# Data File Masker V2 Instruction Manual

**Summary:** The Data File Masker (DFM) is a program that masks data files of any type (jpg, pdf, txt, etc.) to the exact same size, name, and type. The primary purpose of this program is to mask data files so consumer privacy is protected when working with the data. There are currently **six** options to mask files:

1. Single File Mask:
2. Directory of Files Mask:
3. CSV File Metadata Mask
4. CSV File Metadata (DIR) Mask
5. **Oracle File Metadata Mask**
6. **Oracle File Metadata Mask using repository**

Each option has its specific use case and can be only used in certain situations.

**Note:** Option five listed above will mask the file using binary 1s. The good thing about this option is it creates a file of the exact same size, type, and name. However, when you open the file it will open to just 1s. Option six listed above does almost the same thing however instead of using 1s to create the masked file it uses a repository of non-sensitive documents to use in place of the sensitive documents. The good thing about this option is you can open the file and it shows some relevant data however the exact size of the real file compared to the new “masked” file will be off.

## How DFM Works

The DFM algorithm is simple yet complex. It relies on a set of “fill files” which hold a set number of bytes which are 4 megabytes (mb), 1mb, 1 kilobyte (kb), and 1 byte. A byte is represented by a binary “1” and each masked file is filled with the appropriate amount of bytes. In java, simply writing singular bytes takes a while so having a “fill file” with an exact amount of bytes speeds up the process tremendously. The main masking algorithm calculation is made as follows:

1. FileLengthInBytes \ 4mb = numOf4mbToFill
2. X = 4mb \* qBytesToFill (qBytesToFill is the integer value of numOf4mbToFill so if essentially 1.5 would be 1 as an int)
3. takeQBytes = FileLengthInBytes – X (The amount of qBytes needed )
4. numOfMbToFill = takeQBytes \ 1mb
5. mbToFill = numOfMbFill (Which is the integer value of numOfMbFill)
6. leftOverMb = numOfMbFill – mbToFill
7. kbToFill = leftOverMb \* 1kb
8. kbOutFill = kbToFill (integer value of kbToFill)
9. byteToFill = kbToFill = kbOutFill
10. bytesFill = byteToFill \* 1024

The integer value of each double is used in to calculate the exact number of fill files to be applied all the way to byte level. This code is located in the **Mask.java** file

### Masked Output

The output of the DFM aims to create exactly how the file storage system would look on the database. This means that every path that is identified in the table is created in the output and every path will hold that file or set of files. The output file can be changed in the **OutputFileConfig.ini.** The files are opened by their respective program however in reality all the masked files hold are ones.

## Program Setup

The DFM has many configuration files which allow for user customization in terms of connecting to Oracle Database, Output Mask folder, and Oracle Table Compatibility.

The **DbConnect** folder contains the **dbConnect.ini** file which contains the information for program to dictate how to connect to the Oracle Database and how many rows to mask. The **README** in the **DbConnect** directory contains the information to setup and customize the file. The **ColumnConfig.ini** is **important** if the user wishes to use the program on multiple different tables. DFM uses **FOUR** fundamental attributes when masking a file: File **name, path, size, and type.** The **ColumnConfig.ini** allows you to map those attributes to the **corresponding** column name in the desired table to be masked. More information regarding the **ColumnConfig.ini**  is located in the **README**.

The **Instructions** folder contains the text documents that are used to show the user how to generally use their selected option. These can be changed for further understanding at any point

The **MaskFillFiles** folder is a folder that **DOES NOT** require changing. These files are used to write bytes to the current masked file. Please see **How DFM Works** for a more in-depth description of these files.

