# FolderScanUltra



Source code (C++) and latest “portable” download:
https://github.com/MaximumOctopus/FolderScanUltra

Alternative link for executable in “portable” format:
https://sourceforge.net/projects/xinorbis-and-tools/files/FolderScanUltra.zip/download



(c) Paul Alan Freshney 2026

## Introduction

FolderScanUltra is a powerful folder/drive analysis tool. If you have any suggestions on how to improve this software or identify any bugs, then please email me at the address on the credits page.

The software is capable of giving a quick analysis of the distribution and contents of any folder. For desktop users and network administrators it should make finding unwanted content much easier.

Once a folder or drive has been scanned then you can create many different reports from that data.

## Installation

It's advised to run FolderScanUltra in administrator mode. This isn't mandatory but will allow it to read and write the entire file system. You may now need this, but if you get an read/write errors while using it them administrator mode may help.

FolderScanUltra doesn't need installing in the normal sense. Either add the folder where FolderScanUltra is located to the "path" environmental variable or refer to it by its full path in batch files (etc.).

The database can be anywhere, but a local install is best (if possible) as it will increase write/read performance. Writing to a local database (`sqlite` or `odbc`) is also kinder to the network.

There should be ten files in the root of `FolderScanUltra\system\` folder. The 9 ".txt" files contain associations that are used to decide which category a file belongs to. Each file contains a list of file extensions (without leading full stop) one per line. Feel free to edit these!

A new log file is created every FolderScanUltra is run, they can be found in the logs folder.

**IMPORTANT: The sqlite3.dll files for 32-bit and 64-bit version FolderScanUltra have the same file names but are not compatible with the opposite version.**

**IMPORTANT: Keep a backup of your FolderScanUltra folder. Unzipping a new version will overwrite any edits you've made.**

## A sad addendum

You'll notice occasional references to xinorbis in this document. FolderScanUltra is an updated version of X.Robot, which was a powerful companion tool to xinorbis. As a console tool it could be used to quietly generate and store data without user intervention (in batch files, for instance).

FolderScanUltra still has all of these features (and more!). It's a great tool for scanning to servers in a batch file or scheduled task.

Xinorbis is a powerful GUI version of this tool. Due to the stupidly high cost of Delphi (with which Xinorbis was developed), around £1200, this is no longer being maintained. I would like to do a C++ or C# rewrite but it would be a multi-year task and for now it's on hold (though not impossible it will be finished one day).

FolderScanUltra *is* actively being maintained and will see many new features and improvements over the next few years. If you have any suggestions, then please get in touch (paul@freshney.org).

**Conventions**

When dealing with file sizes FolderScanUltra uses the following conventions.

| | |
|---|---|
| 1 kilobyte (1K) | = 1024 bytes |
| 1 megabyte (1MB) | = 1048576 bytes (1024 x 1024) |
| 1 gigabyte (1GB) | = 1073741824 bytes (1024 x 1024 x 1024) |
| 1 terabyte (1TB) | = 1099511627776 bytes (1024 x 1024 x 1024 x 1024) |

Not all applications follow this convention but as well as being the most commonly used throughout the industry it's also how Windows reports file sizes.

It's worth noting that hard disk manufacturers tend to use a slightly different method when reporting hard disk sizes.

| | |
|---|---|
| 1 gigabyte (1GB) | = 1000000000 bytes (1000 x 1000 x 1000) |
| 1 terabyte (1TB) | = 1000000000000 bytes (1000 x 1000 x 1000 x 1000) |

**Command Line Parameters**

The first parameter can be either:

The path to scan (unless you're using one of the help commands or *load*ing from a saved file):

```
fsu.exe folder_to_scan  (eg. fsu "e:\pictures\cats")
```

or

the name of a CSV file to import and process:

```
fsu.exe import_from.csv (eg. fsu cats.csv)
```

The CSV file **must** follow the same structure as an FSU CSV report

If FolderScanUltra is executed with just a single parameter, then it will output a simple summary of the scan/import to the console.

All comments and suggestions are gratefully received, please email them to me at *paul@freshney.org*

**Miscellaneous Commands**

**nothing**
Outputs version and basic "about" information" (web address and email etc.)

`/?` or `/h`
Simple help, and a few stats.

`/o`
Disables status output.

`/p`
Show scan and processing progress

`/test`
Test parameters and custom.ini settings (doesn't scan anything)

`/allowvirtual`
By default, FolderScanUltra will ignore all virtual files. Running with this command line switch will enable processing of virtual files (this is likely to cause unreliable file count and file size results).

