



APPX LIBRARY

APPX ODBC 5.5.4 Cookbook (32-bit)

APPX 5.5.4, 32-bit clients

Updated October 20, 2026: SQL names, reserved words and views checked.
Verified October 14, 2026 on Windows 11 Pro 25H2, build 26200.6584 (64-bit), with 32-bit clients. Uses the APPX 5.5.4 engine, gateway and matching ASQL with the updated Dharma 14.01 client driver from APPX's official 6.2.1 package.

OFFLINE FILES

Using this PDF offline

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

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

This PDF/A-3u edition embeds the checked examples below. Save them through the PDF viewer's Attachments panel, or extract `appx-odbc-554-offline.zip`. The scripts-only archive is `appx-odbc-554-scripts.zip`. Read `README-offline.txt` for the required 32-bit clients and Base registration; verify extracted files with `SHA256SUMS.txt`.

The Base connection/query requires live APPX access. The Calc workbook contains a sample snapshot and refreshes through that connection. Neither saves a password. Attachments do not run automatically. PDF download links lead [here](#); external vendor links still require Internet access. If a browser hides attachments, download the PDF and use a viewer with an Attachments panel.

Embedded file and SHA-256

APPX-554-COUNTRY.odb
040510efe1a00369191f81516e4938fabee1f4bef8069c6b19afa63bae96340f

APPX-554-COUNTRY.ods
e6ade081e9d25d6f64101f7b0ec34eddf6a8f2ef9d99e96287167286939bebea

COUNTRY-columns.csv
3170dbb1d5cc7c5e9ebcd15eab3bc95332f5c031c4839c416407abc44d69d117

COUNTRY-expected.csv
3ebd56a425a722096e90724381919f2a7b535dac8ae17cd19102a64834896790

COUNTRY-report.sql
e3a19509c7c7f9e7e2873e6836c6bfb08cffe7e703c0a974a5a4e5f9f68e2718

COUNTRY.sql
9480b8ec2461d006bc2ae6a1ab8398f26bac47cc9e4e29db0669041852dfe177

README-offline.txt
01b836cb266c2e8f0925de5b1e26058a45ede9165fe29f41e5e21e21b21d8f6a

SHA256SUMS.txt
fba11fb00ad71aed30203afb195299f8d2da34f8dac99baa64d82d1c4162b4c1

prepare-554-gateway-runtime.ps1
a22c35c75327c1084aef075c0ed0bcc1fcd247dff6d49eb1f33e84e0f2fe72c7

query-appx-country-32.ps1
aee11a07ef6a606f938ed5c1adcb40a99a9e04cbd738136f5ce7447d6d4231b6

Embedded file and SHA-256`read-appx-odbc-32.ps1``febe68779ff090f89ff683ee3b842d2d02d696c09c1a60bcff007d4b5c1440f6`

`setup-554-compatibility-driver.ps1``02d5e9998fa7bd4123a160b18c2dcab246c8418893e1cede5c657b7e9725fa95`

Contents

Connection glossary: Service, sqlnw, DSNs and ports	8
Chapter 1 Before You Start	6
What You Need and Where to Begin	6
Check Versions and Choose 32-bit Programs	6
What Works Better in Later Releases.....	8
Identify the Server, Ports and Connection Names	8
Connection Glossary: Service, sqlnw, DSNs and Ports	8
Decide What to Install on Each Computer.....	13
Download the Tested Example Files	13
Chapter 2 Install and Check APPX ODBC	14
Install the 5.5.4 Gateway and ASQL	14
Start the Gateway and Check Required Microsoft Files.....	18
Install the 6.2.1 Client Driver	20
Check the Saved AppxODBC Connection (DSN).....	21
Chapter 3 Make APPX Files Available to SQL	23
What ASQL Does and Where to Find It.....	23
Add the COUNTRY Table to SQL with ASQL	23
Check Every COUNTRY Record and Field.....	26
When to Create or Update SQL Table Definitions.....	27
How APPX Files and Fields Become SQL Tables and Columns	28
SQL Name Rules and Reserved Words.....	29
Review and Edit the SQL That ASQL Creates.....	32
Use a View for Friendly Names and Shorter Queries.....	34
Check Limits for Large Tables and Long SQL Statements.....	35
Chapter 4 Read APPX Data with WinSQL.....	38
Connect WinSQL and Read COUNTRY	38
Chapter 5 Read APPX Data with Excel, Access and LibreOffice.....	39
Choose 32-bit Excel and Access	39
Load COUNTRY into Excel and Refresh It.....	39
Import COUNTRY into Access	41
Choose 32-bit LibreOffice Calc and Base.....	43
Connect LibreOffice Base and Save a COUNTRY Query	44

Load COUNTRY into LibreOffice Calc and Refresh It	45
Help LibreOffice Find a Saved Base Connection.....	47
Chapter 6 Create Reports with Crystal Reports 2016	48
Install Crystal Reports 2016.....	48
Create a COUNTRY Report	48
Chapter 7 Read APPX Data with 32-bit PowerShell	50
Run a Query with a Secure Login Prompt.....	50
Check the Driver Architecture and Gateway Connection	52
Work with Query Results in PowerShell.....	52
Filter Records with SQL Parameters.....	53
Export COUNTRY to CSV and JSON.....	54
Summarize and Check the Returned Data.....	54
List Tables and Columns and Identify the Driver	55
Set Query Limits and Read Error Details	56
Chapter 8 Solve Connection and Import Problems.....	57
Set Connection and Diagnostic Options.....	57
Record a Diagnostic Log, Then Turn Logging Off	59
Check the Driver, Ports and Login Before Retrying	62
Check the dhsodbc.ini and appx.env Settings Files	62
SQLDriverConnect Failed: Check the Microsoft Runtime.....	63
Fix Driver Setup Error 126.....	64
Fix the ASQL System Directory Error	65
Fix Missing Files When the Gateway Will Not Start	68
Fix Import Errors Caused by Field Names.....	70

CHAPTER 1

Before You Start

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

What You Need and Where to Begin

Connect 32-bit Windows applications to APPX 5.5.4 using a single ODBC data source. The verified clients include LibreOffice [Base](#) and [Calc](#). Configure the gateway, import the DMO COUNTRY sample, then follow the recipe for [WinSQL](#), [Excel](#), [Access](#), [Crystal Reports](#) or [PowerShell](#).

Client driver for current Windows. This recipe keeps the APPX 5.5.4 engine, gateway and matching ASQL Dictionary Importer, and uses the updated 32-bit Dharma 14.01 client driver distributed in APPX's official 6.2.1 package. It preserves the default `appx` dictionary and `AppxODBC` data source. The [driver extraction steps](#) extract only the client driver and required runtime files; they do not replace ASQL or upgrade the APPX application server.

Check the tested versions and client bitness, record the server names and ports, and choose which components to install.

Requirements

- An installed and licensed APPX 5.5.4 system, an authorized APPX login, and its server name and login-manager port.
- The matching 5.5.4 ODBC gateway and ASQL packages, plus the [compatibility driver](#) and [gateway runtime files](#).
- Administrator access for installation and [System DSN](#) creation. Use an ordinary account for the client recipes.
- A 32-bit client: WinSQL32, Microsoft 365 Excel or Access, Crystal Reports 2016, LibreOffice Calc or Base, or Windows PowerShell.
- The DMO database and DMO application with the COUNTRY file, for the sample recipes.

Check Versions and Choose 32-bit Programs

Component	Tested version	Bitness
Windows	Windows 11 Pro, version 25H2, OS build 26200.6584	64-bit
APPX engine	5.5.4	32-bit
APPX ODBC gateway	5.5.4	32-bit
Client ODBC driver	Dharma 14.01 from the APPX 6.2.1 package	32-bit
ASQL Dictionary Importer	From the APPX 5.5.4 package	32-bit

Component	Tested version	Bitness
WinSQL, Excel, Access, Crystal Reports and PowerShell	Releases specified in their recipes	32-bit
LibreOffice Calc and Base	26.2.6.3, after an update from the starting 25.8.7 x86 package	32-bit

Choose the driver and ODBC Administrator that match the **client application's bitness**. The engine and gateway are separate components connected over the network. This cookbook verifies the combination above; a different APPX release or bitness needs its corresponding installation and client recipe.

The tested configuration uses **Windows 11 Pro, version 25H2, OS build 26200.6584 (64-bit)**, verified on October 14, 2026, with additional PowerShell and LibreOffice checks on October 15. The clients are WinSQL 15, Microsoft 365 version 16.0.20430.20182, and Crystal Reports 2016 SP9 Patch 21. Every ODBC client and its driver in this recipe must be **32-bit**, even though Windows is 64-bit.

Matching and mixed versions

The 5.5.4 download includes the legacy Dharma 9.01 driver. For the Windows 11 Pro 25H2 configuration documented here, use the [updated Dharma 14.01 driver](#): catalog access, the native ASQL import and the full-value COUNTRY query were verified with that driver. WinSQL, Excel, Access, Crystal Reports 2016 and the PowerShell example also passed the client checks described in this edition. The legacy driver returned a catalog-query error in this environment. Earlier Windows installations may have different compatibility requirements; this test does not identify a particular Windows update as the cause or establish a Windows release cutoff.

The verified combination here is **5.5.4 engine + 5.5.4 gateway + ASQL from the 5.5.4 package + Dharma 14.01 client driver from the 6.2.1 package**. Extracting that driver does not install the 6.2.1 gateway or change the APPX engine. This specific compatibility repair passes the COUNTRY import and full-value query comparison.

Use matching APPX engine and gateway releases as the starting point. ASQL is a separate client: the reviewed 5.x and 6.x sources both call the gateway's `bld_ddl` and `get_ddl` functions, so compatible interfaces make some ASQL mixing plausible. The sources also differ in script-fetch handling. A newer ASQL is therefore a candidate to test, not an automatic substitute, and it does not upgrade the gateway's field conversion or the APPX engine.

Before approving another combination, verify the ASQL and driver bitness, [DSN](#) discovery, file listing, import, generated definitions and complete field values. Record the exact package versions and hashes. This edition does not claim acceptance of a newer ASQL program against the 5.5.4 server. Later editions need their own compatibility results and limits.

The [5.5.2 cookbook](#) documents a different accepted combination: matching 5.5.2 engine/gateway/ASQL with the supplied Dharma 9.01 x86 driver. Its [Access imports fail](#), while this edition's complete configuration passes Access. Its reported table/name limits are also smaller. These differences require the documented component combinations; the tests do not establish a driver-only upgrade for 5.5.2.

What Works Better in Later Releases

One later configuration has a measured capacity improvement over this edition, shown below. No client compatibility improvement or removed limitation has been verified. Retain [this edition's SQL and size limits](#) and [diagnostic settings](#) for its matching gateway and tested Dharma 14.01 x86 driver.

Apart from the row below, later 6.2.1 configurations are not presented as an upgrade remedy. A verified later result belongs here with its exact engine, gateway, ASQL, driver and client combination and a direct procedure link.

Capability or capacity	This edition	Verified later configuration and procedure
Longest SQL name	32 characters, although the driver reports 64.	64 characters with the 6.2.1 64-bit engine, gateway and supplied driver and 64-bit clients, measured 2026-10-20. See 6.2.1 64-bit SQL name rules . The 6.2.1 32-bit configuration has not been measured.

