



APPX LIBRARY

APPX ODBC 6.2.1 Cookbook (64-bit)

APPX 6.2.1, 64-bit clients

Updated October 20, 2026: ASQL expert mode, SQL names, reserved words and views checked. Tested October 15, 2026. Read the observed Unicode text and filtering failures before using query output.

APPX Software, Inc.

Made on October 20, 2026 from the pages of the APPX Library, <https://wiki.appx.com/manuals/library/appx-odbc-621-64/>. The pages are the current version.

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 embeds the public examples listed below. Read README-offline.txt and verify extracted files with SHA256SUMS.txt. Attachments do not run automatically. Live queries and refresh require the matching driver, gateway, APPX access and your authorized login. No saved password is included. Expected native values and observed client values are identified separately; an observed failure sample is not correct business data.

Embedded file	SHA-256
APPX-621-64-COUNTRY.odb	41b0b9a856a168236c6f4ccefb7b2148eb63abba0f82b5b8a3d05d985960f93
APPX-621-64-COUNTRY.ods	a0b7616d942aa24b7fa813b4eae674b846169cefaa1422c31de92bc89cf6718f
APPX-621-64-COUNTRY.xlsx	377232d3769bc3b72ca577f85a0f897c28730eaf086581553127a2b3425c8740
COUNTRY-columns.csv	1c3c4cef75989041641fb3281759380e8cc34db29e55e8c66af6a5c0f9af157e
COUNTRY-expected.csv	2bc3db2064958174f0644a567dc98ca87cd32803e0de2f5ebbdca0aa2df6e5523
COUNTRY-observed.csv	d624a7beb99104504db2759b96a036240cafb702bbca1cb4e9d01db331474436
COUNTRY-report.sql	47f9348f0dd74d0b34599aee6b910289d6fcf8061cb176f030990a624e8b8af3
COUNTRY.sql	fb25dbf415bc4251bb57576b32e49e6fa292edd2d8953de12b6a6ebccbbe636a
PROSPECT-unicode-expected.csv	5bbcb2ac7a1e32b80e44a1d3a49c26755485c60d5a51adaf49bea8d9f56e1f29
PROSPECT-unicode-observed-codepoints.csv	a6e81af0ee53dd77a23f2d12897bcbb1b54d1b2ca1329b0afa55e43e8c2f3535
PROSPECT-unicode.sql	fa8ae13802e1ce61b634ee9d957bc46e959f9e507d5a2008cce1e23114c7bfdc
README-offline.txt	64015c2e651c8b79e69be9179f15c2329a5ae58b8e2c83f580d838dc167eb807
SHA256SUMS.txt	12a6b6fd1a18d0d0edbf497a7b2d8b07f58153077eeFeba80407e4e97e873942
appx-odbc-621-64-offline.zip	5042f02f2fa45f4cb414430b90497a5190a1746d7b170edc9f6a44fcf3fd6287
appx-odbc-621-64-scripts.zip	c300efd88c0d66ff6869bb31e366924181ab5a12768cb2508d3238d61a381a0e
query-appx-country-64.ps1	a14683b60a4a450aa99400948cd6f81b11004694fffb6ccefcceb76fbd2166b1d
read-appx-odbc-64.ps1	592904baae0ac5f2572b14843949d42ba3776779c6163c395f3528d8df81973d

Contents

Connection glossary: Service, sqlnw, DSNs and ports	7
Chapter 1 Before You Start	5
What You Need and Where to Begin	5
Check Versions and Choose 64-bit Programs	5
What Works Better in Later Releases.....	7
Identify the Server, Ports and Connection Names	7
Connection Glossary: Service, sqlnw, DSNs and Ports	7
Decide What to Install on Each Computer.....	12
Download the Tested Example Files	12
Chapter 2 Install and Check APPX ODBC	13
Install the 6.2.1 Gateway and ASQL	13
Start the Gateway and Check Required Microsoft Files.....	14
Install the Driver on Another Windows Computer	15
Check the Client-to-Gateway and Gateway-to-APPX Connections.....	15
Check the Saved AppxODBC Connection (DSN).....	15
Chapter 3 Make APPX Files Available to SQL	17
What ASQL Does and Where to Find It.....	17
Add the COUNTRY Table to SQL with ASQL	17
Check Every COUNTRY Record and Field.....	18
Show Only the APPX Application You Need.....	22
When to Create or Update SQL Table Definitions.....	22
How APPX Files and Fields Become SQL Tables and Columns	22
SQL Name Rules and Reserved Words.....	24
Review and Edit the SQL That ASQL Creates.....	27
Use a View for Friendly Names and Shorter Queries.....	29
Check Limits for Large Tables and Long SQL Statements.....	30
Chapter 4 Read APPX Data with WinSQL	32
Connect WinSQL and Read COUNTRY	32
Chapter 5 Read APPX Data with Excel, Access and LibreOffice	33
Choose 64-bit Excel and Access	33
Load COUNTRY into Excel and Refresh It.....	33
Check the Access Import Result.....	34

Choose 64-bit LibreOffice Calc and Base	35
Connect LibreOffice Base and Save a COUNTRY Query	35
Load COUNTRY into LibreOffice Calc and Refresh It	36
Help LibreOffice Find a Saved Base Connection.....	38
Chapter 6 Create Reports with Crystal Reports 2025	40
Choose a Compatible Crystal Reports Version	40
Create a COUNTRY Report	40
Chapter 7 Read APPX Data with 64-bit PowerShell	43
Run a Query with a Secure Login Prompt.....	43
Check the Driver Architecture and Gateway Connection	44
Work with Query Results in PowerShell.....	45
Filter Records with SQL Parameters	45
Export COUNTRY to CSV and JSON.....	46
Summarize and Check the Returned Data.....	47
List Tables and Columns and Identify the Driver	47
Set Query Limits and Read Error Details	48
Chapter 8 Solve Connection and Import Problems.....	50
Set Connection and Diagnostic Options.....	50
Record a Diagnostic Log, Then Turn Logging Off	53
Check the Driver, Ports and Login Before Retrying	55
Check the dhsodbc.ini and appx.env Settings Files	55
Fix Driver Setup Error 126	56
Fix the ASQL System Directory Error	56
Fix Import Errors Caused by Field Names.....	56

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 64-bit Windows programs to APPX 6.2.1 through the matching gateway and supplied driver. Install the components, create SQL definitions with ASQL, then check complete field values before using a query result.

Observed limitation: the tested stack returns COUNTRY and PROSPECT records, but Unicode-storage text is truncated or damaged and text predicates return incorrect rows. Numeric, date and RAW fields match the independently exported native files. No verified ODBC client setting restores the missing text. For the nearest complete sample-data success, see the [5.5.4 32-bit configuration](#), which uses its matching older engine/gateway and the 6.2.1 32-bit client driver. That result requires a different engine/gateway combination; changing this edition's client driver alone is not an accepted repair.

Read [the tested versions](#) and [the field-value comparison](#) before continuing. A successful login or correct record count does not establish correct data.

Requirements

- A licensed APPX 6.2.1 installation and an authorized APPX login.
- The matching Windows 64-bit gateway and driver-with-ASQL packages.
- Administrator access for installation and [System DSN](#) creation.
- A 64-bit client application and the 64-bit ODBC Administrator.
- The DMO database/application COUNTRY sample for the recipes. Use a disposable test copy when importing metadata.

Check Versions and Choose 64-bit Programs

Component	Tested identity	Architecture
Windows	Windows 11 Pro 25H2, build 26200.6584	64-bit
APPX engine	6.2.1.0; build marker January 5, 2026	64-bit
APPX ODBC gateway	6.2.1.0 from the official 6.2.1 package	64-bit
Supplied ODBC driver	Dharma 14.01.00.0000	64-bit
Supplied ASQL	Executable resource version 6.1.0.0 in the 6.2.1 package	64-bit
WinSQL	15.0.261.801	64-bit

Component	Tested identity	Architecture
Excel and Access	Microsoft 365 Apps 16.0.20430.20182	64-bit
LibreOffice Base and Calc	26.2.6.3	64-bit
Crystal Reports	SAP Crystal Reports 2025 Patch 1, 14.4.0.5335	64-bit
Windows PowerShell	5.1.26100.6584	64-bit

The test retained the existing VM engine at `C:\appx\Appx.621-64bit\Appx.exe`, verified as 6.2.1.0 with build marker January 5, 2026, 09:44:08 and SHA-256

`D77B1F45FB897AFE4030DDFD9E38233B3AB8FD2CF0666558827429366AFBF896`. The matching gateway and driver were installed from the checked vendor packages. The engine EXE package was downloaded and hashed separately, but its payload was not independently extracted and matched byte for byte to this retained engine. These results identify the installed stack; they do not establish a clean Windows baseline or an exact source-to-binary build record.

Match the **driver to the client program's architecture**. Windows being 64-bit does not establish the bitness of an installed application. The ODBC SQL component reports 14.01 as its DBMS version; that value is not the APPX engine release.

Results by client

Client or check	Observed result	Recipe
ASQL	COUNTRY and PROSPECT definitions imported.	Import COUNTRY
WinSQL	Eight COUNTRY rows and ten PROSPECT rows; Unicode text incorrect.	WinSQL query
Excel	Saved workbook, full close, fresh password prompt and refresh completed; text incorrect.	Excel refresh
Access	Authentication and catalog succeeded; COUNTRY import stalled and became Not Responding.	Access result
Base	Saved direct query reopened, prompted for a password and refreshed eight rows; new database wizard save/open stalled.	Base query
Calc	Saved workbook, full exit, fresh password and refresh completed; 22 of 72 values differ from native text.	Calc refresh
Crystal Reports	Direct RDO command, save, full exit, fresh password/refresh and PDF exports complete; 22 of 32 projected values incorrect.	Crystal report
PowerShell	Objects, metadata and exports returned; text predicates incorrect; intermittent connection-open stalls.	PowerShell results

What Works Better in Later Releases

No later configuration is accepted as an improvement until its matching engine, gateway, supplied driver and client architecture have been tested independently. The separately labeled 6.2.2 Beta 1 editions must not be treated as the final 6.2.2 release or assumed to repair these results.

Capability or capacity	This edition	Verified later configuration and procedure
Complete Unicode text and text predicates	Fails the current native-value comparison.	No verified later improvement recorded yet.
SQL statement capacity	Reports 35000; 32769 succeeds and 33000 fails.	No verified later improvement recorded yet.
Client import reliability	Direct-query routes and intermittent stalls are documented separately.	No verified later improvement recorded yet.

