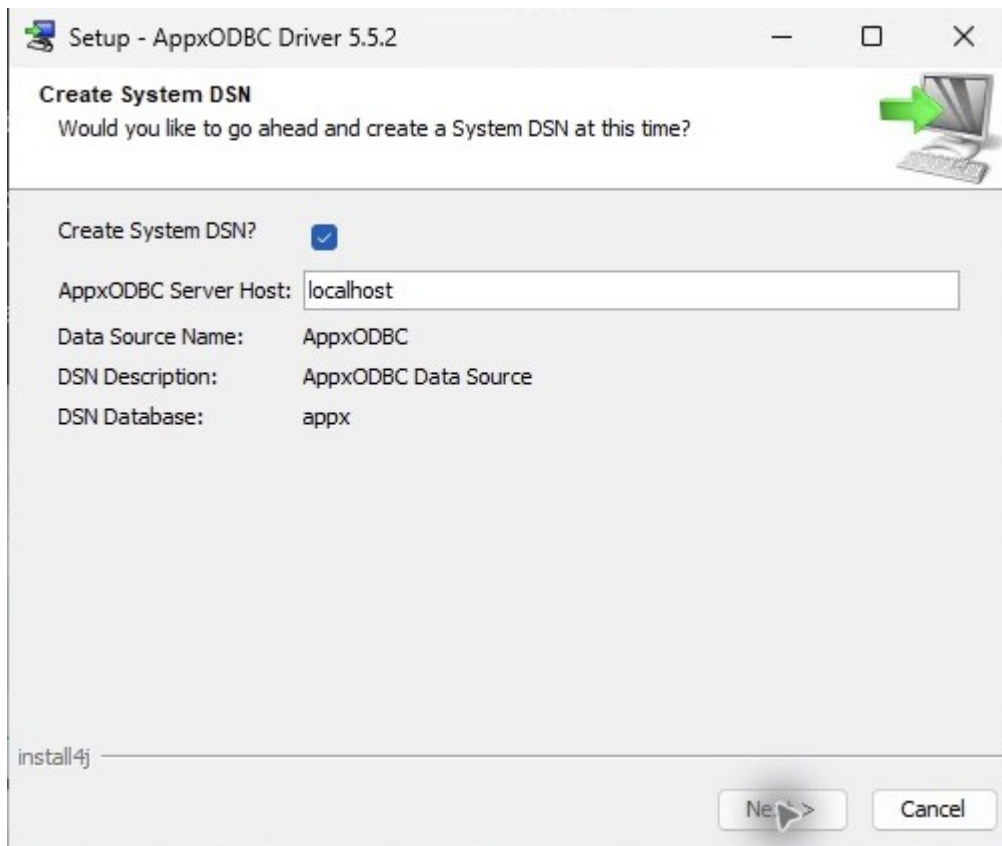


Figure 4. Keep the default AppxODBC data source and appx dictionary.

The installer creates the default dictionary; an additional `mdcreate` command is unnecessary. Preserve an existing gateway dictionary and configuration before replacing an installation. Use the [ASQL import recipe](#) after the connection checks.

Start the Gateway and Check Required Microsoft Files

Start and check the gateway

Here **service** means the Windows background service named **Appx ODBC Data Gateway Server for Windows**. The `sc.exe` commands below manage that program; they do not take the **ODBC Service** field's `sqlnw` alias. The alias instead selects a TCP port through the local services file. Read [Windows services and service accounts](#) and the optional **ODBC Service** field.

Open **Command Prompt** as **administrator**. Run the following separately:

```
sc.exe qc "Appx ODBC Data Gateway Server for Windows"
sc.exe start "Appx ODBC Data Gateway Server for Windows"
sc.exe queryex "Appx ODBC Data Gateway Server for Windows"
netstat.exe -ano -p tcp
```

Check that `BINARY_PATH_NAME` is `C:\Program Files (x86)\AppxODBCserver\bin\AppxODBC.exe`. The expected state is **RUNNING**. Match its PID to the listening TCP socket. With a single `sqlnw 1990/tcp` mapping, the exact 5.5.2 package above listened on **1990**, matching the installer default. An inherited `sqlnw` mapping can select a different port. Check the actual listener and keep the client and gateway mappings consistent. [Match the client mapping to the listener](#).

The freshly installed gateway started with its packaged libraries, including ICU 51. No Boost library or ICU 59 replacement was added. Apply a dependency repair only when a startup error identifies the need.

If a component will not start

The matching 5.5.2 binaries and reviewed release-5.x build projects identify these Microsoft runtime families:

Component	Runtime family identified	Official Microsoft package
Gateway and ASQL	Visual C++ 2005, <code>MSVCR80.dll</code> ; ASQL also imports <code>MSVCP80.dll</code>	Visual C++ 2005 SP1 MFC Security Update
Packaged ICU 51 libraries	Visual C++ 2010, <code>MSVCR100.dll</code> and, in some libraries, <code>MSVCP100.dll</code>	Visual C++ 2010 SP1 MFC Security Update
Supplied Dharma 9.01 driver	Windows <code>MSVCRT.dll</code> and system libraries	Use the matching APPX driver package and check the full loader error .

Select `vcredist_x86.exe`, even on 64-bit Windows. The 2005 and 2010 downloads have the same filename; save them in separate folders. Check the Microsoft digital signature, run the appropriate installer and restart if requested. These are legacy runtime packages, and the download pages list older supported operating systems. They identify the dependency family; they do not certify this APPX release for Windows 11.

The test computer already had Visual C++ 2010 x86 **10.0.40219** installed. The gateway package includes additional runtime files. This test does not establish that a clean Windows installation needs no runtime installation, or identify the oldest acceptable runtime patch. The newer **v14** redistributable used in other cookbook editions is not a substitute for VC80 or VC100.

If Windows reports a side-by-side configuration or missing-DLL error, retain the complete message and check **Event Viewer** → **Windows Logs** → **Application**. Use the named dependency and its architecture to choose a repair; do not download individual runtime DLLs from unrelated sites.

Install the Driver on Another Windows Computer

A desktop connects to the **gateway**, not directly to the APPX login manager. Use the supplied 32-bit 5.5.2 driver with the verified client recipes in this edition. A 64-bit Office or LibreOffice process cannot load this driver, even when the 32-bit **DSN** exists on the computer.

1. Record the gateway's hostname and measured listening port from [the gateway Windows-service/listener check](#). The cookbook gateway uses TCP 1990.
2. Download the [matching 5.5.2 driver and ASQL package](#) and compare its SHA-256 with the [package table](#). The combined package is the one tested here. A routine user does not need to run ASQL or repeat the shared import.
3. Run the installer as administrator and keep the default **AppxODBCDriver** folder. Keep **Create System DSN** selected. Enter **AppxODBC**, the gateway server's hostname, and Database **appx**. Use **localhost** only when the gateway is on this desktop.
4. Back up this desktop's [Windows services file](#) and ensure **sqlnw** resolves to the gateway's actual port. The client and gateway have separate local services files; editing one does not edit the other. Resolve conflicting or duplicate mappings deliberately.
5. Open **C:\Windows\SysWOW64\odbcad32.exe**, select **System DSN** → **AppxODBC** → **Configure**, and check Host, Database and Service against [the default data-source recipe](#). The optional **ODBC Service** field can be left blank or set explicitly to **sqlnw**. Both select the TCP port through the client's **sqlnw** mapping; [definition and consequences](#).
6. Close and reopen client programs after installing or changing a driver. Run the [credential-prompting PowerShell query](#) before creating a workbook or report. Verify both the count and [complete sample values](#).

The separate desktop deployment is the same client setup with a different Host; the recorded acceptance tests ran on the gateway computer. A remote desktop also needs DNS resolution, firewall reachability and an authorized APPX account. Use [the two-hop connection checks](#) to isolate that path. Do not copy the gateway program or its SQL dictionary to every desktop.

Check the Client-to-Gateway and Gateway-to-APPX Connections

Connection	Initiating computer	Destination	Sample TCP port	Configuration
ODBC client to gateway	Windows desktop running Excel, Calc, PowerShell or another client	Gateway server	1990	DSN Host and Service; local sqlnw mapping
Gateway to APPX	Gateway server	APPX server's login manager	31053	<code>appx.dbs\appx.env</code> , <code>APPX_DATA_SERVER</code>

These endpoints may be on one computer, two computers, or three. **Pipe** is the tested login manager's interface to its Windows APPX engine; it does not remove either network **endpoint** above. Use the site's configured ports rather than changing them to match this laboratory example.

On the desktop, run this in PowerShell, replacing the hostname:

```
Test-NetConnection gateway.example.internal -Port 1990
```

On the gateway server, check its APPX **endpoint**:

```
Test-NetConnection appx.example.internal -Port 31053
```

TcpTestSucceeded : True establishes a TCP connection to that address and port. It does not test the APPX login, SQL dictionary, table definitions or field conversion. Run the **COUNTRY query** for those checks. A failed first hop directs attention to the gateway listener, DNS, routing and the client-facing firewall; a failed second hop directs attention to the login manager and gateway-to-APPX path.

Configure firewall access for the required clients and backend connection through the site's normal administration process. Do not open a login-manager or gateway port to unrelated networks merely to troubleshoot a local **DSN**. The **services** file resolves a name to a port; it does not start a listener or create a firewall rule.

Check the Saved AppxODBC Connection (DSN)

Use the default dictionary

The gateway installer creates the **appx** SQL dictionary and writes the APPX server **endpoint** to `appx.dbs\appx.env`. Use that default dictionary. No additional **mdcreate** command is required for this recipe.

The default names refer to different things:

Name	Purpose
appx	SQL dictionary on the gateway
AppxODBC	ODBC data source on the Windows client
DMO / DMO	Database ID and Application ID inside APPX

On an existing test installation, preserve the current **DSN** before replacing it under the same **AppxODBC** name. Keep the existing **appx** dictionary and import **COUNTRY** into it with **ASQL**.

Check the APPX login-manager interface

The tested 5.5.2 login manager uses **Pipe** and listens on TCP 31053. Preserve the working interface when following this edition. The ODBC gateway still connects to the login manager over its network **endpoint**; Pipe is the Windows login manager's engine interface.

Check the **APPX login-manager Windows service** and intended engine path before diagnosing a failed connection. Do not change a production login manager's interface merely because another APPX release uses a different setting.

Check the client driver and DSN

Leaving the **ODBC Service** field blank is valid for the standard connection. Blank uses the default **sqlnw** TCP-name lookup; entering **sqlnw** explicitly uses that same lookup. Leave Service **empty** for this recipe, as in the current login screenshots. The supplied driver installer omits this optional value; entering **sqlnw** is an explicit alternative. Keep Database **appx**: it selects the SQL dictionary and does not belong in Service. With either blank or explicit **sqlnw**, verify that the client's mapping reaches this gateway on TCP 1990. Read **Service, its blank default and custom aliases** before changing the field.

Use the supplied 5.5.2 driver installed above. Then check the gateway port mapping and the data source below.

1. On the client computer, back up `C:\Windows\System32\drivers\etc\services` and ensure that the **sqlnw** mapping resolves to the verified gateway port. For this cookbook the mapping is:

sqlnw	1990/tcp
-------	----------

Resolve duplicate or conflicting **sqlnw** entries before proceeding. Leave the **ODBC Service** field **blank** for this recipe. Entering the TCP alias **sqlnw** explicitly is also valid. Both use this mapping to reach TCP 1990; neither starts the gateway or changes its listener. Use a different named alias only for a deliberately configured alternate port. **How the services file and fallback work.**

2. Open `C:\Windows\SysWOW64\odbcad32.exe` and check **System DSN** for **AppxODBC**. This is the 32-bit ODBC Administrator. The program in System32 manages 64-bit drivers.
3. Select **Configure**. Keep Host as the gateway hostname, Database **appx**, and optional **ODBC Service blank**. Entering **sqlnw** explicitly is an alternative using the same TCP-name lookup. Keep the supplied **AppxODBC Driver**. Its DLL is `C:\Program Files (x86)\AppxODBCDriver\dhsodbc.dll`; Windows may display the equivalent short path. Driver version is **09.01.0000**.

Next, **import the COUNTRY metadata with ASQL**. An empty dictionary cannot answer the sample queries until that import completes.

CHAPTER 3

Make APPX Files Available to SQL

Use ASQL to create and update SQL table definitions for APPX files.

What ASQL Does and Where to Find It

ASQL Dictionary Importer is an **administration tool**. It reads APPX file definitions and generates SQL definitions for the gateway: column names and types, indexes, and public synonyms. The generated tables use `STORAGE_ATTRIBUTES 'metadata_only'`. The business records remain in APPX and are read when an ODBC query runs. This is different from Access's **Import**, which copies records into an Access database.

The definitions are stored in the gateway's `appx.dbs` dictionary. One administrator imports them for everyone connecting to that gateway dictionary. Each desktop needs the appropriate ODBC driver and its own `AppxODBC` configuration; each desktop does **not** need its own ASQL import.

For this cookbook, keep ASQL on one administrator-maintained Windows computer. The gateway server is a convenient choice when interactive administration is allowed; an authorized administration workstation can also connect remotely. ASQL does not need to remain running for Excel or other clients to query APPX. Routine user desktops need only the driver; use the driver-only package when provided for that release, reserving the driver-and-ASQL package for administrators. APPX's [ODBC installation instructions](#) describe this separation between driver-only desktops and the administration installation.

A shared definition does not replace APPX authorization. Use each user's authorized APPX login and the site's APPX access controls. Removing ASQL from a desktop is an installation choice, not a complete security boundary against SQL administration tools.

Add the COUNTRY Table to SQL with ASQL

The matching ASQL program imported COUNTRY through the supplied 5.5.2 driver and default data source. The complete seven-record, nine-field result matches the native APPX sample.

1. Open **ASQL Dictionary Importer** from the installed 5.5.2 package.
2. Choose **AppxODBC**, enter an authorized APPX user and password, and enter **DMO** for both Database ID and Application ID.
3. Select **OK**. The DMO dictionary tree appears.



Figure 5. ASQL connects to the default AppxODBC source with DMO sample IDs.

