Accessing Zen v14 from VBScript Using the ODBC Interface

A White Paper From

For more information, see our web site at
http://www.goldstarsoftware.com

Accessing Zen v14 from VBScript Using the ODBC Interface

Last Updated: February 2021

The Actian Zen database engine (formerly known as Actian PSQL) supports a wide variety of application programming interfaces (APIs) to access the data. Some of these interfaces leverage the lower-level interface, commonly known as the Btrieve API, to provide the needed performance and flexibility. However, other interfaces leverage SQL through several common interfaces, including ADO and ODBC. The nice thing about these interfaces is that they work to a wide variety of databases using the "least common denominator" access methods. However, for simple operations, you can accomplish quite a bit through these standard calls.

When accessing any database via ODBC, you have to have a programming environment in which to write your code. With some API's, you must stick with 3G and 4G languages like C, C++, Magic, Delphi, COBOL, and so forth. However, other API's (including ODBC) are directly accessible through scripting environments like VBScript and PowerShell.

This paper provides a straightforward example of using VBScript to query a table via a statement in SQL and return two fields from each row in the result set. The basis for this example is taken from a real-world customer example, but it has been simplified to show you some basic constructs.

Important Pre-requisites

The following pre-requisites should be in place before you start.

1) You should have a Windows operating system properly installed with CScript or WScript available.
2) You should have Zen v14 installed and running. If you are working on a stand-alone development machine, you can install either the Workgroup Engine or the Server Engine. (Both will work just fine with a 30-day trial.) If you are working within a shared environment, you should have a Zen v14 Workgroup Engine or Server Engine installed where the database files are located, and the Zen v14 Client installed on your development computer.

Clarifying the Problem Statement

The first step of any development project is to develop or refine the **problem statement** so that you know exactly what you are trying to do. You can then move on towards defining the functional requirements (what the code must do) and qualitative requirements (how the code must do it), and then you can start getting into the exact specifications (i.e. data input formats, data output formats, data structures needed, and so on).

This is a very simple project, so we will lump these all in together. Put simply, we want to issue a SQL statement against a database to return a number of rows of data, and then we want to display that data on the screen. This is fairly simplistic, but we can use the same logic to perform much more complex functions.

Verifying the Source Data

If we did our job correctly, then the Problem Statement, Requirements, and Specifications documents will already contain the exact file names field names, data types, and other information. However, it doesn't hurt to double check this right now. You'll then use the real field names in your scripts.

In our example, we are simply going to query the "Person" table in the DEMODATA database and retrieve the First_Name and Last_Name fields from each of the first 10 rows. Since this database already exists on each Zen/PSQL installation, you can move right to the next step.

Creating a VBScript Script

VBScript "scripts" are simply text files that have the extension of ".vbs". You can use Notepad to create and work with the files, or you can use any text editor. Make sure that youe file ends with ".vbs" and not ".vbs.txt".

In our case, we are going to create a simple text file called "QueryPersonNames.vbs" and open it, then paste in the code from Appendix A.

Understanding the Script

Our sample code is broken down into 3 sections, so let's look at each in turn.

The first section is used to set up some variables that will be used within the script:

- Line 1 is used to tell VBScript that we will explicitly define every variable. While this is an extra step, it does prevent some hard-to-find errors introduced by typing errors.
- Line 2 defines the two object variables we will be using in the script.

The second section sets up the ODBC/ADO access environment.

- Line 4 simply lets the user know we are opening the connection, in case something goes sideways.
- Line 5 creates a new **ADODB Connection** object.
- Lines 6 provide a connection string to be used with the **ADODB Connection** object. Note that we are on a 64-bit Windows box, so we are using the name of the 64-bit ODBC Driver. If you are running a 32-bit workstation, then you may need to be using the 32-bit ODBC Driver name, which is {Pervasive ODBC Client Interface} instead.
- Line 7 configures the ADODB Connection to use client-side cursors, where the data is extracted in its entirety and available for use after running the query.

The third section is where the real work of the process is done.

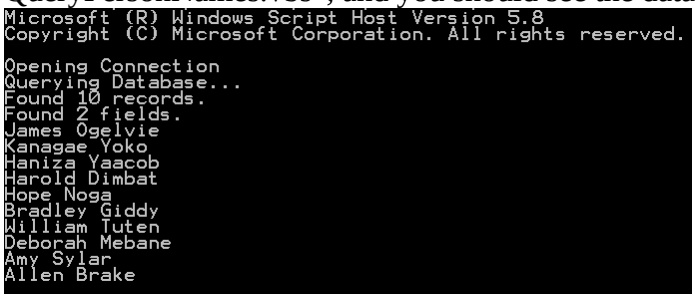
- Line 9 is again there just for user feedback.
- Line 10 creates a new **RecordSet** object and assign it to the object *rs*.
- Line 11 opens the **RecordSet** object and passes in the SQL query to be run. For a complicated query or for one with a huge record set, this statement may take some time to complete.
- Lines 12-13 are again providing user feedback to confirm that the query ran and returned a data set with a set number of records.