Identify the Server, Ports and Connection Names

Name	Meaning in this recipe
<code>localhost</code>	Gateway and APPX backend share the test computer. Use the real hostname when they are separate.
<code>AppxODBC</code>	64-bit System DSN on the client computer.
<code>appx</code>	Gateway SQL dictionary selected by the DSN's Database field.
<code>DMO / DMO</code>	APPX database and application IDs used for the sample.
<code>1992</code>	Measured gateway TCP listener.
<code>31062</code>	This test installation's APPX login-manager port, not a general product default.
<code>blank Service</code>	Uses the tested client's sqlnw lookup; fallback 1992 when the mapping is absent.

Connection Glossary: Service, sqlnw, DSNs and Ports

Start here when a setup page uses an unfamiliar name. These definitions apply throughout this 6.2.1 64-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 field**. 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:

<code>sqlnw</code>	<code>1992/tcp</code>
--------------------	-----------------------

This line means “when this computer looks up `sqlnw` for TCP, use port 1992.” 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 **Supplied Dharma 14.01 x64** client falls back to **TCP 1992**. 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 script adds `sqlnw 1992/tcp` when its substring check finds no existing 1992 text. It does not reconcile duplicate service names. This test deliberately removed a conflicting older mapping and verified one `sqlnw 1992/tcp` entry against the actual listener.

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 **1992**. 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 **64-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\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 **1992/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:1992**. The gateway separately contacts the APPX login manager at **localhost:31062**, 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-621-31062-OSAuth**. Its name identifies the managed login-manager process, while **31062** 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 **Socket**. 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 1992	Tested ODBC gateway listener for APPX 6.2.1	Match the gateway process PID to its listener, then check the client's <code>sqlnw</code> mapping or chosen custom alias.

Port or name	Meaning	What to check
TCP 1992	Fallback in this edition's tested Supplied Dharma 14.01 x64 client when <code>sqlnw</code> has no mapping	A fallback is driver-specific; it does not override a conflicting named mapping.
TCP 31062	APPX login-manager endpoint in this cookbook	Match <code>APPX_DATA_SERVER</code> to the running login-manager service and intended APPX engine.
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 31062 instead; retain the site's configured port.
<code>sqlnw</code>	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.
<code>appx</code>	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

Install the gateway where it can reach APPX. Install the driver on every client computer. Install ASQL on the administration computer that maintains SQL definitions. Client applications reuse the shared gateway dictionary; they do not each import a private copy of APPX records.

The examples put all components on one Windows computer. For a remote client, change [DSN Host](#) to the gateway hostname. For a remote APPX backend, change the gateway dictionary's `APPX_DATA_SERVER`. These are separate connections.

Download the Tested Example Files

Use the [complete offline ZIP](#), [PowerShell scripts ZIP](#), [README](#) and [checksums](#). [COUNTRY-expected.csv](#) contains the independent native values. [COUNTRY-observed.csv](#) records the different ODBC result; do not substitute it for correct text.

The examples contain no saved password. Refreshing a connection requires your authorized login and live APPX access. Safari and Edge may hide PDF attachments; use the ordinary ZIP links or download the PDF and open its Attachments panel in desktop Acrobat/Reader.

CHAPTER 2

Install and Check APPX ODBC

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

Install the 6.2.1 Gateway and ASQL

1. Preserve the existing gateway dictionary, settings and service configuration. Follow the vendor's [uninstall-first replacement procedure](#) when replacing an older installation.
2. Download the official [6.2.1 64-bit gateway](#) and [64-bit driver with ASQL](#). Check them against the vendor's 6.2.1 SHA-256 manifest.
3. Run gateway setup as administrator. Use `C:\Program Files\AppxODBCserver`. Enter the actual APPX backend hostname and login-manager port; this installation uses `localhost` and `31062`.
4. Check `C:\Program Files\AppxODBCserver\appx.dbs\appx.env`. The tested backend assignment is `APPX_DATA_SERVER=localhost:31062`.
5. Install the matching driver-with-ASQL into `C:\Program Files\AppxODBCDriver`. Create **System DSN AppxODBC**, Host **localhost**, Database **appx**. Leave optional Service blank.
6. Check the registered [Windows service](#) executable, actual process architecture and listener before importing definitions. An installer filename alone does not identify the running gateway.

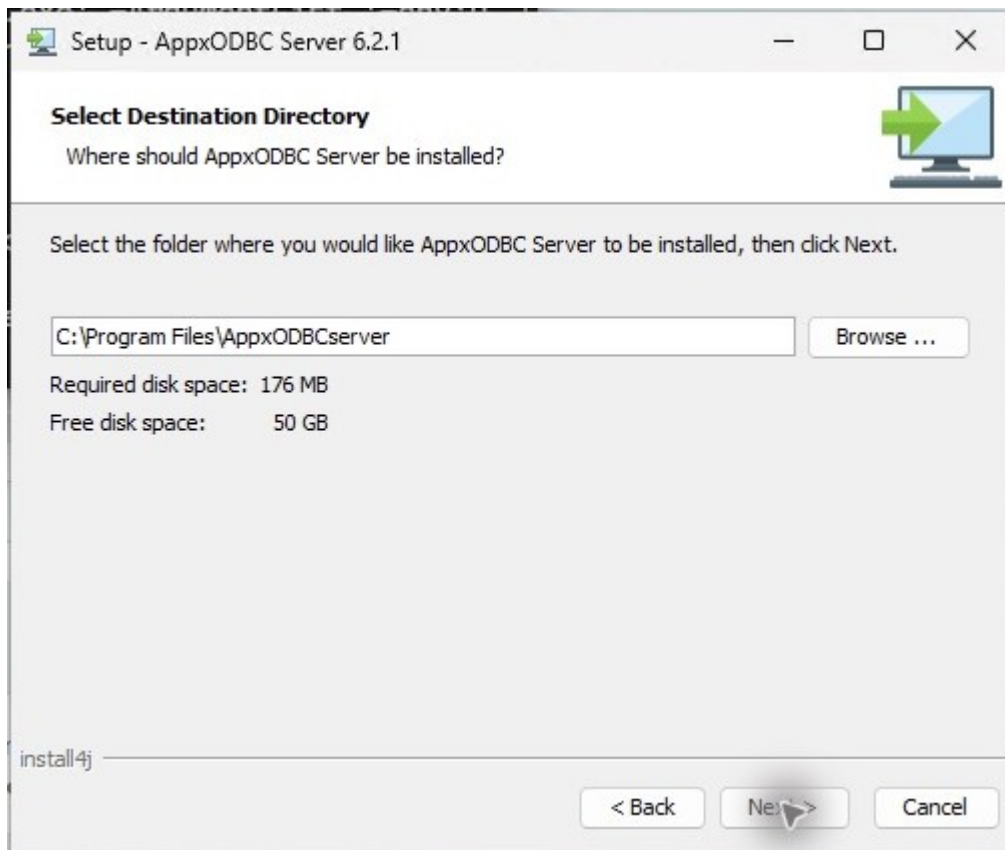


Figure 1. The matching 64-bit gateway installer selects its Program Files destination.

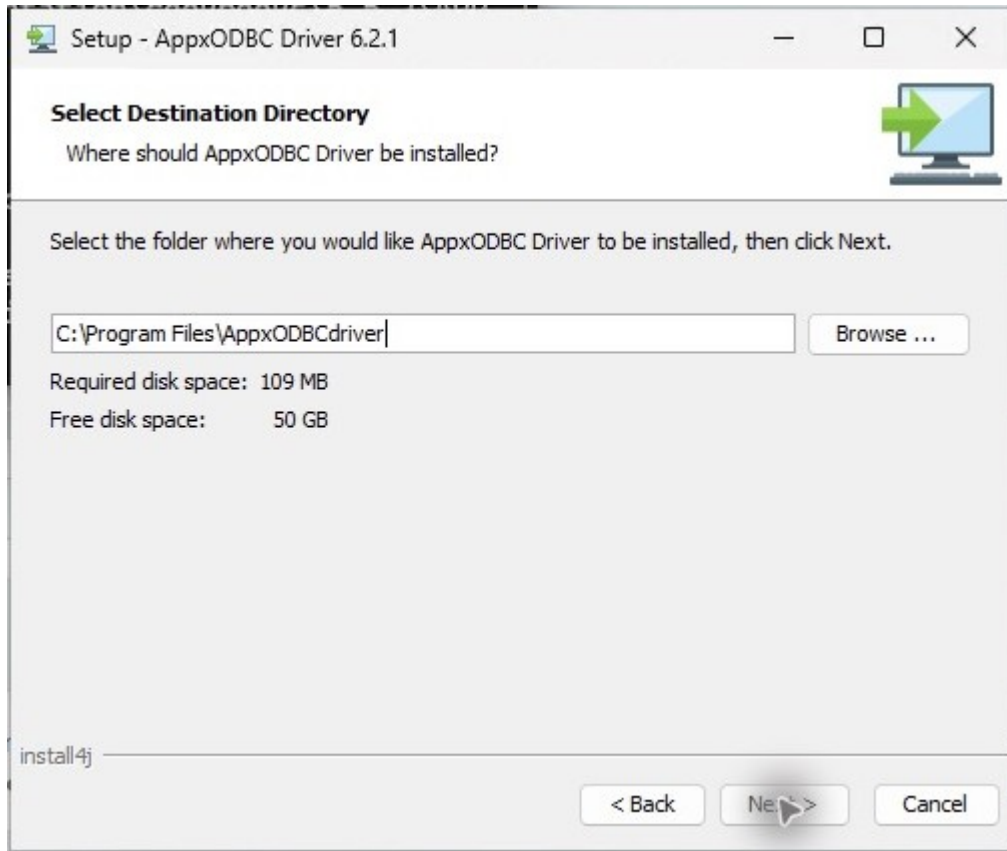


Figure 2. The matching driver-with-ASQL installer uses its separate Program Files destination.

The package hashes are 43095ba483ad810646906681247ce9167e6604c4c4227efb2a69feb71c094b45 for the gateway and 970b98cd4db179b233509434a164b6e6a5332fc3934592299a0f8ff4b8033323 for driver-with-ASQL. The unlisted 6.2.1 server ZIP inspected during testing contained a 6.2.0 engine; use the official 6.2.1 EXE and verify the installed engine's own version.

