VennDexer Overview

Analyzing Attachments the Easy Way

Contents

[1. Purpose of this Document 2](#_Toc382946363)

[2. Scope of Application 2](#_Toc382946364)

[3. What Is VennDexer? 2](#_Toc382946365)

[4. Why Would I Want to Use It? (A Use Case) 2](#_Toc382946366)

[5. How Do I Run It? 3](#_Toc382946367)

[6. The Config File 3](#_Toc382946368)

[6.1 Root node: <VennDexerConfig>…</VennDexerConfig> 4](#_Toc382946369)

[6.2 Batch settings: <fileSet>…</fileSet> 4](#_Toc382946370)

[6.3 Location of source files: <srcDirs><src>…</src></srcDirs> 4](#_Toc382946371)

[6.4 Zip indicator: <areZipped>…</areZipped> 5](#_Toc382946372)

[6.5 Extraction indicator: <doExtract>…</doExtract> 6](#_Toc382946373)

[6.6 Location for the extracted files: <extractDir>…</extractDir> 6](#_Toc382946374)

[6.7 Where to put your results: <resultsDir>…</resultsDir> 6](#_Toc382946375)

[6.8 Settings for your index file: <index>…</index> 6](#_Toc382946376)

[6.9 Location of your index file: <file>…</file> 6](#_Toc382946377)

[6.10 Is the file delimited: <delimited>…</delimited> 6](#_Toc382946378)

[6.11 The delimiting character: <delimiter>…</delimiter> 6](#_Toc382946379)

[6.12 The column that contains the file name: <fileNameColumn>…</fileNameColumn> 7](#_Toc382946380)

[7. What It Doesn’t Do 7](#_Toc382946381)

[7.1 It Doesn’t Clean Your Index 7](#_Toc382946382)

[7.2 It Doesn’t Extract All Zip Files 7](#_Toc382946383)

[7.3 It Doesn’t Extract Zips within Zips 7](#_Toc382946384)

[8. In the Works 8](#_Toc382946385)

# Purpose of this Document

This overview will describe the typical use case to which VennDexer should be applied, how to set up a proper VennDexer configuration document, and what VennDexer doesn’t do.

# Scope of Application

This overview applies to VennDexer v1.0, released on 2014.03.17. Screenshots were taken on a Windows 8 machine.

# What Is VennDexer?

Technically speaking, VennDexer is a working title. The name could change at any moment, so be ready for that.

The title applies to a .NET library that performs analysis on a set of files based on an index of those files. We’ll cover exactly what the analysis does, but the short description is that it tells you whether or not you’ve got what the index says you should have.

# Why Would I Want to Use It? (A Use Case)

Here’s the use case that prompted the development of VennDexer:

A client was changing CRM vendors and needed to move several gigs worth of attachments out of one database into another. They gave us two things:

* a delimited index (think CSV) dump of the table with a list of their attachments and associated IDs
* a set of 8 compressed folders that comprised about 13 gigs of data once extracted

Immediately we saw a problem. We had 58k files (excluding duplicates), but only about 22k records in the index. We had no processes or tools in place for handling attachments in batches. (In fact, at that time, we didn’t even know if we could mass upload attachments. Turns out we could, so we had to figure out this whole attachment thing now.)

Of course, comparing two lists is not a big deal. Once we had a dir dump of what was in our compressed folders, it didn’t take that long to write an Excel macro to give us a good summary of what we had, and that macro was reusable. But we also saw the chance to take out some of the manual labor of getting that data into Excel in the first place—and do everything faster. That’s where VennDexer came in.

VennDexer does all of the heavy lifting. You just need to have the files and the index, and then VennDexer does the rest:

* If your files are in compressed folders, it will extract them (Windows native zip format only for now)
* Makes a list of the files and compares it to the index provided
* Gives back the following information:
  + Index record count (number of files we should have)
  + File count (number of files we actually have, excluding duplicates)
  + List of files, compiled from source directories. This is not a copy of the index.
  + A list of matches (full lines from delimited index)
  + List of index records that do not match a file (full lines from delimited index)
  + List of files that do not have an index
  + List of duplicate files

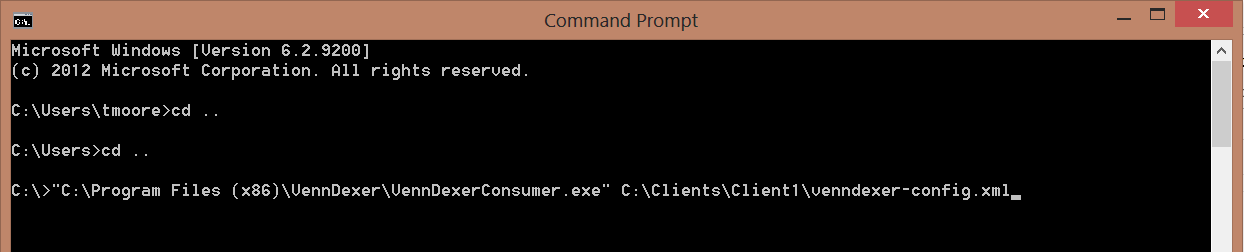
Yup. That’s right. You don’t even have to manually extract your files, you lazy bum. Including extraction time, VennDexer did our whole analysis on this client’s data in about 22 minutes. Just to put that in perspective, the file extraction took about 21 ½ minutes. So VennDexer gave a pretty big boost to our productivity.

