OBJECTIF LUNE

Extracting Database Information in a Data Mapping Configuration

Created By: Rodrigue Noubissie

Last Updated: 01/11/2017

TOC

Introduction

Sometimes, the information and data that is required to build a form is scattered across multiple data sources such as Database Management Systems (DBMS) and it is often a challenge to get all the required data in one source. Luckily, the OLConnect Data Mapper can use the JDBC (Java Database Connectivity) driver to connect to these third-party DBMS. JDBC makes it possible to establish a connection with a database, send SQL statements and process the results.

This guide aims to provide the steps that are required to retrieve data from a remote Microsoft SQL Server using the “Action” step in the Data Mapper

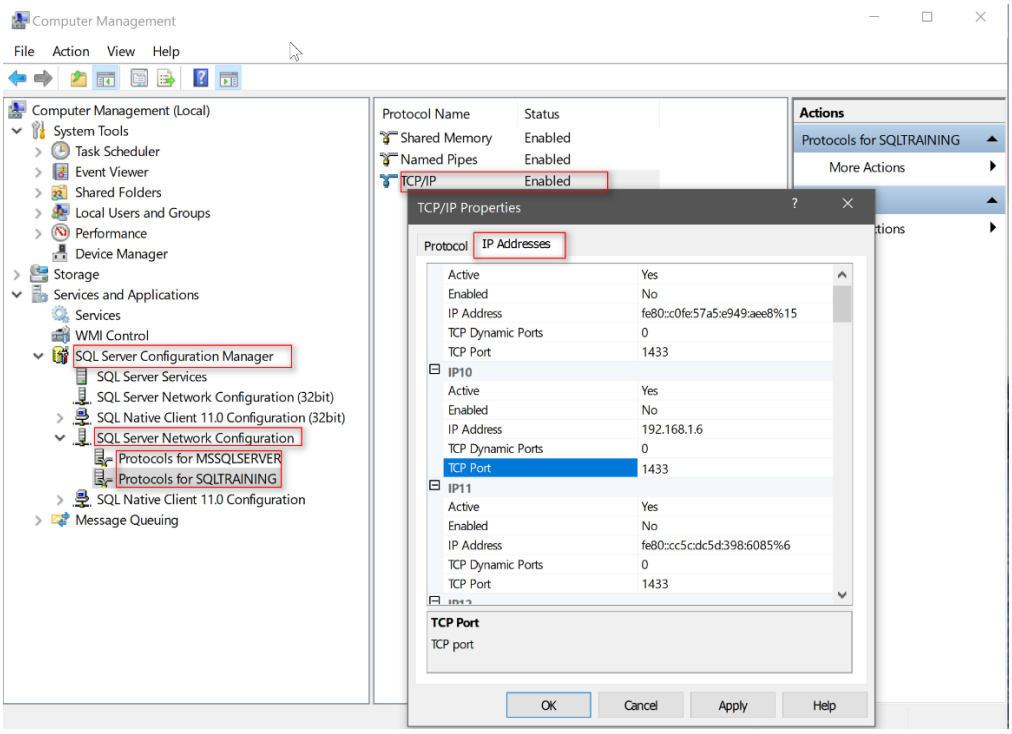
From a Microsoft SQL Database

The first part of this guide aims to provide the steps that are required to retrieve data from a remote Microsoft SQL Server using the “Action” step in the Data Mapper