Start the Gateway and Check Required Microsoft Files

Start and Check the Gateway

Open an administrator CMD window and inspect the Windows background service:

```
sc.exe qc "Appx ODBC Data Gateway Server for Windows"
sc.exe query "Appx ODBC Data Gateway Server for Windows"
netstat -ano | findstr :1992
```

Confirm that BINARY_PATH_NAME selects the intended 64-bit executable under Program Files, and associate its PID with the listener. The tested executable is 6.2.1.0, SHA-256 5E456669E73870E8931A26DF687066F5B79E7256F92C990973EBC030BBC0EE3D.

The local test began with an older 32-bit [gateway service](#) still registered. Its retained path required an explicit correction before the current tests. That installation over an older gateway did not establish a supported replacement-installer defect. Follow the uninstall-first procedure and inspect the result on your computer.

If a Component Will Not Start

Do not copy a 5.5.3/5.5.4 missing-library repair into this edition automatically. Check the actual missing filename and architecture if a component fails to load. Use matching official Microsoft runtime packages rather than unrelated DLL-download sites.

Install the Driver on Another Windows Computer

Install the official 64-bit driver package on a computer running a 64-bit client. Open `C:\Windows\System32\odbcad32.exe`, select [System DSN](#), and create/check **AppxODBC** with the gateway's Host and Database **appx**. Leave Service blank for the standard mapping. The tested driver is Dharma 14.01.00.0000, SHA-256 `C71429E323CA64464F62E1A8126697DCEC77F1662F49668E6534B5C5C4A7F7B5`.

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

Connection	Setting	Tested endpoint
Client to gateway	DSN Host and Service; client's services file	localhost:1992
Gateway to APPX	Dictionary appx.env APPX_DATA_SERVER	localhost:31062

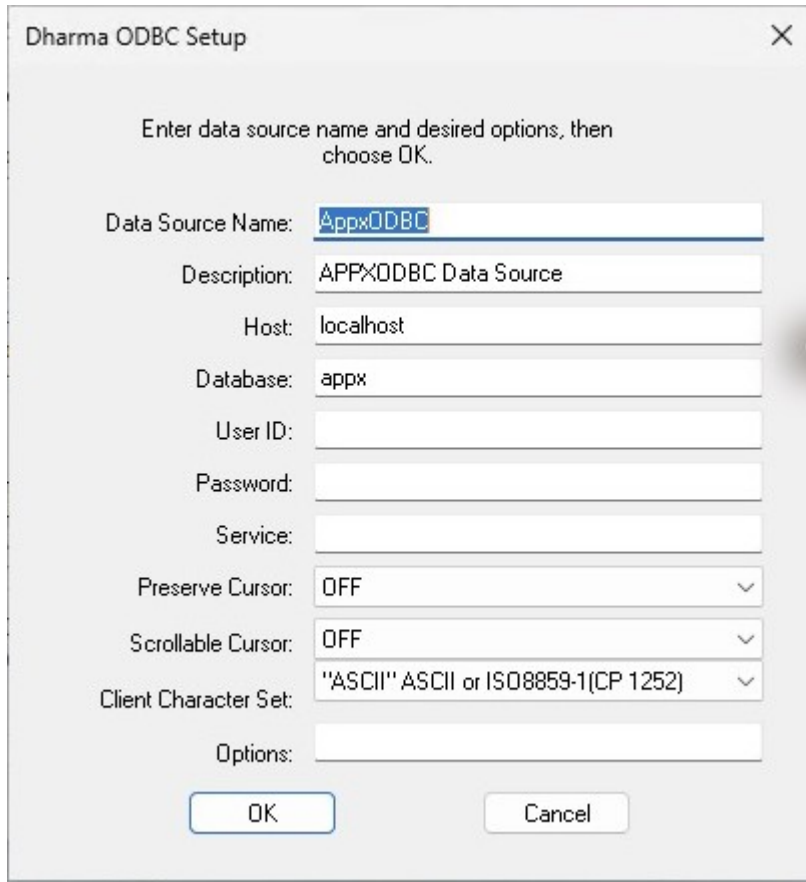
Test each [endpoint](#) separately. An open gateway port does not prove APPX authentication or file access. Configure access through the site's normal firewall process. The client [services-file mapping](#) resolves a TCP name; it does not start a [Windows service](#).

Check the Saved AppxODBC Connection (DSN)

Check the Client Driver and DSN

1. Open the **64-bit** ODBC Administrator, [System DSN](#), AppxODBC, Configure.
2. Check Host **localhost**, Database **appx** and optional Service **blank**. Keep the supplied 64-bit AppxODBC Driver.
3. Check the client's `C:\Windows\System32\drivers\etc\services`. The deliberate tested mapping is `sqlnw 1992/tcp`. Resolve duplicate names rather than assuming the last line wins.
4. Check the gateway dictionary's backend and the intended APPX login-manager instance. The tested backend is [Socket](#), port 31062.

5. Close existing clients before retesting. A saved workbook/report can retain settings that override the [DSN](#); check its connection too.



Dharma ODBC Setup

Enter data source name and desired options, then choose OK.

Data Source Name: AppxODBC

Description: APPXODBC Data Source

Host: localhost

Database: appx

User ID:

Password:

Service:

Preserve Cursor: OFF

Scrollable Cursor: OFF

Client Character Set: "ASCII" ASCII or ISO8859-1(CP 1252)

Options:

OK Cancel

Figure 3. The actual AppxODBC configuration uses Host localhost, Database appx and blank Service, with no saved user name or password.

Independent loopback probes measured blank-Service fallback 1992 and selection through both [sqlnw](#) and a custom named mapping. Those port-selection tests used synthetic credentials; the separate live sample queries establish the actual APPX connection.

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 creates SQL metadata describing APPX files. It does not copy their live records into a replacement SQL database. Run `C:\Program Files\AppxODBCDriver\asql.exe` from the matching 6.2.1 package. Its resource version 6.1.0.0 is retained in that package; do not select a different package merely to make the resource string match.

Add the COUNTRY Table to SQL with ASQL

1. Open the matching ASQL, choose AppxODBC and enter your authorized APPX login.
2. Keep the gateway dictionary **appx**. Select APPX database **DMO**, application **DMO**, file **COUNTRY**.
3. Review the selected file and import its definition. Keep the existing field order and types.
4. Import PROSPECT separately when checking Unicode, RAW, numeric and date behavior beyond COUNTRY.
5. Use a fresh client connection and [check every field](#).

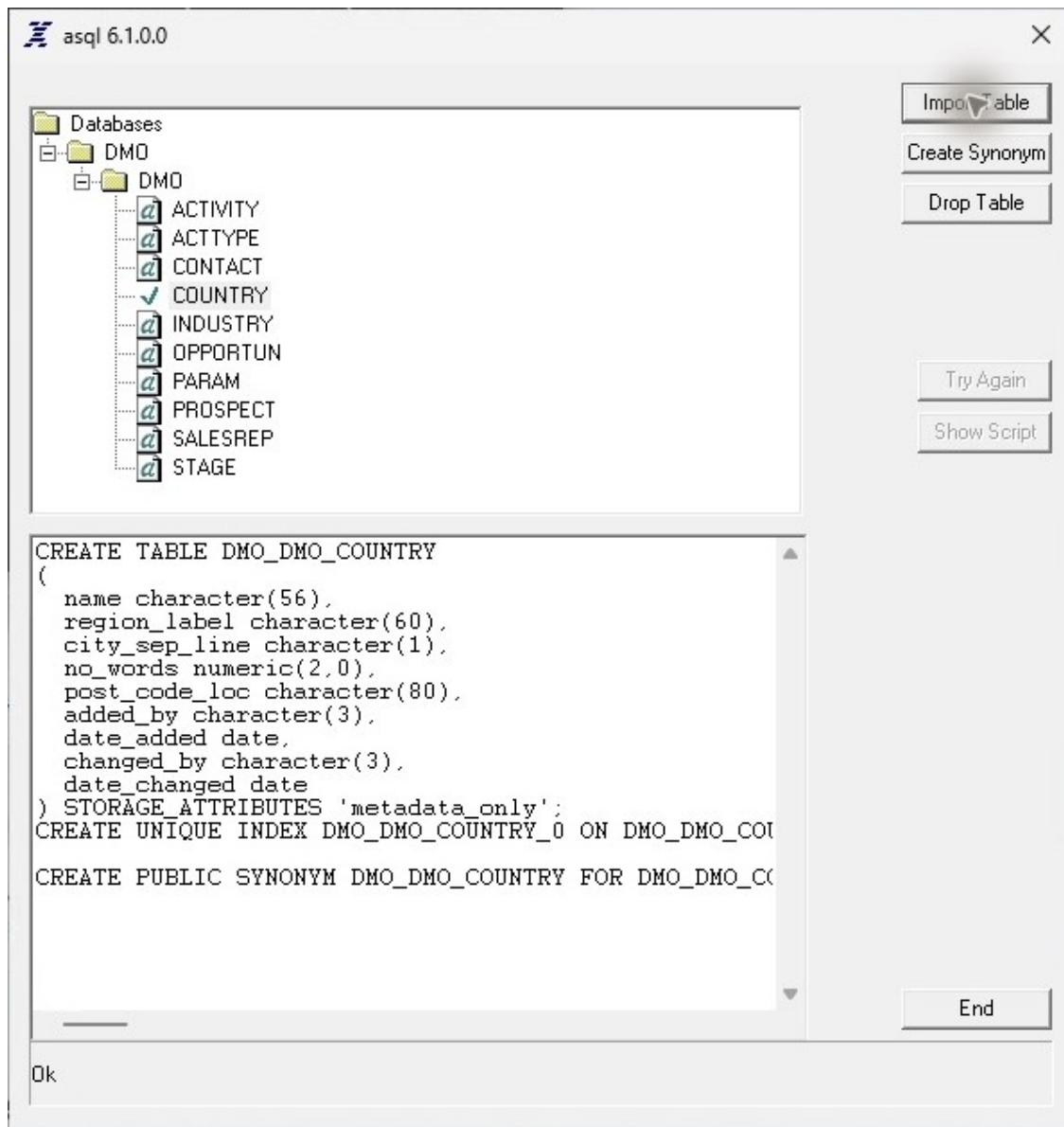


Figure 4. The matching 64-bit ASQL marks COUNTRY imported and displays its generated SQL.

Both definitions imported on the matching 64-bit stack. Import completion does not repair the returned Unicode text.