Identify the Server, Ports and Connection Names

Setting	Value in this cookbook	Value to use
APPX login-manager endpoint	<code>localhost:31055</code>	The APPX server and its configured login-manager port
APPX Windows interface	<code>Socket</code>	The <code>Socket</code> configuration tested here
ODBC gateway endpoint	<code>localhost:1992</code>	The gateway server and its verified listening port
Gateway SQL dictionary	<code>appx</code>	The default dictionary created by the gateway installer
Windows System DSN	<code>AppxODBC</code>	The same DSN in every client recipe
APPX Database ID / Application ID	<code>DMO / DMO</code>	The installed DMO sample IDs
Imported SQL table	<code>DMO_DMO_COUNTRY</code>	The COUNTRY table created by ASQL

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

Connection Glossary: Service, sqlnw, DSNs and Ports

Start here when a setup page uses an unfamiliar name. These definitions apply throughout this 5.5.4 32-bit cookbook. The port values below belong to this edition's tested configuration; use the site's actual server names and verified listeners.

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

Service field

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

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

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

This cookbook's current connection recipe uses a **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 **Dharma 14.01 x86 from APPX 6.2.1** 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 legacy gateway install script still contains `sqlnw 1990/tcp`, while this edition's tested gateway listens on **1992**. The cookbook therefore deliberately corrects the mapping to **1992**; its compatibility-driver helper leaves Service **blank**. The mapping correction is a documented configuration step, not an untouched gateway-installer default. Blank Service then uses the corrected `sqlnw 1992/tcp` mapping. Check the actual listener after installation.

Choosing a custom service name

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

Changing a mapping affects new connections that use that alias on this computer, including blank-Service connections that default to `sqlnw`. Changing one DSN to a distinct custom alias affects clients using that DSN. Preserve the original file and DSN before changing either. Close and reopen the client, then verify the destination port and the sample query. Keep the field as a service name in these recipes; do not substitute a numeric port without separately checking that driver's handling of numeric Service values.

Check the Service setting on this computer

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

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

An empty Service output cell means the DSN has no nonblank override. For the standard setup the displayed alias should be **sqlnw**, and its active line should select **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:31055**, 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-554-31055-OSAuth**. Its name identifies the managed login-manager process, while **31055** 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 5.5.4	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 Dharma 14.01 x86 from APPX 6.2.1 client when <code>sqlnw</code> has no mapping	A fallback is driver-specific; it does not override a conflicting named mapping.
TCP 31055	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 31055 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

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

Desktop users do **not** re-import shared definitions on every computer.

Download the Tested Example Files

Download the [offline examples](#), [Base connection/query](#), [Calc workbook](#), [expected records](#) and [read-only SQL](#). Read [README-offline.txt](#) before moving the connection to another profile.

CHAPTER 2

Install and Check APPX ODBC

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

Install the 5.5.4 Gateway and ASQL

1. Confirm that you can log into APPX 5.5.4 and open the DMO application. Record the APPX server name and login-manager port. This example uses `localhost:31055` and the Windows login manager's **Socket** interface.
2. Download the [5.5.4 gateway installer](#) and the [5.5.4 driver and ASQL installer](#).
3. Open Command Prompt in the download folder and verify each download with `certutil.exe -hashfile "filename.exe" SHA256`.

Download	SHA-256
<code>AppxODBCserver_windows.exe</code>	<code>2ee701c37b684a44ba42d359cb66ee18342f4e3679091319141c55aa26d59c7e</code>
<code>AppxODBCdriver+ASQL_windows.exe</code>	<code>d0fd5e9766f315ca1993534c9186ad953d022fb25b2ff21c63075846eb90c328</code>

4. Right-click the gateway installer and choose **Run as administrator**. Select **Next** on the 5.5.4 welcome page and keep the default installation directory, `C:\Program Files (x86)\AppxODBCserver`.

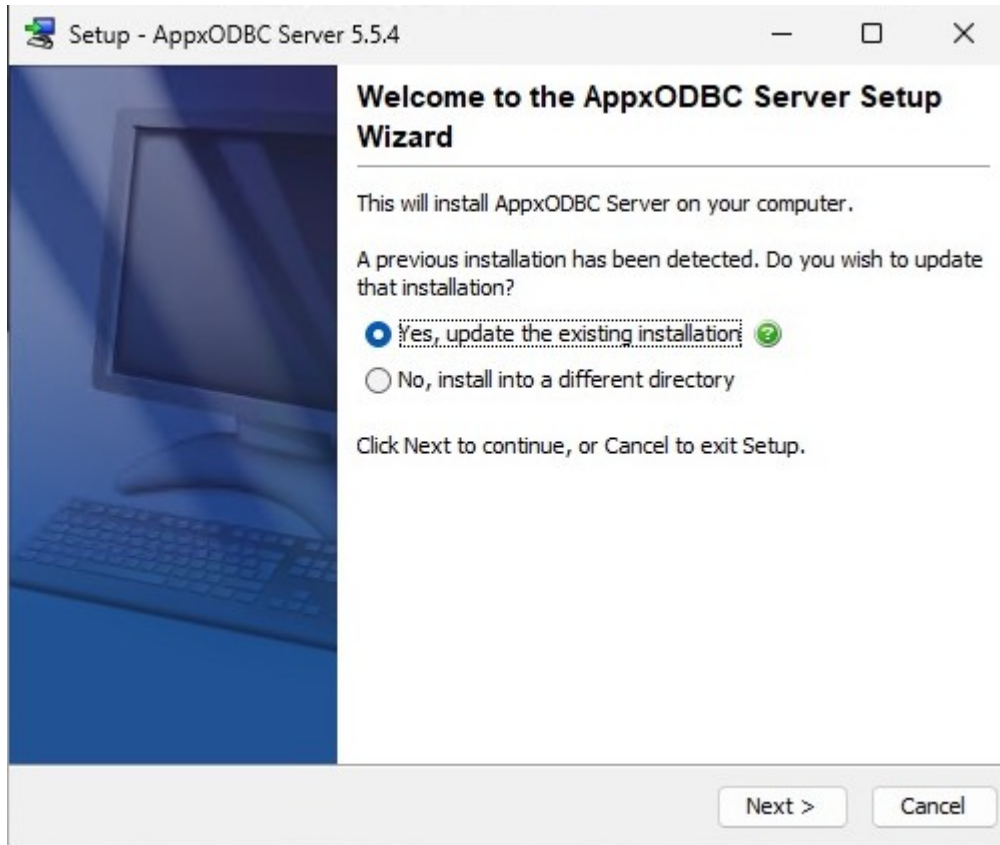


Figure 1. The gateway installer identifies APPX ODBC Server 5.5.4.

5. Enter the APPX server's hostname and login-manager port. For this example, enter `localhost` and `31055`, then complete the wizard.

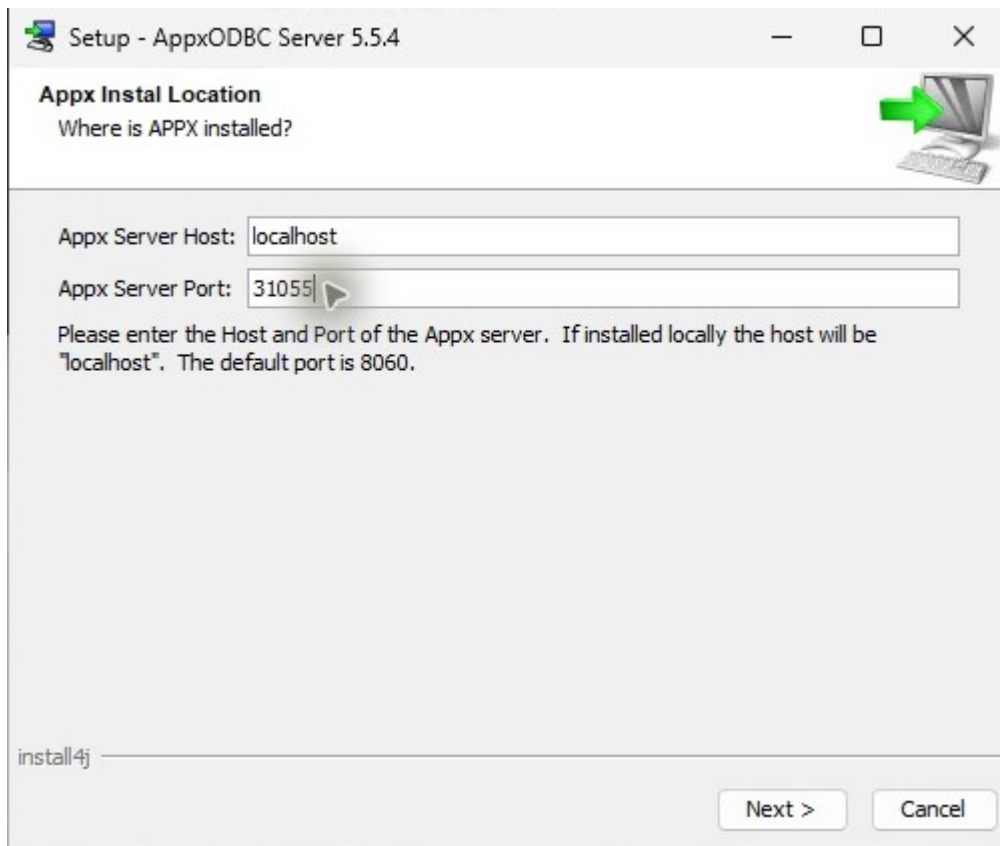


Figure 2. The gateway's backend is the APPX login manager, localhost:31055 in this example.

6. On the designated ASQL administration computer, install the 5.5.4 driver and ASQL package as administrator, keeping `C:\Program Files (x86)\AppxODBCDriver`. Keep this version of **ASQL Dictionary Importer** for **Import COUNTRY with ASQL**. Follow **Install the 6.2.1 Client Driver** to extract the 6.2.1 client driver and update the default AppxODBC DSN.



Figure 3. The ASQL package comes from the matching 5.5.4 release.

7. Install the [Microsoft x86 runtime](#), then start and check the gateway. Apply additional package repairs only if that check fails. If optional [DSN](#) creation displays the following error, follow [Resolve System DSN Error 126](#) before reopening the setup dialog:

Error Executing Action

```
CONFIGSYSDSN: Unable to create a data source for the
'AppxODBC Driver' driver:
    The setup routines for the AppxODBC Driver ODBC
    driver could not be loaded due to system error code 126: The
    specified module could not be found.
    (C:\PROGRA~2\APPXOD~2\dhsetup.dll). with error code
    -2147467259
```

The [client-driver recipe](#) extracts selected DLLs from the 6.2.1 package without running its installer. It keeps the 5.5.4 gateway, matching ASQL and existing [appx](#) dictionary. Test any changes on a copy of an existing production installation first.

Start the Gateway and Check Required Microsoft Files

Install Microsoft's 32-bit Visual C++ runtime first

Ensure that the [Microsoft Visual C++ v14 Redistributable \(x86\)](#) is installed on the gateway computer and on each Windows computer running the 32-bit ODBC driver. An existing current installation does not need to be installed again. Choose **x86**, including on 64-bit Windows: the runtime architecture must match the application. See [Microsoft's runtime download guidance](#).

Why this runtime: the reviewed 5.5.3 and 5.5.4 Windows build projects select the Visual Studio 2019 v142 toolset. ASQL's Release configuration uses the dynamic C/C++ runtime (`MultiThreadedDLL`, or `/MD`), and both downloaded gateway executables import `VCRUNTIME140.dll`. This identifies the **Microsoft v14 runtime family**. Use the official **x86** download linked above for these 32-bit components. Visual Studio or the Boost development tools do not need to be installed on user desktops.

The source projects do not pin an exact compiler patch or redistributable installer version, so they do not establish the oldest acceptable runtime patch. Microsoft's current v14 redistributable supplies compatible runtime updates. Version 14.51.36247.0 was installed and tested for this cookbook; it is a tested version, not a claimed minimum.

1. Save the Microsoft download as `vc_redist.x86.exe`.
2. Open its **Properties > Digital Signatures** and confirm the signer is **Microsoft Corporation**.
3. Run the installer, choose **Install** or **Repair**, and restart Windows if requested.
4. [Start and check the gateway service](#). Complete the [client driver setup](#) and [COUNTRY check](#); if both succeed, continue without adding other runtime files.