`/xd;pattern`
Exclude all folders matching *pattern*.

`/dn`
Display duplicate files by file name to the console.

`/ds`
Display duplicate files by size to the console.

**Loading and Saving Configuration**

`/save;file_name`
Saves the specified command-line commands to a file called `file_name`. No extension is required, ".fsuproject" is appended automatically.

Neither `/load` or `/save` commands are saved to the file.

`/load;file_name`
Loads and executes the command-line commands from a file called `file_name`. No extension is required, as above, ".fsuproject" is appended automatically.

This must be the first option used. Other commands can be added after `/load` if required.

Saved configurations can be extended in this way:

eg. `fsu /load:cats /html`

**Compare**

`/compare;folder`
Compares the contents of `folder` against the primary folder being scanned. Will output a list of missing and new files, plus modified date and file size changes.

```
fsu d:\cats /compare;d:\cats_backup
```

**Optimisations**

`/notemp`
Temporary files will still be scanned, but the temporary file charts and lists will not be created. This might improve scan performance.

`/nouser`
Ignore file owner details. This option will improve scan performance, however it will mean that reports will not be able to show usage statistics based on file owner/creator.

`/noprocess`
If you only plan on producing file lists, or updating a database, then use this option as it will stop FolderScanUltra from unnecessarily processing the file data.

**Filter Input**

`/filter;n`
Filter by category, where *n* can be any of the following:

| | |
|---|---|
| `PROG` | program files |
| `SYS` or `SYSTEM` | system files |
| `GFX` or `GRAPHIC` | graphic files |
| `MOVIE` or `FILM` | movie files |
| `SND` or `SOUND` | audio files |
| `COD` or `CODE` | programming files |
| `COM` or `ZIP` | compressed files |
| `OTH` or `OTHER` | *other* files |
| `C`*i* | where i is 1 to 10 for custom category 1 to 10 |
| `CX` | for all custom categories |

**Report Output**

If no report options are selected, then a basic summary of the directory scan will be output to the console.

The configuration options (eg. `110`) are *optional*. The defaults are specified where appropriate.

Most of FolderScanUltra's options are either on or off, toggling an option is simply a matter of replacing the letters (a, b, c etc.) with either:

**0** to disable the option
**1** to enable the option

Where an option has many settings then this is specified separately.

If no output file name is specified, then FolderScanUltra will save the file to either:

The folder specified in the custom.ini file:

```
[FolderScanUltra]
DataPath=<folder>
```

Or if the above cannot be found, or is blank:

```
[Main]
DataPath=<folder>
```

If DataPath is blank then FolderScanUltra will use:

`<data>\FolderScanUltra\Reports\<report_type>\fsu_yyyymmdd_hhmmss.xxx`

typically: `c:\ProgramData\MaximumOctopus\xinorbis\FolderScanUltra\Reports\...`

If custom.ini cannot be found then, FolderScanUltra will use:

`<install>\system\Reports\FolderScanUltra\<report>\fsu_yyyymmdd_hhmmss.xxx`

**Generate CSV report**

Outputs a CSV file to the path specified in *file_name*.

*File name* and *options* are optional. Default shows the values that will be used if no parameters are specified.

Usage: `/CSV;abc;file_name`

Default: `/CSV;110;<output folder>\fsu_yyyymmdd_hhmmss.csv`

option a:
- 0 - category output only
- 1 - full file list (default)

option b:
- 0 - Don't add "heading row"
- 1 - Add "heading row" (default)

option c:
- 0 - File sizes in most convenient format (1MB, 400bytes etc.)
- 1 - All file sizes in bytes
- 2 - All file sizes in kilobytes
- 3 – All file sizes in megabytes

If a file name is not specified, then it will be saved using the rules on **Page 6 – Report Output**.

Example:

`/CSV;1`

Create only a list of files in the scan

`/CSV;10`

Create only a list of files in the scan and don't add a heading row.

**Generate HTML report**

Outputs an HTML report to the optional *file name*.

*File name* and *options* are optional. Default shows the values that will be used if no parameters are specified.