Check Every COUNTRY Record and Field

The independent native sample contains eight records and nine fields. The ODBC result also contains eight records, but the names are C, F, I, N, R, U, U, U. The native names are Canada, France, Ireland, Netherlands, Russia, UK, United States and USA.

All six numeric/date/RAW COUNTRY fields agree in all eight records. The Unicode name field agrees in zero records; the other Unicode fields agree only where the native value is empty. The separate ten-record, 24-field PROSPECT comparison produces the same distinction. Together these checks compare **312 field values**. A subsequent fresh-process comparison after removing the inherited RDBMS debug-log overrides reproduced exactly the same field differences in both complete samples. This repeats the data result; it does not establish that every diagnostic setting or connection wait has been isolated.

Compare the [native expected CSV](#) with the [observed ODBC CSV](#). Treat the damaged text as unusable.

`name = ?` with Canada returns zero rows; `no_words >= ? AND name <> ?` with 2 and Canada incorrectly returns three rows including Canada. This affects filtering as well as display. There is no accepted client-only repair. The [5.5.4 32-bit sample-data recipe](#) is a separate verified engine/gateway combination, not a downgrade or driver-replacement recommendation for an existing application.

Check actual non-ASCII values

COUNTRY's sample names use ASCII characters stored in Unicode fields. That checks the field storage path, but cannot establish support for other scripts. PROSPECT contains actual euro signs, accented Latin and Cyrillic. Its native values were decoded independently from the current APPX records before comparison with ODBC.

Numeric key and field	Native APPX value	Observed ODBC value
2, salesrep_name	Mona € Atwater	M
20, salesrep_name	Chambéry Anderson	C
21, address_1	43 Buffalo Place €	4
40, division_name	Автозапчасти	U+0010 U+0004
40, salesrep_name	Джо Байден	U+0014 U+0004

The last two results are control characters, written here as code points so they remain visible. They are not the literal words U+0014 and U+0004. The failure is therefore broader than every value being shortened to one character.

Run the numeric-key checks before testing text predicates. These three individual queries completed without putting non-ASCII text into their predicates:

```
SELECT prospect_no, salesrep_name
FROM DMO_DMO_PROSPECT WHERE prospect_no = 2;

SELECT prospect_no, salesrep_name
FROM DMO_DMO_PROSPECT WHERE prospect_no = 20;

SELECT prospect_no, salesrep_name
FROM DMO_DMO_PROSPECT WHERE prospect_no = 40;
```

Compare the [native Unicode sample](#) with the [observed code-point representation](#). The [combined numeric-key query](#) also completed and selected all four sample records, with the same damaged text. Its earlier bounded attempts waited during connection Open, before SQL execution. A later completed query does not erase those connection waits or establish a reliable connection-time workaround.

Check expressions and selection separately

Operation on the current native sample	Observed result
<code>CHAR_LENGTH(name)</code>	1 for all eight COUNTRY names, rather than their native lengths.
<code>UPPER(name)</code> and <code>SUBSTRING(name,1,3)</code>	Expressions execute, but return damaged text.
<code>name = 'Canada'</code> , <code>name LIKE 'Can%'</code> , and <code>name IN ('Canada', 'France')</code>	Zero rows, despite the corresponding native records being present.
<code>name = 'C'</code>	Zero rows; this is the expected negative control. It does not show that filtering uses the displayed first character.
<code>SELECT DISTINCT name ... ORDER BY name</code>	Eight rows, displayed as C, F, I, N, R, U, U, U. The row count alone passes neither uniqueness nor value acceptance.
<code>COUNT(DISTINCT name)</code>	6, although the native file contains eight distinct names.
<code>GROUP BY name</code> with <code>COUNT(*)</code>	Six groups. UK, United States and USA collapse into a displayed U group with count 3.
<code>ORDER BY name DESC</code>	Rows display U, U, U, R, N, I, F, C; the full native names remain lost.
<code>no_words >= 2 ORDER BY name</code>	The three expected records are selected, but their projected names are C, N and U.
<code>CHAR_LENGTH(salesrep_name)</code>	1 for the first nine PROSPECT records and 2 for the Cyrillic record; neither result matches its native text length.
<code>RAW added_by</code> and <code>changed_by</code>	All ten PROSPECT records agree with their native values.
COUNTRY name self-join	A later fresh connection executed after removing the inherited RDBMS trace overrides and returned zero rows, rather than the eight expected self-pairs. Earlier Open waits remain preserved.
PROSPECT self-join on numeric <code>prospect_no</code>	All ten expected key pairs agree with the native records. This does not establish text-key join correctness.
Numeric <code>GROUP BY no_words</code>	The three groups match native data: 0 has one record, 1 has four, and 2 has three.
<code>date_added IS NULL ORDER BY name</code>	Four rows, displayed as F, N, U, U. The projected names differ from the native values; duplicated damaged names do not establish every row association.

Literal and parameter predicates for the euro, accented and Cyrillic sales representative names also returned zero rows in the executions that completed. The euro literal predicate also completed in a later fresh process and returned zero rows. Earlier connection Open waits remain in the receipts; they are not SQL execution results.

Check the input path without reading a text field

Separate a stored-field failure from conversion of text supplied by the client. The input controls select a literal, or `CAST(? AS VARCHAR(120))`, from the record with numeric key 2. Parameter controls bind `VarChar` and `NVarChar` separately; their result expression is still `VARCHAR`.

ASCII Canada, Mona € Atwater and Chambéry Anderson survive all three completed input paths: literal, `VarChar` parameter and `NVarChar` parameter. The Cyrillic literal, `VarChar` and `NVarChar` controls all return question marks in completed executions; earlier `NVarChar` attempts waited during connection Open. Combining `e` plus `U+0301` returns `e?` through the literal and both parameter paths. The supplementary character `U+1F642` returns `??` through the literal and both parameter paths.

These controls document the tested conversions. They do not establish support for an `NVARCHAR` result type, native APPX storage of combining or supplementary characters, or a particular SQL collation. Loss through a `VARCHAR` expression does not, by itself, identify an APPX storage defect. Keep those limits distinct from the independently reproduced damage when reading existing native Unicode fields.

The SQL and parameter matrix covers 51 distinct cases across 72 bounded workers. Every case eventually executed: 38 comparisons differ from the independent expectation and 13 controls match. Twenty connection Open waits remain in the attempt history, including cases that later completed. These are comparison counts, not a count of separate defects. The two complete sample-file repeats after trace cleanup are additional checks.

Check ANSI and Unicode API calls

A separate native Windows ODBC check uses `SQLConnectA/SQLExecDirectA` and `SQLConnectW/SQLExecDirectW`, then retrieves the result independently as `SQL_C_CHAR` or `SQL_C_WCHAR`. Each case has a fresh process and connection. Returned buffer bytes are retained; the measured Windows ANSI code page is 1252.

Value or control	ANSI API, narrow result	ANSI API, wide result	Unicode API, narrow result	Unicode API, wide result
Literal Canada	Canada	Canada	Canada	Canada
Native Mona € Atwater, numeric key 2	M	M	M	M
Native Chambéry Anderson, numeric key 20	C	C	C	C
Native Джо Байден, numeric key 40	U+0014 U+0004	U+0014 U+0004	U+0014 U+0004	U+0014 U+0004
Cyrillic input literal	Question marks	Question marks	Question marks	Question marks
Supplementary input literal U+1F642	??	??	No execution: Open wait	??

Trailing SQL character padding is omitted from the displayed results. These checks reproduce native-field damage through all four API/buffer combinations; requesting a wide result buffer is not a verified repair. The attempts retain ten connection Open waits; the Unicode-API/narrow-buffer supplementary-input case still has no execution result after its bounded attempts. No value is inferred for that case.

The supplied driver does not export `SQLConnectW`. Windows can map a Unicode application's calls and buffers onto an ANSI driver, so a successful wide API call alone does not prove Unicode support end to end. See [Microsoft's Unicode driver rules](#). Keep the measured stored-field failures separate from code-page conversion of input literals.

Check every saved export and client document after refresh. UTF-8 CSV, JSON or a spreadsheet can faithfully preserve already damaged ODBC text; changing the export encoding cannot reconstruct the native value. The [Excel save and refresh check](#) compares all 72 saved COUNTRY values rather than accepting its eight-row count alone.

Show Only the APPX Application You Need

Select the intended APPX database and application before choosing files. This limits the working list and avoids importing unrelated definitions. It is not an access-control boundary and does not replace APPX permissions.

When to Create or Update SQL Table Definitions

Import when first making a file available to SQL or after a relevant APPX dictionary change. Ordinary record additions do not require re-import. Preserve custom SQL names, views, indexes and edited scripts before regenerating metadata. Close readers and follow the site's maintenance procedure when replacing a definition.

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> .

In APPX	In SQL	Example
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 <code>n</code> is the field's length in the record. In this edition a field stored as Unicode takes 4 bytes per character, so a 15-character field is declared <code>character(60)</code> .
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 <code>p</code> digits in all and <code>s</code> 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 64 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`. Keep names in double quotes to 32 characters: in testing, longer quoted names failed with `-30034`, or were accepted and then the next statement failed with `-30008 Packet Size Mismatch`.

These names were tried on this edition's gateway:

Name	Result
<code>name_2, _name, RegionLabel</code>	Accepted.
<code>2name</code>	<code>-20003 Syntax error. "2name" works.</code>
<code>region-label, region label</code>	<code>-20003 Syntax error. Both work in double quotes.</code>
A name containing <code>\$</code> , <code>#</code> , <code>@</code> or <code>.</code>	<code>-20003 Syntax error. It works in double quotes.</code>
A name of 65 characters or more	<code>-20248 Identifier too long.</code>
A table name that starts with a digit, such as <code>0SA_DMO_FILE</code>	<code>-20003 Syntax error. It works in double quotes.</code>

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 14.01 SQL server. It accepts most SQL function names, such as SUFFIX, PREFIX and LENGTH, which the 5.5.1 and 5.5.2 gateways (Dharma 9.01) refuse. It also refuses a few words those gateways accept, such as DAY, BOTH, LEADING and TRAILING. If you are moving from 5.5.1 or 5.5.2, check your column names against the list below: a column named DAY, for example, now has to be quoted in every query.

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 SUFFIX, PREFIX and LENGTH. The tables below show what this edition's gateway refused when each word was tried as a column name.

