



# *Tools*

ZIMMERMAN / RATHBUN

# EZ Tools Manuals

Andrew Rathbun and Eric Zimmerman

This book is for sale at <http://leanpub.com/eztoolsmanuals>

This version was published on 2023-03-06

ISBN 978-1-959497-02-8



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2022 - 2023 Andrew Rathbun and Eric Zimmerman

## **Tweet This Book!**

Please help Andrew Rathbun and Eric Zimmerman by spreading the word about this book on [Twitter](#)!

The suggested hashtag for this book is [#eztoolsmanuals](#).

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

[#eztoolsmanuals](#)

*This book would not exist but for the heroic effort of Andrew Rathbun to suggest it and put in an enormous amount of work to get it off the ground. I am in your debt sir!*

# Contents

Enabling Update Notifications on Leanpub .....	1
<b>Introduction to EZ Tools .....</b>	<b>2</b>
What are EZ Tools? .....	2
Download EZ Tools .....	2
CLI vs GUI .....	3
.NET 4 vs .NET 6 EZ Tools .....	3
What is this book? .....	3
Mastering EZ Tools .....	3
Content by Eric Zimmerman .....	4
Content by the DFIR Community about EZ Tools .....	4
<b>EZ Tools - Common Switches .....</b>	<b>5</b>
Common Switches .....	5
<b>EZ Tools - PowerShell vs CMD .....</b>	<b>9</b>
Common Scenarios .....	9
<b>EZ Tools - CLI .....</b>	<b>11</b>
<b>AmcacheParser .....</b>	<b>12</b>
AmcacheParser Introduction .....	12
AmcacheParser Switches .....	13
AmcacheParser Command Examples .....	15
AmcacheParser Output .....	15
AmcacheParser Key Takeaways .....	16
AmcacheParser References .....	17
<b>AppCompatCacheParser .....</b>	<b>18</b>
AppCompatCacheParser Introduction .....	18
AppCompatCacheParser Switches .....	19
AppCompatCacheParser Command Examples .....	21
AppCompatCacheParser Output .....	21
AppCompatCacheParser Key Takeaways .....	22
AppCompatCacheParser References .....	23

## CONTENTS

<b>bstrings</b> . . . . .	25
bstrings Introduction . . . . .	25
bstrings Switches . . . . .	26
bstrings Command Examples . . . . .	34
bstrings References . . . . .	36
<b>EvtxECmd</b> . . . . .	37
EvtxECmd Introduction . . . . .	37
EvtxECmd Switches . . . . .	38
EvtxECmd Command Examples . . . . .	42
EvtxECmd Output . . . . .	43
EvtxECmd Key Takeaways . . . . .	45
EvtxECmd References . . . . .	46
<b>IISGeoLocate</b> . . . . .	47
IISGeoLocate Introduction . . . . .	47
IISGeoLocate Switches . . . . .	48
IISGeoLocate Output . . . . .	49
IISGeoLocate References . . . . .	49
<b>JLECmd</b> . . . . .	50
JLECmd Introduction . . . . .	50
JLECmd Switches . . . . .	51
JLECmd Command Examples . . . . .	58
JLECmd Output . . . . .	59
JLECmd Sample Output . . . . .	66
JLECmd Key Takeaways . . . . .	71
JLECmd References . . . . .	72
<b>LECmd</b> . . . . .	73
LECmd Introduction . . . . .	73
LECmd Switches . . . . .	74
LECmd Command Examples . . . . .	77
LECmd Sample Output . . . . .	79
LECmd Output . . . . .	80
LECmd Key Takeaways . . . . .	88
LECmd References . . . . .	90
<b>MFTECmd</b> . . . . .	91
MFTECmd Introduction . . . . .	91
File Types Parsed by MFTECmd . . . . .	91
MFTECmd Switches . . . . .	93
MFTECmd Command Examples . . . . .	100
MFTECmd Output . . . . .	102

## CONTENTS

MFTECmd References . . . . .	106
<b>PECmd . . . . .</b>	<b>107</b>
PECmd Introduction . . . . .	107
PECmd Switches . . . . .	108
PECmd Command Examples . . . . .	112
PECmd Output . . . . .	113
PECmd Key Takeaways . . . . .	114
PECmd References . . . . .	115
<b>RBCmd . . . . .</b>	<b>116</b>
RBCmd Introduction . . . . .	116
RBCmd Switches . . . . .	117
RBCmd Command Examples . . . . .	117
RBCmd Output . . . . .	119
RBCmd Key Takeaways . . . . .	119
RBCmd References . . . . .	120
<b>RecentFileCacheParser . . . . .</b>	<b>121</b>
RecentFileCacheParser Introduction . . . . .	121
RecentFileCacheParser Switches . . . . .	122
RecentFileCacheParser Command Examples . . . . .	122
RecentFileCacheParser Output . . . . .	123
RecentFileCacheParser References . . . . .	125
<b>RECcmd . . . . .</b>	<b>126</b>
RECcmd Introduction . . . . .	126
RECcmd Switches . . . . .	127
RECcmd Command Examples . . . . .	133
RECcmd Output . . . . .	134
RECcmd References . . . . .	135
<b>RLA . . . . .</b>	<b>137</b>
RLA Introduction . . . . .	137
RLA Switches . . . . .	138
RLA Command Examples . . . . .	138
RLA References . . . . .	139
<b>SBECmd . . . . .</b>	<b>140</b>
SBECmd Introduction . . . . .	140
SBECmd Switches . . . . .	141
SBECmd Command Examples . . . . .	143
SBECmd Output . . . . .	144
SBECmd Key Takeaways . . . . .	144

## CONTENTS

SBECmd References . . . . .	145
<b>SQLECmd . . . . .</b>	<b>146</b>
SQLECmd Introduction . . . . .	146
SQLECmd Switches . . . . .	147
SQLECmd Command Examples . . . . .	149
SQLECmd References . . . . .	150
<b>SrumECmd . . . . .</b>	<b>151</b>
SrumECmd Introduction . . . . .	151
SrumECmd Switches . . . . .	152
SrumECmd Command Examples . . . . .	153
SrumECmd Output . . . . .	154
SrumECmd Sample Data . . . . .	154
SrumECmd References . . . . .	155
<b>SumECmd . . . . .</b>	<b>156</b>
SumECmd Introduction . . . . .	156
SumECmd Switches . . . . .	157
SumECmd Command Examples . . . . .	157
SumECmd Output . . . . .	158
SumECmd References . . . . .	159
<b>VSCMount . . . . .</b>	<b>160</b>
VSCMount Introduction . . . . .	160
VSCMount Switches . . . . .	161
VSCMount Command Examples . . . . .	164
VSCMount References . . . . .	164
<b>WxTCmd . . . . .</b>	<b>165</b>
WxTCmd Introduction . . . . .	165
WxTCmd Switches . . . . .	166
WxTCmd Command Examples . . . . .	166
WxTCmd Output . . . . .	166
WxTCmd Key Takeaways . . . . .	168
WxTCmd References . . . . .	169
<b>EZ Tools - GUI . . . . .</b>	<b>170</b>
<b>EZViewer . . . . .</b>	<b>171</b>
EZViewer Introduction . . . . .	171
EZViewer Screenshot . . . . .	172
EZViewer Key Takeaways . . . . .	172
EZViewer References . . . . .	174

## CONTENTS

<b>Hasher</b> . . . . .	175
Hasher Introduction . . . . .	175
Hasher Screenshot . . . . .	175
Hasher Features . . . . .	176
Hasher References . . . . .	180
<b>JumpList Explorer</b> . . . . .	182
JumpList Explorer Introduction . . . . .	182
JumpList Explorer Functionality . . . . .	184
JumpList Explorer References . . . . .	189
<b>MFT Explorer</b> . . . . .	190
MFT Explorer Introduction . . . . .	190
MFT Explorer Features . . . . .	191
MFT Explorer References . . . . .	192
<b>Registry Explorer</b> . . . . .	193
Registry Explorer Introduction . . . . .	193
RECcmd . . . . .	264
Version changes . . . . .	271
<b>SDB Explorer</b> . . . . .	278
SDB Explorer Introduction . . . . .	278
SDB Explorer References . . . . .	280
<b>Shellbags Explorer</b> . . . . .	281
Requirements . . . . .	281
What are ShellBags? . . . . .	281
ShellBags location in the registry . . . . .	282
Using RegEdit to view ShellBag data . . . . .	282
Why another ShellBags program? . . . . .	290
ShellBagsExplorer.exe . . . . .	292
Menus . . . . .	294
Workflow overview . . . . .	296
SBECmd.exe . . . . .	314
General usage tips and tricks . . . . .	319
Version changes . . . . .	320
<b>TimeApp</b> . . . . .	324
TimeApp Introduction . . . . .	324
TimeApp Screenshots . . . . .	324
TimeApp References . . . . .	325
<b>Timeline Explorer</b> . . . . .	326
Timeline Explorer Introduction . . . . .	326

## CONTENTS

Timeline Explorer Features . . . . .	327
Timeline Explorer Settings . . . . .	348
Timeline Explorer Layout Files . . . . .	349
Timeline Explorer Plugins . . . . .	350
Timeline Explorer References . . . . .	355
<b>XWFIM . . . . .</b>	<b>356</b>
Using XWFIM . . . . .	357
XWFIM References . . . . .	367
<b>Errata . . . . .</b>	<b>368</b>
Reporting Errata . . . . .	368

# Enabling Update Notifications on Leanpub

Please, before you go any further in reading this book, enable update notifications for this book (and any others) so you're notified when this book is updated! This can be done in [your Leanpub library](#)<sup>1</sup>. This book is a live document and will constantly be updated as time goes on. You don't want to miss out!



**EZ Tools Manuals**  
Andrew Rathbun and Eric Zimmerman  
  
This book is 85% complete  
LAST UPDATED ON 2022-10-22

---

**Read this book**

-  [PDF](#) (For reading on a computer)
-  [EPUB](#) (For reading on phones and tablets)
-  [Read on Leanpub](#) (Read in your browser)
-  [Send to Kindle](#) (Send to your Kindle devices or apps)

**Email Settings**

**New version available**  
Email me when the author publishes a new version of this book.

**Share my email address with the author**  
This will allow the author to contact you directly.

**Actions**

**Archive this book**  
Hide this book and move this book to your archive. [Archive](#)

**Email me a receipt**  
Send a receipt to [@gmail.com.](mailto:@gmail.com) [Request Receipt](#)

**Request a refund**  
This purchase was free and cannot be refunded. [Request Refund](#)

**Update Notifications**

<sup>1</sup>[https://leanpub.com/user\\_dashboard/library](https://leanpub.com/user_dashboard/library)

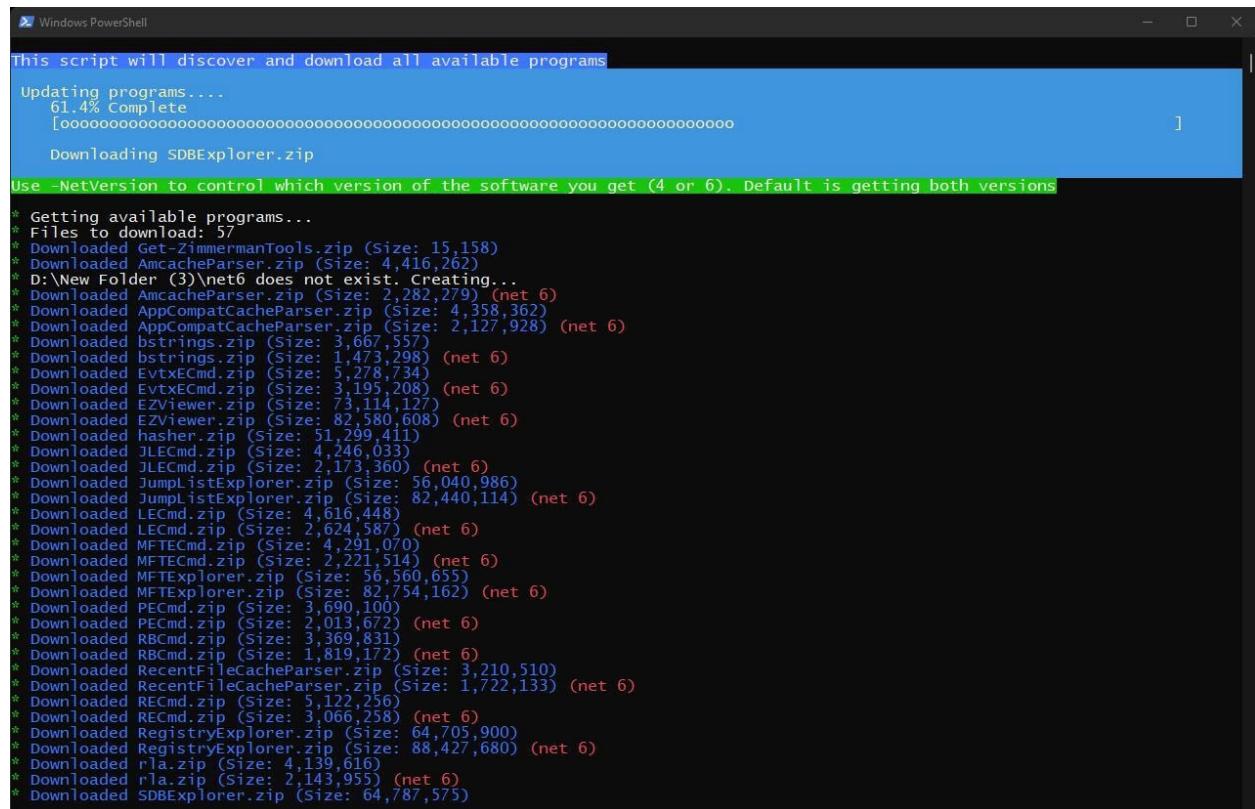
# Introduction to EZ Tools

## What are EZ Tools?

EZ Tools are free and open-source digital forensics tools written by [Eric Zimmerman](#)<sup>2</sup>.

## Download EZ Tools

Eric Zimmerman's Tools can be downloaded [here](#)<sup>3</sup>. Run the `Get-ZimmermanTools.ps1` PowerShell script to download EZ Tools.



```
Windows PowerShell
This script will discover and download all available programs

Updating programs....
61.4% Complete
[oooooooooooooooooooooooooooooooooooooooooooo]

Downloading SDBExplorer.zip

Use -NetVersion to control which version of the software you get (4 or 6). Default is getting both versions

* Getting available programs...
* Files to download: 57
* Downloaded Get-ZimmermanTools.zip (Size: 15,158)
* Downloaded AmcacheParser.zip (Size: 4,416,262)
* D:\New Folder (3)\net6 does not exist. Creating...
* Downloaded AmcacheParser.zip (Size: 2,282,279) (net 6)
* Downloaded AppCompatCacheParser.zip (size: 4,358,362)
* Downloaded AppCompatCacheParser.zip (Size: 2,127,928) (net 6)
* Downloaded bstrings.zip (Size: 3,667,557)
* Downloaded bstrings.zip (Size: 1,473,298) (net 6)
* Downloaded EvtxECmd.zip (Size: 5,278,734)
* Downloaded EvtxECmd.zip (Size: 3,195,208) (net 6)
* Downloaded EZViewer.zip (Size: 73,114,127)
* Downloaded EZViewer.zip (Size: 82,580,608) (net 6)
* Downloaded hasher.zip (Size: 51,299,411)
* Downloaded JLECmd.zip (Size: 4,246,033)
* Downloaded JLECmd.zip (Size: 2,173,360) (net 6)
* Downloaded JumpListExplorer.zip (Size: 56,040,986)
* Downloaded JLECmd.zip (Size: 82,440,114) (net 6)
* Downloaded LECmd.zip (Size: 4,616,448)
* Downloaded LECmd.zip (Size: 2,624,587) (net 6)
* Downloaded MFTECmd.zip (Size: 4,291,070)
* Downloaded MFTECmd.zip (Size: 2,221,514) (net 6)
* Downloaded MFTExplorer.zip (Size: 56,560,655)
* Downloaded MFTExplorer.zip (Size: 82,754,162) (net 6)
* Downloaded PEcmd.zip (Size: 3,690,100)
* Downloaded PEcmd.zip (Size: 2,013,672) (net 6)
* Downloaded RBcmd.zip (Size: 3,369,831)
* Downloaded RBcmd.zip (Size: 1,819,172) (net 6)
* Downloaded RecentFileCacheParser.zip (Size: 3,210,510)
* Downloaded RecentFileCacheParser.zip (Size: 1,722,133) (net 6)
* Downloaded RECmd.zip (Size: 5,122,256)
* Downloaded RECmd.zip (Size: 3,066,258) (net 6)
* Downloaded RegistryExplorer.zip (Size: 64,705,900)
* Downloaded RegistryExplorer.zip (Size: 88,427,680) (net 6)
* Downloaded rla.zip (Size: 4,139,616)
* Downloaded rla.zip (Size: 2,143,955) (net 6)
* Downloaded SDBExplorer.zip (Size: 64,787,575)
```

<sup>2</sup><https://www.sans.org/profiles/eric-zimmerman/>

<sup>3</sup><https://ericzimmerman.github.io/#index.md>

## CLI vs GUI

EZ Tools comprise of CLI (Command Line Interface) and GUI (Graphical User Interface) tools. The CLI tools will be covered first followed by the GUI tools. Generally speaking, the CLI tools are updated more often and are the preferable method of parsing the respective artifact the tool is designed to parse.

## .NET 4 vs .NET 6 EZ Tools

The main difference between the .NET 4 version and .NET 6 version of EZ Tools is the speed and cross-platform compatibility. To compare the speed between .NET 4 and .NET 6 versions of EZ Tools, check out the [Benchmarks<sup>4</sup>](#) on the EZ Tools site.

.NET 6 allows for EZ Tools (and any other application) to run on Linux and macOS in addition to Windows. In order to run both the CLI and GUI tools, download the latest version of the .NET Desktop Runtime [here<sup>5</sup>](#).

## What is this book?

This book serves as the official manuals for EZ Tools. This manual does not cover KAPE as KAPE has its own manual which can be found [here<sup>6</sup>](#).

This book was made on GitHub and published through Leanpub. The GitHub repository for this book's manuscripts is [here<sup>7</sup>](#). This book is open-source and can be improved by anyone! Please suggest improvements by submitting an [Issue<sup>8</sup>](#) or propose changes by submitting a [Pull Request<sup>9</sup>](#).

## Mastering EZ Tools

The best way to master EZ Tools, or any tools for that matter, is to use them. Use tools against sample images found on [CFReDS<sup>10</sup>](#), [Digital Corpora<sup>11</sup>](#), or [AboutDFIR<sup>12</sup>](#). Additionally, the [DFIRArtifactMuseum<sup>13</sup>](#) will be referenced often throughout this manual as there are multiple samples of the raw artifacts that each EZ Tool is designed to parse.

<sup>4</sup><https://ericzimmerman.github.io/#!benchmarks.md>

<sup>5</sup><https://dotnet.microsoft.com/en-us/download/dotnet/6.0>

<sup>6</sup><https://ericzimmerman.github.io/KapeDocs/#index.md>

<sup>7</sup><https://github.com/EZToolsManuals/EZToolsManuals>

<sup>8</sup><https://github.com/EZToolsManuals/EZToolsManuals/issues>

<sup>9</sup><https://github.com/EZToolsManuals/EZToolsManuals/pulls>

<sup>10</sup><https://cfreds.nist.gov/>

<sup>11</sup><https://digitalcorpora.org/>

<sup>12</sup><https://aboutdfir.com/resources/tool-testing/>

<sup>13</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum>

## Content by Eric Zimmerman

- A Guide to Eric Zimmerman's command line tools (EZ Tools)<sup>14</sup>
- Behind The Incident Eric Zimmerman<sup>15</sup>
- DFIR Summit 2016: Plumbing the Depths - Windows Registry Internals<sup>16</sup>
- Exploring Registry Explorer<sup>17</sup>
- Forensic Lunch 7/3/15 with Eric Zimmerman and more<sup>18</sup>
- Forensic Lunch 3/8/19 Eric Zimmerman, Lee Whitfield , Kape, Forensic 4Cast, Nominations<sup>19</sup>
- From Tool Building to Scalable Automation - SANS DFIR Summit 2019 Keynote<sup>20</sup>
- KAPE + EZ Tools and Beyond - OSDFCon 2019 - Eric Zimmerman<sup>21</sup>
- Oxygen Forensics Episode 114<sup>22</sup>
- Plumbing the Depths: ShellBags - SANS DFIR SUMMIT<sup>23</sup>

## Content by the DFIR Community about EZ Tools

- Enabling KAPE at Scale<sup>24</sup>
- Fast, Scalable Results with EZ Tools and the New Command line poster<sup>25</sup>
- Triage Collection and Timeline Analysis with KAPE<sup>26</sup>
- Eric Zimmerman's Results in Seconds at the Command-Line Poster<sup>27</sup>
- EZ Tools/KAPE: How to Contribute to and Benefit from Open Source Contributions<sup>28</sup>
- Enhancing Event Log Analysis with EvtxEcmand using KAPE<sup>29</sup>
- How to Use KAPE and SQLECcmd with EventTranscript.db<sup>30</sup>

---

<sup>14</sup><https://youtu.be/GhCZfCzn2l0>

<sup>15</sup><https://youtu.be/luiLGzm7k34>

<sup>16</sup><https://youtu.be/bsWLg1fWelk>

<sup>17</sup><https://youtu.be/x5mUYqnh00>

<sup>18</sup><https://youtu.be/NKyczOFyykc>

19<https://youtu.be/Lwu1Deb6-xg>20<https://youtu.be/RIDNVRcDuAY>21<https://youtu.be/ZCj7cbWwUOs>22<https://youtu.be/qbFBmAJlbIU>23<https://youtu.be/bWxbfARqBPY>24<https://youtu.be/YF-jDoh8BFM>25<https://youtu.be/yEmLuj3oDzs>26<https://www.youtube.com/watch?v=iYyWZSNBNcw>27<https://www.sans.org/posters/eric-zimmermans-results-in-seconds-at-the-command-line-poster/>28<https://www.youtube.com/watch?v=mIb1GQP3ciE>29<https://www.youtube.com/watch?v=BIkyWexMF0I>30<https://www.youtube.com/watch?v=DoVStoCJrog>

# **EZ Tools - Common Switches**

## **Common Switches**

While each of the EZ Tools provide unique functionality based on the artifact they were developed specifically to parse, most EZ Tools share common switches that will be covered here so they're not repeated in each EZ Tools' respective Switches section.

### **-f**

Use **-f** to point the tool to a single file to be parsed.

#### **Examples**

```
.\JLECmd.exe -f D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\573770283dc3d854.automations  
.LECMsg.exe -f C:\temp\test.lnk  
.MFTECmd.exe -f 'C:\temp\webinar6152022\tout\C\$MFT'
```

### **-d**

Use **-d** to point to a directory for LECmd to parse. Please note, this switch does work recursively.

#### **Examples**

```
.\LECMsg.exe -d C:\temp
```

### **--csv**

This switch will instruct LECmd to output parsed data into a CSV at the specified location.

#### **Examples**

```
.\LECMsg.exe -d C:\temp\LNKFiles --csv C:\temp\LNKFiles\output  
.JLECmd.exe -d C:\temp\JumpLists --csv C:\temp\JumpLists\output  
.MFTECmd.exe -f 'C:\some\random\$MFT' --csv C:\temp\output
```

**--csvf**

This switch will instruct LECmd to output parsed data into a CSV at the specified location and name it using the value specified for this switch.

**Examples**

```
. \LECmd.exe -d C:\temp\LNKFiles --csv C:\temp\LNKFiles\output --csvf LNKFileOutput.csv  
. \JLECmd.exe -d C:\temp\JumpLists -csv C:\temp\JumpLists\output -csvf JumpListOutput.csv
```

**--xml**

This switch will instruct LECmd to output parsed data into XML at the specified location. Please note, this will output an XML file for each .LNK file parsed.

**Examples**

```
. \LECmd.exe -d C:\temp\LNKFiles --xml C:\temp\LNKFiles\output
```

**--html**

This switch will instruct LECmd to output parsed data into XML at the specified location.

**Examples**

```
. \LECmd.exe -d C:\temp\LNKFiles --html C:\temp\LNKFiles\output  
. \JLECmd.exe -d C:\temp\JumpLists --html C:\temp\JumpLists\output
```

**--json**

This switch will instruct LECmd to output parsed data into JSON at the specified location. Please note, this will output a single JSON file regardless of the number of .LNK files parsed.

**Examples**

```
. \LECmd.exe -d C:\temp\LNKFiles --json C:\temp\LNKFiles\output  
. \JLECmd.exe -d C:\temp\JumpLists --json C:\temp\JumpLists\output
```

**--pretty**

This switch will instruct LECmd to output parsed data into pretty printed JSON at the specified location.

## Examples

```
. \LECmd.exe -d C:\temp\LNKFiles --json C:\temp\LNKFiles\output --pretty  
. \JLECmd.exe -d C:\temp\JumpLists --json C:\temp\JumpLists\output --pretty
```

### --dt

This switch will instruct AmcacheParser to display timestamps in a specified format. The options for timestamp format can be found here: <https://docs.microsoft.com/en-us/dotnet/standard/base-types/custom-date-and-time-format-strings?redirectedfrom=MSDN>

### --dedupe

This switch informs the tool to deduplicate results parsed from by the tool. That way, your output won't include multiple identical rows for the leanest output possible.

### --vss

This switch informs the tool to mount Volume Shadow Copies from the drive letter specified using the -f or -d switch and parse Prefetch files present within.

## Examples

```
recmd  
pecmd
```

### -q

This switch hides the processing details when running a command. This means potentially faster generated output since your computer doesn't have to worry about generating output to the console window during processing.

Example: RECcmd.exe -d C:\Windows\system32\config --bn BatchExamples\Kroll\_Batch.reb --nl false --csv C:\temp -q

### --debug

This switch will provide log messages that are helpful for debugging code. --debug can be added to any command for more verbose messaging.

### --trace

This switch will provide log messages that are helpful for tracing the execution of code. --trace can be added to any command for more verbose messaging.

**--version**

This switch will display the current version of the EZ Tool binary.

**--help**

This switch will print the same help information as executing the EZ Tool from the command line without any switches.

# EZ Tools - PowerShell vs CMD

There are some important scenarios to consider when using PowerShell vs Command Prompt.

## Common Scenarios

When using MFTECmd, using the -f switch will require you to point to a file that has a dollar sign (\$) in the filename, such as \$MFT, \$J, etc. If you are using PowerShell when running MFTECmd, you will need to use single backticks instead of quotation marks around your file path.

For those not familiar with PowerShell, variables are assigned using dollar signs. For instance, '\$variable' can be assigned any value which can be used throughout a script without having to retype a value repeatedly.

Here's an example of this:

```
$filePath = C:\User\TestUser\Desktop\Folder\AnotherFolder\LongPath\LongerPath\filename.exe
```

We are assigning that very long file path to the \$filePath variable. Therefore, in a script, we could simple use \$filePath every time we want to inject that file path into our script. Additionally, it allows for us to declare the file path once in the script and the corrected file path is reflected wherever that variable is used.

How this relates to MFTECmd's usage is when pointing to a \$MFT file, if you use quotation marks, PowerShell will be expecting a value to have been assigned to \$MFT or \$J. When running a one-liner command with MFTECmd, we are not declaring variables called \$MFT or \$J, rather, we're pointing to files that have those specific names.

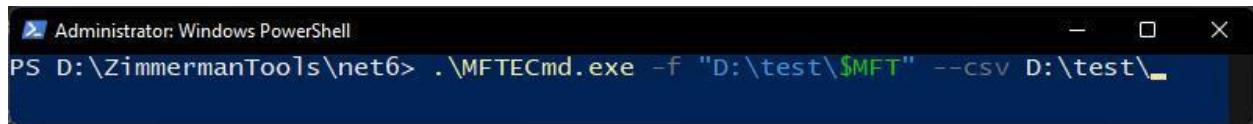
Here is an example of a bad command using PowerShell:

```
.\MFTECmd.exe -f "C:\temp\$someMFT" --csv C:\temp
```

Here is an example of a correct command using PowerShell:

```
.\MFTECmd.exe -f 'C:\temp\$someMFT' --csv C:\temp
```

You will notice a difference in the colors



The screenshot shows a Windows PowerShell window titled 'Administrator: Windows PowerShell'. The command entered is '.\MFTECmd.exe -f 'C:\temp\\$someMFT' --csv C:\temp'. The output of the command is visible below the command line.



```
Administrator: Windows PowerShell
PS D:\ZimmermanTools\net6> .\MFTECmd.exe -f 'D:\test\$MFT' --csv D:\test\
```

Only when you run the command with single backticks will you have output generated with MFTECmd.

# EZ Tools - CLI

Name	Size	Type	Modified	Attr
EvtxECmd		File Folder	Today 22:26	-----
EZViewer		File Folder	Today 22:26	-----
Hasher		File Folder	Today 22:26	-----
iisGeolocate		File Folder	Today 22:29	-----
JumpListExplorer		File Folder	Today 22:26	-----
MFTExplorer		File Folder	Today 22:26	-----
net6		File Folder	Today 22:29	-----
RECmd		File Folder	Today 22:27	-----
RegistryExplorer		File Folder	Today 22:27	-----
SDBExplorer		File Folder	Today 22:27	-----
ShellBagsExplorer		File Folder	Today 22:28	-----
SQLECmd		File Folder	Today 22:28	-----
TimelineExplorer		File Folder	Today 22:28	-----
XWFIM		File Folder	Today 22:29	-----
!!!RemoteFileDetails.csv	8.46 KB	Microsoft Excel Com...	Today 22:29	-a-----
AmcacheParser.exe	4.45 MB	Application	2022-07-06 10:09	-a-----
AppCompatCacheParser.exe	4.31 MB	Application	2022-01-21 11:28	-a-----
bstrings.exe	3.90 MB	Application	2022-05-20 12:38	-a-----
ChangeLog.txt	31.7 KB	Text document	Today 22:29	-a-----
Get-ZimmermanTools.ps1	31.2 KB	PowerShell Script File	2022-01-22 22:58	-a-----
JLECmd.exe	4.59 MB	Application	2022-07-25 16:18	-a-----
LECmd.exe	4.87 MB	Application	2022-06-15 10:53	-a-----
MFTECmd.exe	4.29 MB	Application	Wednesday 11:23	-a-----
PECmd.exe	3.79 MB	Application	2022-01-28 12:08	-a-----
RBCCmd.exe	3.52 MB	Application	2022-08-05 13:05	-a-----
RecentFileCacheParser.exe	3.20 MB	Application	2022-06-15 11:03	-a-----
rla.exe	4.09 MB	Application	2022-07-16 10:54	-a-----
SBECmd.exe	4.68 MB	Application	2022-02-23 08:38	-a-----
SrumECmd.exe	4.33 MB	Application	2022-02-06 12:57	-a-----
SumECmd.exe	3.60 MB	Application	2022-01-21 08:53	-a-----
TimeApp.exe	339 KB	Application	2022-01-24 14:46	-a-----
VSCMount.exe	3.17 MB	Application	2022-01-21 21:38	-a-----
WxTCmd.exe	5.43 MB	Application	2021-05-27 09:10	-a-----

# **AmcacheParser**

## **AmcacheParser Introduction**

AmcacheParser is a tool created by Eric Zimmerman used to parse Amcache.hve files, commonly found at C:\Windows\appcompat\Programs\Amcache.hve.

## **AmcacheParser Use Cases**

### **Law Enforcement**

For those in Law Enforcement, this tool is useful for parsing the Amcache.hve artifact which can help provide insight as to which programs were installed on a computer. Additionally, the artifact can provide hardware information which may be useful context to have for the purpose of evidence identification.

### **Private Sector**

For those in the Private Sector, this tool is useful for parsing the Amcache.hve artifact which can help provide evidence of program execution. Unassociated entries will often be where malicious applications can be found along with the associated SHA1 hash value which can then be leveraged into a resource like VirusTotal to learn more about the application.

## AmcacheParser Switches

In a PowerShell window, running .\AmcacheParser.exe will provide the following options when running AmcacheParser:

```

-f <f> (REQUIRED)           Amcache.hve file to parse
-i                           Include file entries for Programs entries [default: False]
-w <w>                      Path to file containing SHA-1 hashes to *exclude* from the\
results. Blacklisting overrides
                               whitelisting
-b <b>                      Path to file containing SHA-1 hashes to *include* from the\
results. Blacklisting overrides
                               whitelisting
--csv <csv> (REQUIRED)      Directory to save CSV formatted results to. Be sure to inc\
lude the full path in double quotes
--csvf <csvf>                File name to save CSV formatted results to. When present, \
overrides default name
--dt <dt>                    The custom date/time format to use when displaying time st\
amps. See https://goo.gl/CNVq0k for
                               options [default: yyyy-MM-dd HH:mm:ss]
--mp                         Display higher precision for time stamps [default: False]
--nl                         When true, ignore transaction log files for dirty hives. D\
efault is FALSE
                               [default: False]
--debug                      Show debug information during processing [default: False]
--trace                      Show trace information during processing [default: False]
--version                     Show version information
-?, -h, --help                Show help and usage information

```

## Switch Descriptions

### -i

This switch will instruct AmcacheParser to include output for ProgramEntries.

Example: .\AmcacheParser.exe -f "D:\DFIRArtifactMuseum\Windows\Amcache\Win10\APTSimulatorVM\Amcache.hve" --csv D:\Amcache -i

### -w

This switch will instruct AmcacheParser to read a specified file containing SHA-1 hashes to exclude from the results. This means that hashes specified in this file will NOT be included in the CSV output.

Example: .\AmcacheParser.exe -f "D:\DFIRArtifactMuseum\Windows\Amcache\Win10\APTSimulatorVM\Amcache.hve" --csv D:\Amcache -w D:\Amcache\Hashes.txt

**-b**

This switch will instruct AmcacheParser to read a specified file containing SHA-1 hashes to include from the results. This means that hashes specified in this file will be the ONLY results included in the CSV output.

Example: .\AmcacheParser.exe -f "D:\DFIRArtifactMuseum\Windows\Amcache\Win10\APTSimulatorVM\Amcache.hve" --csv D:\Amcache -b D:\Amcache\Hashes.txt

**--mp**

This switch will instruct AmcacheParser to provide more verbose timestamps. For instance, running .\AmcacheParser.exe -f "D:\Amcache\Amcache.hve" --csv D:\Amcache resulted in the following:

```
File Key Last Write Timestamp
2022-03-20 21:24:56.000000
```

Running .\AmcacheParser.exe -f "D:\DFIRArtifactMuseum\Windows\Amcache\Win10\APTSimulatorVM\Amcache.hve" --csv D:\Amcache --mp resulted in the following:

```
File Key Last Write Timestamp
2022-03-20 21:24:56.028214
```

**--nl**

This switch will instruct AmcacheParser to ignore transaction logs (\*.LOG files)

Example: .\AmcacheParser.exe -f "D:\DFIRArtifactMuseum\Windows\Amcache\Win10\APTSimulatorVM\Amcache.hve" --csv D:\Amcache --nl

# AmcacheParser Command Examples

## Example AmcacheParser Commands

### Parse an Amcache.hve File and Output to CSV

```
AmcacheParser.exe -f "C:\Temp\amcache\AmcacheWin10.hve" --csv C:\temp
```

### Parse an Amcache.hve file and Output to CSV (specified filename) with ProgramEntries

```
AmcacheParser.exe -f "C:\Temp\amcache\AmcacheWin10.hve" -i on --csv C:\temp --csvf fo.csv
```

### Parse an Amcache.hve file and Output to CSV While Only Including Results That Match Specified Hashes

```
AmcacheParser.exe -f "C:\Temp\amcache\AmcacheWin10.hve" -w "c:\temp\whitelist.txt" -c csv C:\temp
```

## AmcacheParser Output

### Analyzing AmcacheParser Output - CSV

--csv is required, so there's no console output with AmcacheParser

AmcacheParser will produce the following CSVs:

```
%timestamp%_Amcache_AssociatedFileEntries.csv  
%timestamp%_Amcache_DeviceContainers.csv  
%timestamp%_Amcache_DevicePnps.csv  
%timestamp%_Amcache_DriveBinaries.csv  
%timestamp%_Amcache_DriverPackages.csv  
%timestamp%_Amcache_ProgramEntries.csv  
%timestamp%_Amcache_ShortCuts.csv  
%timestamp%_Amcache_UnassociatedFileEntries.csv
```

%timestamp% will look something similar to this: 20220904032628

## AmcacheParser Key Takeaways

### Important Data Points

The Amcache.hve will provide SHA1 hash values for binaries. This can be helpful in Incident Response engagements where you can leverage the SHA1 hash value for something like VirusTotal to learn more about a potentially malicious binary. Please note that often malicious binaries introduced by threat actors will be found in the UnassociatedEntries CSV. These will be programs that don't have program entries within the Amcache. This is an easy way to find malware because malware doesn't come with an installer! Additionally, the File Key ID Last Write Timestamp can be a reliable indicator of evidence of execution.

If you want to analyze the Amcache.hve artifact in a GUI tool, you can simply ingest it into Registry Explorer similar to other common Registry hives.

# AmcacheParser References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/AmcacheParser> is the AmcacheParser repo

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- Locked file support added to AmcacheParser, AppCompatCacheParser, MFTECmd, ShellBags Explorer (and SBECmd), and Registry Explorer (and RECmd)<sup>31</sup>
- Everything gets an update, Sept 2018 edition<sup>32</sup>
- Updates to the left of me, updates to the right of me, version 1 releases are here (for the most part)<sup>33</sup>
- (Am)cache still rules everything around me (part 2 of 1)<sup>34</sup>

## Download AmcacheParser

AmcacheParser can be downloaded from <https://ericzimmerman.github.io/#!index.md>

## Amcache Sample Data

Sample Amcache.hve artifacts can be found at the [DFIRArtifactMuseum](#)<sup>35</sup>.

---

<sup>31</sup><https://binaryforay.blogspot.com/2019/01/locked-file-support-added-to.html>

<sup>32</sup><https://binaryforay.blogspot.com/2018/09/everything-gets-update-sept-2018-edition.html>

<sup>33</sup><https://binaryforay.blogspot.com/2018/03/updates-to-left-of-me-updates-to-right.html>

<sup>34</sup><https://binaryforay.blogspot.com/2017/10/amcache-still-rules-everything-around.html>

<sup>35</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/Amcache>

# **AppCompatCacheParser**

## **AppCompatCacheParser Introduction**

AppCompatCacheParser is a tool created by Eric Zimmerman used to parse the AppCompatCache, commonly referred to as the ShimCache, which can be found in the SYSTEM hive.

## **AppCompatCacheParser Use Cases**

### **Law Enforcement**

For those in Law Enforcement, this tool is useful for providing evidence of file knowledge. Artifacts seen in the AppCompatCache, depending on the version of Windows, may provide indicators of an executable existing on the file system at a given point in time.

### **Private Sector**

For those in the Private Sector, this tool is useful for providing evidence of file knowledge. Artifacts seen in the AppCompatCache, depending on the version of Windows, may provide indicators of an executable existing on the file system at a given point in time. Additionally, previous versions of Windows had indicators of program execution but more reliably than same information can be found in Prefetch, UserAssist, etc.

## AppCompatCacheParser Switches

In a PowerShell window, running .\AppCompatCacheParser.exe will provide the following options when running AppCompatCacheParser:

```

-f <f>           Full path to SYSTEM hive to process. If this option is not
                  specified, the live Registry will be used
--csv <csv> (REQUIRED) Directory to save CSV formatted results to. Be sure to include the
                  full path in double quotes
--csvf <csvf>   File name to save CSV formatted results to. When present, \
                  overrides
                  default name
--c <c>          The ControlSet to parse. Default is to extract all control\
                  sets
                  [default: -1]
-t               Sorts last modified timestamps in descending order [default\
                  : False]
--dt <dt>        The custom date/time format to use when displaying time st\
                  amps. See
                  https://goo.gl/CNVq0k for options [default: yyyy-MM-dd HH:\\
                  mm:ss]
--nl             When true, ignore transaction log files for dirty hives [default:
                  False]
--debug         Show debug information during processing [default: False]
--trace          Show trace information during processing [default: False]
--version        Show version information
-?, -h, --help  Show help and usage information

```

## Switch Descriptions

### -c

This switch informs the tool to parse a specific ControlSet. The default is to parse all ControlSets specified within the Registry.

Example: .\AppCompatCacheParser.exe -f "C:\temp\System" --csv C:\temp\appcompatcachetest  
--c 1

The above command will display the following message in the console window:

```
1 Found 1,024 cache entries for Windows10Creators in ControlSet001
2
3 Results saved to 'C:\temp\appcompatcachetest\20220805212738_Windows10Creators_System\
4 _ControlSet001_AppCompatCache.csv'
```

**-t**

This switch will inform the tool to sort the last modified timestamps in descending order.

Example: .\AppCompatCacheParser.exe -f "D:\temp\SYSTEM" --nl false --csv "D:\temp" -t

**-nl**

This switch will inform the tool whether to replay transaction logs or not.

Below is an example of replaying transaction logs:

Example: .\AppCompatCacheParser.exe -f "D:\temp\SYSTEM" --nl false --csv "D:\temp"

When processing transaction logs, you will see a message similar to this:

```
1 Two transaction logs found. Determining primary log...
2 Primary log: D:\temp\SYSTEM.LOG2, secondary log: D:\temp\SYSTEM.LOG1
3 Replaying log file: D:\temp\SYSTEM.LOG2
4 Replaying log file: D:\temp\SYSTEM.LOG1
5 At least one transaction log was applied. Sequence numbers have been updated to 0x91\
6 DD. New Checksum: 0xAD729723
7 Found 1,024 cache entries for Windows10Creators in ControlSet001
```

Below is an example of ignoring transaction logs:

Example: .\AppCompatCacheParser.exe -f "D:\temp\SYSTEM" --nl true --csv "D:\temp"

When ignoring transaction logs, you will see a message similar to this:

```
1 Registry hive is dirty and transaction logs were found in the same directory, but --\
2 nl was provided. Data may be missing! Continuing anyways...
3 Sequence numbers do not match! Hive is dirty and the transaction logs should be reviewed \
4 for relevant data!
5 Found 1,024 cache entries for Windows10Creators in ControlSet001
```

## AppCompatCacheParser Command Examples

### Example AppCompatCacheParser Commands

**Parse the AppCompatCache from the SYSTEM Registry hive and output to a specified location while only parsing ControlSet2**

```
AppCompatCacheParser.exe --csv c:\temp -c 2
```

**Parse the AppCompatCache from the SYSTEM Registry hive and output to a specified location while outputting a CSV named results.csv**

```
AppCompatCacheParser.exe --csv c:\temp --csvf results.csv
```

## AppCompatCacheParser Output

### Analyzing AppCompatCacheParser Output - CSV

AppCompatCacheParser will output a filename similar to these:

```
%timestamp%_XXXXXX_SYSTEM_AppCompatCache.csv
```

According to [AppCompatCacheParser's code<sup>36</sup>](#), the following versions can replace the XXXXXX:

```
WindowsXP,  
WindowsVistaWin2k3Win2k8,  
Windows7x86,  
Windows7x64_Windows2008R2,  
Windows80_Windows2012,  
Windows81_Windows2012R2,  
Windows10,  
Windows10Creators,  
Unknown
```

---

<sup>36</sup><https://github.com/EricZimmerman/AppCompatCacheParser/blob/95d4e1e084fdf8289175fdb47a15747753ec8e77/AppCompatCache/AppCompatCache.cs#L55>

## **AppCompatCacheParser Key Takeaways**

### **Important Data Points**

The Last Modified timestamp will not be the last time of execution, rather, it'll be the last modified timestamp of the file itself. In some instances, this timestamp could be similar to a file's execution as seen elsewhere in UserAssist, Prefetch, etc. In that scenario, consider the file may have been copied from elsewhere and then executed immediately upon the file copy operation's completion. With the file being copied, the last modified timestamp will reflect the time of the file copy operation's completion, which being immediately followed by executing the file could make the last modified timestamp found in the AppCompatCache seem like a reliable timestamp, in some cases. Ideally, rely on other artifacts for more reliable evidence of execution timestamps.

# AppCompatCacheParser References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/AppCompatCacheParser> is the GitHub repository for AppCompatCacheParser

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- Introducing AppCompatCacheParser<sup>37</sup>
- AppCompatCacheParser v0.0.5.1 released<sup>38</sup>
- AppCompatCacheParser v0.0.5.2 released<sup>39</sup>
- AppCompatCacheParser v0.9.0.0 released and some AppCompatCache/shimcache parser testing<sup>40</sup>
- Windows 10 Creators update vs shimcache parsers: Fight!!<sup>41</sup>
- Updates to the left of me, updates to the right of me, version 1 releases are here (for the most part)<sup>42</sup>
- Everything gets an update, Sept 2018 edition<sup>43</sup>
- Locked file support added to AmcacheParser, AppCompatCacheParser, MFTECmd, ShellBags Explorer (and SBECmd), and Registry Explorer (and RECcmd)<sup>44</sup>

## Community Resources

- 13Cubed - Windows Application Compatibility Forensics<sup>45</sup>
- 13Cubed - Let's Talk About Shimcache - The Most Misunderstood Artifact<sup>46</sup>

## Download AppCompatCacheParser

AppCompatCacheParser can be downloaded from <https://ericzimmerman.github.io/#!index.md>

<sup>37</sup><https://binaryforay.blogspot.com/2015/05/introducing-appcompatcacheparser.html>

<sup>38</sup><https://binaryforay.blogspot.com/2015/05/appcompatcacheparser-v0051-released.html>

<sup>39</sup><https://binaryforay.blogspot.com/2015/05/appcompatcacheparser-v0052-released.html>

<sup>40</sup><https://binaryforay.blogspot.com/2016/05/appcompatcacheparser-v0900-released-and.html>

<sup>41</sup><https://binaryforay.blogspot.com/2017/03/windows-10-creators-update-vs-shimcache.html>

<sup>42</sup><https://binaryforay.blogspot.com/2018/03/updates-to-left-of-me-updates-to-right.html>

<sup>43</sup><https://binaryforay.blogspot.com/2018/09/everything-gets-update-sept-2018-edition.html>

<sup>44</sup><https://binaryforay.blogspot.com/2019/01/locked-file-support-added-to.html>

<sup>45</sup><https://youtu.be/ZKlyu-HOvxY>

<sup>46</sup>[https://youtu.be/7byz1dR\\_CLg](https://youtu.be/7byz1dR_CLg)

## AppCompatCache Sample Data

Sample data for the AppCompatCache can be found in the DFIRArtifactMuseum [here<sup>47</sup>](#).

---

<sup>47</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/AppCompatCache>

# **bstrings**

## **bstrings Introduction**

bstrings is a tool created by Eric Zimmerman used to search files for strings. This can be accomplished by using user-specified search strings or regular expressions or built-in regular expressions.

## **bstrings Use Cases**

### **Law Enforcement**

For those in Law Enforcement, this tool is useful for searching through files for keywords or regular expressions of interest, including but not limited to: credit cards, Social Security numbers, phone numbers, email addresses, cryptocurrency wallets, and many more.

### **Private Sector**

For those in the Private Sector, this tool is useful for searching through files for keywords or regular expressions of interest, including but not limited to: IPv4 addresses, IPv6 addresses, SID numbers, cryptocurrency wallets, and many more.

## bstrings Switches

In a PowerShell window, running .\bstrings.exe will provide the following options when running bstrings:

```
-f <f>          File to search. Either this or -d is required
-d <d>          Directory to recursively process. Either this or -f is required
-o <o>          File to save results to
-a              If set, look for ASCII strings. Use -a false to disable [default: \
True]
-u              If set, look for Unicode strings. Use -u false to disable [default\
: True]
-m <m>          Minimum string length [default: 3]
-b <b>          Chunk size in MB. Valid range is 1 to 1024. Default is 512 [defau\
t: 512]
-q              Quiet mode (Do not show header or total number of hits) [default: \
False]
-s              Really Quiet mode (Do not display hits to console. Speeds up proce\
ssing when
               using -o) [default: False]
-x <x>          Maximum string length. Default is unlimited [default: -1]
-p              Display list of built in regular expressions [default: False]
--ls <ls>        String to look for. When set, only matching strings are returned
--lr <lr>        Regex to look for. When set, only strings matching the regex are r\
eturned
--fs <fs>        File containing strings to look for. When set, only matching strin\
gs are returned
--fr <fr>        Directory to save bodyfile formatted results to. --bd1 is also req\
uired when
               using this option
--ar <ar>        Range of characters to search for in 'Code page' strings. Specify \
as a range of
               characters in hex format and enclose in quotes. Default is [\x20 - \
\x7E]
[default: [ -~]]
--ur <ur>        Range of characters to search for in 'Code page' strings. Specify \
as a range of
               characters in hex format and enclose in quotes. Default is [\x20 - \
\x7E]
[default: [ -~]]
--cp <cp>        Code page to use. Default is 1252. Use the Identifier value for co\
de pages at
```

```
https://goo.gl/ig6DxW [default: 1252]
--mask <mask> When using -d, file mask to search for. * and ? are supported. Thi\
s option has no effect when
--ms <ms> When using -d, maximum file size to process. This option has no ef\
fect when
--ro When true, list the string matched by regex pattern vs string the \
pattern was found in (This may result in duplicate strings in output. ~ denote\
s approx.
--off Show offset to hit after string, followed by the encoding (A=1252,\U=Unicode)
[default: False]
--sa Sort results alphabetically [default: False]
--sl Sort results by length [default: False]
--debug Show debug information during processing [default: False]
--trace Show trace information during processing [default: False]
--version Show version information
-?, -h, --help Show help and usage information
```

## Switch Descriptions

### -o

This switch informs the tool to save the results of a search to a specified location.

Example: `.\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --ls powershell -o "D:\DFIRArtifactMuseum\Windows\results.txt"`

Please note that if you run multiple searches with the `-o` switch, which outputs to a file with a specified filename, subsequent searches after the initial search will append the results to that output file with each search if the output filename is the same for each search.

### -a

This switch informs the tool to look for [ASCII<sup>48</sup>](#) strings. Default is true, so if you want to disable it, use `-a false`.

Example: `.\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --ls powershell -o "D:\DFIRArtifactMuseum\Windows\results.txt" -a false`

### "-u"

This switch informs the tool to look for [Unicode<sup>49</sup>](#) strings. Default is true, so if you want to disable it, use `-u false`.

Example: `.\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --ls powershell -o "D:\DFIRArtifactMuseum\Windows\results.txt" -u false`

### -m

This switch informs the tool to look for strings that are a minimum length of the specified value.

Example: `.\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --ls powershell -o "D:\DFIRArtifactMuseum\Windows\results.txt" -m 12`

### -b

This switch informs the tool to search a specified chunk size. The default is 512 but the valid range is 1 to 1024.