# How Do I Run It?

Right now, you interact with VennDexer using a console application in Windows.

1. Open a command window
2. Call VennDexerConsumer.exe (wherever that is on your system), and pass it the name and location of the VennDexer config file. That’s it. Then just follow the instructions on screen.

Example:



Oh… you don’t know about the config file? Well, let me tell you what’s what.

# The Config File

The VennDexer config file is vital. It contains all the information you’re going to feed VennDexer. If you don’t get this right, your analysis isn’t going to work. VennDexer is designed to do a pretty good job of telling you when it runs into a problem, but usually the only problems it runs into are that something is wrong with the config file. So you want to understand this pretty well.

Here’s a default config file for VennDexer:

<?xml version="1.0"?>

<VennDexerConfig>

<fileSet>

<srcDirs>

<src>C:\users\%username%\Desktop\srcLoc\</src>

<areZipped></areZipped>

<doExtract></doExtract>

<extractDir></extractDir>

</srcDirs>

<resultsDir>c:\users\%username%\Desktop\Results\</resultsDir>

<index>

<file>c:\users\%username%\Desktop\fileIndex.csv</file>

<delimited>no</delimited>

<delimiter></delimiter>

<fileNameColumn></fileNameColumn>

</index>

</fileSet>

</VennDexerConfig>

We made an effort to name things in a way that’s pretty self-explanatory, but here’s the blow-by-blow for clarification:

## Root node: <VennDexerConfig>…</VennDexerConfig>

This is just the root element. Don’t ever delete it. Or do, and see what happens. You’re daring. You’re brave… You accept responsibility for the consequences of your actions.

## Batch settings: <fileSet>…</fileSet>

The container for all your settings for a given file set.

While it’s unlikely you’ll be analyzing several file sets back to back, it’s possible. The sample config accounts for a single file set. If you wanted to run more, you would just copy everything from <fileSet> to </fileSet>, including those tags, and paste them between the last </fileSet> tag and </VennDexerConfig>. Then you would adjust the settings for each file set.

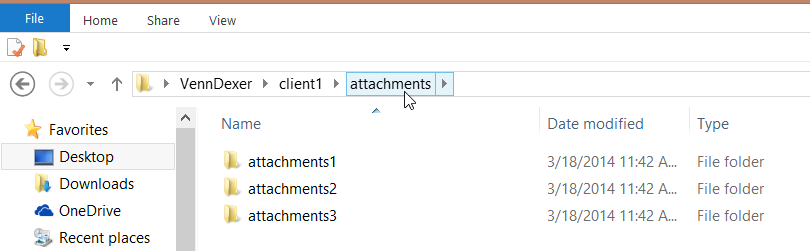
Keep in mind that if you use the batch process feature, VennDexer will run the entire batch before returning any results. If you want your results as soon as possible, create multiple config files and process them separately.

## Location of source files: <srcDirs><src>…</src></srcDirs>

You’ll have only one <srcDirs> container per fileSet, but you can have as many <src> entries as you want.

You create one <src> entry per directory that holds your files. These directories should *only* contain the attachment files (zipped or otherwise) themselves. For example, you don’t want your index in the same directory as your files. VennDexer doesn’t distinguish those. All of your stats will be thrown off by 1 if your index and files are in the same place.

The other thing you need to know about the src setting is that VennDexer automatically grabs any subdirectories. You only need to include top-level, or parent, directories. For example, say I have three directories with attachments in them under the parent container **attachments**:



I could set up my config with the following entry, only referencing the parent directory:

<srcDirs>

<src>C:\VennDexer\client1\attachments\</src>

…

</srcDirs>

Or I could enter all of the children:

<srcDirs>

<src>C:\VennDexer\client1\attachments\attachments1\</src>

<src>C:\VennDexer\client1\attachments\attachments2\</src>

<src>C:\VennDexer\client1\attachments\attachments3\</src>

…

</srcDirs>

However, the following would be a waste of time:

<srcDirs>

<src>C:\VennDexer\client1\attachments\</src>

<src>C:\VennDexer\client1\attachments\attachments1\</src>

<src>C:\VennDexer\client1\attachments\attachments2\</src>

<src>C:\VennDexer\client1\attachments\attachments3\</src>

…

</srcDirs>

When VennDexer looks in C:\VennDexer\client1\attachments\, it’s going to find the other three directories and add those as sources, as well. So by including the parent, you’ve captured the children. VennDexer is still going to go through the list you provided, but it’s just going to see that it already added the other directories and keep moving.

If you’re like me, you’ll just use one parent directory and fill it with all of the files and directories you need to analyze. But this config setup basically allows you to analyze files from all over the place.