Refused everywhere: 224 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, BOTH, BY
C	CALL, CASCADE, CASE, CHECK, CLOSE, CLUSTERED, COALESCE, COLGROUP, COLLATE, COLUMN, COMMIT, COMPRESS, CONNECT, CONSTRAINT, CONTAINS, CONVERT, COUNT, CREATE, CROSS, CURRENT, CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP, CURRENT_USER, CURSOR
D	DATABASE, DATAPAGES, DATE, DAY, DAYOFYEAR, 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

Starts with	Words
H	HASH, HAVING, HOUR
I	IDENTIFIED, IDENTITY, IN, INDEX, INDEXPAGES, INDICATOR, INNER, INOUT, INT, INTEGER, INTERSECT, INTO, IS
J	JOIN
K	KEY
L	LEADING, LEFT, LIKE, LINK, LOCK, LONG, LVARBINARY, LVARCHAR
M	MAX, METADATA_ONLY, MIN, MINUS, MINUTE, MODE, MODIFY, MONEY, MONTH
N	NATIONAL, NATURAL, NCHAR, NEWROW, NOCOMPRESS, NOT, NOWAIT, NULL, NULLIF, NUMBER, NUMERIC
O	ODBC_CONVERT, OF, OLDROW, ON, OPEN, OPTION, OR, ORDER, OUT, OUTER
P	PCTFREE, PRECISION, PRIMARY, PRIVILEGES, PROCEDURE, PUBLIC
Q	QUARTER
R	RAW, REAL, REFERENCES, REFERENCING, RENAME, RESOURCE, RESTRICT, RESULT, RETURN, REVOKE, RIGHT, ROLLBACK, ROW, ROWID, ROWNUM
S	SECOND, SELECT, SESSION_USER, SET, SHARE, SIZE, SMALLINT, SOME, 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, TRAILING, TRANSACTION, TRIGGER, TYPE
U	UID, UNION, UNIQUE, UPDATE, USER, USING
V	VALUES, VARBINARY, VARCHAR, VARYING, VIEW
W	WEEK, WHEN, WHERE, WITH, WORK
Y	YEAR

Refused only as a new name: 44 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, AN, ARRAY
B	BEGIN, BIND, BINDING
C	CLEANUP, COMPLEX, CONTINUE, CVAR
D	DECLARATION, DESCRIBE, DESCRIPTOR, DHTYPE
E	EXEC, EXIT, EXPLICIT
F	FIELD, FOUND

Starts with	Words
G	GO, GOTO
I	IMMEDIATE, INPUT, INTERFACE
L	LIST
M	MAIN
O	OUTPUT
P	PREPARE
R	RECORD
S	SECTION, SERVICE, SHORT, SQL, SQLERROR, SQLWARNING, START, STOP
T	TIMEOUT, TPE
U	UNSIGNED, UUID
V	VARIABLES, VERSION
W	WHENEVER

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\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.

Button	What it does
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, and a box titled **Try Again Failed** opens with the message `OnTryAgain() import_script() failed`. Choose **OK** to close it; the real error is the one on the status line.