Example: `.\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows\Pagefile\" --ls powershell -o "D:\DFIRArtifactMuseum\Windows\results.txt" -b 10`

Running the above command will result in console messages similar to the below:

<sup>48</sup><https://en.wikipedia.org/wiki/ASCII>

<sup>49</sup><https://en.wikipedia.org/wiki/Unicode>

```
1 Searching 3 chunks (10 MB each) across 22.034 MB in 'D:\DFIRArtifactMuseum\Windows\P\  
2 agefile\Win10\APTSimulatorVM\pagefile.7z'  
3  
4 Chunk 1 of 3 finished. Total strings so far: 311,770 Elapsed time: 0.252 seconds. Av\  
5 erage strings/sec: 1,239,057  
6 Chunk 2 of 3 finished. Total strings so far: 580,462 Elapsed time: 0.649 seconds. Av\  
7 erage strings/sec: 894,890  
8 Chunk 3 of 3 finished. Total strings so far: 624,488 Elapsed time: 0.713 seconds. Av\  
9 erage strings/sec: 875,745  
10 Primary search complete. Looking for strings across chunk boundaries...  
11 Search complete.
```

Change the value to 1 would result in the following messages:

```
1 Searching 23 chunks (1 MB each) across 22.034 MB in 'D:\DFIRArtifactMuseum\Windows\P\  
2 agefile\Win10\APTSimulatorVM\pagefile.7z'  
3  
4 Chunk 1 of 23 finished. Total strings so far: 33,330 Elapsed time: 0.029 seconds. Av\  
5 erage strings/sec: 1,153,463  
6 Chunk 2 of 23 finished. Total strings so far: 66,074 Elapsed time: 0.052 seconds. Av\  
7 erage strings/sec: 1,266,474  
8 Chunk 3 of 23 finished. Total strings so far: 98,578 Elapsed time: 0.082 seconds. Av\  
9 erage strings/sec: 1,195,192  
10 Chunk 4 of 23 finished. Total strings so far: 130,789 Elapsed time: 0.111 seconds. A\  
11 verage strings/sec: 1,178,745  
12 Chunk 5 of 23 finished. Total strings so far: 162,189 Elapsed time: 0.137 seconds. A\  
13 verage strings/sec: 1,186,162  
14 Chunk 6 of 23 finished. Total strings so far: 192,861 Elapsed time: 0.207 seconds. A\  
15 verage strings/sec: 933,171  
16 Chunk 7 of 23 finished. Total strings so far: 223,261 Elapsed time: 0.229 seconds. A\  
17 verage strings/sec: 975,082  
18 Chunk 8 of 23 finished. Total strings so far: 253,191 Elapsed time: 0.260 seconds. A\  
19 verage strings/sec: 974,158  
20 Chunk 9 of 23 finished. Total strings so far: 282,837 Elapsed time: 0.281 seconds. A\  
21 verage strings/sec: 1,004,852  
22 Chunk 10 of 23 finished. Total strings so far: 311,770 Elapsed time: 0.312 seconds. \  
23 Average strings/sec: 998,170  
24 Chunk 11 of 23 finished. Total strings so far: 340,242 Elapsed time: 0.426 seconds. \  
25 Average strings/sec: 798,514  
26 Chunk 12 of 23 finished. Total strings so far: 368,446 Elapsed time: 0.446 seconds. \  
27 Average strings/sec: 825,609  
28 Chunk 13 of 23 finished. Total strings so far: 396,369 Elapsed time: 0.471 seconds. \  
29 Average strings/sec: 840,736
```

```
30 Chunk 14 of 23 finished. Total strings so far: 423,846 Elapsed time: 0.510 seconds. \
31 Average strings/sec: 831,394
32 Chunk 15 of 23 finished. Total strings so far: 450,770 Elapsed time: 0.536 seconds. \
33 Average strings/sec: 840,528
34 Chunk 16 of 23 finished. Total strings so far: 477,418 Elapsed time: 0.560 seconds. \
35 Average strings/sec: 851,896
36 Chunk 17 of 23 finished. Total strings so far: 503,561 Elapsed time: 0.623 seconds. \
37 Average strings/sec: 808,210
38 Chunk 18 of 23 finished. Total strings so far: 529,823 Elapsed time: 0.664 seconds. \
39 Average strings/sec: 798,020
40 Chunk 19 of 23 finished. Total strings so far: 555,164 Elapsed time: 0.691 seconds. \
41 Average strings/sec: 803,042
42 Chunk 20 of 23 finished. Total strings so far: 580,462 Elapsed time: 0.714 seconds. \
43 Average strings/sec: 812,530
44 Chunk 21 of 23 finished. Total strings so far: 605,518 Elapsed time: 0.776 seconds. \
45 Average strings/sec: 780,189
46 Chunk 22 of 23 finished. Total strings so far: 624,488 Elapsed time: 0.821 seconds. \
47 Average strings/sec: 760,773
48 Chunk 23 of 23 finished. Total strings so far: 624,488 Elapsed time: 0.824 seconds. \
49 Average strings/sec: 757,961
50 Primary search complete. Looking for strings across chunk boundaries...
51 Search complete.
```

#### -q

This switch informs the tool to run in quiet mode, where the header and footer won't display as results are being displayed in the console window.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --ls powershell -o "D:\DFIRArtifactMuseum\Windows\results.txt" -a false -q

#### -x

This switch informs the tool to search for strings with a maximum length of a specified value.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --ls powershell -o "D:\DFIRArtifactMuseum\Windows\results.txt" -x 15

#### -p

This switch informs the tool to display the list of regular expressions built in to the tool.

Example: .\bstrings.exe -p

This command will display the following information:

	Name	Description
1	aeon	Finds Aeon wallet addresses
2	b64	Finds valid formatted base 64 strings
4	bitcoin	Finds BitCoin wallet addresses
5	bitlocker	Finds Bitlocker recovery keys
6	bytecoin	Finds ByteCoin wallet addresses
7	cc	Finds credit card numbers
8	dashcoin	Finds DashCoin wallet addresses (D*)
9	dashcoin2	Finds DashCoin wallet addresses (7 X)*
10	email	Finds embedded email addresses
11	fantomcoin	Finds Fantomcoin wallet addresses
12	guid	Finds GUIDs
13	ipv4	Finds IP version 4 addresses
14	ipv6	Finds IP version 6 addresses
15	mac	Finds MAC addresses
16	monero	Finds Monero wallet addresses
17	reg_path	Finds paths related to Registry hives
18	sid	Finds Microsoft Security Identifiers (SID)
19	ssn	Finds US Social Security Numbers
20	sumokoin	Finds SumoKoin wallet addresses
21	unc	Finds UNC paths
22	url3986	Finds URLs according to RFC 3986
23	urlUser	Finds usernames in URLs
24	usPhone	Finds US phone numbers
25	var_set	Finds environment variables being set (OS=Windows_NT)
26	win_path	Finds Windows style paths (C:\folder1\folder2\file.txt)
27	xml	Finds XML/HTML tags
28	zip	Finds zip codes
29		
30		To use a built in pattern, supply the Name to the --lr switch

**--ls**

This switch informs the tool to search for a specified string.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --ls powershell

**--lr**

This switch informs the tool to search for a specified built-in regular expression.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --lr ipv4

**--fs**

This switch informs the tool to use a specified list of search terms within a specified file.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --fs C:\temp\searchterms.txt

**--ar**

This switch informs the tool to search using a specified range to search for in “code page” strings.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --lr ipv4 --fr C:\temp\results.txt --ar [\x20-\x37] -q

**--ur**

This switch informs the tool to search using a specified range to search for in Unicode strings.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --lr ipv4 --fr C:\temp\results.txt --ur [\u0020-\u007E] -q

**--cp**

This switch informs the tool to use a specified code page. The default is 1252. Check out [this link<sup>50</sup>](#) for information on other code pages available.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --lr ipv4 --fr C:\temp\results.txt --ur [\u0020-\u007E] -q --cp 1256

**--mask**

This switch informs the tool to search a specified file mask. For instance, to search through all .exe files, one would specify --mask \*.exe.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --lr ipv4 --fr C:\temp\results.txt --mask \*.txt

The above command would search through all .txt files using the IPv4 regular expression as a search term.

**--ms**

This switch informs the tool of the maximum size file (in bytes) to process when using the -d switch.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --lr ipv4 --fr C:\temp\results.txt --mask \*.txt --ms 10

The above command will provide messages similar to the following:

<sup>50</sup><https://docs.microsoft.com/en-us/windows/win32/intl/code-page-identifiers?redirectedfrom=MSDN>

```
1 'D:\DFIRArtifactMuseum\Windows\AppCompatCache\ReadMe.txt' is bigger than max file size  
2 of 10 bytes! Skipping...  
3 Searching via RegEx pattern: \b(?:(:?25[0-5]|2[0-4][0-9]|1[0-9][0-9]|1[1-9]?[0-9])\.)\b  
4 {3}(:?25[0-5]|2[0-4][0-9]|1[0-9][0-9]|1[1-9]?[0-9])\b
```

**--ro**

This switch informs the tool to list the string matched as the search is running.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --lr ipv4 -o C:\temp\results.txt --ro -q

Try the above command with and without the --ro and see the difference. --ro will list each IPv4 address hit and running the command without --ro will list the files where hits are being located.

**--off**

This switch informs the tool to show the offset of the hit and code page for each hit.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --lr ipv4 -o C:\temp\results.txt --off -q

The above command's first hit will be similar to the following:

```
C:\Program Files (x86)\Common Files\Microsoft Shared\Vshub\1.0.0.0\Microsoft.VsHub.Server.HttpHostx64  
0x5D82 (U)
```

**--sa**

This switch informs the tool to sort the results alphabetically.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --lr ipv4 -o C:\temp\results.txt --sa -q

**"-sl"**

This switch informs the tool to sort the results by length.

Example: .\bstrings.exe -d "D:\DFIRArtifactMuseum\Windows" --lr ipv4 -o C:\temp\results.txt --sl -q

## bstrings Command Examples

### Example bstrings Commands

#### Search a specified file for the string URL

```
bstrings.exe -f "C:\Temp\UsrClass 1.dat" --ls URL
```

#### Search a specified file for the GUID regular expression

```
bstrings.exe -f "C:\Temp\someFile.txt" --lr guid
```

#### Search a specified file using a file with specified search strings and a file with specified regular expressions

```
bstrings.exe -f "C:\Temp\aBigFile.bin" --fs c:\temp\searchStrings.txt --fr c:\temp\s\searchRegex.txt
```

#### Search a specified directory for all files with the .dll file extension

```
bstrings.exe -d "C:\Temp" --mask "*.*.dll"
```

#### Search a specified directory for a range of specified characters within a code page

```
bstrings.exe -d "C:\Temp" --ar "[\x20-\x37]"
```

#### Search a specified directory using the Cyrillic (Mac) code page

```
bstrings.exe -d "C:\Temp" --cp 10007
```

#### Search a specified directory for the string test

```
bstrings.exe -d "C:\Temp" --ls test
```

#### Search a specified file for credit cards using regular expressions and sort the results alphabetically

```
bstrings.exe -f "C:\Temp\someOtherFile.txt" --lr cc --sa
```

#### Search a specified file using the credit card regular expression, sorting the results alphabetically, with a minimum string length of 15, and a maximum string length of 22

```
bstrings.exe -f "C:\Temp\someOtherFile.txt" --lr cc --sa -m 15 -x 22
```

**Search a specified file for the string `mui` and sort the results by length**

```
bstrings.exe -f "C:\Temp\UsrClass 1.dat" --ls mui --sl
```

# bstrings References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/bstrings> is the GitHub repository for bstrings

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- Introducing bstrings, a Better Strings utility!<sup>51</sup>
- bstrings 0.9.0.0 released<sup>52</sup>
- bstrings 0.9.5.0 released<sup>53</sup>
- A few updates<sup>54</sup>
- bstrings 0.9.7.0 released<sup>55</sup>
- bstrings 0.9.8.0 released<sup>56</sup>
- bstrings 0.9.9.0 released!<sup>57</sup>
- bstrings 1.0 released!<sup>58</sup>
- bstrings v1.1 released!<sup>59</sup>

### Community Resources

- Data Recovery Blog<sup>2</sup><sup>60</sup>

## Download bstrings

bstrings can be downloaded from <https://ericzimmerman.github.io/#!index.md>

---

<sup>51</sup><https://binaryforay.blogspot.com/2015/07/introducing-bstrings-better-strings.html>

<sup>52</sup><https://binaryforay.blogspot.com/2015/07/bstrings-0900-released.html>

<sup>53</sup><https://binaryforay.blogspot.com/2015/07/bstrings-0950-released.html>

<sup>54</sup><https://binaryforay.blogspot.com/2015/08/a-few-updates.html>

<sup>55</sup><https://binaryforay.blogspot.com/2015/11/bstrings-0970-released.html>

<sup>56</sup><https://binaryforay.blogspot.com/2015/12/bstrings-0980-released.html>

<sup>57</sup><https://binaryforay.blogspot.com/2016/02/bstrings-0990-released.html>

<sup>58</sup><https://binaryforay.blogspot.com/2016/02/bstrings-10-released.html>

<sup>59</sup><https://binaryforay.blogspot.com/2016/04/bstrings-v11-released.html>

<sup>60</sup><https://leahycenterblog.champlain.edu/2020/05/01/data-recovery-blog-2-2/>

# **EvtxECmd**

## **EvtxECmd Introduction**

EvtxECmd is a tool created by Eric Zimmerman used to parse event logs from Windows. Versions of Windows from Vista and beyond have utilized the .evtx format, which incorporates XML payloads within the .evtx files. EvtxECmd only parses .evtx files, so if you're dealing with .evt files, EvtxECmd will not parse those particular files.

## **EvtxECmd Use Cases**

### **Law Enforcement**

For those in Law Enforcement, this tool is useful for parsing event logs which can provide useful program execution artifacts, NTFS file system artifacts, evidence of USB device connections, and much more.

### **Private Sector**

For those in the Private Sector, this tool is useful for parsing event logs which can provide useful artifacts relating to RDP sessions, account lockouts, failed logons, and much more.

## EvtxECmd Maps

<https://github.com/EricZimmerman/evtx/tree/master/evtx/Maps>

### Creating EvtxECmd Maps

If you are looking for guidance on how to create EvtxECmd Maps, look no further than the following resources:

- [EvtxECmd Maps Guide<sup>61</sup>](#)
- [EvtxECmd Maps Template<sup>62</sup>](#)

Additionally, please check out Andrew Rathbun's 2021 SANS DFIR Summit presentation on EZ Tools/KAPE: How to Contribute to and Benefit from Open Source Contributions. Click [here<sup>63</sup>](#) for a timestamped link to the section of the presentation that relates to EvtxECmd Maps.

## EvtxECmd Switches

In a PowerShell window, running .\EvtxECmd.exe will provide the following options when running EvtxECmd:

```
-f <f>          File to process. This or -d is required
-d <d>          Directory to process that contains evtx files. This or -f is required
--csv <csv>      Directory to save CSV formatted results to
--csvf <csvf>    File name to save CSV formatted results to. When present, overrides default name
--json <json>    Directory to save JSON formatted results to
--jsonf <jsonf>   File name to save JSON formatted results to. When present, overrides default name
--xml <xml>      Directory to save XML formatted results to
--xmlf <xmlf>    File name to save XML formatted results to. When present, overrides default name
--dt <dt>        The custom date/time format to use when displaying time stamps [d\efault: yyyy-MM-dd HH:mm:ss.fffffff]
--inc <inc>       List of Event IDs to process. All others are ignored. Overrides -\exc Format is 4624,4625,5410
--exc <exc>       List of Event IDs to IGNORE. All others are included. Format is 4\
```

<sup>61</sup>[https://github.com/EricZimmerman/evtx/blob/master/evtx/Maps/!Channel-Name\\_Provider-Name\\_EventID.guide](https://github.com/EricZimmerman/evtx/blob/master/evtx/Maps/!Channel-Name_Provider-Name_EventID.guide)

<sup>62</sup>[https://github.com/EricZimmerman/evtx/blob/master/evtx/Maps/!Channel-Name\\_Provider-Name\\_EventID.template](https://github.com/EricZimmerman/evtx/blob/master/evtx/Maps/!Channel-Name_Provider-Name_EventID.template)

<sup>63</sup><https://youtu.be/mlb1GQP3ciE?t=860>

```
624,4625,5410
  --sd <sd>          Start date for including events (UTC). Anything OLDER than this i\
s dropped. Format should match --dt
  --ed <ed>          End date for including events (UTC). Anything NEWER than this is \
dropped. Format should match --dt
  --fj                When true, export all available data when using --json [default: \
False]
  --tdt <tdt>        The number of seconds to use for time discrepancy detection [defa\
ult: 1]
  --met                When true, show metrics about processed event log [default: True]
  --maps <maps>      The path where event maps are located. Defaults to 'Maps' folder \
where program was executed
                      [default: C:\Users\CFUser\OneDrive - Kroll\Desktop\EZ Tools\net6\\\
EvtxeCmd\Maps]
  --vss                Process all Volume Shadow Copies that exist on drive specified by \
-f or -d [default: False]
  --dedupe            Deduplicate -f or -d & VSCs based on SHA-1. First file found wins\
 [default: True]
  --sync              If true, the latest maps from https://github.com/EricZimmerman/ev\
tx/tree/master/evtx/Maps are
                      downloaded and local maps updated [default: False]
  --debug             Show debug information during processing [default: False]
  --trace             Show trace information during processing [default: False]
  --version           Show version information
  -, -h, --help       Show help and usage information
```

## Switch Descriptions

### --inc

This switch will provide EvtxECmd with which event ID(s) to process.

Example: .\EvtxECmd.exe -d C:\Windows\System32\winevt\Logs --inc 4624,4625

The above command WILL process 4624 and 4625 events, but will NOT process anything else.

### --exc

This switch will provide EvtxECmd with which event ID(s) to ignore during process.

Example: .\EvtxECmd.exe -d C:\Windows\System32\winevt\Logs --exc 4624,4625

The above command will NOT process 4624 and 4625 events, but will process everything other event.

### --sd

This switch will provide a starting date for which EvtxECmd will process all events that have occurred AFTER that date.

Example: .\EvtxECmd.exe -f "C:\Windows\System32\winevt\Logs\Microsoft-Windows-SmbClient%4Security.evt" --csv D:\ --sd "2022-07-29 00:00:00.000000"

Not using quotes around the timestamp may cause issues.

### --ed

This switch will provide a starting date for which EvtxECmd will process all events that have occurred BEFORE that date.

Example: .\EvtxECmd.exe -f "C:\Windows\System32\winevt\Logs\Microsoft-Windows-SmbClient%4Security.evt" --csv D:\ --ed "2022-07-27 00:00:00.000000"

Not using quotes around the timestamp may cause issues.

### --fj

This switch will include full details when using the common --json switch.

Example: .\EvtxECmd.exe -d "C:\Users\CFUser\Downloads\EventLogs\logs" --json "C:\Users\CFUser\Downloads\EventLogs\logs\json" --fj

### --tdt

This switch informs the tool with the number of seconds to look for when searching for time discrepancy detection.

Example: EvtxECmd.exe -f "C:\Temp\Application.evtx" --csv "c:\temp\out" --tdt 5

**--met**

This switch informs the tool as to whether or not to provide statistics, with the default being true.

Example: EvtxECmd.exe -f "C:\Temp\Application.evtx" --csv "c:\temp\out" --met false

With --met false, the following statistics will NOT be displayed after each .evt file that is parsed by EvtxECmd:

1	Metrics (including dropped events)
2	Event ID            Count
3	300                1
4	400                666
5	403                404
6	600                4,939
7	800                197

By default, the above statistics will be displayed after each .evt file is parsed by EvtxECmd.

**'-maps'**

This switch will inform the tool to look for EvtxECmd Maps at a specified location other than .\EvtxECmd\Maps.

Example: .\EvtxECmd.exe -d "D:\evt" --csv "D:\evt" --maps "D:\Maps"

**--vss**

This switch informs the tool to mount Volume Shadow Copies from the drive letter specified using the -f or -d switch and parse Prefetch files present within.

Example: .\EvtxECmd.exe -d "D:\evt" --csv "D:\evt" --vss

**'-sync'**

This switch will inform the tool to download all EvtxECmd Maps from GitHub<sup>64</sup> and update the local Maps stored in .\EvtxECmd\Maps.

Example: .\EvtxECmd.exe --sync

---

<sup>64</sup><https://github.com/EricZimmerman/evt/tree/master/evt/Maps>

# EvtxECmd Command Examples

## Example EvtxECmd Commands

### Parse a single .evt file and output to CSV to a specified location with a specified output filename

```
EvtxECmd.exe -f "C:\Temp\Application.evt" --csv "c:\temp\out" --csvf MyOutputFile.c\  
sv
```

### Parse a single .evt file and output to CSV to a specified location

```
EvtxECmd.exe -f "C:\Temp\Application.evt" --csv "c:\temp\out"
```

### Parse a single .evt file and output to JSON to a specified location

```
EvtxECmd.exe -f "C:\Temp\Application.evt" --json "c:\temp\jsonout"
```

## EvtxECmd Output

### Analyzing EvtxECmd Output - JSON

When executing EvtxECmd with JSON output enabled, you will see similar output to the example below:

```
1  {
2      "ChunkNumber": 0,
3      "Computer": "HostnameGoesHere",
4      "Payload": "{\"EventData\":{\"Data\":[{@Name\":\"param1\", \"#text\":\"86400\"},{@Name\":\"param2\", \"#text\":\"SuppressDuplicateDuration\"},{@Name\":\"param3\", \"#text\":\"Software\\\\Microsoft\\\\EventSystem\\\\EventLog\"}]}},
5      "Channel": "Application",
6      "Provider": "Microsoft-Windows-EventSystem",
7      "EventId": 4625,
8      "EventRecordId": "1",
9      "ProcessId": 0,
10     "ThreadId": 0,
11     "Level": "Info",
12     "Keywords": "0x8000000000000000",
13     "SourceFile": "C:\\\\temp\\\\evtx\\\\Application.evtx",
14     "ExtraDataOffset": 0,
15     "HiddenRecord": false,
16     "TimeCreated": "2022-05-19T15:34:34.9710202+00:00",
17     "RecordNumber": 1
18 },
19 }
```

## Analyzing EvtxECmd Output - XML

When executing EvtxECmd with XML output enabled, you will see similar output to the example below:

```
1 <Event>
2   <System>
3     <Provider Name="Microsoft-Windows-EventSystem" Guid="{899daace-4868-4295-afcd-9eb8\
4 fb497561}" EventSourceName="EventSystem" />
5     <EventID Qualifiers="16384">4625</EventID>
6     <Version>0</Version>
7     <Level>4</Level>
8     <Task>0</Task>
9     <Opcode>0</Opcode>
10    <Keywords>0x8000000000000000</Keywords>
11    <TimeCreated SystemTime="2022-05-19 15:34:34.9710202" />
12    <EventRecordID>1</EventRecordID>
13    <Correlation />
14    <Execution ProcessID="0" ThreadID="0" />
15    <Channel>Application</Channel>
16    <Computer>CFL-HostnameGoesHere</Computer>
17    <Security />
18  </System>
19  <EventData>
20    <Data Name="param1">86400</Data>
21    <Data Name="param2">SuppressDuplicateDuration</Data>
22    <Data Name="param3">Software\Microsoft\EventSystem\EventLog</Data>
23  </EventData>
24 </Event>
```

This is the preferred way to do research for creating new EvtxECmd Maps. EvtxECmd Maps use XPath queries which would require XML output to develop a working XPath query.

## **EvtxECmd Key Takeaways**

### **Important Data Points**

The Payload column will always have all the XML data stored within a given event log. The columns where data from the XML Payload are mapped to should be considered the highlights of that particular event, but given that there are only 6 PayloadData columns, some events naturally will have more than 6 relevant data points.

# EvtxECmd References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/evtx> is the GitHub repository for EvtxECmd

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- [Introducing EvtxECmd!!<sup>65</sup>](https://binaryforay.blogspot.com/2019/04/introducing-evtxecmd.html)

### Community Resources

- [Enhancing Event Log Analysis with EvtxEcmand using KAPE<sup>66</sup>](#)

## Download EvtxECmd

EvtxECmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

## Event Log Sample Data

Sample Event Log artifacts can be found at the DFIRArtifactMuseum<sup>67</sup>.

---

<sup>65</sup><https://binaryforay.blogspot.com/2019/04/introducing-evtxecmd.html>

<sup>66</sup><https://www.youtube.com/watch?v=BIkyWexMF0I>

<sup>67</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/EventLogs>

# IISGeoLocate

## IISGeoLocate Introduction

IISGeoLocate can be used to geolocate IP addresses found in IIS Logs. IIS Logs can be located at C:\inetpub\logs\LogFiles on Windows Server machines.

## IISGeoLocate Use Cases

### Law Enforcement

For those in Law Enforcement, this tool may not prove to be immediately useful, but should there be a case where IIS logs are relevant, this tool can parse them!

### Private Sector

For those in the Private Sector, this tool is useful for parsing IIS logs and helping to identify unique IPs that can be geolocated and ran through useful resources like VirusTotal for IOC development. Additionally, this tool can convert the IIS logs from flat text to CSV format for better ingestion into Timeline Explorer.

## IISGeoLocate Switches

In a PowerShell window, running .\IISGeoLocate.exe will provide the following options when running IISGeoLocate:

Description:

```
iisgeolocate version 2.2.0.0
```

```
Author: Eric Zimmerman (saericzimmerman@gmail.com)
```

```
https://github.com/EricZimmerman/iisGeolocate
```

Usage:

```
iisGeolocate [options]
```

Options:

-d <d> (REQUIRED)	The directory that contains IIS logs. This will be recursively searched
	<b>for</b> *.log files
--csv <csv> (REQUIRED)	The directory to <b>write</b> results to
--sbl	When true, <b>do</b> NOT show bad lines to console (they are still logged to a
	file) [ <b>default</b> : False]
--nul	When true, <b>do</b> NOT create updated CSV files <b>in</b> --csv directory [ <b>default</b> :
	False]
--version	Show version information
-?, -h, --help	Show help and usage information

## Switch Descriptions

### --sbl

This switch will instruct IISGeoLocate to not display bad lines to the console. These lines will still be written to the file being written to the location specified with the --csv switch.

```
.\iisgeolocate.exe -d C:\inetpub\logs\LogFiles --csv C:\temp --sbl
```

### --nul

This switch will instruct IISGeoLocate to not update the CSV file at the location specified with the --csv switch.

```
.\iisgeolocate.exe -d C:\inetpub\logs\LogFiles --csv C:\temp --nul true
```

## IISGeoLocate Output

### Analyzing IISGeoLocate Output - CSV

IISGeoLocate will convert IIS logs into CSV format for ingestion into Timeline Explorer as well as adding IP geolocation data to each log entry.

## IISGeoLocate References

### Associated GitHub Repositories

- <https://github.com/EricZimmerman/iisGeolocate> is the GitHub repository for IISGeoLocate

### Download IISGeoLocate

IISGeoLocate can be downloaded from <https://ericzimmerman.github.io/#!index.md>

# **JLECmd**

## **JLECmd Introduction**

### **Description**

JLECmd is a tool created by Eric Zimmerman used to parse JumpList files. JumpLists are native to the Windows operating system and are common indicators of file access.

### **JLECmd Use Cases**

#### **Law Enforcement**

For those in Law Enforcement, this tool is useful for parsing JumpLists which can provide useful evidence of file access that can be attributed to a specific user account. JumpLists can provide insight as to when a file was last opened.

#### **Private Sector**

For those in the Private Sector, this tool is useful for parsing JumpLists which can provide useful evidence of file access that can be attributed to a specific user account. Often, these can be used along with Shellbags and LNK files to determine which files and folders were accessed by a threat actor during a period of unauthorized access

## JLECmd Switches

In a PowerShell window, running .\JLECmd.exe will provide the following options when running JLECmd:

-f <f>	File to <b>process</b> . Either this or -d is required
-d <d>	Directory to recursively <b>process</b> . Either this or -f is required
--all	<b>Process</b> all files <b>in</b> directory vs. only files matching *.automaticDestinations-ms or *.customDestinations-ms [ <b>default</b> : False]
--csv <csv>	Directory to save CSV formatted results to. This or --json required unless --de or --body is specified
--csvf <csvf>	File name to save CSV formatted results to. When present, overrides <b>default</b> name
--json <json>	Directory to save json representation to. Use --pretty <b>for</b> a more human readable layout
--html <html>	Directory to save xhtml formatted results to. Be sure to include the full path <b>in</b> double quotes
--pretty	When exporting to json, use a more human readable layout [ <b>default</b> : False]
-q	Only show the filename being processed vs all output. Useful to speed up exporting to json and/or csv [ <b>default</b> : False]
--ld	Include more information about lnk files [ <b>default</b> : False]
--fd	Include full information about lnk files (Alternatively, dump lnk files using --dumpTo and <b>process</b> with LECmd) [ <b>default</b> : False]
--appIds <appIds>	Path to file containing AppIDs and descriptions (appid description format). New appIDs are added to the built-in list, existing appIDs will have their descriptions updated
--dumpTo <dumpTo>	Directory to save exported lnk files
--dt <dt>	The custom date/time format to use when displaying timestamps. See <a href="https://goo.gl/CNVq0k">https://goo.gl/CNVq0k</a> <b>for</b> options. <b>Default</b> is: yyyy-MM-dd HH:mm:ss [ <b>default</b> : yyyy-MM-dd HH:mm:ss]
--mp	Display higher precision <b>for</b> timestamps [ <b>default</b> : False]
--withDir	When true, show contents of Directory not accounted <b>for</b> in DestList entries [ <b>default</b> : False]

```
--debug           Show debug information during processing [default:  
False]  
--trace           Show trace information during processing [default:  
False]  
--version         Show version information  
-?, -h, --help   Show help and usage information
```

While there are explanations for each switch, let's walk through each one of the switches unique to JLECmd in further detail.

## Switch Descriptions

### --all

This switch will instruct JLECmd to attempt to parse every file in the directory specified, regardless of file extension. Output will be generated only for those files identified to be .LNK files.

Example: .\JLECmd.exe -d D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM  
--all

### -q

This switch will instruct JLECmd to output only what files were processed rather than the parsed data from each JumpList.

For instance, running .\JLECmd.exe -d "D:\DFIRArtifactMuseum" --csv "D:\DFIRArtifactMuseum\" resulted in the following: Processed 15 out of 20 files in 0.6428 seconds

Running .\JLECmd.exe -d "D:\DFIRArtifactMuseum" --csv "D:\DFIRArtifactMuseum\" -q resulted in the following: Processed 15 out of 20 files in 0.1246 seconds

### --ld

This switch will instruct JLECmd to include more details about .LNK files. Specifically, JLECmd will include the following information for each JumpList:

```
--- Link information ---
Flags: VolumeIdAndLocalBasePath

>> Volume information
Drive type: Fixed storage media (Hard drive)
Serial number: 88008C2F
Label: (No label)
Local path: C:\Users\e\AppData\Roaming\Microsoft\Windows\Libraries\Videos.library-\ms
```

### --fd

This switch will instruct JLECmd to include the full details of the .LNK files. An example of the additional information included can be seen below:

```
Lnk target created: 2016-01-16 20:22:25  
Lnk target modified: 2016-01-16 20:23:24  
Lnk target accessed: 2016-01-16 20:23:24
```

--- Header ---

```
Target created: 2016-01-16 20:22:25  
Target modified: 2016-01-16 20:23:24  
Target accessed: 2016-01-16 20:23:24
```

```
File size: 3,483  
Flags: HasTargetIdList, HasLinkInfo,IsUnicode  
File attributes: FileAttributeArchive, FileAttributeNotContentIndexed  
Icon index: 0  
Show window: SwNormal (Activates and displays the window. The window is restored to its original size and position if the window is minimized or maximized.)
```

--- Link information ---

```
Flags: VolumeIdAndLocalBasePath
```

```
>> Volume information  
Drive type: Fixed storage media (Hard drive)  
Serial number: 88008C2F  
Label: (No label)  
Local path: C:\Users\e\AppData\Roaming\Microsoft\Windows\Libraries\Videos.library-ms
```

--- Target ID information (Format: **Type ==> Value**) ---

and

```
-Root folder: GUID ==> User Libraries
```

```
-Variable ==> Videos
```

--- **End** Target ID information ---

--- Extra blocks information ---

```
>> Property store data block (Format: GUID\ID Description ==> Value)  
9f4c2855-9f79-4b39-a8d0-e1d42de1d5f3\7      App User Model Is DestList Link =\r  
=> True
```

```
>> Tracker database block
Machine ID: win7x64
MAC Address: 00:15:5d:01:6d:0b
MAC Vendor: (Unknown vendor)
Creation: 2016-01-16 21:18:45

Volume Droid: 42e937f8-244d-406d-86eb-43604356d1ae
Volume Droid Birth: 42e937f8-244d-406d-86eb-43604356d1ae
File Droid: ba034b9e-bc96-11e5-b231-00155d016d0b
File Droid birth: ba034b9e-bc96-11e5-b231-00155d016d0b

(lnk file not present)
```

#### --appIds

This switch will instruct JLECmd to use an AppIDs.txt file specified by the user. Default is using the built-in AppIDs.txt file included within JLECmd, which can be found here: <https://github.com/EricZimmerman/JumpList/blob/master/JumpList/Resources/AppIDs.txt>

Example: JLECmd.exe -f "D:DFIRArtifactMuseumWindowsJumpListsWin7\1b4dd67f29cb1962.automaticDestinations" --appIds "C:\tempAppIDs.txt"

#### --dumpTo

This switch will instruct JLECmd to dump the .LNK files associated with the JumpList to the specified location.

Example: JLECmd.exe -f "D:DFIRArtifactMuseumWindowsJumpListsWin7\1b4dd67f29cb1962.automaticDestinations" --dumpTo "D:DFIRArtifactMuseum\testJL"

#### --mp

This switch will instruct JLECmd to provide more verbose timestamps. For instance, running .\JLECmd.exe -f "D:\DFIRArtifactMuseum\Windows\JumpLists\Win7\1b4dd67f29cb1962.automaticDestinations" resulted in the following:

```
--- DestList entries ---
Entry #: 4
    MRU: 0
    Path: ::{031E4825-7B94-4DC3-B131-E946B44C8DD5}\Videos.library-ms ==> User Library\
s\Videos.library-ms
    Pinned: False
    Created on: 2016-01-16 21:18:45
    Last modified: 2016-01-16 20:23:24
    Hostname: win7x64
    Mac Address: 00:15:5d:01:6d:0b
    Interaction count: 0

--- Lnk information ---
    Absolute path: User Libraries\Videos
```

Running .\JLECmd.exe -f "D:\DFIRArtifactMuseum\Windows\JumpLists\Win7\1b4dd67f29cb1962.automaticDest\
--mp resulted in the following:

```
--- DestList entries ---
Entry #: 4
    MRU: 0
    Path: ::{031E4825-7B94-4DC3-B131-E946B44C8DD5}\Videos.library-ms ==> User Library\
s\Videos.library-ms
    Pinned: False
    Created on: 2016-01-16 21:18:45.6562590
    Last modified: 2016-01-16 20:23:24.8901093
    Hostname: win7x64
    Mac Address: 00:15:5d:01:6d:0b
    Interaction count: 0

--- Lnk information ---
    Absolute path: User Libraries\Videos
```

#### --withDir

This switch will instruct JLECmd to list directories not account for in the DestList entries.

For example, running .\JLECmd.exe -f "D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\f01\
--withDir resulted in the following extra data:

Directory entries not represented by DestList entries

Directory: a

Absolute path: My Computer\C:\Users\TestUser\Downloads\APTSimulator\_pw\_apt\dist

Directory: b

Absolute path: My Computer\C:\Users\TestUser\Desktop\Targets

# JLECmd Command Examples

## Example JLECmd Commands

### Parse a single JumpList and view the results within the console

```
JLECmd.exe -f "D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\f01b4d95\cf55d32a.automaticDestinations-ms"
```

### Parse a single JumpList and output the results to CSV at specified location

```
JLECmd.exe -f "D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\f01b4d95\cf55d32a.automaticDestinations-ms" --csv C:\Temp
```

### Parse a single JumpList, output to JSON in pretty format

```
JLECmd.exe -f "D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\f01b4d95\cf55d32a.automaticDestinations-ms" --json "D:\jsonOutput" --pretty
```

### Parse a directory, output to CSV at specified location, output to HTML at specified location, output to XML to specified location, while only showing the filename being processed (used to speed up processing)

```
JLECmd.exe -d "C:\Temp" --csv "c:\temp" --html c:\temp --xml c:\temp\xml -q
```

### Parse all files in a directory (regardless of presence of .JumpList extension)

```
JLECmd.exe -d "C:\Temp" --all
```

### Parse a single JumpList and view the results within the console with higher precision timestamps

```
JLECmd.exe -f "D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\f01b4d95\cf55d32a.automaticDestinations-ms" --mp
```

# JLECmd Output

## Analyzing JLECmd Output - Console

When parsing JumpList files with JLECmd without the -q switch, JLECmd will display in the console window parsed data from each .LNK file processed.

```
Administrator: Windows PowerShell
PS D:\ZimmermanTools\net6> ./JLECmd.exe -f "D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\573770283dc3d854.automaticDestinations-ms"
JLECmd version 1.5.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/JLECmd

Command line: -f D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\573770283dc3d854.automaticDestinations-ms

Processing D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\573770283dc3d854.automaticDestinations-ms

Source file: D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\573770283dc3d854.automaticDestinations-ms

--- AppId information ---
AppID: 573770283dc3d854
Description: Windows Defender

--- DestList information ---
Expected DestList entries: 1
Actual DestList entries: 1
DestList version: 4

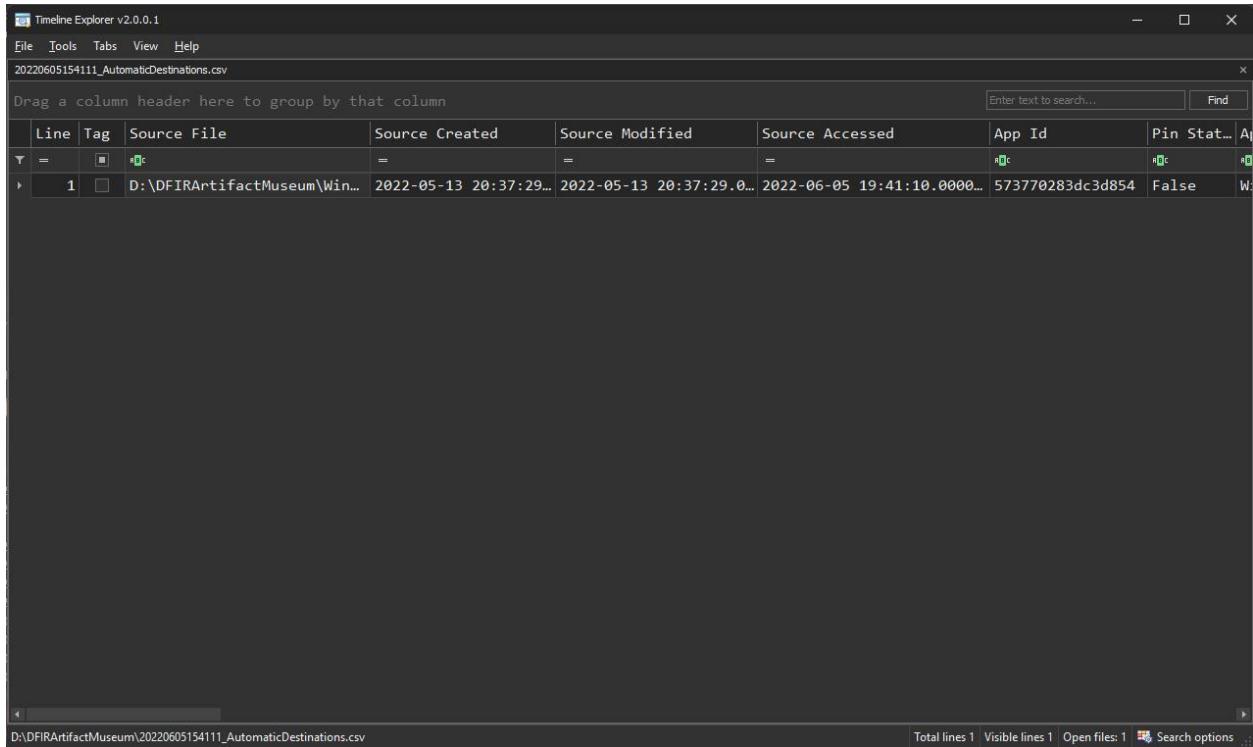
--- DestList entries ---
Entry #: 1
MRU: 0
Path: windowsdefender://threat/
Pinned: False
Created on: 1582-10-15 00:00:00
Last modified: 2022-03-10 19:49:26
Hostname:
Mac Address:
Interaction count: 1

--- Lnk information ---
Absolute path: Internet Explorer (Homepage)\windowsdefender://threat/

----- Processed D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\573770283dc3d854.automaticDestinations-ms in 0.05493240 seconds -----
```

## Analyzing JLECmd Output - CSV

When parsing JumpLists with JLECmd with the `--csv` switch, JLECmd will output the parsed data into a CSV file. The CSV file can then be ingested into a tool like Timeline Explorer to analyze the parsed data.



Line	Tag	Source File	Source Created	Source Modified	Source Accessed	App Id	Pin Stat...	App Name
1		D:\DFIRArtifactMuseum\Win...	2022-05-13 20:37:29...	2022-05-13 20:37:29.0...	2022-06-05 19:41:10.0000...	573770283dc3d854	False	W...

## Analyzing JLECmd Output - JSON

When parsing JumpLists with JLECmd with the `-j json` switch, JLECmd will output the parsed data into a JSON file.

``

```
{
  "Directory": [
    {
      "ClassId": "00450020006e00747200790000000000",
      "UserFlags": 0,
      "ModifiedTime": "\/Date(1646941766733)\/",
      "FirstDirectorySectorId": 3,
      "DirectorySize": 576,
      "PreviousDirectoryId": -1,
      "NextDirectoryId": -1,
      "SubDirectoryId": 1,
      "DirectoryName": "Root Entry",
      "DirectoryType": "RootStorage",
      "NodeColor": "Red"
    },
    {
      "ClassId": "00000000000000000000000000000000",
      "UserFlags": 0,
      "FirstDirectorySectorId": 0,
      "DirectorySize": 309,
      "PreviousDirectoryId": -1,
      "NextDirectoryId": 2,
      "SubDirectoryId": -1,
      "DirectoryName": "1",
      "DirectoryType": "Stream",
      "NodeColor": "Black"
    },
    {
      "ClassId": "0069004c007300740000000000000000",
      "UserFlags": 0,
      "FirstDirectorySectorId": 5,
      "DirectorySize": 216,
      "PreviousDirectoryId": -1,
      "NextDirectoryId": -1,
      "SubDirectoryId": -1,
      "DirectoryName": "DestList",
      "DirectoryType": "Stream"
    }
  ]
}
```

```
        "DirectoryType": "Stream",
        "NodeColor": "Red"
    },
],
"AppId": {
    "AppId": "573770283dc3d854",
    "Description": "Windows Defender"
},
"DestListCount": 1,
"PinnedDestListCount": 0,
"LastUsedEntryNumber": 1,
"DestListVersion": 4,
"SourceFile": "D:\\\\DFIRArtifactMuseum\\\\Windows\\\\JumpLists\\\\Win10\\\\APTSimulatorVM\\\\5\\73770283dc3d854.automaticDestinations-ms",
"DestListEntries": [
    {
        "Hostname": "",
        "VolumeDroid": "00000000000000000000000000000000",
        "VolumeBirthDroid": "00000000000000000000000000000000",
        "FileDroid": "00000000000000000000000000000000",
        "FileBirthDroid": "00000000000000000000000000000000",
        "EntryNumber": 1,
        "MRUPosition": 0,
        "InteractionCount": 1,
        "CreatedOn": "\\\\Date(-1221929280000)\\\\",
        "LastModified": "\\\\Date(1646941766733)\\\\",
        "Pinned": false,
        "Path": "windowsdefender://threat/",
        "MacAddress": "00:00:00:00:00:00",
        "Lnk": {
            "TargetIDs": [
                {
                    "__type": "Lnk.ShellItems.ShellBag0X1F, Lnk",
                    "PropertyStore": {
                        "Sheets": [
                            ...
                        ],
                        "FriendlyName": "Root folder: GUID",
                        "Value": "Internet Explorer (Homepage)",
                        "ExtensionBlocks": [
                            ...
                        ]
                    }
                },
                {
                    ...
                }
            ]
        }
    }
]
```

```
        " __type": "Lnk.ShellItems.ShellBag0X61, Lnk",
        "UserName": "",
        "FriendlyName": "URI",
        "Value": "windowsdefender://threat/",
        "ExtensionBlocks": [
    ]
}
],
"ExtraBlocks": [
{
    " __type": "Lnk.ExtraData.PropertyStoreDataBlock, Lnk",
    "PropertyStore": {
        "Sheets": [
    {
        "Size": 45,
        "Version": "31-53-50-53",
        "GUID": "9f4c2855-9f79-4b39-85d0-00160949734e",
        "PropertyNames": {
            "7": "True"
        },
        "PropertySheetType": "Numerical"
    }
]
}
}
],
"SourceFile": "D:\\DFIRArtifactMuseum\\Windows\\JumpLists\\Win10\\AppList\\573770283dc3d854.automaticDestinations-ms_Directory name_1",
"RawBytes": "TAAAAAEUAgAAAAAAwAAAAAAAeBAKAA<SNIP>",
"Header": {
    "Signature": "0002140100000000c000000000000046",
    "DataFlags": 10485889,
    "FileAttributes": 0,
    "TargetCreationDate": "\/Date(-11644473600000)\/",
    "TargetModificationDate": "\/Date(-11644473600000)\/",
    "TargetLastAccessedDate": "\/Date(-11644473600000)\/",
    "FileSize": 0,
    "IconIndex": 0,
    "HotKey": "",
    "ShowWindow": "SwNormal",
    "Reserved0": 0,
    "Reserved1": 0,
    "Reserved2": 0
}
```

```
        },
        "LocationFlags": 0
    }
}
]
```

## Analyzing JLECmd Output - HTML

When parsing JumpLists with JLECmd with the `--html` switch, JLECmd will output the parsed data into an HTML file. Please note, the HTML file will be placed into a dedicated subfolder, which is not normally the case when outputting to CSV or JSON.

```
D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\573770283dc3d854.automaticDestinations-ms
Source Created: 2022-05-13 20:37:29
Source Modified: 2022-05-13 20:37:29
Source Accessed: 2022-06-05 19:42:00
App ID: 573770283dc3d854
App ID Description: Windows Defender
DestList version: 4
Last Used Entry Number: 1
1 TargetID Absolute Path: Internet Explorer (Homepage)\windowsdefender://threat/
Last Modified: 2022-03-10 19:49:26
Path: windowsdefender://threat/
Pin Status: False
Interaction Count: 1
File Size: 0 (bytes)
File Attributes: 0
Header Flags: HasTargetIdList,IsUnicode,DisableKnownFolderTracking,AllowLinkToLink
Drive Type: (None)
Extra Blocks Present: PropertyStoreDataBlock
```

## JLECmd Sample Output

```
. \JLECmd.exe -d "D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM"

----- Processed D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\cc\
ba5a5986c77e43.customDestinations-ms in 0.01157700 seconds -----

Processing D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\f01b4d95cf55\
d32a.automaticDestinations-ms

Source file: D:\DFIRArtifactMuseum\Windows\JumpLists\Win10\APTSimulatorVM\f01b4d95cf\
55d32a.automaticDestinations-ms

--- AppId information ---
AppID: f01b4d95cf55d32a
Description: Windows Explorer Windows 8.1

--- DestList information ---
Expected DestList entries: 11
Actual DestList entries: 11
DestList version: 4

--- DestList entries ---
Entry #: 11
MRU: 0
Path: C:\Users\TestUser\Desktop\Targets
Pinned: False
Created on: 2022-03-10 16:35:09
Last modified: 2022-03-10 21:40:42
Hostname: desktop-tmku40h
Mac Address: ec:63:d7:72:83:36
Interaction count: 1

--- Lnk information ---
Absolute path: My Computer\C:\Users\TestUser\Desktop\Targets

Entry #: 1
MRU: 1
Path: knownfolder:{754AC886-DF64-4CBA-86B5-F7FBF4FBCEF5} ==> ThisPCDesktopFolder
Pinned: True
Created on: 2022-03-10 19:33:36
```

```
Last modified: 2022-03-10 21:40:42
Hostname: desktop-tmku40h
Mac Address: ec:63:d7:72:83:36
Interaction count: 4
```

```
--- Lnk information ---
Absolute path: My Computer\Desktop
```

```
Entry #: 10
MRU: 2
Path: C:\Users\TestUser\Downloads\APTSimulator_pw_apt\dist
Pinned: False
Created on: 2022-03-10 16:35:09
Last modified: 2022-03-10 19:49:34
Hostname: desktop-tmku40h
Mac Address: ec:63:d7:72:83:36
Interaction count: 1
```

```
--- Lnk information ---
Absolute path: My Computer\C:\Users\TestUser\Downloads\APTSimulator_pw_apt\dist
```

```
Entry #: 9
MRU: 3
Path: C:\Users\TestUser\Downloads\APTSimulator_pw_apt\APTSimulator
Pinned: False
Created on: 2022-03-10 16:35:09
Last modified: 2022-03-10 19:49:34
Hostname: desktop-tmku40h
Mac Address: ec:63:d7:72:83:36
Interaction count: 1
```

```
--- Lnk information ---
Absolute path: My Computer\C:\Users\TestUser\Downloads\APTSimulator_pw_apt\APTSimulator
```

```
Entry #: 8
MRU: 4
Path: C:\Users\TestUser\Downloads\APTSimulator_pw_apt
Pinned: False
Created on: 2022-03-10 16:35:09
```

```
Last modified: 2022-03-10 19:49:34
Hostname: desktop-tmku40h
Mac Address: ec:63:d7:72:83:36
Interaction count: 1

--- Lnk information ---
Absolute path: My Computer\C:\Users\TestUser\Downloads\APTSimulator_pw_apt

Entry #: 7
MRU: 5
Path: C:\Users\TestUser\Downloads\Sysmon
Pinned: False
Created on: 2022-03-10 16:35:09
Last modified: 2022-03-10 19:49:07
Hostname: desktop-tmku40h
Mac Address: ec:63:d7:72:83:36
Interaction count: 1

--- Lnk information ---
Absolute path: My Computer\C:\Users\TestUser\Downloads\Sysmon

Entry #: 6
MRU: 6
Path: knownfolder:{18989B1D-99B5-455B-841C-AB7C74E4DDFC} ==> Videos
Pinned: False
Created on: 2022-03-10 19:33:36
Last modified: 2022-03-10 16:35:24
Hostname: desktop-tmku40h
Mac Address: ec:63:d7:72:83:36
Interaction count: 3

--- Lnk information ---
Absolute path: My Computer\Videos

Entry #: 5
MRU: 7
Path: knownfolder:{4BD8D571-6D19-48D3-BE97-422220080E43} ==> Music
Pinned: False
Created on: 2022-03-10 19:33:36
Last modified: 2022-03-10 16:35:24
```

```
Hostname: desktop-tmku40h
Mac Address: ec:63:d7:72:83:36
Interaction count: 3

--- Lnk information ---
Absolute path: My Computer\Music

Entry #: 3
MRU: 8
Path: knownfolder:{FDD39AD0-238F-46AF-ADB4-6C85480369C7} ==> Documents
Pinned: True
Created on: 2022-03-10 19:33:36
Last modified: 2022-03-10 16:35:24
Hostname: desktop-tmku40h
Mac Address: ec:63:d7:72:83:36
Interaction count: 3

--- Lnk information ---
Absolute path: My Computer\Documents

Entry #: 4
MRU: 9
Path: knownfolder:{33E28130-4E1E-4676-835A-98395C3BC3BB} ==> Pictures
Pinned: True
Created on: 2022-03-10 19:33:36
Last modified: 2022-03-10 16:35:24
Hostname: desktop-tmku40h
Mac Address: ec:63:d7:72:83:36
Interaction count: 3

--- Lnk information ---
Absolute path: My Computer\Pictures

Entry #: 2
MRU: 10
Path: knownfolder:{374DE290-123F-4565-9164-39C4925E467B} ==> Downloads
Pinned: True
Created on: 2022-03-10 19:33:36
Last modified: 2022-03-10 16:34:28
Hostname: desktop-tmku40h
```

Mac Address: ec:63:d7:72:83:36

Interaction count: 3

--- Lnk information ---

Absolute path: My Computer\Downloads

## JLECmd Key Takeaways

JLECmd parses JumpLists, which are a collection of LNK files organized by AppID. JumpLists provide indicators of file access.

JLECmd output, regardless of file format, will provide potentially useful information about user activity on a given Windows system.

## Important Data Points

### Last Modified

This timestamp will provide you with the a reliable indication of when a file was opened with a specific application.

```
1 --- AppId information ---
2     AppID: f01b4d95cf55d32a
3     Description: Windows Explorer Windows 8.1
4
5 --- DestList information ---
6     Expected DestList entries: 11
7     Actual DestList entries:    11
8     DestList version:          4
9
10 --- DestList entries ---
11    Entry #: 11
12    MRU: 0
13    Path: C:\Users\TestUser\Desktop\Targets
14    Pinned: False
15    Created on: 2022-03-10 16:35:09
16    Last modified: 2022-03-10 21:40:42
17    Hostname: desktop-tmku40h
18    Mac Address: ec:63:d7:72:83:36
19    Interaction count: 1
20
21 --- Lnk information ---
22     Absolute path: My Computer\C:\Users\TestUser\Desktop\Targets
```

# JLECmd References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/JumpList> is the C# library for parsing JumpList files
- <https://github.com/EricZimmerman/OleCf> is the C# library for parsing OLE compound files, which Jump Lists are
- <https://github.com/EricZimmerman/JLECmd> is the Command Line wrapper for the JumpList library

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, [Binary Foray](#)<sup>68</sup>:

- Jump lists in depth: Understand the format to better understand what your tools are (or aren't) doing<sup>69</sup>
- Introducing JLECmd!<sup>70</sup>
- PECmd, LECmd, and JLECmd updated!<sup>71</sup>
- LECmd and JLECmd updated<sup>72</sup>
- JLECmd v0.9.6.0 released<sup>73</sup>

## Download JLECmd

JLECmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

## JumpList Sample Data

Sample JumpList files can be found at the [DFIRArtifactMuseum](#)<sup>74</sup>.