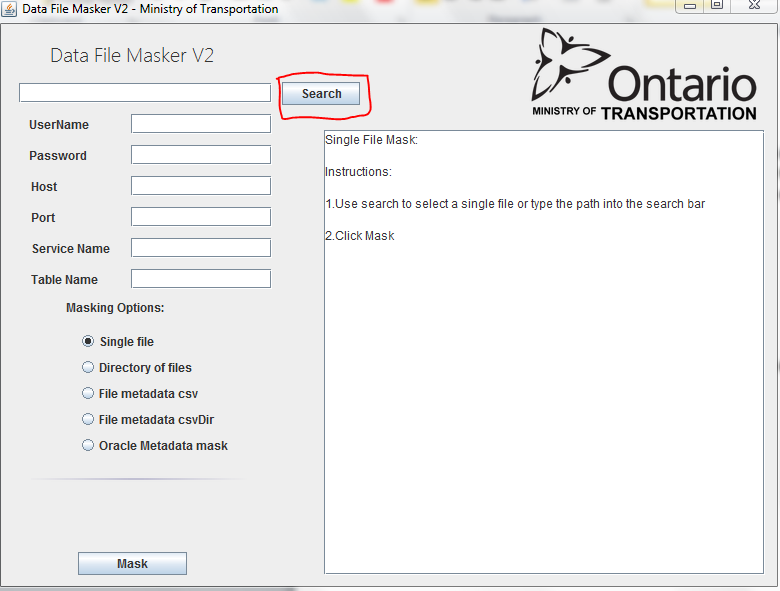
The **OutputFileConfig** folder holds the **OutputFileConfig.ini** file which is used for the program dictate the folder where the masked files are to be created. In-depth instructions on how to use the file are found in the folder.

The **OutputfileConfig** folder also holds the **IBM\_TYPES .ini** and **Extensions.ini.** These files are very important and enable the program to be essentially compatible with any extension. They work in parallel (they create a 2d array in java) meaning if the first line is **RICHTEXT** in the **IBM\_TYPES.ini** then the first line in the **Extensions.ini** will be **rtf.** Lastly to add extensions simply match the IBM type name with the extension on the **same** line.

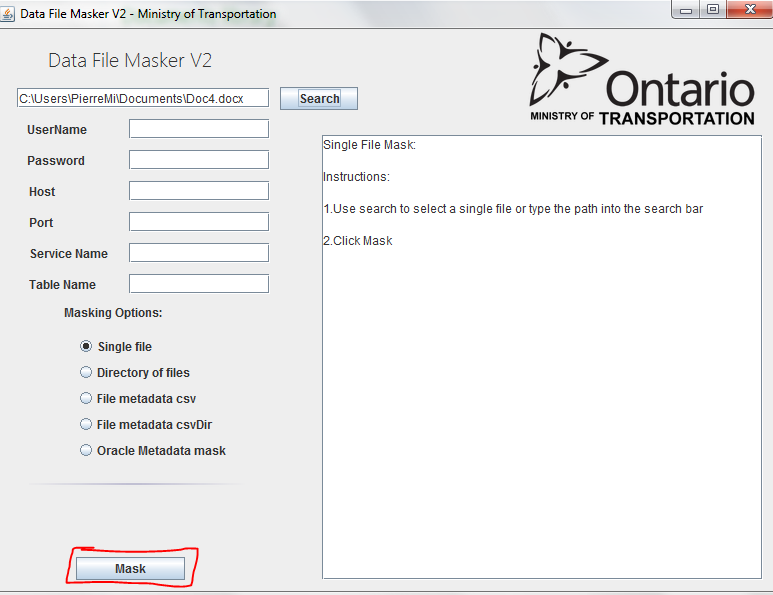
## Single File Mask

This method simply masks one file.