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
```

On this edition the text comes back damaged and the second query returns no rows, because of the Unicode text problem described in [Verify COUNTRY Records and Field Values](#). The view itself is fine: the same view on the 5.5.x editions 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

Distinguish an ODBC-reported capacity, an observed boundary and an ASQL source constant. The supplied 64-bit Dharma 14.01 driver reports 64-character identifiers, 2000 columns per table and a 35000-character statement limit.

ASQL's reviewed 6.2.1 source allocates and grows its generated script in 1024-byte chunks. That allocation unit is not a 1024-character SQL limit. The DDL editor and the driver/gateway statement transport remain separate constraints. Shorten names and remove unnecessary whitespace without changing field positions, semantics or quoted literal values; execute separate valid statements individually where supported.

The complete property inventory follows.

SQLGetInfo property	Reported value
SQL_DRIVER_VER	14.01.00.0000
SQL_DBMS_VER	14.01.00.0000
SQL_IDENTIFIER_QUOTE_CHAR	"
SQL_MAX_COLUMN_NAME_LEN	64
SQL_MAX_COLUMNS_IN_SELECT	0
SQL_MAX_COLUMNS_IN_TABLE	2000
SQL_MAX_STATEMENT_LEN	35000
SQL_MAX_IDENTIFIER_LEN	64
SQL_MAX_TABLE_NAME_LEN	64
SQL_MAX_SCHEMA_NAME_LEN	64
SQL_MAX_CATALOG_NAME_LEN	0
SQL_MAX_COLUMNS_IN_GROUP_BY	0
SQL_MAX_COLUMNS_IN_INDEX	100
SQL_MAX_COLUMNS_IN_ORDER_BY	0
SQL_MAX_INDEX_SIZE	0
SQL_MAX_ROW_SIZE	0
SQL_MAX_ROW_SIZE_INCLUDES_LONG	N
SQL_MAX_TABLES_IN_SELECT	250
SQL_MAX_CHAR_LITERAL_LEN	2000
SQL_MAX_BINARY_LITERAL_LEN	2000
SQL_MAX_DRIVER_CONNECTIONS	0

SQLGetInfo property	Reported value
SQL_MAX_CONCURRENT_ACTIVITIES	0

A reported zero does not prove an unlimited usable capacity. ODBC's DBMS version identifies the SQL component, not APPX. No maximum-size table or row acceptance is claimed by these property calls.

Observed statement lengths

The 36-character `SELECT COUNT(*) FROM DMO_DMO_COUNTRY` query, padded with ASCII trailing spaces, succeeds at 32769 characters. Fresh independent execution at 33000 fails with SQLSTATE k0k14, Dharma -30034. At 35001 the driver returns HY090/-19. Earlier independent 34999/35000 executions also failed; a shared-connection diagnostic varied after an earlier failure. No exact maximum is established.

SQL text length, characters	Completed execution result	Connection-only observations
36	Count 8.	A later independent connection waited at Open.
32768	Count 8.	None accepted as a contrary SQL result.
32769	Count 8 in repeated fresh workers.	None accepted as a contrary SQL result.
33000	k0k14, native -30034, in repeated fresh workers.	None accepted as a contrary SQL result.
34000	No completed execution in the bounded worker.	Waited at Open.
34999	k0k14, native -30034.	Keep its completed execution separate from other lengths.
35000	k0k14, native -30034, in the earlier independent worker.	A later independent worker waited at Open.
35001	HY090, native -19, Invalid string or buffer length.	An earlier worker waited at Open.

Some fresh workers stall during connection Open, before executing any SQL. Those cases do not measure statement length. A per-process test bound contains a stall; it is not a driver-enforced timeout or a retry repair.

CHAPTER 4

Read APPX Data with WinSQL

Connect, run a query and check the returned records.

Connect WinSQL and Read COUNTRY

1. Open **WinSQL64**. Its installation can be under Program Files (x86) despite the executable being 64-bit; inspect the actual executable.
2. Select AppxODBC, enter your authorized login and connect.
3. Execute `SELECT * FROM DMO_DMO_COUNTRY ORDER BY name`.
4. Check eight rows and all nine fields against the native expected result.
5. Run `SELECT * FROM DMO_DMO_PROSPECT ORDER BY prospect_no` when checking the imported PROSPECT definition.

	name	region_label	city_sep_line	no_words	post_code_loc	added_by	date_added	changed_by	date_chang
1	C	P	0	2	A	JPN	2001-03-01	SPF	2001-05-31
2	F	C	0	1	S		NULL	JPN	2001-03-01
3	I	C	0	0		JPN	2001-03-07	JPN	2001-03-07
4	N		0	2	B		NULL	JPN	2001-03-02
5	R	C	0	1	A	ADM	2023-08-08	ADM	2023-08-08
6	U	C	1	2	S		NULL	LDN	2001-03-08
7	U	S	0	1	A		NULL	SPF	2001-05-29
8	U	S	0	1	A	SPF	2001-05-29	SPF	2001-05-29

Figure 5. Fresh WinSQL64 COUNTRY result: eight rows, with truncated Unicode values.

The fresh native WinSQL64 grids show eight COUNTRY and ten PROSPECT rows with the Unicode text failure. A count-only query is useful for connectivity but cannot pass value acceptance.

CHAPTER 5

Read APPX Data with Excel, Access and LibreOffice

Follow the tested procedure for each desktop program.

Choose 64-bit Excel and Access

Check File, Account, About in Office and confirm **64-bit**. The tested Excel and Access executables are Microsoft 365 Apps 16.0.20430.20182 under `C:\Program Files\Microsoft Office\root\Office16`. They require the 64-bit [System DSN](#), independently of the Windows architecture. Preserve existing Office documents before changing an Office installation.

Load COUNTRY into Excel and Refresh It

The saved [Excel workbook](#) uses an explicit query through AppxODBC. It contains an observed sample snapshot with incorrect Unicode text, clearly identified by the [native-value comparison](#). Do not use that snapshot as correct country-name data. The separate [5.5.4 32-bit Excel recipe](#) passes complete sample values on its documented older engine/gateway combination.

1. Install/check this edition's driver and [DSN](#), then download the workbook to a working directory.
2. Inspect its connection under Data, Queries & Connections, Properties. Keep [DSN AppxODBC](#) and clear any old explicit Service override. The command reads `DMO_DMO_COUNTRY` ordered by name.
3. Enable this known external connection when Office requests it. Choose **Data, Refresh All** and enter your authorized APPX login when prompted. Keep the password unsaved.
4. Check **eight rows and all nine fields**. The tested workbook refreshes but displays the Unicode truncation.
5. Save, close Excel completely, reopen the workbook, enter the password again and refresh. Check the new result before saving it.

	A	B	C	D	E	F	G	H	I	J	K	L
1	name	region_label	city_sep_line	no_words	post_code_loc	added_by	date_added	changed_by	date_changed			
2	C	P	0	2	A	JPN	3/1/2001	SPF	5/31/2001			
3	F	C	0	1	S			JPN	3/1/2001			
4	I	C	0	0		JPN	3/7/2001	JPN	3/7/2001			
5	N		0	2	B			JPN	3/2/2001			
6	R	C	0	1	A	ADM	8/8/2023	ADM	8/8/2023			
7	U	C	1	2	S			LDN	3/8/2001			
8	U	S	0	1	A			SPF	5/29/2001			
9	U	S	0	1	A	SPF	5/29/2001	SPF	5/29/2001			
10												

Figure 6. The saved 64-bit Excel workbook after fresh reopen and refresh: eight records with the same text failure.

The fresh reopen/password/refresh sequence completed on this stack. Independent inspection of the saved OpenXML workbook checks 72 field values: numeric/date/RAW fields agree, Unicode text does not. No saved password or savePassword flag is present. This direct-query success does not establish that every Power Query navigator or catalog-browsing route will complete.

Check the Access Import Result

The fresh Access import did not complete. A separate successful direct-query client on this same stack is [Excel](#), which still has the Unicode text limitation. The nearest verified full-value Access import is the [5.5.3 32-bit configuration with the 6.2.1 x86 Dharma 14.01 driver](#). That older engine/gateway combination does not establish a repair for this 64-bit stack.

The bounded reproduction used a new blank database and External Data, New Data Source, Other Sources, ODBC Database, **Import**. Machine Data Source AppxODBC authenticated and displayed its table picker. Selecting `appx.dmo_dmo_country` left Import Objects at **Importing** for more than two minutes. Cancel did not complete, and the actual dialog displayed **Not Responding**. Only the identified Access test process was stopped; its database was preserved.

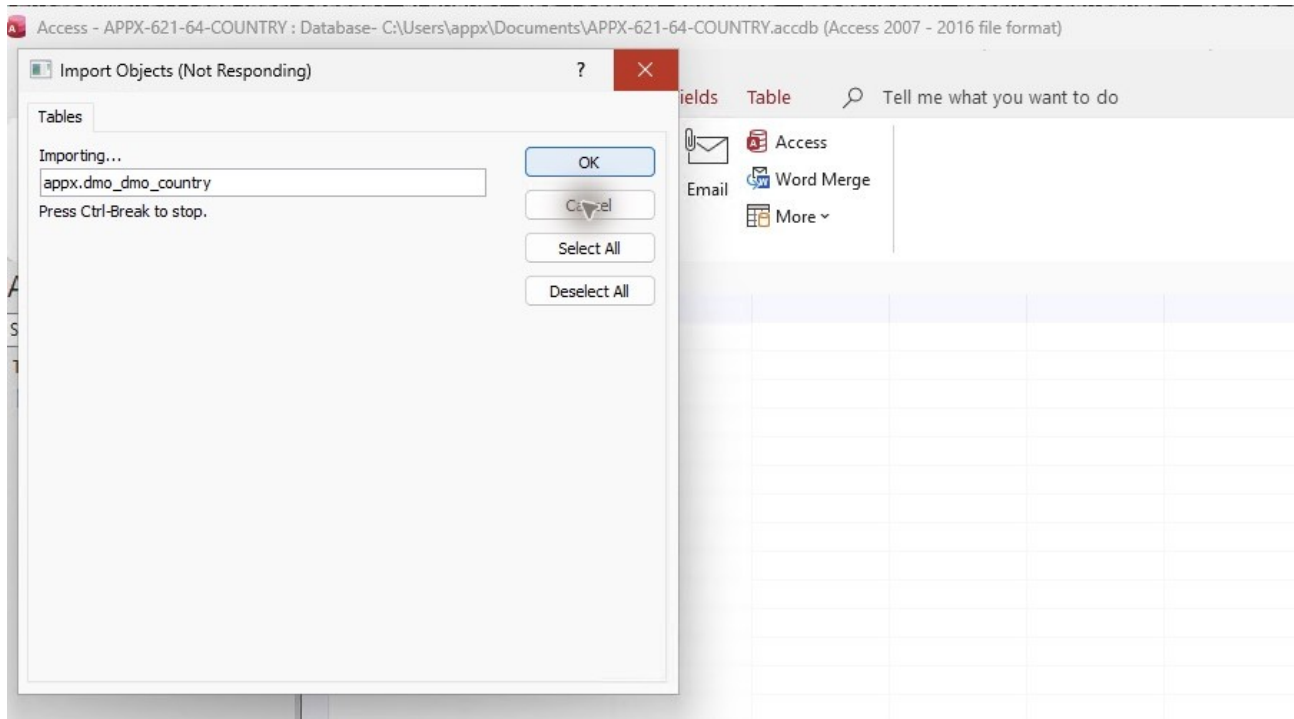


Figure 7. The fresh native Access import remains at Importing and displays Not Responding.

No repeat-to-fix workaround or component cause is established. Earlier Access linking and pass-through-count observations are different routes and do not pass this fresh import or full-value acceptance.

Choose 64-bit LibreOffice Calc and Base

Use 64-bit LibreOffice with the supplied 64-bit driver. Check Help, About LibreOffice and close other LibreOffice processes before selecting an architecture. The native tests use 26.2.6.3. An earlier headless document-API run used 26.2.3.2; its result is identified separately from native screenshots.

Connect LibreOffice Base and Save a COUNTRY Query

The supplied Base document with an explicit direct SQL query reopened in native LibreOffice 26.2.6.3, requested a new password and refreshed **eight records and nine fields**, with the same Unicode text failure. Its registration was checked against the actual ODB in `C:\ODBC-Documentation\cookbooks\6.2.1-64`. A separate fresh File, New, Database wizard with ODBC/AppxODBC, user appx, Password required, registration and Open for editing saved its file but stalled in a blank window for about three minutes. Requesting closure produced Not Responding. Preserve these as separate results; opening the supplied ODB does not establish that the fresh wizard route succeeds.

For the tested explicit query, use this SQL without a trailing semicolon:

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

COUNTRY ordered - APPX-621-64-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
C		P	0	2	A	JPN	03/01/01	SPF	05/31/01
F		C	0	1	S			JPN	03/01/01
I		C	0	0		JPN	03/07/01	JPN	03/07/01
N			0	2	B			JPN	03/02/01
R		C	0	1	A	ADM	08/08/23	ADM	08/08/23
U		C	1	2	S			LDN	03/08/01
U		S	0	1	A			SPF	05/29/01