Prerequisites for Microsoft SQL

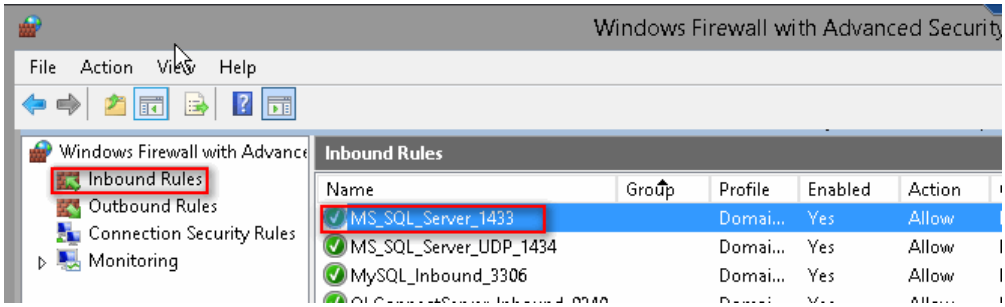
To ensure a successful connection to Microsoft SQL Server, the below configuration should be made on the machine running your SQL Server instance:

1. You need to enable mixed mode security when you install Microsoft SQL Server so that you can connect using a user name and password.
2. The JDBC driver only works with the TCP/IP protocol which is disabled by default on SQL Express. You need to enable the TCP/IP Protocol from the SQL Server Configuration Manager that ships with SQL Express and re-start the SQL Server service. Look under SQL Server Network Configuration ->Protocols for SQLEXPRESS-> TCP/IP->Enable.
3. Set the TCP Port by right-clicking on “TCP/IP”, then click on “Properties” and clicking on the “IP Addresses” tab. The Default port is ***1433.***

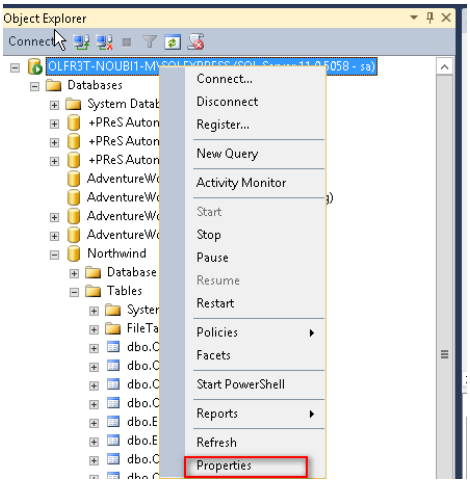


4. Restart the SQL Server Service to apply the changes.

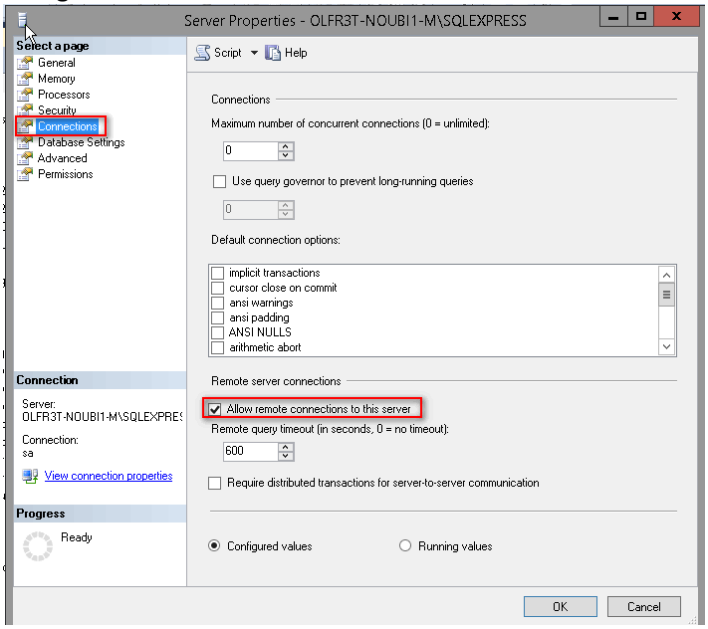
5. Add TCIP Port 1433 and UDP Port 1434 in your firewall Inbound Rules



6. Allow remote connection to the SQL Server. Log into your Microsoft SQL Server instance from SQL Server Management Studio. Right click the server and click on *Properties*.



Navigate to *Connections* and ensure that *Allow remote connections to this server* is checked.