For a scripted installation from an administrator Command Prompt, run:

```
vc_redist.x86.exe /install /quiet /norestart
```

Exit code 0 indicates success; 3010 indicates success with a restart required. Complete that restart before treating the installation as finished.

If the gateway still does not start

Additional runtime files are a [conditional package repair](#), not a general requirement to install the Boost development tools. Prefer a complete matching gateway package or dependency bundle supplied by APPX Support.

The particular 5.5.3 and 5.5.4 gateway downloads tested on October 14, 2026 explicitly reference ICU 59 and `boost_regex-vc142-mt-x32-1_71.dll`, while their installer archives contain ICU 51 and omit that Boost DLL. Installing Microsoft's signed x86 runtime 14.51.36247.0 did not resolve that separate dependency: both gateways failed to start with only the Boost DLL removed and started when it was restored. This finding applies to those downloaded builds; it does not establish that every installation of these APPX releases needs the same repair.

A missing dependency can appear in Windows **Event Viewer > Windows Logs > System** as:

The Appx ODBC Data Gateway Server for **Windows service** failed to start due to the following error:
The service did not respond to the start or control request in a timely fashion.

Event 7009 may also report:

A timeout was reached (30000 milliseconds) while waiting for the Appx ODBC Data Gateway Server for **Windows service** to connect.

These messages alone do not identify the missing DLL. Check the actual gateway package and its dependencies before applying the **conditional gateway package repair**. Microsoft Visual C++ runtime files, ICU and Boost Regex are separate components; one installer does not substitute for all three.

Start and check the gateway

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

Open **Command Prompt as administrator**. Run these commands one at a time:

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

Check that `BINARY_PATH_NAME` points to

`C:\Program Files (x86)\AppxODBCserver\bin\AppxODBC.exe`. The expected service state is **RUNNING**. Match its PID to the **LISTENING** entry on **TCP 1992**. That port is separate from the APPX login-manager port, 31055 in this example.

If an earlier installation left the service pointing to the wrong folder, stop it and correct only that service's path:

```
sc.exe stop "Appx ODBC Data Gateway Server for Windows"
sc.exe config "Appx ODBC Data Gateway Server for Windows" binPath= "\"C:\Program Files (x86)\AppxODBCserver\bin\AppxODBC.exe\"""
sc.exe start "Appx ODBC Data Gateway Server for Windows"
```

Install the 6.2.1 Client Driver

This step installs the **32-bit Dharma 14.01 client driver from APPX 6.2.1** for use with APPX 5.5.4. The helper extracts six DLLs from the download, registers the driver and updates the default AppxODBC DSN. It leaves the existing appx dictionary and matching ASQL program in place.

1. On the Windows client, download the [official APPX 6.2.1 32-bit driver and ASQL package](#). Save it as AppxODBCdriver-ASQL-621-x86.exe.

The verified package SHA-256 is:

```
0453e2a800c1da5132c0f6feaab55619c941d97b62d68e9724dd0c3bba86a23d
```

Check the download from Command Prompt:

```
certutil.exe -hashfile AppxODBCdriver-ASQL-621-x86.exe SHA256
```

2. Download [setup-554-compatibility-driver.ps1](#) into the same folder and read it before running it.
3. Open **Command Prompt as administrator** in that folder. Run the helper to replace the existing data source under its default name:

```
powershell.exe -NoProfile -File .\setup-554-compatibility-driver.ps1 -Installer .  
\AppxODBCdriver-ASQL-621-x86.exe -ReplaceDsn
```

For a separate client computer, append `-GatewayHost gateway.example.com`, replacing the example with the gateway's actual hostname. The script verifies the official package and all six extracted DLL hashes, creates a separate driver directory, and registers a new driver alias and a **32-bit System DSN**. With `-ReplaceDsn`, it backs up the existing AppxODBC registration before replacing that data source. Existing driver files and driver registrations are preserved. Its default driver folder is `C:\APPX-ODBC-554-Client`.

ASQL 5.5.4 requires the data source's enumeration label to be exactly `AppxODBC Driver`. The script supplies that label while retaining the DSN's explicit path to the separate Dharma 14 driver. Keep the original 5.5.4 driver registration intact.

4. Check the script's displayed Driver, Host, Database and Service values. The Driver ends in `APPX-ODBC-554-Client\dhsodbc.dll`; Host names the gateway; Database is `appx`; the optional **ODBC Service** is **blank**. The helper preserves that empty setting; the tested driver uses its default `sqlnw` TCP-port lookup. **Service and the blank default**.

Next, [Check the Default AppxODBC Data Source](#).

Check the Saved AppxODBC Connection (DSN)

Use the default dictionary

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

The default names refer to different things:

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

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

Check the APPX login-manager interface

This edition was verified with **Socket**. An existing login manager explicitly configured for **Pipe** should be reviewed by the APPX administrator before using this recipe. The **APPX login-manager Windows service** is named `Appx-554-31055-0SAuth`; use your actual service name and installation path.

Close active APPX and ODBC sessions before restarting the login manager. Back up that **APPX login-manager Windows service** configuration, then run these commands separately in an administrator Command Prompt if a change to **Socket** is needed:

```
reg.exe export "HKLM\SYSTEM\CurrentControlSet\Services\Appx-554-31055-0SAuth" "C:\ODBC-
Logs\appx-554-login-manager.reg"
"C:\appx\Appx.554\services\appxLoginMgr.exe" -modify -name=Appx-554-31055-0SAuth
-AppxInterface=Socket
sc.exe stop "Appx-554-31055-0SAuth"
sc.exe start "Appx-554-31055-0SAuth"
```

Create the backup folder first if it does not exist. Wait for **STOPPED** before starting that **APPX login-manager Windows service** again, then confirm it is **RUNNING** and listening on the intended APPX port. **Socket** is the login manager's built-in default; its configuration can show `#AppxInterface=Socket` with no explicit `AppxInterface` override.

Check the client driver and DSN

Leaving the **ODBC Service** field blank is valid for the standard connection. Blank uses the default `sqlnw` TCP-name lookup; entering `sqlnw` explicitly uses that same lookup. Leave Service **empty** for this recipe, as in the current login screenshots. The supplied driver installer omits this optional value;

entering `sqlnw` is an explicit alternative. Keep Database **appx**: it selects the SQL dictionary and does not belong in Service. With either blank or explicit `sqlnw`, verify that the client's mapping reaches this gateway on TCP 1992. Read [Service, its blank default and custom aliases](#) before changing the field.

Complete [Install the 6.2.1 Client Driver](#) first. It creates or updates AppxODBC while preserving the default dictionary. Then check the gateway port mapping and the data source below.

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

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

Resolve duplicate or conflicting `sqlnw` entries before proceeding. Leave the **ODBC Service** field **blank** for this recipe. Entering the TCP alias `sqlnw` explicitly is also valid. Both use this mapping to reach TCP 1992; neither starts the gateway or changes its listener. Use a different named alias only for a deliberately configured alternate port. [How the services file and fallback work](#).

2. Open `C:\Windows\SysWow64\odbcad32.exe` and check **System DSN** for AppxODBC. This is the 32-bit ODBC Administrator. The program in System32 manages 64-bit drivers.

Next, [import the COUNTRY metadata with ASQL](#). An empty dictionary cannot answer the sample queries until that import completes.

CHAPTER 3

Make APPX Files Available to SQL

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

What ASQL Does and Where to Find It

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

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

For this cookbook, keep ASQL on one administrator-maintained Windows computer. The gateway server is a convenient choice when interactive administration is allowed; an authorized administration workstation can also connect remotely. ASQL does not need to remain running for Excel or other clients to query APPX. Routine user desktops need only the driver; the [compatibility-driver helper](#) extracts the driver DLLs without installing the newer ASQL program. APPX's [ODBC installation instructions](#) describe this separation between driver-only desktops and the administration installation.

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

Add the COUNTRY Table to SQL with ASQL

1. Open **ASQL Dictionary Importer** from the installed 5.5.4 package.
2. Choose **AppxODBC**, enter an authorized APPX user and password, and enter **DMO** for both Database ID and Application ID.
3. Select **OK**. The DMO dictionary tree appears.



Figure 4. ASQL uses the default AppxODBC DSN and the DMO sample IDs.

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

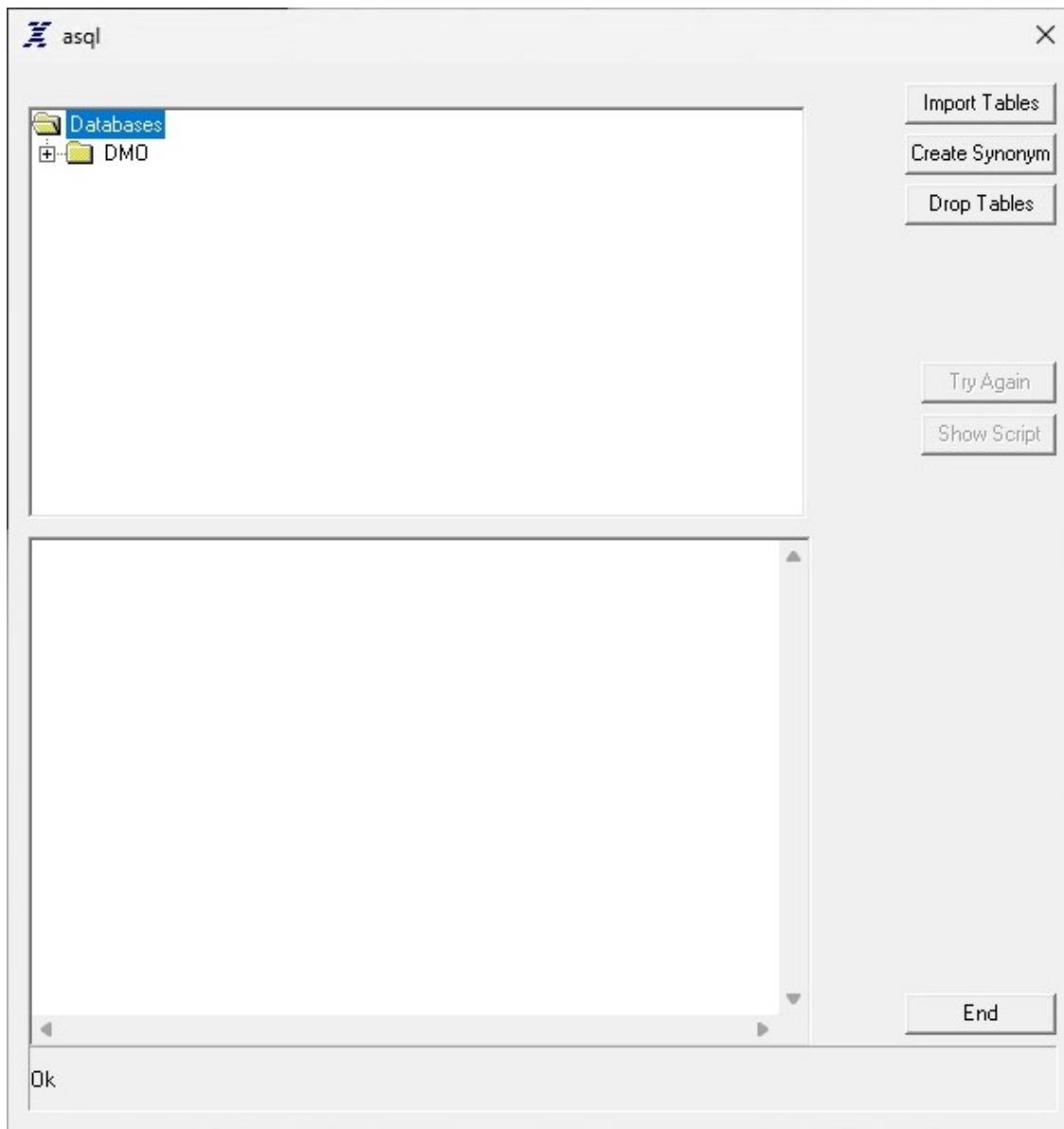


Figure 5. ASQL connects to DMO through the default dictionary.