---

<sup>68</sup><https://binaryforay.blogspot.com/>

<sup>69</sup><https://binaryforay.blogspot.com/2016/02/jump-lists-in-depth-understand-format.html>

<sup>70</sup><https://binaryforay.blogspot.com/2016/03/introducing-jlecmd.html>

<sup>71</sup><https://binaryforay.blogspot.com/2016/03/pecmd-lecmd-and-jlecmd-updated.html>

<sup>72</sup><https://binaryforay.blogspot.com/2016/04/lecmd-and-jlecmd-updated.html>

<sup>73</sup><https://binaryforay.blogspot.com/2016/09/jlecmd-v0960-released.html>

<sup>74</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/JumpLists>

# **LECcmd**

## **LECcmd Introduction**

### **Description**

LECcmd is a tool created by Eric Zimmerman used to parse .LNK files. .LNK files are native to the Windows operating system and are common indicators of file access.

### **LECcmd Use Cases**

#### **Law Enforcement**

For those in Law Enforcement, this tool is useful for parsing LNK files which can provide useful evidence of file access that can be attributed to a specific user account. LNK files can provide insight as to when a file was first opened and last opened.

#### **Private Sector**

For those in the Private Sector, this tool is useful for parsing LNK files which can provide useful evidence of file access that can be attributed to a specific user account. Often, these can be used along with Shellbags and JumpLists to determine which files and folders were accessed by a threat actor during a period of unauthorized access

## LECmd Switches

In a PowerShell window, running .\LECmd.exe will provide the following options when running LECmd:

```
-f <f>          File to process. Either this or -d is required
-d <d>          Directory to recursively process. Either this or -f is required
-r              Only process lnk files pointing to removable drives [default: False]
-e]
-q              Only show the filename being processed vs all output. Useful to speed up exporting to json and/or csv
               [default: False]
--all           Process all files in directory vs. only files matching *.lnk [default: False]
--csv <csv>     Directory to save CSV formatted results to. Be sure to include the full path in double quotes
--csvf <csvf>   File name to save CSV formatted results to. When present, override s default name
--xml <xml>      Directory to save XML formatted results to. Be sure to include the full path in double quotes
--html <html>    Directory to save xhtml formatted results to. Be sure to include the full path in double quotes
--json <json>    Directory to save json representation to. Use --pretty for a more human readable layout
--pretty        When exporting to json, use a more human readable layout [default: False]
--nid           Suppress Target ID list details from being displayed [default: False]
--neb           Suppress Extra blocks information from being displayed [default: False]
--dt <dt>        The custom date/time format to use when displaying time stamps. See https://goo.gl/CNVq0k for options
               [default: yyyy-MM-dd HH:mm:ss]
--mp            Display higher precision for time stamps [default: False]
--debug         Show debug information during processing [default: False]
--trace         Show trace information during processing [default: False]
--version       Show version information
-?, -h, --help  Show help and usage information
```

While there are explanations for each switch, let's walk through each one of the switches unique to LECmd in further detail.

## LECmd Switch Descriptions

### -r

This switch will instruct LECmd to parse .LNK files that point to a file being opened from a removable drive ONLY.

Example: .\LECmd.exe -d D:\DFIRArtifactMuseum\Windows\LNK\Win10 -r

### -q

This switch will instruct LECmd to output only what files were processed rather than the parsed data from each .LNK file.

For instance, running .\LECmd.exe -d "D:\DFIRArtifactMuseum" --csv "D:\DFIRArtifactMuseum\" resulted in the following: Processed 554 out of 554 files in 6.4882 seconds

Running .\LECmd.exe -d "D:\DFIRArtifactMuseum" --csv "D:\DFIRArtifactMuseum\" -q resulted in the following: Processed 554 out of 554 files in 0.3951 seconds

### --all

This switch will instruct LECmd to attempt to parse every file in the directory specified, regardless of file extension. Output will be generated only for those files identified to be .LNK files.

Example: .\LECmd.exe -d D:\DFIRArtifactMuseum\Windows --all

### --nid

This switch will instruct LECmd to suppress the Target ID list details. This will provide a message similar to this in place of the Target ID list details: (Target ID information suppressed. Lnk TargetID count: 6)

Example: .\LECmd.exe -d D:\DFIRArtifactMuseum\Windows\LNK\Win10 --nid

### --neb

This switch will instruct LECmd to suppress the Extra blocks details. This will provide a message similar to this in place of the Target ID list details: (Extra blocks information suppressed. Lnk Extra block count: 2)

Example: .\LECmd.exe -d D:\DFIRArtifactMuseum\Windows\LNK\Win10 --neb

### --mp

This switch will instruct LECmd to provide more verbose timestamps. For instance, running .\LECmd.exe -f "D:\DFIRArtifactMuseum\Windows\LNK\Win10\XWFIM.lnk.test" resulted in the following:

```
Source file: D:\DFIRArtifactMuseum\Windows\LNK\Win10\XWFIM.lnk.test
Source created: 2022-01-31 17:25:58
Source modified: 2022-01-31 17:25:58
Source accessed: 2022-06-05 01:57:47
```

Running .\LECmd.exe -f "D:\DFIRArtifactMuseum\Windows\LNK\Win10\XWFIM.lnk.test" --mp resulted in the following:

```
Source file: D:\DFIRArtifactMuseum\Windows\LNK\Win10\XWFIM.lnk.test
Source created: 2022-01-31 17:25:58.4983924
Source modified: 2022-01-31 17:25:58.0000000
Source accessed: 2022-06-05 01:57:51.8825142
```

# LECmd Command Examples

## Example LECmd Commands

### Parse a single LNK file and view the results within the console

```
LECmd.exe -f "C:\Temp\foobar.lnk"
```

### Parse a single LNK file and output the results to CSV at specified location

```
LECmd.exe -f C:\Temp\foobar.lnk --csv C:\Temp
```

### Parse a single LNK file, output to JSON in pretty format

```
LECmd.exe -f "C:\Temp\somelink.lnk" --json "D:\jsonOutput" --pretty
```

### Parse a directory, output to CSV at specified location, output to HTML at specified location, output to XML to specified location, while only showing the filename being processed (used to speed up processing)

```
LECmd.exe -d "C:\Temp" --csv "c:\temp" --html c:\temp --xml c:\temp\xml -q
```

### Parse a single LNK file while suppressing Target ID list details and Extra blocks information

```
LECmd.exe -f "C:\Temp\some other link.lnk" --nid --neb
```

### Parse all files in a directory (regardless of presence of .LNK file extension)

```
LECmd.exe -d "C:\Temp" --all
```

### Parse all files in a directory (regardless of presence of .LNK file extension), but only provide output for those .LNK files that point to removable drives

```
LECmd.exe -d "C:\Temp" --all -r
```

### Parse a single LNK file and view the results within the console with higher precision timestamps

```
LECmd.exe -f "D:\DFIRArtifactMuseum\Windows\LNK\Win10\xWFIM.lnk.test" --mp
```

## LECmd Sample Output

```
. \LECmd.exe -d "D:\DFIRArtifactMuseum\Windows\LNK\Win10" --mp
```

```
Processing D:\DFIRArtifactMuseum\Windows\LNK\Win10\XPS Viewer.lnk
```

```
Source file: D:\DFIRArtifactMuseum\Windows\LNK\Win10\XPS Viewer.lnk
```

```
Source created: 2022-01-31 17:25:58.4973918
```

```
Source modified: 2022-01-31 17:25:58.0000000
```

```
Source accessed: 2022-06-12 18:38:15.3854180
```

```
--- Header ---
```

```
Target created: null
```

```
Target modified: null
```

```
Target accessed: null
```

```
File size: 0
```

```
Flags: HasName, HasIconLocation,IsUnicode, ForceNoLinkInfo, HasExpString, PreferE\nvironmentPath
```

```
File attributes: 0
```

```
Icon index: -108
```

```
Show window: SwNormal (Activates and displays the window. The window is restored to its original size and position if the window is minimized or maximized.)
```

```
Name: @%systemroot%\system32\XpsRchVw.exe,-103
```

```
Icon Location: %systemroot%\system32\XpsRchVw.exe
```

```
--- Extra blocks information ---
```

```
>> Property store data block (Format: GUID\ID Description ==> Value)
```

```
46588ae2-4cbc-4338-bbfc-139326986dce\0 (Description not available) =\
```

```
=> 0
```

```
9f4c2855-9f79-4b39-a8d0-e1d42de1d5f3\18 App User Model Installed By =\
```

```
=> 1
```

```
>> Environment variable data block
```

```
Environment variables: %systemroot%\system32\xpsrchvw.exe
```

```
----- Processed D:\DFIRArtifactMuseum\Windows\LNK\Win10\XPS Viewer.lnk in 0.004\35160 seconds -----
```

## **LECmd Output**

### **Analyzing LECmd Output - Console**

When parsing .LNK files with LECmd without the -q switch, LECmd will display in the console window parsed data from each .LNK file processed.

```
Administrator: Windows PowerShell
PS D:\ZimmermanTools\net6> .\LECmd.exe -f "D:\DFIRArtifactMuseum\Windows\LNK\Win10\xwfim.lnk.test"
LECmd version 1.5.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/LECmd

Command line: -f D:\DFIRArtifactMuseum\Windows\LNK\Win10\xwfim.lnk.test
Processing D:\DFIRArtifactMuseum\Windows\LNK\Win10\xwfim.lnk.test

Source file: D:\DFIRArtifactMuseum\Windows\LNK\Win10\xwfim.lnk.test
Source created: 2022-01-31 17:25:58
Source modified: 2022-01-31 17:25:58
Source accessed: 2022-06-04 17:31:57

--- Header ---
Target created: 2015-12-25 03:02:06
Target modified: 2015-12-18 17:21:44
Target accessed: 2015-12-25 03:02:09

File size: 34,202,624
Flags: HasTargetIdList, HasLinkInfo, HasWorkingDir,IsUnicode, EnableTargetMetadata
File attributes: FileAttributeArchive
Icon index: 0
Show window: SwNormal (Activates and displays the window. The window is restored to its original size and position if the window is minimized or maximized.)

Working Directory: D:\Dropbox\!Software\xwfim

--- Link information ---
Flags: VolumeIdAndLocalBasePath

>> Volume information
Drive type: Fixed storage media (Hard drive)
Serial number: DAFFD9CE
Label: (No label)
Local path: D:\Dropbox\!Software\xwfim\xWFIM.exe

--- Target ID information (Format: Type ==> Value) ---

Absolute path: My Computer\D:\\\\\
-Root folder: GUID ==> My Computer
-Drive letter ==> D:
-Directory ==> (None)
```

```
Administrator: Windows PowerShell

-Directory ==> (None)
Short name: Dropbox
Modified: 2016-01-24 18:24:26
Extension block count: 1

----- Block 0 (Beef0004) -----
Long name:
Created: 2015-12-25 02:49:22
Last access: 2016-01-24 18:24:26
MFT entry/sequence #: 2977/7 (0xB41/0x7)

-Directory ==> (None)
Short name: !Software
Modified: 2016-01-24 16:58:40
Extension block count: 1

----- Block 0 (Beef0004) -----
Long name:
Created: 2015-12-25 02:49:26
Last access: 2016-01-24 16:58:40
MFT entry/sequence #: 3011/2 (0xBC3/0x2)

-Directory ==> (None)
Short name: xwfim
Modified: 2016-01-21 16:09:52
Extension block count: 1

----- Block 0 (Beef0004) -----
Long name:
Created: 2015-12-25 02:49:28
Last access: 2016-01-21 16:09:52
MFT entry/sequence #: 3063/2 (0xBF7/0x2)

-File ==> (None)
Short name: XWFIM.exe
Modified: 2015-12-18 17:21:44
Extension block count: 1

----- Block 0 (Beef0004) -----
Long name:
Created: 2015-12-25 03:02:08
Last access: 2015-12-25 03:02:10
MFT entry/sequence #: 11268/1 (0x2C04/0x1)

--- End Target ID information ---
--- Extra blocks information ---
>> Tracker database block
```

```
Administrator: Windows PowerShell

-File ==> (None)
Short name: XWFIM.exe
Modified: 2015-12-18 17:21:44
Extension block count: 1

----- Block 0 (Beef0004) -----
Long name:
Created: 2015-12-25 03:02:08
Last access: 2015-12-25 03:02:10
MFT entry/sequence #: 11268/1 (0x2c04/0x1)

--- End Target ID information ---

--- Extra blocks information ---

>> Tracker database block
Machine ID: sagerez
MAC Address: a4:34:d9:43:f3:63
MAC Vendor: INTEL
Creation: 2016-01-24 18:23:46

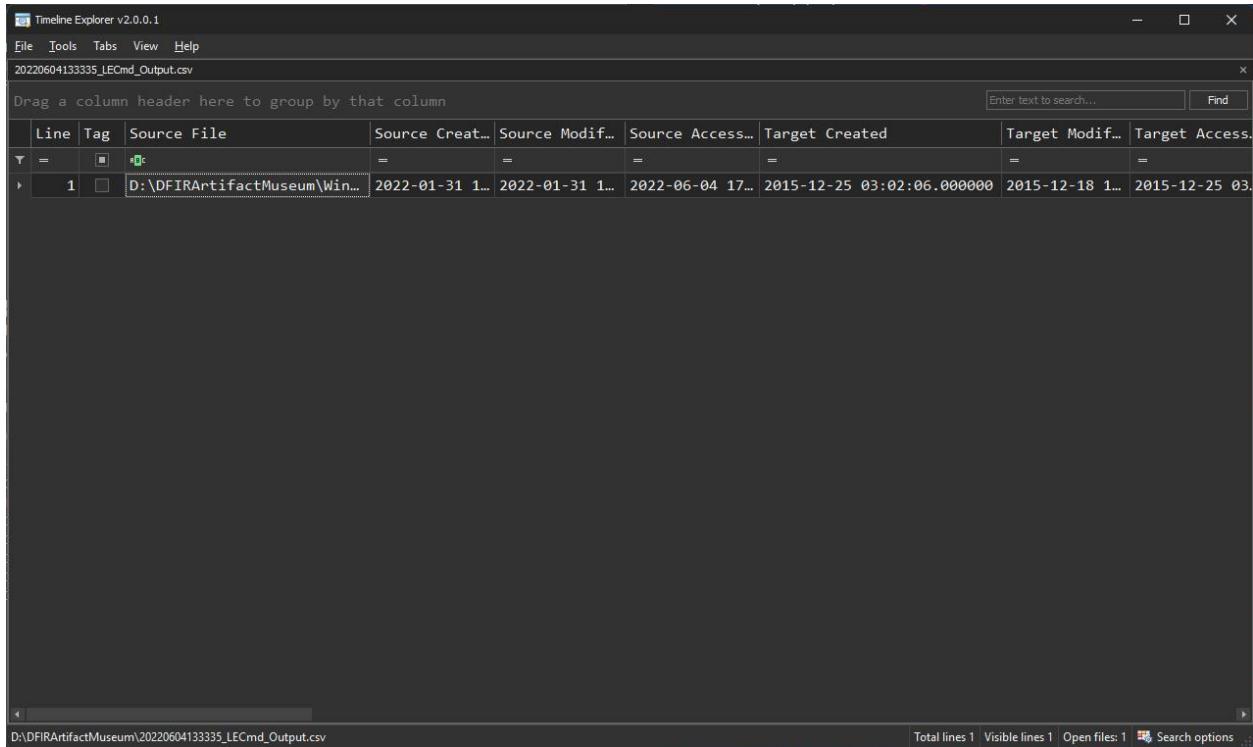
Volume Droid: a8f56eda-fdc9-4d88-ab07-7262c17f9de2
Volume Droid Birth: a8f56eda-fdc9-4d88-ab07-7262c17f9de2
File Droid: 9b53073f-c2c7-11e5-b2a4-a434d943f363
File Droid birth: 9b53073f-c2c7-11e5-b2a4-a434d943f363

>> Property store data block (Format: GUID\ID Description ==> Value)
dabd30ed-0043-4789-a7f8-d013a4736622\100 Item Folder Path Display Narr
ow ==> xwfim (D:\Dropbox\!Software) Item Name Display
b725f130-47ef-101a-a5f1-02608c9eebac\10 Date Created
==> XWFIM.exe
b725f130-47ef-101a-a5f1-02608c9eebac\15 Size
==> 12/25/2015 03:02:08
b725f130-47ef-101a-a5f1-02608c9eebac\12 Type
==> 34202624
b725f130-47ef-101a-a5f1-02608c9eebac\4 Text
==> Application
b725f130-47ef-101a-a5f1-02608c9eebac\14 Date Modified
==> 12/18/2015 17:21:44
28636aa6-953d-11d2-b5d6-00c04fd918d0\30 Parsing Path
==> D:\Dropbox\!Software\xwfim\XWFIM.exe
446d16b1-8dad-4870-a748-402ea43d788c\104 Volume Id
==> Unmapped GUID: 762820c7-0000-0000-0000-100000000000

----- Processed D:\DFIRArtifactMuseum\Windows\LNK\Win10\XWFIM.lnk.test
in 0.11951230 seconds -----
```

## Analyzing LECmd Output - CSV

When parsing .LNK files with LECmd with the `-csv` switch, LECmd will output the parsed data into a CSV file. The CSV file can then be ingested into a tool like Timeline Explorer to analyze the parsed data.



The screenshot shows the Timeline Explorer application window. The title bar reads "Timeline Explorer v2.0.0.1". The menu bar includes "File", "Tools", "Tabs", "View", and "Help". The main window displays a table with one row of data from a CSV file named "20220604133335\_LECmd\_Output.csv". The table has columns: Line, Tag, Source File, Source Creat..., Source Modif..., Source Access..., Target Created, Target Modif..., and Target Access. The data row is as follows:

Line	Tag	Source File	Source Creat...	Source Modif...	Source Access...	Target Created	Target Modif...	Target Access...
1		D:\DFIRArtifactMuseum\Win...	2022-01-31 1...	2022-01-31 1...	2022-06-04 17...	2015-12-25 03:02:06.000000	2015-12-18 1...	2015-12-25 03...

## Analyzing LECmd Output - JSON

When parsing .LNK files with LECmd with the -json switch, LECmd will output the parsed data into a JSON file.

```
1  {
2      "SourceFile": "D:\\DFIRArtifactMuseum\\Windows\\LNK\\Win10\\XWFIM.lnk.test",
3      "SourceCreated": "2022-01-31T17:25:58.4983924+00:00",
4      "SourceModified": "2022-01-31T17:25:58.0000000+00:00",
5      "SourceAccessed": "2022-06-04T17:24:01.7759325+00:00",
6      "TargetCreated": "2015-12-25T03:02:06.2224518+00:00",
7      "TargetModified": "2015-12-18T17:21:44.0000000+00:00",
8      "TargetAccessed": "2015-12-25T03:02:09.8369990+00:00",
9      "FileSize": 34202624,
10     "WorkingDirectory": "D:\\Dropbox\\!Software\\xwfim",
11     "FileAttributes": FileAttributeArchive,
12     "HeaderFlags": "HasTargetIdList, HasLinkInfo, HasWorkingDir,IsUnicode, EnableTargetMetadata",
13     "DriveType": Fixed storage media (Hard drive),
14     "VolumeSerialNumber": DAFDD9CE,
15     "LocalPath": "D:\\Dropbox\\!Software\\xwfim\\XWFIM.exe",
16     "TargetIDAbsolutePath": "My Computer\\D:\\\\\\",
17     "TargetMFTEntryNumber": 0x2C04,
18     "TargetMFTSequenceNumber": 0x1,
19     "MachineID": sagerez,
20     "MachineMACAddress": "a4:34:d9:43:f3:63",
21     "MACVendor": INTEL,
22     "TrackerCreatedOn": "2016-01-24T18:23:46.4858431+00:00",
23     "ExtraBlocksPresent": "TrackerDataBaseBlock, PropertyStoreDataBlock"
24   }
25 }
```

## Analyzing LECmd Output - XML

When parsing .LNK files with LECmd with the `--xml` switch, LECmd will output the parsed data into an XML file.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <CsvOut xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/LECmd">
3      <Arguments i:nil="true" />
4      <CommonPath></CommonPath>
5      <DriveType>Fixed storage media (Hard drive)</DriveType>
6      <ExtraBlocksPresent>TrackerDataBaseBlock, PropertyStoreDataBlock</ExtraBlocksPresent>
7      <FileAttributes>FileAttributeArchive</FileAttributes>
8      <FileSize>34202624</FileSize>
9      <HeaderFlags>HasTargetIdList, HasLinkInfo, HasWorkingDir,IsUnicode, EnableTargetMetadata</HeaderFlags>
10     <LocalPath>D:\Dropbox\!Software\xwfim\XWFIM.exe</LocalPath>
11     <MACVendor>INTEL</MACVendor>
12     <MachineID>sagerez</MachineID>
13     <MachineMACAddress>a4:34:d9:43:f3:63</MachineMACAddress>
14     <NetworkPath></NetworkPath>
15     <RelativePath i:nil="true" />
16     <SourceAccessed>2022-06-04 17:23:45</SourceAccessed>
17     <SourceCreated>2022-01-31 17:25:58</SourceCreated>
18     <SourceFile>D:\DFIRArtifactMuseum\Windows\LNK\Win10\XWFIM.lnk.test</SourceFile>
19     <SourceModified>2022-01-31 17:25:58</SourceModified>
20     <TargetAccessed>2015-12-25 03:02:09</TargetAccessed>
21     <TargetCreated>2015-12-25 03:02:06</TargetCreated>
22     <TargetIDAbsolutePath>My Computer\0:\\\\</TargetIDAbsolutePath>
23     <TargetMFTEntryNumber>0x2C04</TargetMFTEntryNumber>
24     <TargetMFTSequenceNumber>0x1</TargetMFTSequenceNumber>
25     <TargetModified>2015-12-18 17:21:44</TargetModified>
26     <TrackerCreatedOn>2016-01-24 18:23:46</TrackerCreatedOn>
27     <VolumeLabel></VolumeLabel>
28     <VolumeSerialNumber>DAFFD9CE</VolumeSerialNumber>
29     <WorkingDirectory>D:\Dropbox\!Software\xwfim</WorkingDirectory>
30   </CsvOut>
```

## Analyzing LECmd Output - HTML

When parsing .LNK files with LECmd with the `-html` switch, LECmd will output the parsed data into an HTML file. Please note, the HTML file will be placed into a dedicated subfolder, which is not normally the case when outputting to CSV, XML, or JSON.

```
D:\DFIRArtifactMuseum\Windows\LINK\Win10\xWFIM.lnk.test
Source_Created: 2022-01-31 17:25:58
Source_Modified: 2022-01-31 17:25:58
Source_Accessed: 2022-06-04 17:23:53
Target_Created: 2015-12-25 03:02:06
Target_Modified: 2015-12-18 17:21:44
Target_Accessed: 2015-12-25 03:02:09
File_Size: 34202624 (bytes)
Relative_Path:
Working_Directory: D:\Dropbox\!Software\xwfim
File_Attributes: FileAttributeArchive
Header_Flags: HasTargetIdList, HasLinkInfo, HasWorkingDir,IsUnicode, EnableTargetMetadata
Drive_Type: Fixed storage media (Hard drive)
Volume_Serial_Number: DAFFD9CE
Volume_Label:
Local_Path: D:\Dropbox\!Software\xwfim\xWFIM.exe
Network_Path:
Common_Path:
Arguments:
TargetID_Absolute_Path: My Computer\D:\\\\
Target_$MFT_Entry_Number: 0x2C04
Target_$MFT_Sequence_Number: 0x1
MachineID: sagerez
Machine_MAC_Address: a4:34:d9:43:f3:63
MAC_Vendor: INTEL (vendor not included in source .lnk file, auto-resolved by LECmd for end-user upon parsing)
Tracker_Created_On: 2016-01-24 18:23:46
Extra_Blocks_Present: TrackerDataBaseBlock, PropertyStoreDataBlock
```

## LECmd Key Takeaways

LECmd parses .LNK files, which are indicative of file access. .LNK files are created when a file is opened for the first time. If a .LNK file exists for a given file or folder, then that file or folder has been accessed at least once by a user.

LECmd output, regardless of file format, will provide potentially useful information about user activity on a given Windows system.

## Important Data Points

### Timestamps

LECmd output will provide the following timestamps:

- \* `SourceCreated` - The creation timestamp associated with the .LNK file itself
- \* `SourceModified` - The modified timestamp associated with the .LNK file itself
- \* `SourceAccessed` - The accessed timestamp associated with the .LNK file itself
- \* `TargetCreated` - The creation timestamp associated with the file the .LNK file points to
- \* `TargetModified` - The modified timestamp associated with the file the .LNK file points to
- \* `TargetAccessed` - The accessed timestamp associated with the file the .LNK file points to

`SourceCreated` will be the most useful indicator of when a file was first opened by a user. `SourceModified` will be the most useful indicator of when the file was last opened by a user.

The Target timestamps can be useful to cross reference with the timestamps in the MFT. Usually, these timestamps are the 0x10 (\$Standard\_Information) timestamps from the \$MFT. In testing, it appears the timestamps stored within the .LNK file are “refreshed” when a file is opened again. The timestamps stored within a .LNK file are therefore a snapshot in time of when the .LNK file was last modified via opening a file. This can be useful in that if a file was timestamped (aka timestamps altered), those altered timestamps won’t reflect in the .LNK file until the timestamped file has been opened AFTER being timestamped.

### Other Attributes

#### SourceFile

`SourceFile` will provide the file path to the LNK file being parsed.

#### FileSize

`FileSize` will provide the file size of the target file that the .LNK file is pointing to.

#### WorkingDirectory

`WorkingDirectory` will provide the parent directory that the target file resides in.

**DriveType**

DriveType will provide insight as to what type of drive the SourceFile resided on at the time of file access.

**VolumeSerialNumber**

VolumeSerialNumber will provide the Volume Serial Number of the drive the target file resided on at the time of file access.

**LocalPath**

LocalPath will provide the full path to the target file that was accessed by the user.

**TargetMFTEntryNumber**

TargetMFTEntryNumber will provide the MFT Entry number for the Target File in hexadecimal and decimal.

**TargetMFTSequenceNumber**

TargetMFTSequenceNumber will provide the MFT Sequence number for the Target File in hexadecimal and decimal.

**MachineID**

MachineID will provide the hostname of the hostnam the target file resided on at the time of file access, if the tracker database block is present within the LNK file.

**MachineMACAddress**

MachineMACAddress will provide the MAC Address for the host, if the tracker database block is present within the LNK file.

**MACVendor**

MACVendor will resolve the MAC address to provide the associated vendor, if known, and if the tracker database block is present within the LNK file..

**ExtraBlocksPresent**

ExtraBlocksPresent will provide insight as to whether or not extra data blcoks are present within the LNK file. The other blocks that are possible to be present within a LNK file are: TrackerDataBase-Block and PropertyStoreDataBlock.

# LECmd References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/Lnk> is the C# library for parsing .LNK files
- <https://github.com/EricZimmerman/LECmd> is the Command Line wrapper for the .LNK library

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, [Binary Foray](#)<sup>75</sup>:

- [Introducing LECmd!](#)<sup>76</sup>
- [LECmd v0.6.0.0 released!](#)<sup>77</sup>
- [PECmd, LECmd, and JLECmd updated!](#)<sup>78</sup>
- [LECmd and JLECmd updated](#)<sup>79</sup>

## Download LECmd

LECmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

## LNK Sample Data

Sample LNK files can be found at the [DFIRArtifactMuseum](#)<sup>80</sup>.

---

<sup>75</sup><https://binaryforay.blogspot.com/>

<sup>76</sup><https://binaryforay.blogspot.com/2016/02/introducing-lecmd.html>

<sup>77</sup><https://binaryforay.blogspot.com/2016/02/lecmd-v0600-released.html>

<sup>78</sup><https://binaryforay.blogspot.com/2016/03/pecmd-lecmd-and-jlecmd-updated.html>

<sup>79</sup><https://binaryforay.blogspot.com/2016/04/lecmd-and-jlecmd-updated.html>

<sup>80</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/LNK>

# **MFTECmd**

## **MFTECmd Introduction**

MFTECmd is a tool created by Eric Zimmerman used to parse various NTFS metadata artifacts, including but not limited to the \$MFT, \$J, \$Boot, \$SDS, and \$I30 files.

## **File Types Parsed by MFTECmd**

### **#SDS**

\$SDS contains a list of all the security descriptors on a given volume. Learn more about the \$SDS [here<sup>81</sup>](#).

### **\$Boot**

\$Boot is the NTFS metadata file for the NTFS Partition Boot Sector. Learn more about the \$Boot [here<sup>82</sup>](#).

### **\$MFT**

\$MFT contains metadata about every file and folder on an NTFS volume within FILE records. Learn more about the \$MFT [here<sup>83</sup>](#).

### **\$J**

\$J tracks changes that were made to files and folders on an NTFS volume. Learn more about the \$J [here<sup>84</sup>](#).

### **\$I30**

\$I30 files can provide insight into deleted and overwritten files. Learn more about \$I30 files [here<sup>85</sup>](#).

---

<sup>81</sup><https://www.ntfs.com/ntfs-permissions-file-structure.htm>

<sup>82</sup><https://www.ntfs.com/ntfs-partition-boot-sector.htm>

<sup>83</sup><https://www.ntfs.com/ntfs-mft.htm>

<sup>84</sup>[https://en.wikipedia.org/wiki/USN\\_Journal](https://en.wikipedia.org/wiki/USN_Journal)

<sup>85</sup>[https://www.sans.org/blog/ntfs-i30-index-attributes-evidence-of-deleted-and-overwritten-files/#:~:text=Interestingly%2C%20NTFS%20directory%20index%20entries%20utilize%20a%20%24FILE\\_NAME,trove%20of%20information%20about%20the%20file%3A%20Full%20filename](https://www.sans.org/blog/ntfs-i30-index-attributes-evidence-of-deleted-and-overwritten-files/#:~:text=Interestingly%2C%20NTFS%20directory%20index%20entries%20utilize%20a%20%24FILE_NAME,trove%20of%20information%20about%20the%20file%3A%20Full%20filename)

## **MFTECmd Use Cases**

### **Law Enforcement**

For those in Law Enforcement, this tool is useful for parsing NTFS metadata files that can provide invaluable information relating to the files that currently exist, previously existed, and what changes have occurred to those files and at what time. The \$MFT should be examined in nearly every instance as that will provide insight as to which files and folders exist on a system currently. The \$J should be examined in nearly every instance as it will provide insight as to which files changed in which ways, including but not limited to file creations, file deletions, file renames, etc. The \$I30 can provide insight into folders that no longer exist on a system.

### **Private Sector**

For those in the Private Sector, this tool is useful for parsing NTFS metadata files that can provide invaluable information relating to the files that currently exist, previously existed, and what changes have occurred to those files and at what time. The \$MFT can be useful in locating suspicious files and folders that were created by a threat actor during a period of unauthorized access. The \$J can provide insight into files that no longer exist on disk and no longer have references within the \$MFT, including but not limited to threat actor tool output or evidence of malicious executables that were deleted after being executed.

## MFTECmd Switches

In a PowerShell window, running .\MFTECmd.exe will provide the following options when running MFTECmd:

```

-f <f>           File to process ($MFT | $J | $Boot | $SDS | $I30). Required
-m <m>           $MFT file to use when -f points to a $J file (Use this to resolve\
parent path       in $J CSV output).
--json <json>    Directory to save JSON formatted results to. This or --csv requir\
ed unless --de   ed unless --de
                  or --body is specified
--jsonf <jsonf>  File name to save JSON formatted results to. When present, overri\
des default     des default
                  name
--csv <csv>      Directory to save CSV formatted results to. This or --json requir\
ed unless --de   ed unless --de
                  or --body is specified
--csvf <csvf>    File name to save CSV formatted results to. When present, overrid\
es default name
--body <body>     Directory to save bodyfile formatted results to. --bdl is also re\
quired when      quired when
                  using this option
--bodyf <bodyf>  File name to save body formatted results to. When present, overri\
des default     des default
                  name
--bdl <bdl>      Drive letter (C, D, etc.) to use with bodyfile. Only the drive le\
tter itself      tter itself
                  should be provided
--blf             When true, use LF vs CRLF for newlines [default: False]
--dd <dd>         Directory to save exported FILE record. --do is also required whe\
n using this      n using this
                  option
--do <do>         Offset of the FILE record to dump as decimal or hex. Ex: 5120 or \
0x1400 Use        0x1400 Use
                  --de or --debug to see offsets
--de <de>         Dump full details for entry/sequence #. Format is 'Entry' or 'Ent\
ry-Seq' as        decimal or hex. Example: 5, 624-5 or 0x270-0x5.
--dr              When true, dump resident files to dir specified by --csv, in 'Res\
ident'           ident'
                  subdirectory. Files will be named

```

```
'<EntryNumber>-<SequenceNumber>_<FileName>.bin'
--fls When true, displays contents of directory specified by --de. Igno\
red when --de

--ds <ds> points to a file [default: False]
or 0x270 Dump full details for Security Id as decimal or hex. Example: 624\

--dt <dt> The custom date/time format to use when displaying time stamps. S\
ee

https://goo.gl/CNVq0k for options [default: yyyy-MM-dd HH:mm:ss.f\
ffffff]

--sn Include DOS file name types [default: False]
--fl Generate condensed file listing. Requires --csv [default: False]
--at When true, include all timestamps from 0x30 attribute vs only whe\
n they differ

--rs from 0x10 [default: False]
When true, recover slack space from FILE records when processing \
MFT files.

This option has no effect for $I30 files [default: False]
--vss Process all Volume Shadow Copies that exist on drive specified by\
-f [default:
False]

--dedupe Deduplicate -f & VSCs based on SHA-1. First file found wins [defa\
ult: False]

--debug Show debug information during processing [default: False]
--trace Show trace information during processing [default: False]
--version Show version information
-?, -h, --help Show help and usage information
```

## Switch Descriptions

### -m

This switch informs the tool to parse an \$MFT along with its associated \$J. The main benefit for this is the resolution of the Parent Path column in the \$J output, which will save examiners from having to manually cross reference the Parent Entry ID in the \$MFT to the one associated with a change recorded in the \$J. This is absolutely worth the extra effort.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\J\Win10\APTSimulatorVM\\$J' -m 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --csv "C:\temp"

### --body

This switch informs the tool to output to a [bodyfile](#)<sup>86</sup>. Please note, the --bd1 switch is required when using --bd1.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --body C:\temp --bd1 C

### --bodyf

This switch informs the tool to name the bodyfile output a specified filename.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --body C:\temp --bd1 C --bodyf test

The above command will output a file named test.

### --bd1

This switch informs the tool to use a specified drive letter when providing output to bodyfile format.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --body C:\temp --bd1 C

### --blf

This switch informs the tool to use LF instead of CRLF for line endings in . More information about LF vs. CRLF can be found [here](#)<sup>87</sup>.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --csv "C:\temp" --blf

<sup>86</sup><https://forensicscwiki.xyz/wiki/index.php?title=Bodyfile>

<sup>87</sup><https://www.aleksandrkhannisan.com/blog/crlf-vs-lf-normalizing-line-endings-in-git/>

**--dd**

This switch informs the tool to save an exported FILE record to a specified directory. Please note, the --do switch is required when using --dd.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --dd C:\temp --do 679

**--do**

This switch informs the tool of which decimal or hex offset to dump to a specified location.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --dd C:\temp --do 679

**--de**

This switch informs the tool to dump the full details of a specific entry number.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --de 679

**--dr**

This switch informs the tool to dump all files resident within the \$MFT to a specified directory. Please note, the specified directory will be whatever is specified for the --csv switch.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --csv c:\temp --dr

**--fls**

This switch informs the tool to display the contents of a folder based on the data stored within the \$MFT.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --de 500 --fls

The above command will display results similar to below:

```

1 Contents for .\Program Files\WindowsApps\Microsoft.WebMediaExtensions_1.0.42192.0_x6\
2 4_8wekyb3d8bbwe\microsoft.system.package.metadata
3
4 Key Type Name
5 115558-4 File resources.ae44601c.pri
6 115558-4 File S-1-5-21-2212929841-2730996645-2649195864-10\
7 00-MergedResources-1.pri

```

Removing the --f1s switch would display the timestamps and other NTFS metadata stored about that directory within the \$MFT.

--ds

This switch informs the tool to provide information about the specified security descriptor.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\Secure\_-\$SDS\Win10\APTSimulatorVM\\$Secure\_\$SDS' --ds 256

The above command would yield the following output:

```

1 Command line: -f D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\Secure_$SDS\Win10\APTSi\
2 mulatorVM\$Secure_$SDS --ds 256
3
4 File type: Sds
5
6
7 Processed D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\Secure_$SDS\Win10\APTSimulator\
8 VM\$Secure_$SDS in 0.0042 seconds
9
10 SDS entries found in D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\Secure_$SDS\Win10\A\
11 PTSimulatorVM\$Secure_$SDS: 4,626
12
13 Details for security record # 256 (0x100), Found in D:\DFIRArtifactMuseum\Windows\NT\
14 FSArtifacts\Secure_$SDS\Win10\APTSimulatorVM\$Secure_$SDS
15 Hash value: CBC6FE32, Offset: 0x0
16 Control flags: SeDaclPresent | SeSelfRelative
17
18 Owner SID: S-1-5-18: An account that is used by the operating system.
19 Group SID: S-1-5-32-544: A built-in group. After the initial installation of the operating system, the only member of the group is the Administrator account. When a computer joins a domain, the Domain Administrators group is added to the Administrators group. When a server becomes a domain controller, the Enterprise Administrators group also is added to the Administrators group.
20
21
22
23
24

```

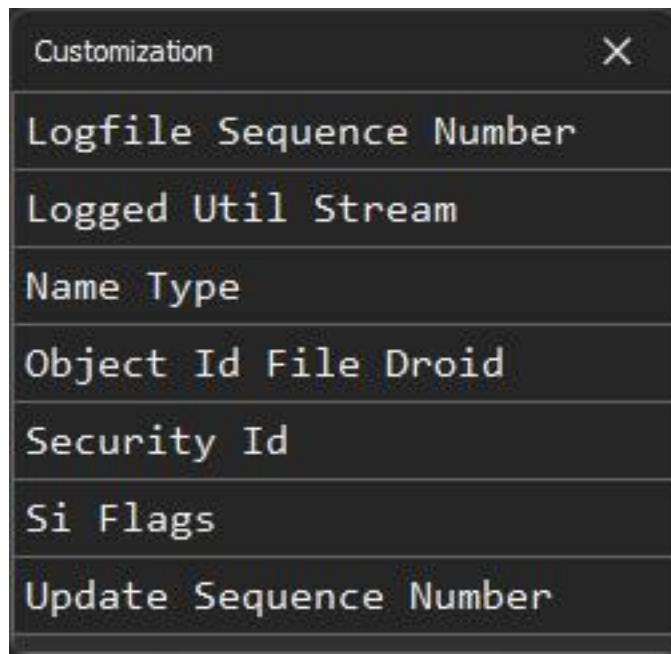
```
25 Discretionary Access Control List
26 ACE record count: 2
27 ACL type: Discretionary
28
29 ----- Ace record #0 -----
30 Type: AccessAllowed
31 Flags: None
32 Mask: FileReadDirList | ReadEa | ReadAttrs | ReadControl | Synchronize
33 SID: S-1-5-18: An account that is used by the operating system.
34
35 ----- Ace record #1 -----
36 Type: AccessAllowed
37 Flags: None
38 Mask: FileReadDirList | ReadEa | ReadAttrs | ReadControl | Synchronize
39 SID: S-1-5-32-544: A built-in group. After the initial installation of the operating\
40 system, the only member of the group is the Administrator account. When a computer \
41 joins a domain, the Domain Administrators group is added to the Administrators group\
42 . When a server becomes a domain controller, the Enterprise Administrators group als\
43 o is added to the Administrators group.
```

**--sn**

This switch informs the tool to include DOS file names as well as a few other columns of output.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --csv C:\temp --sn

The above command output a 191MB CSV compared to 122MB without the --sn switch. Also, the above command adds the following columns within Timeline Explorer that are available in the Column Chooser (right-click on a column header and double click on them to add them in).



mftecmdSNswitchcolumnsTLE

#### --f1

This switch informs the tool to output a simple file listing of the contents of the \$MFT parsed which will include the following columns: FullPath, Extension, IsDirectory, FileSize, Created0x10, and LastModified0x10.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --csv C:\temp --f1

#### --at

This switch informs the tool to display the timestamps even when the 0x10 and 0x30 timestamps are identical. Normally, the 0x30 timestamps would be empty if they're identical to the 0x10 timestamps for the purpose of making anomalies easier to spot.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --csv C:\temp --at

#### --rs

This switch informs the tool to attempt to recover slack space from FILE records.

Example: .\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\MFT\Win10\APTSimulatorVM\\$MFT' --csv C:\temp --rs

## MFTECmd Command Examples

### Example MFTECmd Commands

**Parse a \$MFT and output to CSV to a specified location and name the output file MyOutputFile.csv**

```
MFTECmd.exe -f "C:\Temp\SomeMFT" --csv "c:\temp\out" --csvf MyOutputFile.csv
```

**Parse a \$MFT and output to CSV to a specified location**

```
MFTECmd.exe -f "C:\Temp\SomeMFT" --csv "c:\temp\out"
```

**Parse a \$MFT and output to JSON to a specified location**

```
MFTECmd.exe -f "C:\Temp\SomeMFT" --json "c:\temp\jsonout"
```

**Parse a \$MFT and output to bodyfile for the C:\ drive**

```
MFTECmd.exe -f "C:\Temp\SomeMFT" --body "c:\temp\bout" --bd1 c
```

**Parse a \$MFT and dump the full details for entry number 5, sequence number 5 to the console**

```
MFTECmd.exe -f "C:\Temp\SomeMFT" --de 5-5
```

#### Parse \$Boot and output to console

```
.\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\Boot\Win10\APTSimulatorVM\$Boot'
```

**Parse \$SDS and output to CSV to a specified location**

```
.\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\Secure_$SDS\Win10\APTSimulatorVM\$Secure_$SDS' --csv C:\temp
```

**Parse \$I30 and output to CSV to a specified location**

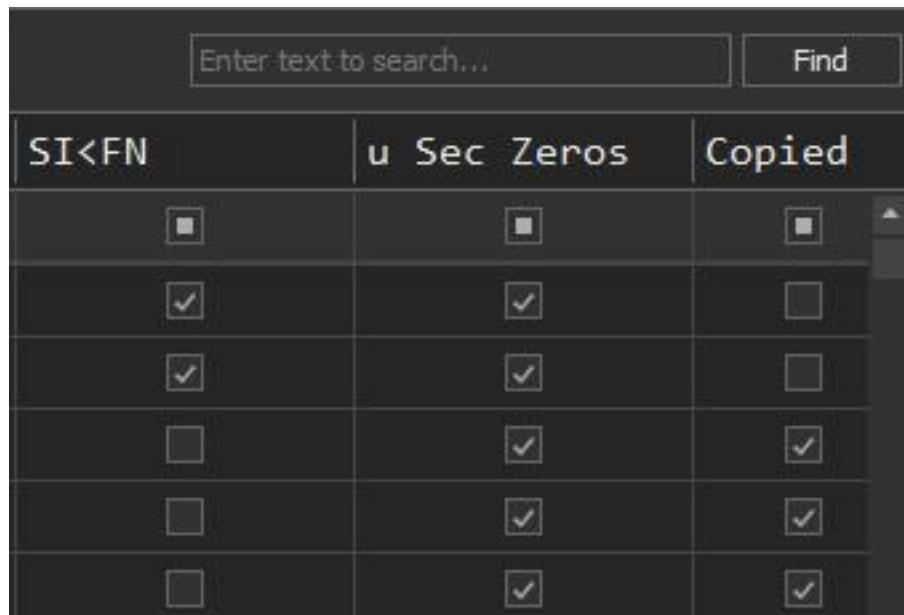
```
.\MFTECmd.exe -f 'D:\DFIRArtifactMuseum\Windows\NTFSArtifacts\$I30\EricZimmerman\Sec\ondDelete\$I30' --csv c:\temp
```

## MFTECmd Output

### Analyzing MFTECmd Output - CSV

MFTECmd output provides a few important data points that are not native to NTFS metadata files but are still very helpful to examiners.

When parsing the \$MFT, output provided will contain three data points unique to MFTECmd:



The screenshot shows a table with three columns: 'SI < FN', 'u Sec Zeros', and 'Copied'. The first column contains binary values (0 or 1). The second column contains binary values (0 or 1). The third column contains binary values (0 or 1). There is a search bar at the top and a 'Find' button.

SI < FN	u Sec Zeros	Copied
0	0	0
1	1	0
1	1	0
0	1	1
0	1	1
0	1	1

mftecmd3columns

#### SI < FN

SI < FN can be translated to Standard Information (0x10) Attribute timestamp is less than (aka earlier than) the File Name (0x30) Attribute timestamp. This is meant to point out a potential indicator of timestamping occurring on the volume being analyzed. Learn more about timestamping [here<sup>88</sup>](#) and [here<sup>89</sup>](#).

#### u Sec Zeros

u Sec Zeros indicates that the subseconds of a timestamp has been zeroed out. An example of a zeroed out subsecond value would be 2022-12-31 12:34:56.0000000. Prior to being timestamped, that timestamp likely looked similar to this: 2022-12-31 12:34:56.1234567 where the subsecond values were not zeroed out.

<sup>88</sup><https://www.kroll.com/en/insights/publications/cyber/anti-forensic-tactics/anti-forensics-tactics-timestamping>

<sup>89</sup><https://www.inversecos.com/2022/04/defence-evasion-technique-timestamping.html>

## Copied

`Copied` indicates that the Last Modified timestamp value occurs before the Creation timestamp of a given file or folder. How can this be? During a file copy operation, the file being copied's Creation timestamp is untouched but the Last Modified timestamp will reflect the time of file copy. This can often result in a file showing as being modified before it existed, which is impossible. MFTECmd recognizes this and will provide an indicator in this `Copied` column as to whether or not the timestamps indicate such a file copy operation may have occurred.

## Analyzing MFTECmd Output - JSON

Executing MFTECmd with JSON output enabled will provide results similar to the example below:

```
1  {
2      "EntryNumber": 0,
3      "SequenceNumber": 1,
4      "ParentEntryNumber": 5,
5      "ParentSequenceNumber": 5,
6      "InUse": true,
7      "ParentPath": ".",
8      "FileName": "$MFT",
9      "Extension": "",
10     "IsDirectory": false,
11     "HasAds": false,
12     "IsAds": false,
13     "FileSize": 274202624,
14     "Created0x10": "2022-03-10T19:30:21.3503859+00:00",
15     "LastModified0x10": "2022-03-10T19:30:21.3503859+00:00",
16     "LastRecordChange0x10": "2022-03-10T19:30:21.3503859+00:00",
17     "LastAccess0x10": "2022-03-10T19:30:21.3503859+00:00",
18     "UpdateSequenceNumber": 0,
19     "LogFileSequenceNumber": 960937622,
20     "SecurityId": 256,
21     "SiFlags": 6,
22     "ReferenceCount": 1,
23     "NameType": 3,
24     "Timestomped": false,
25     "uSecZeros": false,
26     "Copied": false,
27     "FnAttributeId": 3,
28     "OtherAttributeId": 6
29 },
```

## Analyzing MFTECmd Output - Body

Executing MFTECmd with bodyfile output enabled will provide results similar to the example below:

```
1 |0|c:/$MFT|0-128-6|r/rrwxrwxrwx|0|0|274202624|1646940621|1646940621|1646940621|164694\\
2 |0621
3 |0|c:/$MFT ($FILE_NAME)|0-48-3|r/rrwxrwxrwx|0|0|274202624|1646940621|1646940621|164694\\
4 |0621|1646940621
5 |0|c:/$MFTMirr|1-128-1|r/rrwxrwxrwx|0|0|4096|1646940621|1646940621|1646940621|164694\\
6 |0621
7 |0|c:/$MFTMirr ($FILE_NAME)|1-48-2|r/rrwxrwxrwx|0|0|4096|1646940621|1646940621|164694\\
8 |0621|1646940621
9 |0|c:/$LogFile|2-128-1|r/rrwxrwxrwx|0|0|67108864|1646940621|1646940621|1646940621|164\\
10 |6940621
11 |0|c:/$LogFile ($FILE_NAME)|2-48-2|r/rrwxrwxrwx|0|0|67108864|1646940621|1646940621|1646940621|16\\
12 |46940621|1646940621
13 |0|c:/$Volume|3-128-3|r/rrwxrwxrwx|0|0|0|1646940621|1646940621|1646940621|1646940621
14 |0|c:/$Volume ($FILE_NAME)|3-48-1|r/rrwxrwxrwx|0|0|0|1646940621|1646940621|1646940621|1646940621\\
15 |1646940621
16 |0|c:/$AttrDef|4-128-1|r/rrwxrwxrwx|0|0|2560|1646940621|1646940621|1646940621|164694\\
17 |0621
18 |0|c:/$AttrDef ($FILE_NAME)|4-48-2|r/rrwxrwxrwx|0|0|2560|1646940621|1646940621|1646940621|164694\\
19 |0621|1646940621
```

# MFTECmd References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/MFT> is the C# library for parsing \$MFT files
- <https://github.com/EricZimmerman/MFTECmd> is the Command Line wrapper for the MFT library

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- Introducing MFTECmd!<sup>90</sup>
- MFTECmd v0.2.6.0 released<sup>91</sup>
- MFTECmd 0.3.6.0 released<sup>92</sup>
- Locked file support added to AmcacheParser, AppCompatCacheParser, MFTECmd, ShellBags Explorer (and SBECmd), and Registry Explorer (and RECmd)<sup>93</sup>

## Community Resources

- AboutDFIR - MFT Explorer/MFTECmd<sup>94</sup>
- 13Cubed - NTFS Journal Forensics<sup>95</sup>
- 13Cubed - Introduction to MFTECmd - NTFS MFT and Journal Forensics<sup>96</sup>
- MFTECmd\$J\$MFTParser.ps1<sup>97</sup>

## Download MFTECmd

MFTECmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

## NTFS Sample Data

Sample NTFS artifacts can be found at the [DFIRArtifactMuseum](#)<sup>98</sup>.

<sup>90</sup><https://binaryforay.blogspot.com/2018/06/introducing-mftecnd.html>

<sup>91</sup><https://binaryforay.blogspot.com/2018/06/mftecnd-v0260-released.html>

<sup>92</sup><https://binaryforay.blogspot.com/2018/12/mftecnd-0360-released.html>

<sup>93</sup><https://binaryforay.blogspot.com/2019/01/locked-file-support-added-to.html>

<sup>94</sup><https://aboutdfir.com/toolsandartifacts/windows/mft-explorer-mftecnd/>

<sup>95</sup><https://www.youtube.com/watch?v=1mwiShxREm8>

<sup>96</sup>[https://www.youtube.com/watch?v=\\_qElVZJqlGY](https://www.youtube.com/watch?v=_qElVZJqlGY)

<sup>97</sup><https://github.com/AndrewRathbun/DFIRPowerShellScripts/blob/main/MFTECmd%24J%24MFTParser.ps1>

<sup>98</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/NTFSArtifacts>

# **PECmd**

## **PECmd Introduction**

PECmd is a tool created by Eric Zimmerman used to parse Windows Prefetch files (.pf), which can be located at C:\Windows\Prefetch.

## **PECmd Use Cases**

### **Law Enforcement**

For those in Law Enforcement, this tool is useful for parsing Prefetch files which can help provide evidence of program execution. This may be useful in situations where crimes are being investigated that involve media players and photo viewers. Historical run times of the program can be useful in establishing a pattern of activity relating to application use.

### **Private Sector**

For those in the Private Sector, this tool is useful for parsing Prefetch files which can help provide evidence of execution of suspicious applications used by a threat actor during a period of unauthorized access.

## PECmd Switches

In a PowerShell window, running .\PECmd.exe will provide the following options when running PECmd:

-f <f>	File to <b>process</b> . Either this or -d is required
-d <d>	Directory to recursively <b>process</b> . Either this or -f is required
-k <k>	Comma separated list of keywords to highlight <b>in</b> output. By default, 'temp' and
	'tmp' are highlighted. Any additional keywords will be added to these
-o <o>	When specified, save prefetch file bytes to the given path. Useful for looking at
	decompressed Win10 files
-q	<b>Do</b> not dump full details about each file processed. Speeds up processing when
	using --json or --csv [ <b>default</b> : False]
--json <json>	Directory to save JSON formatted results to. Be sure to include the full path
	<b>in</b> double quotes
--jsonf <jsonf>	File name to save JSON formatted results to. When present, overrides <b>default</b>
	name
--csv <csv>	Directory to save CSV formatted results to. Be sure to include the full path <b>in</b>
	double quotes
--csvf <csvf>	File name to save CSV formatted results to. When present, overrides <b>default</b> name
	name
--html <html>	Directory to save XHTML formatted results to. Be sure to include the full path
	<b>in</b> double quotes
--dt <dt>	The custom date/time format to use when displaying time stamps. See <a href="https://goo.gl/CNVq0k">https://goo.gl/CNVq0k</a> for options [ <b>default</b> : yyyy-MM-dd HH:mm:ss]
--mp	When true, display higher precision for timestamps [ <b>default</b> : False]
e]	
--vss	<b>Process</b> all Volume Shadow Copies that exist on drive specified by
-f or -d	[ <b>default</b> : False]
--dedupe	Deduplicate -f or -d & VSCs based on SHA-1. First file found wins
[ <b>default</b> : False]	

