1. The Use case

Use Case

Marketing has received a data file that may have been corrupted when it was exported. We need to extract all of the dates that have occurred in the past from this document.

**Criteria**

1. A date is defined as MMDDYYYY

a. ie: April 3, 2011 would be 03032011.

2. A past date is defined by any date that occurred before today.

a. ie: If today is 09/29/2016, all dates 08282016 or earlier would be valid.

3. The data file does not contain time information.

4. The resulting data output should be a list of dates.

**Developer Notes**

1. The corresponding data file is named MarketingDataFile.txt

2. State any assumptions you are making.

3. You can use any library in the .Net framework.

4. Write a brief explanation about your design and why you chose it.

5. Please post your solution on GitHub. We may ask questions within GitHub as a part of our code review.

a. Handy tip: Performing all of your work within a branch in the repo and creating a Pull Request will allow us to link conversations to specific lines of code.

6. We expect you may have questions about the exercise and requirements. Ask them…

1. Design

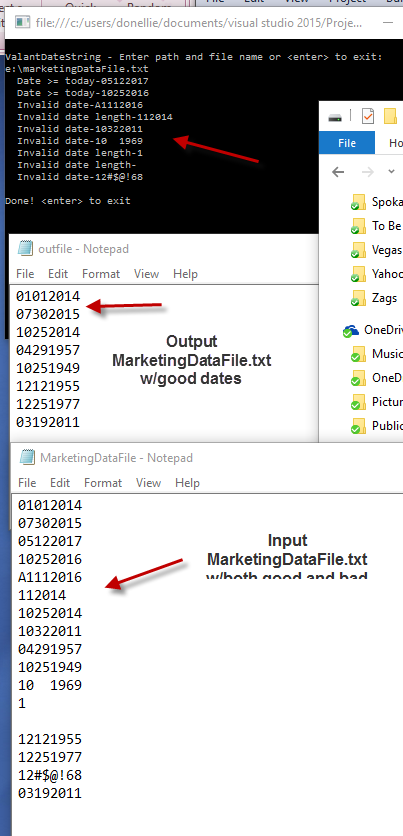
There are elements of this task that SSIS would handle nicely. Reading the input into a sql table for handling would give sorting, a cool SSRS report, etc, But I don’t have any of that loaded on my home pc – I always did my work in the office.

So I loaded .Net framework 4.6.x and Visual Studio 2015 to be used in creating a console app.

The console app –ValantDateString.exe – asks for the MarketingDataInput.txt file, reads it, and then produces a new, uncorrupted MarketingDataInput.txt file in a ..\Output subfolder under the original input.

There’s the normal checking the text (see III Assumptions below). The key instruction is the DateTime.TryParse command that does all the heavy lifting of validating the date.

Below is a screen shot of the process including the console app, the test input file I used, as well as the output file of dates.



1. Assumptions
2. All dates must be 8 chars long. Dates such as 112016 could be corrupted/truncated so the golden rule was 8 chars.
3. Today’s date is set at the start of execution. It avoids the unlikely but possible issue of midnight date rollover causing issues.
4. Output file keeps the same name as the input file MarketingDataFile.txt. But the output from the app is placed in a ..\Output sub directory.
5. Only valid prior dates are written to the output file.
6. Output file adheres to the same MMDDYYYY format as the input file.
7. No sort needed on output file.
8. Dates prior to the 01011000, i.e Dec 15, 999, are not valid. Besides, were calendars even invented then?
9. The intent of this assignment can’t be about the code – it sure is simple enough. It must have been about GitHub. I think I need to do some learning there on how to hook up GitHub w/Visual Studio, editing, etc. We used TFS at Itron for source control bug tracker, code check ins/reviews. But initially, we had a class about it….I think I better do some Youtubing on GitHub.