**Creating a MARATHON Demand File**

Demand records are one of the inputs required to run MARATHON simulations. This Executable Jar File was developed to automate the process of creating the demand files used by MARATHON. This Executable Jar File produces the final Demand file by utilizing data from the Vignette mapping, Vignette consolidated, and FORGE output files. This Executable program requires the input files to be specifically formatted in order to properly run. This tutorial provides step-by-step instructions on how to run the program from start to finish. Follow the steps presented below to learn how to create the final Demand Record using FORGE and the Vignette consolidated data input files.

**Inputs**

Demand Records require multiple input files. The required input files are the Vignette Consolidated, Vignette Mapping, and any number of optional FORGE files. There should be a FORGE file for every scenario listed in the MAP file. All input files need to be exported as tab delimited (.txt) files. (How to export as Tab Delimited (.txt) section fill follow the Inputs Section).

**Vignette Mapping File**

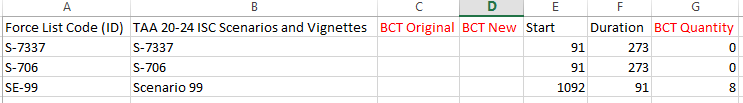
The Vignette Mapping file is a top level file that determines which events will be included in the model. Events not included in the map will not be included in the demand file. If events need to be added, removed, or modified, then the MAP file should be changed to reflect what should be included in the model.

The filename of the mapping file should include MAP\_ at the beginning of the name.

The ordering of the column in the file needs to match exactly with the expected ordering. The names of the columns do not need to match. This might include keeping/adding empty or depreciated columns in the input file.

Example of correctly formatted MAP file:





\*The columns in red are not used and can be empty, however, the spacing of the columns needs to be the same (Event title in column B and Start day in column E, etc.).

**Vignette Consolidated File**

This file Includes data about vignettes that are included in the model.

For each non-scenario event in the map, the natural join of the event timing data with data from this file will create rows in the demand file.

If an event is included in the consolidated file but not the map, it will not be included in the model. If an event is included in the map but not the consolidated file, it will not be included in the model.

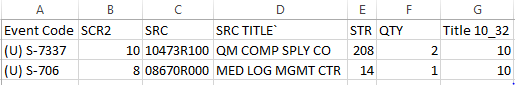
This file need to be exported as a tab delimited (.txt) file (typically this file is given as an excel workbook).

The filename of the consolidated file needs to include CONSOLIDATED\_ as a prefix. The event codes in the map and consolidated file need to match exactly.

The ordering of the column in the file needs to match exactly with the expected ordering. The names of the columns do not need to match. This might include keeping/adding empty or depreciated columns in the input file.

Example of correctly formatted CONSOLIDATED file:





**FORGE Files**

Each FORGE file contains data about scenarios that are included in the model. Each scenario listed in the MAP has a separate FORGE file.

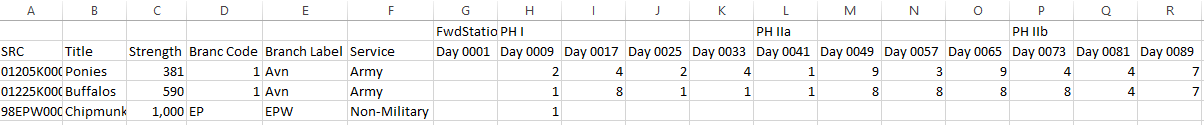
For each scenario in the map, the FORGE the data are joined with the MAP data. FORGE files contain their own timing data starting from day 0 (FORGE time). All FORGE time values have the start day offset from the MAP added to them.

All FORGRE files need to be exported as tab delimited (.txt) files. Each FORGE file should have the naming convention FORGE\_[event code], where event code is the name of the scenario that matches what is found in the map file. All Scenarios listed in the MAP file need to have a corresponding FORGE file. An error window will show if one is missing.

Each FORGE file has two headers that are used to derive information. The first header is the list of phases, where the phase name is directly above the time period the phase starts. The second header has 6 columns of data, followed by any number of cells for quantity by time period.

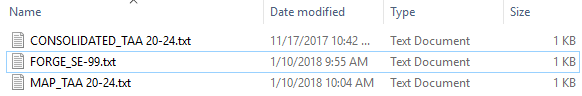
Example of correctly formatted FORGE File (one for each scenario):





Once all the inputs have been checked (primary keys [event code/force code id] match exactly between files) and correctly formatted, they should all be moved to a single directory.

Example of input directory:



**Tutorial for converting Excel to Tab Delimited (.txt)**

In Excel, go to file, save-as, then browse. Follow the naming conventions listed in the previous section for each input file. On the save as type drop down menu, select the option “Text (Tab delimited) (\*.txt)”



Alternatively, in Excel, select all the cells and copy the data (ctrl+a, ctrl+c). Open Notepad or any other text editor and paste the data (ctrl+v). Then save the file, following the correct naming convention.

