How to configure generic ODBC connector with built-in ODBC driver

Last updated by | Xing Wei | Feb 24, 2023 at 6:55 AM PST

Contents

Background

From SHIR version > 5.16, we have added several security enhancement to only allow certain properties in connection string of built-in (Simba/Progress) ODBC drivers (see full allow list here). If customer is using any connStr property not in allow list, they are likely to encounter some error.

To fix it, we need to bring those connStr properties not in allow list to external security review, which usually takes long and hard to make commitment on ETA.

As a workaround, we could try proposing **generic ODBC** connector & **Simba/Progress driver (binary form, DLLs)** & **DSN-less connection string** approach.

DSN vs. DSN-less for ODBC connector

ADF generic ODBC connector supports both DSN & DSN-less approach.

- **DSN** approach means you need to install ODBC driver on SHIR, configure system DSN, then use ODBC connector to consume the DSN. We have TSG dedicated for this DSN approach with Oracle native driver.
- **DSN-less** approach means to directly invoke driver DLL via connection string, where you do not have to install the driver. The driver that Progress delivered to MS is NOT installer-based, but rather binary version, whose DLLs are invoked via DSN-less approach.

  MSFT INTERNAL ONLY: This **DSN-less** approach underlying invokes the exact same code path as the built-in drivers shipped with SHIR.

Connection String

For DSN-less approach the key point is to compose the correct connection string to be filled in generic ODBC connector.

Progress Oracle driver + ODBC connector

We received a lot relevant cases turned out to leverage workaround as **Progress Oracle driver** + ODBC connector approach, so let us take it as example.

Below is a sample Oracle driver connection string. Note that the properties in **bold** are implicitly appended in Oracle connector, to ensure a consistent experience, make sure these properties are also attached to the connection string of ODBC connector

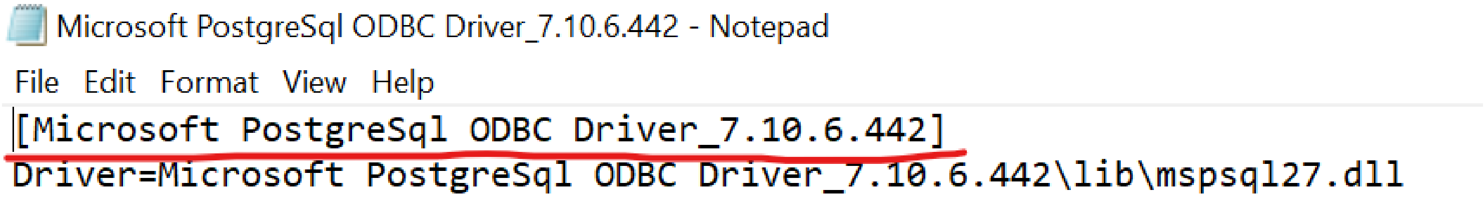
Driver={Microsoft Oracle ODBC Driver};Host=<host>;Port=<port>;Sid=<sid>;uid=<user name>;pwd=<password>;**arraySize=10240000;bulkBinaryThreshold=-1;cachedCursorLimit=0;connectionRetryCount=3;enableNcharSupport=1;procedureRetResults=1;enableDescribeParam=1;enableDescribeParam=0;enableTimestampwithTimezone=1;timestampTruncationBehavior=1;fetchTSWTZasTimestamp=1;enableBulkLoad=1;columnSizeAsCharacter=1;ignoreOdbcTimestampLiteral=1;enableOracleBatchFailureReturnsError=1;**

Other built-in ODBC drivers + ODBC driver

- for 'driver' property, please use the same name as stated in the first line of *.ini file under SHIR driver folder. It's always recommended to use the default driver, whose corresponding .ini file name does not have "_{version number}".

Below is an example of using the side-by-side driver. Make sure the driver name is enclosed with curly brackets.

```
Driver={Microsoft PostgreSql ODBC Driver_7.10.6.442};
```


Microsoft PostgreSql ODBC Driver_7.10.6.442 - Notepad
File Edit Format View Help

```
[Microsoft PostgreSql ODBC Driver_7.10.6.442]
Driver=Microsoft PostgreSql ODBC Driver_7.10.6.442\lib\mspsql27.dll
```

- for rest properties of the connection string, please follow the same as our connector official doc

Relevant information

How to detect connection string property drop?

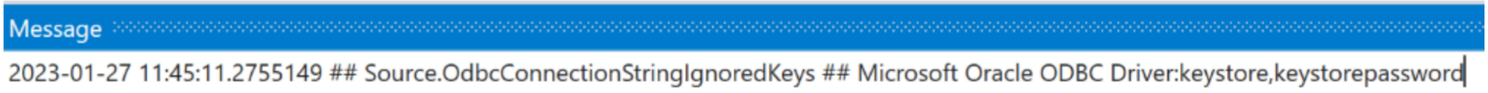
You could run below Kusto function.

```
query_CustomLogEvent(@'activity_run_id')
| whereTraceMessage == "RuntimeTelemetry"
```

or

```
DiagnosticsSchemaApiByActivityId('activity_run_id')
```

If you see similar results like below, it means the connection string contains some properties not in allow list and got dropped during runtime.



```
2023-01-27 11:45:11.2755149 ## Source.OdbcConnectionStringIgnoredKeys ## Microsoft Oracle ODBC Driver:keystore,keystorepassword
```

This screenshot shows that two connection string properties **keystore**, **keystorepassword** got **dropped** in runtime, the two dropped properties are related with Oracle Wallet SSL authentication.