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

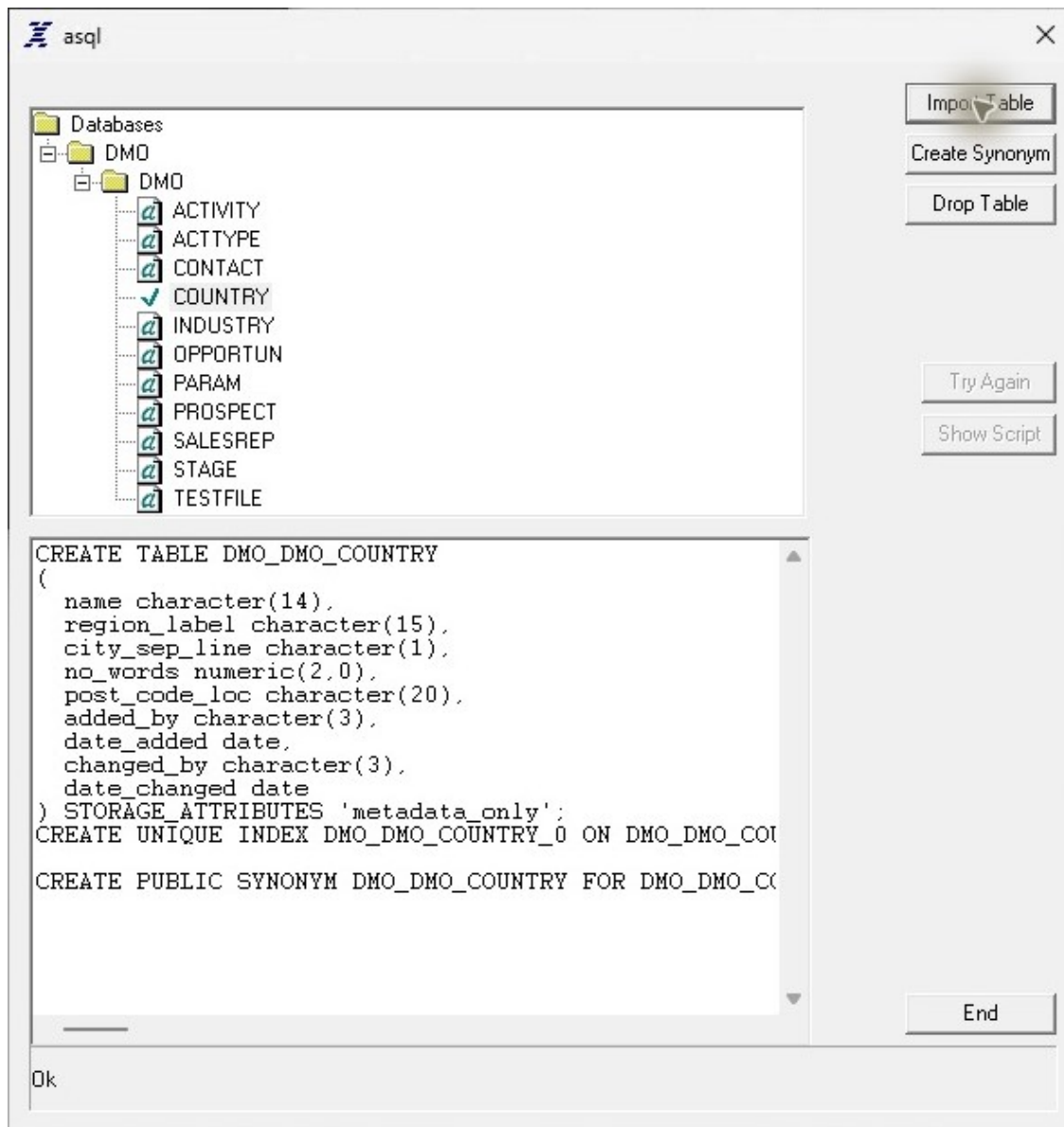


Figure 6. The COUNTRY import finishes with a check mark and Ok.

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

Check Every COUNTRY Record and Field

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

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

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

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

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

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

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

When to Create or Update SQL Table Definitions

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

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

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

Do not assume that every ASQL release automatically replaces an existing table. The 5.5.4 source executes generated `CREATE` statements and stops at a failing statement. If the existing definition causes **Table/View/Synonym already exists**, have the administrator perform a controlled metadata replacement, then import and verify the complete rows again. Avoid using **Drop Tables** on a parent folder. The [historical import manual](#) explains the need to re-import after structure changes; its automatic-replacement description should not be assumed for this package.

How APPX Files and Fields Become SQL Tables and Columns

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

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

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

How field names become column names

- Letters become lower case. Spaces, hyphens and any other character that isn't a letter, digit or underscore become underscores. REGION LABEL becomes `region_label`, and CITY-STATE becomes `city_state`.
- Many applications start every field name with the file's name. ASQL drops that part, so in a file named CUSTOMER a field named CUSTOMER NAME becomes `name`. It keeps the whole name when what's left would be one of the gateway's keywords or would start with a digit, so CUSTOMER NO stays `customer_no`, because NO is a keyword.

- ASQL doesn't check the whole name against the reserved words, and it doesn't change a name that starts with a digit. Either one stops the import. See [SQL Name Rules and Reserved Words](#), then [rename the column in the script](#).
- Two different field names can end up the same, for example CITY-STATE and CITY STATE. The import then stops with `-20010 Duplicate column specification`. Rename one of them in the script.

Group, substring and synonym fields don't become columns

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

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

Data types

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

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

SQL Name Rules and Reserved Words

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

Plain names:

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

This edition's driver reports 64 (`SQL_MAX_IDENTIFIER_LEN`), but the gateway refuses a longer name with `-20248 Identifier too long.`, quoted or not. See [What Works Better in Later Releases](#) for a later configuration with longer names.

These names were tried on this edition's gateway:

Name	Result
<code>name_2, _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 33 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
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

Starts with	Words
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
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 (x86)\AppxODBCDriver\asql.exe"
```

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

```
set "ASQL_EXPERT_MODE="
```

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

What each button does with the script

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

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

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

Rename a column before it is created

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

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

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

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

Why a change can seem to disappear

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

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

Use a View for Friendly Names and Shorter Queries

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

Create the view once, for example in WinSQL:

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

Then read it like a table:

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

The second query returns France, Ireland and UK.

A few things to know:

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

Check Limits for Large Tables and Long SQL Statements

Script length, table width and the editor's input limit are separate constraints. With this cookbook's Dharma 14.01 driver and 5.5.4 gateway, `SQLGetInfo` returned:

Property	Reported value
Driver version (<code>SQL_DRIVER_VER</code>)	14.01.00.0000
DBMS version reported by ODBC (<code>SQL_DBMS_VER</code>)	14.01.00.0000
Identifier quote character (<code>SQL_IDENTIFIER_QUOTE_CHAR</code>)	"
Column name length (characters) (<code>SQL_MAX_COLUMN_NAME_LEN</code>)	64
Columns in SELECT (<code>SQL_MAX_COLUMNS_IN_SELECT</code>)	0
Columns in one table (<code>SQL_MAX_COLUMNS_IN_TABLE</code>)	2000
SQL statement length (characters) (<code>SQL_MAX_STATEMENT_LEN</code>)	35000
Identifier length (characters) (<code>SQL_MAX_IDENTIFIER_LEN</code>)	64
Table name length (characters) (<code>SQL_MAX_TABLE_NAME_LEN</code>)	64
Schema name length (characters) (<code>SQL_MAX_SCHEMA_NAME_LEN</code>)	64
Catalog name length (characters) (<code>SQL_MAX_CATALOG_NAME_LEN</code>)	0
Columns in GROUP BY (<code>SQL_MAX_COLUMNS_IN_GROUP_BY</code>)	0
Columns in one index (<code>SQL_MAX_COLUMNS_IN_INDEX</code>)	100
Columns in ORDER BY (<code>SQL_MAX_COLUMNS_IN_ORDER_BY</code>)	0
Index size (bytes) (<code>SQL_MAX_INDEX_SIZE</code>)	0
Row size (bytes) (<code>SQL_MAX_ROW_SIZE</code>)	0
Reported row size includes long values (<code>SQL_MAX_ROW_SIZE_INCLUDES_LONG</code>)	N
Tables in SELECT (<code>SQL_MAX_TABLES_IN_SELECT</code>)	250
Character literal length (characters) (<code>SQL_MAX_CHAR_LITERAL_LEN</code>)	2000

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

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

These values belong to this tested combination. Do not apply the older reference's 500-column or 32-character limits, or these newer values, indiscriminately to another APPX/ODBC edition.

What happens near the length limit

The 5.5.4 ASQL source retrieves generated SQL in chunks and executes its semicolon-separated statements individually. Its **Try Again** buffer is allocated from the editor's text length; the old fixed 4 KB retry buffer was corrected in 2017. A complete script can therefore contain several statements even when no individual statement may exceed the driver limit.

The ordinary Windows edit control also has a default **32,767-character text-entry limit**. That affects typing or pasting; it does not establish the maximum size of an automatic import. Windows permits larger text loaded programmatically. See [Microsoft's edit-control limit documentation](#).

The advertised 35,000-character SQL limit is not a reliable target for this connection. Fresh-connection, read-only probes passed at 32,768 characters but failed at 33,000 through 35,000 with:

```
ERROR [k0k14] [Dharma][ODBC Dharma Driver 14.01.00.0000][Dharma/SDK] -30034
```

At 35,001 characters, the driver returned:

```
ERROR [HY090] [Dharma][ODBC Dharma Driver 14.01.00.0000] -19 Invalid string or buffer
length
```

The first diagnostic did not supply additional message text. These probes establish observed failures, not an exact maximum for every generated `CREATE TABLE`. Keep individual statements comfortably below 32 KB and inspect the actual error if a wide-file import fails. A failure partway through a multi-statement import can leave some metadata created and other definitions unfinished.

Shorten SQL without changing its meaning

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

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

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

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

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

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

CHAPTER 4

Read APPX Data with WinSQL

Connect, run a query and check the returned records.

Connect WinSQL and Read COUNTRY

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

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

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

Figure 7. WinSQL32 returns all seven COUNTRY records through the default AppxODBC source against APPX 5.5.4.

CHAPTER 5

Read APPX Data with Excel, Access and LibreOffice

Follow the tested procedure for each desktop program.

Choose 32-bit Excel and Access

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

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

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

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

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

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

Load COUNTRY into Excel and Refresh It

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

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

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

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

Edition verification: the saved Legacy Microsoft Query workbook was reopened and refreshed through **AppxODBC** against the 5.5.4 gateway. The refreshed, saved result contains all seven records and all nine columns, matching the native APPX sample after date and text-padding normalization. No password is stored in the workbook connection. The initial wizard sequence is the shared client recipe; this edition's fresh check was the reopen-and-refresh route.

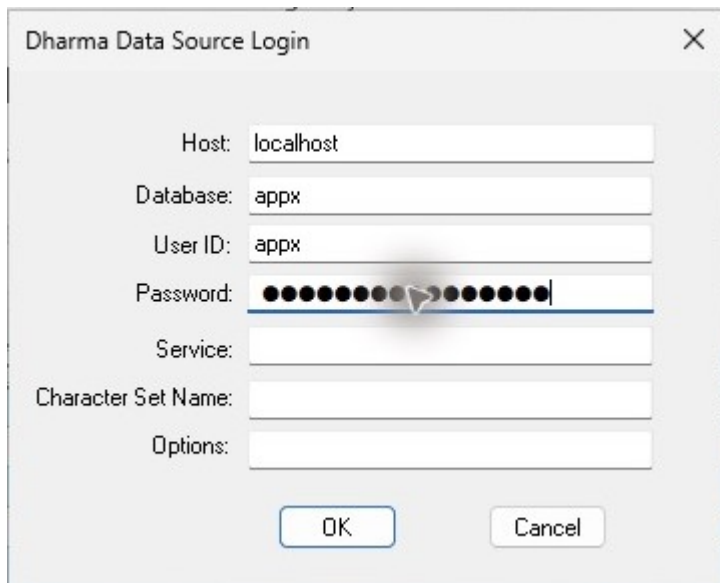


Figure 8. Excel authenticates to the default appx dictionary with the password masked.

Leave **Service** empty, as pictured. This native refresh passed with the tested Dharma 14.01 x86 compatibility driver; all seven records and nine fields match the sample. Blank selects the default `sqlnw` lookup, reaching this gateway on TCP 1992. **Database appx** selects the SQL dictionary. A saved workbook can retain `Service=sqlnw` even after the **DSN** is cleared: clear that field in its login dialog, finish the refresh, and save the workbook. **Service and saved-connection overrides**.

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

Figure 9. Excel's refreshed result contains all seven COUNTRY records.

Import COUNTRY into Access

This recipe creates a **local copy** in the Access database. Later APPX changes do not appear automatically in that copy.

1. Open 32-bit Access, select **Blank database**, enter a filename such as `APPX-554-COUNTRY.accdb`, and select **Create**.
2. Select **External Data** → **New Data Source** → **From Other Sources** → **ODBC Database**.
3. Choose **Import the source data into a new table in the current database** and select **OK**.
4. Select **Machine Data Source**, choose **AppxODBC**, and select **OK**.
5. Enter the APPX user and password in the Dharma login dialog, keeping **Host** as the gateway computer, **Database appx** as its SQL dictionary, and **Service empty** (blank uses the default `sqlnw` TCP-port lookup; [details](#)), then select **OK**.

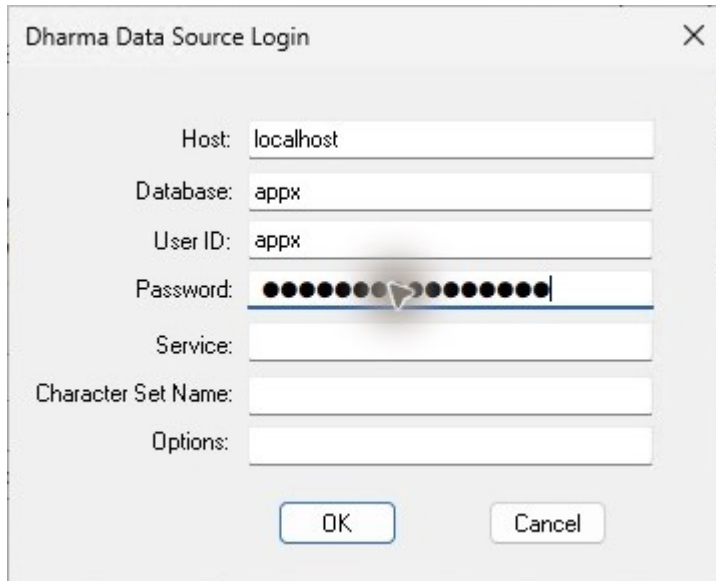


Figure 10. Access uses the default appx dictionary and a masked APPX password.

Leave **Service** empty, as pictured. The repeated native Access import passed with this edition's tested Dharma 14.01 x86 compatibility driver; all seven records and nine fields match the sample. Blank uses the default `sqlnw` lookup, reaching this gateway on TCP 1992. **Database appx** selects the SQL dictionary. [Meaning of these fields.](#)

6. In **Import Objects**, select one `appx.dmo_dmo_country` entry and select **OK**. The table and its synonym can appear as two entries with the same displayed name; select one.
7. Wait for **All objects were imported successfully**, then select **Close**.
8. Double-click `appx_dmo_dmo_country` in the Navigation Pane. It contains **7 records** and **9 columns**.

SECURITY WARNING Some active content has been disabled. Click for more details. [Enable Content](#)

All Access Obj... < appx_dmo_dmo_country1 X

name	region_label	city_sep_lin	no_words	post_code_l	added_by	date_added	changed_by	date_change
Canada	Province	0	2	AFTER REGION	JPN	3/1/2001	SPF	5/31/2001
France	County	0	1	SEPARATE LINE			JPN	3/1/2001
Ireland	County	0	0		JPN	3/7/2001	JPN	3/7/2001
Netherlands		0	2	BEFORE CITY			JPN	3/2/2001
UK	County	1	2	SEPARATE LINE			LDN	3/8/2001
USA	State	0	1	AFTER REGION	SPF	5/29/2001	SPF	5/29/2001
United States	State	0	1	AFTER REGION			SPF	5/29/2001

Figure 11. Access displays all seven records after the blank-Service import; a repeat copy receives the suffix 1.

Verified result: all seven saved records and all nine columns match the native APPX sample. Fixed-width text can retain trailing spaces, and Access can sort names differently from the gateway. Compare records by name after trimming trailing padding. A local copy is not a live link.

Choose 32-bit LibreOffice Calc and Base

The native Base wizard, table browser and saved query passed with this 5.5.4 configuration. Calc import, save, fresh-process reopen and authenticated Refresh Range also passed. All seven records and nine fields match the independent native sample.

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

The installation history and the tested version are different:

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

LibreOffice labels the **32-bit Windows architecture** deprecated. This is the vendor's support label for x86, not an APPX instruction to install an obsolete release. Check the [official LibreOffice download page](#) for an available 32-bit Windows build and its support status. If the current download selector no longer offers x86, an older vendor package may be available from the [official archive](#). An archived package is an installation source; its presence does not establish current maintenance or APPX compatibility.

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

Keep an existing 64-bit installation while testing x86

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

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

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

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

Close existing test instances before selecting another profile. If the updater restarts LibreOffice, check the running version and the [database registration](#) in the profile actually in use. The 5.5.1 test reproduced a missing registration after an updater restart selected another profile. Re-registering its saved .odb file restored Calc refresh. The separate 5.5.4 native profile retained its registration through the tested reopen.

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

Connect LibreOffice Base and Save a COUNTRY Query

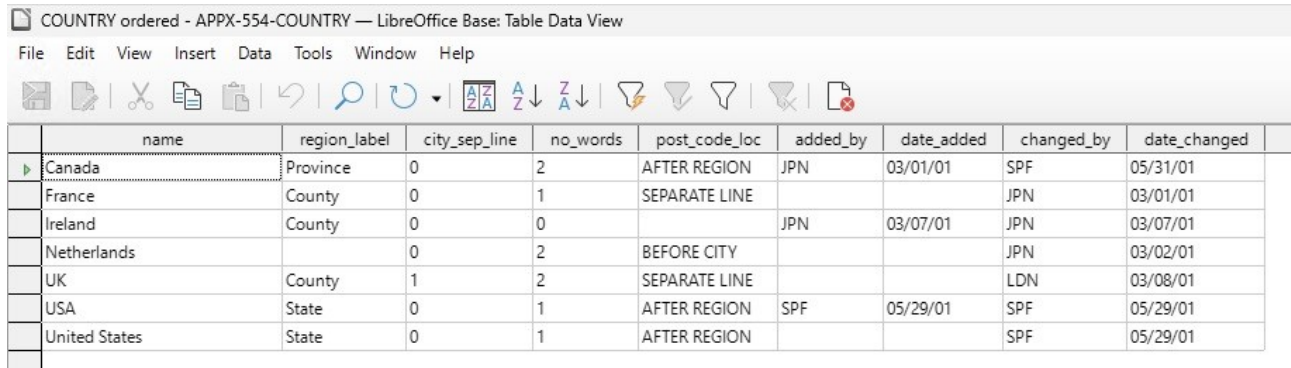
The Base document stores a connection and saved queries. It does **not** copy the APPX records into an embedded Base database. Use the existing shared SQL definitions from [ASQL](#).

1. Start the verified 32-bit **LibreOffice Base**. In Database Wizard, select **Connect to an existing database**, choose **ODBC**, and select **Next**.
2. On **Set up ODBC connection**, select **Browse**, choose **AppxODBC**, and select **OK**. If the [DSN](#) is missing, check [client bitness](#) before creating another one.
3. Select **Next**. Enter the authorized APPX user, select **Password required**, and select **Next**. Enter the password only when the connection prompts for it.
4. Keep **Yes, register the database for me** and **Open the database for editing** selected. Leave **Create tables using the table wizard** unchecked. Select **Finish** and save APPX-554-COUNTRY.odb in a local working folder.
5. Supply the APPX password in the **Authentication Required** dialog. Under **Tables**, expand **appx** and double-click **dmo_dmo_country**. The table view contains **7 records and 9 fields**. Widen columns to read the complete values; a clipped display is different from truncated returned data.
6. Close the table-data window. In the Base document, select **Queries → Create Query in SQL View**. Enter:

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

7. Select **Edit → Run Query**, or press **F5**. Check the seven records and all nine fields against the [native sample](#).
8. Select the query's **Save** button and name it **COUNTRY ordered**. Close the query-design window, then save the Base document itself.

9. Reopen the Base document in the same 32-bit LibreOffice profile, open the saved query, supply the APPX password when prompted, and run it again. The reopened saved query returned all seven complete records. A saved query is a definition; the gateway and APPX server must be available when it runs.



The screenshot shows the LibreOffice Base application window titled "COUNTRY ordered - APPX-554-COUNTRY — LibreOffice Base: Table Data View". The window has a menu bar (File, Edit, View, Insert, Data, Tools, Window, Help) and a toolbar with various icons. Below the toolbar is a table with the following data:

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

Figure 12. Base returns all seven COUNTRY records after reopening the saved query.

Use the exact qualified lowercase identifiers shown. In the 5.5.1 test, the unqualified uppercase spelling was rewritten by Base's SQL handling and returned -20005 Table/View/Synonym not found. The explicit "appx"."dmo_dmo_country" form passed. This local saved query is distinct from creating a shared SQL view in the gateway dictionary. See the [LibreOffice ODBC reference](#) for the connection wizard.

Load COUNTRY into LibreOffice Calc and Refresh It

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

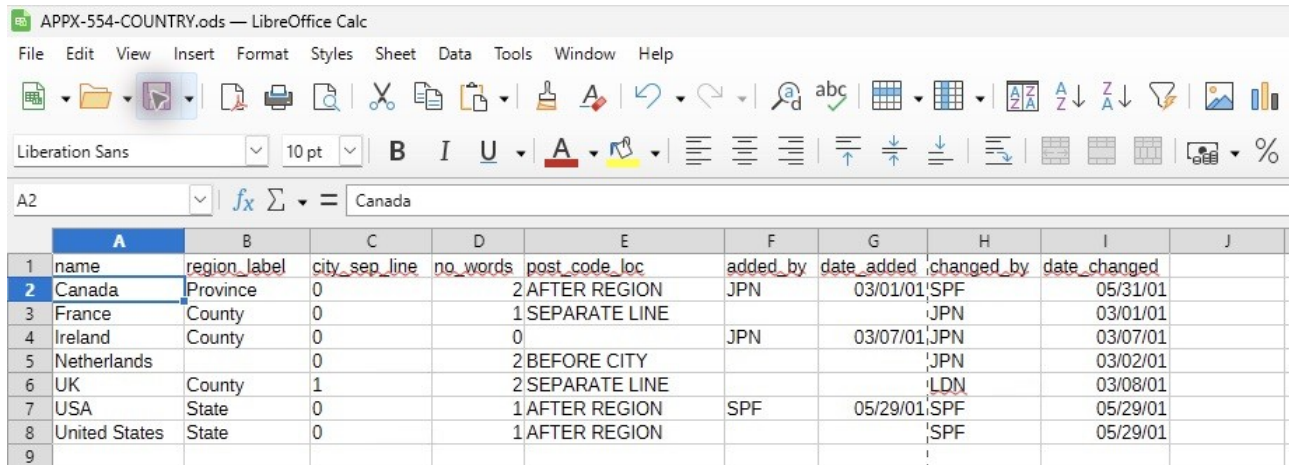
1. From Base, select **File** → **New** → **Spreadsheet**, or start the matching 32-bit Calc executable. Create a blank spreadsheet and select cell A1.
2. Select **View** → **Data Sources**. Expand APPX-554-COUNTRY → **Queries** and select **COUNTRY ordered**. Supply the APPX login if requested. The data browser displays the query's rows above the worksheet.
3. Leave worksheet cell A1 selected. In the data browser, click the small blank corner above the row selectors, to the left of the field headings, to select all seven records. Select **Data to Text** on the data-browser toolbar. This inserts the headings and records as a refreshable database range. Ordinary **Copy** → **Paste** produced static cells in this test; use Data to Text for the live range.
4. Check A1:I8: one heading row, 7 data rows, and 9 columns. Enter A:I in the Name Box to select the nine columns, then use **Format** → **Columns** → **Optimal Width** to fit the contents.

5. Select a cell within the imported range, then select **Data → Refresh Range**. Wait for completion and check the same values. This re-runs the source query; editing the worksheet is not a supported way to update APPX.
6. Save as **APPX-554-COUNTRY.ods**, retaining the ODF Spreadsheet format. Keep **APPX-554-COUNTRY.odb** at its registered location. The workbook's database range refers to the registered connection name and saved query.
7. Exit LibreOffice, reopen the ODS in the matching 32-bit installation and profile, select a cell inside the imported range, and use **Data → Refresh Range** again. Supply the APPX password when prompted, wait for completion, and save.

The saved 5.5.4 workbook passed import, refresh, save and fresh-process reopen/refresh. Its seven records and nine fields match the independently checked native sample. The ODS retains a database range at **Sheet1.A1:I8**, using registered source **APPX-554-COUNTRY** and query **COUNTRY ordered**. No password is saved.

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

Figure 13. Calc imports the saved Base query into a refreshable range.



	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	Canada	Province	0		2 AFTER REGION	JPN	03/01/01	SPF	05/31/01	
3	France	County	0		1 SEPARATE LINE			JPN	03/01/01	
4	Ireland	County	0		0	JPN	03/07/01	JPN	03/07/01	
5	Netherlands		0		2 BEFORE CITY			JPN	03/02/01	
6	UK	County	1		2 SEPARATE LINE			LDN	03/08/01	
7	USA	State	0		1 AFTER REGION	SPF	05/29/01	SPF	05/29/01	
8	United States	State	0		1 AFTER REGION			SPF	05/29/01	
9										

Figure 14. The saved Calc workbook refreshes after a fresh password prompt.

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

Help LibreOffice Find a Saved Base Connection

If Calc reports **Error accessing data source 'APPX-554-COUNTRY' with com.sun.star.container.NoSuchElementException**, check the LibreOffice registration before changing the APPX gateway or password. The 5.5.1 test reproduced that message after reopening the workbook in a different profile that did not contain the registration.

1. Confirm that **APPX-554-COUNTRY.odb** still exists and that the current LibreOffice process is 32-bit.
2. Open **Tools → Options → LibreOffice Base → Databases** in that same profile.
3. Select **New**. Set **Database file** to the existing .odb file, and **Registered name** to exactly **APPX-554-COUNTRY**, matching the workbook's saved source name.
4. Select **OK**, then **OK** in Options. If a registration already exists but its file moved, use **Edit** and correct its path instead of adding a conflicting second name.
5. Return to the workbook, select a cell in the database range, and use **Data → Refresh Range**. Supply the APPX login when requested.

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

CHAPTER 6

Create Reports with Crystal Reports 2016

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

Install Crystal Reports 2016

Use **SAP Crystal Reports 2016, 32-bit** for this driver. The tested package is 2016 SP9 Patch 21, with `crw32.exe` version 14.2.9.5104.

1. Obtain the installer and a trial or licensed product key through the [official SAP Crystal Reports page](#).
2. Run the installer as administrator, select the required language, check the prerequisites, and follow the license and product-key screens.
3. Choose **Typical**, complete installation, and select **Finish**.
4. Start **SAP Crystal Reports 2016**. The evaluation message confirms a **30-day trial**; select **OK** and dismiss the information notice.

Keep SAP credentials and product key private. Crystal Reports 2020 and 2025 use a different bitness and are outside this 32-bit recipe.

For an earlier gateway, use its own provider procedure: [5.5.2 uses an ODBC \(RDO\) SQL command](#) with the supplied Dharma 9.01 x86 driver, while [5.5.1 uses the Microsoft OLE DB-to-ODBC bridge](#). The 5.5.2 command test does not accept direct table selection. Keep each result with its matching engine/gateway/ASQL and tested client.

Create a COUNTRY Report

1. Select **Report wizard** on the Crystal Reports start page.
2. Expand **Create New Connection** → **ODBC (RDO)**, choose **AppxODBC**, and select **Next**.
3. Enter the APPX user and password and select **Finish**.
4. Expand **appx**, select **dmo_dmo_country**, and move it into **Selected Tables** with the right arrow.
5. Select **Next**, then add **name**, **region_label**, **no_words** and **post_code_loc** to **Fields to Display**.
6. Select **Finish**. The Preview displays **7 records** with the four selected fields.

name	region level	no words	post code loc
Canada	Province	2.00	AFTER REGION
France	County	1.00	SEPARATE LINE
Ireland	County	0.00	
Netherlands		2.00	BEFORE CITY
UK	County	2.00	SEPARATE LINE
USA	State	1.00	AFTER REGION
United States	State	1.00	AFTER REGION

Figure 15. Crystal Reports refreshed from the 5.5.4 gateway returns all seven sample records.

7. Select **File** → **Save As**, choose a local folder and save `APPX-ODBC-COUNTRY.rpt`.
8. To make a PDF, select **File** → **Export** → **Export Report**. Choose **PDF** (Adobe Acrobat) and **Disk file**, select **OK**, choose **All pages**, and save the PDF locally.

For an existing report, select **Report** → **Refresh Report Data** (F5), confirm the refresh, and supply the APPX login when prompted. Saved report data can be displayed without contacting APPX; a refresh is required to check the current connection.

The screenshot shows a 'Dharma Data Source Login' dialog box. It contains the following fields and values:

- Host: localhost
- Database: appx
- User ID: appx
- Password: [masked with 12 dots]
- Service: (empty)
- Character Set Name: (empty)
- Options: (empty)

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

Figure 16. The refresh uses the same appx dictionary and masked APPX login.

Leave **Service** empty, as pictured. Blank uses the default `sqlnw` lookup to reach this gateway on TCP 1992. **Database appx** selects the SQL dictionary; the APPX user and password authenticate the live refresh. [Meaning of these fields.](#)

Verified result: this edition reopened a version-specific copy of the report, refreshed its data from the 5.5.4 gateway, and saved the report and PDF export. All seven records and the four displayed fields match the COUNTRY sample. The report-wizard steps above use the same Crystal Reports 2016 procedure as the 5.5.3 edition.

CHAPTER 7

Read APPX Data with 32-bit PowerShell

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

Run a Query with a Secure Login Prompt

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

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

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

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

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

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

