**Technical User Guide**

Trading Transactions Summary Report

|  |  |
| --- | --- |
| **Prepared By:** | Qaisar Khan |
| **Document Owner:** | ABN AMRO |
| **Date Issued:** | 15-Dec-2021 |
| **Document Version:** | 0.1 |
| **Document Status:** | Draft  Approved |

# Table of Contents

[1. Table of Contents 2](#_Toc90446084)

[2. Document Controls 3](#_Toc90446085)

[2.1. Revision History 3](#_Toc90446086)

[3. Introduction 3](#_Toc90446087)

[3.1. Requirements 3](#_Toc90446088)

[3.2. Document Purpose 3](#_Toc90446089)

[4. Code Directory Tree 4](#_Toc90446090)

[4.1. Tree Diagram 4](#_Toc90446091)

[5. Config File 5](#_Toc90446092)

[6. Instructions to execute the program on Mac/Linux 5](#_Toc90446093)

[6.1. Create Parent Folder 5](#_Toc90446094)

[6.2. Create Python3 Virtual Environment 5](#_Toc90446095)

[6.3. Activate the virtual environment 5](#_Toc90446096)

[6.4. Install additional python libraries 5](#_Toc90446097)

[6.5. Check the newly installed python libraries 6](#_Toc90446098)

[6.6. Create Input Directory and add input file 6](#_Toc90446099)

[6.7. Execute the Python program 7](#_Toc90446100)

[6.7.1. Run single input test file 7](#_Toc90446101)

[6.7.2. Run multiple test files in the input directory 7](#_Toc90446102)

[6.7.3. Run via shell script 7](#_Toc90446103)

[7. Running the program via command line interface 8](#_Toc90446104)

[8. Sample Output – Output.csv 9](#_Toc90446105)

[9. Log File 9](#_Toc90446106)

[9.1. Sample Output of the Log File 9](#_Toc90446107)

[10. Unit Testing 10](#_Toc90446108)

[10.1. Unit Test Case Scenario - Client Number is invalid 11](#_Toc90446109)

[10.2. Unit Test Case Scenario - Record code is invalid 12](#_Toc90446110)

[10.3. Unit Test Case Scenario - Expiry date format is invalid 13](#_Toc90446111)

[11. Assumptions 16](#_Toc90446112)

[12. Python Library for parsing fixed width file 16](#_Toc90446113)

# Document Controls

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Comments** |
| V0.01 | Qaisar Khan | 15-Dec 2021 | Initial Draft |

# Introduction

System A has produced the file Input.txt, which is a Fixed Width text file that contains the Future Transactions done by client 1234 and 4321.

## Requirements

* The Business user would like to see the Total Transaction Amount of each unique product they have done for the day.
* The Business user would like a program that can read in the Input file and generate a daily summary report.
* The Daily summary report should be in CSV format (called Output.csv) with the following specifications
* The CSV has the following Headers
  + Client\_Information
  + Product\_Information
  + Total\_Transaction\_Amount
* Client\_Information should be a combination of the CLIENT TYPE, CLIENT NUMBER, ACCOUNT NUMBER, SUBACCOUNT NUMBER fields from Input file.
* Product\_Information should be a combination of the EXCHANGE CODE, PRODUCT GROUP CODE, SYMBOL, EXPIRATION DATE.
* Total\_Transaction\_Amount should be a Net Total of the (QUANTITY LONG - QUANTITY SHORT) values for each client per product.
* Each Record in the input file represents ONE Transaction from the client for a particular product. Please focus on code re-usability. Code can be either Java, Python, or Perl

## Document Purpose

The purpose of this document is to provide:

* Instructions on how to run the software and any troubleshooting
* Complete the code with unit tests
* Generate Log file
* Provide the output file (Output.csv)

# Code Directory Tree

## Tree Diagram

A picture containing timeline

Description automatically generated

# Config File

Config file is used in this program for creating:

* Output directory
* Log File configuration
* Process ID creation

# Instructions to execute the program on Mac/Linux

## Create Parent Folder

mkdir trade\_transactions

## Create Python3 Virtual Environment

Assuming the computer running the program has Python3 installed and is able to run the command to create virtual environment

python3 -m venv env

## Activate the virtual environment

Text

Description automatically generated

## Install additional python libraries

Text

Description automatically generated

## Check the newly installed python libraries

Text

Description automatically generated with low confidence

## Create Input Directory and add input file

Although the input directory is already created and has the test input file. Instruction is written only for the completion of the document.

