**User Guide**

Census Bureau Name and Address Parsing Project in Python, Version 0.0.1

Contents

[Project Overview 3](#_Toc99407053)

[Code Repository 3](#_Toc99407054)

[Running the Program 3](#_Toc99407055)

[Test Data 3](#_Toc99407056)

[Logic Overview 4](#_Toc99407057)

[US Address Tokenization 4](#_Toc99407058)

# Project Overview

The US Address Data Preparation Function is designed to parse an unstructured address string into a set of Address Components. The address components produced by the US Address Data Preparation Function can be used as item values in HiPER indices, Boolean rules, and scoring rules. The implementation of the US Address Data Preparation Function and other Data Preparation functions require the definition of new XML elements in the Entities Script to invoke these functions.

## Code Repository

The Census Bureau Address Parsing is written completely in Python. The code has been developed using the Spyder Development Environment. The most current version of the code can be found on GitHub under the name CensusBureauNameAddress.

<https://github.com/Univ-Of-Arkansas-at-LITTLE-ROCK/CensusBureauNameAddress>

## Running the Program

There are 2 phases of the program; the first is Address and Name Tokenization and Parsing phase and next is Exception Handling Phase, both of these phases have different approach, likewise there are two python programs, one is Address Parser File and Exception Handler file.

**Name and Address Tokenization and Parsing Phase** focuses on taking a file which has ID | Address Name i.e., Pipe Delimited file, next the program will first split the address and ID (one by one) and clean the addresses like special characters in the address with the exception of comma and #, rest all the characters in the addresses are treated as delimiters.

**Exception Handling Phase** focuses on generating the dictionaries as much as possible from the user through a CLI, this process is not automated as the knowledge of human is taken into consideration.

# Test Data

We have tested our program by giving many files as input but the most common files we chosen are S4G.txt files and AddressResv.txt files, which are pipe delimited with ID and Address, the main focus of our program is to generate as much of Mask to Dictionaries, and that too by ensuring Data Quality.

**Sample Data Set**

2|Dean. Alex Abadi , Jr, QA, 14646 RANCHERO RD, HESPERIA ,CA 92345

3|Dr. Lyla Abbatiello , Jr, CLIA, 517 N MOUNTAIN AVE, # 202, UPLAND ,CA 91786

4|Mr. Kylia Abbott , Jr, QC, 2731 PONKAN MEADOW DR, APOPKA ,FL 32712

# Logic Overview

The Activity diagram in Figure 1 shows the basic flow of the first program i.e., Address and Name Parsing and Converting it in the final Output based on the Pre-Entered Address Mappings. Same is for Name Parsing



Figure 1: Address Parsing First Step

The Second step i.e., taking care of exceptions through user involvement is shown in a sequence diagram but before that let us understand the files involved and its functions.

**AddressParser0.0.1.py file** generally is responsible for address tokenization and comparison of its generated masks with the masks that has been already been generated by the user (in the second phase), this program runs on its own and doesn’t require any user involvement in between the runtime, as its role is to simply clean the addresses, compare masks an dump in the final output

## US Name Address Tokenization

All letters are converted to upper case. During the tokenization, the addresses and Name are split into tokens like for an example

**Sample Address** Dr. John Talburt, Jr, IQCP, 123-1/2 N. Oak Street, Apt 3A, Little Rock, ARK 72203-4352

Mask:- “NDWF,SN,WW,TN” Address Table

|  |  |  |
| --- | --- | --- |
| Pos | Token | Code |
| 1 | 123-1/2 | N |
| 2 | N | D |
| 3 | OAK | W |
| 4 | STREET | F |
| 5 | APT | S |
| 6 | 3A | N |
| 7 | LITTLE | W |
| 8 | ROCK | W |
| 9 | ARK | T |
| 10 | 72203-4352 | N |

Mask: -“PGL,J,Q”

|  |  |  |
| --- | --- | --- |
| Pos | Token | Code |
| 1 | DR | P |
| 2 | JOHN | G |
| 3 | TABLURT | L |
| 4 | JR | J |
| 5 | IQCP | Q |

