Setting Up The Environment

Find herein how to set up your development environment.

Additional details about how to install and configuration some of these prerequisites is available on Provisioning new environments for Fandango and Fiduciary Analytics - Windows Server Configuration

Fangango UI

The README on the UI source repo has setup instructions.

Fandango Service

I suggest using Visual Studio 2017 Professional, and these instructions will concentrate on that IDE. Most of this set up is Windows-specific.

Setting up Oracle ODBC

1. You need to download both (1) **Instant Client Package – ODBC** item as well as (2) **Instant Client Package – Basic Light** item. The installation instructions are on this page: http://www.oracle.com/technetwork/database/features/oci/odbc-ic-releasenotes-094306.html . Follow the instructions for your *operating system*
2. Unzip them both into the same location i.e. the resultants of these two extracts go into same folder. Say *C:\oracle\instantclient_12_2*
3. Add that directory to your PATH
4. You might also need to install the Visual Studio 2013 redistributable
5. Run oracle_install.exe from an elevated/admin command prompt
6. Create a subdirectoy of the path created above like *C:\oracle\instantclient_12_2\network\admin*
   1. Drop this tnsnames.ora file at this location
   2. Verify this TNS setup using command *tnsping svdev* from a command prompt, where *svdev* is the name of the database you created in the previous oracle installation step.

NOTE

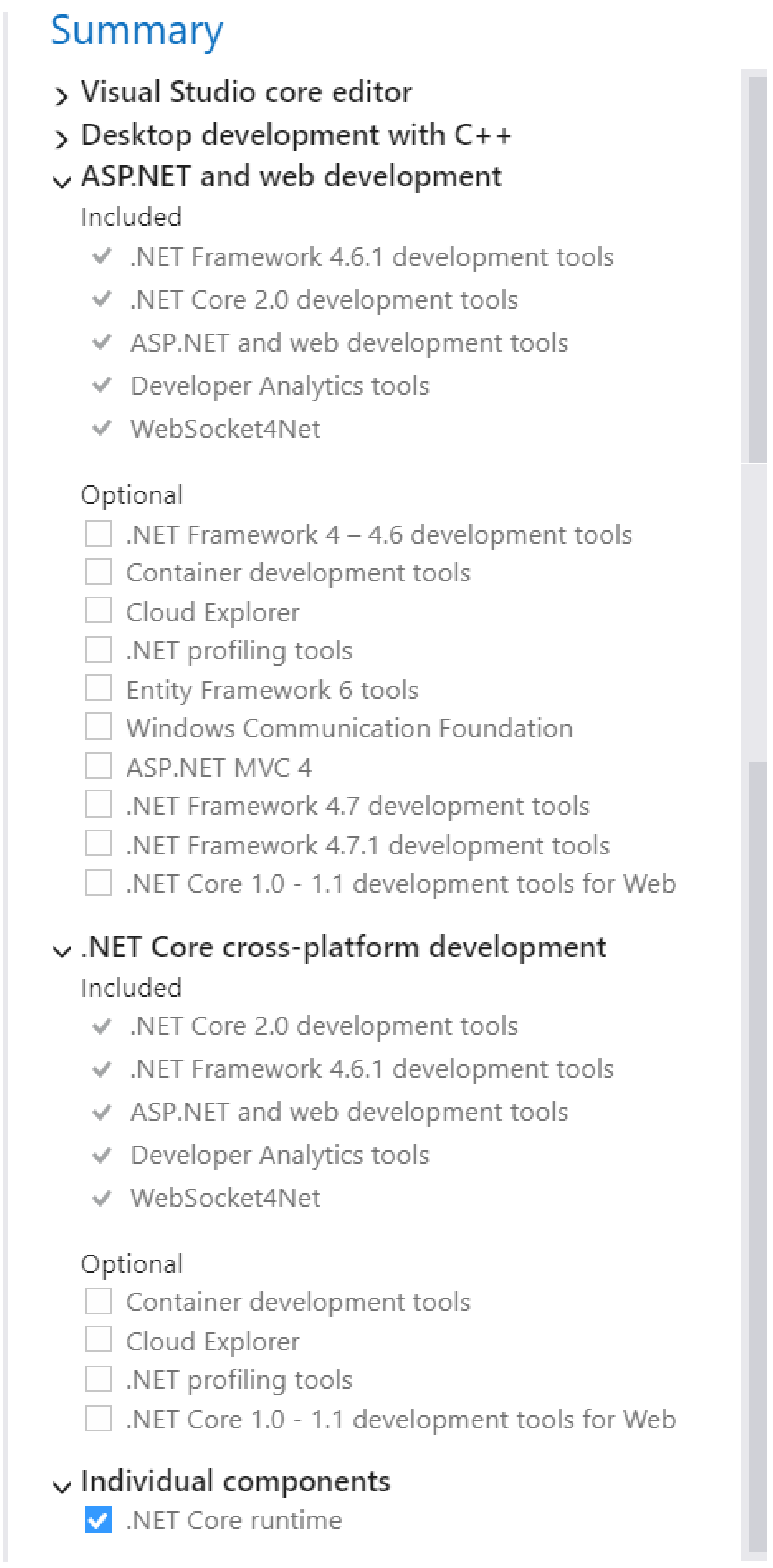
We're using a File DSN (Fandango.dsn, located at fiduciary-analytics-service\com.broadridge.adc.fiduciaryanalytics\com.broadridge.adc.fiduciaryanalytics.service), so you don't need to set up any System DSN.