Processing SQL Statements with JDBC

In this example, we are supplied with a CSV data file, which contains information about customers' orders' details such as the OrderID, CustomerID and Shipping Address and dates details; but what is missing from the CSV data file is the actual customer's name (CompanyName) and a contact name (ContactName). This information available in the Customers Table of the Northwind database, which resides on a Microsoft SQL Server.

In general, to process any SQL statement with JDBC, you follow these steps:

- Establish a connection to the database on the SQL Server.
- Create a statement.
- Execute the query.
- Process the ResultSet object.
- Close the connection.

Example: The below script establishes a connection to the Northwind database on a local SQL Server, queries the [Customers] table using the CustomerID field taken from the CSV data file.

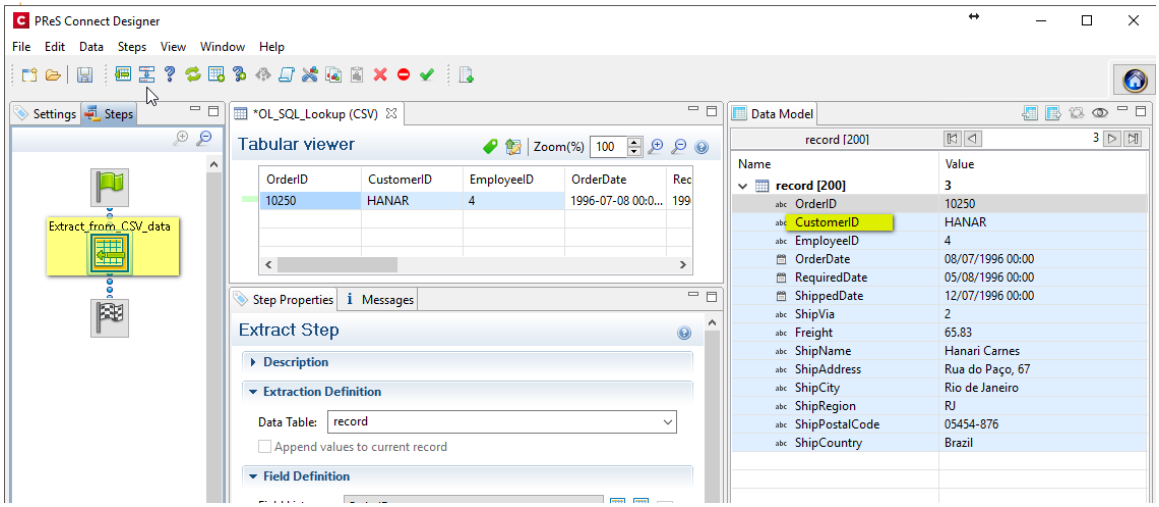
```
var connectionURL =
"jdbc:sqlserver://localhost\\SQLEXPRESS:1433;integratedSecurity=false;databaseName=Northwind";

var custID = data.extract('CustomerID',0).trim();

if(custID){

        var sqlConnection = db.connect(connectionURL, "username", "password");

        var sqlQuery = "SELECT [CompanyName],[ContactName] FROM [dbo].[Customers] where
        CustomerID=" + "'" + custID + "'";

        resultSet = sqlConnection.createStatement().executeQuery(sqlQuery);

        resultSet.next();

        resultSet.close();
        sqlConnection.close();
}
```

| *The documentation on how to build the Connection URL is available [on the MSDN website]*