```
--debug      Show debug information during processing [default: False]
--trace      Show trace information during processing [default: False]
--version    Show version information
-?, -h, --help Show help and usage information
```

## Switch Descriptions

-k

This switch informs the tool to highlight a list of keywords (comma separated)

Example: .\PECmd.exe -f "C:\temp\prefetch\TIMELINEEXPLORER.EXE-959F92B3.pf" -k timeline,explorer

Within the results, one can see the (Keyword True) appended to the end of a line where the specified keyword appears:

```
19: \VOLUME{01d7b51f5d5ed7cd-b45d68a4}\USERS\CFUSER\ONEDRIVE
20: \VOLUME{01d7b51f5d5ed7cd-b45d68a4}\USERS\CFUSER\ONEDRIVE\Desktop
21: \VOLUME{01d7b51f5d5ed7cd-b45d68a4}\USERS\CFUSER\ONEDRIVE\Desktop\EZ_TOOLS
22: \VOLUME{01d7b51f5d5ed7cd-b45d68a4}\USERS\CFUSER\ONEDRIVE\Desktop\EZ_TOOLS\NET6
23: \VOLUME{01d7b51f5d5ed7cd-b45d68a4}\USERS\CFUSER\ONEDRIVE\Desktop\EZ_TOOLS\NET6\T\
IMELINEEXPLORER (Keyword True)
```

-o

This switch informs the tool to output the decompressed bytes of the Prefetch file to a specified location. Please note, this works in Windows 10 and newer as those operating systems compress Prefetch files.

Example: .\PECmd.exe -f "C:\temp\prefetch\TIMELINEEXPLORER.EXE-959F92B3.pf" -o D:\Prefetch

Please note, in the above example a binary file called Prefetch will be output to the root of the D:\ drive. Do not expect PECmd to output the bytes as a binary file into the D:\Prefetch folder.

--mp

This switch will instruct PECmd to provide more verbose timestamps. For instance, running .\PECmd.exe -f "C:\temp\prefetch\TIMELINEEXPLORER.EXE-959F92B3.pf" --mp resulted in the following:

```
Created on: 2022-05-28 03:22:25.3924904
Modified on: 2022-07-08 12:42:40.1435546
Last accessed on: 2022-09-09 03:03:13.7696931
```

Running .\PECmd.exe -f "C:\temp\prefetch\TIMELINEEXPLORER.EXE-959F92B3.pf" resulted in the following:

Created on: 2022-05-28 03:22:25  
Modified on: 2022-07-08 12:42:40  
Last accessed on: 2022-09-09 03:03:23

**--vss**

This switch informs the tool to mount Volume Shadow Copies from the drive letter specified using the -f or -d switch and parse Prefetch files present within.

Example: .\PECmd.exe -d "C:\temp\prefetch" --vss

# PECmd Command Examples

## Example PECmd Commands

### Parse a Prefetch file and output to the console window

```
PECmd.exe -f "C:\Temp\CALC.EXE-3FBEF7FD.pf"
```

### Parse a Prefetch file and output to formatted JSON to a specified location

```
PECmd.exe -f "C:\Temp\CALC.EXE-3FBEF7FD.pf" --json "D:\jsonOutput" --jsonpretty
```

### Parse a directory of Prefetch files and output to console window with specified keywords highlighted in the output

```
PECmd.exe -d "C:\Temp" -k "system32, fonts"
```

### Parse a directory of Prefetch and output to CSV to a specified location with the filename foo.csv and output to JSON to a specified location

```
PECmd.exe -d "C:\Temp" --csv "c:\temp" --csvf foo.csv --json c:\temp\json
```

### Parse a directory of Prefetch files

```
PECmd.exe -d "C:\Windows\Prefetch"
```

## PECmd Output

### Analyzing PECmd Output - CSV

PECmd will output multiple CSVs with the following filenames:

```
%timestamp%_PECmd_Output.csv  
%timestamp%_PECmd_Output_Timeline.csv
```

The timeline output will provide all of the entries from the PECmd output in a single timeline sorted on the run time stored within Prefetch.

## Analyzing PECmd Output - JSON

Here's an example of PECmd's JSON output:

```
1  {
2      "SourceFilename": "C:\\\\temp\\\\prefetch\\\\TIMELINEEXPLORER.EXE-959F92B3.pf",
3      "SourceCreated": "2022-05-28 03:22:25",
4      "SourceModified": "2022-07-08 12:42:40",
5      "SourceAccessed": "2022-09-09 03:22:30",
6      "ExecutableName": "TIMELINEEXPLORER.EXE",
7      "Hash": "959F92B3",
8      "Size": "447074",
9      "Version": "Windows 10 or Windows 11",
10     "RunCount": "24",
11     "LastRun": "2022-07-08 12:42:38",
12     "PreviousRun0": "2022-06-29 17:56:46",
13     "PreviousRun1": "2022-06-28 13:06:32",
14     "PreviousRun2": "2022-06-26 02:57:37",
15     "PreviousRun3": "2022-06-22 18:12:52",
16     "PreviousRun4": "2022-06-13 20:37:15",
17     "PreviousRun5": "2022-06-11 20:57:04",
18     "PreviousRun6": "2022-06-10 17:50:59",
19     "Volume0Name": "\\\\VOLUME{01d7b51f5d5ed7cd-b45d68a4}",
20     "Volume0Serial": "B45D68A4",
21     "Volume0Created": "2021-09-29 10:47:14",
22     "Directories": TooManyToListHere
23     "FilesLoaded": TooManyToListHere
24     "ParsingError": false
```

## Analyzing PECmd Output - HTML

PECmd's JSON output will be very similar to the JSON output, except in an HTML file.

## PECmd Key Takeaways

### Important Data Points

Prefetch files can store multiple run times for a single executable. This can help track historical execution of a file of interest.

# PECmd References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/Prefetch> is the C# library for parsing Prefetch files
- <https://github.com/EricZimmerman/PECmd> is the Command Line wrapper for the Prefetch library

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- Windows Prefetch parser in C#<sup>99</sup>
- Introducing PECmd!<sup>100</sup>
- PECmd v0.6.0.0 released<sup>101</sup>
- PECmd, LECmd, and JLECmd updated!<sup>102</sup>

## Community Resources

- Prefetch Forensics<sup>103</sup>
- Forensic Investigation : Prefetch File<sup>104</sup>
- DFIR Playbook - Windows Forensics(WIP APR21)<sup>105</sup>

## Download PECmd

PECmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

## PECmd Sample Data

Sample Prefetch artifacts can be found at the [DFIRArtifactMuseum<sup>106</sup>](#).

<sup>99</sup><https://binaryforay.blogspot.com/2016/01/windows-prefetch-parser-in-c.html>

<sup>100</sup><https://binaryforay.blogspot.com/2016/01/introducing-pecmd.html>

<sup>101</sup><https://binaryforay.blogspot.com/2016/01/pecmd-v0600-released.html>

<sup>102</sup><https://binaryforay.blogspot.com/2016/03/pecmd-lecmd-and-jlecmd-updated.html>

<sup>103</sup><https://or10nlabs.tech/prefetch-forensics/>

<sup>104</sup><https://www.hackingarticles.in/forensic-investigation-prefetch-file/>

<sup>105</sup>[https://angry-bender.github.io/blog/WIP\\_Windows\\_Artifacts/#prefetch-files](https://angry-bender.github.io/blog/WIP_Windows_Artifacts/#prefetch-files)

<sup>106</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/Prefetch>

# **RBCmd**

## **RBCmd Introduction**

RBCmd is a tool created by Eric Zimmerman used to parse the SUM database. The SUM database is located at C:\Windows\System32\LogFiles\SUM where multiple \*.mdb files can be located.

## **RBCmd Use Cases**

### **Law Enforcement**

For those in Law Enforcement, this tool is useful for parsing Recycle Bin files which can provide indications of what files were deleted when and by which user. This can be crucial for crimes involving contraband multimedia files as key evidence.

### **Private Sector**

For those in the Private Sector, this tool is useful for parsing Recycle Bin files which can provide indications of what files a threat actor deleted before ending a period of unauthorized access. Often, there will be indicators of threat actor tool output having been deleted by a compromised user account within the Recycle Bin. This can help build out the timeline of events during a period of unauthorized access

## RBCmd Switches

In a PowerShell window, running .\RBCmd.exe will provide the following options when running RBCmd:

```
-d <d>          Directory to recursively process. Either this or -f is required
-f <f>          File to process. Either this or -d is required
-q              Only show the filename being processed vs all output. Useful to speed up
               exporting to json and/or csv
--csv <csv>    Directory to save CSV formatted results to. Be sure to include the\full path in
               double quotes
--csvf <csvf>  File name to save CSV formatted results to. When present, override\& default name
--dt <dt>        The custom date/time format to use when displaying time stamps. See
               https://goo.gl/CNVq0k for options. Default is: yyyy-MM-dd HH:mm:ss\[default:
               yyyy-MM-dd HH:mm:ss]
--debug         Show debug information during processing [default: False]
--trace         Show trace information during processing [default: False]
--version       Show version information
-?, -h, --help  Show help and usage information
```

## Switch Descriptions

Thankfully, RBCmd doesn't have any unique switches that aren't already covered in the common switches chapter.

## RBCmd Command Examples

### Example RBCmd Commands

#### Parse a legacy INFO2 Recycle Bin artifact and output to the console window

```
RBCmd.exe -f "C:\Temp\INFO2"
```

#### Parse a \$I file and output to CSV to a specified location

```
RBCmd.exe -f "C:\Temp\$I3VPA17" --csv "D:\csvOutput"
```

### **Parse a directory of files and output to CSV to a specified location**

```
RBCmd.exe -d "C:\Temp" --csv "c:\temp"
```

## RBCmd Output

### Analyzing RBCmd Output - Console

When running RBCmd with no specified output, you will see an entry similar to the one below for each Recycle Bin artifact parsed:

```
1 Source file: C:\temp\$Recycle.Bin\S-1-5-21-868105201-2749328792-1957117261-1001\$IZU\  
2 DHPA.md  
3  
4 Version: 2 (Windows 10/11)  
5 File size: 304 (304B)  
6 File name: C:\Users\CFUser\Documents\GitHub\DFIRArtifactMuseum\iOS\Address Book\iOS \  
7 14.1 - Hickman\README.md  
8 Deleted on: 2022-05-13 16:29:11
```

### Analyzing RBCmd Output - CSV

When running RBCmd with CSV output, a CSV will be output with similar data that's output to the console.

## RBCmd Key Takeaways

### Important Data Points

The Deleted On timestamp will be useful in determining when a file was deleted by a specific user. The Recycle Bin metadata files can be tracked to a user by their SID. That way, one can attribute which user deleted the file a given metadata file is pointing to.

## RBCmd References

### Associated GitHub Repositories

- <https://github.com/EricZimmerman/RBCmd> is the GitHub repository for RBCmd

### Blog Posts

#### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- [Quick Post: Notes on the Win10 Recycle Bin<sup>107</sup>](https://thinkdfir.com/2018/11/23/quick-post-notes-on-the-win10-recycle-bin/)

### Download RBCmd

RBCmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

### Recycle Bin Sample Data

Sample Recycle Bin artifacts can be found at the [DFIRArtifactMuseum<sup>108</sup>](#).

---

<sup>107</sup><https://thinkdfir.com/2018/11/23/quick-post-notes-on-the-win10-recycle-bin/>

<sup>108</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/%24Recycle.Bin>

# **RecentFileCacheParser**

## **RecentFileCacheParser Introduction**

RecentFileCacheParser is a tool created by Eric Zimmerman used to parse the RecentFileCache. The RecentFileCache can be located at C:\Windows\AppCompat\Programs\RecentFileCache.bcf.

## RecentFileCacheParser Switches

In a PowerShell window, running .\RecentFileCacheParser.exe will provide the following options when running RecentFileCacheParser:

```
-f <f>          File to process. Required
--csv <csv>      Directory to save CSV formatted results to. Be sure to include the\
full path in
                  double quotes
--csvf <csvf>    File name to save CSV formatted results to. When present, override\
s default name
--json <json>     Directory to save json representation to. Use --pretty for a more \
human readable
                  layout
--pretty          When exporting to json, use a more human readable layout [default:\
False]
-q               Only show the filename being processed vs all output. Useful to sp\
eed up
                  exporting to json and/or csv [default: False]
--version         Show version information
-?, -h, --help    Show help and usage information
```

## Switch Descriptions

Thankfully, RecentFileCacheParser doesn't have any unique switches that aren't already covered in the common switches chapter.

## RecentFileCacheParser Command Examples

### Example RecentFileCacheParser Commands

#### Parse the RecentFileCache and output to CSV to a specified location

```
RecentFileCacheParser.exe -f "C:\Temp\RecentFileCache.bcf" --csv "c:\temp"
```

#### Parse the RecentFileCache and output to JSON to a specified location

```
RecentFileCacheParser.exe -f "C:\Temp\RecentFileCache.bcf" --json "D:\jsonOutput"
```

## RecentFileCacheParser Output

### Analyzing RecentFileCacheParser Output - Console

```
Source file: D:\DFIRArtifactMuseum\Windows\RecentFileCache\Win7\EricZimmerman\Recent\  
FileCache.bcf
```

```
Source created: 2022-07-03 02:06:55
```

```
Source modified: 2022-07-03 02:06:55
```

```
Source accessed: 2022-09-09 17:11:42
```

File names

```
c:\windows\system32\werfault.exe  
c:\program files\jetico\bcwipe\bcwipesvc.exe  
c:\program files\jetico\bcwipe\bcwipetm.exe  
c:\windows\system32\icacls.exe  
c:\windows\system32\systempropertiesprotection.exe  
c:\windows\bcuninstall.exe
```

### Analyzing RecentFileCacheParser Output - CSV

When executing RecentFileCacherParser with CSV output enabled, it will output a filename similar to the example below:

```
%timestamp%_RecentFileCacheParser_Output.csv
```

## Analyzing RecentFileCacheParser Output - JSON

When executing RecentFileCacheParser with JSON output enabled, it will produce output similar to the example below:

```
1  {
2      "SourceFile": "D:\\DFIRArtifactMuseum\\Windows\\RecentFileCache\\Win7\\EricZimmerma\\
3 n\\RecentFileCache.bcf",
4      "SourceCreated": "\\\Date(1656814015362)\\",
5      "SourceModified": "\\\Date(1656814015362)\\",
6      "SourceAccessed": "\\\Date(1662743665086)\\",
7      "FileNames": [
8          "c:\\windows\\system32\\werfault.exe",
9          "c:\\program files\\jetico\\bcwipe\\bcwipesvc.exe",
10         "c:\\program files\\jetico\\bcwipe\\bcwipetm.exe",
11         "c:\\windows\\system32\\icacls.exe",
12         "c:\\windows\\system32\\systempropertiesprotection.exe",
13         "c:\\windows\\bcuninstall.exe"
14     ]
15 }
```

## RecentFileCacheParser References

### Associated GitHub Repositories

- <https://github.com/EricZimmerman/RecentFileCacheParser> is the GitHub repository for RecentFileCacheParser

### Download RecentFileCacheParser

RecentFileCacheParser can be downloaded from <https://ericzimmerman.github.io/#!index.md>

### RecentFileCache Sample Data

Sample RecentFileCache artifacts can be found at the [DFIRArtifactMuseum<sup>109</sup>](#).

---

<sup>109</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/RecentFileCache>

# RECmd

## RECmd Introduction

RECmd is a tool created by Eric Zimmerman used to parse Registry hives. Registry hives are located at C:\Windows\System32\config and contain multiple hives of interest.

## RECmd Use Cases

### Law Enforcement

For those in Law Enforcement, this tool is useful for parsing Registry hives which can contain a wealth of artifacts that can be attributed to a specific user of interest. There are artifacts within the Windows Registry that can help prove evidence of program execution, evidence of file opening, evidence of user logon, and many more. The Registry is simply too valuable to ignore on almost any possible criminal investigation.

### Private Sector

For those in the Private Sector, this tool is useful for parsing Registry hives which can contain possible indicators of threat actor persistence. Additionally, the user-attributable artifacts listed above can be useful when targeting the activity of compromised user accounts.

## RECmd Batch Files

<https://github.com/EricZimmerman/RECmd/tree/master/BatchExamples>

### Creating RECmd Batch Files

If you are looking for guidance on how to create RECmd Batch Files, look no further than the following resources:

- [RECmd Batch Files Guide<sup>110</sup>](https://github.com/EricZimmerman/RECmd/blob/master/BatchExamples/!RECmdBatch.guide)
- [RECmd Batch Files Template<sup>111</sup>](https://github.com/EricZimmerman/RECmd/blob/master/BatchExamples/!RECmdBatch.template)

Additionally, please check out Andrew Rathbun's 2021 SANS DFIR Summit presentation on EZ Tools/KAPE: How to Contribute to and Benefit from Open Source Contributions. Click [here<sup>112</sup>](#) for a timestamped link to the section of the presentation that relates to RECmd Batch Files.

<sup>110</sup><https://github.com/EricZimmerman/RECmd/blob/master/BatchExamples/!RECmdBatch.guide>

<sup>111</sup><https://github.com/EricZimmerman/RECmd/blob/master/BatchExamples/!RECmdBatch.template>

<sup>112</sup><https://youtu.be/mlb1GQP3ciE?t=318>

## RECmd Switches

In a PowerShell window, running .\RECmd.exe will provide the following options when running RECmd:

```

-d <d>          Directory to look for hives (recursively). -f or -d is required
-f <f>          Hive to search. -f or -d is required
--kn <kn>        Display details for key name. Includes subkeys and values
--vn <vn>        Value name. Only this value will be dumped
--bn <bn>        Use settings from supplied file to find keys/values. See included sample
--csv <csv>      file for examples
is used
--csvf <csvf>   Directory to save CSV formatted results to. Required when -bn \
overrides default

--saveTo <saveTo> File name to save CSV formatted results to. When present, overwrites
a FILE
--json <json>    Saves --vn value data in binary form to file. Expects path to \
vn is specified
--jsonf <jsonf>  Export --kn to directory specified by --json. Ignored when --\
--details         When true, compress names for profile based hives.
--base64 <base64> Show more details when displaying results [default: False]
bytes)           Find Base64 encoded values with size >= Base64 (specified in \
--minSize <minSize> Find values with data size >= MinSize (specified in bytes)
--sa <sa>        Search for <string> in keys, values, data, and slack
--sk <sk>        Search for <string> in value record's key names
--sv <sv>        Search for <string> in value record's value names
--sd <sd>        Search for <string> in value record's value data
--ss <ss>        Search for <string> in value record's value slack
--literal        If true, --sd and --ss search value will not be interpreted as \
ASCII or
--nd            Unicode byte strings [default: False]
else]           If true, do not show data when using --sd or --ss [default: F\
--regex          expression [default: False]
regular
--dt <dt>        The custom date/time format to use when displaying time stamp\
s [default]:

```

```

        yyyy-MM-dd HH:mm:ss.fffffff]
--nl          When true, ignore transaction log files for dirty hives [defa\
ult: False]
--recover      If true, recover deleted keys/values [default: False]
--vss          Process all Volume Shadow Copies that exist on drive specific\
d by -f or -d
                [default: False]
--dedupe       Deduplicate -f or -d & VSCs based on SHA-1. First file found \
wins [default:
                False]
--sync          If true, the latest batch files from
                https://github.com/EricZimmerman/RECmd/tree/master/BatchExamp\
les are
                downloaded and local files updated [default: False]
--debug         Show debug information during processing [default: False]
--trace         Show trace information during processing [default: False]
--version       Show version information
-?, -h, --help Show help and usage information

```

## Switch Descriptions

### --kn

This switch informs the tool to dump the full details of a provided key within the Registry.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry\Win10\APTSimulatorVM"  
--kn ROOT\ControlSet001\Control\BackupRestore\FilesNotToBackup

### --vn

This switch informs the tool to dump the full details of the value name specified. Please note, one of the following switches is required when using --vn: --sk, --sv, --sd, --ss, --kn, --Base64, --MinSize, and --bn.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry\Win10\APTSimulatorVM"  
--kn "ROOT\Microsoft\Windows\CurrentVersion\Authentication\LogonUI" --vn  
LastLoggedOnSAMUser

### --bn

This switch informs the tool to use a specified batch file when parsing registry hives.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry\Win10\APTSimulatorVM"  
--bn BatchExamples\Kroll\_Batch.reb --csv C:\temp

Please note, the Kroll Batch file is the most updated and actively maintained RECmd Batch file.

**--saveTo**

This switch informs the tool to save the binary data from a specified value name to a specified location.

```
Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry\Win10\APTSimulatorVM"  
--kn      "ROOT\Microsoft\Windows\CurrentVersion\Authentication\LogonUI"      --vn  
LastLoggedOnSAMUser --saveTo c:\temp\output
```

Please note, output is a binary file that can be opened in a text editor to view the contents.

**--details**

This switch informs the tool to display more details when displaying results.

```
Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry" --sd VMware
```

**--base64**

This switch informs the tool to search for Base64 values that are larger than the amount of bytes specified.

```
Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry\Win11\RathbunVM"  
--base64 10
```

The above command provides results similar to the following:

- 1 Found 2 search hits with size greater or equal to 10 bytes in D:\DFIRArtifactMuseum\\
- 2 Windows\Registry\Win11\RathbunVM\DEFAULT (NTUSER)\NTUSER.DAT
- 3 Key: AppEvents\EventLabels\Notification.Default, Value: (default), Size: 26
- 4 Key: Control Panel\PowerCfg\PowerPolicies\2, Value: Name, Size: 26

**--minSize**

This switch informs the tool to search for values with a minimum size in bytes.

```
Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry\Win11\RathbunVM"  
--minSize 500
```

The above command provides results similar to the following:

```

1 Found 18 search hits with size greater or equal to 500 bytes in D:\DFIRArtifactMuseum\
2 m\Windows\Registry\Win11\RathbunVM\SAM
3 Key: SAM\Domains\Account\Users\000001F4, Value: V, Size: 688
4 Key: SAM\Domains\Account\Users\000001F5, Value: V, Size: 664
5 Key: SAM\Domains\Account\Users\000001F7, Value: V, Size: 680
6 Key: SAM\Domains\Account\Users\000001F8, Value: V, Size: 836
7 Key: SAM\Domains\Account\Users\000001F8, Value: SupplementalCredentials, Size: 1,168
8 Key: SAM\Domains\Account\Users\000003E9, Value: V, Size: 580
9 Key: SAM\Domains\Builtin\Aliases\00000221, Value: C, Size: 560
10 Key: SAM\Domains\Builtin\Aliases\00000222, Value: C, Size: 600
11 Key: SAM\Domains\Builtin\Aliases\00000223, Value: C, Size: 520
12 Key: SAM\Domains\Builtin\Aliases\00000227, Value: C, Size: 504
13 Key: SAM\Domains\Builtin\Aliases\0000022C, Value: C, Size: 564
14 Key: SAM\Domains\Builtin\Aliases\0000022F, Value: C, Size: 664
15 Key: SAM\Domains\Builtin\Aliases\00000232, Value: C, Size: 504
16 Key: SAM\Domains\Builtin\Aliases\00000242, Value: C, Size: 504
17 Key: SAM\Domains\Builtin\Aliases\00000243, Value: C, Size: 584
18 Key: SAM\Domains\Builtin\Aliases\00000244, Value: C, Size: 744
19 Key: SAM\Domains\Account\Users\000003E8, Value: V, Size: 632 (Deleted: True)
20 Key: SAM\Domains\Account\Users\000003E8, Value: SupplementalCredentials, Size: 1,552\
21 (Deleted: True)

```

**-sa'**

This switch informs the tool to search for a specified string within keys, values, data, and slack.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry" --sa VMware

**--sk**

This switch informs the tool to search for a specified string within a value record's key names.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry" --sk VMware

The above command provides results similar to the following:

```

1 Found 2 search hits in D:\DFIRArtifactMuseum\Windows\Registry\Win11\RathbunVM\NTUSER\
2 .DAT
3     Key: Software\VMware, Inc.
4     Key: Software\VMware, Inc.\VMware Tools

```

**--sv**

This switch informs the tool to search for a specified string within a record's value names.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry" --sv VMware

**--sd**

This switch informs the tool to search for a specified string within a value record's value data.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry" --sd VMware

The above command provides results similar to the following:

```
1 Found 5 search hits in D:\DFIRArtifactMuseum\Windows\Registry\Win11\RathbunVM\SYSTEM
2           Key: ControlSet001\Control\DeviceContainers\{00000000-0000-0000-FFFF-FFFFFFFFFF}
3 FFFFFF}\BaseContainers\{00000000-0000-0000-FFFF-FFFFFFFFFF}, Value: SCSI\CdRom&Ven_\
4 NECVMWar&Prod_Vmware_SATA_CD01\5&260e6d66&0&010000
5           Key: ControlSet001\Control\DeviceContainers\{8a1a5449-86cd-11ec-ae32-806e6f6\
6 e6963}\BaseContainers\{8a1a5449-86cd-11ec-ae32-806e6f6e6963}, Value: SCSI\Disk&Ven_N\
7 VMe&Prod_Vmware_Virtual_N\5&25a13950&0&000000
8           Key: ControlSet001\Services\bam\State\UserSettings\S-1-5-21-4063860464-15085\
9 73791-632671957-1001, Value: \Device\HarddiskVolume3\Program Files\VMware\VMware Too\
10 ls\vmtoolsd.exe
11           Key: DriverDatabase\DriverPackages\vmci.inf_amd64_5e38a278d114b813\Strings, \
12 Value: loc.vmwaremanufacturer
13           Key: DriverDatabase\DriverPackages\vmci.inf_amd64_5e38a278d114b813\Strings, \
14 Value: loc.vmwarebusdevicedesc
```

**--ss**

This switch informs the tool to search for a specified string within a value record's slack.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry" --ss VMware

The above command will provide no results.

**--literal**

This switch informs the tool to not interpret the specified string as ASCII or Unicode byte strings.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry" --sd VMware --literal

**--nd**

This switch informs the tool to not show data when using --sd or --ss.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry" --sd VMware --nd

**--regex**

This switch informs the tool to treat the specified string as a regular expression for the following switches: --sk, --sv, --sd, and --ss.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry" --sd "\b(?:25[0-5]|2[0-4][0-9]|1[0-9][0-9]|1[1-9]?[0-9])\.\){3}(?:25[0-5]|2[0-4][0-9]|1[0-9][0-9]|1[1-9])" --regex

The above regular expression is the IPv4 regular expression taken from [here](#)<sup>113</sup>.

**--nl**

This switch informs the tool to ignore replaying transaction logs.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry" --sd VMware --nl

Please note, using this switch will provide the following message: Registry hive is dirty and transaction logs were found in the same directory, but --nl was provided. Data may be missing! Continuing anyways...

**--recover**

This switch informs the tool to attempt to recover deleted keys/values.

Example: .\RECmd.exe -d "D:\DFIRArtifactMuseum\Windows\Registry" --sd VMware --recover

**'-sync'**

This switch will inform the tool to download all RECmd Batch Files from [GitHub](#)<sup>114</sup> and update the local Batch Files stored in .\RECmd\BatchExamples.

Example: ".\RECmd.exe -sync"

<sup>113</sup><https://github.com/AndrewRathbun/DFIRRegex>

<sup>114</sup><https://github.com/EricZimmerman/RECmd/tree/master/BatchExamples>

## RECmd Command Examples

### Example RECmd Commands

**Parse a specified file for URL within the key names without recovering deleted keys/values and without replaying transaction logs**

```
RECmd.exe --f "C:\Temp\UsrClass_1.dat" --sk URL --recover false --nl
```

**Parse a specified file for specified regular expression within the value names**

```
RECmd.exe --f "D:\temp\UsrClass_1.dat" --regex --sv "(App|Display)Name"
```

## RECmd Output

### Analyzing RECmd Output - CSV

Using RECmd in batch mode is the ideal way to parse the Registry when CSV output is desired. 99% of the Windows Registry is not forensically interesting to examiners, so Batch files serve as a filter that tells RECmd to only parse what is specified within the Batch file, and ignore the rest. This allows for all the quick win artifacts in the Registry to be output to a single CSV output. Generally speaking, the Description and Category columns are going to be very important when trying to make sense of the CSV output.

An artifact will be given an accurate Description, which can be similar to UserAssist or TypedURLs, for example. Next, an artifact will be given a Category, so UserAssist's Category would be Program Execution and TypedURLs would be User Activity, as an example. This allows for examiners to filter on either the specific artifact by Description or filter by general category of artifact, such as Program Execution, Web Browsers, System Info, and many more.

The Kroll Batch file has received the most care and curation from the DFIR community and should be considered the de facto option when choosing a batch file for RECmd.

### Analyzing RECmd Output - JSON

Dumping Registry hives to JSON can be helpful as a supplement to CSV output. Dumping from the ROOT key to JSON will allow for the entire contents of the Registry hives parsed to be grepped through with a program like PowerGREP.

Example: .\RECmd.exe -f C:\temp\NTUSER.dat --kn ROOT --nl false --json C:\temp\output --jsonf NTUSER\_ROOT.json

The above command will dump from the ROOT (aka topmost) key in the hive, regardless of what it's called, to JSON.

# RECmd References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/RECmd> is the GitHub repository for RECmd

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- Introducing RECmd!<sup>115</sup>
- RECmd 0.6.0.0 released!<sup>116</sup>
- RECmd 0.6.1.0 released!<sup>117</sup>
- Reintroducing Registry Explorer and RECmd!<sup>118</sup>
- Registry Explorer/RECmd 0.7.1.0 released!<sup>119</sup>
- Registry values starting with a NULL character<sup>120</sup>
- Updates to the left of me, updates to the right of me, version 1 releases are here (for the most part)<sup>121</sup>
- Everything gets an update, Sept 2018 edition<sup>122</sup>
- Registry Explorer and RECmd 1.2.0.0 released!<sup>123</sup>
- Locked file support added to AmcacheParser, AppCompatCacheParser, MFTECmd, ShellBags Explorer (and SBECmd), and Registry Explorer (and RECmd)<sup>124</sup>

### Community Resources

- AboutDFIR - Registry Explorer/RECmd<sup>125</sup>

## Download RECmd

RECmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

<sup>115</sup><https://binaryforay.blogspot.com/2015/05/introducing-recmd.html>

<sup>116</sup><https://binaryforay.blogspot.com/2015/05/recmd-0600-released.html>

<sup>117</sup><https://binaryforay.blogspot.com/2015/06/recmd-0610-released.html>

<sup>118</sup><https://binaryforay.blogspot.com/2015/07/reintroducing-registry-explorer-and.html>

<sup>119</sup><https://binaryforay.blogspot.com/2015/07/registry-explorerrecmd-0710-released.html>

<sup>120</sup><https://binaryforay.blogspot.com/2016/01/registry-values-starting-with-null.html>

<sup>121</sup><https://binaryforay.blogspot.com/2018/03/updates-to-left-of-me-updates-to-right.html>

<sup>122</sup><https://binaryforay.blogspot.com/2018/09/everything-gets-update-sept-2018-edition.html>

<sup>123</sup><https://binaryforay.blogspot.com/2019/01/registry-explorer-and-recmd-1200.html>

<sup>124</sup><https://binaryforay.blogspot.com/2019/01/locked-file-support-added-to.html>

<sup>125</sup><https://aboutdfir.com/toolsandartifacts/windows/registry-explorer-recmd/>

## Registry Sample Data

Sample Registry artifacts can be found at the [DFIRArtifactMuseum](#)<sup>126</sup>. More can also be found [here](#)<sup>127</sup> for nearly every version of Windows.

---

<sup>126</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/Registry>

<sup>127</sup><https://github.com/AndrewRathbun/VanillaWindowsRegistryHives>

# **RLA**

## **RLA Introduction**

RLA is a tool that can be used to repair dirty Windows Registry hives. This is particularly useful when using tools that don't handle dirty Registry hives themselves.

## **RLA Use Cases**

This tool is useful for replaying Registry transaction logs which will ensure you're parsing all possible data within a given Registry hive.

## RLA Switches

In a PowerShell window, running .\RLA.exe will provide the following options when running RLA:

```

-f <f>           Hive to process. -f or -d is required
-d <d>           Directory to look for hives (recursively). -f or -d is required
--out <out>      Directory to save updated hives to. Only dirty hives with logs app\
lied will end
                  up in --out directory
--ca              When true, always copy hives to --out directory, even if they aren\
't dirty.
                  [default: True]
--cn              When true, compress names for profile based hives. [default: True]
--debug           Show debug information during processing [default: False]
--trace           Show trace information during processing [default: False]
--version         Show version information
-?, -h, --help   Show help and usage information

```

## Switch Descriptions

### --out

This switch will output updated hives (aka dirty -> clean) to the specified directory. This will only occur if there are dirty hives with transaction logs applied.

Example: .\rla.exe -d "D:\TODO" --out "D:\TODO\clean"

### --ca

This switch will tell RLA.exe to always copy hives to the specified directory, even if they aren't dirty.

Example: .\rla.exe -d "D:\Hives" --out "D:\Hives\clean" --ca

### --cn

This switch will compress the names for profile-based hives.

Example: .\rla.exe -d "D:\Hives\NTUser" --out "D:\Hives\clean" --cn

## RLA Command Examples

### Example RLA Commands

**Replay transaction logs from a specific Registry hive and output the clean Registry hive to a specified location**

```
rla.exe -f "C:\Temp\UsrClass 1.dat" --out C:\temp
```

**Replay transaction logs from a directory or Registry hives and output the clean Registry hives to a specified location**

```
rla.exe -d "D:\temp\" --out c:\temp
```

## RLA References

### Download RLA

RLA can be downloaded from <https://ericzimmerman.github.io/#!index.md>

# **SBECmd**

## **SBECmd Introduction**

SBECmd is a tool created by Eric Zimmerman used to parse the NTUSER.dat and UsrClass.dat Registry hives. These hives contains shell items that are recorded by Windows which indicate which folders a user has traversed.

## **SBECmd Use Cases**

### **Law Enforcement**

For those in Law Enforcement, this tool is useful for parsing the NTUser.dat and UsrClass.dat user Registry hives which will contain artifacts of folder traversal. Since the NTUser.dat and UsrClass.dat Registry hives exist for each user, one can attribute the folder traversal artifacts to a specific account. For Law Enforcement, these artifacts may provide pointers to folders or ZIP files that no longer exist. This artifact will provide the first and last time the specific user interacted with a specific folder or ZIP file, in most cases.

### **Private Sector**

For those in the Private Sector, this tool is useful for enumerating what a user of interest did during unauthorized access to a given host. Often, artifacts during periods of unauthorized access will show the threat actor accessing and viewing files and folders that are highly sensitive to the client's business.

## SBECmd Switches

In a PowerShell window, running .\SBECmd.exe will provide the following options when running SBECmd:

```

-d <d>          Directory to look for registry hives. This or -l is required
-l              Process live registry (Requires Administrator rights). This or -d \
is required
                [default: False]
--csv <csv>    Directory to save CSV formatted results to. This or --json require\
d unless --de
                or --body is specified
--csvf <csvf>  File name to save CSV formatted results to. When present, override\
s default name
--dedupe       When true, SBECmd processes all hives in -d <directory> and remove\
s duplicates.
                See manual for details [default: False]
--dt <dt>       The custom date/time format to use when displaying time stamps. See
                https://goo.gl/CNVq0k for options [default: yyyy-MM-dd HH:mm:ss]
--nl           When true, ignore transaction log files for dirty hives [default: \
False]
--debug        Show debug information during processing [default: False]
--trace         Show trace information during processing [default: False]
--version       Show version information
-?, -h, --help  Show help and usage information

```

## Switch Descriptions

### -l

This switch will inform SBECmd to process the live user Registry hives.

Example: .\SBECmd.exe -l --csv D:\temp\sbecmd

The above command will output a filename similar to this: YYYYMMDD\_HHMMSS\_hostname\_LIVE\_-  
REGISTRY.csv

### -nl

This switch will inform the tool whether to replay transaction logs or not.

Below is an example of replaying transaction logs:

Example: .\SBECmd.exe -d "D:\temp\hives" --nl false --csv "D:\temp"

When processing transaction logs, you will see a message similar to this:

```
Two transaction logs found. Determining primary log...
Primary log: D:\temp\hives\SYSTEM.LOG2, secondary log: D:\temp\hives\SYSTEM.LOG1
Replaying log file: D:\temp\hives\SYSTEM.LOG2
Replaying log file: D:\temp\hives\SYSTEM.LOG1
At least one transaction log was applied. Sequence numbers have been updated to 0x91\
DD. New Checksum: 0xAD729723
Found 1,024 cache entries for Windows10Creators in ControlSet001
```

Below is an example of ignoring transaction logs:

Example: .\SBECmd.exe -d "D:\temp\hives" --nl true --csv "D:\temp"

When ignoring transaction logs, you will see a message similar to this:

```
Registry hive is dirty and transaction logs were found in the same directory, but --\
nl was provided. Data may be missing! Continuing anyways...
Sequence numbers do not match! Hive is dirty and the transaction logs should be revi\
ewed for relevant data!
Found 1,024 cache entries for Windows10Creators in ControlSet001
```

## SBECmd Command Examples

### Example SBECmd Commands

#### Parse offline user Registry hives and output to CSV to a specified location

```
SBECmd.exe -d c:\temp\hives --csv c:\temp\sbeout
```

#### Parse offline user Registry hives and output to CSV to a specified location with timestamps adjusted to a specified timezone

```
SBECmd.exe -d c:\temp\hives --csv c:\temp\sbeout --tz "US Eastern Standard Time"
```

```
#### Parse offline user Registry hives and output to CSV to a specified location with deduplicated output
```

```
SBECmd.exe -d c:\temp\hives --csv c:\temp\sbeout --dedupe
```

## SBECmd Output

### Analyzing SBECmd Output - CSV

When executing SBECmd with CSV output enabled, it will output filenames similar to the examples below:

```
Username_NTUSER.csv  
Username_usrClass.csv
```

If the system you're analyzing has had 100 users who have traversed directories at some point in time, you should see 200 CSVs for your output!

## SBECmd Key Takeaways

### Important Data Points

#### **FirstInteracted and LastInteracted**

The FirstInteracted timestamp for a shell item will provide you with the timestamp of the first time a given user had traversed a directory within Windows. The LastInteracted timestamp for a shell item will provide you with the last time a given user had traversed a directory within Windows. There are no timestamps for in between those two bookends within this artifact.

# SBECmd References

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- ShellBags Explorer v0.5.2.0 released!<sup>128</sup>
- ShellBags Explorer 0.5.4.0 released!<sup>129</sup>
- A few updates<sup>130</sup>
- ShellBags Explorer v0.8.0.0 released!<sup>131</sup>
- ShellBags Explorer 0.9.5.0 released!<sup>132</sup>
- A fluery of updates!<sup>133</sup>
- Locked file support added to AmcacheParser, AppCompatCacheParser, MFTECmd, ShellBags Explorer (and SBECmd), and Registry Explorer (and RECmd)<sup>134</sup>

## Download SBECmd

SBECmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

## Shellbags Sample Data

Sample Shellbags (NTUser.dat and UsrClass.dat) artifacts can be found at the [DFIRArtifactMuseum](#)<sup>135</sup>.

---

<sup>128</sup><https://binaryforay.blogspot.com/2015/02/shellbags-explorer-v0520-released.html>

<sup>129</sup><https://binaryforay.blogspot.com/2015/02/shellbags-explorer-0540-released.html>

<sup>130</sup><https://binaryforay.blogspot.com/2015/08/a-few-updates.html>

<sup>131</sup><https://binaryforay.blogspot.com/2017/01/shellbags-explorer-v0800-released.html>

<sup>132</sup><https://binaryforay.blogspot.com/2017/09/shellbags-explorer-0950-released.html>

<sup>133</sup><https://binaryforay.blogspot.com/2018/05/a-fluery-of-updates.html>

<sup>134</sup><https://binaryforay.blogspot.com/2019/01/locked-file-support-added-to.html>

<sup>135</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/Registry>

# SQLECmd

## SQLECmd Introduction

SQLECmd is a tool created by Eric Zimmerman used to parse SQLite databases. SQLite databases will only be parsed if a Map exists for that specific SQLite database.

## SQLECmd Use Cases

### Law Enforcement

For those in Law Enforcement, this tool is useful for parsing SQLite databases which often store common web browser artifacts. SQLECmd can provide as a validation tool to many commonly used forensic suites that parse browsing history, as well. The SQLite queries are available on GitHub and can be modified by the community if ever the database schema evolves enough to where current SQLite queries do not work any longer. Additionally, this tool can be used to parse mobile artifacts which can serve as a validation tool for common mobile forensic suites.

### Private Sector

For those in the Private Sector, this tool is useful for parsing SQLite databases which often store common web browser artifacts.

## SQLECmd Maps

<https://github.com/EricZimmerman/SQLECmd/tree/master/SQLMap/Maps>

### Creating SQLECmd Maps

If you are looking for guidance on how to create SQLECmd Maps, look no further than the following resources:

- [SQLECmd Maps Guide<sup>136</sup>](https://github.com/EricZimmerman/SQLECmd/blob/master/SQLMap/Maps/!OS_Application_OptionalDescription.guide)
- [SQLECmd Maps Template<sup>137</sup>](https://github.com/EricZimmerman/SQLECmd/blob/master/SQLMap/Maps/!OS_Application_OptionalDescription.template)

Additionally, please check out Andrew Rathbun's 2021 SANS DFIR Summit presentation on EZ Tools/KAPE: How to Contribute to and Benefit from Open Source Contributions. Click [here<sup>138</sup>](#) for a timestamped link to the section of the presentation that relates to SQLECmd Maps.

<sup>136</sup>[https://github.com/EricZimmerman/SQLECmd/blob/master/SQLMap/Maps/!OS\\_Application\\_OptionalDescription.guide](https://github.com/EricZimmerman/SQLECmd/blob/master/SQLMap/Maps/!OS_Application_OptionalDescription.guide)

<sup>137</sup>[https://github.com/EricZimmerman/SQLECmd/blob/master/SQLMap/Maps/!OS\\_Application\\_OptionalDescription.template](https://github.com/EricZimmerman/SQLECmd/blob/master/SQLMap/Maps/!OS_Application_OptionalDescription.template)

<sup>138</sup><https://youtu.be/mlb1GQP3ciE?t=682>

## SQLECmd Switches

In a PowerShell window, running .\SQLECmd.exe will provide the following options when running SQLECmd:

```
-f <f>          File to process. This or -d is required
-d <d>          Directory to process that contains SQLite files. This or -f is req\uired
--csv <csv>    Directory to save CSV formatted results to
--json <json>   Directory to save JSON formatted results to
--dedupe        Deduplicate -f or -d files based on SHA-1. First file found wins [\default: True]
--hunt          When true, all files are looked at regardless of name and file header is used to identify SQLite files, else filename in map is used to find databases [default: False]
--maps <maps>  The path where event maps are located. Defaults to 'Maps' folder w\here program was executed [default: C:\Users\CFUser\OneDrive - Kroll\Desktop\EZ Tools\net6\SQLECmd\Maps]
--sync          If true, the latest maps from https://github.com/EricZimmerman/SQLECmd/tree/master/SQLMap/Maps a\re downloaded and local maps updated [default: False]
--debug         Show debug information during processing [default: False]
--trace         Show trace information during processing [default: False]
--version       Show version information
-?, -h, --help  Show help and usage information
```

## Switch Descriptions

### '-hunt'

This switch informs the tool to hunt for SQLite databases. This is useful due to SQLite databases often having inconsistent file extensions (or sometimes none at all). This switch will search for the file header for SQLite databases and inform examiners which files are SQLite databases. Please note, only those databases that have Maps made for them will be parsed.

Example: .\SQLECmd.exe -d "D:\DFIRArtifactMuseum\Windows" --csv C:\temp --hunt

SQLECmd will display the following message when a SQLite file is not found:

```
D:\DFIRArtifactMuseum\Windows\SRUM\Win10\RathbunVM\Clean\SRU00004.log is not a SQLite  
file! Skipping...
```

SQLECmd will display the following message when a SQLite file is found:

```
Processing D:\DFIRArtifactMuseum\Windows\WindowsTimeline\Win10\APTSimulatorVM\ActivitiesCache.db...
```

### **'-maps'**

This switch will inform the tool to look for SQLECmd Maps at a specified location other than .\SQLECmd\Maps.

Example: .\SQLECmd.exe -d "D:\sql" --csv "D:\sql" --maps "D:\Maps"

### **'-sync'**

This switch will inform the tool to download all SQLECmd Maps from [GitHub](#)<sup>139</sup> and update the local Maps stored in .\SQLECmd\Maps.

Example: .\SQLECmd.exe --sync

---

<sup>139</sup><https://github.com/EricZimmerman/SQLECmd/tree/master/SQLMap/Maps>

## SQLECmd Command Examples

### Example SQLECmd Commands

#### **Parse someFile.db and output to CSV to a specified location**

```
SQLECmd.exe -f "C:\Temp\someFile.db" --csv "c:\temp\out"
```

#### **Parse SQLite databases located within C:\Temp and output to CSV to a specified location**

```
SQLECmd.exe -d "C:\Temp\" --csv "c:\temp\out"
```

#### **Hunt for SQLite Databases recursively in a specified directory and output CSVs of any mapped databases to a specified location**

```
SQLECmd.exe -d "C:\Temp\" --hunt --csv "c:\temp\out"
```

The --hunt command will inform you as to which files are SQLite databases! This is very helpful when searching for new SQLite databases to research.

## SQLCmd References

### Associated GitHub Repositories

- <https://github.com/EricZimmerman/SQLCmd> is the GitHub repository for SQLCmd

### Download SQLCmd

SQLCmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

### SQLite Sample Data

Sample SQLite databases can be found at the [DFIRArtifactMuseum<sup>140</sup>](https://github.com/AndrewRathbun/DFIRArtifactMuseum) in the Android and Windows directories.

---

<sup>140</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum>

# **SrumECmd**

## **SrumECmd Introduction**

SrumECmd is a tool created by Eric Zimmerman used to parse the SRUM database. The SRUM database is located at C:\Windows\System32\sru\SRUDB.dat and serves as the backend for Windows Task Manager.

## **SrumECmd Use Cases**

### **Law Enforcement**

For those in Law Enforcement, this tool is useful for parsing the SRUM database which can provide another source of program execution for media players, photo viewers, etc. Additionally, being able to see Bytes Read and Bytes Written by various programs may help provide insight as to the size of files certain applications were handling. This may be important relating to crimes involving contraband multimedia files. Additionally, for P2P cases, Bytes Sent and Bytes Received artifacts can prove to be crucial datapoints for the purpose of the investigation.

### **Private Sector**

For those in the Private Sector, this tool is useful for parsing the SRUM database which can provide another source of program execution for potentially malicious executables. Additionally, Bytes Sent and Bytes Received can sometimes be the only indicator of data exfiltration in the instance of a ransomware case. If a case is known to involve data exfiltration, the SRUM database should be a mandatory artifact to parse and analyze so long as the suspected data exfiltration occurred within the 30 days of when the SRUM database was parsed.

## SrumECmd Switches

In a PowerShell window, running .\SrumECmd.exe will provide the following options when running SrumECmd:

```
-f <f>           SRUDB.dat file to parse
-r <r>           SOFTWARE hive to process. This is optional, but recommended
-d <d>           Directory to recursively process, looking for SRUDB.dat an\
d SOFTWARE
                  hive. This mode is primarily used with KAPE so both SRUDB.\\
dat and
                  SOFTWARE hive can be located
--csv <csv> (REQUIRED) Directory to save CSV formatted results to. Be sure to inc\
lude the full
                  path in double quotes
--dt <dt>         The custom date/time format to use when displaying time st\
amps. See
                  https://goo.gl/CNVq0k for options
                  [default: yyyy-MM-dd HH:mm:ss]
--debug          Show debug information during processing [default: False]
--trace          Show trace information during processing [default: False]
--version        Show version information
-?, -h, --help   Show help and usage information
```

## Switch Descriptions

### -r

This switch informs the tool to parse a SOFTWARE Registry hive at a specified location. This is beneficial because applications referenced in the SRUM database can be resolved using data stored within the SOFTWARE Registry hive for better results.

Example: .\SrumECmd.exe -d "C:\temp\SRUMdb" -r "C:\temp\SOFTWAREhive" --csv "C:\temp\SRUMoutput"

Please note, that using the -d switch against a directory where a SRUM database and a SOFTWARE Registry hive resides within the subdirectories, SrumECmd will find both without needing to utilize the -r switch to specifically point to the SOFTWARE Registry hive.

Example: .\SrumECmd.exe -d "C:\temp\KapeTriage\tout\C" --csv "C:\temp\SRUMoutput"

## SrumECmd Command Examples

### Example SrumECmd Commands

**Parse a SRUM database at a specified location and a SOFTWARE Registry hive at a specified location and output to CSV to a specified location**

```
SrumECmd.exe -f "C:\Temp\SRUDB.dat" -r "C:\Temp\SOFTWARE" --csv "C:\Temp\"
```

**Parse a SRUM database and output to CSV to a specified location**

```
SrumECmd.exe -f "C:\Temp\SRUDB.dat" --csv "c:\temp"
```

**Process a specified location looking for a SRUM database and/or SOFTWARE Registry hive to parse and output to CSV to a specified location**

```
SrumECmd.exe -d "C:\Temp" --csv "c:\temp"
```

## SrumECmd Output

### Analyzing SrumECmd Output - CSV

SrumECmd will produce the following CSVs:

- %timestamp%\_SrumECmd\_AppResourceUseInfo\_Output.csv
- %timestamp%\_SrumECmd\_EnergyUsage\_Output.csv
- %timestamp%\_SrumECmd\_NetworkConnections\_Output.csv
- %timestamp%\_SrumECmd\_NetworkUsages\_Output.csv
- %timestamp%\_SrumECmd\_PushNotifications\_Output.csv
- %timestamp%\_SrumECmd\_Unknown312\_Output.csv
- %timestamp%\_SrumECmd\_UnknownD8F\_Output.csv

%timestamp% will look something similar to this: 20220904032628

**%timestamp%\_SrumECmd\_AppResourceUseInfo\_Output.csv**

This output can be useful for seeing which applications were running at a given time.

**%timestamp%\_SrumECmd\_NetworkUsages\_Output.csv**

This output can be useful for seeing which applications were sending and receiving data. This is helpful in Incident Response engagements where data exfiltration is an important part of the mission for examiners.

## SrumECmd Sample Data

Sample SRUM database artifacts can be found at the [DFIRArtifactMuseum<sup>141</sup>](#).

---

<sup>141</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/SRUM>

# SrumECmd References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/Srum> is the GitHub repository for SrumECmd

## Blog Posts

### Community Resources

- [AboutDFIR - App Timeline Provider – SRUM Database<sup>142</sup>](#)

## Download SrumECmd

SrumECmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

## SRUM Sample Data

Sample SRUM artifacts can be found at the [DFIRArtifactMuseum<sup>143</sup>](#).

---

<sup>142</sup><https://aboutdfir.com/app-timeline-provider-srum-database/>

<sup>143</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/SRUM>

# **SumECmd**

## **SumECmd Introduction**

SumECmd is a tool created by Eric Zimmerman used to parse the SUM database. The SUM database is located at C:\Windows\System32\LogFiles\SUM where multiple \*.mdb files can be located.

## **SumECmd Use Cases**

### **Law Enforcement**

For those in Law Enforcement, SumECmd may not have an immediate use unless the system(s) being analyzed is attached to a domain. The SUM database can provide authentication history within a Domain, IP resolution within the Domain, and the first and last time an account has been authenticated within a Domain.

### **Private Sector**

For those in the Private Sector, SumECmd can provide helpful information regarding which compromised accounts have authenticated where and when within a Domain. Additionally, having visibility into the DNS history of a Domain for the current year and previous 2 years is incredibly helpful when trying to figure out which IP address resolved to which host when.

## SumECmd Switches

In a PowerShell window, running .\SumECmd.exe will provide the following options when running SumECmd:

```
-d <d>          Directory to process, looking for SystemIdentity.mdb, Current.mdb, \
etc.                         Required.
--csv <csv>      Directory to save CSV formatted results to. Be sure to include the\
full                         path in double quotes
--wd             Generate CSV with day level details. Default is TRUE [default: Tru\
e]
--dt <dt>        The custom date/time format to use when displaying time stamps. See
                  https://goo.gl/CNVq0k for options [default: yyyy-MM-dd HH:mm:ss]
--debug          Show debug information during processing [default: False]
--trace          Show trace information during processing [default: False]
--version        Show version information
-?, -h, --help   Show help and usage information
```

## Switch Descriptions

### --wd

This switch will generate CSVs with day level details. This means an extra CSV will be generated named `TIMESTAMP_SumECmd_DETAIL_ClientsDetailed_Output.csv`. When running SumECmd with `--wd false`, this CSV will not generate.

Example: .\SumECmd.exe -d "D:\Test\SUM" --csv "D:\temp" --wd

## SumECmd Command Examples

### Example SumECmd Commands

#### Parse the SUM database and output CSVs to a specified location

```
SumECmd.exe -d "C:\Temp\sum" --csv "C:\Temp\"
```

## SumECmd Output

### Analyzing SumECmd Output - CSV

SumECmd will output the following CSVs when ran against a Domain Controller's SUM database with the ADDS (Active Directory Domain Services) role assigned to it:

```
%timestamp%_SumECmd_DETAIL_Clients_Output.csv  
%timestamp%_SumECmd_DETAIL_ClientsDetailed_Output.csv  
%timestamp%_SumECmd_DETAIL_DnsInfo_Output.csv  
%timestamp%_SumECmd_DETAIL_RoleAccesses_Output.csv  
%timestamp%_SumECmd_SUMMARY_ChainedDbInfo_Output.csv  
%timestamp%_SumECmd_SUMMARY_RoleInfos_Output.csv  
%timestamp%_SumECmd_SUMMARY_SystemIdentInfo_Output.csv
```

If a Domain Controller does NOT have the ADDS role assigned to it, you will still see the above output, but you likely will not see the DnsInfo output.

#### `%timestamp%_SumECmd_DETAIL_Clients_Output.csv`

This output is useful for helping examiners observe which accounts authenticated where within a given domain. Insert Date and Last Access can be interpreted as the first and last time an account authenticated to the IP Address listed in the column of the same name.

#### `%timestamp%_SumECmd_DETAIL_ClientsDetailed_Output.csv`

This output will provide a different look at the most of the same info from the output mentioned above. Between this output and the above output, examiners can glean good insight as to which accounts authenticated where, when, and how many times along with an idea of when the first and last authentication occurred.

#### `%timestamp%_SumECmd_DETAIL_DnsInfo_Output.csv`

This output is incredibly useful for providing the examiner with historical DNS information within the domain. If an examiner needs to know which host a certain IP address resolves to within a domain, first locate a Domain Controller that has the ADDS role assigned (often trial and error) and locate this table's output!

As a protip, in Timeline Explorer, sort ascending on the hostname and then secondary sort (hold down shift while clicking on the next column you want to sort on) on Last Seen. This will give you a great overview of which hosts had which IP addresses when.

#### `%timestamp%_SumECmd_DETAIL_RoleAccesses_Output.csv`

This output will provide examiners with an idea of which roles were assigned to the Windows Server endpoints within a given domain along with their Role GUIDs.