ASQL selects the **AppxODBC DSN**, which supplies Host, Database and optional Service. **DMO** in this dialog selects the APPX database and application to import; it is separate from the SQL dictionary **appx**. [Meaning of these names.](#)

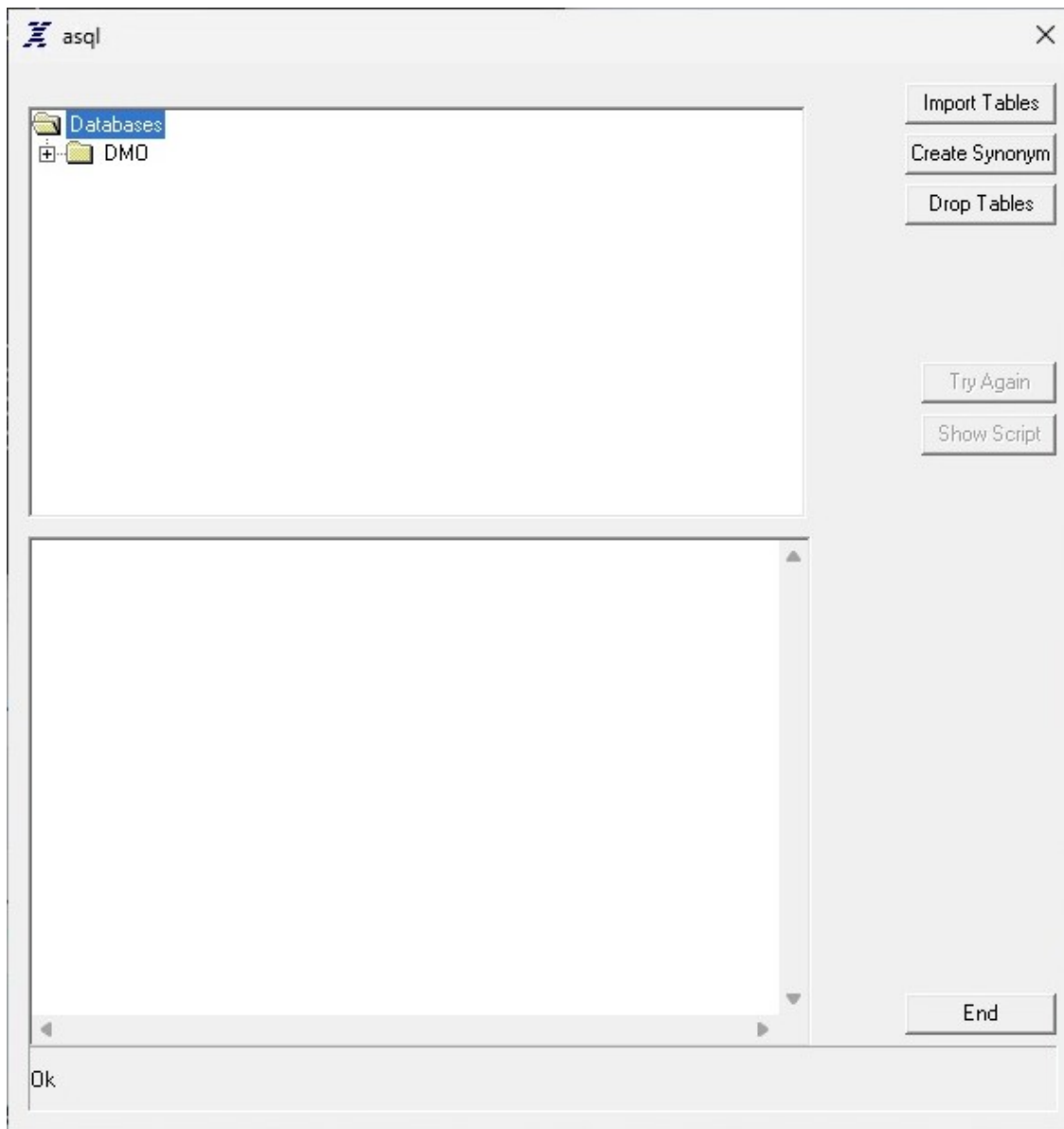


Figure 6. The matching ASQL program lists the DMO database.

4. Expand **DMO** → **DMO**, select **COUNTRY**, and select **Import Tables**.
5. Wait for the check mark beside **COUNTRY** and **Ok** in the status line.

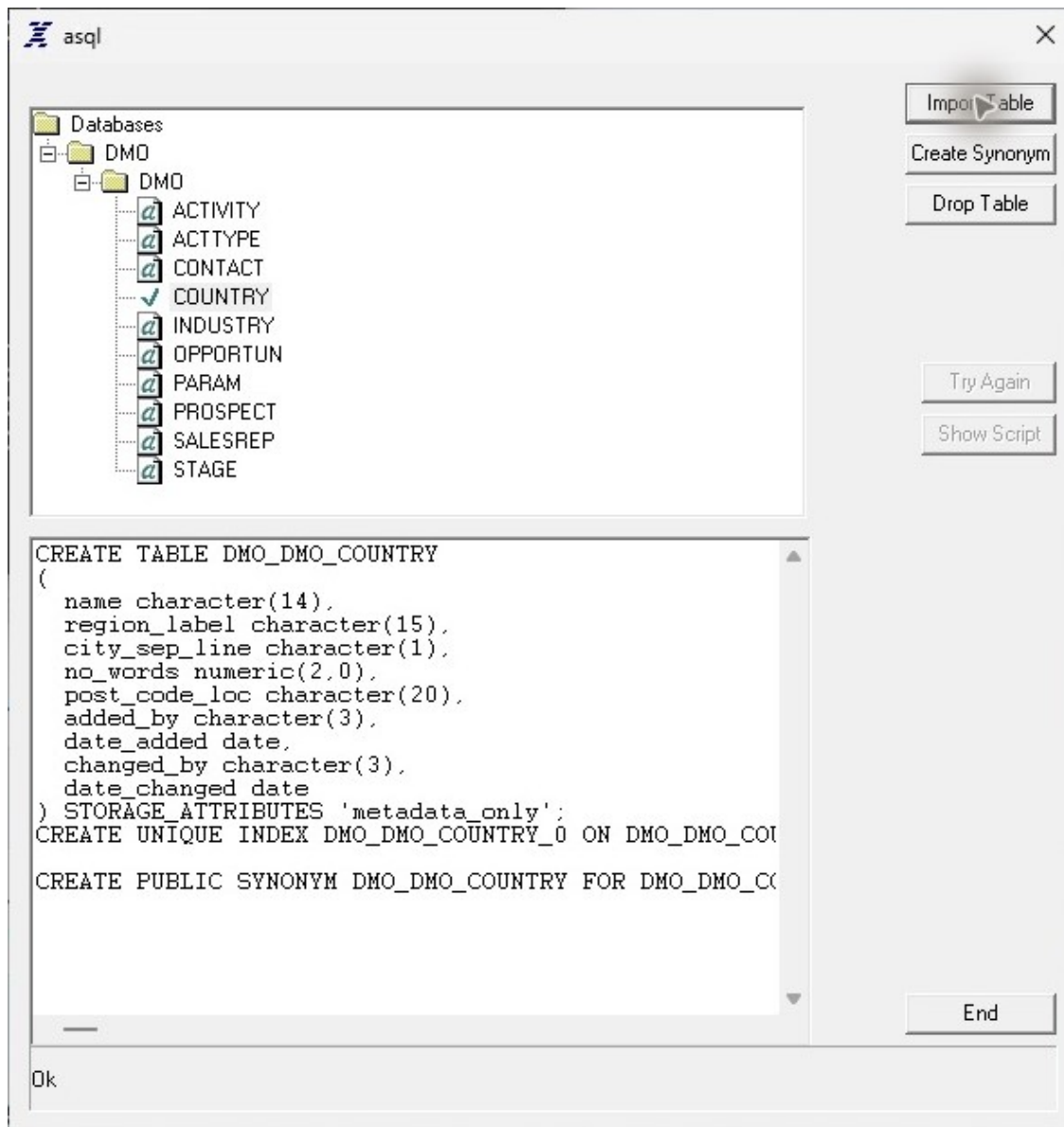


Figure 7. COUNTRY imports successfully with the supplied 5.5.2 driver.

ASQL creates the SQL table metadata, index and synonym. It reads the records from the APPX file when the table is queried; this dictionary import does not copy the business records into a separate SQL database.

Check Every COUNTRY Record and Field

Execute these as separate statements, using [WinSQL](#) or the [PowerShell recipe](#):

```
SELECT COUNT(*) FROM DMO_DMO_COUNTRY
```

```
SELECT * FROM DMO_DMO_COUNTRY ORDER BY name
```

The unmodified test sample contains **7 records and 9 columns**. Its country names are Canada, France, Ireland, Netherlands, UK, USA and United States. USA and United States are separate records in this sample.

name	region_label	no_words	post_code_loc
Canada	Province	2	AFTER REGION
France	County	1	SEPARATE LINE
Ireland	County	0	
Netherlands		2	BEFORE CITY
UK	County	2	SEPARATE LINE
USA	State	1	AFTER REGION
United States	State	1	AFTER REGION

The other columns are `city_sep_line`, `added_by`, `date_added`, `changed_by` and `date_changed`. Some audit dates and text values are empty. Compare the complete rows, including those empty dates, with the APPX COUNTRY file. A count alone does not establish that the values are correct.

If the DMO sample has been changed, compare the query result with the current APPX file. Test other applications and data types with representative records before relying on them.

Show Only the APPX Application You Need

For the COUNTRY recipe, enter **DMO** as both Database ID and Application ID before connecting. These optional fields restrict the definitions ASQL discovers. They are distinct from the gateway's **appx** SQL dictionary and the Windows **AppxODBC** data source.

Login fields	Definitions requested
Database DMO, Application DMO	The DMO application's files in database DMO
Database DMO, Application blank	Applications available within database DMO
Database blank, Application DMO	Matching DMO application entries across available databases
Both blank	The available System Directory entries across databases and applications

In the reviewed 5.5.2 source, ASQL prepares a System Directory query based on these fields, fetches its results into a list, and then builds the tree. Leaving both fields blank requests the whole available list. Expanding a folder afterward is not the same as issuing a narrowly filtered first discovery query. Start with known IDs for a focused import, especially on an installation containing many applications or files.

If the window remains at **Query dictionary...** or **Fetching results...**, record the stage and elapsed time. Check a fresh **small SQL query**, reconnect with the known database/application IDs, and compare the result. This source review explains the larger query scope; it does not establish a measured network latency cause or a universal timeout. Keep a slow-discovery investigation separate from changing business-file metadata.

Do not repeatedly choose an application or database folder's **Import Tables** command just to make the tree appear. Discovery lists available definitions; import changes the shared gateway dictionary. Follow the **individual COUNTRY import** once discovery completes.

When to Create or Update SQL Table Definitions

Situation	Action
A file has not been exposed through this gateway dictionary	Import its definition once.
A new or replacement gateway dictionary has no definitions	Import the files needed by its users.
APPX fields, field order, occurrences, storage layout or key definitions change	Have the administrator review and replace the affected SQL metadata after the APPX dictionary/structure changes are complete.
Records are added, changed or deleted in an existing APPX file	Query or refresh the client; an ASQL import is not needed just to obtain current records.
Another user installs Excel, Access or the ODBC driver	Configure that desktop's DSN ; do not repeat the shared import.
An Access local copy from a supported configuration is stale	Repeat that configuration's client data import; ASQL does not refresh a local copy. This edition's Access import fails as documented.

Importing changes shared SQL metadata. A generated metadata import does not copy or rewrite the native APPX records, but replacing definitions can interrupt queries, change names used by reports, or discard custom SQL edits. Treat re-import as administration rather than a routine Refresh button.

Before replacing an existing definition, preserve the gateway dictionary and any customized SQL, identify dependent reports or views, and arrange a quiet interval for the affected users. Select the individual file in ASQL. Selecting a database or application folder can apply an operation to many files.

Do not assume that every ASQL release automatically replaces an existing table. The 5.5.2 source executes generated **CREATE** statements and stops at a failing statement. If the existing definition causes **Table/View/Synonym already exists**, have the administrator perform a controlled metadata replacement, then import and verify the complete rows again. Avoid using **Drop Tables** on a parent folder. The **historical import manual** explains the need to re-import after structure changes; its automatic-replacement description should not be assumed for this package.

How APPX Files and Fields Become SQL Tables and Columns

When ASQL imports an APPX file, it creates a SQL table definition for it. The records stay in the APPX file; the definition tells the gateway how to read them. The gateway writes the definition the same way in every edition.

In APPX	In SQL	Example
A file	A table named for the database, application and file, joined by underscores	File COUNTRY in application DMO and database DMO is the table <code>DMO_DMO_COUNTRY</code> .
A field	A column, in the same order as the fields in the record	The field REGION LABEL is the column <code>region_label</code> .
A field with occurrences	One column for each occurrence, numbered from 1	A field AMOUNT with three occurrences becomes <code>amount_1</code> , <code>amount_2</code> and <code>amount_3</code> .
A key	An index named for the table and the key's number, unique unless the key allows duplicates	Key 0 of COUNTRY is the unique index <code>DMO_DMO_COUNTRY_0</code> on <code>name</code> .
A group, substring or synonym field	Nothing	See below.

Each import also creates a public synonym with the same name as the table.

How field names become column names

- Letters become lower case. Spaces, hyphens and any other character that isn't a letter, digit or underscore become underscores. REGION LABEL becomes `region_label`, and CITY-STATE becomes `city_state`.
- Many applications start every field name with the file's name. ASQL drops that part, so in a file named CUSTOMER a field named CUSTOMER NAME becomes `name`. It keeps the whole name when what's left would be one of the gateway's keywords or would start with a digit, so CUSTOMER NO stays `customer_no`, because NO is a keyword.
- ASQL doesn't check the whole name against the reserved words, and it doesn't change a name that starts with a digit. Either one stops the import. See [SQL Name Rules and Reserved Words](#), then [rename the column in the script](#).
- Two different field names can end up the same, for example CITY-STATE and CITY STATE. The import then stops with `-20010 Duplicate column specification`. Rename one of them in the script.

Group, substring and synonym fields don't become columns

These fields describe space in the record that other fields already cover, so ASQL leaves them out.

- A group is a name for several fields together. Each field inside it gets its own column.

- A synonym is another name for the same space as another field, so its data is in that field's column. Adding a synonym field in APPX doesn't add a SQL column. To show a column under another name, rename it in the script or [use a view](#).
- A substring is part of another field. To show that part, use the SUBSTRING function in a query or a view. See [SUBSTRING in the ODBC and SQL Reference](#).

Data types

APPX field	SQL type
Alpha or logic	<code>character(n)</code> , where n is the field's length in the record.
Date	<code>date</code> , <code>time</code> or <code>timestamp</code> , depending on which parts the field stores. Parts the field doesn't store are filled in from the current date and time.
Binary number of 1, 2 or 4 bytes	<code>tinyint</code> , <code>smallint</code> or <code>integer</code>
Packed number	<code>numeric(p,s)</code> , with p digits in all and s of them after the decimal point

The gateway finds the APPX file from the table's name, so don't change the table name in the script. For a friendlier name, use a view.

SQL Name Rules and Reserved Words

A table, column or view name can be written plain, the way ASQL writes them, or inside double quotes.

Plain names:

- start with a letter or an underscore,
- contain only letters, digits and underscores,
- are at most 32 characters long,
- can't be one of the reserved words listed below, and
- ignore capitals: `Region_Label` and `region_label` are the same column.

A longer name fails with `-20248 Identifier too long.` , quoted or not. See [What Works Better in Later Releases](#) for a later configuration with longer names.

These names were tried on this edition's gateway:

Name	Result
<code>name_2</code> , <code>_name</code> , <code>RegionLabel</code>	Accepted.
<code>2name</code>	<code>-20003 Syntax error. "2name" works.</code>
<code>region-label</code> , <code>region label</code>	<code>-20003 Syntax error.</code> Both work in double quotes.
A name containing <code>\$</code> , <code>#</code> , <code>@</code> or <code>.</code>	<code>-20003 Syntax error.</code> It works in double quotes.

Name	Result
A name of 33 characters or more	-20248 Identifier too long.
A table name that starts with a digit, such as 0SA_DMO_FILE	-20003 Syntax error. It works in double quotes.

A name in double quotes can use any characters and can be a reserved word, but it keeps its exact capitals, and every statement has to quote it the same way. A column created as "suffix" is found by "suffix" but not by "SUFFIX", which fails with -20006 Column not found/specified. A plain name is read in lower case, so quote in lower case, the way ASQL writes names. Renaming a column is usually simpler than quoting it; see [Rename a column before it is created](#).

Reserved words

The gateway refuses two kinds of words as plain names, and they behave differently:

- **Refused everywhere**, with -20003 Syntax error. These are part of SQL itself, such as NUMBER, TYPE, DATE, ORDER and FILE. A column with one of these names has to be in double quotes in every statement that uses it, including every query, report and spreadsheet. Rename the column instead.
- **Refused only as a new name**, with -20049 Keyword used for a name. These fail where a table or view is created, but queries can use them without quotes. Quoting the name once in the CREATE TABLE statement, in lower case, is enough.

This edition's gateway uses the Dharma 9.01 SQL server. It refuses every word in Dharma's own reserved word table except RANGE, including function names such as SUFFIX, PREFIX and LENGTH. The 5.5.3, 5.5.4 and 6.2.1 gateways use Dharma 14.01, which accepts most function names but refuses a few words this one accepts, such as DAY, BOTH and LEADING. Check the list again before moving to a later release. See [What Works Better in Later Releases](#).

ASQL's **Reserved Keywords** list, shown in expert mode, is only a guide. It combines the ODBC standard's keyword list, ASQL's own list and the driver's list, so it includes many words this gateway accepts, such as DAY, LEVEL and POSITION. The tables below show what this edition's gateway refused when each word was tried as a column name.