Step 1: Click search or type the exact PATH to the file

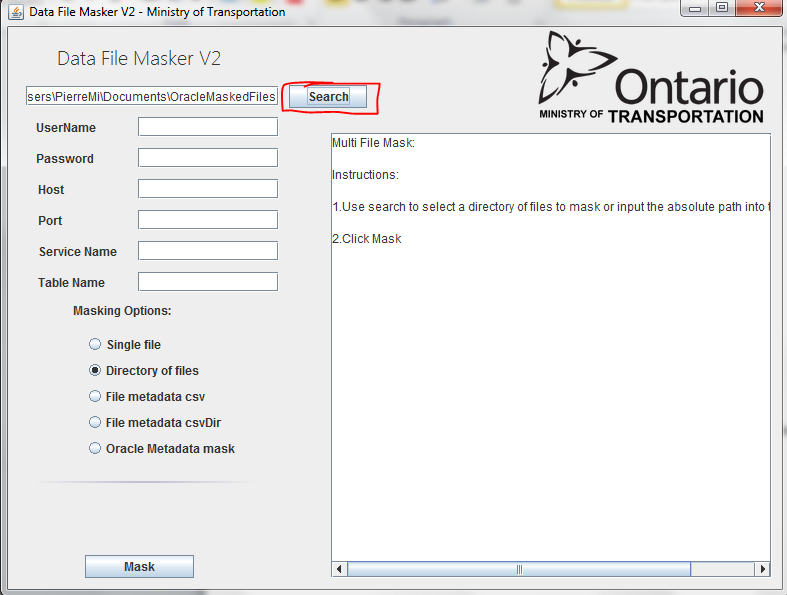


Step 2: Click Mask and the file will be masked (The output directory will be opened when finished)

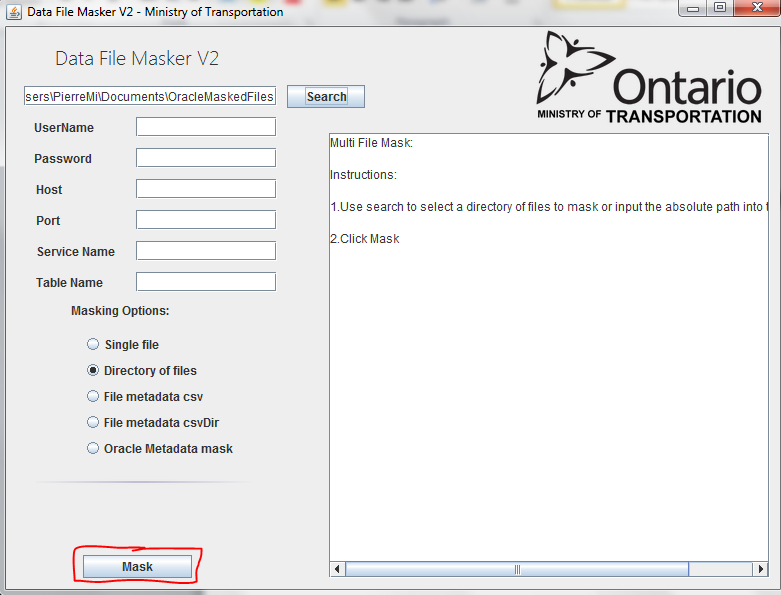


## Directory of Files Mask

Step 1: Click search or type in the exact path to the directory (Have a premade directory with just individual files to mask (no subfolders))