```
COUNTRY count: 7
```

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

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

```
& {
param(
    [string]$Dsn = 'AppxODBC',
    [System.Management.Automation.PSCredential]$Credential
)

$ErrorActionPreference = 'Stop'
```

```

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

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

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

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

```

```
COUNTRY count: 7
```

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

```
PS C:\Users\appx> |
```

Figure 17. The 32-bit PowerShell example returns seven COUNTRY records.

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 -Platform '32-bit' | Where-Object Name -like '*APPX*' | Select-Object Name, Platform
Get-OdbcDsn -Name AppxODBC -DsnType System -Platform '32-bit' | Select-Object Name, DriverName, Platform, DsnType
Test-NetConnection localhost -Port 1992 -InformationLevel Quiet
```

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

After changing an installed driver or **DSN**, close and restart clients, including PowerShell. An existing process can retain a previously loaded driver even though the DSN now points to a different file. The 5.5.4 query was verified from a fresh process.

Work with Query Results in PowerShell

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

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

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

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

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

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

Filter Records with SQL Parameters

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

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

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

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

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

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

Export COUNTRY to CSV and JSON

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

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

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

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

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

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

Summarize and Check the Returned Data

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

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

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

A second query can be compared with the earlier snapshot:

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

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

List Tables and Columns and Identify the Driver

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

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

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

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

The optional `-Columns` switch requests unrestricted ODBC column metadata. The focused `-Describe` query above is the verified column-description route for this edition. Unrestricted column discovery has not been accepted here; on a large dictionary it can also be expensive. Some legacy driver combinations treat missing, null and empty restriction arguments differently. `-CommandTimeout` applies to SQL commands, not `GetSchema` or connection opening.

Set Query Limits and Read Error Details

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

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

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

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

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

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

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

CHAPTER 8

Solve Connection and Import Problems

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

Set Connection and Diagnostic Options

Set a variable for the **process that reads it**. An environment setting in an Excel user's command window does not change an already-running gateway or an APPX engine started by the login manager. These scopes follow the reviewed 5.5.4 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:31055</code>	Selects the APPX backend host and login-manager port.	On the gateway , in <code>C:\Program Files (x86)\AppxODBCserver\appx.dbs\appx.env</code> . Use the actual APPX hostname for a remote server.
<code>APPX_ODBC_LOG=C:\ODBC-Logs\gateway.log</code>	Traces the APPX storage-manager part of the gateway, including file mapping and reads. Its default fallback is <code>C:\tmp\appx-odbc.log</code> if that path is writable.	In the gateway dictionary's <code>appx.env</code> , before opening a fresh connection. A new gateway task can overwrite the file; reproduce with one diagnostic connection.
<code>APPX_ODBC_CODE=-1</code>	Selects gateway trace categories; this is the source's default of all categories. <code>0</code> suppresses messages.	Alongside <code>APPX_ODBC_LOG</code> in the gateway dictionary's <code>appx.env</code> .
<code>APPX_NETIO_LOG=C:\ODBC-Logs\gateway-netio.log</code>	Traces the APPX/Net file-manager client used by the gateway to reach APPX.	In the gateway dictionary's <code>appx.env</code> for the gateway side of the backend connection.
<code>APPX_NETIO_CODE=-1</code>	Selects that file-manager client's logging categories.	Alongside <code>APPX_NETIO_LOG</code> in the gateway dictionary's <code>appx.env</code> .
<code>APPX_NET_LOG=C:\ODBC-Logs\network</code>	Traces APPX communication. The source adds a PID and role suffix to the filename.	In the gateway dictionary's <code>appx.env</code> for gateway traffic; in the login manager's engine Environment subkey for the APPX server side. Each side writes locally.
<code>APPX_NET_LOG_MASK=-1</code>	Selects communication trace categories.	In the same process environment as <code>APPX_NET_LOG</code> .

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

ASQL: one command window

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

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

After the reproduction, close ASQL and clear both overrides:

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

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-554-31055-0SAuth\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.