Refused everywhere: 210 words. Without double quotes these fail in any statement, queries included, with -20003 Syntax error.

Starts with	Words
A	ADD, AFTER, ALL, ALTER, AND, ANY, AS, ASC, AVG
B	BEFORE, BETWEEN, BIGINT, BINARY, BIT, BY
C	CALL, CASCADE, CASE, CAST, CHAR, CHARACTER, CHECK, CLOSE, CLUSTERED, COALESCE, COLGROUP, COLLATE, COLUMN, COMMIT, COMPRESS, CONNECT, CONSTRAINT, CONTAINS, COUNT, CREATE, CROSS, CURRENT, CURSOR

Starts with	Words
D	DATABASE, DATAPAGES, DATE, DBA, DEC, DECIMAL, DECLARE, DEFAULT, DEFINITION, DELETE, DESC, DISTINCT, DOUBLE, DROP
E	EACH, ELSE, END, ESCAPE, EXCLUSIVE, EXECUTE, EXISTS
F	FETCH, FILE, FLOAT, FOR, FOREIGN, FROM, FULL
G	GRANT, GROUP
H	HASH, HAVING
I	IDENTIFIED, IN, INDEX, INDEXPAGES, INDICATOR, INNER, INOUT, INSERT, INT, INTEGER, INTERSECT, INTO, IS
J	JOIN
K	KEY
L	LEFT, LIKE, LINK, LOCK, LOG, LONG, LVARBINARY, LVARCHAR
M	MAX, METADATA_ONLY, MIN, MINUS, MODE, MODIFY, MONEY
N	NATIONAL, NATURAL, NCHAR, NEWROW, NOCOMPRESS, NOT, NOWAIT, NULL, NULLIF, NUMBER, NUMERIC
O	OF, OLDROW, ON, OPEN, OPTION, OR, ORDER, OUT, OUTER
P	PCTFREE, PRECISION, PRIMARY, PRIVILEGES, PROCEDURE, PUBLIC
R	RAW, REAL, REFERENCES, REFERENCING, RENAME, RESOURCE, RESTRICT, RESULT, RETURN, REVOKE, RIGHT, ROLLBACK, ROW, ROWID, ROWNUM
S	SELECT, SET, SHARE, SIZE, SMALLINT, SOME, SPACE, SQL_BIGINT, SQL_BINARY, SQL_BIT, SQL_CHAR, SQL_DATE, SQL_DECIMAL, SQL_DOUBLE, SQL_FLOAT, SQL_INTEGER, SQL_LONGVARBINARY, SQL_LONGVARCHAR, SQL_NUMERIC, SQL_REAL, SQL_SMALLINT, SQL_TIME, SQL_TIMESTAMP, SQL_TINYINT, SQL_TSI_DAY, SQL_TSI_FRAC_SECOND, SQL_TSI_HOUR, SQL_TSI_MINUTE, SQL_TSI_MONTH, SQL_TSI_QUARTER, SQL_TSI_SECOND, SQL_TSI_WEEK, SQL_TSI_YEAR, SQL_VARBINARY, SQL_VARCHAR, STATEMENT, STATISTICS, STORAGE_ATTRIBUTES, STORAGE_MANAGER, STORE_IN_DHARMA, SUM, SYNONYM, SYSDATE, SYSTIME, SYSTIMESTAMP
T	TABLE, THEN, TIME, TIMESTAMP, TINYINT, TO, TRANSACTION, TRIGGER, TYPE
U	UID, UNION, UNIQUE, UPDATE, USER, USING
V	VALUES, VARBINARY, VARCHAR, VARYING, VIEW
W	WHEN, WHERE, WITH, WORK

Refused only as a new name: 134 words. Without double quotes these fail when a table or view is created, with `-20049 Keyword used for a name`. Queries can use them without quotes.

Starts with	Words
A	A, ABS, ACOS, ADD_MONTHS, AN, ARRAY, ASCII, ASIN, ATAN, ATAN2
B	BEGIN, BIND, BINDING

Starts with	Words
C	CEILING, CHARACTER_LENGTH, CHARTOROWID, CHAR_LENGTH, CHR, CLEANUP, COMPLEX, CONCAT, CONTINUE, CONVERT, COS, CURDATE, CURTIME, CVAR
D	DAYNAME, DAYOFMONTH, DAYOFWEEK, DAYOFYEAR, DB_NAME, DECLARATION, DECODE, DEGREES, DESCRIBE, DESCRIPTOR, DHTYPE, DIFFERENCE
E	EXEC, EXIT, EXP, EXPLICIT
F	FIELD, FLOOR, FOUND
G	GO, GOTO, GREATEST
H	HOUR
I	IFNULL, IMMEDIATE, INITCAP, INPUT, INSTR, INTERFACE
L	LAST_DAY, LCASE, LEAST, LENGTH, LIST, LOCATE, LOG10, LOWER, LPAD, LTRIM
M	MAIN, MINUTE, MOD, MONTH, MONTHNAME, MONTHS_BETWEEN
N	NEXT_DAY, NOW, NULLVALUE, NVL
O	OBJECT_ID, ODBCINFO, ODBC_CONVERT, OUTPUT
P	PI, POWER, PREFIX, PREPARE
Q	QUARTER
R	RADIANS, RAND, RECORD, REPEAT, REPLACE, ROWIDTOCHAR, RPAD, RTRIM
S	SEARCHED_CASE, SECOND, SECTION, SERVICE, SHORT, SIGN, SIMPLE_CASE, SIN, SOUNDEX, SQL, SQLERROR, SQLWARNING, SQRT, START, STOP, SUBSTR, SUBSTRING, SUFFIX, SUSER_NAME
T	TAN, TIMEOUT, TIMESTAMPADD, TIMESTAMPDIFF, TO_CHAR, TO_DATE, TO_NUMBER, TO_TIME, TO_TIMESTAMP, TPE, TRANSLATE
U	UCASE, UNSIGNED, UPPER, USER_ID, USER_NAME, UUID
V	VARIABLES, VERSION
W	WEEK, WHENEVER
Y	YEAR

Review and Edit the SQL That ASQL Creates

Expert mode adds two buttons to ASQL, **Show Script** and **Try Again**, so you can see the SQL that ASQL writes for a file and change it before it runs. It also shows the **Reserved Keywords** list and a **Generate Foreign Keys** box, and the tree starts at **Accessible Files**. An ordinary import doesn't need it. ASQL also turns on Show Script, Try Again and the keyword list by itself after an import fails.

To turn expert mode on, close ASQL and run these two commands in **Command Prompt** on the computer where you use ASQL:

```
set "ASQL_EXPERT_MODE=1"
"C:\Program Files (x86)\AppxODBCDriver\asql.exe"
```

When you are done, close ASQL and clear the setting:

```
set "ASQL_EXPERT_MODE="
```

ASQL only checks whether the variable exists, so even a value of 0 turns expert mode on. The setting belongs to that one Command Prompt window and the ASQL started from it. It doesn't affect the gateway, the APPX login manager or anyone else's ASQL.

What each button does with the script

Button	What it does
Import Table	Builds a new script from the APPX dictionary and runs it. Anything typed in the script pane is ignored.
Show Script	Builds a new script from the APPX dictionary and puts it in the pane, replacing whatever was there.
Try Again	Runs exactly what is in the pane, one statement at a time. Each statement ends with a semicolon. It stops at the first statement that fails and leaves only that statement in the pane.
Drop Table	Removes the file's SQL table, with its indexes and its public synonym. The APPX file and its records are not touched.

So a change in the pane only counts if **Try Again** is the next button you choose. Choosing **Show Script** or **Import Table** after an edit throws the edit away.

When a folder is selected instead of a file, the buttons read **Import Tables** and **Drop Tables** and act on every file in that folder.

Rename a column before it is created

Use this when an APPX field name can't be used as a SQL column name, or when you want a different name. The example assumes a field named NUMBER, which every edition refuses with `-20003 Syntax error`. See [SQL Name Rules and Reserved Words](#) for the rules and the list of refused words.

1. Select the file in the tree. If it already has a SQL table, choose **Drop Table** first.
2. Choose **Show Script**. The pane shows a `CREATE TABLE` statement, a `CREATE INDEX` or `CREATE UNIQUE INDEX` statement for each APPX key, and a `CREATE PUBLIC SYNONYM` statement.
3. Copy the whole script into Notepad and save it with your records.
4. Change the name everywhere it appears, including in the index statements. For example, change `number` to `number_txt`. Keep every column in the same place with the same type. The gateway matches columns to APPX fields by position, not by name, so a renamed column still reads the right field.

5. Choose **Try Again**. When every statement works, the status line reads **Ok** and the pane shows only the last statement, `CREATE PUBLIC SYNONYM ....`
6. Check the table with a query, for example in [WinSQL](#). Don't rely on the pane or the icon in the tree.

When a statement fails, the error appears on the status line at the bottom of the window.

Why a change can seem to disappear

- After a successful **Try Again**, choosing **Show Script** or **Import Table** shows the APPX names again, but the SQL table keeps your names. **Import Table** also stops with `-20041 Table/View/Synonym already exists` and puts a red exclamation mark on the file, because the table is already there.
- When **Try Again** fails, only the failed `CREATE TABLE` statement is left in the pane. Fixing the name there and choosing **Try Again** creates the table, but not its indexes or its public synonym. Queries still work, so this is easy to miss. To finish the job, choose **Drop Table** and start again at step 2.
- The icon next to a file shows the result of the last **Import Table** or **Drop Table** for that file: a check mark when it worked and a red exclamation mark when it failed. **Try Again** never changes the icon.
- A later **Import Table** for the same file, for example after the APPX file changes, builds the APPX names again. Keep the edited script so you can make the same change again.

Older instructions, including the FAQ answer [Is it possible to change the field names that are used by APPX/ODBC?](#), say to delete comment lines and the last semicolon before choosing **Try Again**. That isn't needed here: the script has no comment lines, and **Try Again** skips the empty end after the last semicolon.

Use a View for Friendly Names and Shorter Queries

A view is a saved query that programs can read like a table. It helps when the imported column names are awkward, when people need only a few of the columns, or when the same long query would otherwise be typed into every report. A view doesn't change the APPX file or the imported table, so it is also a safe way to rename columns without editing ASQL's script.

Create the view once, for example in WinSQL:

```
CREATE VIEW country_list (country_name, region_name, words_in_address, postcode_position)
AS SELECT name, region_label, no_words, post_code_loc FROM DMO_DMO_COUNTRY
```

Then read it like a table:

```
SELECT * FROM country_list ORDER BY country_name
SELECT country_name FROM country_list WHERE region_name = 'County' ORDER BY country_name
```

The second query returns France, Ireland and UK.

A few things to know:

- The `CREATE VIEW` statement has to fit within the statement limits in [Handle Wide Files and Long SQL Scripts](#). After that, every query against the view stays short.
- The view keeps the names of the table and columns it reads. If the table is dropped, the view stays, but queries on it fail with `-20005 Table/View/Synonym not found`. Once the table is imported again with the same column names, the view works again on its own. If a new import changes a column name, drop the view and create it again.
- The view's own column names follow the [SQL name rules](#), so a reserved word needs double quotes there too.
- To remove the view, run `DROP VIEW country_list`. The table and the APPX file are not affected.

Check Limits for Large Tables and Long SQL Statements

Script length, table width and an editor's text-entry limit are separate constraints. The supplied **Dharma 9.01 driver with the 5.5.2 gateway** reported these values through `SQLGetInfo`:

Property	Reported value
Driver version (<code>SQL_DRIVER_VER</code>)	09.01.0000
DBMS version reported by ODBC (<code>SQL_DBMS_VER</code>)	09.01.0000
Identifier quote character (<code>SQL_IDENTIFIER_QUOTE_CHAR</code>)	"
Column name length (characters) (<code>SQL_MAX_COLUMN_NAME_LEN</code>)	32
Columns in SELECT (<code>SQL_MAX_COLUMNS_IN_SELECT</code>)	0
Columns in one table (<code>SQL_MAX_COLUMNS_IN_TABLE</code>)	500
SQL statement length (characters) (<code>SQL_MAX_STATEMENT_LEN</code>)	35000
Identifier length (characters) (<code>SQL_MAX_IDENTIFIER_LEN</code>)	32
Table name length (characters) (<code>SQL_MAX_TABLE_NAME_LEN</code>)	32
Schema name length (characters) (<code>SQL_MAX_SCHEMA_NAME_LEN</code>)	32
Catalog name length (characters) (<code>SQL_MAX_CATALOG_NAME_LEN</code>)	0
Columns in GROUP BY (<code>SQL_MAX_COLUMNS_IN_GROUP_BY</code>)	0
Columns in one index (<code>SQL_MAX_COLUMNS_IN_INDEX</code>)	100
Columns in ORDER BY (<code>SQL_MAX_COLUMNS_IN_ORDER_BY</code>)	0
Index size (bytes) (<code>SQL_MAX_INDEX_SIZE</code>)	0
Row size (bytes) (<code>SQL_MAX_ROW_SIZE</code>)	0
Reported row size includes long values (<code>SQL_MAX_ROW_SIZE_INCLUDES_LONG</code>)	N
Tables in SELECT (<code>SQL_MAX_TABLES_IN_SELECT</code>)	250
Character literal length (characters) (<code>SQL_MAX_CHAR_LITERAL_LEN</code>)	2000

Property	Reported value
Binary literal length (hex characters) (<code>SQL_MAX_BINARY_LITERAL_LEN</code>)	2000
Active driver connections per environment (<code>SQL_MAX_DRIVER_CONNECTIONS</code>)	0
Active statements per connection (<code>SQL_MAX_CONCURRENT_ACTIVITIES</code>)	0

All 22 calls succeeded on a fresh connection to this edition. The DBMS version above identifies the ODBC SQL component; it does not replace the APPX engine version. A reported 0 does not establish an unlimited, usable capacity: no maximum may be specified, the limit may be unknown, or the property may not apply. These are reported values, not stress-tested maxima. See [Microsoft's SQLGetInfo definitions](#) for each property.

These limits differ from the updated driver used in the 5.5.3 and 5.5.4 editions. See [Improvements in Later Releases](#) for the tested later combinations and their larger reported table/name capacities. Keep them with this version and driver combination. A reported limit does not guarantee that every statement of that size will execute successfully.

The reviewed release-5.x ASQL source requests the generated script in 2,048-byte chunks and executes semicolon-separated statements individually. The gateway writes generated DDL to a script file and returns requested chunks. Source constants named `MAX_SCRIPT_LEN` are not, by themselves, proof of a whole-import limit. The packaged ASQL executable carries an older file-version label; verify the actual editor behavior rather than assuming every later source change is present in that executable.

A failure after one successful `CREATE` can leave a partial definition. Preserve the full SQL and error, check which tables, indexes and synonyms exist, then arrange a controlled retry. Do not repeatedly import a whole application folder to diagnose one wide file.

Observed long-statement errors

Fresh-connection, read-only catalog queries passed at **32,768 characters**, but failed at 33,000 through 35,000 characters with:

```
ERROR [00000] [Dharma][ODBC Dharma Driver 09.01.0000][Dharma/SDK] -20217
```

At 35,001 characters the driver returned:

```
ERROR [HY090] [Dharma][ODBC Dharma Driver 09.01.0000] -19 Invalid string or buffer length
```

The first diagnostic supplied no additional text. These probes establish observed failures, not an exact maximum for every generated `CREATE TABLE`. Keep statements comfortably below 32 KB and inspect the complete error if an import fails.

Shorten SQL without changing its meaning

First save the complete script. Remove unnecessary comments and indentation, and use shorter distinct column names with matching index references. Retain complete column declarations, commas, closing parentheses and `metadata_only` clauses. Execute complete table, index and synonym statements separately if the SQL tool requires that. Splitting the script does not make an oversized `CREATE TABLE` statement smaller.