U		S	0	1	A	SPF	05/29/01	SPF	05/29/01

Figure 8. Native Base direct-query result, with all nine COUNTRY fields and truncated Unicode text.

Use **Edit, Run SQL command directly** in the query editor. Save the connection and query, keep Password required, and supply the authorized password when opening the query. Use Refresh in its result window and compare every field. A table count or a truncated name must not be accepted as complete data.

The separate [5.5.4 32-bit Base recipe](#) passes its complete sample on the specified older engine/gateway combination. It is not evidence that changing only this edition's client recovers text.

Load COUNTRY into LibreOffice Calc and Refresh It

The native Calc import, save, full exit, fresh reopen, password prompt and refresh completed in 64-bit LibreOffice 26.2.6.3. The saved workbook contains eight COUNTRY rows and nine fields in a refreshable database range. Its 72 values were independently compared with the native APPX records: 22 text values differ, exactly matching the separate ODBC result. Refreshing the workbook does not repair these values. The [5.5.4 32-bit Calc recipe](#) passes its full sample on its specified engine/gateway/driver combination; this is a different stack.

1. Register the supplied [APPX-621-64-COUNTRY.odb](#) as **APPX-621-64-COUNTRY**, using [the registration procedure below](#). Keep its saved query **COUNTRY ordered** and Password required.
2. Create a blank Calc workbook. Choose **View, Data Sources** and expand **APPX-621-64-COUNTRY, Queries, COUNTRY ordered**. Supply the authorized password when requested. Check the returned rows before importing them.
3. Keep **A1** as the sheet insertion cell. Select all query records using the upper-left corner of the query grid, then click **Data to Text** in the data-source toolbar. The imported range is **A1:I9**, including its heading row. Copy and paste alone does not establish this refreshable range.

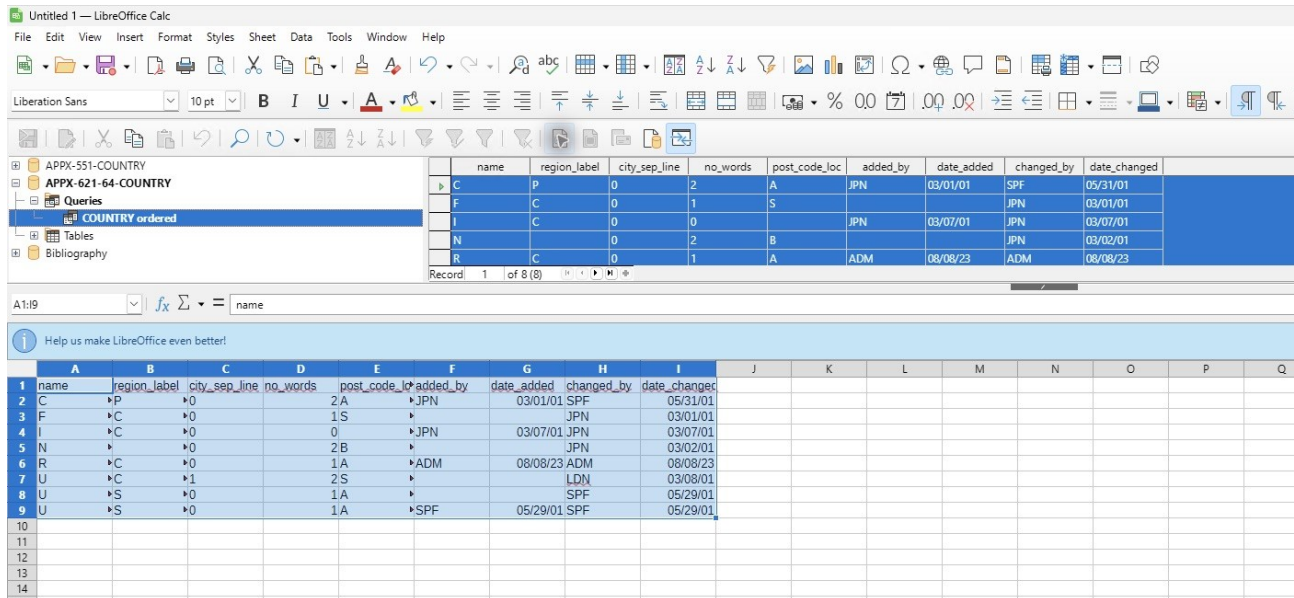


Figure 9. The native Calc Data Sources browser displays the registered COUNTRY query and the Data to Text import.

- Hide the browser with **View, Data Sources**. Select columns A through I and choose **Format, Columns, Optimal Width** if needed. Select a cell within the imported range, then choose **Data, Refresh Range**.
- Save as an **ODF Spreadsheet (.ods)**. Keep **Save with password** unchecked; that option encrypts a document and is separate from the connection's password prompt. Use **File, Exit LibreOffice** to close all LibreOffice windows before testing a fresh connection.
- Open the ODS in **LibreOffice Calc**. If Windows associates ODS files with Excel, use **Open with, LibreOffice**. Select a cell in the imported range and choose **Data, Refresh Range**. Enter the password in the new authentication prompt.

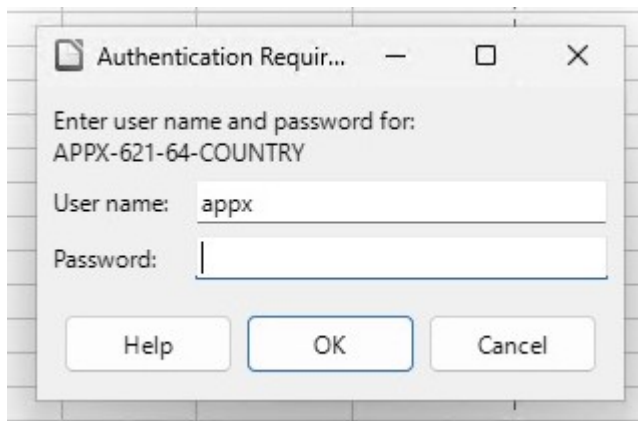


Figure 10. A fresh Calc process requests the connection password when refreshing the saved database range.

- Check all eight rows and nine fields, then save the refreshed workbook. The actual result below retains truncated Unicode text, including three separate names displayed as U. Do not identify rows by those damaged name values.

	A	B	C	D	E	F	G	H	I	J
1	name	region_label	city_sep_line	no_words	post_code_loc	added_by	date_added	changed_by	date_changed	
2	C	P	0	2	A	JPN	03/01/01	SPF	05/31/01	
3	F	C	0	1	S			JPN	03/01/01	
4	I	C	0	0		JPN	03/07/01	JPN	03/07/01	
5	N		0	2	B			JPN	03/02/01	
6	R	C	0	1	A	ADM	08/08/23	ADM	08/08/23	
7	U	C	1	2	S			LDN	03/08/01	
8	U	S	0	1	A			SPF	05/29/01	
9	U	S	0	1	A	SPF	05/29/01	SPF	05/29/01	
10										

Figure 11. The reopened Calc workbook refreshed all eight COUNTRY rows; Unicode text remains damaged.

The supplied [APPX-621-64-COUNTRY.ods](#) preserves this observed result and the **COUNTRY** ordered query reference. Its saved XML and the ODB contain no saved credential attributes. The native fresh-password test confirms that refreshing needs a live connection and an entered password. Compare against [COUNTRY-expected.csv](#) for native values and [COUNTRY-observed.csv](#) for the measured ODBC result. The earlier document-API attempt that produced an empty range is not used as acceptance or a product diagnosis.

Help LibreOffice Find a Saved Base Connection

In Tools, Options, LibreOffice Base, Databases, register the saved ODB under the exact name used by the workbook's database range. Select its actual new file path. Registration belongs to the LibreOffice user profile; moving an ODB does not update an old registration automatically.

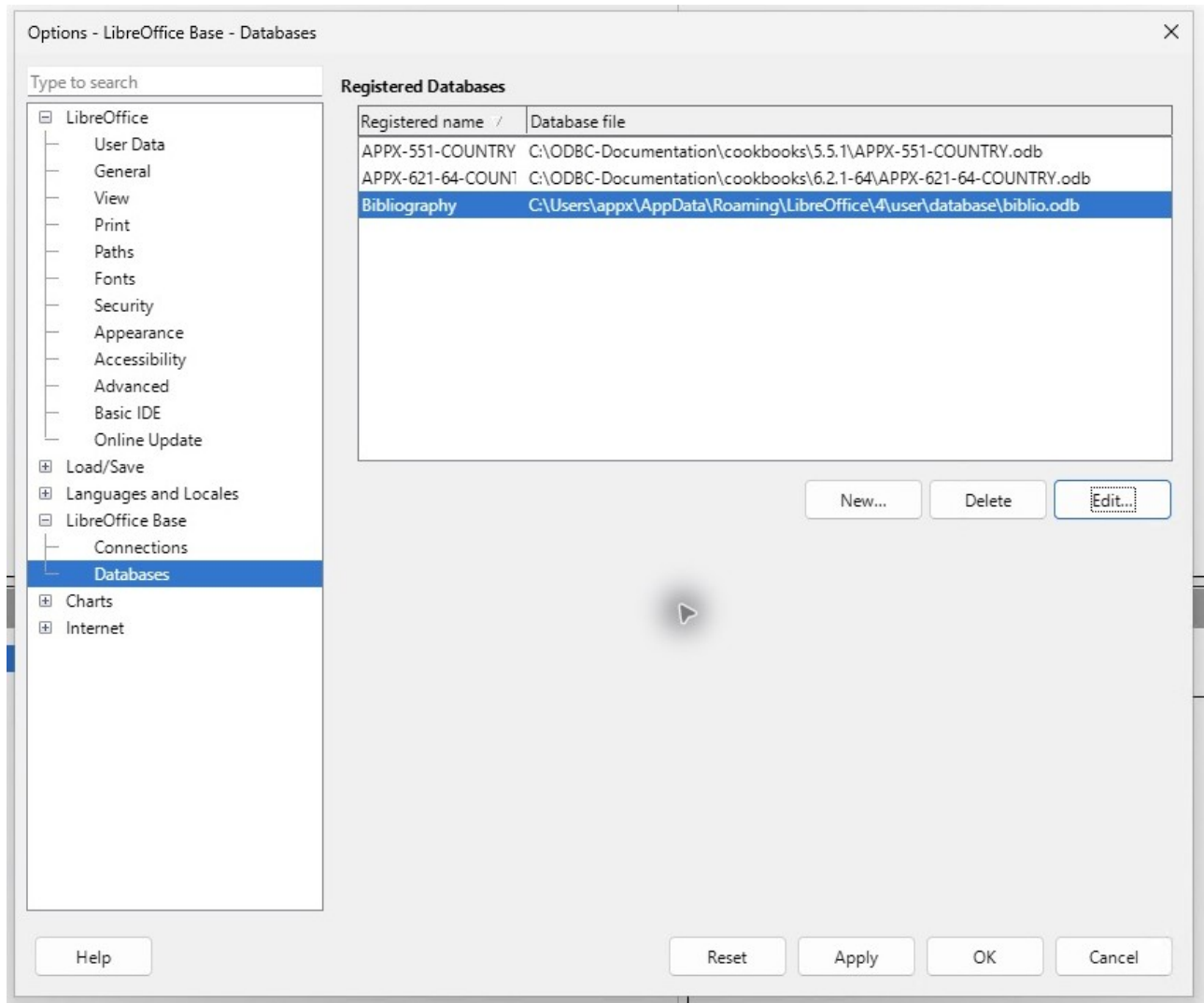


Figure 12. LibreOffice registration points APPX-621-64-COUNTRY at the actual saved ODB.

Open the ODB query first to check its connection. Then open the workbook, select a cell in its imported range and choose **Data, Refresh Range**. An ordinary pasted snapshot is not a refreshable database range.

CHAPTER 6

Create Reports with Crystal Reports 2025

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

Choose a Compatible Crystal Reports Version

Crystal Reports 2016 is a 32-bit client and cannot use this edition's 64-bit driver. Use the matching 32-bit cookbook when choosing that client. Keep the report client's architecture separate from the Windows and APPX server architectures.

Use **SAP Crystal Reports 2025 Patch 1, version 14.4.0.5335, 64-bit**, with the supplied **Dharma 14.01 x64** driver. This test used the matching 6.2.1 engine and gateway, **System DSN AppxODBC**, dictionary **appx**, blank Service, gateway TCP 1992 and **Socket** backend TCP 31062. The installed executable is `win64_x64\crw64.exe`; its parent directory's **Program Files (x86)** name does not make this executable 32-bit.

The direct **ODBC (RDO) SQL command** creates and refreshes an eight-record report, but its Unicode-storage text is incorrect. Both native PDF exports contain 22 differing values among the 32 projected values. For the nearest verified complete four-field report, see [5.5.4 with Crystal Reports 2016 and Dharma 14.01 x86](#). That uses a different engine, gateway and client architecture; it is not a driver-only repair for this edition.

Create a COUNTRY Report

1. Start Crystal Reports 2025 and select **File, New, Standard Report**, or **Report wizard** on its start page.
2. Expand **Create New Connection, ODBC (RDO)**. Select **AppxODBC** and select **Next**. Enter the authorized APPX user ID and password, then select **Finish**. If the Dharma login appears, verify Host **localhost**, Database **appx**, the correct user ID and blank **Service** before submitting. A blank user ID is not an accepted authenticated test.
3. Under the new connection, double-click **Add Command**. Enter the **four-column 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** appears under **Selected Tables**, then select **Next**.

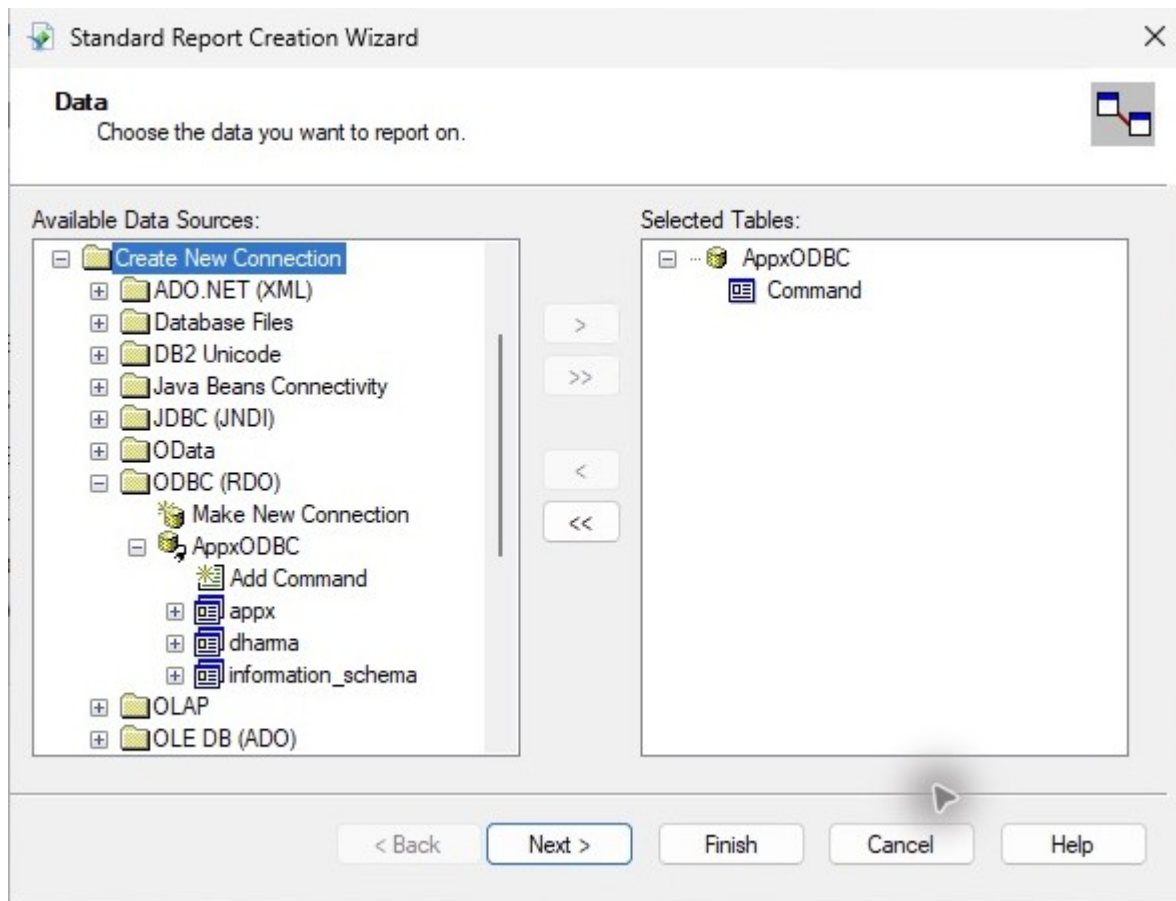


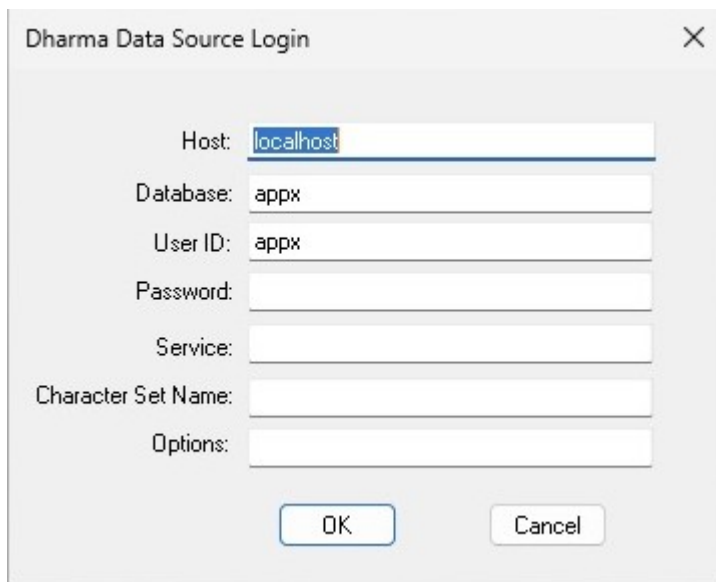
Figure 13. The current 64-bit Crystal connection selects its direct ODBC command.

5. Add all four fields to **Fields to Display** in this order: **name**, **region_label**, **no_words**, **post_code_loc**. Select **Finish** without adding grouping or selection filters.
6. Check **Records: 8** and compare complete values with the [native expected sample](#). In this test, names are C, F, I, N, R, U, U, U instead of complete country names. Region labels and nonblank post-code locations also retain only their first character. All eight numeric **no_words** values agree; the two expected blank text values remain blank.

10/15/2026				
<u>name</u>	<u>region label</u>	<u>no words</u>	<u>post code</u>	<u>loc</u>
C	P	2.00	A	
F	C	1.00	S	
I	C	0.00		
N		2.00	B	
R	C	1.00	A	
U	C	2.00	S	
U	S	1.00	A	
U	S	1.00	A	

Figure 14. Crystal retrieves eight rows, with visibly truncated text on the tested 6.2.1 stack.

7. Select **File, Save As** and save **APPX-621-64-COUNTRY-RDO.rpt** locally. Select **File, Export, Export Report**, choose **PDF, Disk file** and **All pages**, then save the exported PDF. Exporting preserves the damaged result; it does not restore native text.
8. Close Crystal completely. Reopen the saved report in a fresh process and press **F5**. Confirm **Refresh Report Data**. Enter the APPX password in the masked Dharma field, with **Service** blank. The saved report requested a new password in this test.



The image shows a 'Dharma Data Source Login' dialog box with the following fields and values:

Field	Value
Host:	localhost
Database:	appx
User ID:	appx
Password:	
Service:	
Character Set Name:	
Options:	

At the bottom are 'OK' and 'Cancel' buttons.

Figure 15. The reopened report requests the APPX password for a fresh refresh.

9. Wait for **Records: 8** and a new refresh timestamp, save the report again and export a second PDF. Compare the refreshed values, not only the initially displayed saved snapshot.

Both PDFs reproduce the independent ODBC projection: eight rows, 32 values and 22 native-value differences. The final two projected rows are identical and cannot identify their source records uniquely. Creation, save and fresh-process authentication/refresh completed; text-value acceptance failed. Widen automatic headings before distributing a report. The test RPT and PDFs are not finished-layout offline examples.

CHAPTER 7

Read APPX Data with 64-bit PowerShell

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

Run a Query with a Secure Login Prompt

The matching supplied 64-bit driver returns eight COUNTRY rows. Numeric/date/RAW values agree with native data; Unicode fields and text predicates do not. Read the [complete comparison](#).

Intermittent waits during connection Open remain separate from execution results.

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