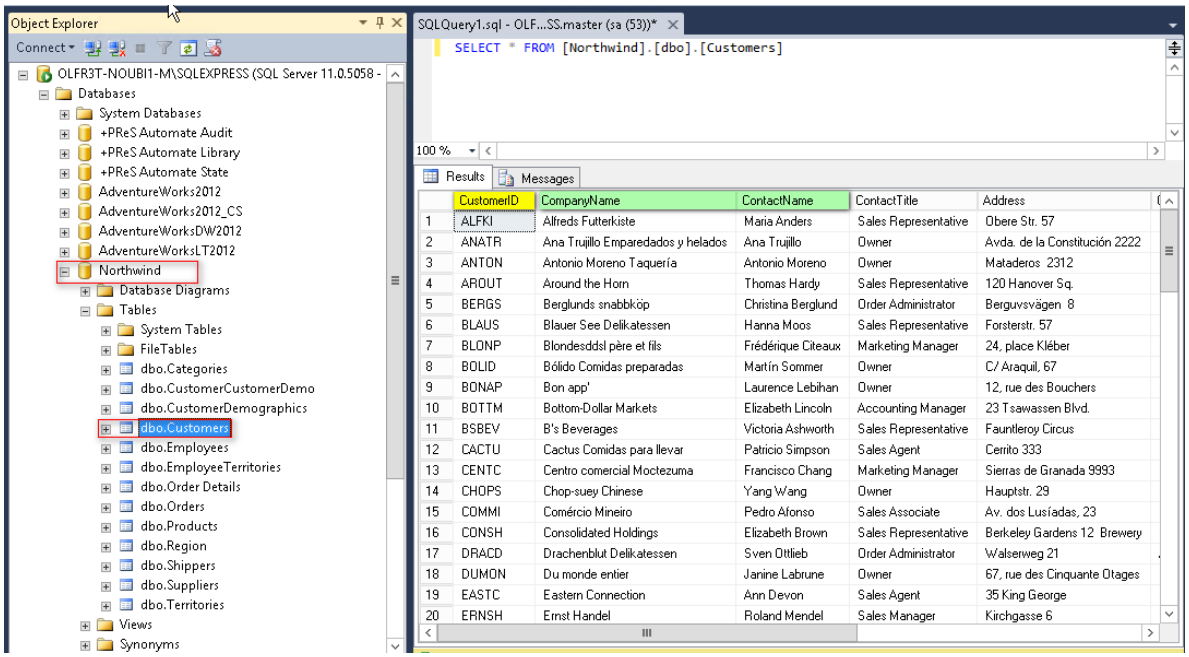
Example Application in Connect Data Mapper

In this example, we have a CSV data file of customers' orders, which does not include the actual customers name and contact details. Instead, this information resides on a remote Microsoft SQL Database table. This example will demonstrate how the *companyName* and *contactName* fields can be retrieved from the *[dbo].[Customers]* table of the *[Northwind]* database.

The first step is to load the CSV in the Data Mapper, then Add Extract Step to perform a standard:

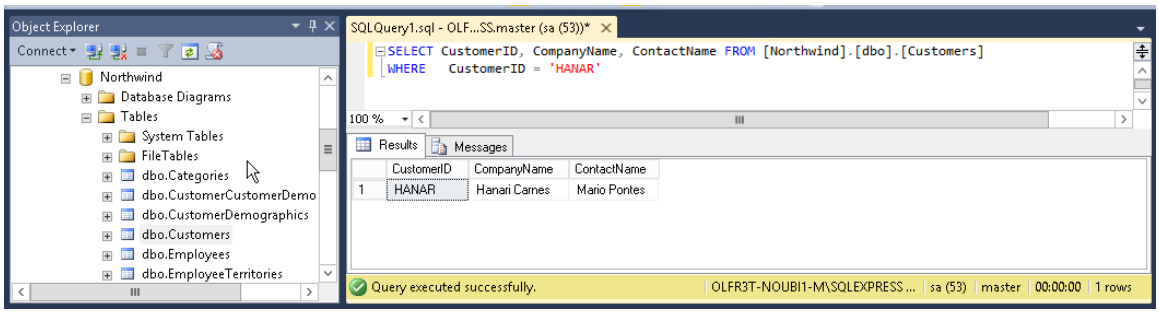


Now that the order details have been extracted, we can use the *customerID* field common between the CSV data file and the *[dbo].[Customers]* table in a SQL SELECT statement to then retrieve the corresponding customer's details such as *companyName* and *contactName*