Usage: `/HTML;abcdefghijklm;file_name`

Default: `/HTML;1111111111120;<output folder>\fsu_yyyymmdd_hhmmss.htm`

| | | |
|---|---|---|
| option a: | graphs on (1 default) or off (0) | |
| option b: | include File Attributes section | (default is 1) |
| option c: | include Category section | (default is 1) |
| option d: | include Directories section | (default is 1) |
| option e: | include Magnitude section | (default is 1) |
| option f: | include File Extension list section | (default is 1) |
| option g: | include Null Files section | (default is 1) |
| option h: | include File Dates section | (default is 1) |
| option i: | include Top 50 section | (default is 1) |
| option j: | include Users section | (default is 1) |
| option k: | include Temporary files section | (default is 1) |
| option l: | 0 - report is 800 pixels wide | |
| | 1 - report is 1024 pixels wide | |
| | 2 - report is 1280 pixels wide | (default) |
| option m: | 0 - File sizes in most convenient format (1MB, 400bytes etc.) | |
| | 1 - All file sizes in bytes | |
| | 2 - All file sizes in kilobytes | |

If a file name is not specified, then it will be saved using the rules on **Page 6 – Report Output**.

`/deephtml`

This is a new HTML report option that will include the second level of folder hierarchy below the root. This extra information is likely to be more useful than a top-level folder listing.

This uses the same options as a standard HTML report but includes more information at the bottom of the report.

**Generate JSON report**

Outputs a complete list of files and properties in JSON format.

*File name* and *options* are optional. Default shows the values that will be used if no parameters are specified.

Usage: `/JSON;a;file_name`

Default: `/JSON;0;<output folder>\fsu_yyyymmdd_hhmmss.csv`

option a: 0 – file names and paths only (default)
1 - full file and property list

If a file name is not specified, then it will be saved using the rules on **Page 6 – Report Output**.

**Generate Date Reports (ASCII)**

Outputs a text report where the files are listed by day (`/dated`), month (`/datem`) or year (`/datey`).

*File name* and *options* are optional. Default shows the values that will be used if no parameters are specified.

Usage: `/dated;a;file_name`

Default: `/dated;1;<output folder>\fsu_yyyymmdd_hhmmss_day.htm`

Usage: `/datem;a;file_name`

Default: `/datem;1;<output folder>\fsu_yyyymmdd_hhmmss_month.htm`

Usage: `/datey;a;file_name`

Default: `/datey;1;<output folder>\fsu_yyyymmdd_hhmmss_year.htm`

option a: use file's created date for sorting (if this is zero, then modified date is used instead)

**Generate Xinorbis report**

This was a report specific to the Xinorbis application. Support for this format is included for historical reasons, but it is not advised that you use it.

**Generate a simple summary of findings**

Outputs a summary to the console.

Usage: `/SUM`

Default: `/SUM`

Only outputs data to the console.

**Top 20 lists**

**All options output to the console.**

`/top20`

Top 20 largest files

`/bottom20`

Top 20 largest files

`/new20`

Top 20 newest files

`/old20`

Top 20 oldest files

`/folderstop20`

Shows the top largest folders in the root of the scan folder, ordered by size.

**Other console output options**

`/allfolders`

Shows all folders in the root of the scan folder, ordered by size (with a nice graph)

The following five options output the respective text report sections:

`/attributes`

Shows a selection of file attributes; the number of files with the attribute, and the total size of those files. Because a file can have multiple attributes, the percentage values can add up to more than 100%.

`/categories`

Show category overview. The number of files per category, and the size of those files.

Will only show `custom` categories if the file count is non-zero.

`/extensions`

A list of categories and the corresponding file extensions that have been detected during the current scan. File count, and total file size for each file extension (and each category) are shown.

Will only show `custom` categories if the file count is non-zero.

`/users`

A list of users and their corresponding file count and total file size from the current scan (only shows users who have files found in the current scan).

**Generate an ASCII text report**

Outputs an ASCII text report file to the path specified in *file name*.

*File name* and *options* are optional. Default shows the values that will be used if no parameters are specified.

Usage: `/text;abcdefghij;file_name`

Default: `/text;1111111111;<output folder>\fsu_yyyymmdd_hhmmss.txt`

| | | |
|---|---|---|
| option a: | UNUSED | |
| option b: | include File Attributes section | (default is 1) |
| option c: | include Category section | (default is 1) |
| option d: | include Directories section | (default is 1) |
| option e: | include Magnitude section | (default is 1) |
| option f: | include File Extension list section | (default is 1) |
| option g: | include Null Files section | (default is 1) |
| option h: | include File Dates section | (default is 1) |
| option i: | include Top 50 section | (default is 1) |
| option j: | include Users section | (default is 1) |

If a file name is not specified, then it will be saved using the rules on **Page 6 – Report Output**.

`/deeptext`

This is a new Text report option that will include the second level of folder hierarchy below the root. This extra information is likely to be more useful than that a top-level folder listing.