```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

Read the complete example below and paste its entire & { ... } block into that 64-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-64.ps1](#) and save it in a working folder. If your site's policy permits the approved script, run:

```
.\query-appx-country-64.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 32-bit PowerShell process and shows the path to the required 64-bit executable. The observed result begins COUNTRY count: 8; country names are C, F, I, N, R, U, U, U. This output demonstrates the data limitation and is not a successful text comparison.

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 (-not [Environment]::Is64BitProcess) {
    throw 'Open 64-bit Windows PowerShell: C:
\Windows\System32\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 '64-bit' | Select-Object Name, Platform
Get-OdbcDsn -Name AppxODBC -DsnType System -Platform '64-bit' | Select-Object Name,
DriverName, Platform, DsnType
Test-NetConnection localhost -Port 1990 -InformationLevel Quiet

```

The first two commands should show the **64-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 6.2.1 query was verified from a fresh process.

Work with Query Results in PowerShell

Download [read-appx-odbc-64.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 **64-bit Windows PowerShell 5.1** with the site's approved script-execution method. Keep execution policy unchanged.

Open the [64-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-64.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 8 objects, including damaged Unicode text. 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-64.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-64.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
```

The native expected result is one Canada record, Province and 2. The current 64-bit stack instead returns zero records for this parameter. Do not accept this text predicate for business use. 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-64.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 native expected names are **Netherlands** and **UK**. The observed result instead contains **C, N and U**, incorrectly retaining Canada. Numeric-only PROSPECT key filtering separately returned the correct key and employee count, with its company name still truncated. 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 -NoTypeInfo -Encoding UTF8 -NoClobber
$csv = @(Import-Csv -LiteralPath .\COUNTRY.csv)
$csv.Count
$csv | Format-Table -AutoSize
```

Expect 8 data rows; exported text remains damaged. 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 this eight-record sample, `no_words` sums to 10, with minimum 0 and maximum 2. Observed region keys are C (4), P (1), S (2) and blank (1), reflecting damaged text. The numeric-only local filter selects the three records whose names display C, N and U; consult the native expected CSV for their full names. 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-64.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-64.ps1 -Credential $credential -ConnectionInfo | Format-List
$tables = @(.\read-appx-odbc-64.ps1 -Credential $credential -Tables)
```

```
$tables | Where-Object { $_.TABLE_NAME -match 'COUNTRY' } |
    Select-Object TABLE_SCHEM, TABLE_NAME, TABLE_TYPE
$columns = @(.\read-appx-odbc-64.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 earlier [5.5.1 metadata checks](#) used a different supplied driver generation and are not acceptance evidence for this stack. 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-64.ps1 -Credential $credential `
    -Sql 'SELECT * FROM DMO_DMO_COUNTRY ORDER BY name' -CommandTimeout 30 -MaxRows 100
    -Verbose)
```

The helper buffers rows before returning them and raises an error when MaxRows is exceeded. The corrected fresh-process check returned the expected MaxRows=3 error with **zero partial objects**. 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 6.2.1 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:31062</code>	Selects the APPX backend host and login-manager port.	On the gateway , in <code>C:\Program Files\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.

The separate **APPX_SQL_CMD** and **APPX_DBG_CODE** settings log APPX access to an external relational database. They are not the APPX ODBC gateway trace described above and do not change its Unicode field mapping. APPX documents APPX_SQL_CMD as a writable log filename and APPX_DBG_CODE as a category mask; unset the overrides after a focused reproduction. See [the APPX environment-variable reference](#). The complete sample comparison was repeated in fresh processes after removing these two inherited overrides and returned the same damaged values.

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\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-621-31062-0SAuth\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 [Socket](#) backend. The Pipe helper log applies only when a separately configured instance actually uses Pipe; it does not diagnose this selected Socket path.

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>System32\odbcad32.exe</code> for these 64-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\AppxODBCserver\appx.dbs\appx.env"
```

The relevant INI entries are:

```
[Environment]
TPEROOT="C:\Program Files\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:31062
```

This is the APPX backend [endpoint](#), separate from the gateway's measured TCP 1992. 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 64-bit package is installed and that the [DSN](#) points to its actual `dhsodbc.dll`.

The supplied 64-bit Dharma 14.01 [DSN](#)-creation action completed. The older 32-bit CTL3D32 workaround is not a dependency finding for this driver; inspect the actual loader error before applying any repair. 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. The existing configured 6.2.1 installation provided the structures used by the successful ASQL imports. This test does not claim a fresh first-start structure-generation sequence.

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
<code>-20003 Syntax error</code>	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.

Error	What it means	What to do
-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 64 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".
-30008 Packet Size Mismatch	Seen on this edition after a statement with a quoted name longer than 32 characters. The connection then fails every statement.	Close and reopen the connection, and keep quoted names to 32 characters.

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](#).