For instance, in Microsoft SQL Management Studio, we would normally run the below *SELECT* statement to retrieve the CompanyName and ContactName of the order where the CustomerID is 'HANAR':

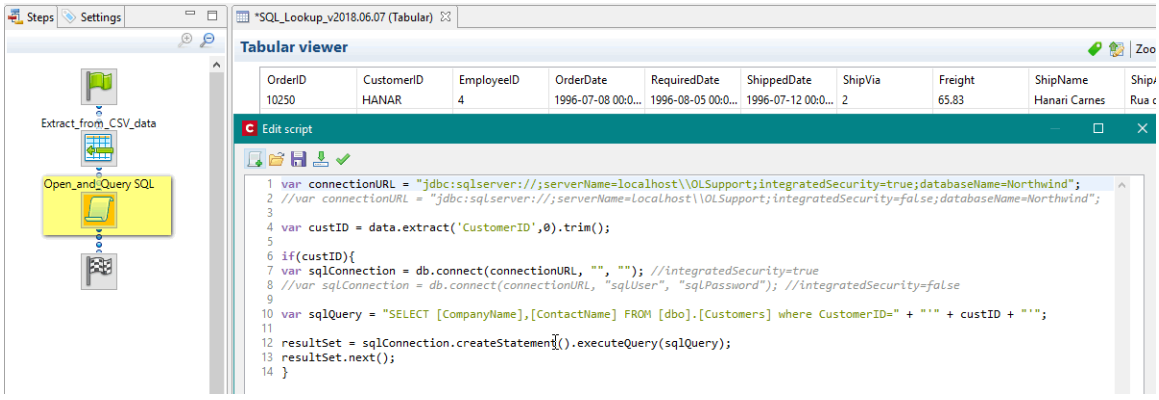
```
SELECT CustomerID, CompanyName, ContactName FROM [Northwind].[dbo].[Customers]
WHERE  CustomerID = 'HANAR'
```



We can execute the same command in the Data Mapper with the following steps:

After the above Extract step, Add Action step to open a connection to the SQL database, then query relevant table and save the result of the query in the *resultSet* with the following script:

```
var connectionURL =
"jdbc:sqlserver://;serverName=localhost\\OLSupport;integratedSecurity=true;databaseName=Northwind";
//var connectionURL =
//"jdbc:sqlserver://;serverName=localhost\\OLSupport;integratedSecurity=false;databaseName=Northwind";

var custID = data.extract('CustomerID',0).trim();

if(custID){
var sqlConnection = db.connect(connectionURL, "", ""); //integratedSecurity=true
//var sqlConnection = db.connect(connectionURL, "sqlUser", "sqlPassword"); //integratedSecurity=false

var sqlQuery = "SELECT [CompanyName],[ContactName] FROM [dbo].[Customers] where CustomerID=" + "'" +
custID + "'";

resultSet = sqlConnection.createStatement().executeQuery(sqlQuery);
resultSet.next();
}
```



Once we have a *resultSet*, we can now extract the relevant *CompanyName* and *ContactName* from it. It can be necessary to make sure the resultSet is not empty. After **resultSet.next()**, check the current row with **getRow()**. If it returns 0, then no data was included in the resultSet. If getRow()>=1, then you have at least one row.

To do this, simply add an Extract Step. A new field named *Field* is automatically added in the Data Model. To rename the field, click on the *Order and Rename* icon in the *Field Definition* window under the Extract Step properties.