There's code in Startup.cs that finds this file and with the "DefaultConnection" value in appsettings.json, creates the appropriate connection string.

You can verify the installation by opening the Data Sources (ODBC) utility (64-bit) and looking in the Drivers tab. You'll see it as "Oracle in instantclient_12_2" if you used the folder structure above.

Install .NET Core

When setting up Visual Studio 2017, you can include .NET Core with the install. Make sure at least the following components are selected:


Summary

> Visual Studio core editor

> Desktop development with C++

ASP.NET and web development

Included

- ✓ .NET Framework 4.6.1 development tools
- ✓ .NET Core 2.0 development tools
- ✓ ASP.NET and web development tools
- ✓ Developer Analytics tools
- ✓ WebSocket4Net

Optional

- ☐ .NET Framework 4 – 4.6 development tools
- ☐ Container development tools
- ☐ Cloud Explorer
- ☐ .NET profiling tools
- ☐ Entity Framework 6 tools
- ☐ Windows Communication Foundation
- ☐ ASP.NET MVC 4
- ☐ .NET Framework 4.7 development tools
- ☐ .NET Framework 4.7.1 development tools
- ☐ .NET Core 1.0 - 1.1 development tools for Web

.NET Core cross-platform development

Included

- ✓ .NET Core 2.0 development tools
- ✓ .NET Framework 4.6.1 development tools
- ✓ ASP.NET and web development tools
- ✓ Developer Analytics tools
- ✓ WebSocket4Net

Optional

- ☐ Container development tools
- ☐ Cloud Explorer
- ☐ .NET profiling tools
- ☐ .NET Core 1.0 - 1.1 development tools for Web

Individual components

- ☑ .NET Core runtime

If you're not using Visual Studio, download the appropriate SDK version here: https://github.com/dotnet/core/blob/master/release-notes/download-archives/2.0.5-download.md

Run the service

Get the latest code from Stash/Git. There are many ways of doing this, so here's how I did it, using Git for Windows: https://git-scm.com/downloads (specifically, Git Bash)

1. Create a folder close to the root (the project names are long). I used C:\Source
2. Open a bash window, or a command line.
3. You may need to set your proxy. Instructions are here: https://gist.github.com/evantoli/f8c23a37eb3558ab8765
    1. I had to call **git config --global https.sslVerify false**
4. Run **git clone https://<username>@salesvisiondev.broadridge.com/stash/scm/applicationsource/fiduciary-analytics-service.git** (replace <username> with your network login).
5. Decend into the **fiduciary-analytics-service** folder, then to the **com.broadridge.adc.fiduciaryanalytics** folder.
6. If using Visual Studio, open the com.broadridge.adc.fiduciaryanalytics.sln solution.

