Getting started with the PDO_ODBC PHP extension on the IBM i

With the introduction of Open Source Package Management, IBM has been providing more and more 64-bit versions of open source software compiled to run on IBM i. In that spirit, they have provided a unixODBC driver and driver manager that implement unixODBC, which seeks to be the definitive standard for ODBC on non MS Windows platforms. The unixODBC driver is now the preferred driver to access DB2 on IBM i, deprecating the SQL-CLI driver that has long been the DB2 driver used in PASE projects, including PHP.

Zend is providing the PDO_ODBC PHP driver for our customers who want to move to the new unixODBC driver. PDO_ODBC is available in Zend Server version 2021.1 and in ZendPHP. The ibm_db2 and pdo_ibm PHP extensions will continue to be available for customers not wanting to make the switch. For those customers wanting to move to unixODBC, this article provides some tips that can help you get started.

Note: The unixODBC driver is 64-bit, so we can only provide PDO_ODBC in our 64-bit packages. Zend Server 2019 and earlier versions are 32-bit, so they do not support the driver.

STEP 1: Install the Open Source Package Management, if needed.

To use PDO_ODBC on IBM i, you must first download and install the unixODBC driver and driver manager from IBM i Access. This involves downloading the package from the IBM i Access Client Solutions download page, and installing using yum. A prerequisite is to have Open Source Package Management installed on your IBM i, which provides the yum utility.

If you are using version 2020 or newer of Zend Server, or any version of ZendPHP, then you likely already have Open Source Package Management. If you don't have it, instructions to install it are here:

Getting started with Open Source Package Management in IBM i ACS

STEP 2: Install the unixODBC driver and driver manager, if needed.

The unixODBC driver and driver manager are bundled with the ibm-iaccess package. You can check in the "Installed packages" tab of Open Source Package Management to see if it is installed. You can also check for it with this PASE command:

yum list ibm-iaccess

If you do not have the ibm-iaccess package, you can find the instructions to install the ibm-iaccess package here:

IBM i Access ODBC Installation

STEP 3: Install and enable the PDO_ODBC extension, if needed.

Note: This step differs depending on whether you are using Zend Server or ZendPHP. Please choose the instructions for the correct Zend product you are using.

Zend Server - enable

If you are using Zend Server, the PDO_ODBC extension will be installed automatically starting in version 2021.1. You can enable and disable the extension using the Zend Server User Interface, just as you can do for any extension.

Enabling PDO_ODBC will disable the odbc, pdo_ibm, and ibm_db2 extensions.

The unixODBC driver creates a linking failure in PHP if any extension using the old SQL-CLI (libdb400.a) driver is also loaded. This is because there is a naming conflict between the drivers that will cause the linked SQLSetEnvAttrr to be called from the wrong driver, and PHP will fail to start. So, before you can enable the PDO_ODBC extension, you need to disable the odbc, pdo_ibm, and ibm_db2 extensions. (You need to disable the odbc extension because it uses SQL-CLI instead of unixODBC. This was done to maintain backward compatibility with older versions of Zend Server).

This will happen automatically in Zend Server. If you enable PDO_ODBC, the others will be disabled. If you enable any of the others, PDO_ODBC will be disabled. It is important to understand that you can not run scripts using incompatible DB2 drivers at the same time. For example, if you migrate some of your ibm_db2 applications to PDO_ODBC, you must migrate all of your ibm_db2 applications to PDO_ODBC.

ZendPHP - install and enable/disable

If you are using ZendPHP, which is a different product than Zend Server, you will need to install the extension for the version of PHP you are using. For example, if you are using version 7.3 of PHP, the extension can be installed with this PASE command:

yum install php73zend-php-odbc

To install the extension for a different version of PHP, just replace the '73' to your correct version. For example, for PHP 7.4, the package would be php74zend-php-odbc. It is important that the PHP version matches. If the PHP version does not match, the extension will not load.

You may also opt to use Open Source Package Management to install the extension for ZendPHP, since you already have the repository defined from the installation of ZendPHP.

You can enable and disable the pdo_odbc extension in the 30-pdo_odbc.ini file, by enabling or disabling the "extension=pdo_odbc" statement. To disable it, place a semicolon in the first character of the line to make it a comment. To enable it, remove the semicolon.

Enabled:

```
extension=pdo_odbc
```

Disabled:

```
;extension=pdo_odbc
```

The location of this file is PHP version dependent. For example, if you are using PHP 7.3, you will find the file here:

```
/QOpenSys/etc/php/73zend/conf.d/30-pdo_odbc.ini
```

STEP 4: Test your ODBC connection using the PDO_ODBC extension.

Here is a simple example script you can use to verify your ODBC connection is working (put in valid values for user and password):

pdo_test.php

```
<?php
/* Connect to a local DB2 database using driver invocation */

echo "<pre>Test ODBC connection<br>";
ini_set("display_errors", 1);

$dsn = 'odbc:*LOCAL';
$user = 'user';
$password = 'password';

$db = new PDO($dsn, $user, $password);
if (!$db) die("Connection failed");
if ($db) echo "Connected<br>";
$library = "QGPL";
$sql = "select * from qsys2.systables where table_schema = '$library'";
$data=$db->query($sql,PDO::FETCH_ASSOC);
foreach ($data as $row) {
print $row['TABLE_NAME'] . "<br>";
}
?>
```

The expected output is the heading, "Connected", and a list of files (tables) in the QGPL library (schema).

STEP 5: Begin to explore the available ODBC connection settings and PDO_ODBC attributes.

odbc.ini keywords and values

You can specify many more connection options in the odbc.ini file, which is here in the IFS:

```
/QOpenSys/etc/odbc.ini
```

The contents of odbc.ini will look something like this:

```
### IBM provided DSN - do not remove this line ###
[*LOCAL]
Description = Default IBM i local database
Driver      = IBM i Access ODBC Driver
System      = localhost
UserID      = *CURRENT
### Start of DSN customization
### End of DSN customization
### IBM provided DSN - do not remove this line ###
```

You will notice there is a section in odbc.ini with the heading [*LOCAL]. This is why we can just specify the connection string ($dsn) as 'odbc:*LOCAL'.
You will also notice these comments:

```
### Start of DSN customization
### End of DSN customization
```

You can add any additional keywords and values here. You will notice they have already provided values for the Description, Driver, System, and UserID values, and you can simply follow the pattern of Keyword = Value to add more in the customization section. You can find the keywords and values here:

Connection string keywords

Referring to the table linked above, use the keywords labeled 'ODBC.INI' when you add them to the odbc.ini file.

Back up your odbc.ini file.

Remember to keep a copy of your customized odbc.ini file someplace safe. This file is part of a distributed package, so you may need to restore your changes after an upgrade or system restore.

PDO attributes

The construct, which is the "new PDO($dsn, $user, $password);" statement in the example script pdo_test.php, can contain an array of PDO attributes. You can see a list of PDO attributes on the PDO::setAttribute documentation page here:

PHP: PDO::setAttribute

For example, you can set up the error mode like this:

```
$db = new PDO($dsn, $user, $password, array(
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,));
```

Attributes can also be set in a separate statement using the setAttribute() method:

```
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

The error mode attribute defines how errors are handled. You can learn more about that here:

PHP: PDO Errors and error handling

Also notice you can set the user and password in the construct, overriding what is in odbc.ini.