This uses the same options as a standard Text report but includes more information at the bottom of the report.

**Generate a Tree report**

Outputs a tree diagram showing the structure of all folders and files.

*File name* and *options* are optional. Default shows the values that will be used if no parameters are specified.

Usage: `/tree;ab;file_name`

Default: `/tree;11;<output folder>\fsu_yyyymmdd_hhmmss.txt`

| | | |
|---|---|---|
| option a: | include file attributes | (default is 1) |
| option b: | include size information | (default is 1) |

**Generate an XML report**

Outputs an XML 1.0 compliant report file to the optional file name. Report can either be a complete list of files along with properties, or, category usage data.

*File name* and *options* are optional. Default shows the values that will be used if no parameters are specified.

Usage: `/XML;abcdefghijk;file_name`

Default: `/XML;01111111111;<output folder>\fsu_yyyymmdd_hhmmss.xml`

| | | |
|---|---|---|
| option a: | file list (1) or summary of results | (default is 0) |
| option b: | include File Attributes section | (default is 1) |
| option c: | include Category section | (default is 1) |
| option d: | include Directories section | (default is 1) |
| option e: | include Magnitude section | (default is 1) |
| option f: | include File Extension list section | (default is 1) |
| option g: | include Null Files section | (default is 1) |
| option h: | include File Dates section | (default is 1) |
| option i: | include Top 50 section | (default is 1) |
| option j: | include Users section | (default is 1) |
| option k: | include Temporary files section | (default is 1) |

If a file name is not specified, it will be saved using the rules on **Page 6 – Report Output**.

**Generate an XML file list report**

File name parameter is optional. Default shows the values that will be used if no parameters are specified.

Usage: `/XFL;file_name`

Default: `/XFL`

Outputs an XML 1.0 compliant report to the path specified in <file name>.

If a file name is not specified, then it will be saved using the rules on **Page 6 – Report Output**.

**Generate a Benford's Law report (also known as the first-digit law)**

Shows a Benford's law distribution to the console.

Usage: `/benford`

For more information see here: https://en.wikipedia.org/wiki/Benford%27s_law

**Extra Commands**

Usage: `/listroot`

Lists only the folders and files in the root of the selected scan folder. Doesn't scan anything.

If you're experimenting with access permissions to FolderScanUltra then this option will let you see what FolderScanUltra can see.

Usage: `/test`

Will test the validity of the FolderScanUltra command-line parameters. Use it to check odbc connectivity (and the connection string), status of the `sqlite` database, and various other system checks.

Usage: `/meow`

This option must **never** be used.

Usage: `/console` *(new for version 5.5)*

Displays a command console after scanning and analysis has finished. This console gives access to many functions and commands for extracting, searching, and outputting data based on the scanned folder.

See the next section for details on to use the console.

**Console**

`help`

Displays some help information.

`exit`

Exits the console, and the application.

`report` *`report_type display_count`*

where *`report_type`* is one the following:

```
topfolders
topfiles
bottomfiles
newfiles
oldfiles
summary
```

and *`display_count`* is the number of items to display (optional, default is 20).

plus, the standard report types (where *`display_count`* does nothing):

`csv`, `html`, `text`, `tree`, and `xml`

`save [file name]`

Save the search results to a CSV file. File name is optional, default is `search_yyyymmdd_hhmmss.csv`.

`search` *`term`*

Search within file names for `term`.

`filter` *`parameters`*

Search using one or more search parameters, detailed in the section below:

# Filter Syntax

FolderScanUltra has a very powerful search system. This document lists every search option available in the latest version, and examples of usage.

The rules and information in this document apply across the entire application.

All of the following can be combined to create very powerful searches.

**Combining search terms**

Example: *(size>10MB) #GFX*

Finds all image files that are 10MB in size or larger.

Example: *cat #GFX @MODIFIED*

Finds all image files that were modified today and who have a file name containing the word "cat".

Example: `#OFF (user=Paul)`

Find office/productivity files created by the user Paul.

Example: `(date>!7)`

Find all files created in the last 7 days.

**Keyword(s)**

Find files and folders containing one or more keywords. Use the asterisk as a wildcard at the beginning or ending of a keyword to find specific file names.

Usage: `keyword`

Example: *fish*

Will find all files or folders containing “fish”.

Usage: `keyword1 keyword2`

Example: *fish documents*

Will find all files/folders containing “fish” *and* “documents”.

Usage: `keyword*`

Example: *paul**

Will find all file names starting with “paul”.

Usage: `*keyword`

Example: **paul.jpg*

Will find all file names ending in “paul.jpg”.