Do not simply cut the SQL at a byte count, delete a middle column, or reorder columns. The gateway maps columns to APPX fields by position. Renaming preserves that position; removing a middle column shifts later mappings and can produce wrong values or conversion errors.

If a reduced import is necessary, retaining an unchanged **leading sequence of columns** and omitting only trailing declarations preserves the earlier positions in the reviewed source. This is an administrator workaround that must be tested against the specific file. Repair the final comma/parenthesis, omit indexes or constraints that reference removed columns, preserve the original script, and compare every retained field with native APPX records. This cookbook's COUNTRY tests do not establish that workaround for an arbitrary large file.

When the full table imports successfully but a client cannot handle its width, leave the base definition intact and select only the required columns, or **use a view**. For example:

```
SELECT name AS country_name, region_label, no_words, post_code_loc
FROM DMO_DMO_COUNTRY
ORDER BY name
```

This reduces the result width without disturbing APPX field mapping. A view cannot repair a base table that has not imported successfully.

CHAPTER 4

Read APPX Data with WinSQL

Connect, run a query and check the returned records.

Connect WinSQL and Read COUNTRY

Both statements were executed in a fresh WinSQL connection through the supplied 5.5.2 driver.

1. Open **WinSQL32**. The tested release is WinSQL 15.0.261.801, 32-bit.
2. Select **File** → **New Connection** and choose **AppxODBC**.
3. Enter the APPX user and password and leave **Remember Password** unchecked.
4. Choose **Generic ODBC** as the database plug-in and select **OK**.
5. Execute `SELECT COUNT(*) FROM DMO_DMO_COUNTRY`. The result is 7.
6. Replace that statement with `SELECT * FROM DMO_DMO_COUNTRY ORDER BY name` and execute it separately.

	name	region_label	city_sep_line	no_words	post_code_loc	added_by	date_added	changed_by	date_changed
1	Canada	Province	0	2	AFTER REGION	JPN	2001-03-01	SPF	2001-05-31
2	France	County	0	1	SEPARATE LINE		NULL	JPN	2001-03-01
3	Ireland	County	0	0		JPN	2001-03-07	JPN	2001-03-07
4	Netherlands		0	2	BEFORE CITY		NULL	JPN	2001-03-02
5	UK	County	1	2	SEPARATE LINE		NULL	LDN	2001-03-08
6	USA	State	0	1	AFTER REGION	SPF	2001-05-29	SPF	2001-05-29
7	United States	State	0	1	AFTER REGION		NULL	SPF	2001-05-29

Figure 8. WinSQL returns all seven COUNTRY records and nine fields through the supplied driver.

The result should contain the full country names and the [expected sample values](#). Run one statement at a time. Submitting both statements together produces a syntax error in this setup.

CHAPTER 5

Read APPX Data with Excel, Access and LibreOffice

Follow the tested procedure for each desktop program.

Choose 32-bit Excel and Access

Install **32-bit Microsoft 365 Apps**, including Excel and Access, using a suitable license. Close Office programs before changing an Office installation and remove an incompatible earlier Office installation through its supported uninstaller.

With Microsoft's [Office Deployment Tool](#), use this configuration file, saved as `office-32.xml` beside its `setup.exe`:

```
<Configuration>
  <Add OfficeClientEdition="32" Channel="Current">
    <Product ID="0365ProPlusRetail">
      <Language ID="en-us" />
    </Product>
  </Add>
  <Display Level="Full" AcceptEULA="FALSE" />
</Configuration>
```

Use the Product ID covered by the installed product license; `0365ProPlusRetail` is the Microsoft 365 Apps for enterprise product used here. From Command Prompt as administrator in that folder, run each command separately:

```
.\setup.exe /download .\office-32.xml
.\setup.exe /configure .\office-32.xml
```

Complete the displayed setup and licensing steps, sign in with a licensed account, and check **File → Account → About Excel** or **About Access** for 32-bit before connecting.

Load COUNTRY into Excel and Refresh It

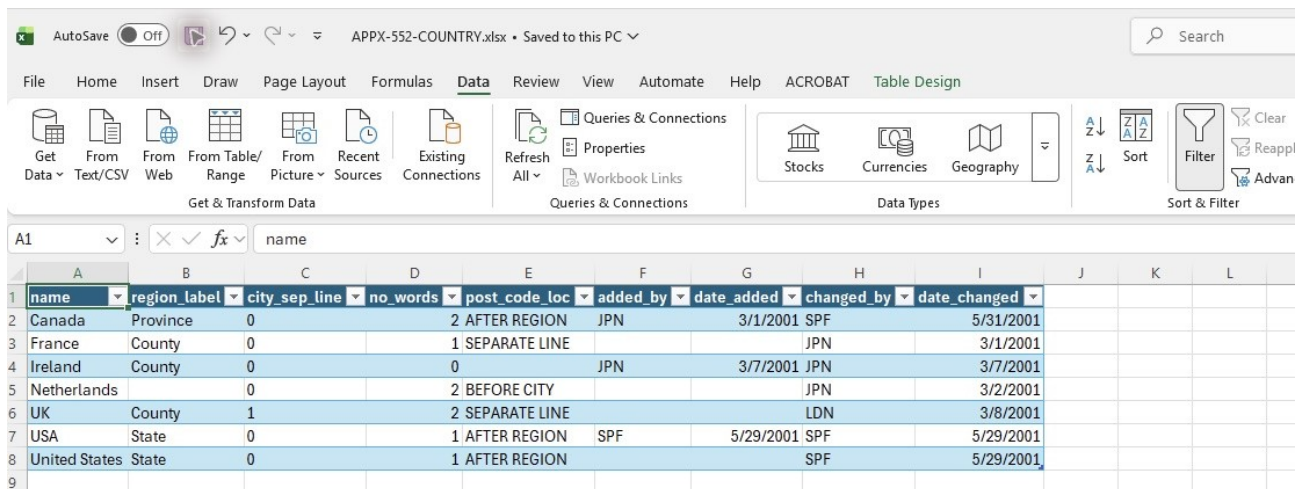
Use the **Legacy Microsoft Query** route in 32-bit Excel. If the command is hidden, open **File → Options → Data** and enable **From Microsoft Query (Legacy)** under **Show legacy data import wizards**.

1. Create a blank workbook and select **Data → Get Data → Legacy Wizards → From Microsoft Query (Legacy)**.
2. On **Databases**, select **AppxODBC**, keep **Use the Query Wizard to create/edit queries** checked, and select **OK**.

3. Enter the APPX user and password in the Dharma login dialog.
4. In the Query Wizard, expand `dmo_dmo_country` and add its nine columns to **Columns in your query**.
5. Select **Next**, keep the first sample unfiltered, and sort by `name`.
6. Select **Next**, keep **Return Data to Microsoft Excel** selected, and select **Finish**.
7. In **Import Data**, choose **Table**, choose cell `$A$1` on the existing worksheet, and select **OK**.
8. Wait until Excel returns to **Ready**. The worksheet contains **7 data rows and 9 columns**, plus the heading row.
9. Select **Table Design** → **Refresh**, wait for **Ready**, and check the same seven complete records.
10. Save the workbook locally. On reopening, refresh the external query and supply the APPX login when prompted.

Compare the saved worksheet with the native APPX sample after converting Excel date serials to dates. The workbook retains an external query that can refresh. Leave password saving disabled.

The fresh workbook was saved, reopened and refreshed through `AppxODBC`. All seven records and nine fields match the independently read native APPX sample, including dates and blank fields. No password is saved in the workbook connection.



	A	B	C	D	E	F	G	H	I	J	K	L
	name	region_label	city_sep_line	no_words	post_code_loc	added_by	date_added	changed_by	date_changed			
1	Canada	Province	0		2 AFTER REGION	JPN	3/1/2001	SPF	5/31/2001			
2	France	County	0		1 SEPARATE LINE			JPN	3/1/2001			
3	Ireland	County	0		0	JPN	3/7/2001	JPN	3/7/2001			
4	Netherlands		0		2 BEFORE CITY			JPN	3/2/2001			
5	UK	County	1		2 SEPARATE LINE			LDN	3/8/2001			
6	USA	State	0		1 AFTER REGION	SPF	5/29/2001	SPF	5/29/2001			
7	United States	State	0		1 AFTER REGION			SPF	5/29/2001			

Figure 9. The saved 5.5.2 workbook after a successful refresh.

A second login during the first import

The query wizard and worksheet use separate connections. A successful wizard login does not supply the password for the worksheet's connection. If another login appears when returning data or refreshing, confirm the intended `DSN` and enter the APPX password there too. Leave password saving disabled. Wait for all seven records before accepting the import. The saved workbook passed a fresh reopen, new password prompt and refresh.

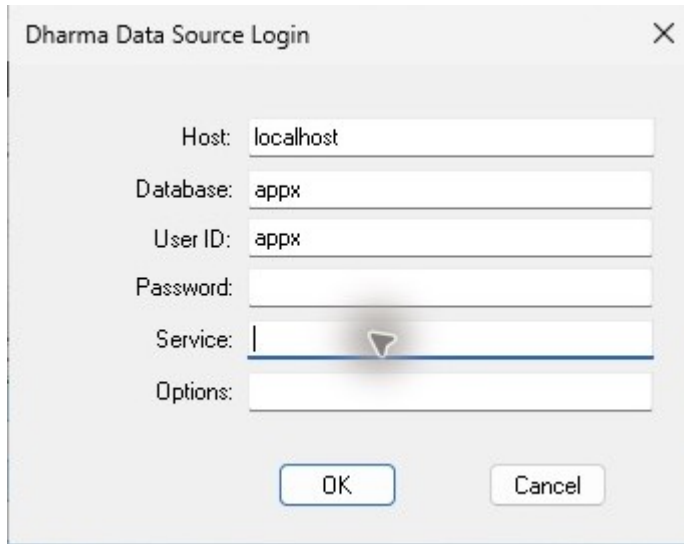


Figure 10. Excel asks for the APPX login when refreshing the saved workbook.

Leave **Service** empty, as pictured. This native refresh passed with the supplied 5.5.2 driver; all seven records and nine fields match the sample. Blank selects the default `sqlnw` lookup, reaching this gateway on TCP 1990. **Database appx** selects the SQL dictionary. A saved workbook can retain `Service=sqlnw` even after the DSN is cleared: clear that field in its login dialog, finish the refresh, and save the workbook. [Service and saved-connection overrides](#).

Create the connection with this edition's installed driver. A workbook copied from a newer-driver configuration can first report **Invalid connection string attribute**; create a fresh query through the wizard instead of assuming those saved attributes are compatible.

Access Import Fails with the Supplied Driver

Access import fails repeatedly with the supplied 5.5.2 driver. No working Access import procedure has been verified for this configuration. Nearest verified Access success: the [5.5.3 Access recipe](#) passes with the 5.5.3 engine, gateway and matching ASQL, the 32-bit Dharma 14.01 driver from the official APPX 6.2.1 package, and Microsoft 365 Access 16.0.20430.20182, 32-bit. Reproducing that result requires its complete documented configuration; a driver-only repair on the 5.5.2 gateway has not been verified.

The native test used **External Data → New Data Source → From Other Sources → ODBC Database → Import**, then **Machine Data Source → AppxODBC**. Access authenticated and listed COUNTRY, but selecting it returned **ODBC--connection to 'AppxODBC' failed**. A fresh Access process failed again after selecting the other listed COUNTRY entry. Neither attempt produced an accepted COUNTRY table. A further fresh-process check with the newly installed default DSN and Service left empty returned the same error; clearing Service did not fix Access. Repeating the import is not a verified workaround.



Figure 11. A fresh Access import fails after COUNTRY is selected.

A separate fresh PowerShell ODBC count and complete query immediately afterward returned **7 records and all 9 fields**, agreeing with the independently checked APPX sample. That successful query ran in PowerShell, not Access. It shows that the gateway answered a separate client; it does not accept Access or establish the cause of its failure. Do not assume that a different edition's IM006 or SQLSetConnectAttr error was observed here: these 5.5.2 attempts displayed the general connection error above.

For this edition, use the verified [Excel procedure](#), [Base direct SQL procedure](#), or [PowerShell query](#).

Choose 32-bit LibreOffice Calc and Base

Base direct SQL works with **32-bit LibreOffice 26.2.6.3** and the **supplied APPX 5.5.2 ODBC driver**. Its saved query passes after a fresh reopen/password prompt. Calc imports that saved direct SQL query, saves a refreshable workbook, and refreshes it after a fresh reopen/password prompt. All seven records and nine fields agree with the independent native APPX sample.

Use a 32-bit LibreOffice client with this 32-bit ODBC driver. A 64-bit LibreOffice process cannot load this driver or use its 32-bit [AppxODBC DSN](#). The bitness requirement is the reason for choosing the x86 package; the tests do not establish a requirement to use an older LibreOffice release.

The installation history and the tested version are different:

Item	What was used
Starting download	Official LibreOffice 25.8.7 Windows x86 MSI, obtained while that package was available from the vendor.
Version used for the successful tests	26.2.6.3, x86, after LibreOffice's own updater ran on first launch.
Existing installed copy	The separate 64-bit LibreOffice installation was kept. It was not used for these ODBC tests.

LibreOffice labels the **32-bit Windows architecture** deprecated. This is the vendor's support label for x86, not an APPX instruction to install an obsolete release. Check the [official LibreOffice download page](#) for an available 32-bit Windows build and its support status. If the current download selector no

longer offers x86, an older vendor package may be available from the [official archive](#). An archived package is an installation source; its presence does not establish current maintenance or APPX compatibility.

The 25.8.7 commands below reproduce the starting package used in this test. They are not a requirement to downgrade. Record the running version under **Help → About LibreOffice** after updates and verify that the executable is x86; the download filename alone does not identify the process that ultimately connects.

Keep an existing 64-bit installation while testing x86

For this test, the official MSI was administratively extracted into its own folder. This exposes the x86 program without replacing the installed 64-bit product or changing Windows file associations. With a locally downloaded MSI and an unused target folder, run this in Command Prompt as administrator:

```
msiexec.exe /a "C:\Downloads\LibreOffice_25.8.7_Win_x86.msi" TARGETDIR="C:\Tools\LibreOffice-x86" /qb
```

Substitute the actual download path. Wait for extraction to finish and check the resulting `program\soffice.exe`. Use that exact executable to start the 32-bit client. A separate profile also keeps its database registrations and settings distinct:

```
"C:\Tools\LibreOffice-x86\program\soffice.exe" -env:UserInstallation=file:///C:/Tools/LibreOffice-x86-profile --base
```

Close existing test instances before selecting another profile. If the updater restarts LibreOffice, check the running version and the [database registration](#) in the profile actually in use. The 5.5.1 test reproduced a missing registration after an updater/profile change; [its repair procedure](#) applies to the same LibreOffice runtime. Keep the saved 5.5.2 connection registered in the profile used here.

A normal 32-bit LibreOffice installation can also provide Calc and Base. Use its supported installer when replacing an existing installation; the administrative extraction is the coexistence method tested here. Finish the [driver and DSN checks](#) before either client recipe. No Java or embedded-database setup was used for the ODBC connection described below.

Connect LibreOffice Base and Save a COUNTRY Query

Use **direct SQL mode for this edition**. The saved direct SQL query passed with the matching 5.5.2 engine/gateway/ASQL, supplied Dharma 9.01 x86 driver and 32-bit LibreOffice 26.2.6.3. Ordinary table browsing and Base's parsed-query route returned **-20217 Unable to fetch data**; the working alternative is the direct SQL procedure immediately below. A visible table name or successful login alone does not accept the browse route.

The Base document stores a connection and query definition. It does **not** copy APPX records into an embedded Base database. First create the shared COUNTRY table definition with [ASQL](#).

1. Start **32-bit LibreOffice Base**. In Database Wizard, select **Connect to an existing database** → **ODBC**, then **Next**.
2. Select **Browse**, choose the existing **AppxODBC** data source, and confirm. The [DSN](#) already supplies Host, Database and optional Service. Keep Service blank in that saved DSN for this recipe.