If you're using Visual Studio, just run the service project, and it will open a window to the canned "values" endpoint.

If not, you can descend into the **com.broadridge.adc.fiduciaryanalytics.service** folder, and run the **dotnet run** command. That will start the service and tell you what port it's listening on.

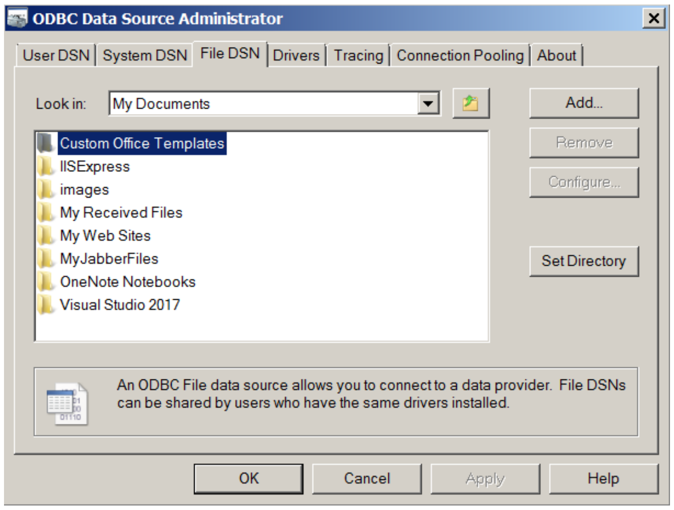
Setting the Connection String

In order to connect to a data source, we need to set it up.

Create a File DSN

Open the Data Sources (ODBC) control panel applet (64-bit). And easy way of doing that in Windows is to click Start and search for "Data Source".

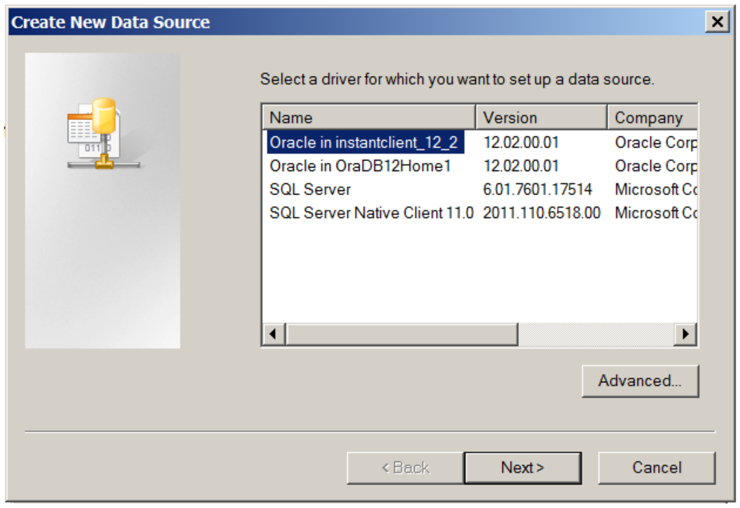
Go to the File DSN tab:



An ODBC File data source allows you to connect to a data provider. File DSNs can be shared by users who have the same drivers installed.

Click Add...

In the Create New Data Source screen, select the "Oracle in instantclient_12_2" driver (assuming you used in the steps above).