```
%timestamp%_SumECmd_SUMMARY_ChainedDbInfo_Output.csv
```

This output will provide examiners with an idea of which calendar years are covered within the SUM database for a Domain Controller within a given domain. Please note, that the second a Domain Controller connects to a domain for the first time, it will only have information from that point forward. It will not inherit historical information within the domain prior to the time of host coming online within the domain.

## SumECmd References

### Associated GitHub Repositories

- <https://github.com/EricZimmerman/Sum> is the GitHub repository for SumECmd

### Blog Posts

### Community Resources

- A new type of User access log<sup>144</sup>
- Windows User Access Logs (UAL)<sup>145</sup>

### Download SumECmd

SumECmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

### SUM Sample Data

Sample SUM artifacts can be found at the [DFIRArtifactMuseum](#)<sup>146</sup>.

---

<sup>144</sup><https://advisory.kpmg.us/blog/2021/digital-forensics-incident-response.html>

<sup>145</sup><https://svchost.medium.com/windows-user-access-logs-ual-9580f1100635>

<sup>146</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/SUM>

# **VSCMount**

## **VSCMount Introduction**

VSCMount is a tool created by Eric Zimmerman used to mount Volume Shadow Copies (VSCs). Volume Shadow Copies can be used to uncover artifacts that no longer reside on the computer.

## **VSCMount Use Cases**

### **Law Enforcement**

For those in Law Enforcement, VSCs may provide access to files that no longer currently exist on the forensic image being analyzed. Taking advantage of VSCs may provide more evidence for an investigation.

### **Private Sector**

For those in the Private Sector, VSCs may provide more visibility into historical artifacts within event logs, Registry hives, and NTFS metadata files. These are common sources for quick wins that are often subject to data rolling over, so the more visibility, the better.

## VSCMount Switches

In a PowerShell window, running .\VSCMount.exe will provide the following options when running VSCMount:

```
--dl <d1>           Source drive to look for Volume Shadow Copies (C:, D:, or F:\ for e\  
example)  
--mp <mp>           The base directory where you want VSCs mapped to  
--ud                Use VSC creation timestamps (yyyyMMddTHHmmss.fffffff) in symbolic \  
link                names [default: True]  
--debug              Show debug information during processing [default: False]  
--version            Show version information  
-?, -h, --help       Show help and usage information
```

## Switch Descriptions

### --d1

This required switch will allow for the examiner to specify which drive letter to search for Volume Shadow Copies to be mounted. In the below example, a folder will be created at the root of the C:\ drive labeled vssroot\_C and the following folders will be created, as an example:

- vss067-20220802T071544.8950680
- vss069-20220802T111643.1360760
- vss073-20220802T151751.7332050
- vss075-20220802T191857.4091500
- vss077-20220802T231954.8229610
- vss079-20220803T145849.8753330
- vss081-20220803T154303.5886250
- vss082-20220803T185949.1160530
- vss084-20220804T122438.3318770
- vss085-20220804T125916.5830320
- vss087-20220804T170016.7516520
- vss089-20220805T002628.8687860

Example: .\VSCMount.exe --d1 C --mp C:\vssroot

### --mp

This switch will allow for the examiner to specify the directory in which the Volume Shadow Copies found with the --d1 switch will be mounted to for easy access.

Example: .\VSCMount.exe --d1 C --mp C:\vssroot

### --ud

This switch will use the creation timestamps (yyyyMMddTHHmmss.ffffffff) in symbolic link names. Given that the default is true, if you run the below command, you will find the following output compared to the exmaple listed above for the --d1 switch:

- vss067
- vss069
- vss073
- vss075
- vss077
- vss079
- vss081

- vss082
- vss084
- vss085
- vss087
- vss089

Example: .\VSCMount.exe --dl C --mp C:\vssroot --ud false

## VSCMount Command Examples

### Example VSCMount Commands

**Mount VSCs from the C:\ drive and map them to a specified directory while using VSC creation timestamps in symbolic link names with debug messages enabled**

```
. \VSCMount.exe --dl C --mp C:\VssRoot --ud --debug
```

## VSCMount References

### Associated GitHub Repositories

- <https://github.com/EricZimmerman/VSCMount> is the GitHub repository for VSCMount

### Blog Posts

#### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- [Introducing VSCMount<sup>147</sup>](#)

### Download VSCMount

VSCMount can be downloaded from <https://ericzimmerman.github.io/#!index.md>

---

<sup>147</sup><https://binaryforay.blogspot.com/2018/09/introducing-vscmount.html>

# **WxTCmd**

## **WxTCmd Introduction**

WxTCmd is a tool created by Eric Zimmerman used to parse the Windows Timeline<sup>148</sup>. The Windows Timeline feature was added to Windows 10 with the 1804 (April 2018) feature update. The Windows Timeline can be found at the following location: C:\Users\%User%\AppData\Local\ConnectedDevicesPlatform\L.CFUser\ActivitiesCache.db.

## **WxTCmd Use Cases**

### **Law Enforcement**

For those in Law Enforcement, this tool is useful for parsing Windows Timeline artifacts which can provide program execution activity that can be attributed to a specific user account. This can be useful for providing attribution of activity for various crimes involving media players, photo viewers, etc. Establish which Windows account the suspect was using during the time of interest and see what this artifact can provide to you.

### **Private Sector**

For those in the Private Sector working ransomware cases, this tool is useful for parsing Windows Timeline artifacts which can provide program execution activity that can be attributed to a specific user account. For instance, an account compromised by a threat actor may have useful artifacts within the user of interest's Windows Timeline.

For insider threat cases, this artifact can provide useful program execution artifacts that can be attributed to a specific user of interest. Any applications used by the user during the timeframe of interest may be recorded here so long as the events occurred within 30 days of acquiring the Windows Timeline artifact.

---

<sup>148</sup><https://support.microsoft.com/en-us/windows/get-help-with-timeline-febc28db-034c-d2b0-3bbe-79aa0c501039>

## WxTCmd Switches

In a PowerShell window, running .\WxTCmd.exe will provide the following options when running WxTCmd:

```
-f <f>          File to process. Required
--csv <csv>      Directory to save CSV formatted results to. Be sure to include the \
full
                  path in double quotes
--dt <dt>        The custom date/time format to use when displaying timestamps. See
                  https://goo.gl/CNVq0k for options [default: yyyy-MM-dd HH:mm:ss]
--debug          Show debug information during processing [default: False]
--trace          Show trace information during processing [default: False]
--version        Show version information
-?, -h, --help   Show help and usage information
```

## Switch Descriptions

Thankfully, WxTCmd doesn't have any unique switches that aren't already covered in the common switches chapter.

## WxTCmd Command Examples

### Example WxTCmd Commands

#### Parse the Windows Timeline and Output to CSV to a Specified Location

```
WxTCmd.exe -f "C:\Users\eric\AppData\Local\ConnectedDevicesPlatform\L.eric\Activitie\
sCache.db" --csv c:\temp
```

## WxTCmd Output

### Analyzing WxTCmd Output - CSV

#### '%timestamp%\_Activity\_PackageIDs.csv'

This CSV will contain GUIDs of applications that were executed on disk. Generally speaking, this CSV is not going to be as useful as the CSV highlighted below.

**'%timestamp%\_Activity.csv**

This CSV will contain the Start Time and End Time of an application's execution by a given user. This database is user specific, so this activity can be associated with a specific user account. Additionally, there is a JSON payload for each entry within the Windows Timeline SQLite database that may provide further context for an entry within the database.

## **WxTCmd Key Takeaways**

### **Important Data Points**

#### **Timestamps**

WxTCmd will output the following timestamps of interest:

- \* StartTime - The timestamp of when the user opened an application
- \* EndTime - The timestamp of when the user closed an application

# WxTCmd References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/WxTCmd> is the GitHub repo for WxTCmd

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, [Binary Foray](https://binaryforay.blogspot.com/)<sup>149</sup>:

- [Introducing WxTCmd!](https://binaryforay.blogspot.com/2018/05/introducing-wxtcmd.html)<sup>150</sup>

### Community Resources

- Windows Timeline: Putting the what & when together<sup>151</sup>
- Windows 10 Timeline Feature<sup>152</sup>
- Some thoughts about Windows 10 “Timeline” forensics artifacts<sup>153</sup>
- Windows Artifacts for Forensics Investigation<sup>154</sup>

## Download WxTCmd

WxTCmd can be downloaded from <https://ericzimmerman.github.io/#!index.md>

## Windows Timeline Sample Data

Sample Windows Timeline artifacts can be found at the [DFIRArtifactMuseum](https://dfirartifactmuseum.com/)<sup>155</sup>.

---

<sup>149</sup><https://binaryforay.blogspot.com/>

<sup>150</sup><https://binaryforay.blogspot.com/2018/05/introducing-wxtcmd.html>

<sup>151</sup><https://niiconsulting.com/checkmate/2021/10/windows-timeline-putting-the-what-when-together/>

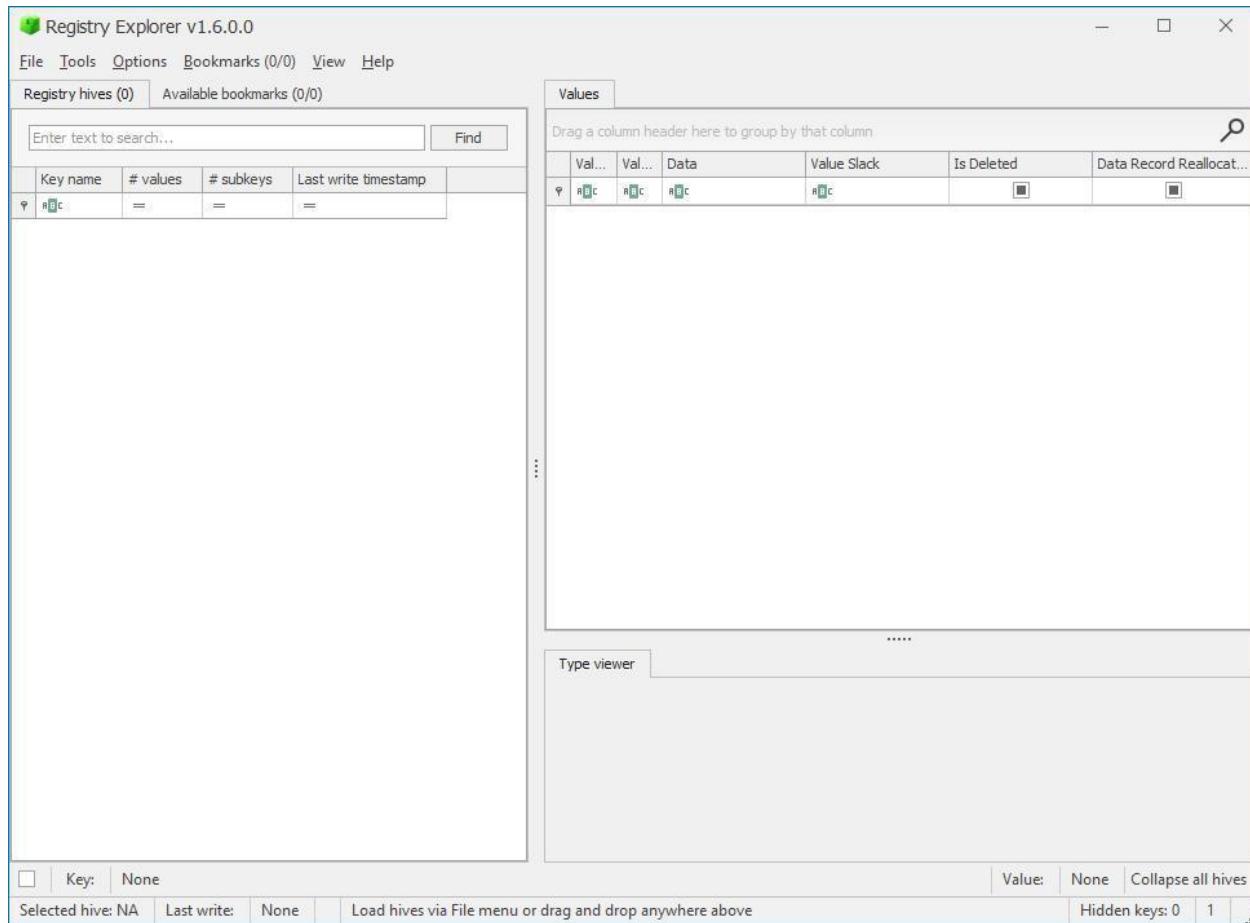
<sup>152</sup><https://www.datadigitally.com/2019/04/windows-10-timeline-feature.html>

<sup>153</sup><https://andreafortuna.org/2019/10/03/some-forensic-thoughts-about-windows-10-timeline/>

<sup>154</sup><https://tajdini.net/blog/forensics-and-security/windows-artifacts-for-forensics-investigation/>

<sup>155</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/WindowsTimeline>

# EZ Tools - GUI



# **EZViewer**

## **EZViewer Introduction**

EZViewer is a standalone, zero dependency viewer for .doc, .docx, .xls, .xlsx, .txt, .log, .rtf, .otd, .htm, .html, .mht, .csv, and .pdf. Any non-supported files are shown in a hex editor (with data interpreter!). Please note, this tool is read only and therefore cannot edit any files!

## **EZViewer Use Cases**

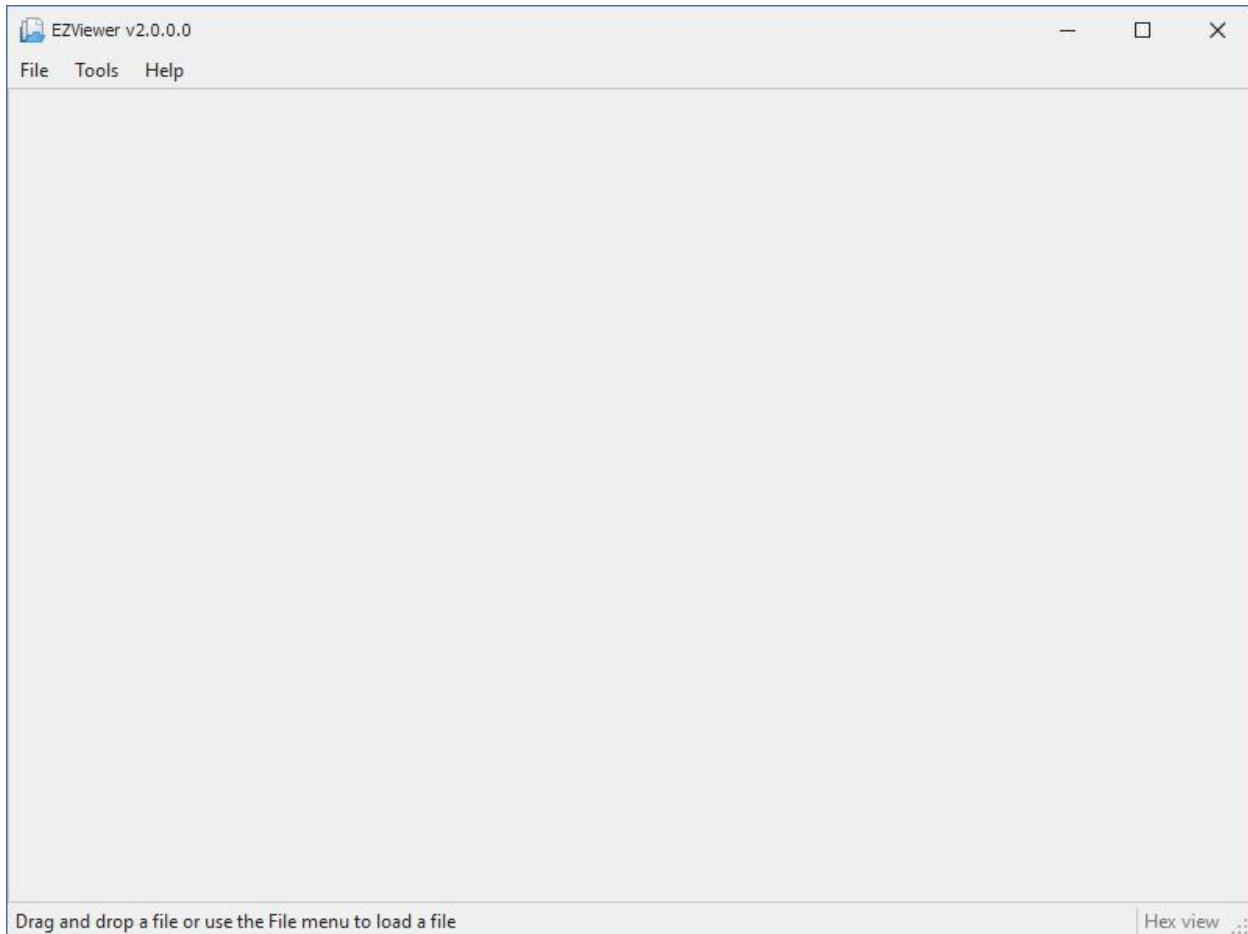
### **Law Enforcement**

For those in Law Enforcement, this tool is useful for bringing on site to a search warrant and not having to worry about having a certain application to open a certain file type.

### **Private Sector**

For those in the Private Sector, this tool is useful for bringing on site to a client engagement and not having to worry about having a certain application to open a certain file type.

## EZViewer Screenshot

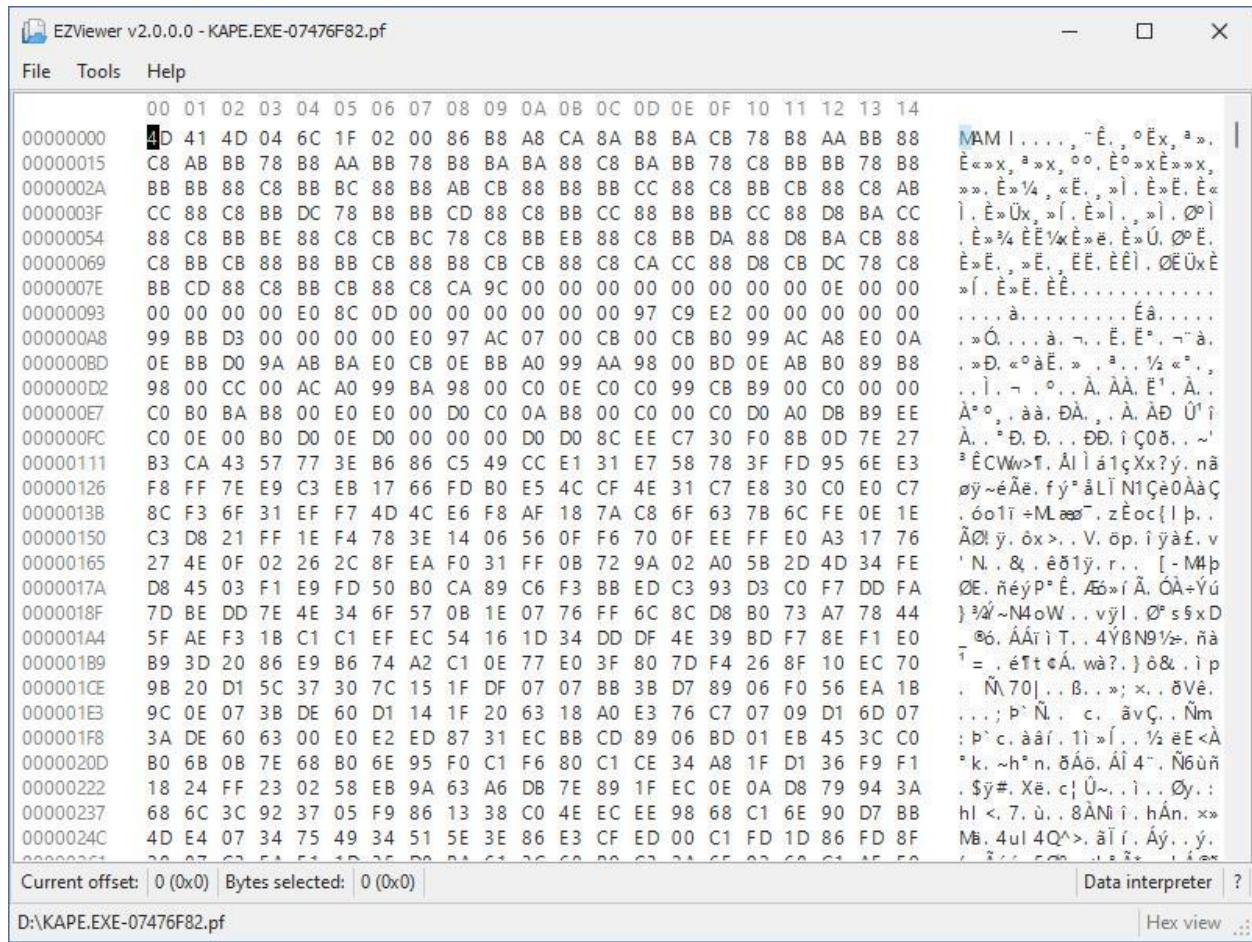


## EZViewer Key Takeaways

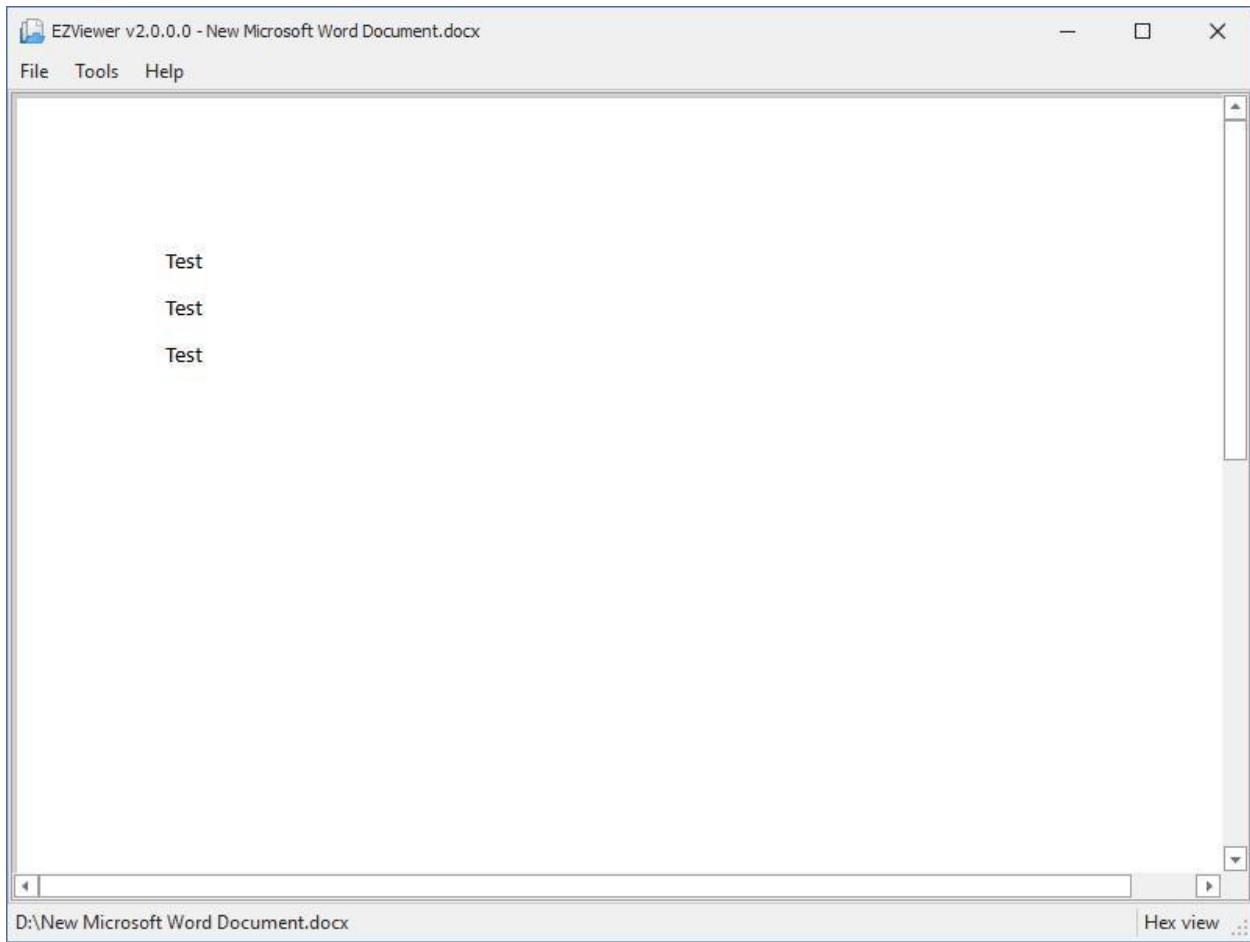
EZViewer is a great tool to have on your flash drive when you're on site at a search warrant or at a client's site. There is no need to deal with Microsoft Office. Just use EZViewer in all of its fully portable glory so you can view those files without any of the first-party dependencies!

Pro-tip: hold down control and scroll with your mouse wheel to zoom in and out.

As stated above, you can open any file in EZViewer! If it's not supported, it'll open in hex view, as seen below:



Supported file formats like Word (.docx) will appear as such:



## EZViewer References

### Download EZViewer

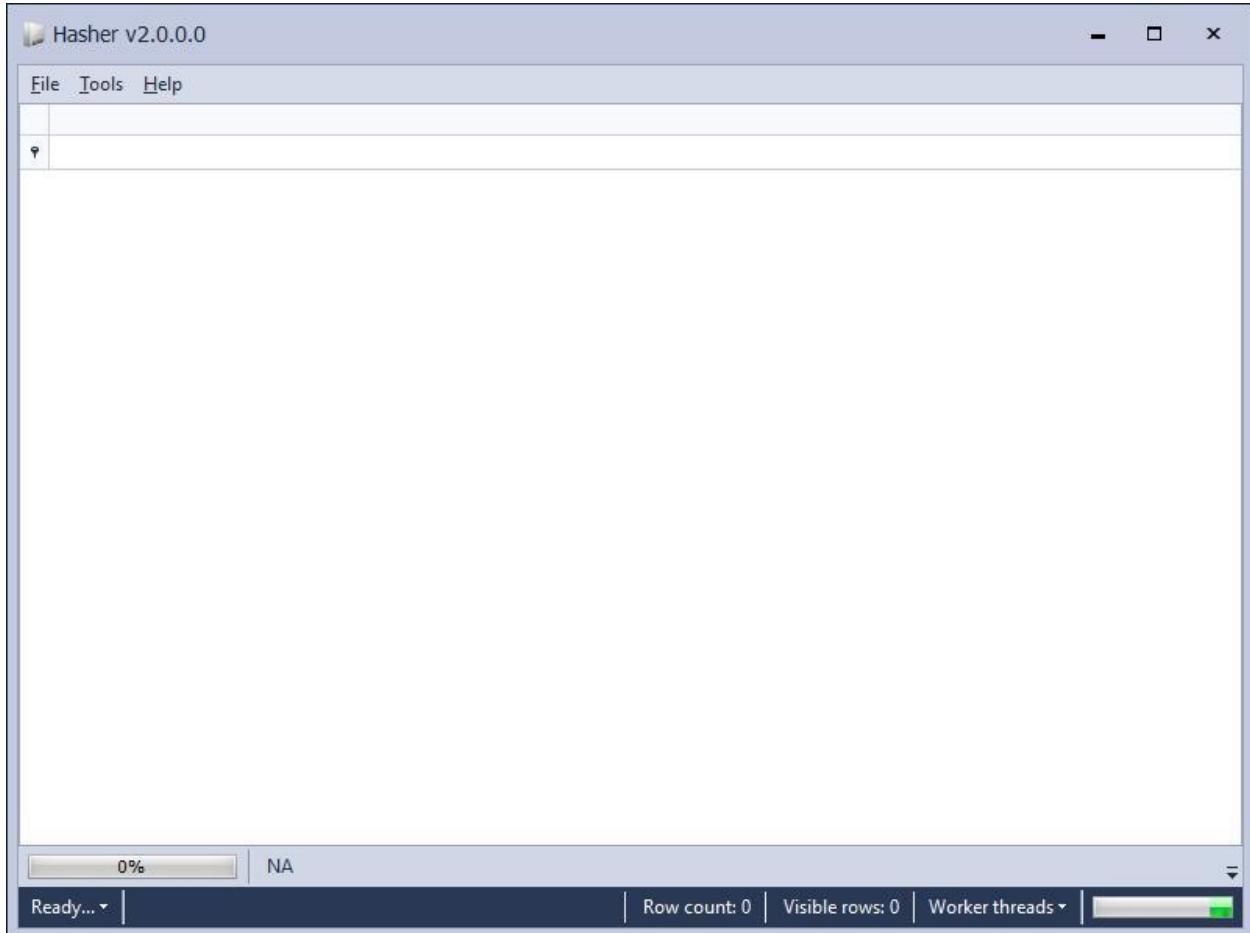
EZViewer can be downloaded from <https://ericzimmerman.github.io/#!index.md>

# **Hasher**

## **Hasher Introduction**

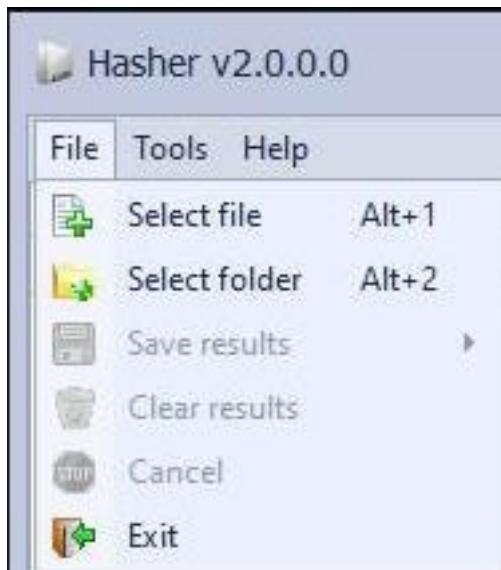
Hasher is a tool created by Eric Zimmerman used to generate hash values for files and/or folders.

## **Hasher Screenshot**



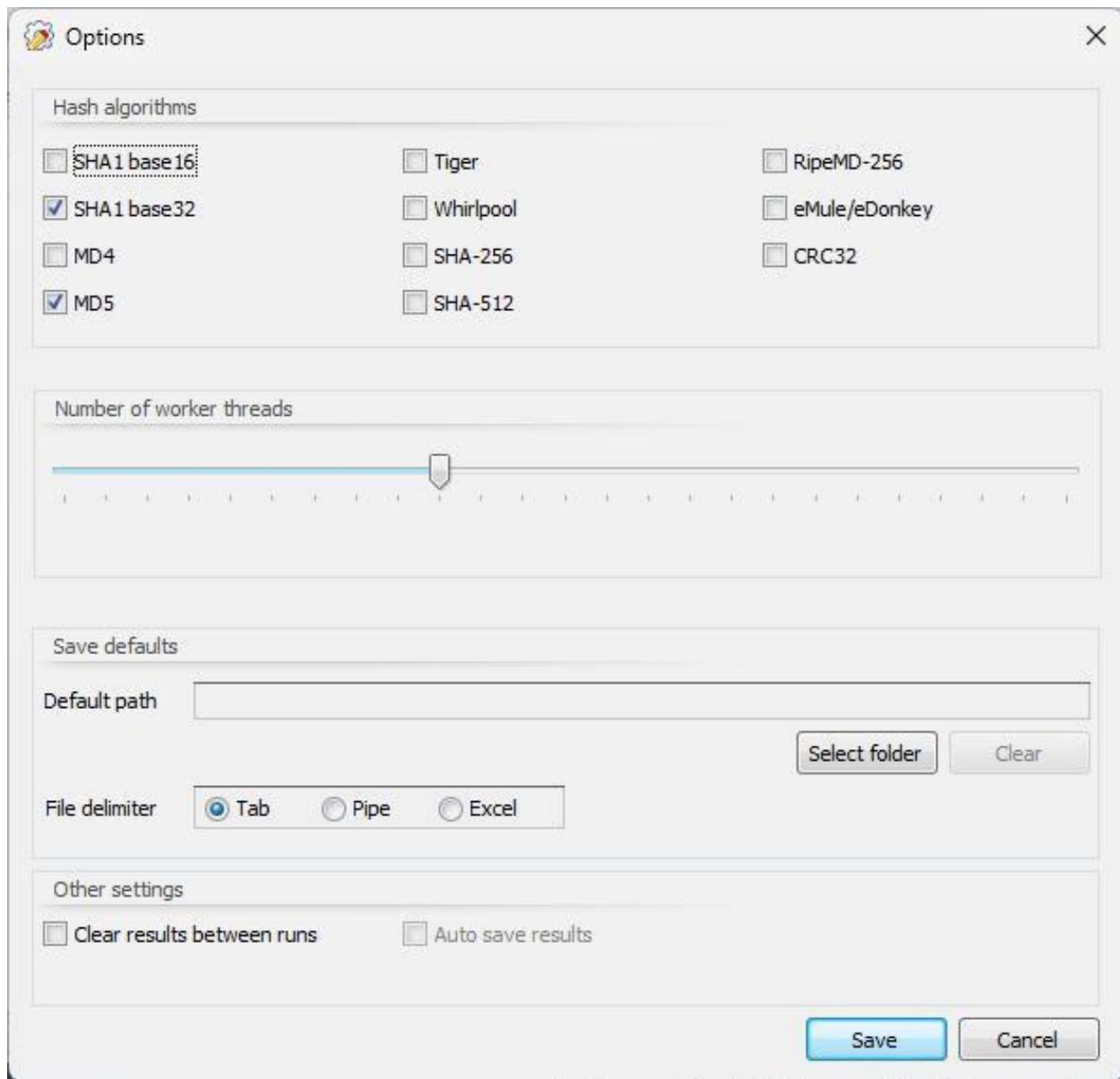
## Hasher Features

Hasher can generate hash values for a single file or a folder of files, as seen below.



## Hasher Options

Hasher provides examiners with multiple options in the Tools -> Options menu.



Within these options, examiners can choose which hash algorithms Hasher should calculate, how many worker threads, the default save path, default file delimiter, and other settings.

These settings will be saved within a Hasher.ini settings file.

## Hasher Settings

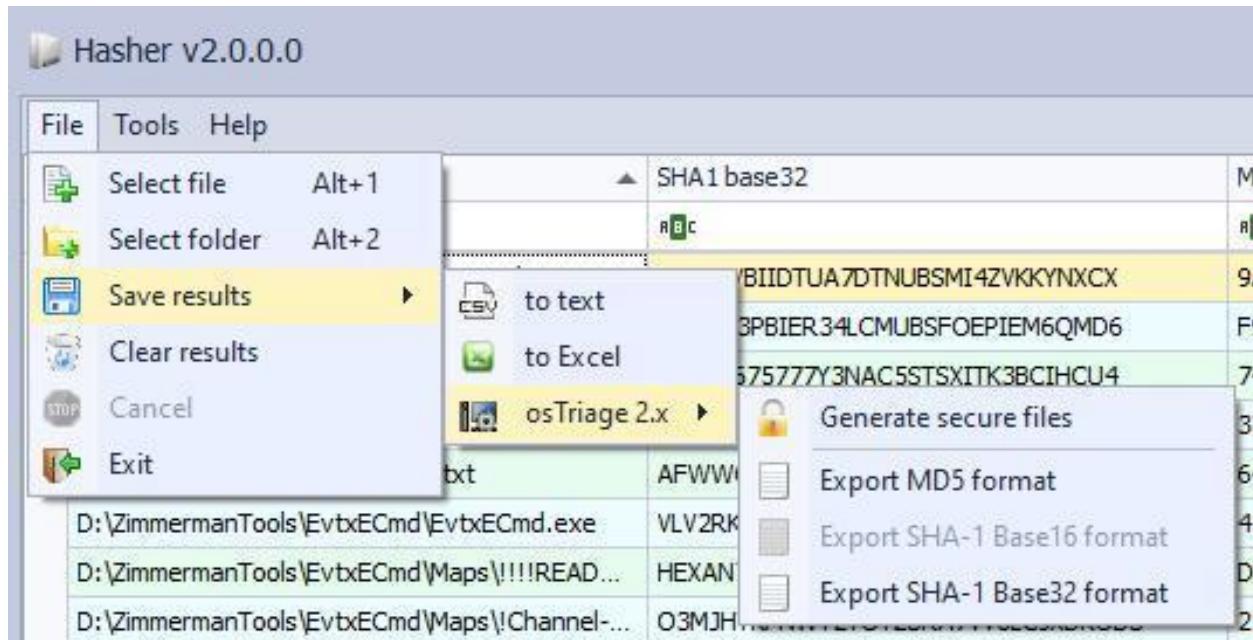
Below is an example of Hasher.ini.