Diagnostic	Default and file behavior	Activation and category selection
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 is not part of that tested ODBC connection, so `APPX_COMM_LOG` is not a trace of its Socket data path. Keep the tested Socket configuration; use the engine/network logs for that route.

Use the complete Windows trace mask

The gateway, APPX/Net client and APPX network source parse their category masks with signed `strtoul`. 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 in the client	Check that both the client and driver are 32-bit and open <code>SysWOW64\odbcad32.exe</code> .
ASQL does not list the DSN	Use the setup script's <code>AppxODBC Driver</code> enumeration label and retain its separate Dharma 14 DLL path.
<code>-20038 Cursor not closed</code> or the first ASQL query fails	Check the actual Driver path. The supplied Dharma 9 driver fails here; use the 6.2.1 compatibility-driver setup .
Login returns <code>-1016</code>	Re-enter the APPX credentials and check the gateway Host, dictionary name, <code>APPX_DATA_SERVER endpoint</code> and the running gateway Windows service and APPX login-manager Windows service . The code alone does not identify the cause.
SQLDriverConnect Failed	Try installing or repairing the official Microsoft v14 x86 runtime, then retry. If it persists, collect the full message and check connectivity and credentials.
The gateway does not start	Install or repair the Microsoft x86 runtime first. Check the executable path and Windows event messages , then use the conditional package repair only if a dependency is still missing.
The connection waits or times out	Check the actual gateway listener and the client's sqlnw-to-port mapping . This recipe uses <code>Socket</code> for the 5.5.4 backend; keep the tested <code>Socket</code> setting.
The query returns Table/View/Synonym not found	Check the selected SQL dictionary and repeat the ASQL COUNTRY metadata import.
Two statements produce a syntax error	Run the count and full-row query separately.
Excel's Power Query route fails	Use the tested Legacy Microsoft Query recipe .
Access shows old records after APPX changes	The table is a local copy. Repeat the import into a new table to obtain a fresh copy.

Stop a client attempt that does not complete, retain the full error, and check a small count query before retrying. The PowerShell script limits each query to 30 seconds; that limit does not bound the driver's initial connection call.

Check the dhsodbc.ini and appx.env Settings Files

If the **DSN** looks right but the connection still fails, check these two text files **on the gateway computer**. The paths below are from this installation. A different installation folder will change the second path.

File	What to check
<code>C:\Windows\dhsodbc.ini</code>	The <code>[Environment]</code> section and the gateway installation directory in <code>TPEROOT</code> .

File	What to check
C:\Program Files (x86)\AppxODBCserver\appx.dbs\appx.env	The APPX server name and login-manager port in APPX_DATA_SERVER.

The second file belongs in the selected SQL dictionary's folder, **appx.dbs**, for this recipe. Do not assume that a file named `appx.env` in **bin** is the one being read.

Open Command Prompt and run these commands separately:

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

The relevant `dhsodbc.ini` entries in this installation are:

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

Check that `TPEROOT` points to the gateway's installation folder, **not bin**, and that the folder contains `appx.dbs`. Keep the SQL dictionary name **appx**; this is different from the APPX database ID **DMO**. If the **DSN** selects another SQL dictionary, check that dictionary's `.dbs` folder instead.

For this edition, `appx.dbs\appx.env` contains:

```
APPX_DATA_SERVER=localhost:31055
```

That address identifies the **APPX login manager**. It is separate from the ODBC gateway address and port used by the desktop **DSN**. This recipe uses gateway port **1992** and APPX port **31055**. On a gateway running on another computer, `localhost` means that gateway computer; replace it with the APPX server's actual name when needed.

The reviewed Windows gateway code reads `TPEROOT` from the INI file, appends the selected dictionary name plus `.dbs`, and reads `appx.env` there. Setting `TPEROOT` in a Command Prompt window does not correct that INI lookup.

Make a backup before editing either file. Use a text editor and keep the `.ini` and `.env` extensions; do not save an extra `.txt` extension. Close active ODBC clients before restarting the **gateway service**, then reopen the client and retry. Editing these connection settings does not itself require another ASQL import. If the change points to a different dictionary or APPX installation, check its imported definitions and sample query before using existing reports.

SQLDriverConnect Failed: Check the Microsoft Runtime

If ASQL or a 32-bit ODBC client reports:

SQLDriverConnect Failed

check the **Microsoft Visual C++ v14 Redistributable (x86)** as one possible cause. A Windows Server 2022 installation was reported to recover from this symptom after installing the 32-bit runtime. This is a reported troubleshooting remedy, not a guarantee that every connection failure has the same cause.

1. Confirm that the ODBC client and driver are both **32-bit** and that the intended **AppxODBC DSN** exists in the 32-bit ODBC Administrator.
2. Install or repair the **official Microsoft x86 Visual C++ runtime**. Use **x86** for these clients even when Windows is 64-bit. Microsoft's **runtime download guidance** identifies the supported package.
3. Restart Windows if requested, reopen the client, and retry the connection.
4. If the error remains, record the full ODBC message and SQLSTATE, then check the gateway listener, **DSN Host** (gateway computer), optional **ODBC Service** (gateway TCP-port alias), and APPX credentials. `SQLDriverConnect Failed` is a general connection-stage message; it does not by itself identify a missing runtime.

For other symptoms, see [error 126 while opening the legacy DSN setup dialog](#), [-20038 Cursor not closed](#), or [a gateway service that cannot start](#). The runtime installation is a potential remedy for the connection symptom above; it did not resolve those separately tested gateway and cursor failures.

Fix Driver Setup Error 126

If the 5.5.4 setup DLL cannot load CTL3D32, use the verified Microsoft file below. Close ODBC Administrator and ASQL first, then open elevated 64-bit Windows PowerShell:

```
$download = Join-Path $env:TEMP 'ctl3d32-microsoft.dll'
Invoke-WebRequest -Uri `
    'https://msdl.microsoft.com/download/symbols/ctl3d32.dll/3005CC7211000/ctl3d32.dll' `
    -OutFile $download
$expected = '6896C1F21CC9A5BC17E2E2FB645669AE708CB378C63E5EEF11B3E95527F3DA32'
if ((Get-FileHash $download -Algorithm SHA256).Hash -ne $expected) {
    throw 'Download checksum differs.'
}
if ((Get-AuthenticodeSignature $download).Status -ne 'Valid') {
    throw 'Windows has not verified this file signature.'
}
$target = Join-Path $env:WINDIR 'SysWOW64\CTL3D32.DLL'
if (Test-Path $target) {
    Copy-Item $target ($target + '.before-' + (Get-Date -Format 'yyyyMMdd-HH:mm:ss'))
}
Copy-Item $download $target
Get-AuthenticodeSignature $target | Format-List Status, SignatureType
```

The verified Microsoft file is version 2.31.000, 27,136 bytes, with a **Valid** Microsoft Windows catalog signature. Put it in **SysWOW64**. If an earlier attempt placed CTL3D32 beside `dhsetup.dll`, preserve that copy in a backup folder outside the driver directory; that location caused a system-directory warning in the native setup dialog.

Retry the original 5.5.4 installer setup dialog. Expected result: the missing-DLL error is resolved. Then apply the [6.2.1 compatibility-driver recipe](#) and verify the DSN's actual driver path. The original setup dialog configures the original driver; saving it after the compatibility setup can replace that setup. Use [setup-554-compatibility-driver.ps1](#) as described in the [driver recipe](#) to preserve the required Dharma 14 path.

Fix the ASQL System Directory Error

An unusual incomplete structure on the tested APPX installation prevented the earlier ASQL file-list query. The APPX system dictionary and its data existed; this was not a missing SYSDIR definition in APPX. The missing generated file was `0SA\Struct\SYSDIR.dat`, alongside existing `0SA\Data\SYSDIR.dat` and `.key` files. This is a troubleshooting case, not a routine installation step.

The recorded ODBC diagnostic was:

```
ERROR [00000] [Dharma][ODBC Dharma Driver 14.01.00.0000][Dharma/SDK] -1007 Can't open file
```

First confirm the gateway points to the intended APPX installation and the APPX [service account](#) can access its files. This error can have other file or permission causes. If the generated SYSDIR structure alone is absent:

1. Preserve the APPX system files and sign into the matching APPX release as a system administrator.
2. Open **System Administration** → **System Setup** → **System Administration File Management**.
3. Open **File Selection**, choose **Unselect All**, then **Select** with Application `0SA` and Name `SYSDIR`. Leave Database and Version blank. Confirm and return with **Cancel/End**.
4. Confirm that exactly **one file** is selected, then choose **Create Files**.
5. Existing data should report **File Already Exists**. Confirm `0SA\Struct\SYSDIR.dat` was generated, leave file management, and retry ASQL.

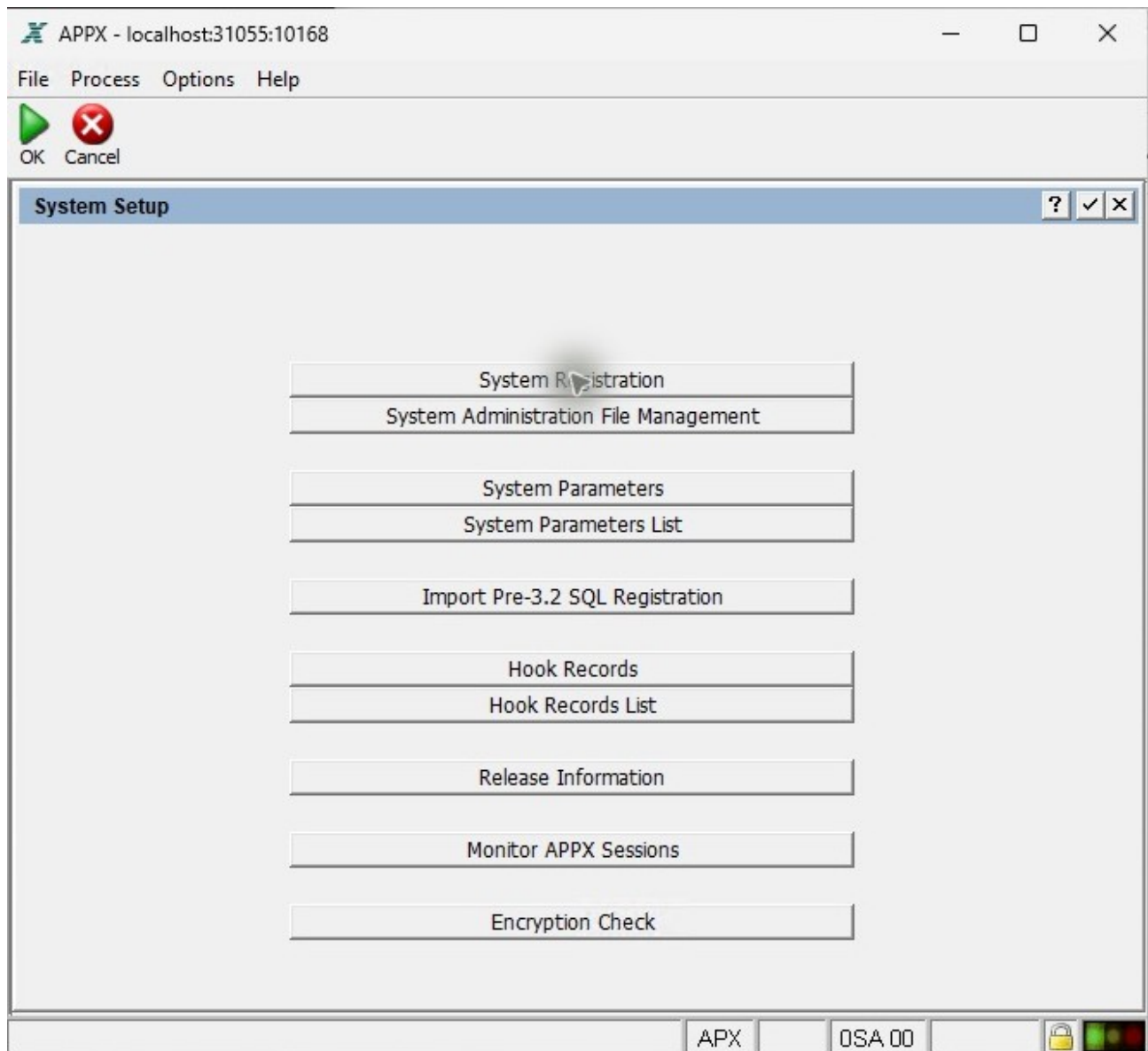


Figure 18. System Administration provides the matching file-management function.