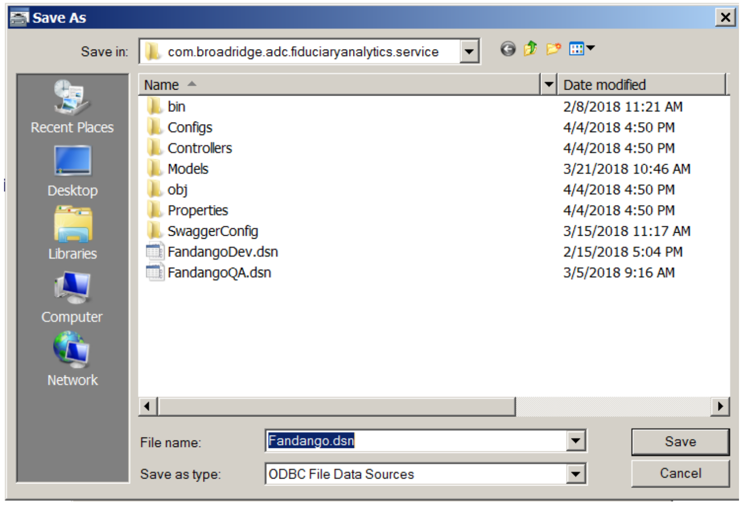
Create New Data Source

Select a driver for which you want to set up a data source.

Name	Version	Company
Oracle in instantclient_12_2	12.02.00.01	Oracle Corp
Oracle in OraDB12Home1	12.02.00.01	Oracle Corp
SQL Server	6.01.7601.17514	Microsoft Co
SQL Server Native Client 11.0	2011.110.6518.00	Microsoft Co

Click Next >

In the next screen, select or create a new file to save your DSN to. I suggest adding the file to the same directory as your service project file.



Save As

Save in: com.broadridge.adc.fiduciaryanalytics.service

Name	Date modified
bin	2/8/2018 11:21 AM
Configs	4/4/2018 4:50 PM
Controllers	4/4/2018 4:50 PM
Models	3/21/2018 10:46 AM
obj	4/4/2018 4:50 PM
Properties	4/4/2018 4:50 PM
SwaggerConfig	3/15/2018 11:17 AM
FandangoDev.dsn	2/15/2018 5:04 PM
FandangoQA.dsn	3/5/2018 9:16 AM

File name: Fandango.dsn

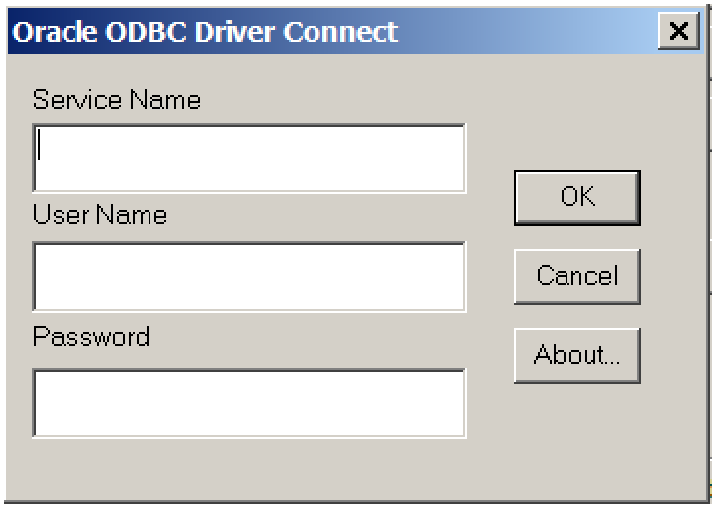
Save as type: ODBC File Data Sources

Click Save

Click Next >

Click Finish.

The Oracle ODBC Driver Connect screen displays:



Enter the SERVICE_NAME value from the TNS you entered in the TNSnames.ora file.

Enter the user name and password for a user that can access the the database.