```
1 [General]
2 DefaultSavePath =
3 AutoSaveResults = False
4 FileDelimiter = Tab
5 ClearResultsBetweenRuns = False
6 MAXThreads = 10
7 [Algorithms]
8 MD5 = True
9 SHA1b32 = True
10 SHA1b16 = False
11 MD4 = False
12 Tiger = False
13 Whirlpool = False
14 SHA-256 = False
15 SHA-512 = False
16 RipeMD-256 = False
17 eMule = False
18 CRC32 = False
19 [Theme]
20 Theme = Visual Studio 2013 Blue
```

File name	SHA1 base32	MD5
D:\ZimmermanTools\!!!RemoteFileDetails.csv	7QDXIVBIIDTUA7DTNUBSMI4Z...	9A6A3DC47E32A80E7873B312...
D:\ZimmermanTools\AmcacheParser.exe	E12X7P3PBIER34LCMUBSFOEP...	F9DA0978EC5D1597174C4296...
D:\ZimmermanTools\AppCompatCacheParser.exe	MKK2S675777Y3NAC5STSXITK...	74376B9320F992363A3B92B23...
D:\ZimmermanTools\pstrings.exe	XGNTRI5J2BJCRX7SQ04QNMD...	3767936B7666F4DCFA37D59A...
D:\ZimmermanTools\ChangeLog.txt	AFWWCJLBWMWME4VJFPL7R...	64D7FE4DCF8613A4C5F3651E...
D:\ZimmermanTools\EvtxECmd\EvtxECmd.exe	VLV2RK4KKAHWUAAV3PF33GB...	419C6B84D5935A4EB71DD1ED...
D:\ZimmermanTools\EvtxECmd\Maps\!!!README.md	HEXAN7UV7DWKEZR4WMEF4...	D29ECD4233EF8075C1AACD9...
D:\ZimmermanTools\EvtxECmd\Maps\Channel-Name_Provider-Name_EventID.g...	O3MJHYKPWNWY2TOTZSRA7VV...	21BEE82162393AB2E368BB139...
D:\ZimmermanTools\EvtxECmd\Maps\Channel-Name_Provider-Name_EventID.te...	E43REN5WEQXEKEP2YHQN4GI...	AC470778FBE051D5DAA6B53B...
D:\ZimmermanTools\EvtxECmd\Maps\adPWDManager_adPWDManager_110.map	J26ARZ2Q05TOLKF3WN5KUD...	C6FE1E273411BD4AB6ADFDD9...
D:\ZimmermanTools\EvtxECmd\Maps\Application_Application-Error_1000.map	NZOTLJWA2QVKYFLOCYTS7G6...	46753B086F1A65C077238DFE...
D:\ZimmermanTools\EvtxECmd\Maps\Application_Application-Hang_1002.map	H22NX5NI4WJDCIDOXG7HTLE...	355E78CA63B3FB8B551F34AA...
D:\ZimmermanTools\EvtxECmd\Maps\Application_CarbonBlackDefense_1.map	Y2Q63YLETGV7QPERYDFKHP2...	15899256658012A87A882A1C...
D:\ZimmermanTools\EvtxECmd\Maps\Application_CarbonBlackDefense_17.map	2ZMO7HFAHT4A4PB7WUTT55...	7435293E8AB3BD65EBA5D21E...
D:\ZimmermanTools\EvtxECmd\Maps\Application_CarbonBlackDefense_33.map	H2PP5DM4UIBOB7J4LGP3WVY...	8D0B8F09FD86B43AEFFA354D...
D:\ZimmermanTools\EvtxECmd\Maps\Application_CarbonBlackDefense_49.map	FD57UMKGQFQ2CVDEBJZA4Z4...	2EA592384423AF7AA28C1F06...
D:\ZimmermanTools\EvtxECmd\Maps\Application_Citrix-Desktop-Service_1027.map	GRCF3UBM22GZU2VHWXKXYT...	EED21D5B8E97F3FBB77A569E...
D:\ZimmermanTools\EvtxECmd\Maps\Application_Citrix-Desktop-Service_1049.map	6IV3ZQHYYTQRZOCFTWVK2P...	2E009418857411570B0999E75...
D:\ZimmermanTools\EvtxECmd\Maps\Application_CylanceSvc_1.map	UFRFIVFMY76IMNNW1J4YQ3Q4...	15046CD023A0FAF71B474E8F...
D:\ZimmermanTools\EvtxECmd\Maps\Application_CylanceSvc_2.map	6YORDBFZ6FA2ENYCALDIQE...	CC76CDB21A090763A0D601B2...
D:\ZimmermanTools\EvtxECmd\Maps\Application_HitmanPro-Alert_911.map	55KTRL7JMRNNX615OYOWRT...	B7E4343C098A284R1303C284...

## Exporting Hasher Results

Once files/folders have been hashed in Hasher, examiners can export the results to multiple formats.



Exporting to .txt will provide an output file named similarly to 2022-11-12-222212-HasherResults.txt.

Below is an example of Hasher output.

1	File name	SHA1 base32	MD5	
2	D:\ZimmermanTools\PECmd.exe		Z5AODWE3HRXUWLJMJBEMFVXGK7APOAQU	1AFED4AFCB86C8AC6BA2AA3\
3	C6160072A			

## Hasher References

### Quick Help

Below is the information located in Help -> Quick Help.

- 1 To use, select some files **and/or** folders **and** drag/drop them onto the main program window. Alternatively, use the File menu options to select files **or** a folder.
- 2
- 3
- 4 As files are hashed, you can sort, filter, search, etc. by interacting with the column headers: Left click to sort **and** right click **for** a context menu with many more options.
- 5
- 6
- 7
- 8 CTRL-C will copy the highlighted row to the clipboard. ALT-C will copy only the selected cell to the clipboard.
- 9
- 10

- 11 You can **export** the search via the File menu once hashing **is** complete.
- 12
- 13 You can set various options via the Options menu.

## Download Hasher

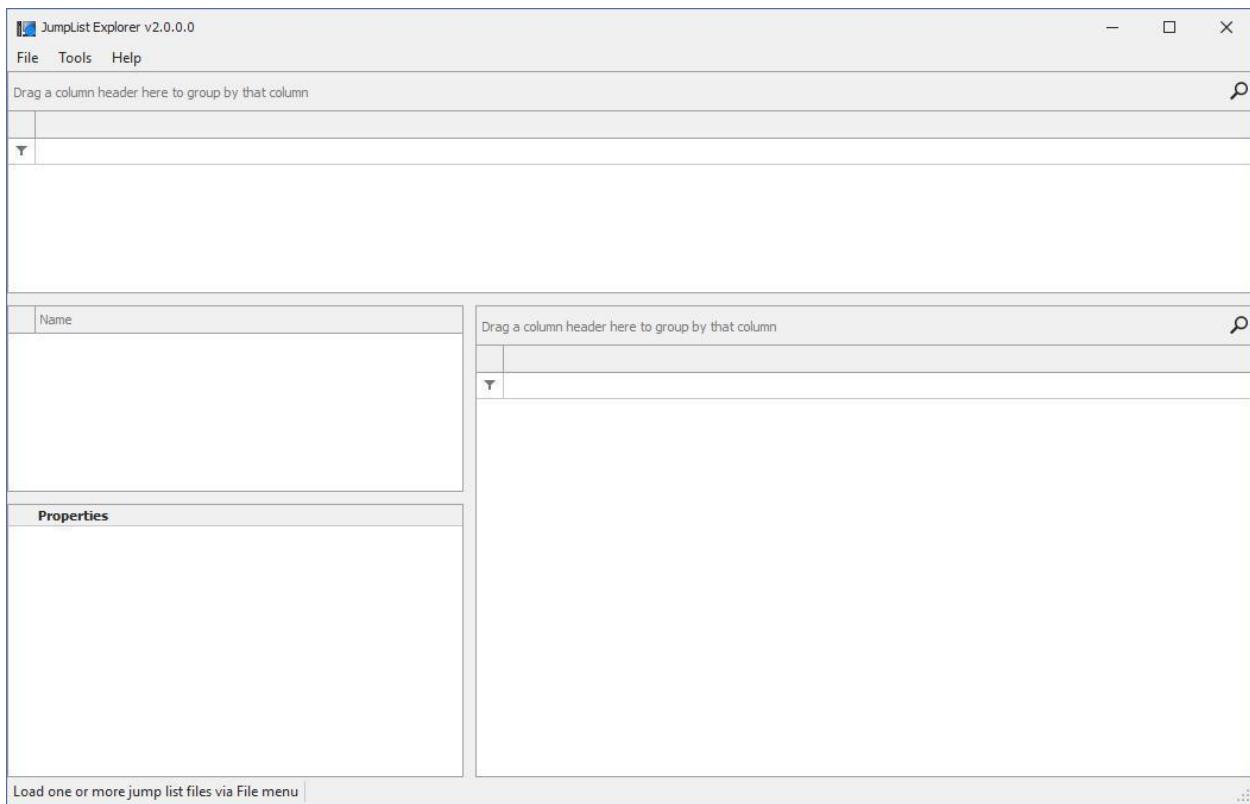
Hasher can be downloaded from <https://ericzimmerman.github.io/#!index.md>

# JumpList Explorer

## JumpList Explorer Introduction

JumpList Explorer is a tool created by Eric Zimmerman that can be used to visually parse JumpList files. JumpList files are [OLE containers<sup>156</sup>](#) like many other filetypes within the Windows operating system.

Below is a screenshot of JumpList Explorer without any artifacts loaded.



---

<sup>156</sup><https://learn.microsoft.com/en-us/cpp/mfc/ole-background?view=msvc-170>

Below is a screenshot of JumpList Explorer with all JumpLists ingested from .\DFIRArtifactMuseum\Windows\JumpLists\Win10\RathbunVM artifacts

The screenshot shows the JumpList Explorer interface with several data tables and a properties panel.

**Top Table:**

Source File Name	Jump List Type	App ID	App ID Description	Lnk File Count	File Size
T 5f7b5f1e01b83767	=	5f7b5f1e01b83767	Unknown AppId	8	16,896
D:\DFIRArtifactMuseum\Wind...	Automatic	9b9cdc69c1c24e2b	Unknown AppId	7	15,360
D:\DFIRArtifactMuseum\Wind...	Automatic	f01b4d95cf55d32a	Unknown AppId	9	12,288
D:\DFIRArtifactMuseum\Wind...	Custom	590aee7bdd69b59b	Unknown AppId	3	5,449
D:\DFIRArtifactMuseum\Wind...	Custom	ccba5a5986c77e43	Unknown AppId	2	3,888

**Second Table:**

Name	Entry Num...	Target Cre...	Target Mod...	Target Acc...	Absolute P...	Extra Block...	Interaction ...
5f7b5f1e01b83767.automaticDestinations-ms	9	2022-02-0...	2022-02-0...	2022-02-0...	My Comput...	2	1
	8	2022-02-0...	2022-02-0...	2022-02-0...	My Comput...	2	1
	7	2022-02-0...	2022-02-0...	2022-02-0...	My Comput...	2	1
	5	2022-02-0...	2022-02-0...	2022-02-0...	My Comput...	2	1
	4	2022-02-0...	2022-02-0...	2022-02-0...	My Comput...	2	1
	3	2022-02-0...	2022-02-0...	2022-02-0...	My Comput...	2	1
	1	2022-02-0...	2022-02-0...	2022-02-0...	My Comput...	2	2
	2	2022-02-0...	2022-02-0...	2022-02-0...	My Comput...	2	1

**Properties Panel:**

AppId	5f7b5f1e01b83767
AppId description	Unknown AppId
Pinned count	0
Entries count	8
Last used entry #	9
Version	4

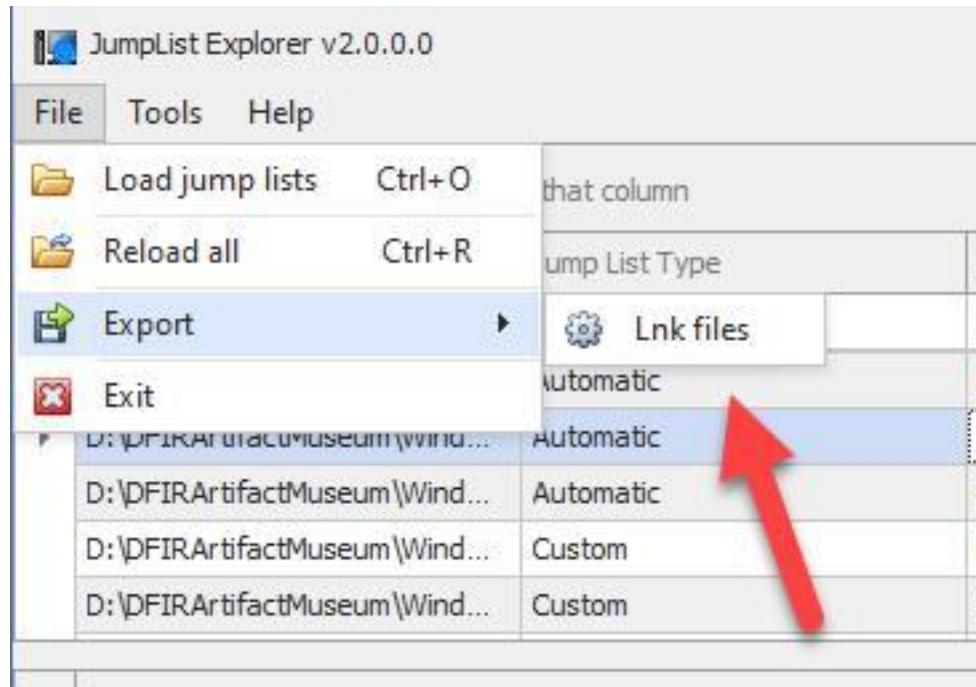
**Message Bar:**

Displaying jump list: 5f7b5f1e01b83767.automaticDestinations-ms

## JumpList Explorer Functionality

### Exporting LNK Files

JumpList files are simply a collection of LNK files associated with a given application. JumpList Explorer provides functionality for exporting out the LNK files associated with a given application via File -> Export -> Lnk Files, as seen below.



## Exploring Automatic Destinations

JumpList Explorer can parse AutomaticDestinations files, as seen below.

The screenshot shows the JumpList Explorer interface with the title bar "JumpList Explorer v2.0.0.0". The menu bar includes "File", "Tools", and "Help". The main window displays two panes. The left pane shows a table of automatic destination entries:

Source File Name	Jump List Type	App ID	App ID Description	Lnk File Count	File Size
D:\DFIRArtifactMuseum\Wind...	Automatic	5f7b5f1e01b83767	Unknown AppId	8	16,896
D:\DFIRArtifactMuseum\Wind...	Automatic	9b9cdc69c1c24e2b	Unknown AppId	7	15,360
D:\DFIRArtifactMuseum\Wind...	Automatic	f01b4d95cf55d32a	Unknown AppId	9	12,288
D:\DFIRArtifactMuseum\Wind...	Custom	590aee7bdd69b59b	Unknown AppId	3	5,449
D:\DFIRArtifactMuseum\Wind...	Custom	ccba5a5986c77e43	Unknown AppId	2	3,888

The right pane shows the detailed properties of the selected entry (Entry #: 0001 - My Computer\...):

Name	Value
TargetCreationDate	2022-02-05 19:32:54
TargetModificationDate	2022-02-05 19:33:07
TargetLastAccessedDate	2022-02-05 19:33:30
Header.DataFlags	HasTargetIdList, HasLinkInfo, IsUnicode, Disable...
Header.FileAttributes	FileAttributeArchive, FileAttributeCompressed
Header.FileSize	28,392
Header.IconIndex	0
Header.ShowWindow	SwNormal
Absolute path	My Computer\C:\Users\AndrewRathbun\AppData\Local\Microsoft\OneDrive\setup\logs\Update_2022-02-05_193254_1500-1878.log
LocalPath	C:\Users\AndrewRathbun\AppData\Local\Microsoft\OneDrive\setup\logs\Update_2022-02-05_193254_1500-1878.log
LocationFlags	VolumeIdAndLocalbasePath

At the bottom of the right pane, a status bar displays the message "Loaded Ink (entry #: 1) ==> My Computer\C:\Users\AndrewRathbun\AppData\Local\Microsoft\OneDrive\setup\logs\Update\_2022-02-05\_193254\_1500-1878.log".

## Exploring Custom Destinations

JumpList Explorer can parse CustomDestinations files, as seen below.

The screenshot shows the interface of JumpList Explorer version 2.0.0.0. The main window displays two tables of data. The top table lists 'Source File Name', 'Jump List Type', 'App ID', 'App ID Description', 'Lnk File Count', and 'File Size'. The bottom table shows 'Properties' for a selected item, with columns for 'Name' and 'Value'. A sidebar on the left shows a tree view of custom destinations, with the path '590aee7bdd69b59b.customDestinations-ms' expanded to show three entries: 'Lnk #: 000 - Shared Documents Folder (Users Files)\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Windows PowerShell\Windows PowerShell.lnk', 'Lnk #: 001 - My Computer\C:\Windows\System32\WindowsPowerShell\v1.0\powershell...', and 'Lnk #: 002 - My Computer\C:\Windows\System32\WindowsPowerShell\v1.0\powershell...'. The status bar at the bottom indicates 'Loaded Lnk ==> Shared Documents Folder (Users Files)\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Windows PowerShell\Windows PowerShell.lnk'.

Source File Name	Jump List Type	App ID	App ID Description	Lnk File Count	File Size
D:\DFIRArtifactMuseum\Wind...	Automatic	5f7b5f1e01b83767	Unknown AppId	8	16,896
D:\DFIRArtifactMuseum\Wind...	Automatic	9b9cdc69c1c24e2b	Unknown AppId	7	15,360
D:\DFIRArtifactMuseum\Wind...	Automatic	f01b4d95cf55d32a	Unknown AppId	9	12,288
D:\DFIRArtifactMuseum\Wind...	Custom	590aee7bdd69b59b	Unknown AppId	3	5,449
D:\DFIRArtifactMuseum\Wind...	Custom	ccba5a5986c77e43	Unknown AppId	2	3,888

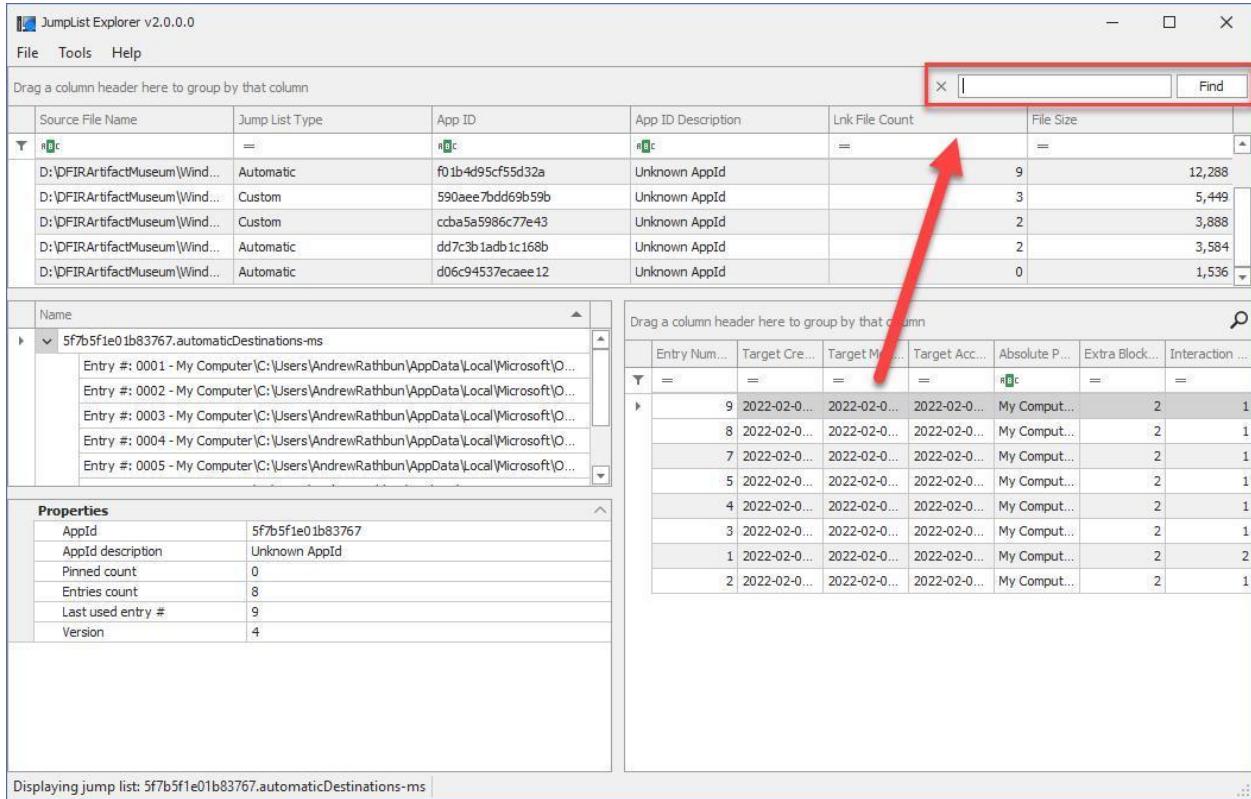
Name	Value
TargetCreationDate	2022-02-05 19:31:20
TargetModificationDate	2019-12-07 09:12:42
TargetLastAccessedDate	2022-02-08 20:58:00
Header.DataFlags	HasTargetIdList, HasLinkInfo, HasIconLocation, I...
Header.FileAttributes	FileAttributeArchive
Header.FileSize	2,539
Header.IconIndex	0
Header.ShowWindow	SwNormal
Absolute path	Shared Documents Folder (Users Files)\AppData\...
IconLocation	%SystemRoot%\system32\WindowsPowerShell\...
LocalPath	C:\Users\AndrewRathbun\AppData\Roaming\Mi...
LocationFlags	VolumeIdAndLocalBasePath

## Search

The search bar can be enabled by clicking on the magnifying glass in the top right corner, as seen below.

The screenshot shows the JumpList Explorer application interface. At the top, there's a menu bar with File, Tools, and Help. Below the menu is a toolbar with several icons. A red arrow points to the magnifying glass icon in the top right corner of the window frame. A red callout box with the text "Click here to enable the search functionality" is positioned over the search icon. The main area contains two tables. The left table has columns: Source File Name, Jump List Type, App ID, App ID Description, Lnk File Count, and File Size. The right table has columns: Name, Entry Num..., Target Cre..., Target Mod..., Target Acc..., Absolute P..., Extra Block..., and Interaction ... . Both tables have a header row with filter icons. On the left, there's a tree view under the heading "Properties" showing entries like AppId, AppId description, Pinned count, etc. At the bottom, a status bar displays "Displaying jump list: 5f7b5f1e01b83767.automaticDestinations-ms".

Once the magnifying glass is clicked on, the search bar will be visible, as seen below.



# JumpList Explorer References

## Associated GitHub Repositories

- <https://github.com/EricZimmerman/JumpList> is the C# library for parsing JumpList files
- <https://github.com/EricZimmerman/OleCf> is the C# library for parsing OLE compound files, which Jump Lists are

Please note, JumpList Explorer is not an open source tool.

## Download JumpList Explorer

JumpList Explorer can be downloaded from <https://ericzimmerman.github.io/#!index.md>

## JumpList Sample Data

Sample JumpList files can be found at the [DFIRArtifactMuseum](#)<sup>157</sup>.

---

<sup>157</sup><https://github.com/AndrewRathbun/DFIRArtifactMuseum/tree/main/Windows/JumpLists>

# MFT Explorer

## MFT Explorer Introduction

MFT Explorer is a tool created by Eric Zimmerman that can be used by forensic examiners to ingest \$MFT files and visually explore the contents. Please note, this tool is not meant to ingest large \$MFT files and therefore should be used on smaller \$MFTs and purely for educational purposes. Everyday forensic examiners should use MFTECmd to parse \$MFT files and export the output to CSV.

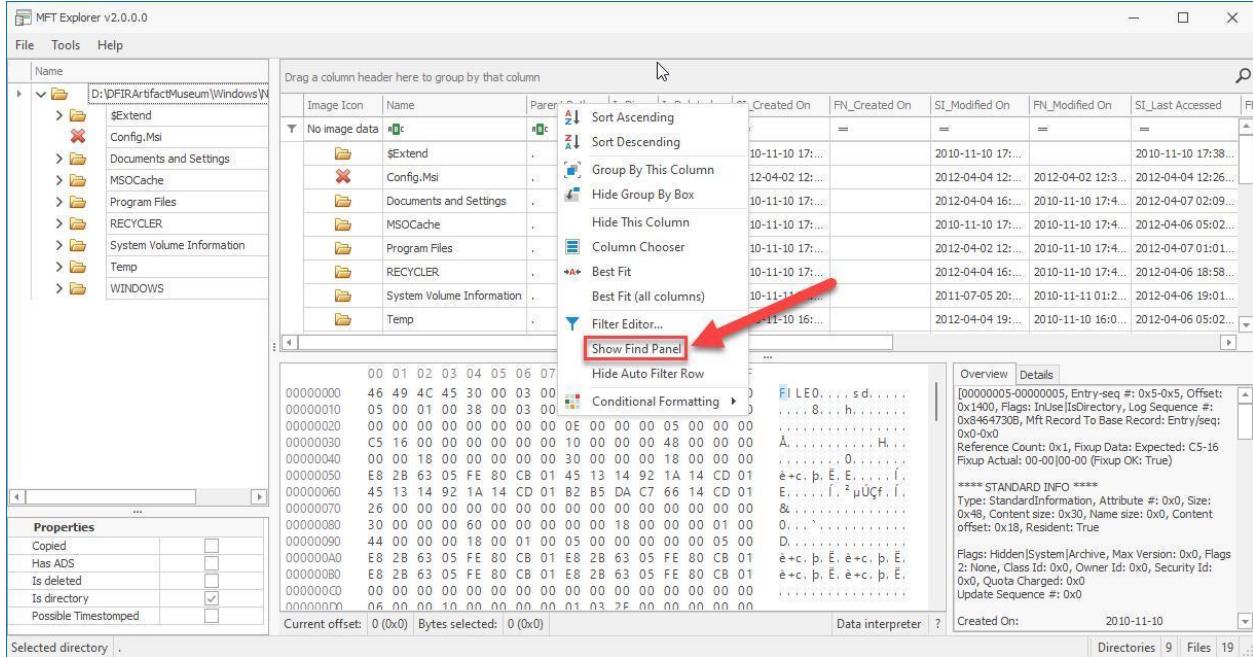
The screenshot shows the MFT Explorer interface. On the left, a tree view displays the file system structure under 'D:\DFIRArtifactMuseum\Windows\N'. The main pane shows a table of file entries with columns: Image Icon, Name, Parent Path, Is Dir, Is Deleted, SI\_Created On, FN\_Created On, SI\_Modified On, FN\_Modified On, SI\_Last Accessed, and FN\_Last Accessed. Below the table is a hex dump of a file entry, showing bytes from 00 to DE. To the right of the hex dump are two tabs: 'Overview' and 'Details'. The 'Overview' tab shows standard information like Entry-seq #, Offset, Flags, and Log Sequence #. The 'Details' tab provides more specific details such as Attribute #, Size, Content size, Name size, Content offset, and Resident status. At the bottom, there are buttons for 'Properties', 'Copied', 'Has ADS', 'Is deleted', 'Is directory', 'Possible Timestamped', 'Current offset', 'Bytes selected', 'Data interpreter', and 'Directories' and 'Files' counts.

MFT Explorer with the tdungan \$MFT ingested

# MFT Explorer Features

## Find Panel

Enabling the Find panel is done by simply right-clicking on a column header and clicking Show Find Panel, as seen below.



Enabling the Find panel in MFT Explorer

Once enabled, the Find panel will be visible in the top right corner, as seen below.

The screenshot shows the MFT Explorer interface with the 'Find' panel activated. The left pane displays a file tree under 'D:\DFIRArtifactMuseum\Windows\N'. The main pane shows a table of file and directory entries. The top right corner features a search bar with 'Enter text to search...' and a 'Find' button, both highlighted with a red box. Below the table, there's a hex dump of selected data and detailed information about the selected item.

Name	Image Icon	Name	Parent Path	Is Dir	Is Deleted	SI_Created On	FN_Created On	SI_Modified On	FN_Modified On	SI_Last Accessed
No image data	No image	\$Extend	D:\DFIRArtifactMuseum\Windows\N			=	=	=	=	=
		\$Extend	.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	2010-11-10 17:...		2010-11-10 17:...		2010-11-10 17:38...
		Config.Msi	.	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2012-04-02 12:...		2012-04-02 12:3...		2012-04-04 12:26...
		Documents and Settings	.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	2010-11-10 17:...		2012-04-04 16:...	2010-11-10 17:4...	2012-04-07 02:09...
		MSOCache	.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	2010-11-10 17:...		2010-11-10 17:...	2010-11-10 17:4...	2012-04-06 05:02...
		Program Files	.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	2010-11-10 17:...		2012-04-02 12:...	2010-11-10 17:4...	2012-04-07 01:01...
		RECYCLER	.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	2010-11-10 17:...		2012-04-04 16:...	2010-11-10 17:4...	2012-04-06 18:58...
		System Volume Information	.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	2010-11-10 17:...		2011-07-05 20:...	2010-11-10 01:2...	2012-04-06 19:01...
		Temp	.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	2010-11-10 16:...		2012-04-04 19:...	2010-11-10 16:0...	2012-04-06 05:02...
		WINDOWS	.							

Selected directory: .

Properties:

- Copied:
- Has ADS:
- Is deleted:
- Is directory:
- Possible Timestamped:

Current offset: 0 (0x0) Bytes selected: 0 (0x0) Data interpreter ?

Overview Details  
[00000005-00000005, Entry-seq #: 0x5-0x5, Offset: 0x1400, Flags: InUse|IsDirectory, Log Sequence #: 0x8464730B, MFT Record To Base Record: Entry/seq: 0x0-0x0 Reference Count: 0x1, Fixup Data: Expected: C5-16 Fixup Actual: 00-00|00-00 (Fixup OK: True)  
\*\*\*\*\* STANDARD INFO \*\*\*\*\*  
Type: StandardInformation, Attribute #: 0x0, Size: 0x48, Content size: 0x30, Name size: 0x0, Content offset: 0x18, Resident: True  
Flags: Hidden|System|Archive, Max Version: 0x0, Flags 2: None, Class Id: 0x0, Owner Id: 0x0, Security Id: 0x0, Quota Charged: 0x0  
Update Sequence #: 0x0  
Created On: 2010-11-10  
Directories 9 | Files 19 | ...]

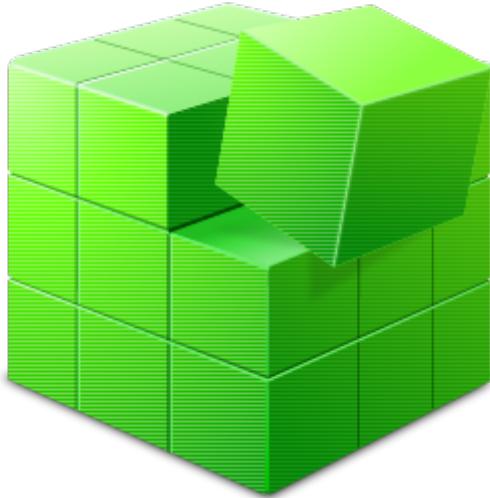
Find panel enabled in MFT Explorer

## MFT Explorer References

### Download MFT Explorer

MFT Explorer can be downloaded from <https://ericzimmerman.github.io/#!index.md>

# Registry Explorer



## Revision history

2015-07-01 Rev. 1 – Initial release  
2016-06-08 Rev. 2 – Updated for v0.8.1.0  
2017-05-19 Rev. 3 – Updated for v0.9.0.0  
2017-03-02 Rev. 4 – Updated for v1.0.0.0  
2019-01-15 Rev. 5 – Updated for v1.2.0.0  
2022-07-14 Rev. 6 - Leanpub release

## Registry Explorer Introduction

Registry Explorer is a GUI based tool used to view the contents of offline Registry Hives. It can load multiple hives at once, search across all loaded hives using strings or regular expressions, exporting of data, and much more.

## Why Another Registry Tool?

The need for Registry Explorer and RECcmd came about out of writing a fully managed offline Registry hive parser in C#. Existing parsers did not offer the features I was looking for and as such, research and coding began. The Registry project serves as the basis for several programs including ShellBags Explorer, AppCompatParser, etc. Once the back end was mature, I wanted an easy to use and powerful way to expose the capabilities of the parser.

Registry Explorer fills the gaps in existing tools and expands the capabilities of Registry viewers in many unique and powerful ways. It is GUI based and contains powerful searching, filtering, and other visualization concepts that makes exploring Registry hives very easy while exposing all the technical information contained in Registry hives.

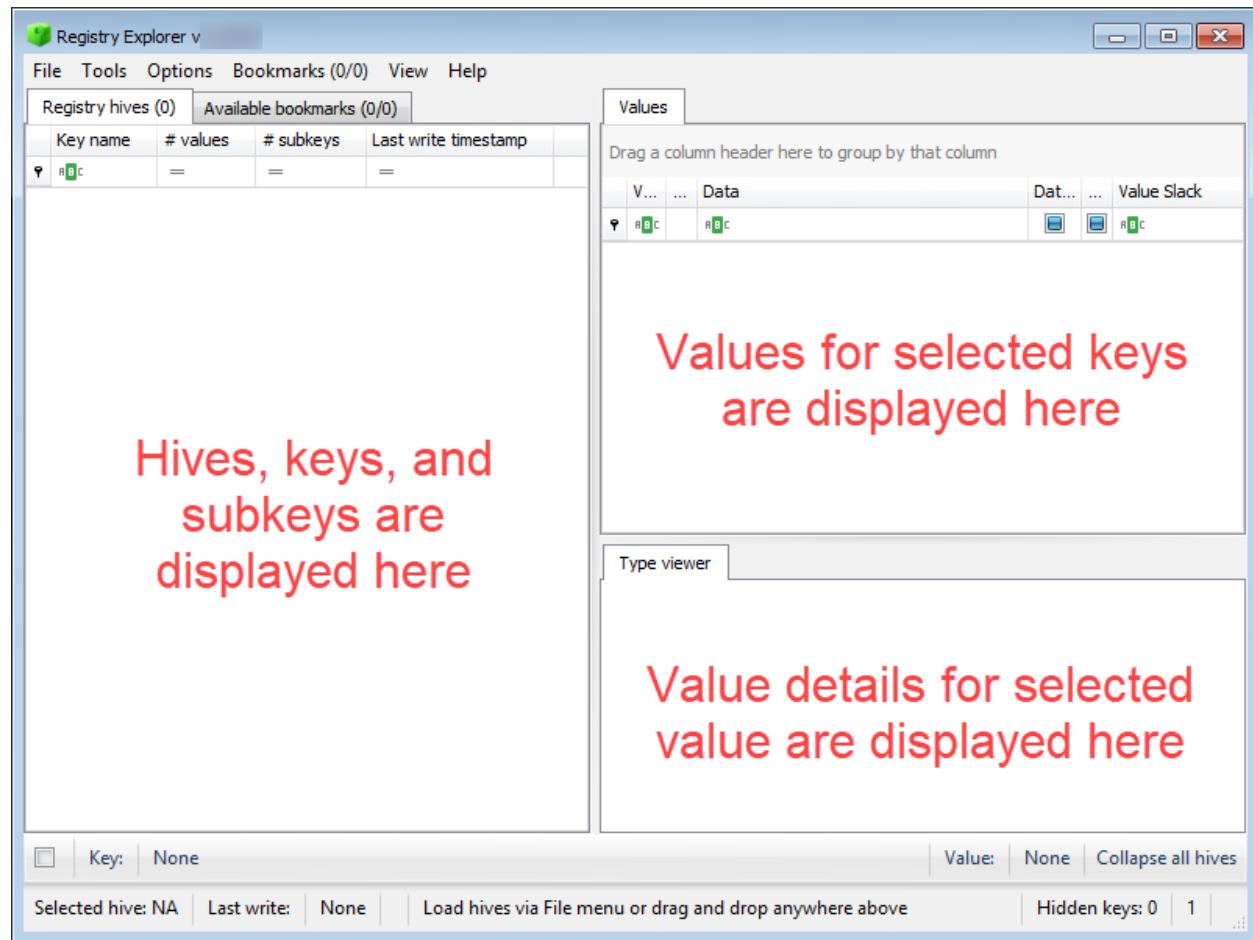
RECmd was created in order to be able to script access to Registry hives, conduct new research, and automate searching across multiple Registry hives at once from the command line.

Because both tools use the same back end, both have the same searching and viewing capabilities including the full recovery of deleted keys and values. The parser also exposes value slack.

In summary, the capabilities of Registry Explorer and RECmd allows for quickly examining multiple hives at once and they can be leveraged to find new places where currently understood data is located in an easy to use and systematic way. It can be used in educational settings to not only understand the Registry from a functional level, but also from a deeply technical perspective.

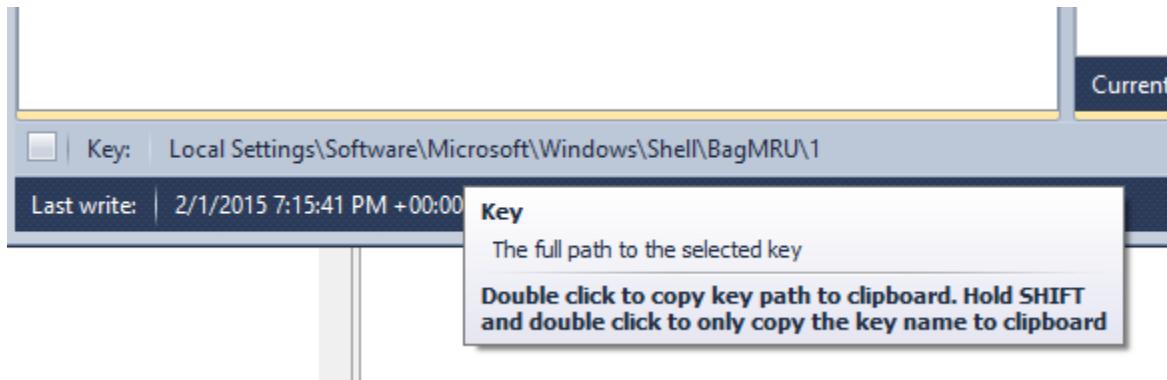
## Getting Started

After starting Registry Explorer, the main interface is displayed.



Settings for various things like program options, window size, slider positions, window positions, recent searches, etc. are all saved and reloaded between program executions. You can reset these options by deleting the relevant files under the Settings directory in the main Registry Explorer folder. The .1layout files are for the trees and grids.

Tooltips are shown when hovering over different areas of the program. For example, hovering over the Key section of the status bar shows the following:



## Interface sections

There are five sections to the main interface.

### Registry hives

On the left side of the window is the Registry hives tab. This tab displays the Registry hives that have been loaded and the keys contained therein. Once at least one hive is loaded and a key is selected, a context menu is available by right clicking on a key. The context menu options will be discussed below in the Key context menu section.

### Available bookmarks

Next to the Registry hives tab is the Available bookmarks tab. This tab will be discussed in detail below.

### Values

The Values grid shows all the values contained in the key that is selected in the Registry hives tab. Once a value is selected, a context menu is available by right clicking on a value. The context menu options will be discussed below.

## Value details

The Value details area contains one or more tabs that dynamically adjust depending the type of value selected. In every case, a type viewer will be displayed that shows the value of the selected key. If a value has slack, a separate tab will be shown that allows you to view the slack space in a hex viewer.

These concepts will be explained in more detail in the Using Registry Explorer section below.

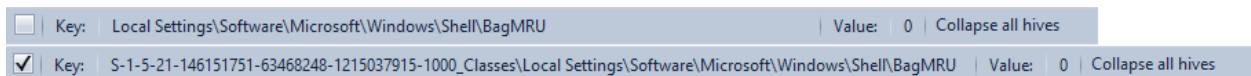
## Status bars

Across the bottom of the interface are several status bars as seen below.



### Top status bar

The top status bar contains details about the path to the selected key and the selected value. On the far left is a check box that toggles whether to show the root key name in the key path. By default, the root key path is not shown. The screen shot below shows what this option does when turned on and off.



By hiding the root key name, longer key paths will not be truncated as different keys are selected. To the far right of the top status bar is a button that, when clicked, will collapse all loaded hives back to their default state. This is a handy shortcut to clean up the Registry hives tree after interacting with it and expanding many keys and subkeys.

Double clicking the key path will copy the key path to the clipboard. Holding Shift and double clicking will copy only the key name to the clipboard.

Double clicking the value will copy the value's name to the clipboard. Holding Shift and double clicking will copy the value's data to the clipboard.

### Bottom status bar

The bottom status bar contains the last write timestamp, the status of filters for values, a section for general status messages, an indicator of the total number of keys that are hidden from view, and the total number of messages available on the Messages form.

Double clicking on the Total messages counter will show the Messages form. If there are any errors in the Messages form, the background will be changed to yellow. If there are any errors, the background

will be changed to red. When the Messages tab is viewed, the background color will be changed back to its default.

Double clicking the last write timestamp will copy it to the clipboard.

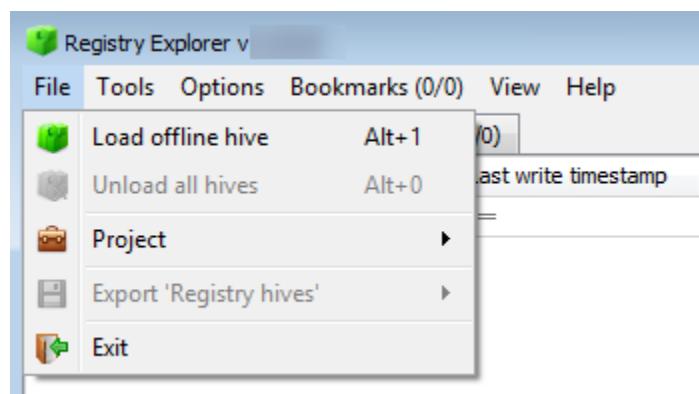
## Main menu

The main menu contains options that allows for loading hives, searching hives, opening bookmarks, and so on. In many cases, the menu items will have shortcut keys associated with them. Pressing the keys shown by a menu item on the keyboard will activate that menu item.

The various sections below will explain these submenus. Where things are obvious (like File | Exit), no additional information will be provided.

### File

The File menu contains options for loading hives (you can also simply drag and drop one or more hives onto the main interface to load them) and exporting.



- Load offline hive: Allows for loading one or more hives. To select more than one file, select a file, then hold Shift and select the last file to load. You can also hold Ctrl and click files to select them individually.
- Unload all hives: Unloads all hives at once vs. removing one at a time
- Project: Allows for loading/saving of projects. Projects will be discussed below.
- Export 'Registry hives': Exports what is shown in the Registry hives tab to a variety of formats. As an example, if the Registry hives tree looked like this:

The screenshot shows the Registry Explorer interface with two open registry hives:

- C:\Temp\NTUSER\_dblake.DAT**: Contains subkeys like AppEvents, AppXBackupContentType, Control Panel, Environment, EUDC, Identities, Keyboard Layout, Network, Printers, Software, System, and WXP. It also contains two red folder icons labeled "Associated deleted records" and "Unassociated deleted records".
- C:\Temp\SYSTEM\_dblake**: Contains subkeys like CsiTool-CreateHive-{00000000-0000-0000-0000-000000000000} and two red folder icons labeled "Associated deleted records" and "Unassociated deleted records".

The table lists the registry keys under each hive, showing their key name, # values, # subkeys, and Last write timestamp.

Key name	# values	# subkeys	Last write timestamp
R E C	=	=	=
▲ C:\Temp\NTUSER_dblake.DAT			2013-08-22 19:11:44
↳ CsiTool-CreateHive-{00000000-0000-0000-0000-000000000000}	0	13	2013-10-23 02:56:24
↳ AppEvents	0	2	2013-09-23 19:17:31
↳ AppXBackupContentType	0	2	2013-09-23 19:48:31
▶ Console	37	2	2013-09-23 19:17:31
↳ Control Panel	0	15	2013-10-07 11:37:40
↳ Environment	2	0	2013-09-23 19:17:31
↳ EUDC	0	4	2013-09-23 19:17:31
↳ Identities	6	1	2013-09-23 19:47:21
↳ Keyboard Layout	0	3	2013-09-23 19:22:19
↳ Network	0	0	2013-10-21 20:01:56
↳ Printers	0	5	2013-10-01 21:07:39
↳ Software	0	32	2013-10-23 03:09:34
↳ System	0	1	2013-09-23 19:17:31
↳ WXP	0	0	2013-09-23 19:17:52
↳ Associated deleted records	0	0	
↳ Unassociated deleted records	0	0	
▲ C:\Temp\SYSTEM_dblake			2013-08-22 19:11:50
↳ CsiTool-CreateHive-{00000000-0000-0000-0000-000000000000}	0	8	2013-10-23 02:56:10
↳ Associated deleted records	0	0	
↳ Unassociated deleted records	0	0	

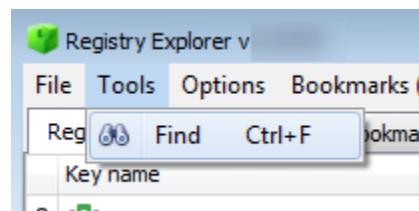
Exporting to PDF would generate a PDF file that contains the following:

Key name	# values	# subkeys	Last write timestamp
C:\Temp\NTUSER_dblake.DAT			2013-08-22 19:11:44
CsiTool-CreateHive-{00000000-0000-0000-0000-0000}	0	13	2013-10-23 02:56:24
AppEvents	0	2	2013-09-23 19:17:31
AppXBackupContentType	0	2	2013-09-23 19:48:31
Console	37	2	2013-09-23 19:17:31
Control Panel	0	15	2013-10-07 11:37:40
Environment	2	0	2013-09-23 19:17:31
EUDC	0	4	2013-09-23 19:17:31
Identities	6	1	2013-09-23 19:47:21
Keyboard Layout	0	3	2013-09-23 19:22:19
Network	0	0	2013-10-21 20:01:56
Printers	0	5	2013-10-01 21:07:39
Software	0	32	2013-10-23 03:09:34
System	0	1	2013-09-23 19:17:31
WXP	0	0	2013-09-23 19:17:52
Associated deleted records	0	0	