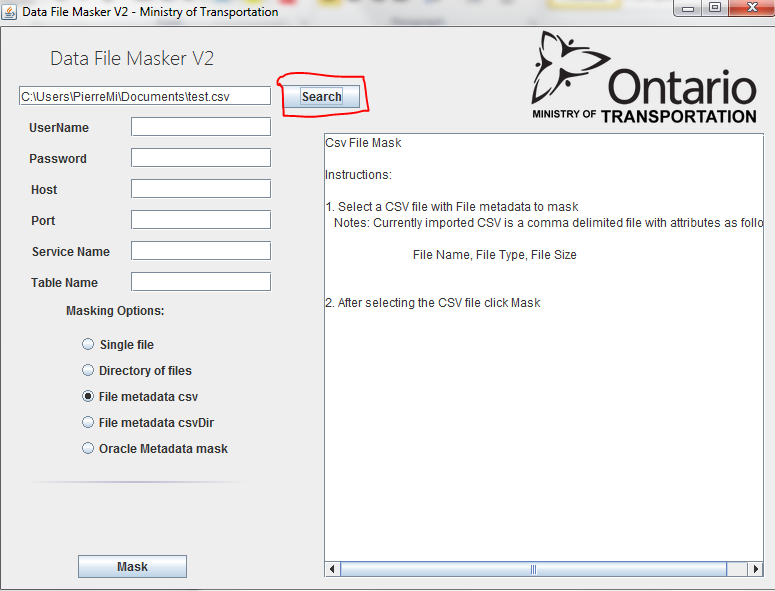


Step 2: Click mask

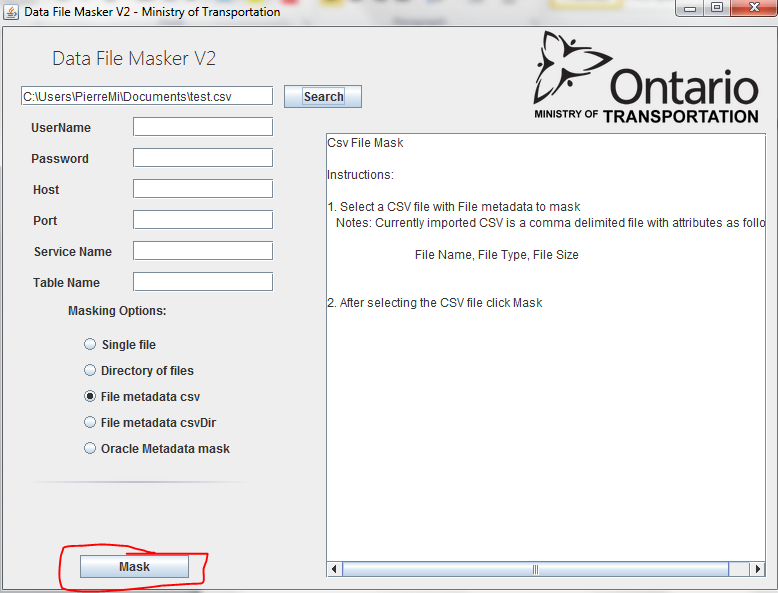


## File Metadata CSV mask

Step 1: Click CSV option and then click search (only one CSV file format is supported)

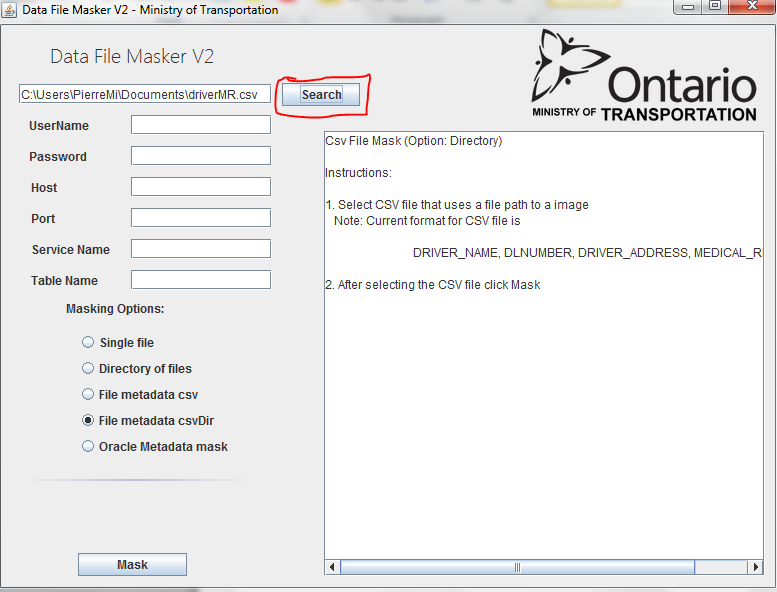


Step 2: Click Mask



## File Metadata CSV Mask (DIR)

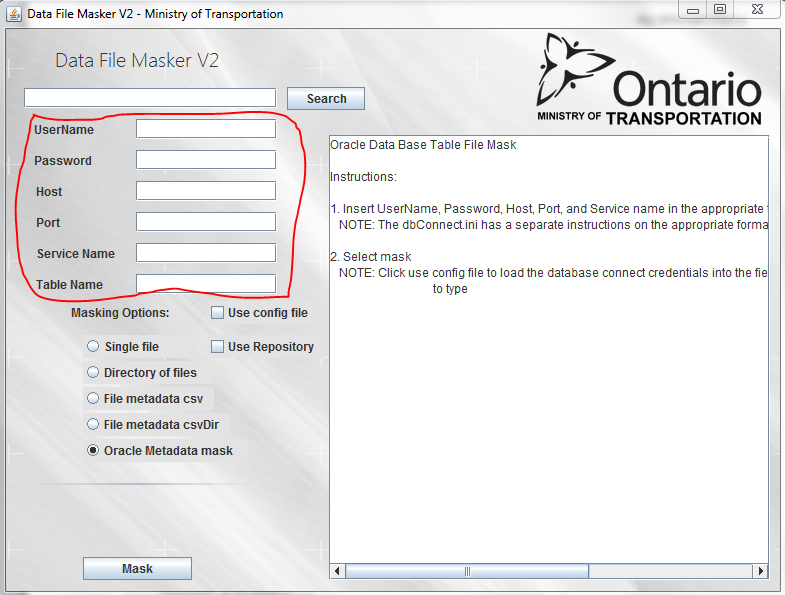
Step 1: Click search and find appropriate CSV file