**File Name / Path**

Use keywords for finding specific text. The following allow for filtering by file name or path length.

Usage: `(filenamelength=*length*)`

Example: `(filenamelength=10)`

Will find file names (including extension) of exactly 10 characters. Ignores full path.

Usage: `(filenamelength<*length*)`

Example: `(filenamelength<10)`

Will find file names (including extension) of 10 characters or fewer. Ignores full path.

Usage: `(filenamelength>*length*)`

Example: `(filenamelength>10)`

Will find file names (including extension) of 10 characters or more. Ignores full path.

Usage: `(filepathlength=*length*)`

Example: `(filepathlength=20)`

Will find full file paths of exactly 20 characters.

Usage: `(filepathlength<*length*)`

Example: `(filepathlength<20)`

Will find full file paths of 20 characters or fewer.

Usage: `(filepathlength>*length*)`

Example: `(filepathlength>20)`

Will find full file paths of 20 characters or more.

Usage: `(includefolder=*text*)`

Example: `(includefolder=cats)`

Includes folders matching text “cats” in search results.

Usage: `(excludefolder=*text*)`

Example: `(excludefolder=cats)`

Excludes folders matching text “cats” from search results.

**File Size**

Filtering results by file size. When no unit is specified, the value will be assumed to be in bytes.

The following units are available:

| | |
|---|---|
| K | kilobytes (1024 bytes) |
| MB | megabytes (1024 kilobytes) |
| GB | gigabytes (1024 megabytes) |

Usage: `(size=x)`

Example: `(size=1000)`

Will find all files of exactly 1000 bytes.

Usage: `(size>x)`

Example: `(size>1MB)`

Will find all files greater than or equal to 1MB.

Usage: `(size<x)`

Example: `(size<100K)`

Will find all files less than or equal to 100KB.

Example: `(size>100K) (size<1MB)`

Will find all files between 100K and 1MB in size (inclusive).

**File Owner (user)**

Filter by file owner (user).

Usage: `(user=`*`name`*`)`

Example: `(user=paul)`

Will find all files where the owner is “paul”.

Usage: `(user!=`*`name`*`)`

Example: `(user!=paul)`

Will find all files where the owner is NOT “paul”.

Usage: `(user~`*`name`*`)`

Example: `(user~fresh)`

Will find all files where the owner contains the text “fresh”.

Usage: `(user!~`*`name`*`)`

Example: `(use!~paul)`

Will find all files where owner doesn’t contain the text “fresh”.

**File Date**

Date searching supports three date formats:

`dd/mm/yyyy`, `yyyy/mm/dd`, and `yyyymmdd`

It also supports two special tokens that can be used instead of a set date:

| | |
|---|---|
| `!x` | Where x is the number of days before the current (today's) date. |
| `$x` | Where x is the number of months before the current (today's) date. |

**Created date**

Usage: `(date=`*`dd/mm/yyyy`*`)`

Example: `(date=17/04/2019)`

Will find all files created on the 17th of April 2019.

Usage: `(date<`*`dd/mm/yyyy`*`)`

Example: `(date<`*`17/04/2019`*`)`

Will find all files created on or before 17th of April 2019.

Usage: `(date>`*`dd/mm/yyyy`*`)`

Example: `(date>`*`17/04/2019`*`)`

Will find all files created on or after the 17th of April 2019.

**Accessed date**

Usage: `(adate=`*`dd/mm/yyyy`*`)`

Example: `(adate=17/04/2019)`

Will find all files that were last accessed on the 17th of April 2019.

Usage: `(adate<`*`dd/mm/yyyy`*`)`

Example: `(adate<17/04/2019)`

Will find all files that were last accessed on or before 17th of April 2019.

Usage: `(adate>`*`dd/mm/yyyy`*`)`

Example: `(adate>17/04/2019)`

Will find all files that were last accessed on or after the 17th of April 2019.

**Modified date**

Usage: (mdate=*dd/mm/yyyy*)

Example: (mdate=17/04/2019)

Will find all files that were last modified on the 17th of April 2019.

Usage: (mdate<*dd/mm/yyyy*)

Example: (mdate<17/04/2019)

Will find all files that were last modified on or before 17th of April 2019.

Usage: (mdate>*dd/mm/yyyy*)

Example: (mdate>17/04/2019)

Will find all files that were last modified on or after the 17th of April 2019.

**File Times**

Find files based on their creation time, time of last access, or the time there were last modified.

All time searching ignores the value of seconds stored. A search for a time of 9:25am will search for 9:25am and zero seconds to 9:25am and 59 seconds.