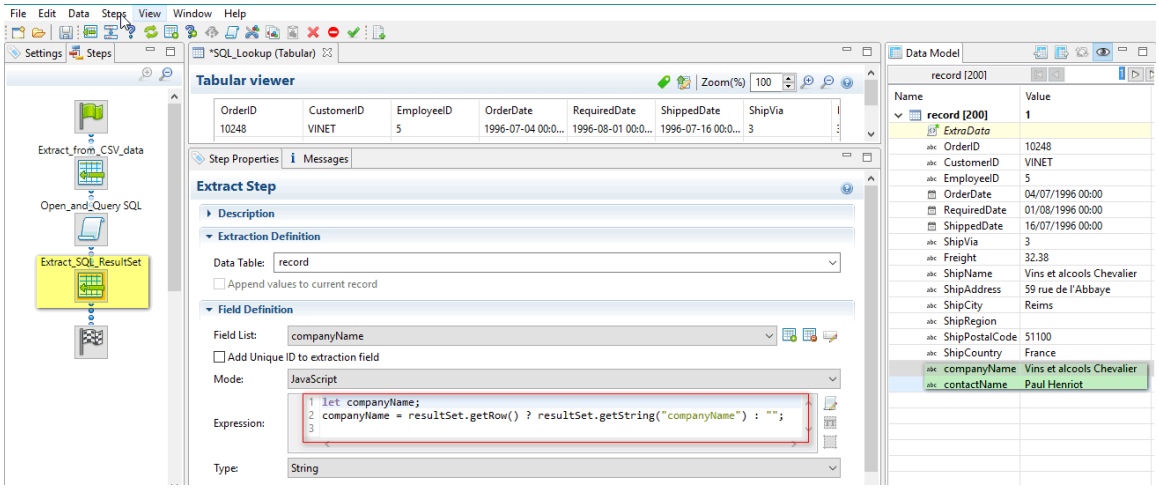
Make sure the Field Definition Mode is set to JavaScript and insert the following JavaScript Ternary expression to extract the *CompanyName*:

```
let companyName;
companyName = resultSet.getRow() ? resultSet.getString("companyName") : "";
```

To extract the *ContactName,* right-click on the above Extract step, select "*Add a Step*" and then select *"Add Extract Field".* Rename the field, set its Definition Mode to JavaScript and insert the following expression:

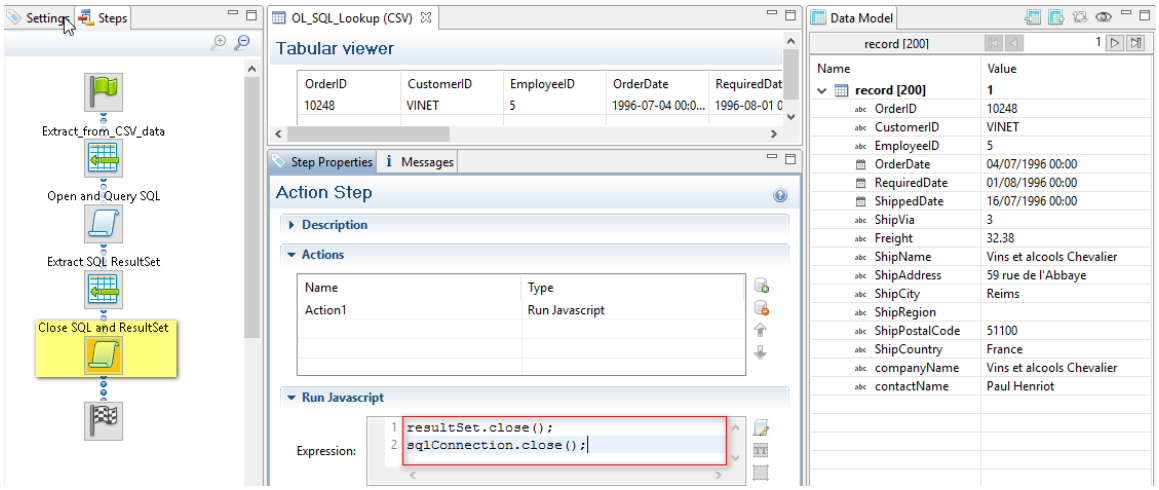
```
let contactName;
contactName = resultSet.getRow() ? resultSet.getString("ContactName") : "";
```

Note: In case where the ResultSet is empty, instead of returning the empty string, you may choose to output a default value or run another expression.