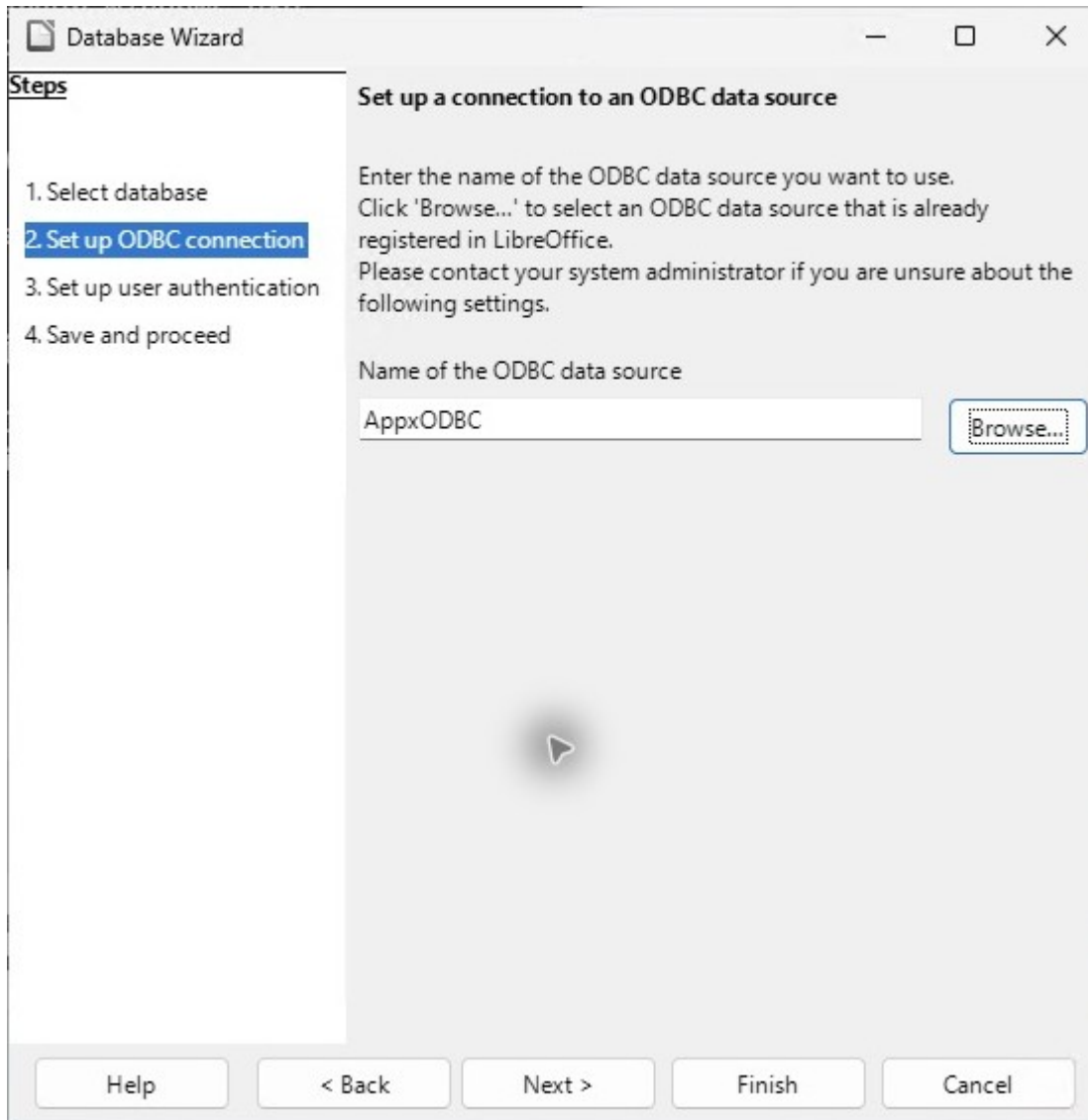


Figure 12. Base selects the existing AppxODBC connection.

3. Enter the authorized APPX user, keep **Password required** selected, and continue. Supply the password only in the masked authentication prompt.
4. Keep **Yes, register the database for me** and **Open the database for editing** selected. Leave the table wizard unchecked. Select **Finish** and save **APPX-552-COUNTRY.odt** locally.

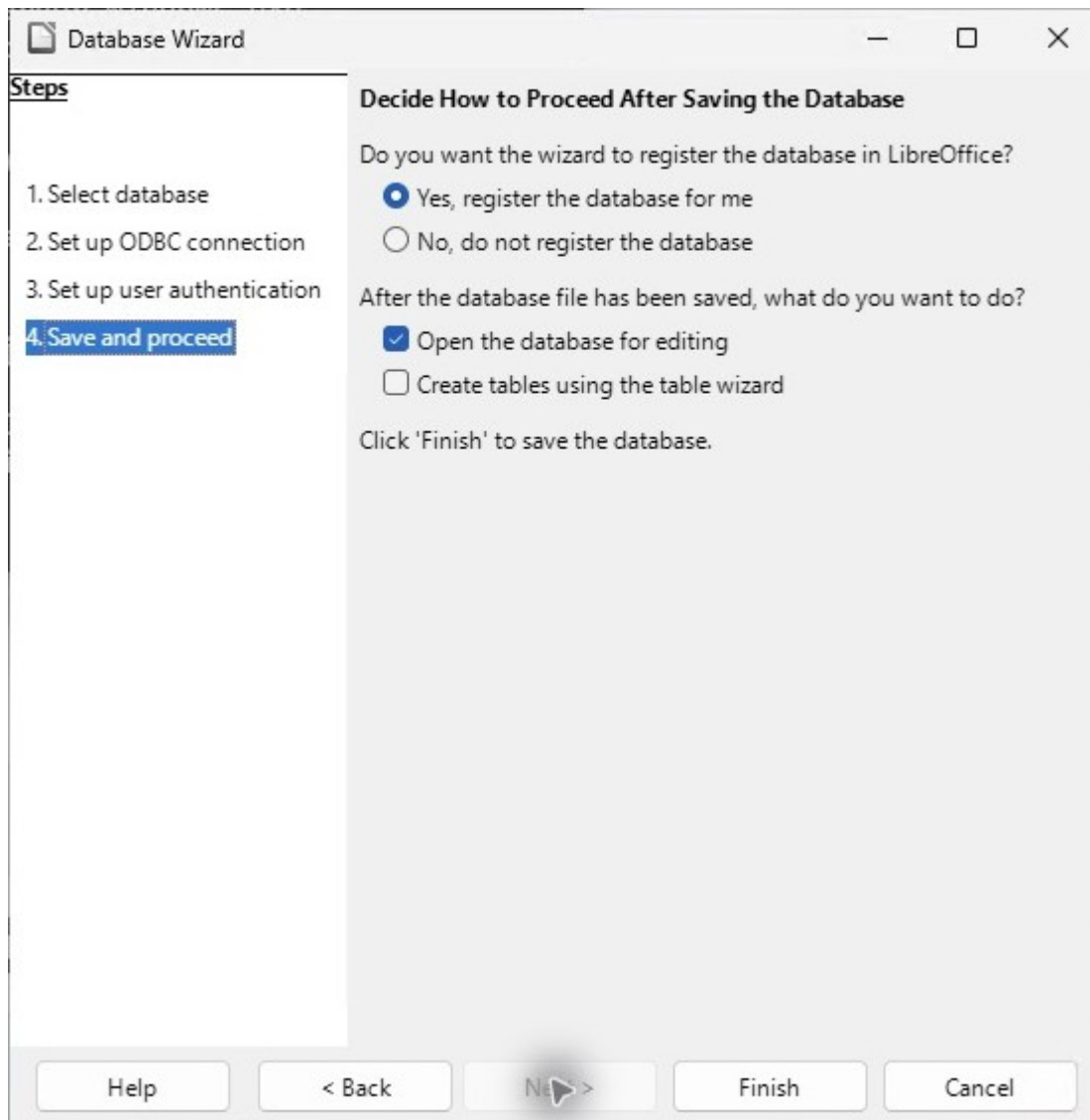


Figure 13. Register the saved Base connection so Calc can locate it.

5. Select **Queries → Create Query in SQL View**. Enter the statement below without a trailing semicolon:

```
SELECT * FROM "appx"."dmo_dmo_country" ORDER BY "name"
```

6. Select **Edit → Run SQL command directly**. Confirm the SQL command toolbar button is selected. This bypasses Base's SQL parser for the query; it does not change the driver or APPX server.
7. Select **Edit → Run Query** or press **F5**. Check all seven records and nine fields against the **complete expected result**.

APPX-552-COUNTRY.odb : Query1 — LibreOffice Base: Query Design

File Edit View Insert Tools Window Help

SQL

	name	region_label	city_sep_line	no_words	post_code_loc	added_by	date_added	changed_by	date_changed
▶	Canada	Province	0	2	AFTER REGION	JPN	03/01/01	SPF	05/31/01
	France	County	0	1	SEPARATE LINE			JPN	03/01/01
	Ireland	County	0	0		JPN	03/07/01	JPN	03/07/01
	Netherlands		0	2	BEFORE CITY			JPN	03/02/01
	UK	County	1	2	SEPARATE LINE			LDN	03/08/01
	USA	State	0	1	AFTER REGION	SPF	05/29/01	SPF	05/29/01
	United States	State	0	1	AFTER REGION			SPF	05/29/01

Record 1 of 7

SELECT * FROM "appx"."dmo_dmo_country" ORDER BY "name"

Figure 14. The direct SQL query returns all seven COUNTRY records and nine fields.

8. Select the query's **Save** button and name it **COUNTRY**. Close the query editor, then save the Base document itself. Direct SQL mode must remain stored with the query.
9. Close Base completely and reopen **APPX-552-COUNTRY.odb** in the same 32-bit LibreOffice profile. Select **Queries**, supply the APPX password when prompted, and double-click **COUNTRY**.

COUNTRY - APPX-552-COUNTRY — LibreOffice Base: Table Data View

File Edit View Insert Data Tools Window Help

	name	region_label	city_sep_line	no_words	post_code_loc	added_by	date_added	changed_by	date_changed
▶	Canada	Province	0	2	AFTER REGION	JPN	03/01/01	SPF	05/31/01
	France	County	0	1	SEPARATE LINE			JPN	03/01/01
	Ireland	County	0	0		JPN	03/07/01	JPN	03/07/01
	Netherlands		0	2	BEFORE CITY			JPN	03/02/01
	UK	County	1	2	SEPARATE LINE			LDN	03/08/01
	USA	State	0	1	AFTER REGION	SPF	05/29/01	SPF	05/29/01
	United States	State	0	1	AFTER REGION			SPF	05/29/01

Figure 15. The saved COUNTRY query returns its records after a fresh Base reopen.

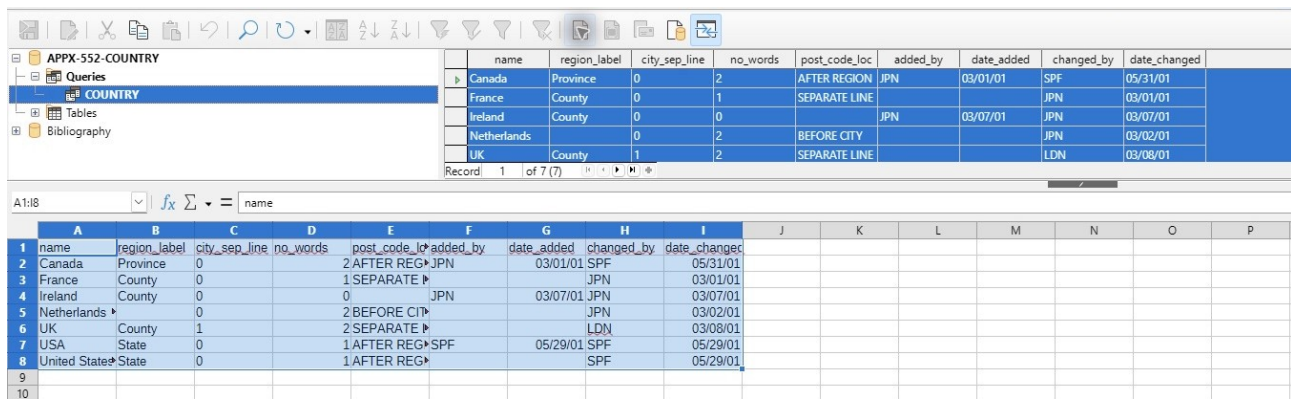
The fresh reopen passed with a new password prompt. The saved ODB identifies query **COUNTRY**, stores the exact qualified SQL above, and has SQL escape processing disabled. It requires a password without storing one. A saved query needs a live gateway and APPX server whenever it runs.

Do not switch back to design/parsed SQL mode and assume the same result. For the failed table-browse or parsed-query route, use this saved direct SQL query instead. The [LibreOffice ODBC reference](#) describes the connection wizard.

Load COUNTRY into LibreOffice Calc and Refresh It

First complete the [registered Base connection and COUNTRY direct SQL query](#). Use Calc from the same 32-bit LibreOffice installation and profile. The AppxODBC Windows DSN and the registered Base connection have different roles: the DSN selects the gateway; the registered name locates the .odb file and its saved query.

1. From Base, select **File** → **New** → **Spreadsheet**, or start the matching 32-bit Calc executable. Create a blank spreadsheet and select cell A1.
2. Select **View** → **Data Sources**. Expand **APPX-552-COUNTRY** → **Queries** and select **COUNTRY**. Supply the APPX login if requested. The data browser displays the query's rows above the worksheet.



	name	region_label	city_sep_line	no_words	post_code_loc	added_by	date_added	changed_by	date_changed
1	Canada	Province	0	2	AFTER REGION	JPN	03/01/01	SPF	05/31/01
2	France	County	0	1	SEPARATE LINE	JPN	03/01/01	JPN	03/01/01
3	Ireland	County	0	0		JPN	03/07/01	JPN	03/07/01
4	Netherlands	County	0	2	BEFORE CITY	JPN	03/02/01	JPN	03/02/01
5	UK	County	1	2	SEPARATE LINE	LDN	03/08/01	LDN	03/08/01
6	USA	State	0	1	AFTER REGION	SPF	05/29/01	SPF	05/29/01
7	United States	State	0	1	AFTER REGION	SPF	05/29/01	SPF	05/29/01

Figure 16. Calc can use the query registered by Base.

3. Click the data browser's upper-left corner to select all seven records, then select **Data to Text** on its toolbar. The records appear at the selected worksheet cell A1, with field headings and a refreshable database range. Ordinary Copy/Paste produces a static copy and does not establish this connection.
4. Check **A1:I8**: one heading row, 7 data rows, and 9 columns. Select columns A through I and use **Format** → **Columns** → **Optimal Width** to fit their contents if necessary. Use **View** → **Data Sources** again to hide the data browser and see the worksheet.
5. Select a cell within the imported range, then select **Data** → **Refresh Range**. Wait for completion and check the same values. This re-runs the source query; editing the worksheet is not a supported way to update APPX.