**Created time**

Usage: `(time=`*`hhmm`*`)`

Example: `(time=0930)`

Will find all files created at 9:30am.

Usage: `(time<`*`hhmm`*`)`

Example: `(time<0930)`

Will find all files created on or before 9:30am.

Usage: `(time>`*`hhmm`*`)`

Example: `(time>0930)`

Will find all files created on or after 9:30am.

**Accessed time**

Usage: `(atime=`*`hhmm`*`)`

Example: `(atime=0930)`

Will find all files last accessed at 9:30am.

Usage: `(atime<`*`hhmm`*`)`

Example: `(atime<0930)`

Will find all files last accessed on or before 9:30am.

Usage: `(atime>`*`hhmm`*`)`

Example: `(atime>0930)`

Will find all files last accessed on or after 9:30am.

**Modified time**

Usage: `(mtime=`*`hhmm`*`)`

Example: `(mtime=0930)`

Will find all files last modified at 9:30am.

Usage: `(mtime<`*`hhmm`*`)`

Example: `(mtime<0930)`

Will find all files last modified on or before 9:30am.

Usage: `(mtime>`*`hhmm`*`)`

Example: `(mtime>0930)`

Will find all files last modified on or after 9:30am.

**File Attributes**

Filter by various *Windows* and *FolderScanUltra*-specified file attributes.

For more information on Windows file attributes see here:
https://docs.microsoft.com/en-us/windows/win32/fileio/file-attribute-constants

All of these filters can be used in the negative (to exclude files) by appending a hyphen: *@SYSTEM-*

**Windows File Attributes**

Usage: `@ARCHIVE`

Find archived files.

Usage: `@COMPRESSED`

Find compressed files. This is not the same as the Xinorbis compressed category *#COM*. This uses the Windows attributes to determine compressed status.

Usage: `@ENCRYPTED`

Find encrypted files.

Usage: `@HIDDEN`

Find hidden files.

Usage: `@NOTCONTENTI`

Find files that are not indexed by Windows.

Usage: `@OFFLINE`

Find files that are stored offline.

Usage: `@READONLY`

Find read-only files.

Usage: `@RECALLONDATAACCESS`

Find files that are not fully present locally.

Usage: `@RECALLONOPEN`

Find files that have no physical representation on the local system.

Usage: `@REPARSEPOINT`

Find *reparsepoint* files.

Usage: `@SPARSEFILE`

Find *sparsefiles*.

Usage: `@SYSTEM`

Find system files.

**FolderScanUltra functions**

Usage: `@CREATED`

Find files created today.

Usage: `@ACCESSED`

Find files last accessed today.

Usage: `@MODIFIED`

Find files last modified today.

Usage: `@FILE`

Find only files.

Usage: `@FOLDER`

Find only folders.

Usage: `@VIRTUAL`

Find all files that are considered virtual on this system. This is a combination of the following three attributes: *`@RECALLONDATAACCESS`*, *`@RECALLONOPEN`*, and *`@OFFLINE`*.

Usage: `@TEMP`

Find temporary files.

**Category**

Filters using the FolderScanUltra file category types.

All of these filters can be used in the negative (to exclude files) by appending a hyphen: *#GFX-*

Usage: `#COD`

Find only files from the *coding/programming* category.

Usage: `#COM`

Find only files from the *compressed* category.

Usage: `#GFX` or `#GRAPHIC`

Find only files from the *graphics/images* category.

Usage: `#MOVIE`

Find only files from the *movie/video* category.

Usage: `#OFF`

Find only files from the *office/prod* category.

Usage: `#OTH`

Find only files from the *other* category (these are files that don't fit in an other category.

Usage: `#PROG`

Find only files from the *applications* category.

Usage: `#SND` or `#SOUND`

Find only files from the *audio/sound* category.

Usage: `#SYS`

Find only files from the *system* category.

Usage: `#C1, #C2, #C3, #C4, #C5, #C6, #C7, #C8, #C9, #C10`

Find only files from the specified custom category (1 – 10).