Finally close the ResultSet and the SQL connection as well

```
resultSet.close();
sqlConnection.close();
```



Microsoft Access, Microsoft Excel and CSV Lookup

Prerequisites

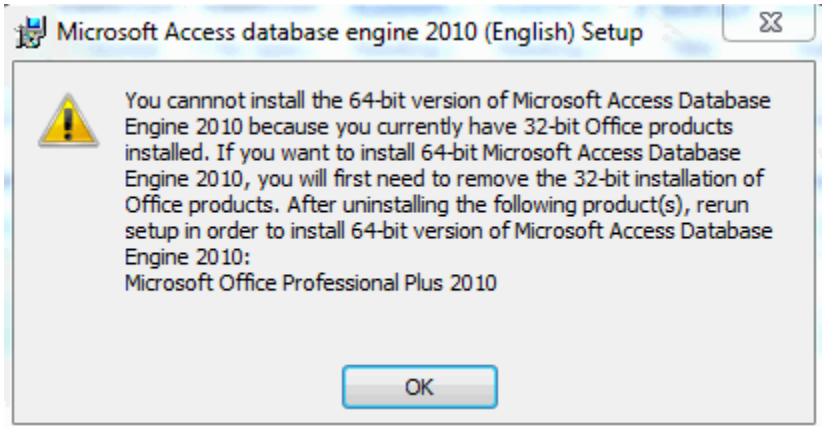
Since OL Connect is a 64-bit application, we will need the following two prerequisites

- ☐ Microsoft Access Database Engine 2010 for 64-bit Windows
- ☐ 64-bit DSN

Microsoft Access Database Engine 2010 for 64-bit Windows

Download and install Microsoft Access Database Engine 2010 for 64-bit Windows from the following link:
http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=13255

Note that launching the installation of a Microsoft Access Database Engine in the usual way, on a machine with an Office installation architecture different from the current one (e.g. 32-bit on 64-bit), may cause the installation to fail.

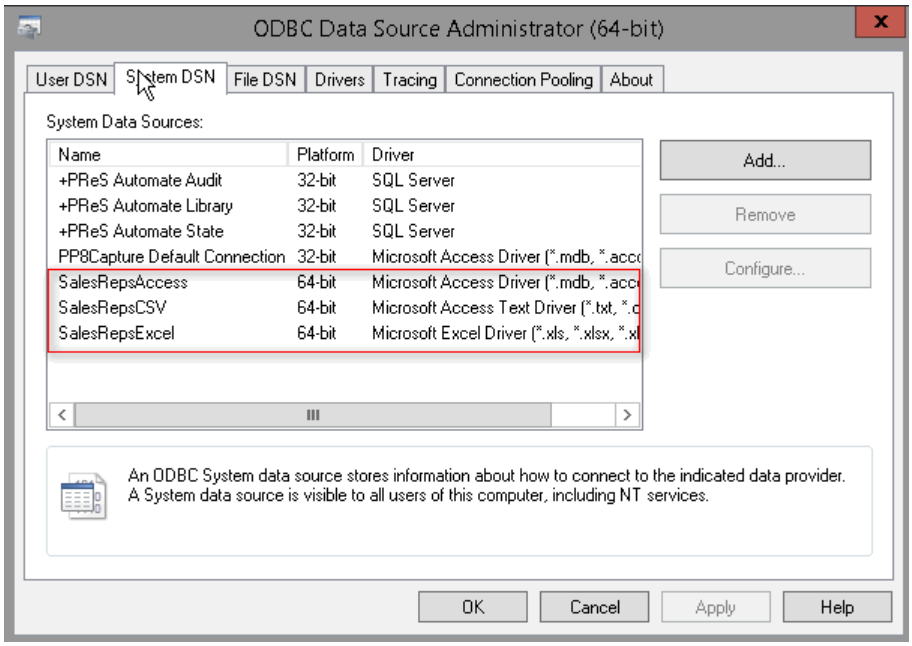


To have it run properly, you need to launch it from a command line with the "**/passive**" argument specified:

Create 64-bit Data Source Names (DSN)

To create a 64-bit System DSN:

- Open the Windows Control Panel and navigate to Administrative Tools > ODBC Data Source(64-bit)
- Click on the "System DSN" tan and click on Add
- Select the driver that corresponds to your file type
  - For Access files, select Microsoft Access Driver (*.mdb, *.accdb)
  - For CSV files, select Microsoft Text Driver (*.txt, *.csv)
  - For Excel files, select Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
- Click Finish to select your Access, CSV or Excel file
- Give your DSN a name. We will call it "SalesRep" for this example



Building JDBC-ODBC Connection URL

The JDBC-ODBC Bridge allows Java applications to use the JDBC API with many existing ODBC drivers.

The general form of the connection URL for Access, Excel and CSV files is:

```
"jdbc:odbc:DSN_Name"
```

where: *DSN_Name* is the ODB Data Source Name;

Hence, to connect to either of the above database, we can use the following code:

```
//To connect to a Excel database
var connectionURL = "jdbc:odbc:SalesRepsExcel";
var excelConnection = db.connect(connectionURL,"","");

/* To connect to CSV database
var connectionURL = "jdbc:odbc:SalesRepsCSV";
var csvConnection = db.connect(connectionURL,"","");
*/

/* To connect to Access database
var connectionURL = "jdbc:odbc:SalesRepsAccess";
var accessConnection = db.connect(connectionURL,"","");
*/
```

Querying a Microsoft Access Database

Once we have a JDBC connection object, we can then use it to create statements and execute queries, which will return a result set. For a Microsoft Access database, the code is as follow

```
var connectionURL = "jdbc:odbc:SalesRepsAccess";
var repID = record.fields.RepID;

if(repID){
var accessConnection = db.connect(connectionURL,"","");
var accessQuery = "SELECT * FROM SalesReps WHERE RepID=" + "'" + repID + "'";
resultSet = accessConnection.createStatement().executeQuery(accessQuery);
resultSet.next();
}
```

Querying a Microsoft Excel Database

Querying a Microsoft Excel database is similar with the procedure for Access, the only difference here is that is that Microsoft Excel Sheet name you are querying from must be followed by the $ sign and enclosed in square brackets:

```
var connectionURL = "jdbc:odbc:SalesRepsExcel";
var repID = record.fields.RepID;

if(repID){
var excelConnection = db.connect(connectionURL,"","");
var excelQuery = "SELECT * FROM [SalesReps$] WHERE RepID=" + "'" + repID + "'";
resultSet = excelConnection.createStatement().executeQuery(excelQuery);
resultSet.next();
}
```

Querying a CSV file

The code for querying a CSV file is similar with the only difference that the table name to query from is the actual CSV file name and must be enclosed in double quotes:

```
var connectionURL = "jdbc:odbc:SalesRepsCSV";
var repID = record.fields.RepID;

if(repID){
var csvConnection = db.connect(connectionURL,"","");
var csvQuery = "SELECT * FROM \"SalesReps.csv\" WHERE RepID=" + "'" + repID + "'";
resultSet = csvConnection.createStatement().executeQuery(csvQuery);
resultSet.next();
}
```

Retrieving data from a Result Set

The above query return a result set object. We can then use the methods of the ResultSet object, such as getString(), to retrieve the desired data.

For example, use the following code in an Extract step to retrieve the value of the current Rep's Email from the Access, Excel or CSV database and assign to a field in the Data Mapper:

```
let repEmail;
repEmail = resultSet.getRow() ? resultSet.getString("Email") : "";
```

Closing JDBC Connections

It is good practice to explicitly close all connections to the database to end each database session rather than leaving the task to the Java's garbage collection.

To close the above opened connection, you should call *close()* method as follows:

Reference JDBC Basics

https://docs.oracle.com/javase/tutorial/jdbc/basics/