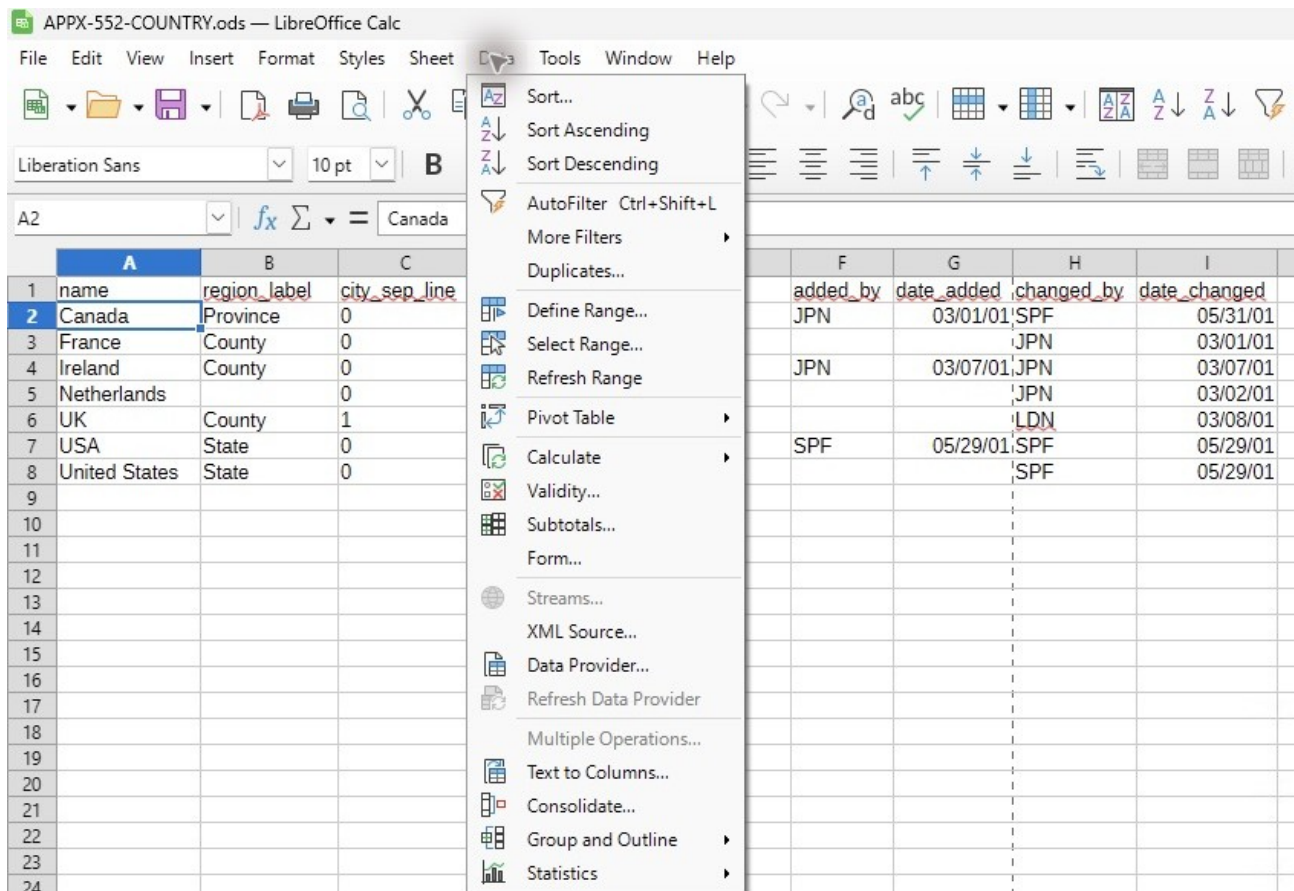
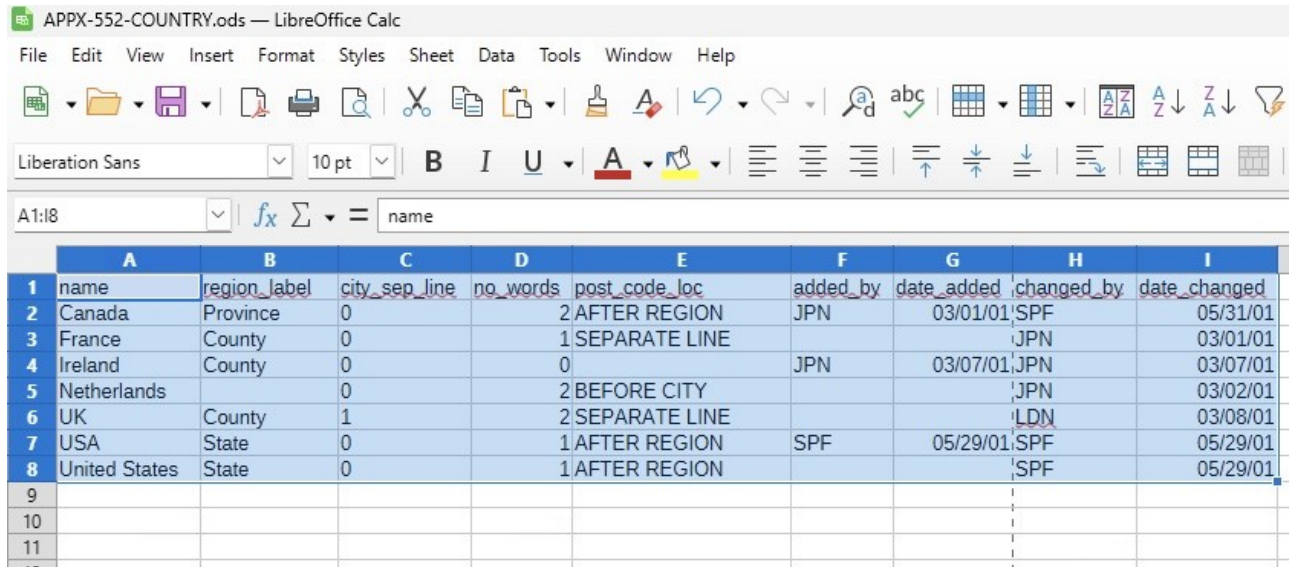


Figure 17. Refresh the imported database range from its saved query.

6. Save as **APPX-552-COUNTRY.ods**, retaining the ODF Spreadsheet format. Keep **APPX-552-COUNTRY.odt** at its registered location. The workbook's database range refers to the registered connection name and saved query.
7. Exit LibreOffice, reopen the ODS in the matching 32-bit installation and profile, select a cell inside the imported range, and use **Data → Refresh Range** again. Supply the APPX password when prompted, wait for completion, and save.



	A	B	C	D	E	F	G	H	I
1	name	region_label	city_sep_line	no words	post_code_loc	added_by	date_added	changed_by	date_changed
2	Canada	Province	0		2 AFTER REGION	JPN	03/01/01	SPF	05/31/01
3	France	County	0		1 SEPARATE LINE			JPN	03/01/01
4	Ireland	County	0			JPN	03/07/01	JPN	03/07/01
5	Netherlands		0		2 BEFORE CITY			JPN	03/02/01
6	UK	County	1		2 SEPARATE LINE			LDN	03/08/01
7	USA	State	0		1 AFTER REGION	SPF	05/29/01	SPF	05/29/01
8	United States	State	0		1 AFTER REGION			SPF	05/29/01
9									
10									
11									

Figure 18. The saved Calc workbook contains the complete COUNTRY sample.

The ODS was reopened in a fresh process with its existing profile registration, and refresh completed after the APPX password prompt. The refreshed, saved ODS values were compared with an independent native APPX export: all 7 records and all 9 fields agree, including empty audit dates. The saved file contains database range **Import1**, target **Sheet1.A1:I8**, and source query **COUNTRY**. Date cells remain date values even when the displayed date format differs from another client.

Keep the .odb connection, its LibreOffice registration, the Windows DSN and gateway access available for refresh. A workbook copied to another computer carries its saved worksheet data, but it does not install a driver, create a DSN or register its Base connection. Use the [registration repair](#) when the named source cannot be found. LibreOffice describes the [Refresh Range command](#) separately from refreshing a data-browser display.

Help LibreOffice Find a Saved Base Connection

If Calc reports **Error accessing data source 'APPX-552-COUNTRY'** with **com.sun.star.container.NoSuchElementException**, check the LibreOffice registration before changing the APPX gateway or password. A profile that does not contain the named registration cannot find that saved Base connection. The [5.5.1 registration example](#) shows the reproduced error with this LibreOffice runtime.

1. Confirm that **APPX-552-COUNTRY.odb** still exists and that the current LibreOffice process is 32-bit.
2. Open **Tools → Options → LibreOffice Base → Databases** in that same profile.
3. Select **New**. Set **Database file** to the existing .odb file, and **Registered name** to exactly **APPX-552-COUNTRY**, matching the workbook's saved source name.
4. Select **OK**, then **OK** in Options. If a registration already exists but its file moved, use **Edit** and correct its path instead of adding a conflicting second name.

5. Return to the workbook, select a cell in the database range, and use **Data → Refresh Range**. Supply the APPX login when requested.

A Windows [System DSN](#) is computer configuration; a LibreOffice registered database belongs to a LibreOffice user profile. Copying the ODS or `.odb` alone does not create that registration. Keep each connection layer's name and location in the deployment notes so a moved workbook can be repaired without rebuilding the shared APPX SQL definitions.

CHAPTER 6

Create Reports with Crystal Reports 2016

Choose the tested connection method, build a report and refresh its data.

Choose a Compatible Crystal Reports Version

Use SAP Crystal Reports 2016 SP9 Patch 21, 32-bit, with the supplied Dharma 9.01 x86 driver. The report client and driver must have the same architecture. These checks used the matching 5.5.2 engine, gateway and ASQL with AppxODBC, dictionary appx and gateway TCP 1990.

This edition passed a direct ODBC (RDO) SQL command, including report creation, save, PDF export and fresh-process reopen/password/refresh. Direct table selection was not accepted as part of this test. Follow the command procedure below rather than treating every provider route as interchangeable.

Create a COUNTRY Report through ODBC

1. Start 32-bit Crystal Reports 2016. Select **File** → **New** → **Standard Report** and begin on the wizard's **Data** page.
2. Expand **Create New Connection** → **ODBC (RDO)**. Select **AppxODBC**, enter the authorized APPX user and password, and finish the connection. The **DSN** selects the gateway; Database **appx** selects its SQL dictionary. Leave Service empty when that optional field is displayed, using the [default lookup described in the glossary](#).
3. Under the new ODBC connection, double-click **Add Command**. Enter the four-column **COUNTRY** report SQL, without a trailing semicolon:

```
SELECT name, region_label, no_words, post_code_loc
FROM DMO_DMO_COUNTRY
ORDER BY name
```

4. Select **OK**. Confirm **Command** is listed under **Selected Tables**, then select **Next**.

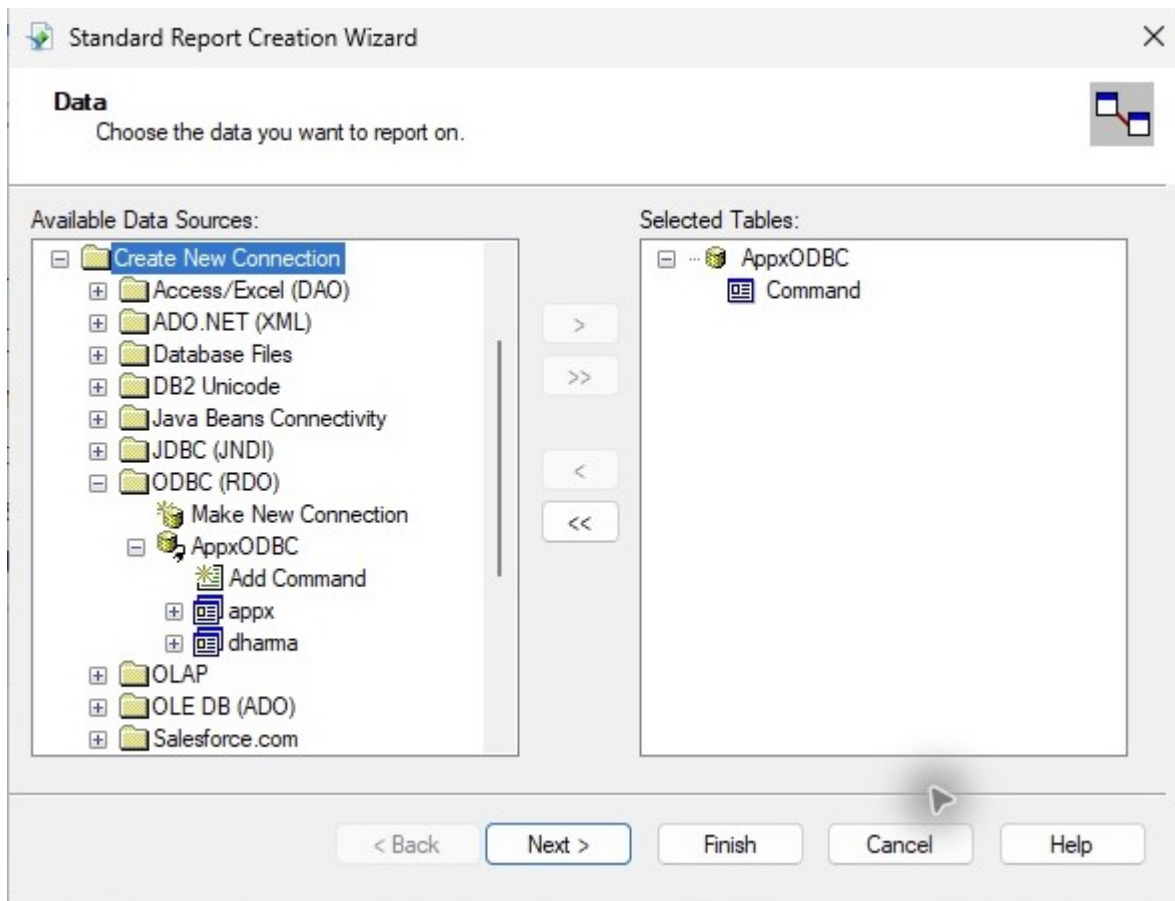


Figure 19. The new ODBC RDO connection selects a four-column SQL command.

5. Add the four fields to **Fields to Display** in order: **name**, **region_label**, **no_words**, **post_code_loc**. Select **Finish** without grouping or selection filters.
6. Check all seven records in Preview, including the blank region for Netherlands and blank post-code location for Ireland. The report must show **Records: 7** after the query completes.

10/15/2026

<u>name</u>	<u>region_label</u>	<u>no_words</u>	<u>post_code_loc</u>
Canada	Province	2.00	AFTER REGION
France	County	1.00	SEPARATE LINE
Ireland	County	0.00	
Netherlands		2.00	BEFORE CITY
UK	County	2.00	SEPARATE LINE
USA	State	1.00	AFTER REGION
United States	State	1.00	AFTER REGION

Figure 20. Crystal retrieves the seven COUNTRY records through its direct ODBC command.

7. Select **File** → **Save As** and save **APPX-552-COUNTRY-RDO.rpt** locally.

8. Select **File** → **Export** → **Export Report**, choose **PDF** and **Disk file**, and export **All** pages. Check the PDF against the [expected sample](#). All 28 report-field values agreed in this test.
9. Close Crystal completely, reopen the saved report and press **F5**. Choose **Refresh Report Data**, supply the APPX password in the masked Dharma login and wait for the new query. Confirm **Records: 7** and a current refresh timestamp, then save again. Initially displayed cached records do not prove a new query ran.

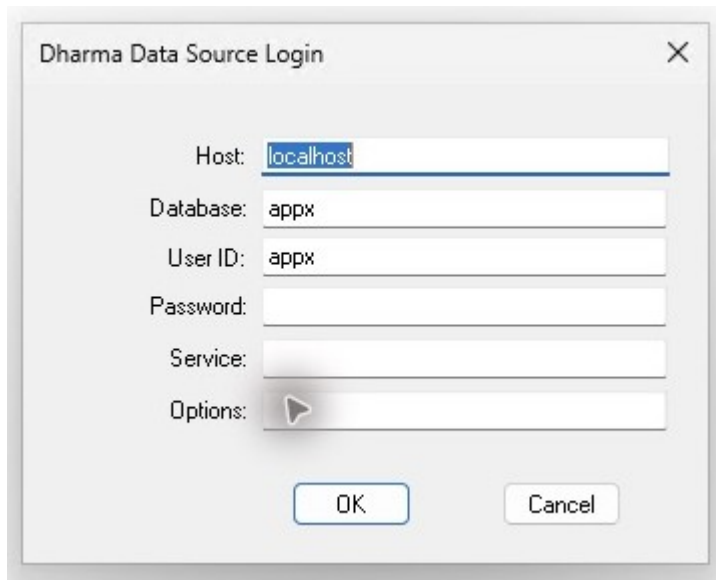


Figure 21. The reopened direct ODBC report requests the APPX password for refresh.