(env) qaisar@Qaisars-Mini trade\_transactions % l

total 32

drwxr-xr-x 6 qaisar staff 192 15 Dec 02:31 env

-rw-r--r--@ 1 qaisar staff 2009 15 Dec 04:20 cli.py

-rwxr-xr-x 1 qaisar staff 79 15 Dec 04:21 run.sh

drwxr-xr-x 7 qaisar staff 224 15 Dec 04:25 src

drwxr-xr-x 3 qaisar staff 96 15 Dec 05:25 logs

drwxr-xr-x 4 qaisar staff 128 15 Dec 05:25 output

-rw-r--r-- 1 qaisar staff 3001 15 Dec 05:39 test\_trading\_transactions.py

drwxr-xr-x 3 qaisar staff 96 15 Dec 05:39 input

-rw-r--r-- 1 qaisar staff 27 15 Dec 05:41 .gitignore

drwxr-xr-x 13 qaisar staff 416 15 Dec 05:41 .

drwxr-xr-x 12 qaisar staff 384 15 Dec 05:41 .git

drwxr-xr-x 11 qaisar staff 352 15 Dec 05:48 ..

drwxr-xr-x 4 qaisar staff 128 15 Dec 06:07 docs

## Execute the Python program

### Run single input test file

python cli.py --dir input -i Input.txt -o Output.csv

### Run multiple test files in the input directory

python cli.py --dir input

Note: Although it was not required, the test program has the ability to read multiple input test files and generate output.

**Assumption:**

Since the file name specification is not provided, the program will assume that all the input test files in the directory has the same format.

### Run via shell script

Shell script is provided to easily run the code and to remember the command

./run.sh

Text

Description automatically generated

# Running the program via command line interface

Text

Description automatically generated

# Sample Output – Output.csv

|  |  |  |
| --- | --- | --- |
| **Client\_Information** | **Product\_Information** | **Total\_Transaction\_Amount** |
| SGX\_FU\_NK\_20100910 | CL\_4321\_2\_1 | 46 |
| SGX\_FU\_NK\_20100910 | CL\_1234\_2\_1 | -52 |
| CME\_FU\_N1\_20100910 | CL\_4321\_3\_1 | -79 |
| CME\_FU\_N1\_20100910 | CL\_1234\_3\_1 | 285 |
| CME\_FU\_NK.\_20100910 | CL\_1234\_3\_1 | -215 |

# Log File

* When the python program is run, the log file is generated by the program.
* Name of the log file is: trade\_transactions\_summary\_report\_log.txt

## Sample Output of the Log File

A picture containing text, curtain

Description automatically generated

# Unit Testing

* Due to time constraint and considering no validation rules were provided, only basic level of unit tests are written for this program.
* There is a lot of capacity of adding more test cases and unit tests should increase in number in a real production environment.
* Considering it is a technical test, I am limited the test cases to only 4.

Graphical user interface, text

Description automatically generated

## Unit Test Case Scenario - Client Number is invalid

As per the requirement given, the file should have only the client number: 1234 and 4321. Any other client number will be tagged as invalid and prompt the user.

315CL 432100020001SGXDC FUSGX NK 20100910JPY01B 0000000001 0000000000000000000060DUSD000000000030DUSD000000000000DJPY201008200012100 687844000092600000000 O

315CL 332100020001SGXDC FUSGX NK 20100910JPY01B 0000000001 0000000000000000000060DUSD000000000030DUSD000000000000DJPY201008200012260 687950000092550000000 O

315CL 432100020001SGXDC FUSGX NK 20100910JPY01B 0000000001 0000000000000000000060DUSD000000000030DUSD000000000000DJPY201008200012280 687956000092550000000 O

Text

Description automatically generated

(env) qaisar@Qaisars-Mini trade\_transactions % python test\_trading\_transactions.py

F...