Unassociated deleted records	0	0	
C:\Temp\SYSTEM_dblake			2013-08-22 19:11:50
CsiTool-CreateHive-{00000000-0000-0000-0000-0000}	0	8	2013-10-23 02:56:10
Associated deleted records	0	0	
Unassociated deleted records	0	0	

This is useful for generating reports or other documentation that is easier to manipulate than simply taking a screen shot.

## Tools

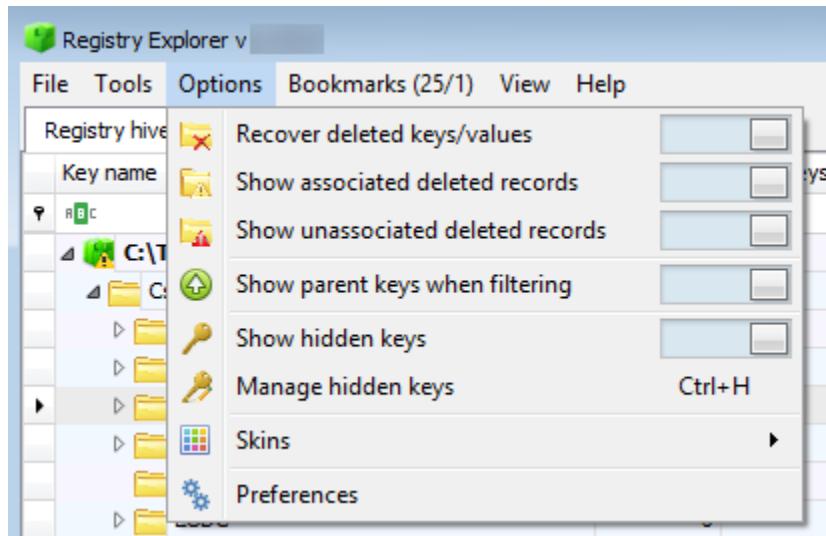
The Tools menu contains a single item, Find.



Using this option will be explained in full detail below in the Using Registry Explorer section

## Options

This menu contains several options that control such things as recovering deleted records, viewing hidden keys, etc.



- Recover deleted keys/values: When enabled (the selector is to the right), Registry Explorer will recover any deleted records available during hive loading
- Show associated deleted records: When enabled, all associated deleted records will be shown in a special group under the main Registry hive data. Any recovered keys that could be associated with an active (that is, not deleted) key will also be shown in relation to the active key. This will be explained more in a subsequent section.
- Show unassociated deleted records: Like the previous option, but this group contains all of the keys that could not be associated with an active key.
- Show parent keys when filtering: This option changes the way the Registry hives tree works when using the column filters. When this option is enabled, any keys that match a filter will be displayed, along with the parent keys that the matching key belongs to as seen in the screen shot below.

The keys highlighted in yellow are parent keys that may not contain the text entered in the filter column.

Key name	#...	Last write tim...
0		
D:\temp\re\UsrClassDeletedBags.dat		2/1/2015 7:1...
S-1-5-21-146151751-63468248-1215037915-1000_Classes	0	9/19/2011 4:...
Local Settings	0	9/19/2011 4:...
Software	0	9/19/2011 4:...
Microsoft	0	9/19/2011 4:...
Windows	0	9/19/2011 4:...
Shell	0	9/19/2011 4:...
BagMRU	4	2/1/2015 7:1...
0	3	2/1/2015 7:1...
0	2	2/1/2015 7:1...
0	2	2/1/2015 7:1...
1	2	2/1/2015 7:1...
0	2	2/1/2015 7:1...
0	3	2/1/2015 7:1...
0	2	9/19/2011 4:...
Bags	0	2/1/2015 7:1...
1	0	9/19/2011 4:...
Shell	0	9/19/2011 4:...
{5C4F28B5-F869-4E84-8E60-F11DB97C5CC7}	12	9/19/2011 4:...
2	0	9/19/2011 4:...
Shell	0	9/19/2011 4:...
{5C4F28B5-F869-4E84-8E60-F11DB97C5CC7}	12	9/19/2011 6:...
3	0	9/19/2011 4:...
Shell	0	9/19/2011 4:...
{5C4F28B5-F869-4E84-8E60-F11DB97C5CC7}	12	9/19/2011 4:...

If we turn off this option, we get a much different result as seen below.

Registry hives		Available bookmarks (1/1)	
Key name	# values		Last write timestamp
0			
S-1-5-21-146151751-63468248-1215037915-1000_Classes	0	9/19/2011 4:30:48 PM +...	
0	3	2/1/2015 7:14:41 PM +...	
0	2	2/1/2015 7:14:41 PM +...	
0	2	2/1/2015 7:14:41 PM +...	
0	3	2/1/2015 7:15:41 PM +...	
0	2	9/19/2011 4:42:44 PM +...	
{5C4F28B5-F869-4E84-8E60-F11DB97C5CC7}	12	9/19/2011 4:41:32 PM +...	
{5C4F28B5-F869-4E84-8E60-F11DB97C5CC7}	12	9/19/2011 6:56:43 PM +...	
{5C4F28B5-F869-4E84-8E60-F11DB97C5CC7}	12	9/19/2011 4:42:44 PM +...	

Notice in this screen shot only the keys that match the filter criteria are shown. This can greatly reduce noise in the results in addition to lessening the need to scroll to the keys that match the filter criteria.

The other aspect this option controls is whether to show subkeys of keys that match a given filter. With the option on, any subkeys of keys that match the filter will continue to be shown. With the option off, subkeys of keys matching the filter are hidden.

- Show hidden keys: When enabled, any keys that have been hidden will be shown in the Registry hives tree. In the screen shot below, several keys are shown.

Key name	# values	Last write timestamp
D:\temp\re\UsrClassDeletedBags.dat		2/1/2015 7:15:49 PM +...
S-1-5-21-146151751-63468248-1215037915-1000_Classes	0	9/19/2011 4:30:48 PM +...
Local Settings	0	9/19/2011 4:31:27 PM +...
MuiCache	0	9/19/2011 7:02:08 PM +...
6	0	9/19/2011 7:02:08 PM +...
52C64B7E	163	2/1/2015 7:15:05 PM +...
Software	0	9/19/2011 4:31:27 PM +...
VirtualStore	0	9/19/2011 6:58:04 PM +...
Associated deleted records	0	

While we haven't discussed how to hide keys yet (it has its own section below), if we right click on a key, an option to hide the selected key (based on the key path, not just the key name) is shown.

For example, if we hide the MuiCache key, it will disappear, as seen below.

Key name	# values	Last write timestamp
D:\temp\re\UsrClassDeletedBags.dat		2/1/2015 7:15:49 PM +0...
S-1-5-21-146151751-63468248-1215037915-1000_Classes	0	9/19/2011 4:30:48 PM +...
Local Settings	0	9/19/2011 4:31:27 PM +...
Software	0	9/19/2011 4:31:27 PM +...
VirtualStore	0	9/19/2011 6:58:04 PM +...
Associated deleted records	0	

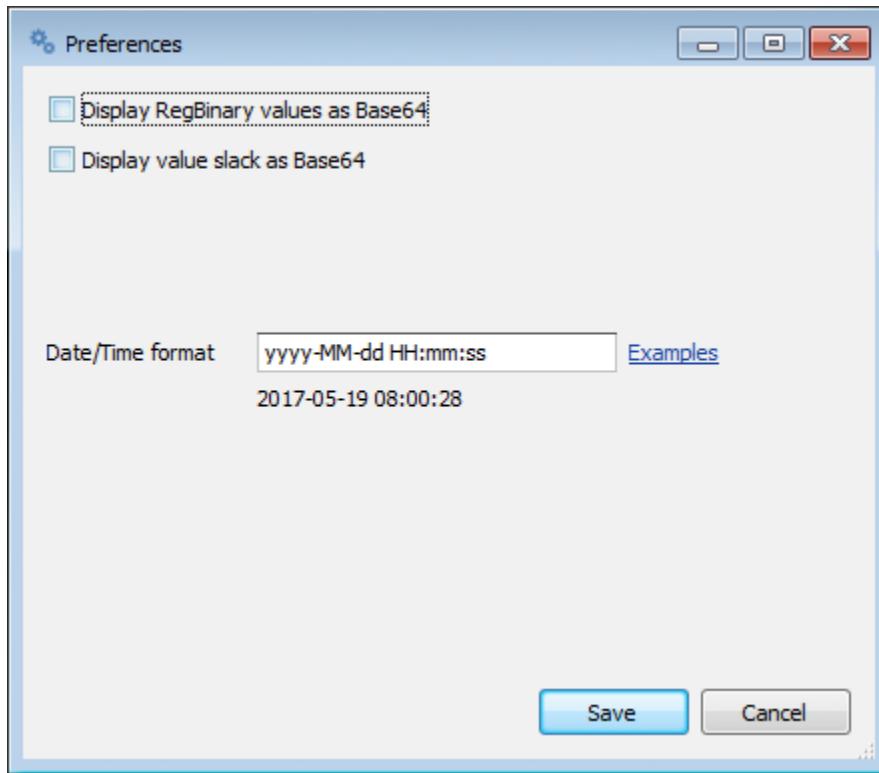
Notice the MuiCache key is no longer visible (assuming the Show hidden key option is off). If we enable this option, the MuiCache key will be shown in its original place, but the icon is different to show that it is in fact a hidden key.

Key name	# values	Last write timestamp
D:\temp\re\UsrClassDeletedBags.dat		2/1/2015 7:15:49 PM +0...
S-1-5-21-146151751-63468248-1215037915-1000_Classes	0	9/19/2011 4:30:48 PM +...
Local Settings	0	9/19/2011 4:31:27 PM +...
MuiCache	0	9/19/2011 7:02:08 PM +...
52C64B7E	163	2/1/2015 7:15:05 PM +0...
Software	0	9/19/2011 4:31:27 PM +...
VirtualStore	0	9/19/2011 6:58:04 PM +...
Associated deleted records	0	

When a key is hidden, the lower right corner will have a red dash to indicate this.

- Manage hidden keys: Brings up an interface to remove keys from the auto hide list. There are two options available when hiding keys: hide for session and hide and add to auto hide. The Manage hidden keys interface only displays key paths that have previously been added to the auto hide list. Any key paths removed from the auto hide list will be unhidden when the Manage hidden keys interface is closed. Additional ways to unhide keys will also be discussed in a subsequent section.
- Skins: Allows for selecting a skin or theme that Registry Explorer will use.
- Preferences: Program options such as timestamp format, binary data as base64, etc.

The Preferences dialog allows you to change the default timestamp format and other parameters as seen below.



## Bookmarks

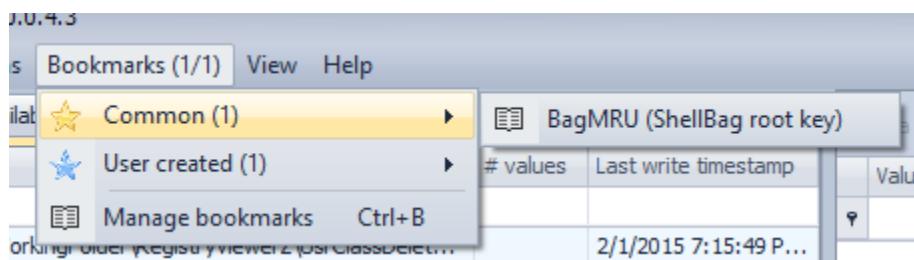
The Bookmarks menu contains both common (included with Registry Explorer) and user created bookmarks to “of interest” Registry keys. Bookmarks can be created for any Registry key (we will see how to create our own bookmarks soon). Bookmarks that are included with Registry Explorer will show up under the ‘Common’ menu and any user created bookmarks will appear under the ‘User created’ menu.

Bookmarks live in a subdirectory of the main Registry Explorer program directory in a directory named Bookmarks. The Bookmarks directory contains two subdirectories, Common and User. To move a user created bookmark from the User created to Common submenu, simply move the bookmark file from the User directory to the Common directory.

The Manage bookmarks interface can be used to edit or delete bookmarks. Additionally, simply deleting the bookmark file from the Common or User directory will also remove the bookmark.

Bookmarks are simple json files and can also be edited with any text editor. Since they are simple json files, exchanging a good set of bookmarks with other users is as easy as sending someone else the bookmark files from the User directory. There is a project on GitHub, found [here<sup>158</sup>](https://github.com/EricZimmerman/RegistryExplorerBookmarks), that you can push your Bookmarks to.

<sup>158</sup><https://github.com/EricZimmerman/RegistryExplorerBookmarks>



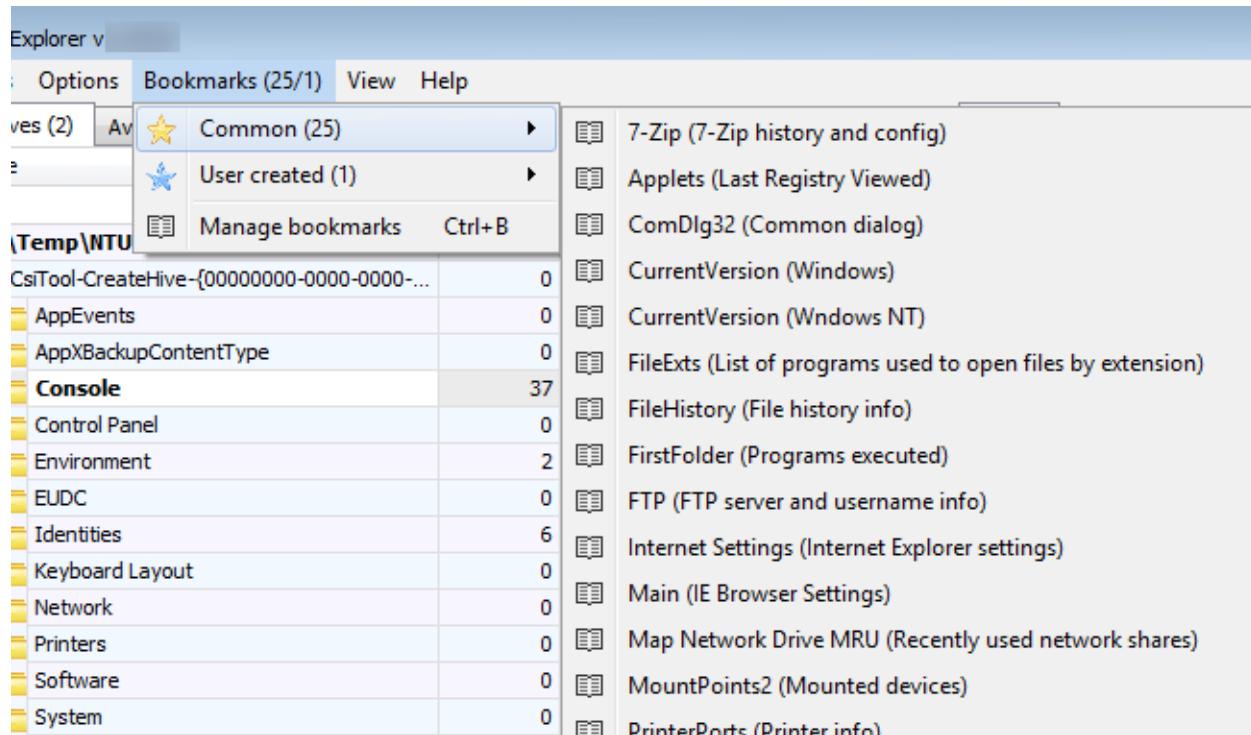
The main Bookmarks menu contains two numbers at the end. The first number is the total number of Common bookmarks that exist in the selected hive and the last number is the number of User bookmarks that exist in the selected hive. Clicking on any of the bookmarks will cause Registry Explorer to jump to the bookmarked key.

Bookmarks are tied to a Registry hive type and a key path within that hive type. When we discuss creating bookmarks below this will become clearer, but for now remember that each bookmark is associated with a certain flavor of Registry hive (NTUSER, UsrClass, SYSTEM, etc.).

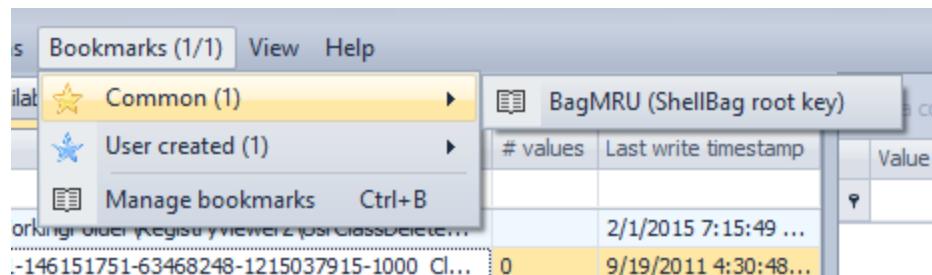
The Bookmarks menus dynamically adjust as hives are loaded and selected. For example, suppose you have the following bookmarks by hive type:

- NTUSER.DAT: 40 bookmarks
- USRCLASS.DAT: 8 bookmarks

You then load an NTUSER and USRCLASS hive. The NTUSER hive contains 25 out of the 40 key paths as defined in the NTUSER.DAT related bookmarks (25 from Common). The USRCLASS hive contains two out of the eight bookmarks (one from common and one from user created). If you click on anything in the NTUSER.DAT hive, the Bookmarks menu will change to show you only the bookmarks that actually exist in the NTUSER hive, like this:



If you then click on the USRCLASS hive, the Bookmarks menu will again dynamically adjust to show what is available in the USRCLASS hive.



Again, clicking a bookmark will jump to the key as defined in the bookmark. For example, clicking on the BagMRU bookmark results in the following key being selected (and of course all parent keys will be expanded so the bookmarked key is visible).

Key name	# values	Last write timestamp
D:\temp\re\UsrClassDeletedBags.dat		2/1/2015 7:15:49 PM +0...
S-1-5-21-146151751-63468248-1215037915-1000_Classes	0	9/19/2011 4:30:48 PM +...
Local Settings	0	9/19/2011 4:31:27 PM +...
MuiCache	0	9/19/2011 7:02:08 PM +...
6	0	9/19/2011 7:02:08 PM +...
52C64B7E	163	2/1/2015 7:15:05 PM +0...
Software	0	9/19/2011 4:31:27 PM +...
Microsoft	0	9/19/2011 4:31:27 PM +...
Windows	0	9/19/2011 4:31:27 PM +...
CurrentVersion	0	9/19/2011 4:31:34 PM +...
Shell	0	9/19/2011 4:41:32 PM +...
BagMRU	4	2/1/2015 7:15:41 PM +0...
Bags	0	2/1/2015 7:14:41 PM +0...
VirtualStore	0	9/19/2011 6:58:04 PM +...
Associated deleted records	0	

Because the Bookmarks menu dynamically adjusts itself based solely on what exists in the active hive, you do not have to click on bookmarks before you know whether they exist. This is a huge time saver and makes drilling down into hives much easier.

As you interact with loaded hives, the Bookmarks menu will show you at a glance how many bookmarks are available, but as we will soon see, Registry Explorer has an even easier way to interact with bookmarks (the Available bookmarks tab).

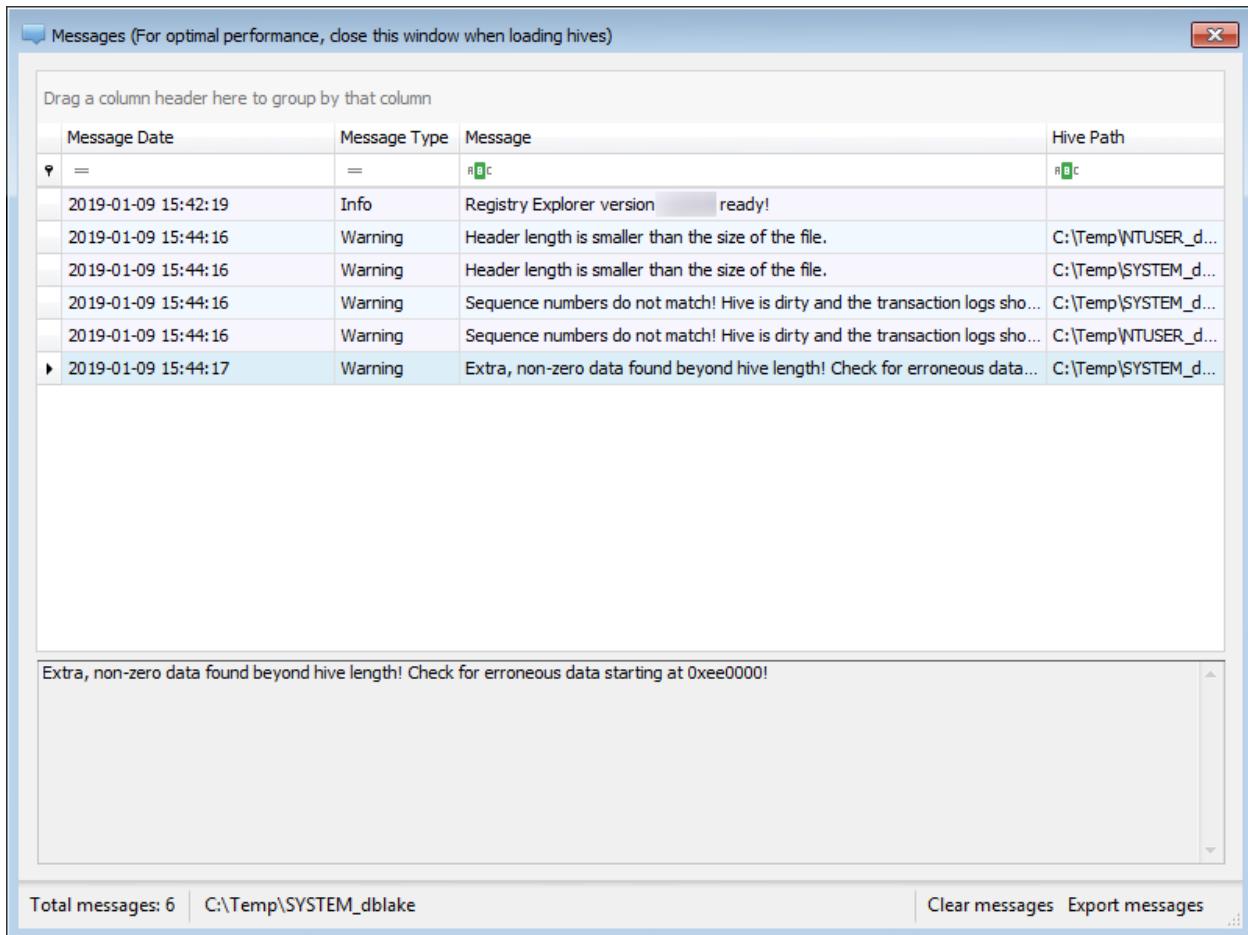
The bookmark names are sorted alphabetically as well so it's easy to find the bookmark you are interested in.

## View

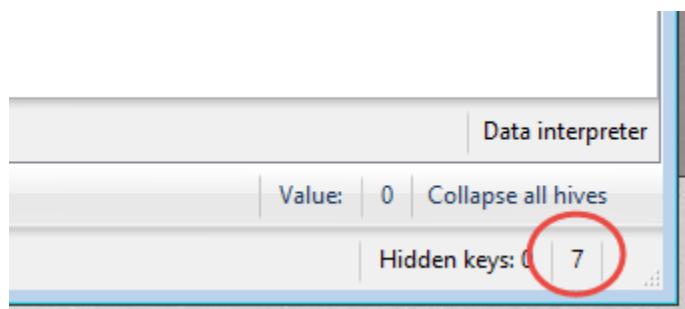
The View menu contains two options: Messages and Plugins.

Messages toggles visibility of the Messages window. The Messages window displays status messages and other feedback as hives are loaded and so on.

Hives tend to process and load faster when the Messages window is hidden, so keep that in mind when loading many hives at once or when processing large hives.

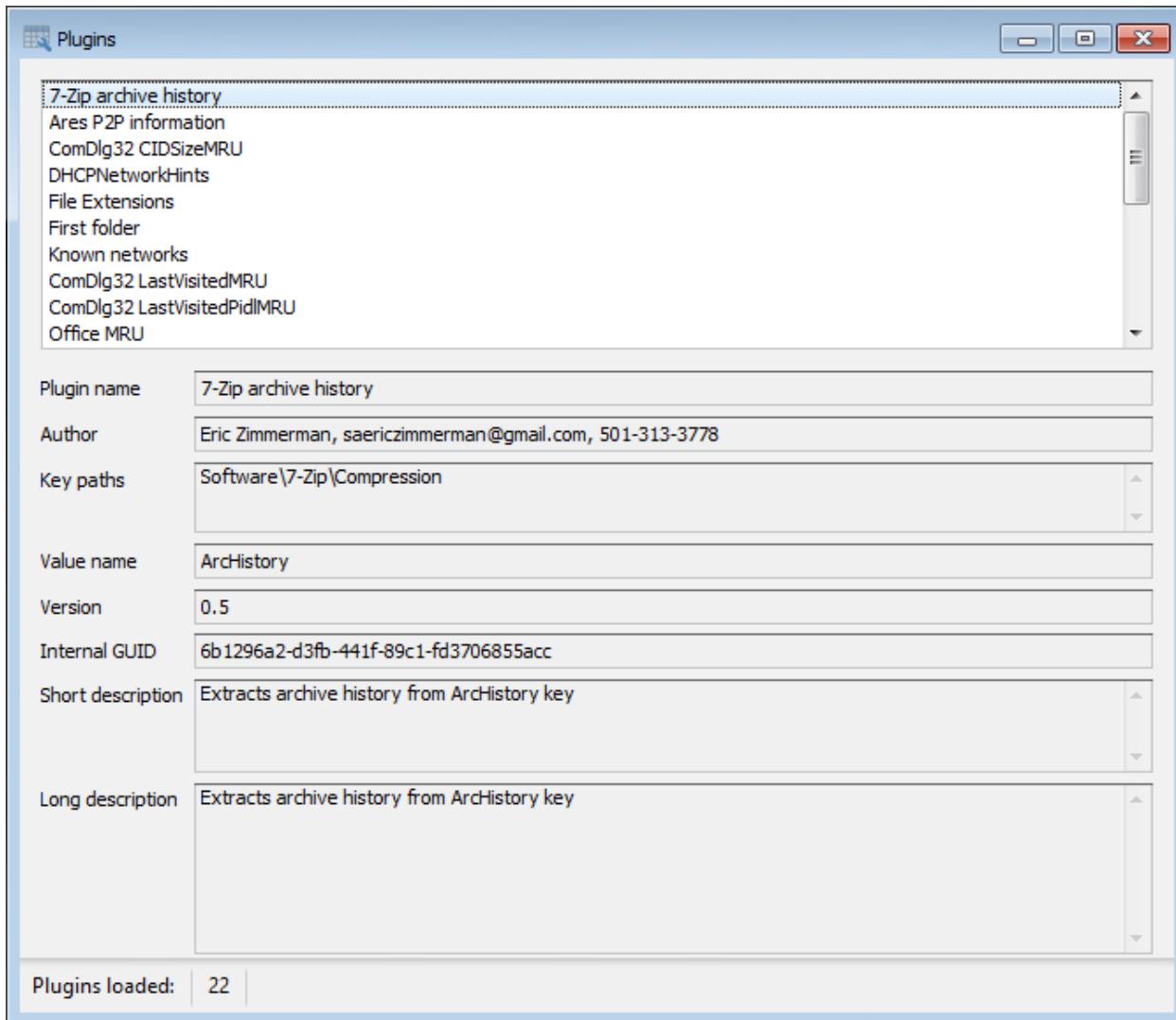


The total number of messages is also shown on the main window's bottom status bar to the far right. Double clicking the message count will show the Messages form.



As mentioned above, the background of the messages count will be yellow if and warning message exists and red if an error message exists. The background color will return to default when the Messages window is shown.

The Plugins option displays a list of available plugins and includes such details as the author, key paths, and descriptions of what the plugin does.

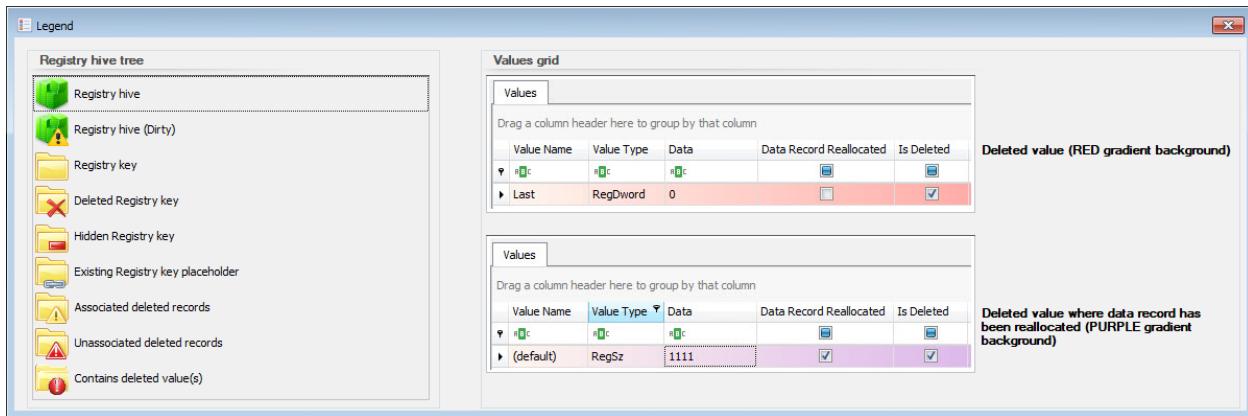


Plugins will be discussed in more detail in a dedicated section of this manual.

## Help

The Help menu contains three options: Quick help, Legend, and About. The Legend shows the various icons seen in the Registry hives tree and a description about them.

The legend contains descriptions for the different icons used for various Registry objects such as hives, keys, and existing key placeholders. The legend can be seen below.



## Using Registry Explorer

### General concepts

Once hives are loaded into Registry Explorer, Registry Explorer allows you to sort, filter, etc. on both the tree on the left as well as any grids on the right.

### Sorting

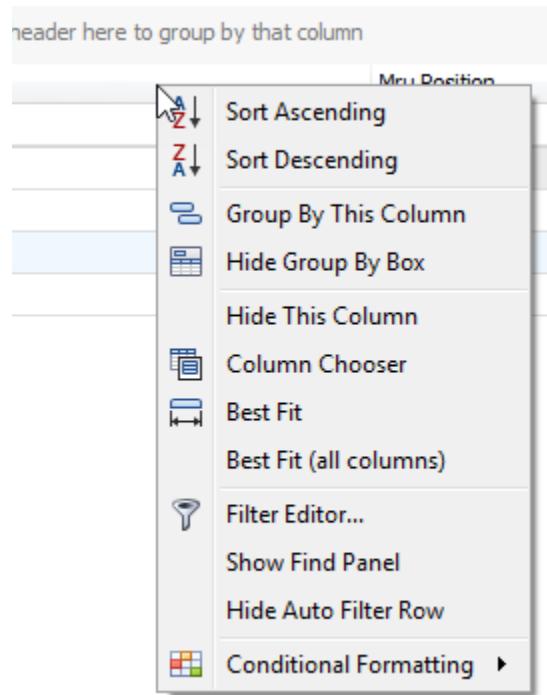
Sorting works like most every other program in that you can click on a column header to sort that column. Here the Value Name column has been sorted.

Value Name	Value Type	Data	Value Type Raw	Value Slack
Drag a column header here to group by that column				
!Do not use this registry key	RegSz	Use the SHGetFolderPath or S...		1 00-00
{00BCFC5A-ED94-4E48-96A1-3F6217F21990}	RegSz	C:\Users\Donald\AppData\Ro...		1 00-00
More registry keys...	RegSz	0x00000000000000000000000000000000		1 42 42 2D 4D

Notice the small arrow indicating the sort order.

### Other options

Right clicking on a column header will bring up a context menu that allows for sorting (as well as removing any existing sorting), grouping, and customization of columns including hiding or showing any columns.

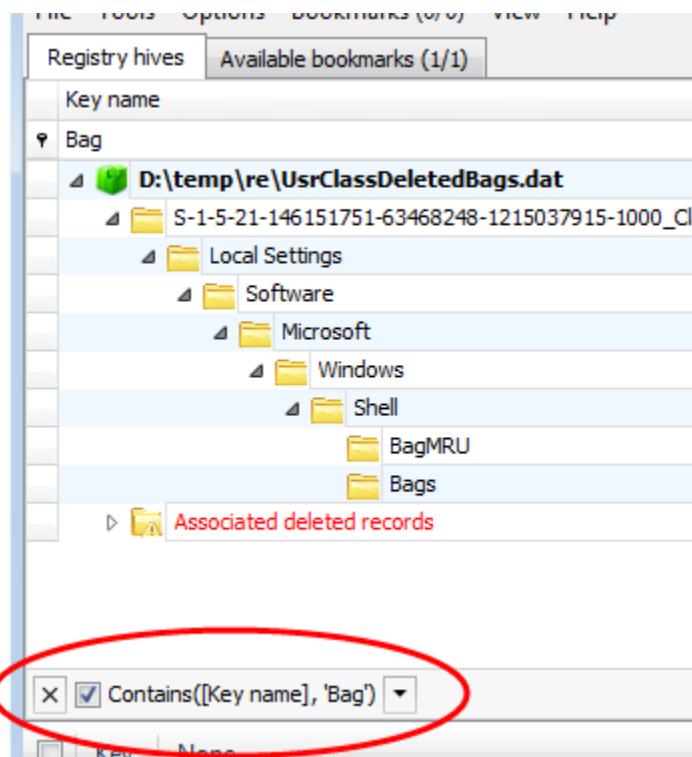


page 21 of 86 done

## Filtering

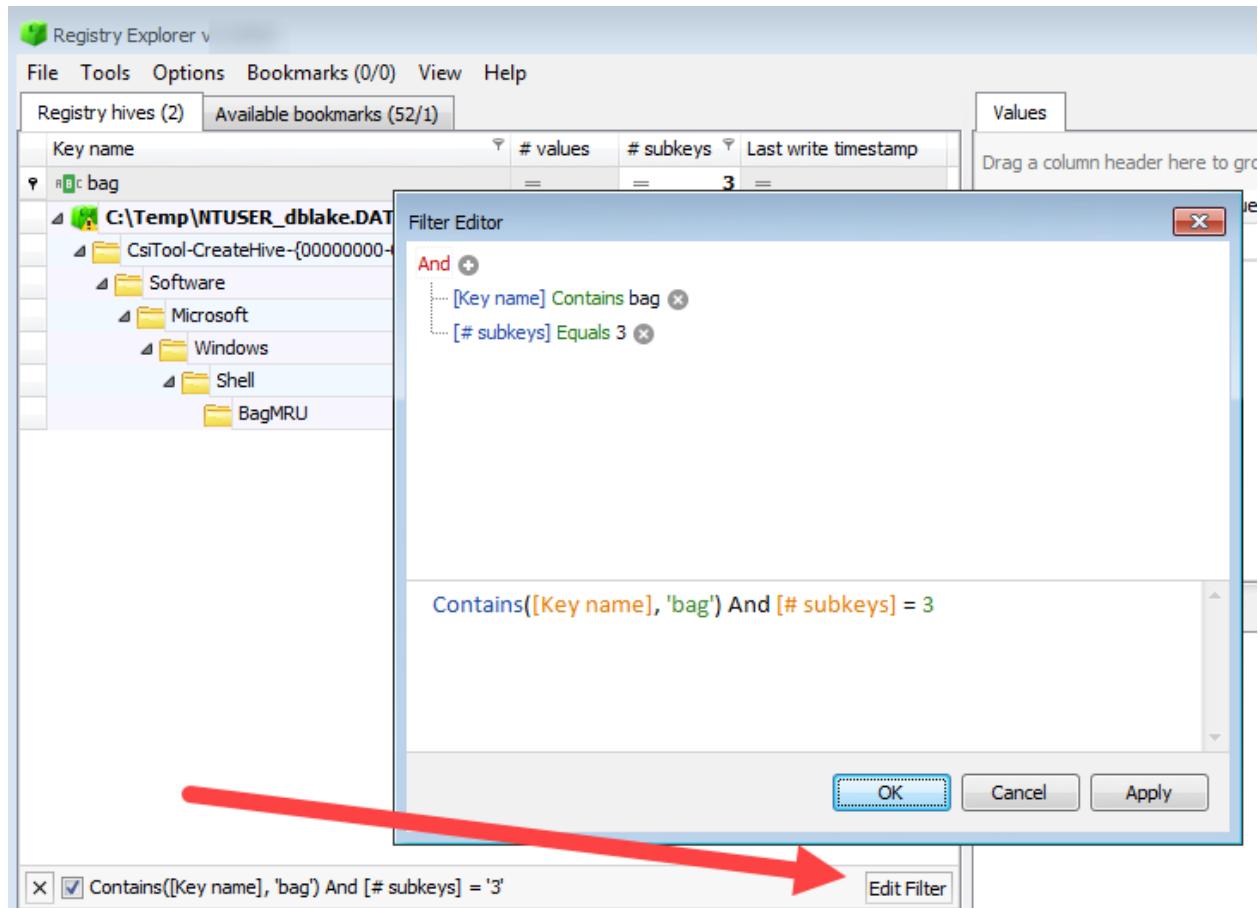
A column can be filtered by clicking in the blank space below the column header and entering something to filter by. The data will be filtered in real time and the bottom status bar will indicate how many things have been filtered out.

When filters are in place (by entering text in the areas below the column name), information about the active filter will be shown at the bottom of the tree or grid as shown below.



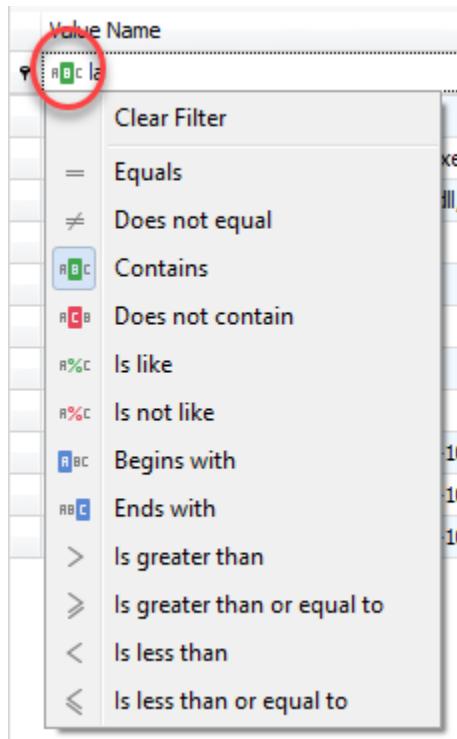
The leftmost X can be used to clear the active filter, the checkbox can be used to disable the filter without clearing it, and the down arrow on the right side contains a history of the different filters that have been recently used.

The 'Edit filter' button on the far right allows you to edit the current filter as needed. This is the same option that is available in the context menu from above.



Using these options, very detailed filters can be created.

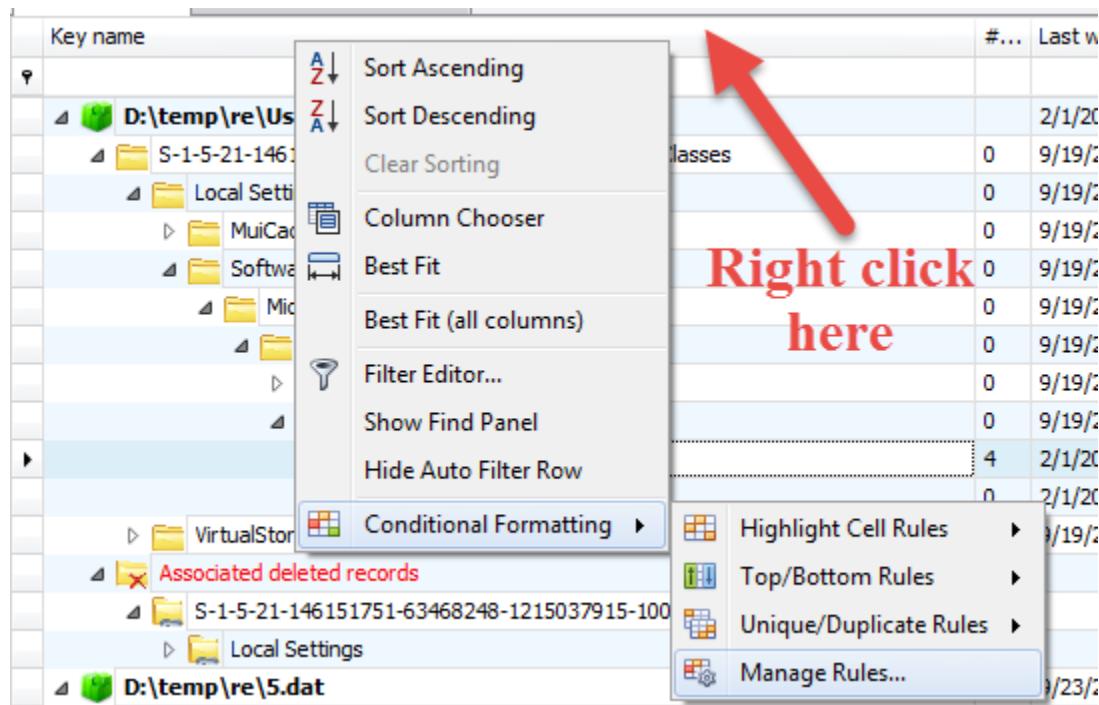
Clicking the icon in the left side of the filter area allows for changing the filter criteria as well. The default is 'Contains'



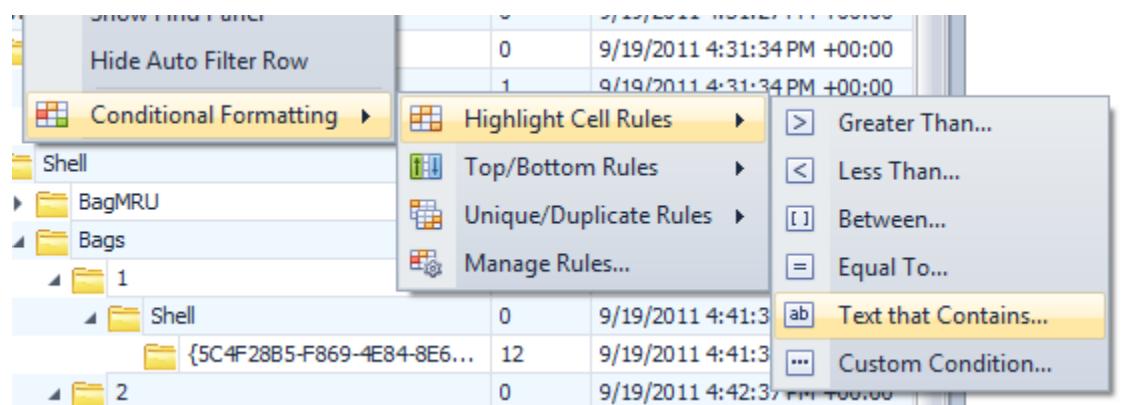
## Conditional formatting

The Registry hives tree, Available bookmarks tree, values grid, and Find results grid all support creating rules to format any column contained therein.

For example, by right clicking on the Key name column in the Registry hives tree, the following menu is shown.



There are options to format things on a variety of conditions, as seen below.



If we select the 'Text that contains...' option and enter 'Bags' along with how we want any matching rows to be formatted, the tree will reflect these changes. For example, if we entered the following conditions:



The Registry hives tree would then look like the screen shot below. The instances of 'Bags' in the highlighted rows have been circled for emphasis.

Key name	#...	Last write time...
D:\temp\re\UsrClassDelete\dBags.dat		2/1/2015 7:1...
S-1-5-21-146151751-63468248-1215037915-1000_Classes	0	9/19/2011 4:...
Local Settings	0	9/19/2011 4:...
MuiCache	0	9/19/2011 7:...
Software	0	9/19/2011 4:...
Microsoft	0	9/19/2011 4:...
Windows	0	9/19/2011 4:...
CurrentVersion	0	9/19/2011 4:...
Shell	0	9/19/2011 4:...
BagMRU	4	2/1/2015 7:1...
Bags	0	2/1/2015 7:1...
VirtualStore	0	9/19/2011 6:...

These formatting options allow you to create powerful visual indicators when data that is relevant to you is present in the hives you are looking at. Of course, all formatting options are remembered.

Use the same conditional formatting menu to edit rules.

## Loading hives

To load hives into Registry Explorer, either select one or more hives and drag/drop them onto the main interface. You can also use File | Load offline hive or press Alt+1 to select hives.

Registry Explorer will load the hives in parallel and as such, smaller hives will show up in the interface before larger ones. When loading more than one hive at a time, check the status bar at the bottom of the main window to see how many more hives are being processed.

After selecting a hive, Registry Explorer will fully process the hive. Once that is done, the hive will be displayed on the main interface. The top-level node for a hive is the full file path to the hive as seen below. The hive node has a green icon and is also in bold to differentiate it from keys.

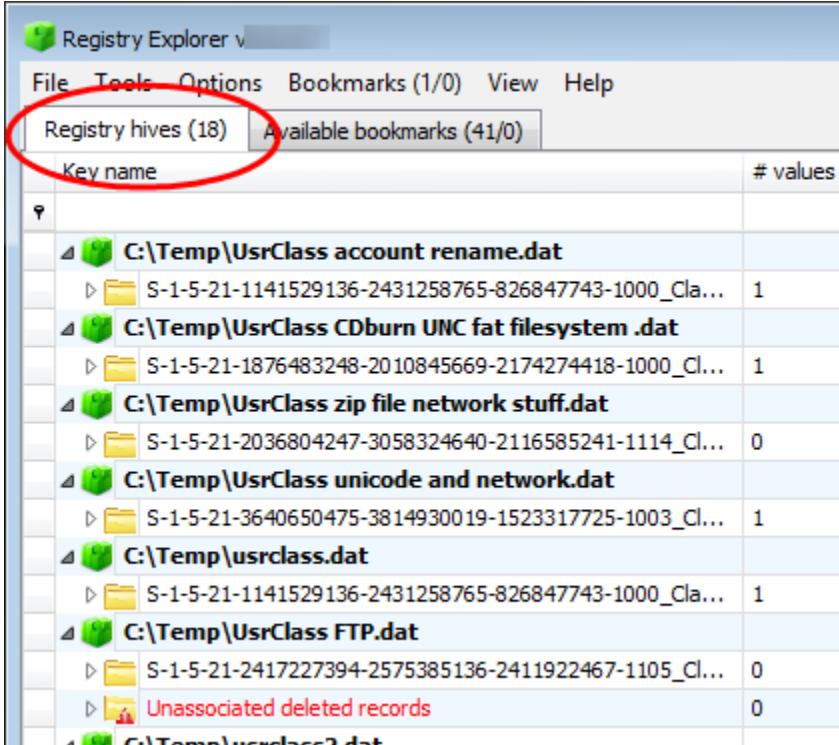
Key name	# values	Last write timestamp
D:\temp\re\UsrClassDeletedBags.dat		2/1/2015 7:15:49 PM +0...
S-1-5-21-1141529136-2431258765-826847743-1000_Classes	0	9/19/2011 4:30:48 PM +...
Local Settings	0	9/19/2011 4:31:27 PM +...
VirtualStore	0	9/19/2011 6:58:04 PM +...
Associated deleted records	0	

The last write timestamp for the hive is the timestamp value from the header of the hive.

Below the hive name is the root key for the hive. The root key name can vary for every hive that is loaded. All other keys in the active (that is, not deleted) portion of the Registry will be displayed under the root key.

If the option to recover deleted records is enabled, up to two different virtual keys may be created: one for Associated deleted records and one for unassociated deleted records. These virtual keys will not be shown if there aren't any deleted records of that type available. As discussed above, these keys can also be hidden using the relevant option under the Options menu.

The number after Registry hives in parenthesis is the total number of hives loaded. In the example below, there are 18 hives loaded.



Key name	# values
C:\Temp\UsrClass account rename.dat	
S-1-5-21-1141529136-2431258765-826847743-1000_Cla...	1
C:\Temp\UsrClass CDburn UNC fat filesystem .dat	
S-1-5-21-1876483248-2010845669-2174274418-1000_Cl...	1
C:\Temp\UsrClass zip file network stuff.dat	
S-1-5-21-2036804247-3058324640-2116585241-1114_Cl...	0
C:\Temp\UsrClass unicode and network.dat	
S-1-5-21-3640650475-3814930019-1523317725-1003_Cl...	1
C:\Temp\usrclass.dat	
S-1-5-21-1141529136-2431258765-826847743-1000_Cla...	1
C:\Temp\UsrClass FTP.dat	
S-1-5-21-2417227394-2575385136-2411922467-1105_Cl...	0
Unassociated deleted records	0
C:\Temp\usrclass2.dat	

## Dirty hives

In some cases, Registry Explorer will inform you that a hive is dirty when it is loaded. This means the hive's header primary and secondary sequence numbers do not match. More importantly, there is uncommitted data in one or more transaction logs that need to be applied to the original hive to ensure the most recent data is presented.

The transaction logs are found in the same directory as the hive itself and end with .LOG1 and .LOG2. Both are needed to accurately replay the transactions contained therein. Registry Explorer will determine which logs to apply and the order to apply them.

Once the logs have been replayed, Registry Explorer will offer to save out the updated hive to a new location. The new hive's header sequence numbers will also be updated to match. This hive (along with the dirty hive, optionally) can then be loaded into Registry Explorer for review.

## Projects

Projects allow you to load one or more hives into Registry Explorer and save the currently loaded hives into a project file. This allows you quickly load the same hives for a particular case quickly vs having to load a bunch of hives individually. You can also drag and drop Registry Explorer project files (.re\_proj) just like you would a registry hive.

## Selecting keys

Selecting keys in Registry Explorer works much the same as it does in regedit or selecting directories in Windows Explorer. Clicking the small arrow to the left of the key name or double clicking a key will expand that key, displaying any subkeys that are present. If the arrow is not visible, the key does not have any subkeys.

Keys can be double clicked and expanded, drilling down into the key hierarchy, until the key you are interested in is located. Alternatively, you can simply start typing a key's name and the keys will be dynamically expanded as matching keys are found in the tree.

For example, assume Registry Explorer looks like this:

Key name	# values	Last write timestamp
D:\temp\re\UsrClassDeletedBags.dat		2/1/2015 7:15:49 PM +0...
S-1-5-21-146151751-63468248-1215037915-1000_Classes	0	9/19/2011 4:30:48 PM +...
Local Settings	0	9/19/2011 4:31:27 PM +...
MuiCache	0	9/19/2011 7:02:08 PM +...
Software	0	9/19/2011 4:31:27 PM +...
VirtualStore	0	9/19/2011 6:58:04 PM +...
Associated deleted records	0	
D:\temp\re\5.dat		9/23/2013 7:17:31 PM +...
S-1-5-21-718126207-1171771683-1750804747-1001_Classes	1	8/1/2013 7:21:56 PM +0...
Associated deleted records	0	
Unassociated deleted records	0	
D:\temp\re\4.dat		5/20/2014 2:19:35 PM +...
S-1-5-21-2417227394-2575385136-2411922467-1105_Classes	0	1/27/2015 4:47:10 AM +...
D:\temp\re\6.dat		4/24/2014 3:02:54 PM +...
S-1-5-21-2208335738-3127931778-3832183526-1002_Classes	2	8/23/2014 3:20:25 AM +...
Associated deleted records	0	

If you want to look at the contents of the BagMRU key, click on either the hive path or the root key, then start typing BagMRU. As each letter is typed, Registry Explorer will search for matching keys and select them. After a few keystrokes, the following key is selected.

Key name	#...	Last write tim...
D:\temp\re\UsrClassDeletedBags.dat		2/1/2015 7:1...
S-1-5-21-146151751-63468248-1215037915-1000_Classes	0	9/19/2011 4:...
Local Settings	0	9/19/2011 4:...
MuiCache	0	9/19/2011 7:...
Software	0	9/19/2011 4:...
Microsoft	0	9/19/2011 4:...
Windows	0	9/19/2011 4:...
CurrentVersion	0	9/19/2011 4:...
Shell	0	9/19/2011 4:...
BagMRU	4	2/1/2015 7:1...
Bags	0	2/1/2015 7:1...
VirtualStore	0	9/19/2011 6:...

Notice also the part of the key that matched what was typed is highlighted. While a bookmark can be used to quickly jump to a particular key, using this technique can save a lot of time when you know the name of the key you are interested in.

## Filtering keys

The top of the Registry hives tree contains areas to enter text to filter that column. One thing to note is that only expanded keys are included in the filter results. To filter against all keys in a hive, use the context menu option to expand all subkeys (or press Alt+Down) before filtering. The next section will cover the context menu in detail.

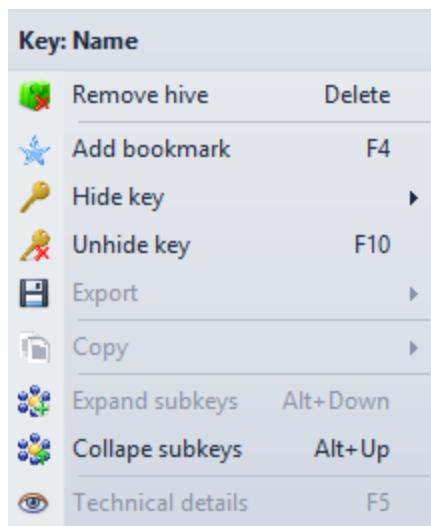
The Options | Show parent keys when filtering option affects what is shown when filtering keys. See the Options section for a full discussion on how this option works.

While it may seem that filtering is the quickest way to find a certain key, it is quite often faster to type the name of a key you are interested in (or better yet, using Tools | Find if it exists in more than one place).

## Key context menu

As in other places, the context menu changes dynamically depending on what you right click on. For example, if you right click on a hive's full path, you will see the option to remove the hive from Registry Explorer. Right clicking anywhere else but the hive's path will hide this option from view. Similarly, if a key is hidden, an option to unhide the key will be shown, else it will be hidden, and so on.

The key context menu looks like this:



The name of the currently selected key is shown at the top. Most options also have shortcuts which can be used in lieu of using the mouse.

- Remove hive: Removes the loaded hive from Registry Explorer. This option is only shown when a hive is selected (denoted by the full path to the hive name, shown in bold, and with a different icon).
- Add bookmark: Creates a new user bookmark. Full details will be discussed below.

- Hide key
  - For this session only: Hides keys matching the selected key's path from all loaded hives until Registry Explorer is restarted
  - Hide and add to auto hide: Same as the above option, except the key's path is remembered between restarts of Registry Explorer. This option is useful to hide non-useful keys in the Registry that get in your way.
- Unhide key: Unhide previously hidden keys with the same path as the selected key. If a key has been auto hidden, this option will remove it from the auto hide list.
- Export
  - To .reg format: Exports the selected key and its values to plaintext format. This file can then be imported into the active Registry by double clicking on the generated file.
  - To .reg format recursively: The same as above, except all keys and values for the selected key and all subkeys are exported.
- Copy

oKey name: Copies the selected key's key name to the clipboard. Double clicking the key path in the status bar while holding Shift also copies the key name to the clipboard.

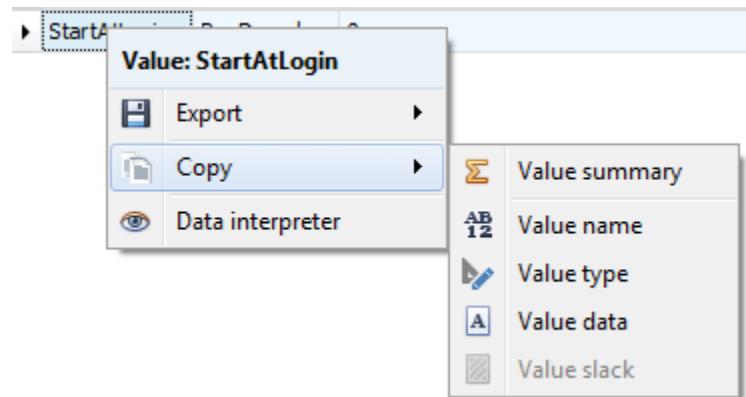
\* Key path: Copies the selected key's key path to the clipboard. Double clicking the key path in the status bar also copies the key path to the clipboard.

\* Last write time: Copies the selected key's last write timestamp to the clipboard. Double clicking the last write timestamp in the status bar also copies the last write timestamp to the clipboard. Double clicking the status bar while holding Shift will copy the key name and last write timestamp to the clipboard.

- Expand subkeys: Recursively expands the selected key and all subkeys. You can also hold the CTRL key while right clicking a node to expand each key.
- Collapse subkeys: Collapse all subkeys below the selected key
- Technical details: Displays full technical details about the selected key, its subkeys, values, security records, and hive header. This option will be fully explored below.

## Value context menu

A typical value context menu may look like this:



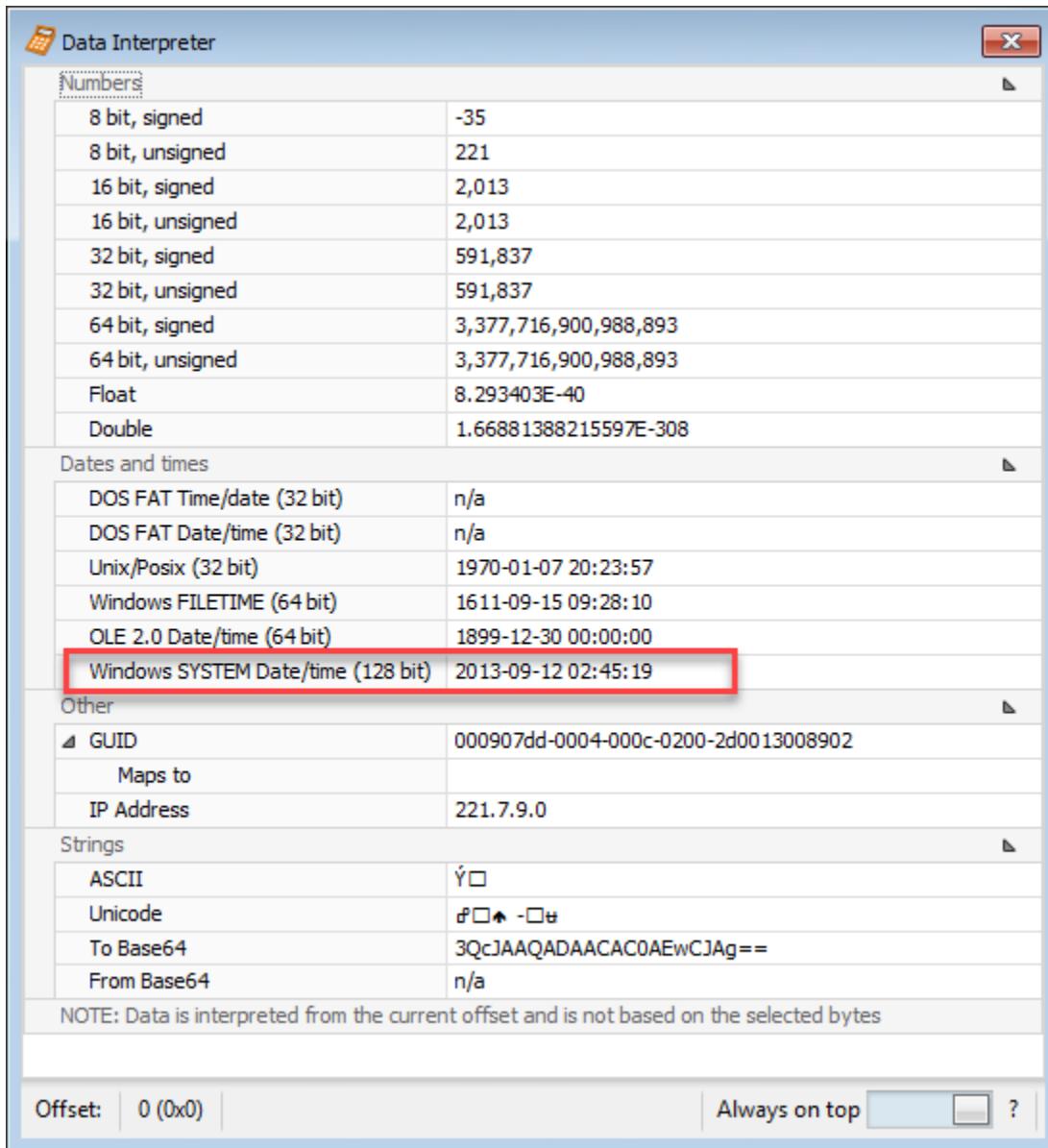
The name of the currently selected value is shown at the top.

- Export
  - Value data: Exports selected value's data in binary form to a file
  - Value slack: Exports selected value's slack data in binary form to a file. If a value has no slack, this option is disabled
- Copy
  - Value summary: Copies a summary of the selected value to the clipboard. An example is shown below.