Creation, exported values and fresh-process refresh all passed with the configuration above. Report formatting is a separate task: widen or shorten the automatic headings before publishing a business report. No RPT or report PDF is included as a finished-layout example in the cookbook attachments.

How This Differs from the 5.5.1 Report Procedure

The 5.5.1 direct ODBC (RDO) command failed with its matching gateway and the same supplied Dharma 9.01 x86 driver; its [Microsoft OLE DB-to-ODBC procedure](#) passed. The independently tested 5.5.2 matching engine/gateway/ASQL plus supplied driver and Crystal Reports 2016 x86 passes the direct RDO command above. This is evidence for that complete later configuration, not proof of a driver-only repair for 5.5.1.

The 5.5.3 and 5.5.4 report recipes passed their separately documented configurations with the updated Dharma 14.01 x86 driver. Keep the gateway/driver/client combinations distinct when comparing results.

CHAPTER 7

Read APPX Data with 32-bit PowerShell

Query records, filter results, export files and inspect connection details.

Run a Query with a Secure Login Prompt

The public helper body passed in a fresh 32-bit Windows PowerShell process with the matching 5.5.2 gateway and supplied driver. COUNTRY count is 7, and the displayed sample matches the native APPX records.

Use **32-bit Windows PowerShell**, even on 64-bit Windows:

```
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
```

Read the complete example below and paste its entire & { ... } block into that 32-bit window. Press **Enter** on an empty line after the closing brace if PowerShell still shows its >> continuation prompt. This interactive form also works when the computer's execution policy disables .ps1 files.

For repeated use, download [query-appx-country-32.ps1](#) and save it in a working folder. If your site's policy permits the approved script, run:

```
.\query-appx-country-32.ps1 -Dsn AppxODBC
```

The script prompts for the APPX login, keeps the password out of its displayed results, and prints the count followed by the four-column sample table. It refuses a 64-bit PowerShell process and shows the path to the required 32-bit executable. Its expected output is:

```
COUNTRY count: 7

name          region_label  no_words  post_code_loc
----          -
Canada        Province     2 AFTER REGION
France        County       1 SEPARATE LINE
Ireland       County       0
Netherlands   County       2 BEFORE CITY
UK            County       2 SEPARATE LINE
USA           State        1 AFTER REGION
United States State        1 AFTER REGION
```

If a saved file produces **running scripts is disabled on this system**, use the reviewed interactive example, or have the administrator arrange the site's approved script-execution method. Keep the existing policy in place.

```
& {
param(
    [string]$Dsn = 'AppxODBC',
```

```

[System.Management.Automation.PSCredential]$Credential
)

$errorActionPreference = 'Stop'
if ([Environment]::Is64BitProcess) {
    throw 'Open 32-bit Windows PowerShell: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe'
}
if (-not $Credential) {
    $Credential = Get-Credential -UserName 'appx' -Message 'APPX login for the selected ODBC data source'
}
if (-not $Credential) { throw 'Login cancelled' }

$connectionString = New-Object System.Data.Odbc.OdbcConnectionStringBuilder
$connectionString['DSN'] = $Dsn
$connectionString['UID'] = $Credential.UserName
$connectionString['PWD'] = $Credential.GetNetworkCredential().Password
$connection = New-Object System.Data.Odbc.OdbcConnection($connectionString.ConnectionString)
$connectionString.Clear()

try {
    $connection.Open()
    $count = $connection.CreateCommand()
    $count.CommandText = 'SELECT COUNT(*) FROM DMO_DMO_COUNTRY'
    $count.CommandTimeout = 30
    Write-Host ('COUNTRY count: ' + $count.ExecuteScalar())
    $count.Dispose()

    $query = $connection.CreateCommand()
    $query.CommandText = 'SELECT * FROM DMO_DMO_COUNTRY ORDER BY name'
    $query.CommandTimeout = 30
    $reader = $query.ExecuteReader()
    $table = New-Object System.Data.DataTable
    $table.Load($reader)
    $reader.Dispose()
    $query.Dispose()
    $table | Format-Table name, region_label, no_words, post_code_loc -AutoSize
}
finally {
    $connection.Dispose()
    $Credential = $null
}
}

```

Check the Driver Architecture and Gateway Connection

These commands inspect the configuration without displaying a password. Run each command separately in Windows PowerShell:

```
Get-OdbcDriver -Name 'AppxODBC Driver' -Platform '32-bit' | Select-Object Name, Platform
Get-OdbcDsn -Name AppxODBC -DsnType System -Platform '32-bit' | Select-Object Name,
DriverName, Platform, DsnType
Test-NetConnection localhost -Port 1990 -InformationLevel Quiet
```

The first two commands should show the **32-bit** driver and **System DSN**. The last returns **True** when a TCP connection to the local gateway succeeds. On a separate desktop, replace `localhost` with the gateway hostname. This tests the TCP listener; the **credential-prompting SQL query** checks the remaining login and data path.

After changing an installed driver or **DSN**, close and restart clients, including PowerShell. An existing process can retain a previously loaded driver even though the DSN now points to a different file. The 5.5.2 query was verified from a fresh process.

Work with Query Results in PowerShell

Download `read-appx-odbc-32.ps1` into a working folder. This helper complements the **first connection example**: it returns one object per record, keeping the fields available for filtering, calculations and exports. Use it in **32-bit Windows PowerShell 5.1** with the site's approved script-execution method. Keep execution policy unchanged.

Open the **32-bit PowerShell executable**, change to the folder containing the helper, and run:

```
$credential = Get-Credential -UserName appx -Message 'APPX ODBC login'
$rows = @(.\read-appx-odbc-32.ps1 -Dsn AppxODBC -Credential $credential -Verbose)
$rows.Count
$rows | Format-Table name, region_label, no_words, post_code_loc -AutoSize
```

The default query reads COUNTRY ordered by name. Expect 7 objects. The `@(...)` wrapper keeps zero, one and many results as arrays. Each call opens a new connection, reads the results, and disposes the reader, command and connection, including on failure. The credential stays in this PowerShell session so later recipes can reuse it without another prompt. Do not put its password into SQL or a connection string typed at the console.

Use `Format-Table` only for display. Keep `$rows` as the original objects for the following recipes. The helper preserves returned numbers and dates, converts database NULL to PowerShell `$null`, and retains fixed-width text padding. It does not silently trim or change business values. Give duplicate or unnamed SQL result columns distinct **AS** aliases.

Read the built-in help with `Get-Help .\read-appx-odbc-32.ps1 -Full`. The helper accepts SELECT statements intended for reading; it does not parse SQL into a security policy or replace the account's database permissions.

Filter Records with SQL Parameters

Use a `WHERE` clause to reduce records read from APPX. Client-side `Where-Object` filters run only after the query has transferred the selected rows. Pass user-entered values through positional ODBC parameters instead of joining them into SQL text:

```
$country = 'Canada'
$selected = @(.\read-appx-odbc-32.ps1 -Credential $credential `
  -Sql 'SELECT name, region_label, no_words FROM DMO_DMO_COUNTRY WHERE name = ?' `
  -Parameter @(@{Type='VarChar'; Size=14; Value=$country}))
$selected | Format-Table -AutoSize
```

Expect one Canada record, **Province**, and **2**. Parameter types should match the APPX field exposed by the SQL definition. COUNTRY's name has a maximum of 14 characters. An ODBC `?` represents a value; it cannot stand for a table name, column name or `ORDER BY` direction. Keep identifiers in reviewed SQL.

For multiple parameters, add their specifications in exactly the same left-to-right order as the question marks. The names in a PowerShell hashtable are helper options, not named SQL parameters:

```
$selected = @(.\read-appx-odbc-32.ps1 -Credential $credential `
  -Sql 'SELECT name, no_words FROM DMO_DMO_COUNTRY WHERE no_words >= ? AND name <> ?
ORDER BY name' `
  -Parameter @(@{Type='Int'; Value=2}, @{Type='VarChar'; Size=14; Value='Canada'}))
$selected | Format-Table -AutoSize
```

The expected names are **Netherlands** and **UK**. The [Microsoft ODBC parameter reference](#) explains this positional behavior. Use `IS NULL` to find SQL NULLs; `= ?` with a null parameter is not the same predicate. Check empty strings, padding, case and dates against native APPX values before extending a filter to a production file.

Export COUNTRY to CSV and JSON

Keep the `unformatted query objects` in `$rows`. Choose a new local filename; `-NoClobber` prevents replacing an existing CSV:

```
$rows | Select-Object name, region_label, no_words, post_code_loc |
  Export-Csv -LiteralPath .\COUNTRY.csv -NoTypeInformation -Encoding UTF8 -NoClobber
$csv = @(Import-Csv -LiteralPath .\COUNTRY.csv)
$csv.Count
$csv | Format-Table -AutoSize
```

Expect 7 data rows. Windows PowerShell 5.1's UTF8 encoding writes a byte-order mark; CSV quotes and escaping are handled by `Export-Csv`. Its default separator is a comma. Use `-UseCulture` on both export and import when the receiving application's regional separator is required. CSV re-import produces strings, so cast numeric fields before calculations. CSV is a snapshot; opening it in Calc or Excel does not create an ODBC refresh connection. See [Microsoft's Export-Csv reference](#).

For JSON, select only the business properties needed by the recipient. Preserve an array even when a filter returns one record:

```
$jsonRows = @($rows | Select-Object name, region_label, no_words, post_code_loc)
$json = ConvertTo-Json -InputObject $jsonRows -Depth 4
if (Test-Path -LiteralPath .\COUNTRY.json) { throw 'Choose a new JSON filename' }
$json | Set-Content -LiteralPath .\COUNTRY.json -Encoding UTF8
$loaded = Get-Content -LiteralPath .\COUNTRY.json -Raw | ConvertFrom-Json
$loaded.Count
```

Expect 7 objects, with `no_words` still numeric. In Windows PowerShell 5.1, assign `ConvertFrom-Json` directly as shown; wrapping that command in `@(...)` can produce an outer array containing the result array and a misleading count of 1. This example deliberately selects four fields. If audit dates are included, choose an explicit interchange format: Windows PowerShell 5.1's default DateTime JSON representation is not an ISO 8601 string. Convert a non-null date using `$value.ToString('o')` when the recipient requires ISO 8601; do not invent a timezone for an APPX date that has none. Exported files contain data and must be stored with the same care as the source records.

Summarize and Check the Returned Data

Use PowerShell for a small analysis after obtaining `$rows`. This does not change the APPX file:

```
$rows | Measure-Object -Property no_words -Sum -Minimum -Maximum
$rows | Group-Object -Property { $_.region_label.TrimEnd() } |
    Sort-Object Name | Select-Object Name, Count
$rows | Where-Object { $_.no_words -ge 2 } |
    Select-Object name, no_words
```

For the unmodified sample, `no_words` sums to 9, with minimum 0 and maximum 2. There are 3 County, 1 Province, 2 State, and 1 blank region labels. The final filter returns Canada, Netherlands and UK. The grouping trims trailing padding only for the grouping key; the original `$rows` remain intact.

A second query can be compared with the earlier snapshot:

```
$again = @(.\read-appx-odbc-32.ps1 -Credential $credential)
Compare-Object -ReferenceObject $rows -DifferenceObject $again `
    -Property name, region_label, city_sep_line, no_words, post_code_loc, added_by,
    date_added, changed_by, date_changed
```

No output means the selected nine fields agree between these two ODBC reads. It does not independently prove the APPX-to-SQL mapping. The [native APPX comparison](#) remains the check for that. If APPX users change records between reads, differences can be expected.

List Tables and Columns and Identify the Driver

These calls use the same credential and return objects that can be inspected or exported:

```
.\read-appx-odbc-32.ps1 -Credential $credential -ConnectionInfo | Format-List
$tables = @(.\read-appx-odbc-32.ps1 -Credential $credential -Tables)
$tables | Where-Object { $_.TABLE_NAME -match 'COUNTRY' } |
    Select-Object TABLE_SCHEM, TABLE_NAME, TABLE_TYPE
$columns = @(.\read-appx-odbc-32.ps1 -Credential $credential -Describe `
    -Sql 'SELECT * FROM DMO_DMO_COUNTRY WHERE 1 = 0')
$columns | Format-Table ColumnName, Ordinal, DataType, Size, AllowDBNull -AutoSize
```

The connection information shows the loaded DLL path and file version, along with the server version and process bitness; compare it with the [tested version table](#). The metadata calls discover definitions already in the selected SQL dictionary. They do not import a new APPX file. A table and its synonym may both be listed; duplicate discovery entries do not mean duplicate APPX business records.

The table-discovery call retrieves unrestricted metadata before filtering in PowerShell. Column description uses the selected SQL result with `WHERE 1 = 0`, which describes the nine COUNTRY fields without returning business records. `-Describe` still executes the supplied SQL, so retain the false predicate and use a SELECT statement.

The optional `-Columns` switch requests unrestricted ODBC column metadata; it was not part of this edition's accepted checks. The focused `-Describe` example above passed and returned the nine COUNTRY columns. The [5.5.1 metadata checks](#) found inconsistent unrestricted discovery with this supplied driver generation, including `-20217 / SQLSTATE 00000`. Keep that separate result with its tested release and use focused discovery here. Some legacy driver combinations treat missing, null and empty restriction arguments differently. On a large dictionary, unrestricted column discovery can be expensive; use it for diagnosis and retain the result rather than running it for every business query. `-CommandTimeout` applies to SQL commands, not `GetSchema` or connection opening.

Set Query Limits and Read Error Details

The helper defaults to a 30-second SQL command timeout and at most 10,000 result rows. Select the needed columns and apply SQL filters before increasing either:

```
$rows = @(.\read-appx-odbc-32.ps1 -Credential $credential `
    -Sql 'SELECT * FROM DMO_DMO_COUNTRY ORDER BY name' -CommandTimeout 30 -MaxRows 100
-Verbose)
```

A query exceeding `MaxRows` fails without returning a partial result. It is a limit on collected rows, not a server-side pagination clause. `-Verbose` reports elapsed time and returned row count without adding status text to the success pipeline. Driver enforcement of a command timeout varies; it does not guarantee that login, metadata discovery, or every native-driver operation will stop within that period. Cancel a stalled client and investigate the [driver, ports and login path](#) before retrying.

ODBC failures include each returned **SQLSTATE**, **native error number**, and message. Retain these details and the failing query shape when requesting help. Keep passwords and private field values out of shared diagnostics. A missing [DSN](#) usually needs the [bitness and DSN check](#); a reachable TCP socket alone does not establish a successful APPX login or query.

When finished, clear the working variables or close the PowerShell window:

```
Remove-Variable credential, rows, again, selected, tables, columns -ErrorAction  
SilentlyContinue
```

Closing the connection does not refresh a previously exported CSV, JSON document, or local Access copy. Run another query to create a new snapshot.

CHAPTER 8

Solve Connection and Import Problems

Check errors and settings, record diagnostic logs and turn logging off afterward.

Set Connection and Diagnostic Options

Set a variable for the **process that reads it**. An environment setting in an Excel user's command window does not change an already-running gateway or an APPX engine started by the login manager. These scopes follow the reviewed 5.5.2 source; other releases should be checked separately.