- Line 14 checks to make sure that there are records to be returned before starting the record retrieval process.
- Line 15 retrieves the first record.
- Line 16 starts a loop to iterate over every row in the *rs* collection, terminating when the EndOfFile (EOF) is reached. Line 17 writes the *First_Name* and *Last_Name* fields from the record to the screen. Note how we can use the ordinal field number offset (starting at 0), or we can use the field name itself here, or we can use them interchangeably. Using the field names may require more typing, but it can allow the query to be more easily modified in the future, and it will likely yield more readable code, easing future maintenance or changes.
- Line 18 increments the *rs* record set object to point to the next record.
- Line 19-20 terminate the DO loop and end the IF statement.
- Line 21 terminates the application. While this is optional, it can be helpful to have this at the end of your parent process if you have subroutines that follow this statement in the source file.

I should also note that this script was written as a teaching tool, and it should not be considered a well-written script over all. First of all, it lacks error handling of any kind, so it will choke on errors. In addition, the code has been written for simplicity and clarity, whereas many of the statements could be combined or otherwise written more concisely.

Running Your VBScript Script

Once you have saved your script, go to a command prompt and start it with the command "cscript QueryPersonNames.vbs", and you should see the data returned directly on the screen:



A few caveats are worth noting at this time:

- If you are running a 32-bit workstation, then you must provide a 32-bit ODBC Driver name in Line 6.
- If you want to use the 32-bit ODBC Driver on a 64-bit machine, then you CAN do this also, but you must use the 32-bit CScript. This can be found in the C:\Windows\SysWow64 folder, making the command: "C:\Windows\SysWow64\cscript QueryPersonNames.vbs".
- If you don't like the command prompt, you can ALSO output each line as a separate dialog box in Windows. Simply change your command from "cscript" to "wscript". In this example, though, you'll be generating 14 separate dialog boxes, so be careful with this!

You have now successfully written your first VBScript script!

Expanding Your Knowledge

Want to know if you understand what is going on? Test your understanding by trying the following:

1) Modify the script to return the user's ID in addition to the name, and format the data into columns. (You'll need to modify the SQL statement AND fix up Line 17.)
2) Modify the script to return the Name and Salary of each Faculty member. (You can either modify the SQL query to join in the Faculty table and return all three fields at once, or you can use two separate queries and loops.)
3) If you look carefully, the connection and record set are opened, but never explicitly closed. They will be closed when the script exits, of course, but this is considered bad form. Add the needed lines to release these resources when you are done with each before terminating the script.
4) **For expert developers only:** Modify the script to prompt for the SQL query, and then output ALL columns from the SQL query to the screen in comma-delimited format. (You may wish to use `Read-Host` for this.) You've just written your own query tool!

From here, you're only limited by your imagination!

Finding More Help

If you have other problems getting this to work, you can contact Actian directly through their web forums at https://communities.actian.com/s/ for more help. If you need some additional hand-holding, Goldstar Software may be able to assist you as well through our Developer Jump Start Program. You can contact us at 1-708-647-7665 or via the web at http://www.goldstarsoftware.com.

Appendix A: Sample Script

The following script makes use of the ADO API and can be used as instructional and sample code.

```
Option Explicit
Dim conn, rs

WScript.Echo "Opening Connection"
Set Conn = CreateObject("ADODB.Connection")
Conn.Open "Driver={Pervasive ODBC Interface};ServerName=localhost;DBQ=Demodata;UID=Master;PWD=None;"
Conn.CursorLocation = 3 'adUseClient

WScript.Echo "Querying Database..."
Set rs = CreateObject("ADODB.RecordSet")
rs.Open "SELECT TOP 10 Last_Name, First_Name FROM Person", Conn, 3, 1
WScript.Echo "Found " & rs.RecordCount & " records."
WScript.Echo "Found " & rs.Fields.Count & " fields."
If rs.RecordCount > 0 Then
  rs.MoveFirst
  Do While NOT rs.EOF
    WScript.Echo rs(1).Value + " " + rs("Last_Name").Value
    rs.MoveNext
  Loop
End If
WScript.Quit 0
```