**Dynamic file name generation**

For use in all options that take a file name as a parameter. These parameters are **case-sensitive**.

| | |
|---|---|
| $XD | The drive where FolderScanUltra was run from |
| $XF | The folder where FolderScanUltra was run from |
| $PC | The name (network) of the PC FolderScanUltra is executed on |
| $User | The name of the user that executed FolderScanUltra |
| $yyyy | Current year (e.g. 2009) |
| $YY | Last two digits of current year (e.g. 08) |
| $mm | Current month (01 for January, 02 for February, etc.) |
| $MM | Current month as short word (Jan, Feb, Mar, Apr, etc.) |
| $dd | Current day (01 through 28, 29, 30 or 31 depending on month) |
| $DD | Current day as short word (Mon, Tue, Wed, etc.) |
| $Th | Hour part of the current time (00 - 23) |
| $Tm | Minute part of the current time (00 - 59) |
| $Ts | Second part of the current time (00 - 59) |

Examples of usage:

Current date in `yyyymmdd` format:

```
/TXT;c:\folderscanultra\$yyyy$mm$dd_dailyscan.txt
```

Store files in individual folders for month and year:

```
/TXT;c:\scan\$yyyy\$mm\$dd_dailyscan.txt
```

**Database Access**

FolderScanUltra can update a folder history database if required to do so (this is probably its most powerful and useful feature). Use FolderScanUltra to update the database (e.g. automatically through a batch file or scheduled task) and then use X.Database, or Xinorbis, to view reports, graphs etc.

`/odbc`
Updates an ODBC compliant database with all file and folder information.

FolderScanUltra will expect a `custom.ini` file with a "connectionstring" parameter in the same format as Xinorbis. A template `custom.ini` file is included in the portable package.

Ideally Xinorbis and FolderScanUltra should live in the same folder; they'll be much happier and can share config information.

`connectionstring`s are database-voodoo, for more help on creating them see this excellent website: https://www.connectionstrings.com/.

`/sqlite`
Updates an SQlite 3 database with all file/folder details.

FolderScanUltra will read the config.ini file and use the same database as Xinorbis. If no *config.ini* file is present then the default path will be used; depending on which version of Windows is installed:

```
C:\documents and settings\<user>\folderscanultra\folderhistory\database\xinorbis.db

C:\users\<user>\folderscanultra\folderhistory\database\xinorbis.db
```

`/updatescanhistory`
Updates the Xinorbis Scan History. Only useful if you use Xinorbis on the same PC as FolderScanUltra.

The command below will scan "somefolder" and update the Xinorbis database via ODBC and update the Xinorbis scan history:

```
fsu somefolder /odbc /updatescanhistory
```

If `somefolder` contains spaces, then you must use quotes around the folder path:

```
fsu “some folder” /odbc /updatescanhistory
```

**Alternative database structure**

It's now possible to save scan data to an alternative database structure. This new structure is more suited to users who wish to interrogate the data with other tools (such as PowerBI).

`/dbstructured`
Will enable the new format.

`/systemtable;tablename`
Set the name of the system table. If this parameter is not specified, then "`XinorbisSystem`" will be used instead.
(it will be created if it doesn't exist)

`/datatable;tablename`
Set the name of the data table. If this parameter is not specified, then the following is used:

```
MD5Checksum(scan path) + yyyymmddHHMMSS
```

(will be created if it doesn't exist)

**System table**

One row per scan.

| | |
|---|---|
| `TableName` | Name of the data table where the scan data is stored |
| `Folder` | Full path to the scanned folder |
| `SizeString` | Size of the scan written in the most convenient format (eg 51.22GB) |
| `Size` | Size of the scan in bytes |
| `Files` | File count |
| `Folders` | Folder count |
| `ScanDate` | Scan data formatted as `yyyymmddHHMMSS`. |

**Data table**

One row per scanned object (file or folder). A field for each object property.

An extra field "ScanDate" matching the "ScanDate" field of the system table can be used to easily retrieve data.

Xinorbis cannot (yet) read data structured in this manner, it's designed for external tools only.

The X.Database tool will be updated soon to understand this format.

**XML Report Structure**

FolderScanUltra outputs fully v1.0 compliant XML.

**Structure for XML file list:**

| | |
|---|---|
| `<folderscanultrafilelist>` | |
| `<file>` | |
| `<name>` | File name |
| `<path>` | Full file path |
| `<sizewords>` | Size represented in most convenient format: 1MB, 500K etc. |
| `<sizebytes>` | Size in bytes |
| `<sizeondiskwords>` | Used disk space in the most convenient format: 1MB, 500K etc. |
| `<sizeondiskbytes>` | Actual used disk space in bytes |
| `<owner>` | File owner |
| `<datecreated>` | Date file created in DD/MM/YYYY format |
| `<datemodified>` | Date file modified in DD/MM/YYYY format |
| `<dateaccessed>` | Date file was last accessed in DD/MM/YYYY format |
| `<category>` | File category (1- programs etc.) |
| `<directory>` | 1 if folder, 0 if file. |
| `<readonly>` | 1 if file is read only |
| `<hidden>` | 1 if file is hidden |
| `<system>` | 1 if file is system file |
| `<archive>` | 1 if file is archive |
| `</file>` | |
| `</folderscanultrafilelist>` | |