Variable	Purpose	Where to set it for this recipe
<code>ASQL_EXPERT_MODE=1</code>	Enables ASQL's script-editing controls. Presence enables it, even if the value is <code>0</code> .	In the command window used to start ASQL on the administration computer. Clear it with <code>set "ASQL_EXPERT_MODE="</code> .
<code>ASQL_LOG=C:\ODBC-Logs\asql.log</code>	Records ASQL's generated SQL and ODBC operations. Opening ASQL overwrites this file.	In the same command window, before starting ASQL.
<code>APPX_DATA_SERVER=localhost:31053</code>	Selects the APPX backend host and login-manager port.	On the gateway , in <code>C:\Program Files (x86)\AppxODBCserver\appx.dbs\appx.env</code> . Use the actual APPX hostname for a remote server.
<code>APPX_ODBC_LOG=C:\ODBC-Logs\gateway.log</code>	Traces the APPX storage-manager part of the gateway, including file mapping and reads. Its default fallback is <code>C:\tmp\appx-odbc.log</code> if that path is writable.	In the gateway dictionary's <code>appx.env</code> , before opening a fresh connection. A new gateway task can overwrite the file; reproduce with one diagnostic connection.
<code>APPX_ODBC_CODE=-1</code>	Selects gateway trace categories; this is the source's default of all categories. <code>0</code> suppresses messages.	Alongside <code>APPX_ODBC_LOG</code> in the gateway dictionary's <code>appx.env</code> .
<code>APPX_NETIO_LOG=C:\ODBC-Logs\gateway-netio.log</code>	Traces the APPX/Net file-manager client used by the gateway to reach APPX.	In the gateway dictionary's <code>appx.env</code> for the gateway side of the backend connection.
<code>APPX_NETIO_CODE=-1</code>	Selects that file-manager client's logging categories.	Alongside <code>APPX_NETIO_LOG</code> in the gateway dictionary's <code>appx.env</code> .
<code>APPX_NET_LOG=C:\ODBC-Logs\network</code>	Traces APPX communication. The source adds a PID and role suffix to the filename.	In the gateway dictionary's <code>appx.env</code> for gateway traffic; in the login manager's engine Environment subkey for the APPX server side. Each side writes locally.
<code>APPX_NET_LOG_MASK=-1</code>	Selects communication trace categories.	In the same process environment as <code>APPX_NET_LOG</code> .

Variable	Purpose	Where to set it for this recipe
APPX_NETIO_SERVER=C:\ODBC-Logs\server-netio.log	Traces the APPX/Net file-manager server serving file requests.	In the APPX login manager's engine Environment subkey, for the APPX engine handling the ODBC connection.
APPX_IO_LOG=C:\ODBC-Logs\io_\$(pid).log	Logs native APPX file I/O; the PID substitution separates engine log files.	In the APPX login manager's engine Environment subkey.
APPX_IO_BITS=0xFFFFFFFF	Selects native APPX file-I/O logging categories.	Alongside APPX_IO_LOG for the APPX engine.
APPXPATH	Identifies an APPX installation's application/data root. ASQL also uses it when searching for local System Administration structures.	Retain the APPX engine's existing configuration. Set it in an ASQL command window only when deliberately using local structures from that installation; it is not the gateway Host setting.
APPX_UID	Carries an APPX login internally when connecting to the backend.	The gateway builds it from the supplied login. Use the client login prompt; do not put a password in a public script or shared appx.env.

ASQL: one command window

Create a local log directory on the ASQL computer. Run these commands individually in **Command Prompt**:

```
mkdir C:\ODBC-Logs
set "ASQL_EXPERT_MODE=1"
set "ASQL_LOG=C:\ODBC-Logs\asql.log"
"C:\Program Files (x86)\AppxODBCDriver\asql.exe"
```

After the reproduction, close ASQL and clear both overrides:

```
set "ASQL_EXPERT_MODE="
set "ASQL_LOG="
```

Keep ASQL logs private. The reviewed ASQL logging code can record the connection string, including a password. Do not attach or publish the raw log; remove credential-bearing lines from a separate copy before sharing it. Turn logging off after the focused reproduction.

This does not enable logging in Excel, WinSQL, the [gateway service](#) or the APPX server. For desktop driver-manager tracing, use the **Tracing** tab in the appropriate Windows ODBC Administrator; it is a separate diagnostic from ASQL_LOG and the gateway logs.

Gateway: the selected dictionary's appx.env

On the gateway computer, preserve `appx.dbs\appx.env` before editing it. Keep its `APPX_DATA_SERVER` line, add only the needed diagnostic assignments, and ensure the `gateway service` account can write the chosen local log directory. Start a fresh ODBC connection after the change; an existing initialized gateway task can retain its earlier settings. Arrange a gateway-service restart only if needed during a maintenance interval, because it affects other connected users.

For a first file-mapping diagnostic, add:

```
APPX_ODBC_LOG=C:\ODBC-Logs\gateway.log
APPX_ODBC_CODE=-1
```

These are text-file assignments, not commands to run at a desktop prompt. Remove the added diagnostic lines after the test, reconnect, and retain the original `endpoint` configuration.

APPX engine: the login manager's Environment subkey

On the APPX server, the Windows login manager reads **named string values** from this registry **subkey** and supplies them to the engines it starts:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Appx-552-31053-0SAuth\Environment
```

Substitute the actual **APPX login-manager Windows service name**, not the ODBC Service alias `sqlnw`. Back up that subkey, then add individual `REG_SZ` values such as `APPX_IO_LOG` and `APPX_IO_BITS`, preserving its existing values. This APPX subkey is different from a generic Windows service's `Environment` multi-string value. Do not replace the entire service environment with a single logging assignment.

Make the log directory writable by the engine's account. Restart the corresponding login manager at an agreed interval, then create a new client connection so a new engine inherits the settings. In the reviewed source, machine-level environment values can override the same names in this subkey when `LoadSystemEnvironment` is enabled; check for duplicate definitions if a setting appears ineffective.

`CreateLogFile` and `LogAllEvents` are **login-manager configuration properties**, not environment variables. They belong under the **APPX login-manager Windows service's Config** subkey. Gateway `sql_server.log` is also a separate SQL-server log. Neither is enabled by setting `ASQL_LOG` on a desktop.

Log files can contain SQL, field values, paths and connection details. Collect one focused reproduction, keep the original evidence private, redact shared copies, then remove the temporary settings. Never publish a credential-bearing environment dump.

Record a Diagnostic Log, Then Turn Logging Off

Use [Environment Variables and Where to Set Them](#) to select the process and configuration location. The defaults below were checked in this release's source. A configured value takes precedence over its source default; record the installation's actual settings before changing them.

Diagnostic	Default and file behavior	Activation and category selection
ASQL expert mode	Disabled when <code>ASQL_EXPERT_MODE</code> is absent. Any present value enables the override.	Set it in ASQL's command window before starting ASQL. Close ASQL and clear the variable to remove the override.
ASQL log	Disabled when <code>ASQL_LOG</code> is absent. A configured file is opened for overwrite when ASQL starts.	Set a writable local path in ASQL's command window. Keep raw logs private because the reviewed code can log a password-bearing connection string.
Gateway storage trace	With <code>APPX_ODBC_LOG</code> absent, the gateway attempts <code>C:\tmp\appx-odbc.log</code> . Opening a new gateway task can overwrite that file.	Set a path in the gateway dictionary's <code>appx.env</code> ; open a fresh connection. <code>APPX_ODBC_CODE</code> defaults to all bits, hexadecimal <code>FFFFFFFF</code> ; use <code>-1</code> to request that complete mask or <code>0</code> to suppress trace messages.
APPX/Net file-manager trace	<code>APPX_NETIO_LOG</code> and <code>APPX_NETIO_SERVER</code> have no default file. Their configured files are opened for overwrite.	Put the client path/mask on the gateway and the server path in the APPX login manager's engine environment. <code>APPX_NETIO_CODE</code> defaults to all bits; use <code>-1</code> for the complete mask. The client transmits its categories to the file-manager server.
APPX network trace	No file unless <code>APPX_NET_LOG</code> is set. The logger appends a PID/role suffix and opens the resulting file for append.	<code>APPX_NET_LOG_MASK</code> defaults to all categories except timer and extra process-ID output. Set <code>-1</code> for all bits or <code>0</code> to suppress categories. Use a fresh process/connection.
APPX native file-I/O trace	No log unless <code>APPX_IO_LOG</code> is set. This logger expands variables in its path, including the <code>\$(pid)</code> example in the environment recipe.	<code>APPX_IO_BITS</code> defaults to all bits and uses unsigned mask parsing; <code>0xFFFFFFFF</code> requests all categories. Set it in the engine environment and start a new engine.
Pipe communication helper	No log unless <code>APPX_COMM_LOG</code> is set. The helper first opens the configured file for overwrite, then appends subsequent messages.	This setting belongs to the Pipe helper selected by <code>AppxPipeExe</code> . Set it in the login manager's engine Environment subkey so the spawned helper inherits it. Its dedicated logger has no category-mask parameter.
Login manager	<code>CreateLogFile=false</code> and <code>LogAllEvents=false</code> in the source defaults. If enabled without a custom log name, the source uses the APPX login-manager Windows service name plus <code>.log</code> under <code>LogDirectory</code> , whose default is <code>C:\</code> .	These are named values under the APPX login-manager Windows service's Config subkey. <code>LoadSystemEnvironment</code> defaults to <code>true</code> , allowing machine environment definitions to override duplicate engine-environment names. Restart the affected login manager to apply changes.

This edition tests a **Pipe** backend. `APPX_COMM_LOG` can help investigate that helper path; it does not replace the engine/network logs.

Use the complete Windows trace mask

The gateway, APPX/Net client and APPX network source parse their category masks with signed `strtol`. A Windows C-runtime check returned `7FFFFFFF` for the input `0xFFFFFFFF`, but `FFFFFFFF` for `-1`. Use these assignments for a focused all-category capture, in the process scopes described above:

```
APPX_ODBC_CODE=-1
APPX_NETIO_CODE=-1
APPX_NET_LOG_MASK=-1
```

`APPX_IO_BITS` uses unsigned parsing and can retain `0xFFFFFFFF`. A mask selects categories; the corresponding log-path setting and a writable directory still matter. Use one diagnostic connection when a logger has a fixed filename that can be overwritten by another connection.

Capture one failure and restore the settings

1. Save the original dictionary `appx.env` and the relevant **APPX login-manager Windows service Environment and Config** subkeys. Record the client/driver versions, failing action, `SQLSTATE` and native error. Keep that connection's timestamp so the client, gateway and engine records can be matched.
2. Choose the layer implicated by the error. Configure its local log directory and only the settings needed for one reproduction. Give the gateway and engine accounts write access to their respective directories. An unwritable configured APPX network log can terminate the affected process in the reviewed source.
3. Close ASQL before changing its command-window settings. Open a fresh ODBC connection after a gateway-environment change. For engine/helper or login-manager changes, restart only the corresponding login manager at an agreed interval, then make a new connection; an existing engine keeps its inherited environment.
4. Reproduce once. Preserve the original logs privately, then redact passwords, connection strings and private field values from a separate copy before sharing it. Windows ODBC Administrator tracing is a separate desktop diagnostic; stop it in its **Tracing** tab after the same reproduction.
5. Close the diagnostic client. Restore the saved values and remove only the overrides added for this capture. For ASQL, close the program and run `set "ASQL_LOG="` and `set "ASQL_EXPERT_MODE="` in its command window. In a registry Environment subkey, remove an added variable rather than leaving an empty value that can still be present to the process.
6. Apply the restoration using the same fresh-connection/restart rule, and verify the **small COUNTRY query**. Check that temporary desktop tracing has stopped and that only the intended restored logging remains.

Removing `APPX_ODBC_LOG` restores the fallback path; it does not turn gateway tracing off. To suppress its messages deliberately, use `APPX_ODBC_CODE=0` and a fresh connection. The file can still be opened or truncated. Restore the previous setting when the diagnostic interval is complete; deleting the mask restores the source's all-category default.

Check the Driver, Ports and Login Before Retrying

Symptom	Action
The DSN is missing	Match the client and driver bitness; use <code>SysWOW64\odbcad32.exe</code> for these 32-bit clients.
A connection waits or fails	Check the gateway's actual listener and the client's <code>sqlnw</code> mapping.
Login returns -1016 Can't Connect to the Appx Data Server	Check credentials, the APPX login-manager service and <code>APPX_DATA_SERVER</code> in <code>appx.env</code> .
Windows cannot start ASQL or the gateway	Check the full loader error and the matching Microsoft runtime family.
A table is not found	Check the selected dictionary and import the individual COUNTRY definition.
Access import fails	Use the verified 5.5.3 Access configuration or this edition's Excel, Base direct SQL or PowerShell procedure. Repeated Access imports did not fix this edition.
An import stops partway through	Save the full SQL and error and review the wide-file guidance.

Check the dhsodbc.ini and appx.env Settings Files

Check these files on the gateway computer, substituting its actual installation folder:

```
type "C:\Windows\dhsodbc.ini"
type "C:\Program Files (x86)\AppxODBCserver\appx.dbs\appx.env"
```

The relevant INI entries are:

```
[Environment]
TPEROOT="C:\Program Files (x86)\AppxODBCserver"
DB_NAME=appx
TPESQLDBG=N
```

`TPEROOT` identifies the gateway root, not its `bin` folder. The dictionary's file is `appx.dbs\appx.env`, containing:

```
APPX_DATA_SERVER=localhost:31053
```

This is the APPX backend [endpoint](#), separate from the gateway's measured TCP 1990. On a separate gateway computer, replace `localhost` with the APPX server's hostname. Keep dictionary `appx` separate from APPX Database ID `DMO`.

Back up both files before editing. Keep their `.ini` and `.env` extensions and preserve other existing settings. Open a fresh client connection after a correction. Connection-setting changes do not themselves require another ASQL import; changing to a different dictionary or APPX installation requires checking that installation's definitions and values.

Fix Driver Setup Error 126

Error 126 means that a driver or setup DLL, or one of its dependencies, could not be loaded. Retain the complete Windows message, including the DLL path. Check that the matching 32-bit package is installed and that the [DSN](#) points to its actual `dhsodbc.dll`.

The 5.5.2 [DSN](#)-creation action completed on this test computer, which already had Microsoft's legacy CTL3D32 installed from the earlier cookbook tests. That does not establish the behavior of a clean Windows installation without that file. If the error identifies a missing CTL3D32 dependency, see the [documented Microsoft-file repair](#). Check the dependency first; an error number alone does not identify it.

Fix the ASQL System Directory Error

A fresh APPX installation must finish its normal first-start configuration and licensing before ASQL is used. In this clean 5.5.2 test, normal startup generated `0SA\Struct\SYSDIR.dat`; no manual structure repair was needed.

If ASQL reports **Can't open file**, check the intended APPX installation, [service-account](#) permissions, and the generated structure alongside the existing system data. An absent generated structure is different from a missing APPX file definition. The 5.5.4 [troubleshooting example](#) shows a controlled, one-file structure-generation repair. Use the matching release's System Administration and preserve existing data; do not initialize or scratch system files to repair a structure.

Fix Import Errors Caused by Field Names

These errors come from the gateway when ASQL, or any other SQL tool, runs a statement for an APPX file. The full text appears on ASQL's status line after `[Dharma][ODBC Dharma Driver ...][Dharma/SDK]`.

Error	What it means	What to do
-20003 Syntax error	A name is a word that is refused everywhere, starts with a digit, or contains something other than letters, digits and underscores.	Rename the column in the script, or put the name in double quotes.
-20049 Keyword used for a name	A column name is a word this gateway refuses as a new name. Queries could still use it.	Rename the column, or quote it once in the <code>CREATE TABLE</code> statement.
-20248 Identifier too long.	A name is longer than 32 characters.	Shorten the name in the script.
-20010 Duplicate column specification	Two columns have the same name, often two field names that differ only in punctuation.	Rename one of them in the script.
-20041 Table/View/Synonym already exists	The file already has a SQL table. After a rename with Try Again, Import Table shows this error and puts the APPX names back in the pane, but the table keeps the new names.	To replace the table, choose Drop Table first. To see the names it has, query it.
-20005 Table/View/Synonym not found	The table doesn't exist, for example a view whose table was dropped, or a file that hasn't been imported.	Import the file again.
-20006 Column not found/specified	A column name in a query doesn't match. A name created in double quotes has to be written with the same capitals.	Use the exact name, for example "suffix", not "SUFFIX".

For the rules behind these errors and the full lists of refused words, see [SQL Name Rules and Reserved Words](#) and [How APPX Files and Fields Become SQL Tables and Columns](#).