======================================================================

FAIL: testClientNumber (\_\_main\_\_.TestTradingTransactions)

----------------------------------------------------------------------

Traceback (most recent call last):

File "/Users/qaisar/Documents/ABN\_AMRO\_TECHNICAL\_TEST/trade\_transactions/test\_trading\_transactions.py", line 33, in testClientNumber

self.assertTrue(observed in expected\_client, f"""Invalid client number found in the file: {observed}. Expected clients should be one the numbers provided in the list: {expected\_client} - Line Number: {index+1}""")

AssertionError: False is not true : Invalid client number found in the file: 3321. Expected clients should be one the numbers provided in the list: [1234, 4321] - Line Number: 704

----------------------------------------------------------------------

Ran 4 tests in 0.135s

FAILED (failures=1)

## Unit Test Case Scenario - Record code is invalid

315CL 123400030001FCC FUCME N1 20100910JPY01S 0000000000 0000000005000000000000DUSD000000000025DUSD000000000000DJPY20100819059481 000314000091950000000 O

315CL 123400030001FCC FUCME N1 20100910JPY01S 0000000000 0000000005000000000000DUSD000000000025DUSD000000000000DJPY20100819059482 000315000092600000000 O

315CL 123400030001FCC FUCME N1 20100910JPY01S 0000000000 0000000005000000000000DUSD000000000025DUSD000000000000DJPY20100819059483 000316000092600000000 O

315CL 123400030001FCC FUCME N1 20100910JPY01S 0000000000 0000000005000000000000DUSD000000000025DUSD000000000000DJPY20100819059618 000451000092100000000 O

315CL 123400030001FCC FUCME N1 20100910JPY01S 0000000000 0000000005000000000000DUSD000000000025DUSD000000000000DJPY20100819059619 000452000092100000000 O

215CL 123400030001FCC FUCME N1 20100910JPY01S 0000000000 0000000006000000000000DUSD000000000030DUSD000000000000DJPY20100819059484 000317000092550000000 O

315CL 123400030001FCC FUCME N1 20100910JPY01S 0000000000 0000000006000000000000DUSD000000000030DUSD000000000000DJPY20100819059620 000453000091950000000 O

315CL 123400030001FCC FUCME N1 20100910JPY01S 0000000000 0000000007000000000000DUSD000000000035DUSD000000000000DJPY20100819059485 000318000092550000000 O

Text

Description automatically generated

(env) qaisar@Qaisars-Mini trade\_transactions % python test\_trading\_transactions.py

..F.

======================================================================

FAIL: testRecordCode (\_\_main\_\_.TestTradingTransactions)

----------------------------------------------------------------------

Traceback (most recent call last):

File "/Users/qaisar/Documents/ABN\_AMRO\_TECHNICAL\_TEST/trade\_transactions/test\_trading\_transactions.py", line 43, in testRecordCode

self.assertEqual(observed, expected, f"""First 3 characters of the record code are not correct. Expected code: {expected} - Code Found: {observed} - Line Number: {index+1}""")

AssertionError: 215 != 315 : First 3 characters of the record code are not correct. Expected code: 315 - Code Found: 215 - Line Number: 659

----------------------------------------------------------------------

Ran 4 tests in 0.161s

FAILED (failures=1)

## Unit Test Case Scenario - Expiry date format is invalid

315CL 123400030001FCC FUCME NK. 20100910USD01S 0000000000 0000000010000000000000DUSD000000000050DUSD000000000000DUSD20100819059923 000756000092350000000 O

315CL 123400030001FCC FUCME NK. 20100910USD01S 0000000000 0000000012000000000000DUSD000000000060DUSD000000000000DUSD20100819059924 000757000091800000000 O

315CL 123400030001FCC FUCME NK. 20100910USD01S 0000000000 0000000012000000000000DUSD000000000060DUSD000000000000DUSD20100819059925 000758000092700000000 O

315CL 123400030001FCC FUCME N1 20100910JPY01B 0000000001 0000000000000000000000DUSD000000000005DUSD000000000000DJPY20100819059435 000268000092600000000 O