## Zip indicator: <areZipped>…</areZipped>

Possible values: yes, no, and empty. Empty will be interpreted as no.

If you are working with zipped or compressed files, you need to enter yes here. VennDexer does not currently support mixing and matching. You can either work with zip files or not, but right now it can’t handle both.

## Extraction indicator: <doExtract>…</doExtract>

Possible values: yes, no or empty. Empty will be interpreted as no.

VennDexer 2.0 added functionality that allows you to analyze files without extracting them. This cuts way down on processing time. If you want to extract the files, enter yes here.

When you choose to extract zip files, VennDexer will automatically use your extraction directory (extractDir) as the source directory once the files are extracted. So you’ll enter the directories of the zip files as your source (src).

## Location for the extracted files: <extractDir>…</extractDir>

If your doExtract indicator was yes, you need an extractDir entry. This tells VennDexer where to put the extracted files. As a reminder, in case your memory only lasts a few seconds or you skipped the last section, when you are extracting zip files, the extraction directory will act as your source, so your src values will be the directories containing your zip files.

## Where to put your results: <resultsDir>…</resultsDir>

resulstDir tells VennDexer where to write your results

When VennDexer completes its analysis, it will show you some stats on the screen, but you probably want to keep those, huh? So it gives you the option to write them to file, as well as the lists of files it has created. (This is where the name VennDexer came from. Someone with a little graphics ability could actually take these results and put them in a Venn diagram.)

## Settings for your index file: <index>…</index>

Much like fileSet and srcDirs, this is just a container. The next four settings are all related to your index file, and this groups them together.

## Location of your index file: <file>…</file>

VennDexer needs to know where your index file is located if it’s going to do any analysis. If this is empty, you’re gonna have a bad time.

## Is the file delimited: <delimited>…</delimited>

Possible values: yes, no, or empty. Empty will be interpreted as no.

If your index file is delimited by any character, this should be yes. It doesn’t have to be a CSV, though that is probably the most common option.

## The delimiting character: <delimiter>…</delimiter>

This should contain the delimiting character. Don’t write the word *comma* here. Just enter , for a comma-separated file.

There’s one catch. There are two characters that are not translated as themselves. This is because some files are delimited by tabs or spaces, but entering white space in a config file can be a bet confusing. So the delimiter setting translates the following into the appropriate white space values:

* t will be translated to a tab character
* s will be translated to a space character

There are no extra escape characters or anything else. Just a t or an s. I couldn’t conceive of a scenario in which a file would be delimited by t’s or s’s, so I feel pretty safe on that one.

## The column that contains the file name: <fileNameColumn>…</fileNameColumn>

The column references in VennDexer are base-zero. That means that the first column is 0, the second column is 1, etc. Suppose you had the following line in a CSV file:

123, 12/4/12 7:00:32, side of building, photo64.jpg

The file name is in the last column, and there are four columns. But since we’re counting base-zero and not base-1, we’re going to tell VennDexer that the file names are in column 3. If that’s confusing, count like a normal person and then subtract one. You’ll be fine

That’s it. That’s probably way more explanation than you needed, but now it’s your fault if you don’t get it. Before you start making any other assumptions about how easy this is going to be, though, make sure you read the section on What It Doesn’t Do.

# What It Doesn’t Do

## It Doesn’t Clean Your Index

VennDexer makes at least one pretty big assumption: you have a clean delimited index file. Right now, there are no checks in place to determine, for example, that your CSV contains a comments field that has commas in it. If that throws off the column count of a given line, your results are going to be skewed. So you still have to do the work required to give VennDexer a clean index.

## It Doesn’t Extract All Zip Files

VennDexer only extracts files from native Windows compressed folders. It doesn’t know how to deal with 7-zip files, for example. So you may still have to extract the files yourself on occasion.

## It Doesn’t Extract Zips within Zips

VennDexer doesn’t extract zip files inside of other zip files. There’s a pretty simple reason, and I’ll go back to our use case to explain it. Our client had zip files as attachments in their old system. Those zip files were indexed *as zip files*. The files they contained were not indexed individually. Had we unzipped everything and then done our analysis, our numbers would have been off by quite a bit, and we would have created problems for ourselves trying to explain why we had so many files that weren’t indexed.

So VennDexer operates on the assumption that if your parent zip files contain zip files, the children should be left alone. If that’s not the case, you’ll have a little extra prep work to do. If this becomes enough of an issue, we might add a config setting to handle it in the future.

# In the Works

We’re continuing work on VennDexer to add little improvements here and there.

* Right now, all the file sets get processed in a batch, and the results come back as a batch. While batch file sets doesn’t seem like a huge part of the scope within which VennDexer will be applied, we could be wrong about that. It might be nice to get your results back as they’re processed, rather than at the end of the batch—especially if the program encounters errors on a batch somewhere in the middle. That would cause you not to get any of your results back at all. We’ll look into it.
* We’d really like to give some visual indicators of progress. Not high priority, but on our list of things to figure out.