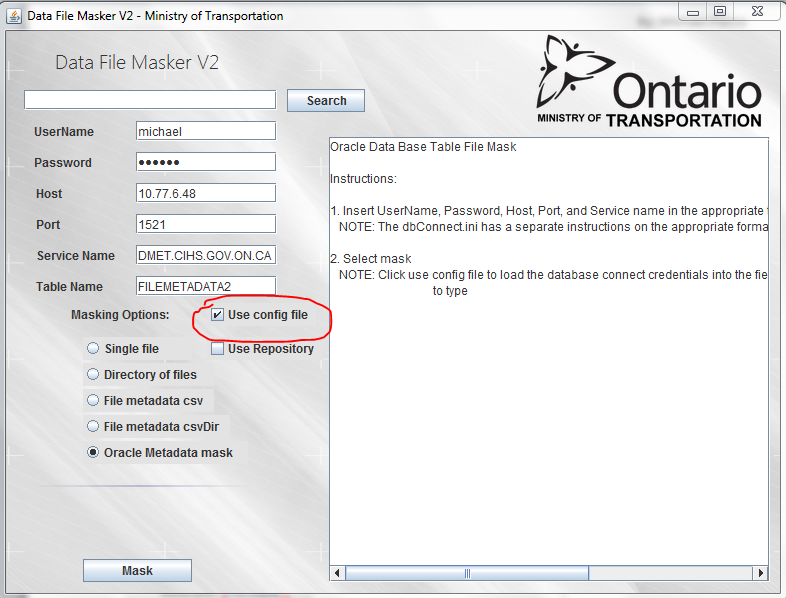


Step 2: Click mask (make sure CSV format matches the accepted column format)

### Oracle Metadata File Masking

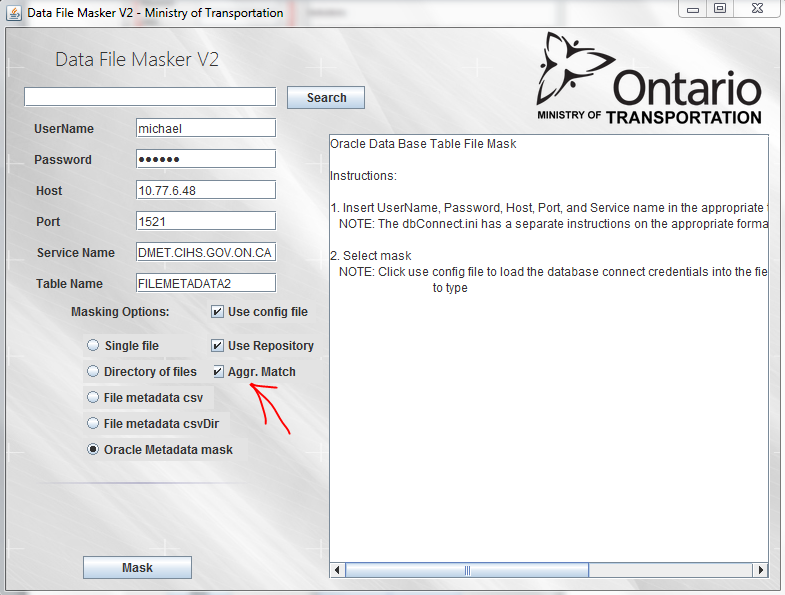
Step1: Type in oracle connection credentials. Alternatively click use config file to load a premade connection.





Step 2: Click Mask

**Alternatively you can use a repository of non-sensitive documents to mask:**

****

# Aggr. Match means that the application will match based on file type and size. With this option unchecked the application will use only file size to match the oracle table metadata with a non-sensitive document from the repository. If Aggr. Match is selected and there is no non-sensitive document to match the file type of the oracle record then it will not be created until one is added to the repository. However unchecked the file is created regardless because it only matches the closest size.