**Structure for XML report output**

| | |
|---|---|
| `<folderscanultrareport>` | |
| `<information>` | |
| `<directory>` | analysed path |
| `<date>` | date of analysis |
| `<numberoffiles>` | number of files included in analysis |
| `<numberofdirectories>` | number of folder in analysis |
| `<sizeoffiles>` | total size of the files analysed |
| `<diskspacefreee>` | disk space available on target drive |
| `<diskspacemax>` | disk space total on target drive |
| `<sectorspercluster>` | number of sectors per cluster |
| `<bytespersector>` | number of bytes per sector |
| `<freeclusters>` | free clusters on hard disk |
| `<totalclusters>` | total clusters on hard disk |
| `<volumename>` | volume name of hard disk |
| `<serialnumber>` | hard disk serial number |
| `<filesystem>` | file system type (FAT32, NTFS etc.) |
| `</information>` | |
| `<categorylist>` | |
| `<category name='">` | one section for each of the 10 file categories |
| `<numberoffiles>` | number of files belonging to relevant category |
| `<numberoffilesaspercent>` | number of files as percentage of total analysed |
| `<sizeoffiles>` | size of files analysed, belonging to relevant category |
| `<sizeoffilesaspercent>` | size of files as percentage of total analysed, relevant category |
| `</categorylist>` | |
| `<dirlist>` | |
| `<dir name="">` | one section for each dir analysed |
| `<numberoffiles>` | number of files in analysed folder |
| `<numberoffilesaspercent>` | number of files as percentage of total analysed |

```
            <sizeoffiles>           combined size of files
            <sizeoffilesaspercent>  combined size of files, as percentage of total analysed
      </dir>
</dirlist>

<magnitudelist>
      <magnitude name="">           one section for each magnitude category
            <numberoffiles>         number of files in analysed folder
            <numberoffilesaspercent>number of files as percentage of total analysed
            <sizeoffiles>           combined size of files
            <sizeoffilesaspercent>  combined size of files, as percentage of total analysed
      </magnitude>
</magnitudelist>

<extensiondata>
      <extensioncategory name="">   one section for each extension category
            <numberoffiles>         number of files in analysed folder
            <numberoffilesaspercent>number of files as percentage of total analysed
            <sizeoffiles>           combined size of files
            <sizeoffilesaspercent>  combined size of files, as percentage of total analysed

            <extension name="">     one section for extension within this category that has more
                                    than one file associated with it

            <numberoffiles>         number of files in analysed folder
            <numberoffilesaspercent>number of files as percentage of total analysed
            <sizeoffiles>           combined size of files
            <sizeoffilesaspercent>  combined size of files, as percentage of total analysed
            </extension>
      </extensioncategory>
</extensiondata>

<top50largest>
      <top50large size="">path to file</top50large>
</top50largest>

<top50smallest>
      <top50small size=””>path to file</top50small>
</top50smallest>

<nullfiles>
      <nullfile name=''">path to</nullfile>
</nullfiles>

</folderscanultrareport>
```

If you need any help with the XML output, please don't hesitate to contact me.

# Credits

| | |
|---|---|
| **Programming** | Paul Alan Freshney |
| **Development Cats** | Rutherford, Freeman, and Maxwell (maximumoctopus.com/developmentcats.htm) |
| **Icon** | https://icon-icons.com/ |
| **Database Engine** | SQlite (www.sqlite.org) |
| **Czech Translation** | Jakub Markytán |
| **French Translation** | Christian Perronnet |
| **German Translation** | Marcus Barkhahn |
| **Hungarian Translation** | Zsolt Brechler |
| **Russian Translation** | kopejkin |
| **Italian Translation** | Victor |
| **Thanks to** | Monpelaud, Dave Mahadevan, Vit, Damiaan Peeters, Mike Dutch, Robert Pallot, Peter Garrety, Fred de Vries, Glyn Selwyn, Tom Grimes, Fredie Botha, and Rod.<br><br>And *everyone* who has sent me feedback. Please keep it coming! |

All of my software is free and open source.

Please donate to your local **cat charity** or **shelter**, thanks.



paul@freshney.org

www.maximumoctopus.com

https://github.com/MaximumOctopus/FolderScanUltra