315CL 123400030001FCC FUCME N1 20100910JPY01B 0000000001 0000000000000000000000DUSD000000000005DUSD000000000000DJPY20100819059436 000269000092600000000 O

315CL 123400030001FCC FUCME N1 0100910JPY01B 0000000001 0000000000000000000000DUSD000000000005DUSD000000000000DJPY20100819059437 000270000092600000000 O

315CL 123400030001FCC FUCME N1 20100910JPY01B 0000000001 0000000000000000000000DUSD000000000005DUSD000000000000DJPY20100819059438 000271000092650000000 O

315CL 123400030001FCC FUCME N1 20100910JPY01B 0000000001 0000000000000000000000DUSD000000000005DUSD000000000000DJPY20100819059439 000272000092650000000 O

315CL 123400030001FCC FUCME N1 20100910JPY01B 0000000001 0000000000000000000000DUSD000000000005DUSD000000000000DJPY20100819059440 000273000092650000000 O

Text

Description automatically generated

(env) qaisar@Qaisars-Mini trade\_transactions % python test\_trading\_transactions.py

.E.F

======================================================================

ERROR: testExpirationDate (\_\_main\_\_.TestTradingTransactions)

----------------------------------------------------------------------

Traceback (most recent call last):

File "/Users/qaisar/Documents/ABN\_AMRO\_TECHNICAL\_TEST/trade\_transactions/src/common/utils.py", line 38, in check\_date\_format

datetime.strptime(date\_text, '%Y%m%d')

File "/usr/local/Cellar/python@3.9/3.9.7\_1/Frameworks/Python.framework/Versions/3.9/lib/python3.9/\_strptime.py", line 568, in \_strptime\_datetime

tt, fraction, gmtoff\_fraction = \_strptime(data\_string, format)

File "/usr/local/Cellar/python@3.9/3.9.7\_1/Frameworks/Python.framework/Versions/3.9/lib/python3.9/\_strptime.py", line 352, in \_strptime

raise ValueError("unconverted data remains: %s" %

ValueError: unconverted data remains: J

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

File "/Users/qaisar/Documents/ABN\_AMRO\_TECHNICAL\_TEST/trade\_transactions/test\_trading\_transactions.py", line 53, in testExpirationDate

self.assertTrue(check\_date\_format(expiration\_date, index+1), f"""Expiration date format is not CCYYMMDD. Date format found: {expiration\_date} - Line Number: {index+1}""")

File "/Users/qaisar/Documents/ABN\_AMRO\_TECHNICAL\_TEST/trade\_transactions/src/common/utils.py", line 42, in check\_date\_format

raise ValueError(f"""Incorrect data format, should be YYYYMMDD and not {date\_text} - Line Number: {line\_number}""")

ValueError: Incorrect data format, should be YYYYMMDD and not 0100910J - Line Number: 480

======================================================================

FAIL: testRowLength (\_\_main\_\_.TestTradingTransactions)

----------------------------------------------------------------------

Traceback (most recent call last):

File "/Users/qaisar/Documents/ABN\_AMRO\_TECHNICAL\_TEST/trade\_transactions/test\_trading\_transactions.py", line 23, in testRowLength

self.assertEqual(observed, expected, f"""File should have 176 characters excluding new line. Expected Length: {expected} - Actual Length: {observed}""")

AssertionError: 175 != 176 : File should have 176 characters excluding new line. Expected Length: 176 - Actual Length: 175

----------------------------------------------------------------------

Ran 4 tests in 0.141s

FAILED (failures=1, errors=1)

(env) qaisar@Qaisars-Mini trade\_transactions %

# Assumptions

Following assumptions were taken, while writing this program:

* Total Net Amount in the output file is converted to two decimal points.
* Since no validation rules were given, test scenarios were created with assumed validation rules.

# Python Library for parsing fixed width file

Panda fixed width library is used to parse the input file. Although there were other ways to parse the file, it is always encouraged to use the python well reputed libraries.

<https://towardsdatascience.com/parsing-fixed-width-text-files-with-pandas-f1db8f737276>