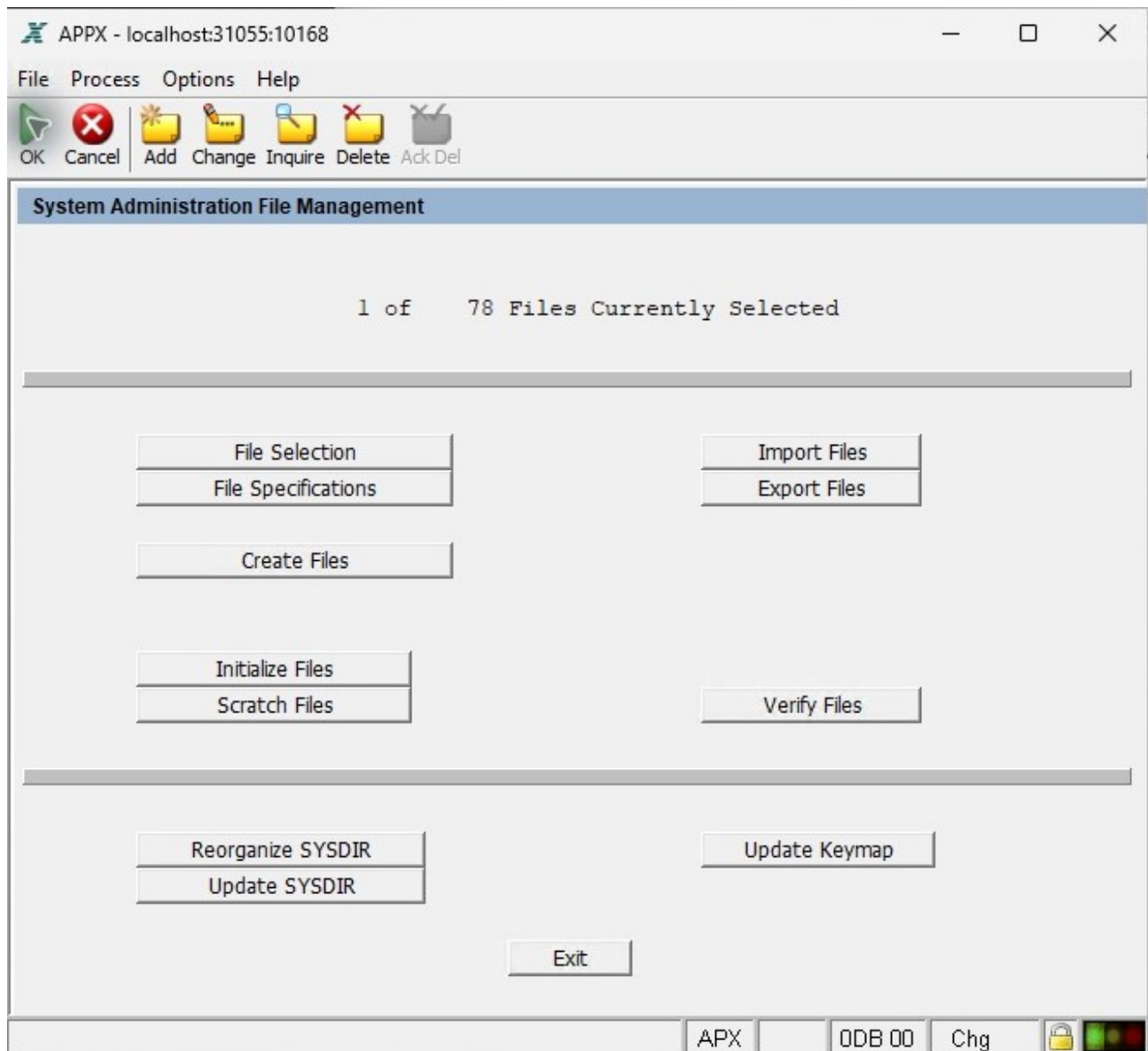


Figure 19. Limit this unusual repair to one selected file.

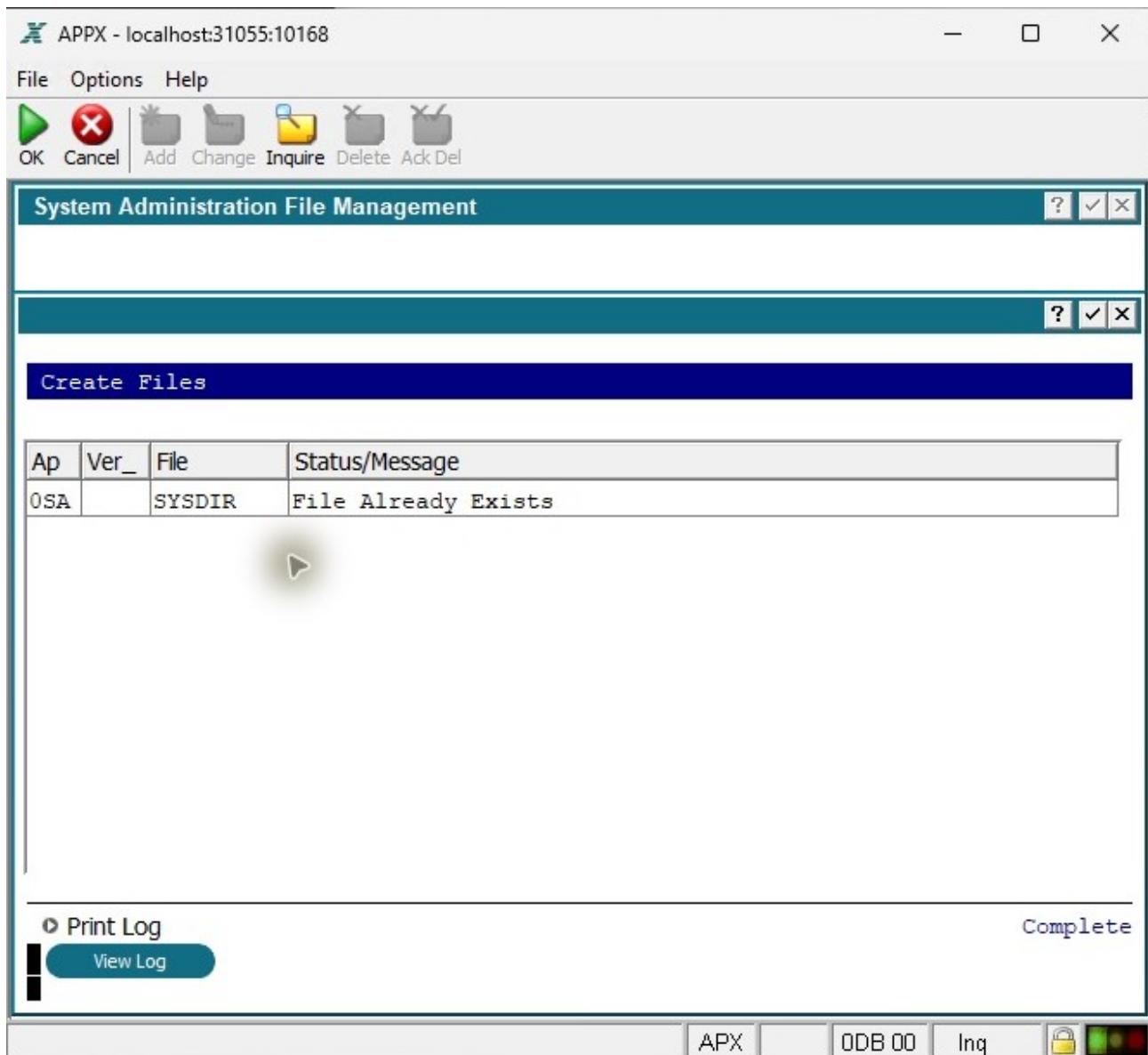


Figure 20. The existing SYSDIR data file is preserved.

In the test, APPX generated its matching structure and ASQL then listed DMO successfully. The existing SYSDIR data and key hashes were unchanged. Do not use **Initialize Files** or **Scratch Files** for this repair.

Fix Missing Files When the Gateway Will Not Start

Use this procedure only after [installing Microsoft's x86 runtime](#) and confirming that the specific gateway build still needs the files below. Ask APPX Support for a complete matching package first. The following separately sourced files reproduce the tested configuration when a complete package is unavailable. Do not apply the repair to a gateway that already starts and queries successfully.

The specific 5.5.4 gateway package tested on October 14, 2026 requires three 32-bit ICU 59 files and the 32-bit Boost Regex 1.71 file. Put these files beside `AppxODBC.exe` in `C:\Program Files (x86)\AppxODBCserver\bin`:

- `icuuc59.dll`
- `icuin59.dll`
- `icudt59.dll`
- `boost_regex-vc142-mt-x32-1_71.dll`

Obtain ICU from the [official APPX 6.2.1 32-bit gateway package](#), extracting only the runtime files. Obtain Boost from the [Boost project's 1.71.0 MSVC 14.2 32-bit binaries](#). Keep the 5.5.4 gateway executable.

1. Save the two downloads as `AppxODBCserver_windows-32.exe` and `boost_1_71_0-msvc-14.2-32.exe` in a working folder.
2. Open **Command Prompt as administrator** in that folder. Check the Boost download with:

```
certutil.exe -hashfile "boost_1_71_0-msvc-14.2-32.exe" SHA256
```

The expected SHA-256 is `29D623991A2B01EE605EF5A4B01181A87A26D108C5BFFBFE5C715A7FCAC4FAB6`. Extract the matching package:

```
.\boost_1_71_0-msvc-14.2-32.exe /VERYSILENT /SUPPRESSMSGBOXES /NORESTART /SP- /DIR=C:\APPX-Boost-171
```

Allow several minutes for extraction. The required DLL is `C:\APPX-Boost-171\lib32-msvc-14.2\boost_regex-vc142-mt-x32-1_71.dll`.

3. Download [prepare-554-gateway-runtime.ps1](#) into the same folder. This helper extracts and verifies the required files. Close active ODBC clients, then run these commands separately:

```
sc.exe stop "Appx ODBC Data Gateway Server for Windows"
powershell.exe -NoProfile -File .\prepare-554-gateway-runtime.ps1 -IcuInstaller .
\AppxODBCserver_windows-32.exe -BoostDll "C:\APPX-Boost-171\lib32-msvc-14.2\boost_regex-vc142-mt-x32-1_71.dll"
```

The script checks the APPX source package hash, extracts only the three ICU files, checks all four DLL hashes, preserves existing gateway copies in its work folder, and copies the verified files into the 5.5.4 gateway's `bin` directory. It refuses an existing work folder; use `-WorkDirectory` with a new path for a separate attempt. Return to [Start and check the gateway](#).

Verification: the four DLL hashes match the working test gateway. The extraction helper completed its hash checks and the matching 5.5.4 gateway restarted on TCP 1992.

Fix Import Errors Caused by Field Names

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

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

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](#).