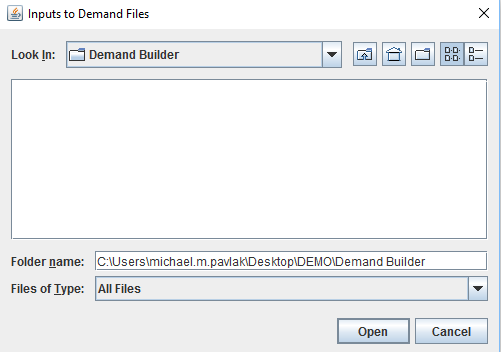
Excel can sometimes encounter errors when exporting to a .txt file. Verify the data was correctly exported by opening in a text editor and check that the file size of the .txt file is smaller than the Excel file.

**Running the Script**

The script is run by opening the jar file.



Once opened, a file select menu will ask the user to locate in directory contain in input file.

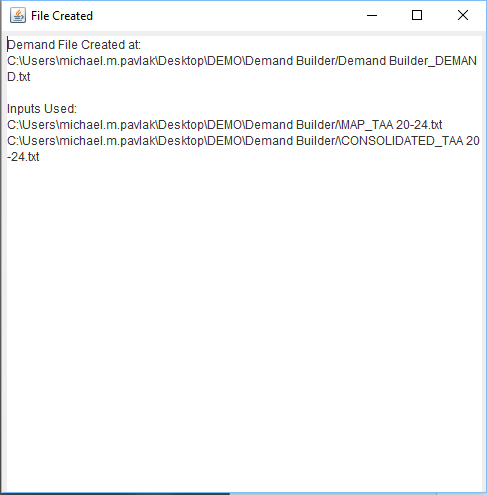


Once at the correct location, click Open to run.

If any of the inputs have been incorrectly formatted, an error message should appear describing the problem.

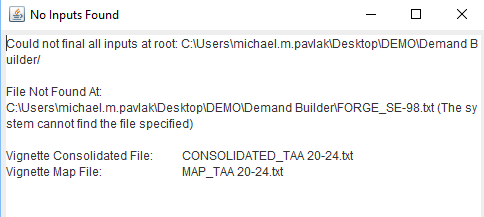
If no errors occur, a window should appear confirming that a file was created.

Example of file creation confirmation screen:



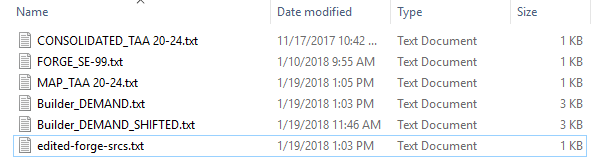
The FORGE files will not be included under inputs used because they are assumed to exist. If a required FORGE file was not found, a separate error window will be seen.

Example of missing FORGE:



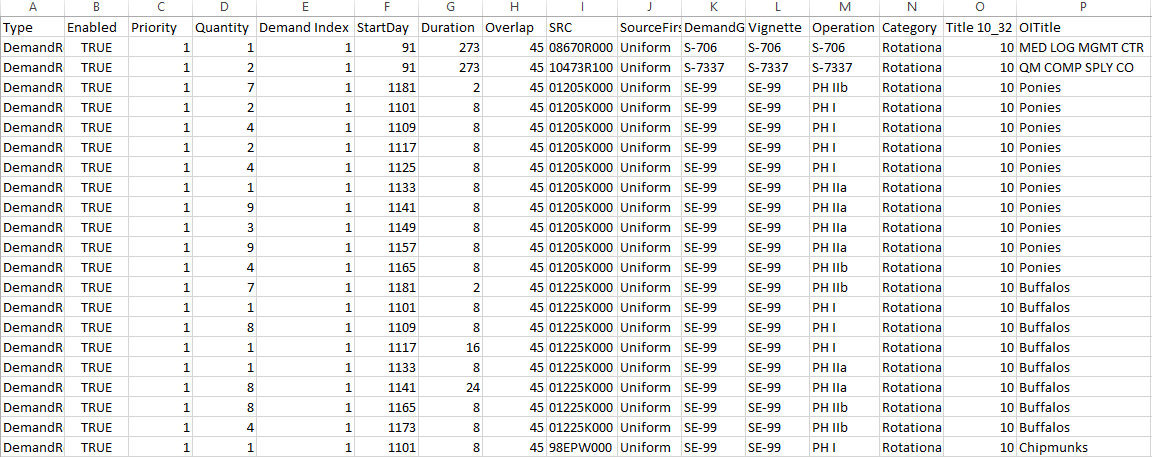
A new file with the suffix \_DEMAND.txt will be created in the same directory as the input file. If there was a mismatch between a scenario ending time in the MAP file and FORGE file, then there will be an additional file edited-forge-srcs.txt detailing the records that needed to be changed. The edits file will contain only the records that needed to be adjusted to match the MAP. The records in this file will also be included in the \_DEMAND file.

Example of directory with output files:



Example of DEMAND file:





Example of edited-forge-srcs file:





**Debugging**

Errors encountered when reading the input files should create a pop up window detailing the cause of the error.

Other error messages might not be displayed when running the program from the jar file. If there is no windows confirming a file was created and no error windows appeared, then there was an error.

These error messages are displayed to standard output (System.out default in java). To see any error message, the jar has to be run using a terminal prompt (powershell/cmd on Windows). To run from terminal use the command:

“java -jar [path to jar file]”, and all output message will be displayed in the terminal.

Error message will also be displayed when running the functions directly from the Clojure environment.