Update appsettings.json

In the appsettings.json, update the "DefaultConnection" node with a connection string. It should have the following components:

- FILEDSN=C:\\Path\\To\\File.dsn
  - This should be the **full path** to the DSN file you saved above.
  - Be sure to escape the backslashes if you're on Windows (in other words, if your path is C:\DSN\Fandango.dsn, you should enter C:\\DSN\\Fandango.dsn.
  - If you find the DSN file in Windows Explorer, you can Shift+Right-click the file and select Copy as path to get the folder.
- UID=username
  - The same user name as you used above
- PWD=password
  - The same password as you used above

I would keep a copy of that connection string somewhere (a git stash, an external text file, whatever).

Print Generation Debugging

When working on Book/Report Generation, there are a few things you need to know:

- There is an InDesign Server instance running on the DEV server (http://10.26.12.208:951/ - see 15c Technical#15cTechnical-Deployment/DemoEnvironments for the most up-to-date info).
  - In the com.broadridge.adc.fiduciaryanalytics.data project, a Connected Service has been added using the WSDL retrieved from http://10.26.12.208:951/service?wsdl.
  - The actual URL used is stored in the appsettings.json file, under ReportGenerationSettings.InDesignServerServiceUrl and can therefore be changed for any deployment.
  - While it is possible to install a evaluation license for InDesign Server on a local development workspace, I suggest just using the one installed on the dev server.
- Two folders need to be configured for use by the services:
  - A "work folder", under which multiple folders and files will be created to house temporary files needed by InDesign Server.
  - A "packaged reports folder", under which a folder for each report will be placed, eventually to include the generated PDF and InDesign file (and its assets).
  - These folders should be cleaned up after every generation is completed via code, but if an exception is thrown anywhere along the way, the folders my persist.
  - These folders must have the exactly the same path on both the InDesign Server (i.e., 10.26.12.208) and the local development workspace.
    - On the dev server, there exists a D: drive (labeled SQL DATA), in which an InDesignServer folder was created, and in that folder the two folders mentioned above were created.
    - Then, on the local development workspace, the D: drive was mapped to the network share (i.e., \\10.26.12.208\d).
- As discussed on Software Needed for Chart Generation, Inkscape should be installed on the local development workspace. This is only needed for Chart Generation however.
  - There is a setting in appsettings.json, under ReportGenerationSettings.InkscapePath, which holds the path to the executable.
  - You must navigate to the **ChartGenScripts** folder in the service project and run **npm install** or **yarn install**.
- You will not be able to step through the JavaScript running in InDesign Server.
  - I've added a logging module (logger.js) which is included in any script run through InDesign Server.
  - Additionally, a logPath argument is passed to every script run through InDesign Server.
  - There some code that needs to be in every script:

- **Boilerplate for logging in InDesign Scripts**

```
var logPath = app.scriptArgs.getValue("logPath");
var logger = new Logger(logPath, "createDocumentAndAddPages");
// once those two lines above are included, you can use the
logger object to log messages, like:
logger.logMessage("Starting the script");
logger.logError("Error while doing something");
// see the existing scripts under InDesignScripts for more
examples.
```

    - The logs are written to the work folder where the script runs.
    - Each line is prefaced with the JavaScript timestamp (i.e., milliseconds since 1970-01-01T00:00:00.000Z). Note that due to the nature of the JavaScript runtime, messages are not necessarily written in chronological order.
- While it is theoretically possible to attach a debugger to the Node.js instance created for running the scripts needed for chart generation, I never was able to get it working.
    - I've added a logging module (logging.js) which can be included in Node scripts using the standard require syntax.

- **Example Node.js script importing logging**

```
const loggerFactory = require("./logging");
const logger = loggerFactory(`${baseFilePath}
\\chartGen_${chartName}_${Date.now()}.log`);
// once the two lines above are included, you can use the
logger, like:
logger.writeToLog("Starting the script");
logger.writeToLog("No special method for logging errors");
```

    - The logs are written to the file specified in the loggerFactory call (see code block above).
    - Each line is prefaced with the JavaScript timestamp (i.e., milliseconds since 1970-01-01T00:00:00.000Z). Note that due to the nature of the JavaScript runtime, messages are not necessarily written in chronological order.