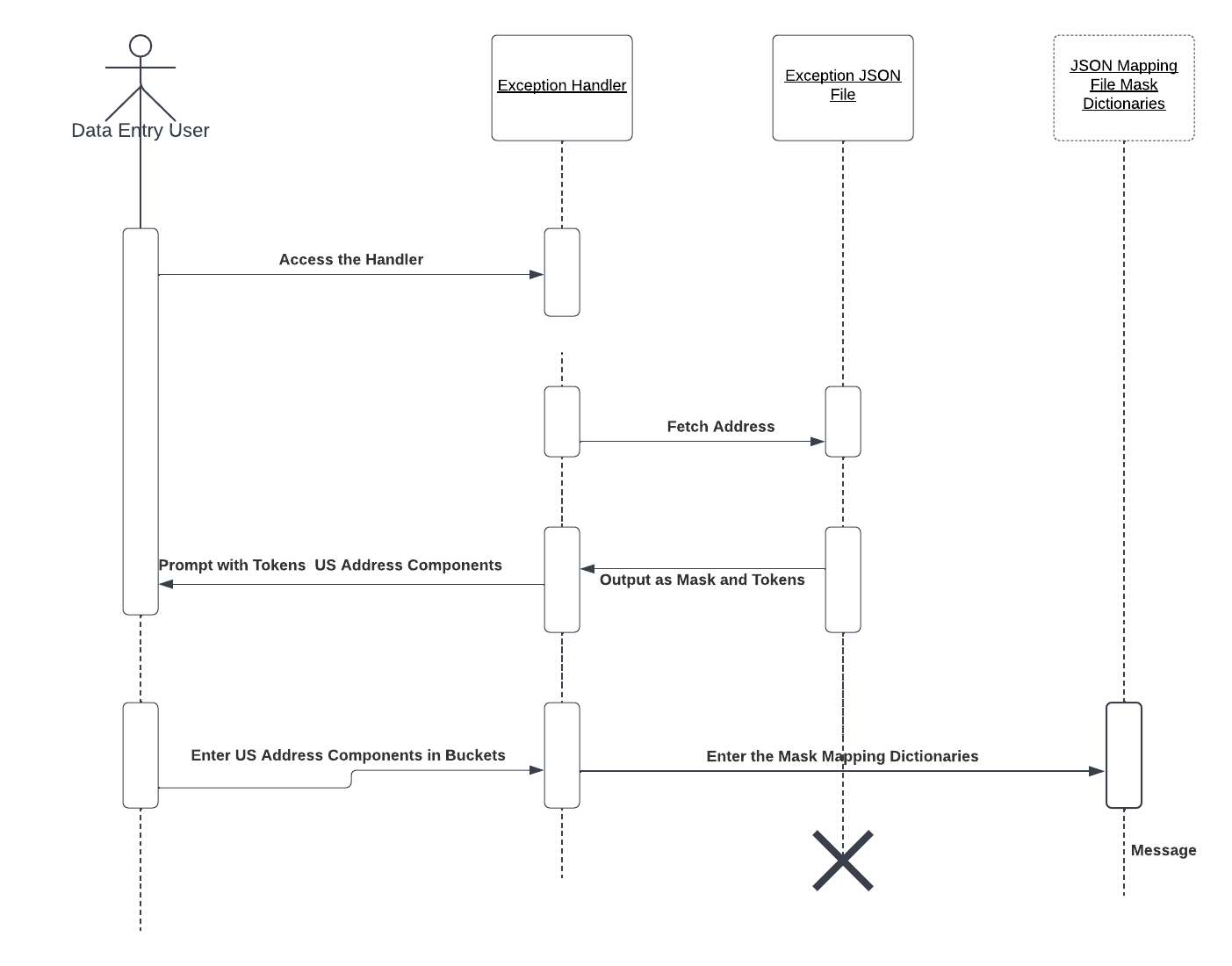
**ExceptionHandler.py file** this file is to take the Exception-File of JSON format and ask the user to give the Address Components with respect to the tokens displayed one by one and store it in a Mapping Dictionary like given below, by preserving the comma.

NDWF,SN,WW,TN @USAD\_SNO(1)|@USAD\_SPR(2)|@USAD\_SNM(3)|

@USAD\_SFX(4)|@USAD\_ANM(5)|@USAD\_ANO(6)|

@USAD\_CTY(7, 8)|@USAD\_STA(9)|@USAD\_ZIP(10)

|  |  |
| --- | --- |
| Component Assignments | |
| Comp Code | Value Assigned |
| @USAD\_SNO | 123-1/2 | @USAD\_ZIP | 72203-4352 |
| @USAD\_SPR | D | @USAD\_ZP4 |  |
| @USAD\_SNM | OAK | @USAD\_BNM |  |
| @USAD\_SFX | STREET | @USAD\_BNO |  |
| @USAD\_SPT |  | @USAD\_RNM |  |
| @USAD\_ANM | APT | @USAD\_RNO |  |
| @USAD\_ANO | 3A |
| @USAD\_CTY | LITTLE ROCK |
| @USAD\_STA | ARK |

The Depiction of the second Phase (Address) It will be same for the Name as well as shown in a form of sequence diagram

The diagram shows that how user is given the address and their tokens one by one and how the user is going to enter the US Address Components against the addresses he perceives, and the Mask to Mapping File is been affected as well.

Now the Overall Scenario in which the address and names are parsed is shown below in the sequence diagram as,