```
.....10.....20.....30.....40.....50.....  
1 Registry file: D:\Sync\RegistryHives\1UsrClass.dat  
2 Key: Local Settings\MuiCache\24\52C64B7E  
3 Last write: 2014-09-11 21:24:22  
4 Value: @C:\Windows\system32\OobeFldr.dll, -33056 (RegSz)  
5 Data: Getting Started  
6 Slack: 69-00-6E-00  
7
```

- \* Value name: Copies the selected value's name to the clipboard
- \* Value type: Copies the selected value's type to the clipboard (RegBinary or RegSz for example)
- \* Value data: Copies the selected value's value data to the clipboard. For RegBinary values, the hex values, separated by a hyphen, are copied to the clipboard as a string
- \* Value slack: Copies the selected value's value slack to the clipboard. This option formats the data the same as the Value data option. If a value has no slack, this option is disabled.

- Data interpreter: Brings up the Data interpreter window for the currently selected value and converts the value's raw data to a variety of formats. The image below is shows how binary data for a 128-bit timestamp gets converted to different formats.

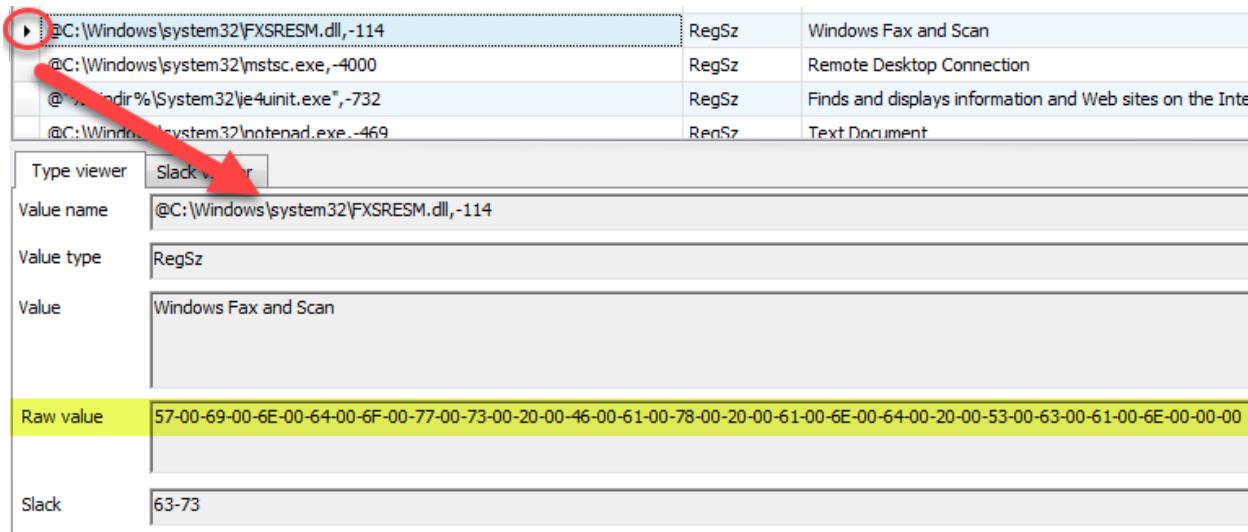


## Value details

The Value details area will change depending on the type of value selected.

## Type viewer

For all values except RegBinary values, a simple string representation of the value is shown as seen below.



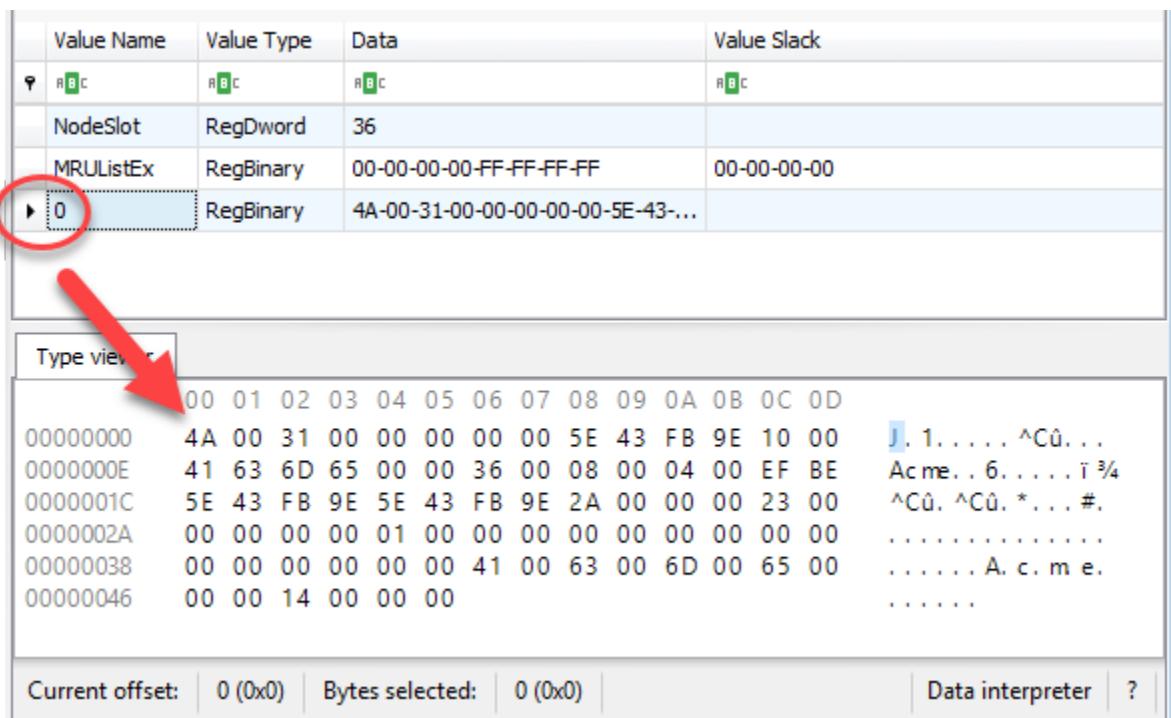
► @C:\Windows\system32\FXSRESM.dll,-114	RegSz	Windows Fax and Scan
@C:\Windows\system32\mstsc.exe,-4000	RegSz	Remote Desktop Connection
@%windir%\System32\ie4uinit.exe,-732	RegSz	Finds and displays information and Web sites on the Internet
@C:\Windows\system32\notepad.exe,-469	RegSz	Text Document

Type viewer	Slack
Value name	@C:\Windows\system32\FXSRESM.dll,-114
Value type	RegSz
Value	Windows Fax and Scan
Raw value	57-00-69-00-6E-00-64-00-6F-00-77-00-73-00-20-00-46-00-61-00-78-00-20-00-61-00-6E-00-64-00-20-00-53-00-63-00-61-00-6E-00-00-00
Slack	63-73

The other thing to notice here is the raw value is also shown (highlighted in yellow above). This allows you to export out raw data into other tools, etc.

For RegBinary keys, a hex viewer will be shown to display the value's binary data.



Value Name	Value Type	Data	Value Slack
NodeSlot	RegDword	36	
MRUListEx	RegBinary	00-00-00-00-FF-FF-FF-FF	00-00-00-00
► 0	RegBinary	4A-00-31-00-00-00-00-00-5E-43-...	

Type viewer
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 00000000 4A 00 31 00 00 00 00 00 5E 43 FB 9E 10 00 J. 1..... ^Cû... 0000000E 41 63 6D 65 00 00 36 00 08 00 04 00 EF BE Acme.. 6..... î ¾ 0000001C 5E 43 FB 9E 5E 43 FB 9E 2A 00 00 00 23 00 ^Cû. ^Cû. *... #. 0000002A 00 00 00 00 01 00 00 00 00 00 00 00 00 00 ..... 00000038 00 00 00 00 00 00 41 00 63 00 6D 00 65 00 ..... A. c. m e. 00000046 00 00 14 00 00 00 .....  Current offset: 0 (0x0) Bytes selected: 0 (0x0) Data interpreter ?

Selecting a byte or a range of bytes will update the Current offset and Bytes selected values at the bottom of the hex viewer.

Value Name	Value Type	Data	Value Slack
RBC	RegDword	00000000	00000000
NodeSlot	RegDword	36	
MRUListEx	RegBinary	00-00-00-FF-FF-FF-FF	00-00-00-00
0	RegBinary	4A-00-31-00-00-00-00-5E-43-...	

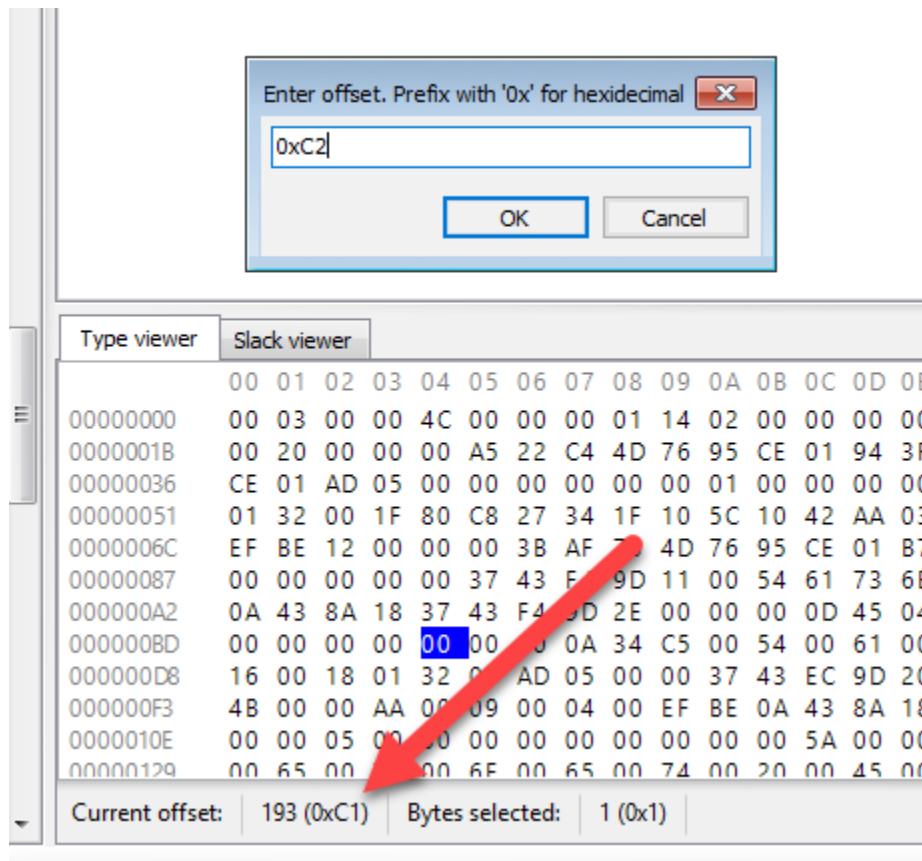
Type viewer
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 00000000 4A 00 31 00 00 00 00 00 5E 43 FB 9E 10 00 J. 1..... ^Cû... 0000000E 41 63 6D 65 00 00 36 00 08 00 04 00 EF BE Acme.. 6..... ï ¾ 0000001C 5E 43 FB 9E 5E 43 FB 9E 2A 00 00 00 00 23 00 ^Cû. ^Cû. *.... #. 0000002A 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 ..... 00000038 00 00 00 00 00 00 41 00 63 00 6D 00 65 00 ..... A. c. m. e. 00000046 00 00 14 00 00 00 ..... 
Current offset: 62 (0x3E) Bytes selected: 8 (0x8) Data interpreter ?

## Slack viewer

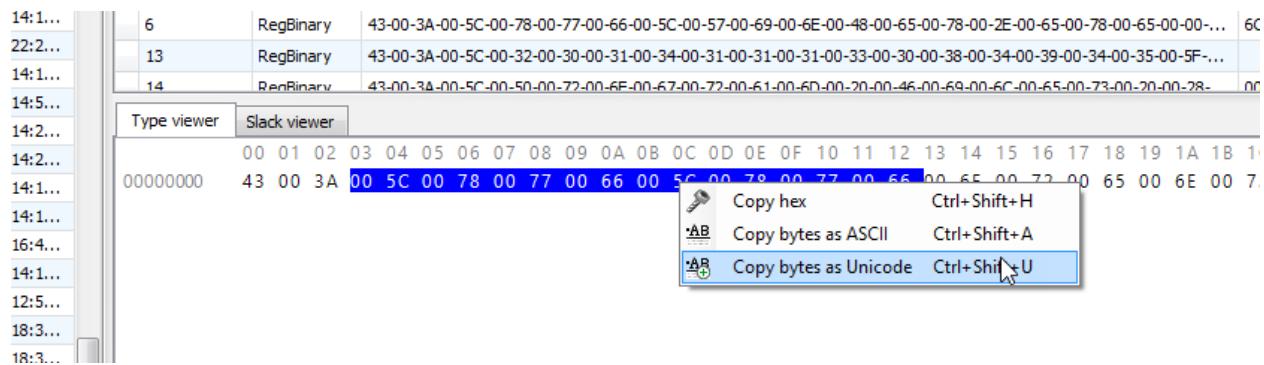
For values that have value slack, a Slack viewer tab will be added. This viewer works the same as the Type viewer for RegBinary values.

Type viewer	Slack viewer
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 00000000 5C 57 69 6E 64 6F ..... \W ndo	
Current offset: 0 (0x0) Bytes selected: 0 (0x0) Data interpreter ?	

Double clicking on the offset allows for entering an offset to jump to in the hex display.



When viewing binary data, you can copy the selected bytes to the clipboard as either hex, ASCII, or Unicode via the context menu:



## Data interpreter

In the lower right corner of the hex viewer is a Data interpreter button. Clicking this button will bring up the Data interpreter that converts the raw hexadecimal data into a variety of formats including dates and times, GUIDs, IP addresses, and more. The Data interpreter window is shown below.

In the example above, a RegBinary value is selected and the 14th byte has been selected (click on a

byte to select it). To the right of the hex display is an ASCII interpretation of the binary data. In this case, 70 corresponds to the 'p' character.

The Data interpreter also shows the same offset, 14, but it goes a step further and decodes the ASCII string 'please' from bytes 70 6C 65 61 73 65. Registry Explorer will look for a single Null terminator for ASCII strings (00) and double Null terminators (00 00) for Unicode strings. If no Null terminators are found, the bytes will be interpreted from the current offset to the end of the data.

The Data interpreter can also convert GUIDs to known folder/location names as seen below.

The screenshot displays the Data Interpreter interface with two main panes. The left pane contains various conversion tables and a 'Strings' section. The 'Strings' section has four entries: ASCII (please), Unicode (please), To Base64 (cGxLYXNIAAA6AAgABADvvitFHLErRQA4KgAAAQDCQQAAAA...), and From Base64 (n/a). A red circle highlights the 'ASCII' entry. Below this, a note states: 'NOTE: Data is interpreted from the current offset and is not based on the selected bytes'. The bottom of the left pane shows an 'Offset' field set to '14 (0xE)' with a red arrow pointing to it. The right pane shows a 'Value Slack' table with entries: 5C-57-69-6E-64-6F and 65-00. The bottom right corner of the right pane also has a red circle.

**Data Interpreter**

**Numbers**

8 bit, signed	112
8 bit, unsigned	112
16 bit, signed	27,760
16 bit, unsigned	27,760
32 bit, signed	1,634,036,848
32 bit, unsigned	1,634,036,848
64 bit, signed	111,546,229,681,264
64 bit, unsigned	111,546,229,681,264
Float	2.645074E+20
Double	5.51111600086297E-310

**Dates and times**

DOS FAT Time/date (32 bit)	2028-11-05 13:35:32
DOS FAT Date/time (32 bit)	2034-03-16 12:11:10
Unix/Posix (32 bit)	2021-10-12 11:07:28
Windows FILETIME (64 bit)	1601-05-10 02:30:22
OLE 2.0 Date/time (64 bit)	1899-12-30 00:00:00
Windows SYSTEM Date/time (128 bit)	n/a

**Other**

GUID	61656c70-6573-0000-3a00-08000400efbe
Maps to	
IP Address	112.108.101.97

**Strings**

ASCII	please
Unicode	please
To Base64	cGxLYXNIAAA6AAgABADvvitFHLErRQA4KgAAAQDCQQAAAA...
From Base64	n/a

NOTE: Data is interpreted from the current offset and is not based on the selected bytes

Offset: 14 (0xE)

**Type viewer** **Slack viewer**

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D
00000000	50	00	31	60	00	00	00	2B	45	1C	B1	10	00	P.....+E.±..
000000E	70	6C	65	61	73	65	00	00	3A	00	08	00	04	please.....
0000001C	EF	BE	2B	45	1C	B1	28	45	00	38	2A	00	00	7.14E.±+E.8*...
0000002A	A0	C2	41	00	00	00	00	00	00	00	00	00	00	ÅA.....
00000038	00	00	00	00	00	00	00	70	00	6C	00	65	00	.....p.l.e.
00000041	61	00	73	00	65	00	00	00	16	00	00	00	00	a.s.e.....

Current offset: 14 (0xE) Bytes selected: 0 (0x0)

Data interpreter ?

In this case, a GUID was found at offset 0x04, 26ee0668-a00a-44d7-9371-beb064c98683, that maps to ‘Control panel’.

To copy values from the Data interpreter to the clipboard, press Ctrl+C.

### Interacting with deleted keys

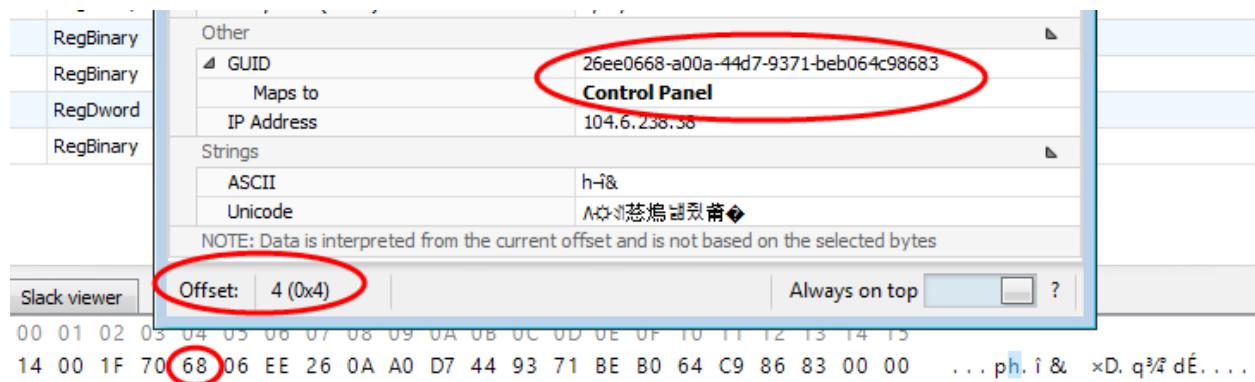
Registry Explorer can recover both deleted Registry keys and values. It also reassociates deleted values with their parent keys and subkeys to their parent keys.

In some cases, it is not possible to reassociate recovered keys to an active Registry key because the deleted key’s parent cell index does not correspond to a key’s offset in the active Registry.

Registry Explorer shows recovered deleted keys in up to three ways: “Inlined” with existing keys (that is, deleted keys are shown where they used to exist), Associated deleted records (the same info as inlined keys, but the parent keys are placeholders), and Unassociated deleted records (no parent key could be found in the active Registry).

### Inlined with existing keys

When Registry Explorer can reassociate a key with an active parent key, it is shown under the root key under its parent key. The icon for the deleted key (and all its subkeys) is the same as an active key, but a red X is shown in the lower right corner to denote it is a deleted key. The font for deleted keys is red.



### Associated deleted records

All associated deleted records are also shown under a virtual key called ‘Associated deleted’ records. Under this key, placeholder keys (keys with a link icon in the lower right) are created that denote active keys, down to the point where the deleted key can be found. In the example below, the same path as seen above is reflected down to the ‘BagMRU’ key. At this point, the icon and font color changes to indicate the key is in fact deleted and has been reassociated.

Key name	# values	Last write ...
C:\ProjectWorkingFolder\RegistryViewerZ\UsrC...	2	2/1/2015 ...
S-1-5-21-146151751-63468248-1215037915-1000_C...	0	9/19/2011...
Local Settings	0	9/19/2011...
MuiCache	0	9/19/2011...
Software	0	9/19/2011...
Microsoft	0	9/19/2011...
Windows	0	9/19/2011...
CurrentVersion	0	9/19/2011...
Shell	0	9/19/2011...
BagMRU	4	2/1/2015 ...
0	3	2/1/2015 ...
1	2	2/1/2015 ...
0	2	2/1/2015 ...
0	3	2/1/2015 ...
0	2	9/19/2011...
Doge	0	2/1/2015 ...
VirtualStore	0	9/19/2011...
Associated deleted records	0	

### Unassociated deleted records

In the cases where an active parent key could not be found, the recovered deleted key will be placed under another virtual key called 'Unassociated deleted records' that functions in a similar way to the Associated deleted records. The primary difference between the two is that there will not be any active parent keys shown for unassociated records. Unassociated records can be explored like any other records (looking at values, viewing Technical details, etc.).

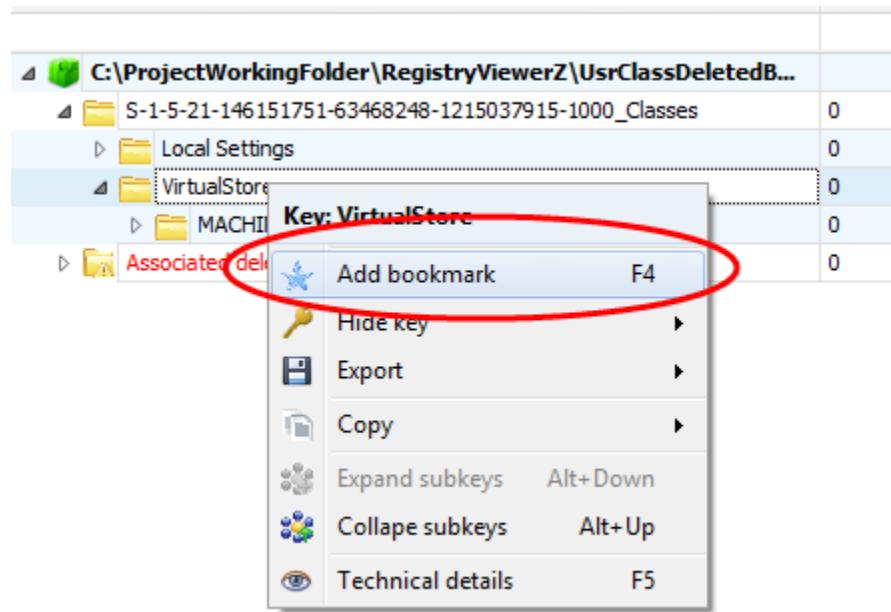
Key name	# values	Last write ...
C:\ProjectWorkingFolder\RegistryViewerZ\UsrC...		2/1/2015 ...
S-1-5-21-146151751-63468248-1215037915-1000_C...	0	9/19/2011...
Associated deleted records	0	
S-1-5-21-146151751-63468248-1215037915-1000...	0	
Local Settings	0	
Software	0	
Microsoft	0	
Windows	0	
Shell	0	
BadMRU	0	
1	2	2/1/2015 ...
0	2	2/1/2015 ...
0	3	2/1/2015 ...
0	2	9/19/2011...

## Creating bookmarks

To create a bookmark, right click on a key and select Add bookmark.

C:\ProjectWorkingFolder\RegistryViewerZ\NTUSER.DAT	6	
CsITool-CreateHive-{00000000-0000-0000-0000-000000000000}	0	1
Associated deleted records	0	
Unassociated deleted records	0	
{1a5c7e00-a12e-4cb3-9cd2-30597f5f1d8e}	3	1
{1BC4FA57-AEBC-4152-BD7A-075EE0B96381}	2	1
{1C6A51C9-D07D-4e82-BD3E-0EB7F88AC004}	5	1
{1F0DA31F-1C61-4b96-B1CC-CBF2D3872353}	5	1
{1F411263-3A1D-43F5-96AF-F5648CB89186}	3	1
{1faf3db1-30ac-40ca-b115-5999e7daf938}	2	1
{1202F5B4-3522-4149-BAD8-58B2079D704F}	12	1
{22189D02-CCA4-40AE-A874-6C2A764FB071}	26	1
{2E36F1D4-B23C-435D-AB41-18E608940038}	34	1
IncompatibleList	6	1
PortSupplier	0	1
--	--	--

Since Registry Explorer knows the hive type and key path already, these values will be prepopulated. In the example below, a UsrClass hive is active and the VirtualStore key is selected.

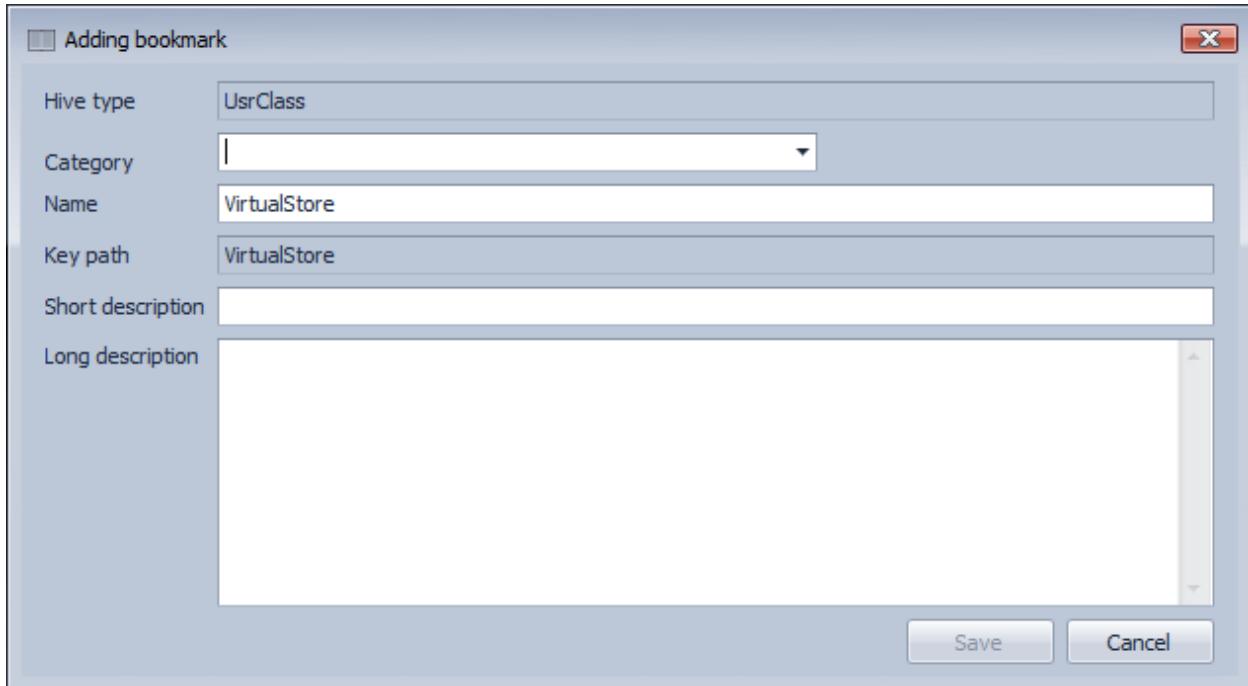


The Category field allows you to place this particular Registry key into a high-level group. This will eventually be used for reporting. Several preexisting categories are included but typing a new Category will add it to the list.

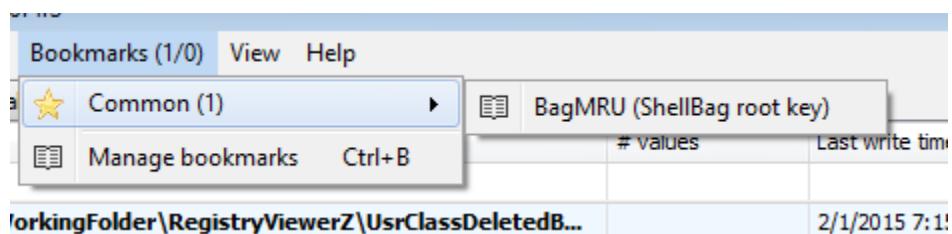
The Short description serves as a summary for what the key means or why it is relevant. The value entered for Short description will show up after the name of the bookmark in the bookmarks menu.

The Long description should contain technical information, links to web pages with more information, or any other information you want to convey.

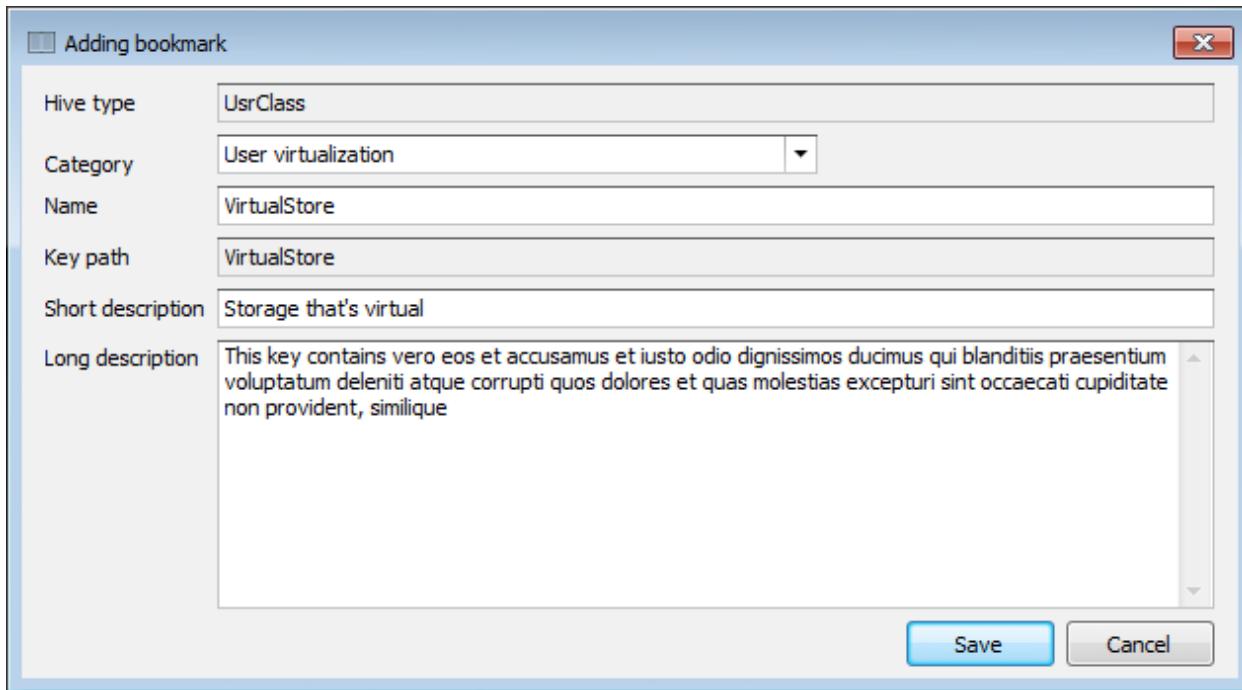
Recall the bookmarks menu is dynamic and will update according to the keys that are available for the selected hive. If, before adding the bookmark, the Bookmarks menu looked like this:



And our new bookmark looks like this:



The Bookmarks menu will look like this once the Save button is clicked:

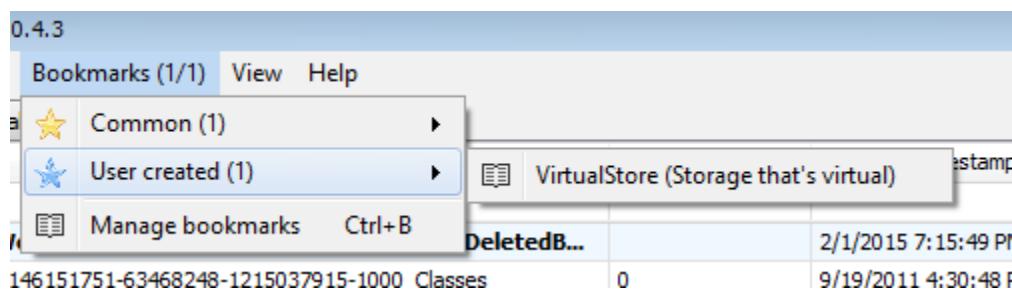


A ‘User created’ menu is now visible as is our VirtualStore bookmark (with the short description shown in parenthesis after the key name).

Selecting the ‘VirtualStore’ bookmark expands all child keys to the bookmarked key in the selected Registry hive.

## Managing bookmarks

Recall bookmarks are kept in two folders, one for included bookmarks and one for user created bookmarks. Registry Explorer contains a Bookmark manager that is available under the Bookmarks menu.

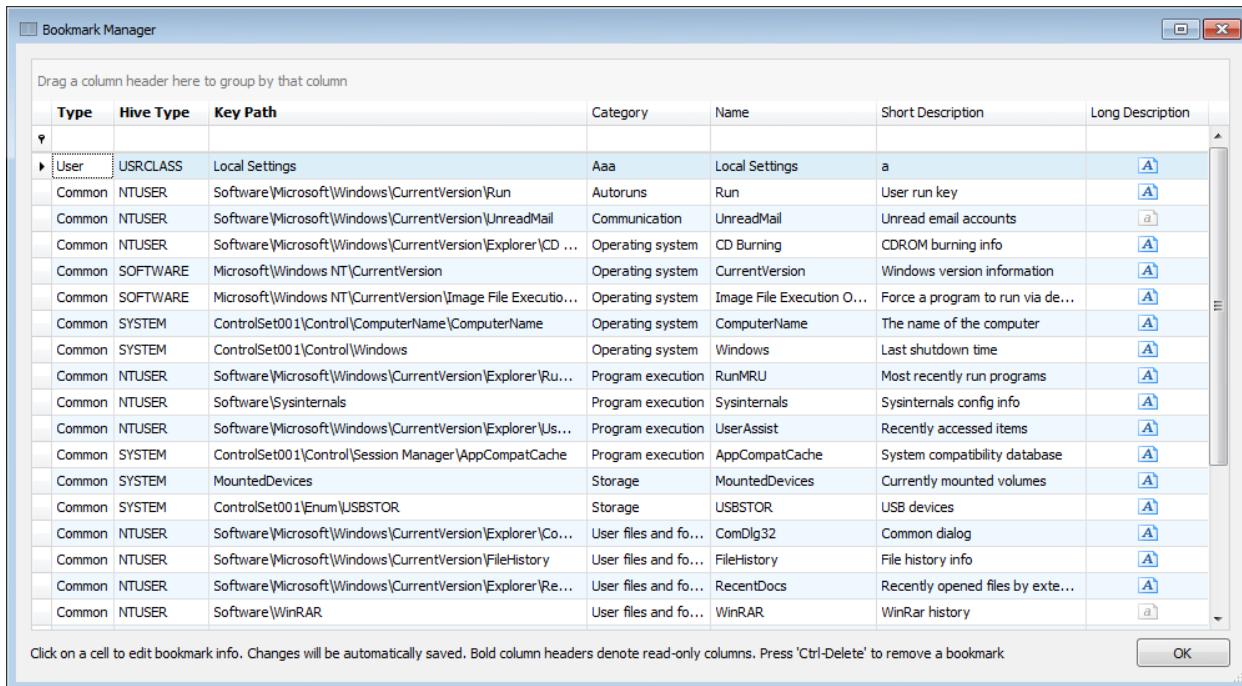


The column headers in bold (Type, Hive Type and Key Path) are read only. To edit any of the other columns, click on that column’s value and adjust. The bookmark is saved automatically, and the Bookmarks menu will be updated accordingly.

## Available bookmarks

The Available bookmarks tab is an optimized way to view all available bookmarks across all loaded hives. Using the Available bookmarks tab allows you to see all bookmarks that exist without the distraction of parent keys or having to drill down into different hives to review things.

After loading one or more hives, click on the Available bookmarks tab. An example of this is shown below.



The screenshot shows the 'Bookmark Manager' window with the title bar 'Bookmark Manager'. Below the title bar is a toolbar with standard window controls (Minimize, Maximize, Close). The main area is a grid-based table with the following columns: Type, Hive Type, Key Path, Category, Name, Short Description, and Long Description. A tooltip at the top of the table says 'Drag a column header here to group by that column'. The table contains approximately 25 rows of bookmark information. The first row, 'User \USRCLASS', is selected, indicated by a bolded 'User' in the 'Type' column. The 'Category' column shows various system categories like 'Local Settings', 'Autoruns', 'Communication', etc. The 'Name' column lists specific bookmarks such as 'Run', 'UnreadMail', 'CD Burning', etc. The 'Short Description' and 'Long Description' columns provide brief explanations for each bookmark. At the bottom of the table, a note says 'Click on a cell to edit bookmark info. Changes will be automatically saved. Bold column headers denote read-only columns. Press 'Ctrl+Delete' to remove a bookmark'. There is also an 'OK' button in the bottom right corner.

Type	Hive Type	Key Path	Category	Name	Short Description	Long Description
User	USRCLASS	Local Settings	Aaa	Local Settings	a	<a href="#">A</a>
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Run	Autoruns	Run	User run key	<a href="#">A</a>
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\UnreadMail	Communication	UnreadMail	Unread email accounts	<a href="#">a</a>
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Explorer\CD ...	Operating system	CD Burning	CDROM burning info	<a href="#">A</a>
Common	SOFTWARE	Microsoft\Windows NT\CurrentVersion	Operating system	CurrentVersion	Windows version information	<a href="#">A</a>
Common	SOFTWARE	Microsoft\Windows NT\CurrentVersion\Image File Executio...	Operating system	Image File Execution O...	Force a program to run via de...	<a href="#">A</a>
Common	SYSTEM	ControlSet001\Control\ComputerName\ComputerName	Operating system	ComputerName	The name of the computer	<a href="#">A</a>
Common	SYSTEM	ControlSet001\Control\Windows	Operating system	Windows	Last shutdown time	<a href="#">A</a>
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Explorer\Ru...	Program execution	RunMRU	Most recently run programs	<a href="#">A</a>
Common	NTUSER	Software\Sysinternals	Program execution	Sysinternals	Sysinternals config info	<a href="#">A</a>
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Explorer\Us...	Program execution	UserAssist	Recently accessed items	<a href="#">A</a>
Common	SYSTEM	ControlSet001\Control\Session Manager\AppCompatCache	Program execution	AppCompatCache	System compatibility database	<a href="#">A</a>
Common	SYSTEM	MountedDevices	Storage	MountedDevices	Currently mounted volumes	<a href="#">A</a>
Common	SYSTEM	ControlSet001\Enum\USBSTOR	Storage	USBSTOR	USB devices	<a href="#">A</a>
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Explorer\Co...	User files and fo...	ComDlg32	Common dialog	<a href="#">A</a>
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\FileHistory	User files and fo...	FileHistory	File history info	<a href="#">A</a>
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Explorer\Re...	User files and fo...	RecentDocs	Recently opened files by exte...	<a href="#">A</a>
Common	NTUSER	Software\WinRAR	User files and fo...	WinRAR	WinRAR history	<a href="#">a</a>

Click on a cell to edit bookmark info. Changes will be automatically saved. Bold column headers denote read-only columns. Press 'Ctrl+Delete' to remove a bookmark

When the root folder for a bookmark is selected (BagMRU in the example above), information about the bookmark is shown at the bottom of the window in the Bookmark information section.

The numbers at the end of the Available bookmarks tab indicate the total number of common bookmarks (20 in this case) and the total number of user created bookmarks (5 in this case). Available bookmarks dynamically updates as hives are loaded/unloaded, bookmarks are created/removed, etc.

Right clicking on a key brings up a context menu. The options work the same way as on the Registry hives tab. The 'Jump to key' option will change the active tab to the 'Registry hives' tab and select the bookmarked key. This is useful to see the bookmarked key in context with other keys.

Registry hives Available bookmarks (20/5)

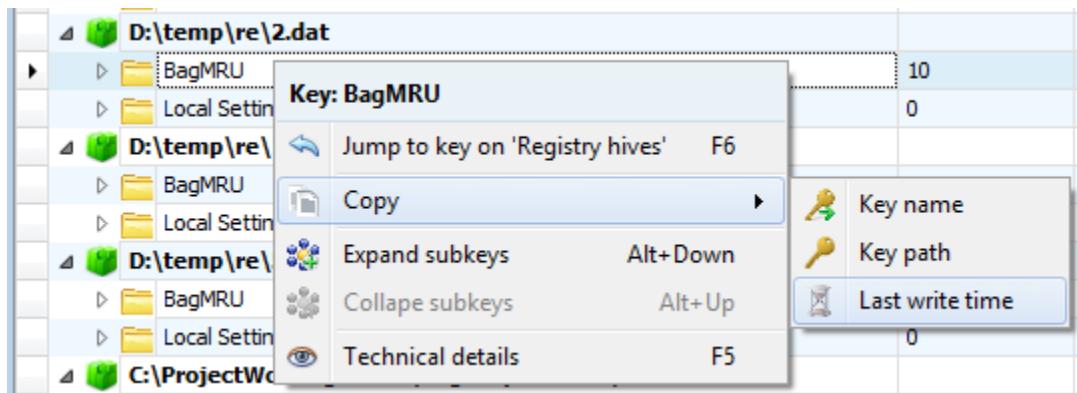
Key name	# values	Last write timestamp
D:\temp\re\3.dat		
BagMRU	14	11/26/2014 4:14:15 PM ...
Local Settings	0	5/20/2014 2:26:24 PM ...
D:\temp\re\4.dat		
BagMRU	30	1/30/2015 7:00:34 PM ...
Local Settings	0	5/20/2014 2:26:24 PM ...
D:\temp\re\2.dat		
BagMRU	10	9/23/2014 11:07:17 PM ...
Local Settings	0	8/1/2014 10:38:18 PM ...
D:\temp\re\1.dat		
BagMRU	156	10/25/2013 2:15:20 PM ...
Local Settings	0	10/23/2009 10:22:25 P... ...
D:\temp\re\5.dat		
BagMRU	17	10/23/2013 3:09:17 AM ...
Local Settings	0	9/23/2013 7:52:03 PM ...
C:\ProjectWorkingFolder\RegistryViewerZ\NTUSER.DAT		
Run	13	12/8/2014 1:19:24 PM ...
UnreadMail	0	7/29/2014 12:26:56 PM ...
CD Burning	2	11/28/2014 4:57:04 PM ...
RunMRU	0	5/20/2014 2:26:30 PM ...
Sysinternals	0	5/29/2014 1:06:41 PM ...
UserAssist	0	5/20/2014 2:31:27 PM ...
ComDlg32	0	5/20/2014 3:21:50 PM ...
FileHistory	0	5/20/2014 2:19:35 PM ...
RecentDocs	150	12/8/2014 2:59:56 PM ...
WinRAR	0	9/12/2014 10:45:53 PM ...
Ares	19	8/26/2014 5:52:22 PM ...
Default	5	11/29/2014 6:06:33 PM ...
FTP	2	11/25/2014 4:52:19 PM ...

Bookmark information

Hive	D:\temp\re\3.dat
Category	User files and folders
Name	BagMRU
Key path	Local Settings\Software\Microsoft\Windows\Shell\BagMRU
Short description	ShellBag root key
Long description	ShellBags hold user activity related to accessing resources on a computer

Finally, common bookmarks are differentiated from user created bookmarks by showing user created

bookmarks in blue, as shown below:



This makes it easy to spot bookmarks you have added vs ones that were included with Registry Explorer.

## Searching

Registry Explorer contains powerful searching capabilities including standard string searches and regular expression-based searches. It can also search for keys where the last write timestamp is before, between or after a given timestamp or pair of timestamps, or for values that have a data size greater than a certain number of bytes.

Registry hives (1)		Available bookmarks (2/1)	
Key name	# values	# subkeys	Last write time
RBC	=	=	=
D:\Sync\RegistryHives\1UsrClass.dat			20
BagMRU	15	12	20
VirtualStore	0	1	20
MuiCache	0	1	20

Registry Explorer allows you to search all hives at once across key names, value names, value data and/or value slack. Searching is done against each hive asynchronously and results will appear as they are available.

## Options menu

Clear recent

When conducting a standard search, search terms in the ‘Search for’ box are remembered between program executions. Use this option to clear these recent searches.

Convert

The convert menu contains options to convert the selected search string in the ‘Search for’ box to its ASCII or Unicode hexadecimal value. This is useful when searching for patterns in Value data.

For example, selecting ‘Eric’ (without the quotes) and using the conversion options results in the following being shown in the ‘Search for’ box:

- ASCII: 45-72-69-63
- Unicode: 45-00-72-00-69-00-63-00

The converted value can now be used to search for the initial string in its encoded form. You can also convert terms to ROT-13 and search for encoded strings as well.

## Help menu

### Search tips

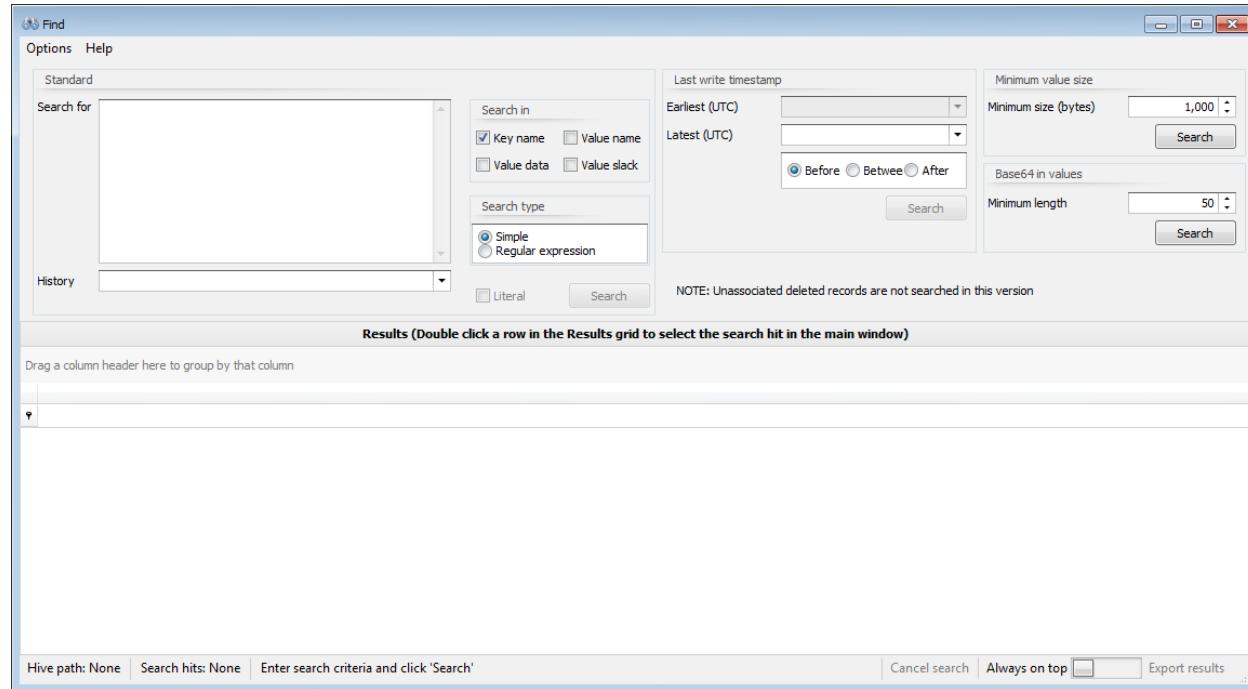
Shows several tips for different kinds of searches

### Regular expressions

Launches a web page with information about creating .net regular expressions

## Standard search

To conduct a standard search, simply enter one or more values in the ‘Search for’ box and click ‘Search.’ You can also press the Enter key twice on an empty line after entering a search term to perform the search.



The Literal checkbox controls whether the term searched for is looked for in binary data when searching in value data and/or slack. This is explained in more detail below.

If you entered a regular expression, change the radio button to ‘Regular expression’ so Registry Explorer knows to use RegEx when searching. The Help menu can be used to get additional help on building .net regular expressions. For additional resources on regular expressions, click here to view the regular expression searching section for RECmd.

The History drop down will contain a list of your recent searches

### **Last write timestamp search**

To conduct a last write timestamp search, choose the date range to search for via the radio buttons and enter the required time stamp values, and then click Search (or press Enter).

The screenshot shows the Registry Explorer interface. A red oval highlights the search bar and the search options. A red arrow points from the 'Search' button to the 'Results' section below. The results grid displays a list of registry keys found in the '1UsrClass.dat' hive.

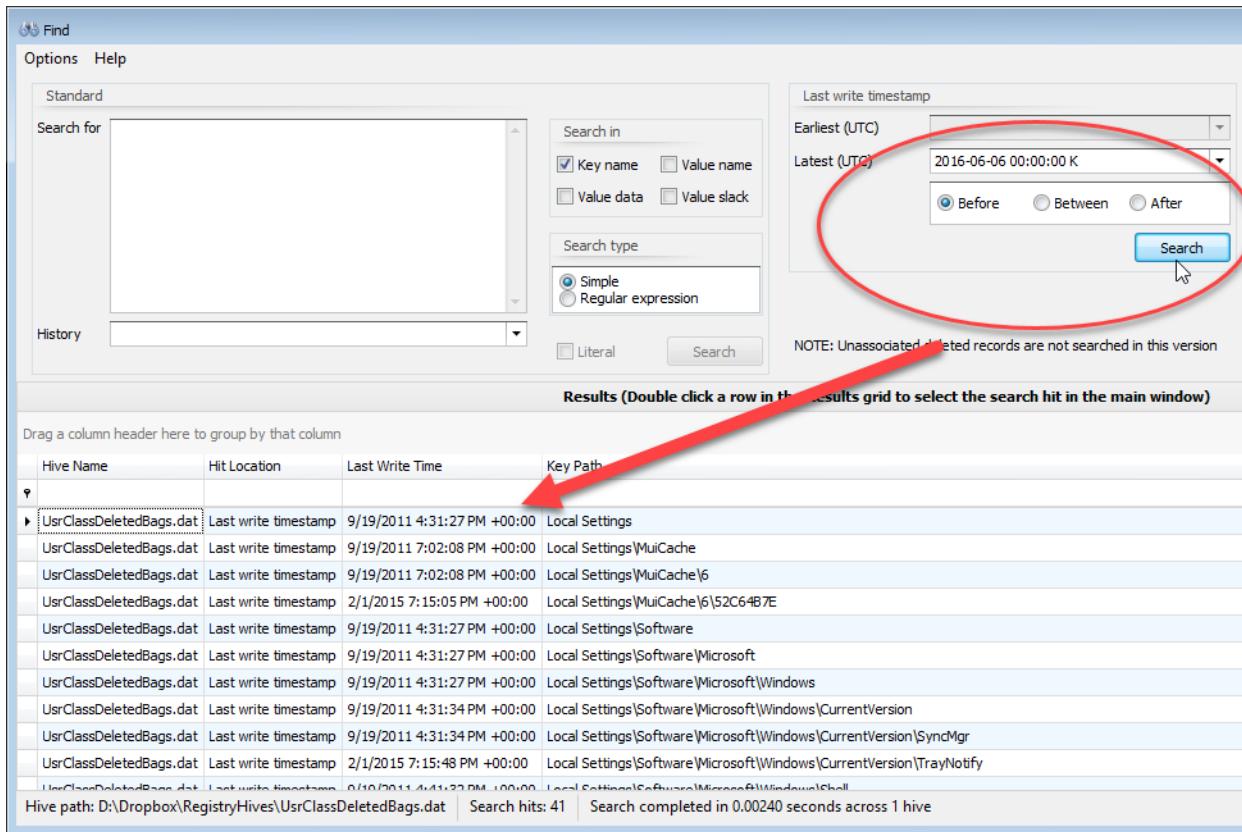
Hive Name	Hit Location	Last Write Time	Key Path
1UsrClass.dat	Key name	2014-03-21 15:14:00	Local Settings\MuiCache
1UsrClass.dat	Key name	2014-09-09 03:09:51	Local Settings\Software\Microsoft\Windows\Shell\MuiCache
1UsrClass.dat	Key name	2014-03-21 15:14:00	Local Settings\MuiCache
1UsrClass.dat	Key name	2014-09-09 03:09:51	Local Settings\Software\Microsoft\Windows\Shell\MuiCache
1UsrClass.dat	Value name	2014-09-11 20:11:48	Local Settings\Software\Microsoft\Windows\CurrentVersi...
1UsrClass.dat	Value name	2013-12-04 19:21:41	Local Settings\Software\Microsoft\Windows\Shell\Bags\...
1UsrClass.dat	Value name	2013-12-04 19:21:41	Local Settings\Software\Microsoft\Windows\Shell\Bags\...
1UsrClass.dat	Value data	2014-09-11 21:24:22	Local Settings\Software\Microsoft\Windows\Shell\BagMRU

Hive path: D:\Sync\RegistryHives\1UsrClass.dat | Search hits: 8 | Search completed in 0.00469 seconds across

Note: Depending on the Date/Time format under Preferences you may see extra characters in the earliest and latest time stamp fields. These can be ignored.

### Minimum value size search

When searching for values above a minimum size, the size of the value data's length is shown in the Value Data column as seen below.

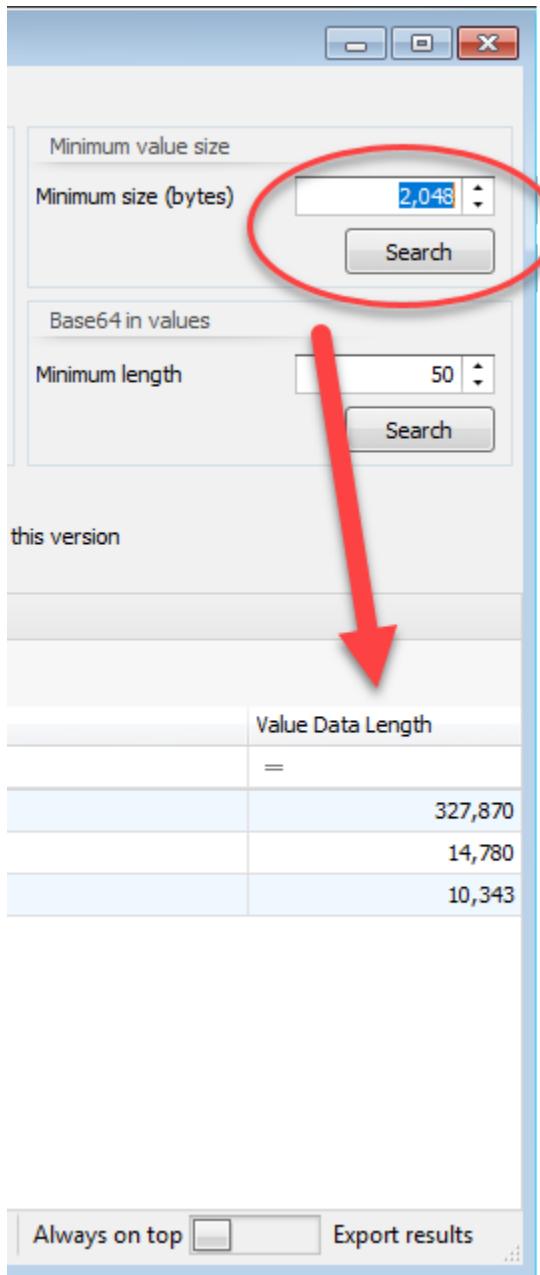


## Base64 in values search

This works the same as a minimum value size but validates the value's data contains a valid base64 encoded string.

## Interacting with search results

Once a search is underway, results will show up in the Results grid at the bottom of the Find window.



In the above example, a simple search was done for the string ‘mui’ and ‘cache’ which resulted in 334 hits. The search results contain the hive the hit was found in, what type of hit it was (key name, value name, etc.), the hit text, and other relevant information.

The columns shown in the Results grid will change depending on what kind of search was done. When searching in value name and/or value data, two additional columns will be shown as seen below.

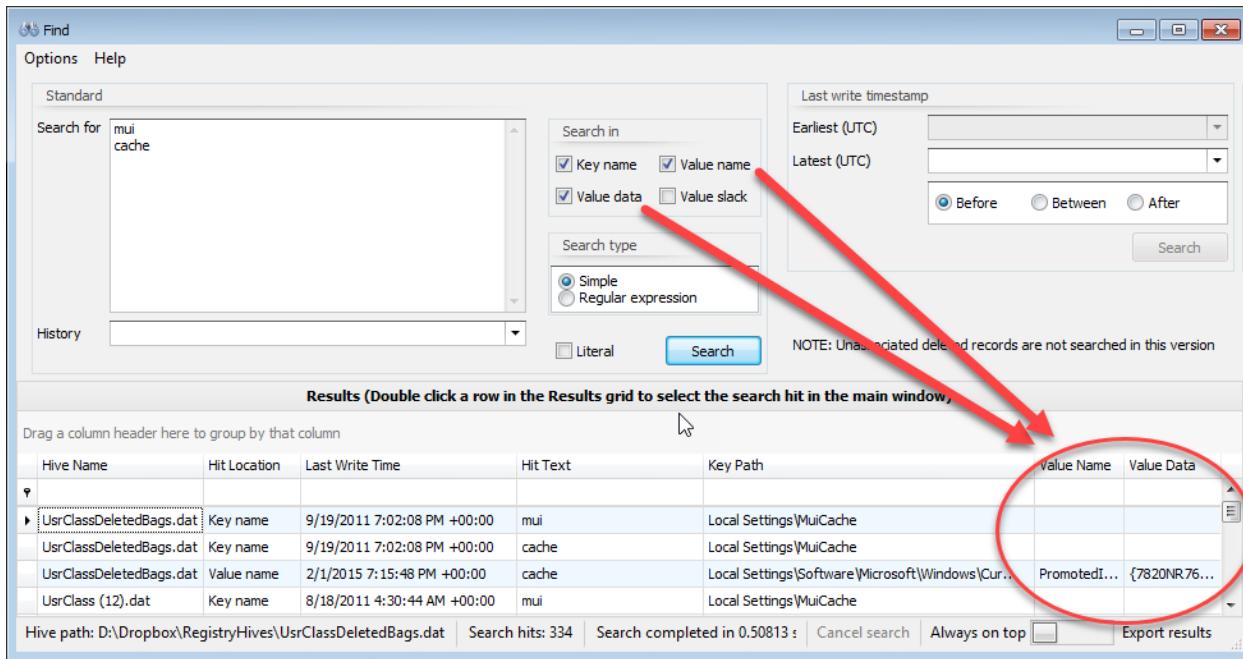
The screenshot shows the Registry Explorer interface with a search term 'mui cache' entered in the search bar. The search results grid displays 11 hits across two hives. The columns include Hive Name, Hit Location, Last Write Time, Key Path, Deleted, Value Name, Hit Text, and Value Data. The 'Hit Text' column shows the found strings 'mui' and 'cache' in green, indicating they were found in the raw value data or slack.

Hive Name	Hit Location	Last Write Time	Key Path	Deleted	Value Name	Hit Text	Value Data
UsrClassDeletedBags.dat	Key name	2011-09-19 19:02:08	Local Settings\MuiCache	<input type="checkbox"/>		mui	
UsrClassDeletedBags.dat	Key name	2011-09-19 19:02:08	Local Settings\MuiCache	<input type="checkbox"/>		cache	
UsrClassDeletedBags.dat	Value name	2015-02-01 19:15:48	Local Settings\Software\Microsoft\Windows\CurrentV...	<input type="checkbox"/>	Promoted...	cache	{7820NR76-23R3-4229-82P1-R41PO6...
1UsrClass.dat	Key name	2014-03-21 15:14:00	Local Settings\MuiCache	<input type="checkbox"/>		mui	
1UsrClass.dat	Key name	2014-09-09 03:09:51	Local Settings\Software\Microsoft\Windows\Shell\Mui...	<input type="checkbox"/>		mui	
1UsrClass.dat	Key name	2014-03-21 15:14:00	Local Settings\MuiCache	<input type="checkbox"/>		cache	
1UsrClass.dat	Key name	2014-09-09 03:09:51	Local Settings\Software\Microsoft\Windows\Shell\Mui...	<input type="checkbox"/>		cache	
1UsrClass.dat	Value name	2014-09-11 20:11:48	Local Settings\Software\Microsoft\Windows\CurrentV...	<input type="checkbox"/>	Promoted...	cache	{7820NR76-23R3-4229-82P1-R41PO6...
1UsrClass.dat	Value name	2013-12-04 19:21:41	Local Settings\Software\Microsoft\Windows\Shell\Ba...	<input type="checkbox"/>	CachedOf...	cache	0
1UsrClass.dat	Value name	2013-12-04 19:21:41	Local Settings\Software\Microsoft\Windows\Shell\Ba...	<input type="checkbox"/>	CachedOf...	cache	1817893

When searching in value data and/or value slack, the Search for term will be found regardless of case or encoding (Western 1252 and/or Unicode to be exact). This makes it easy to find strings that have been encoded in binary data.

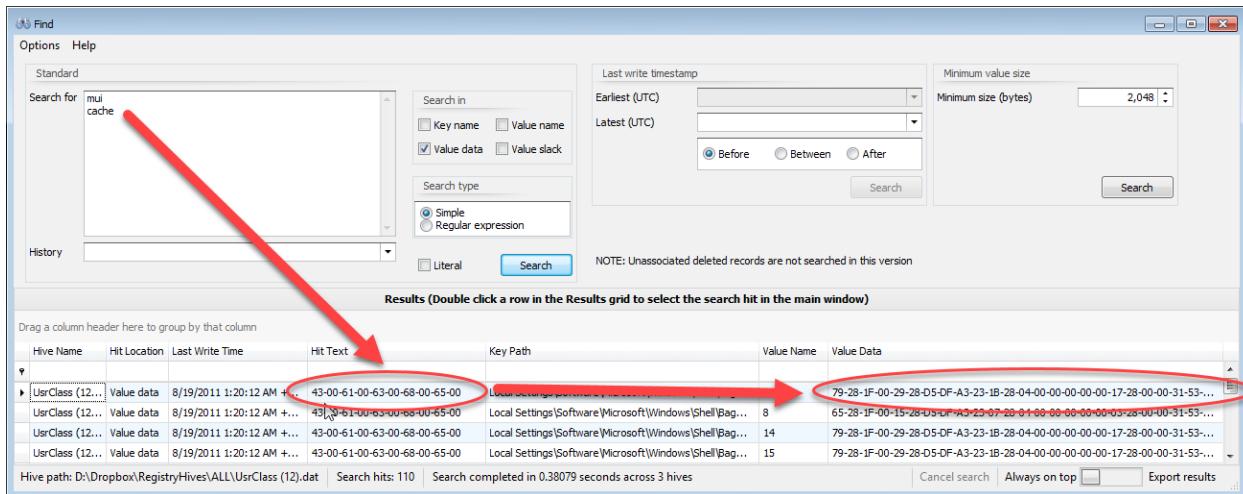
The way this works is to take the raw bytes that make up the value data and/or value slack and convert it to a string (again, in Western 1252 and Unicode), which is then searched using a regular expression. The regex will find the hit with exact capitalization, and the exact hit is then converted back to a byte string. This hit can then be reported back to the application and the data highlighted in context with the rest of the data, regardless of encoding or capitalization.

Here is an example of some search hits for 'cache' that were found in binary data:



If the Literal checkbox is checked, the additional search against the converted data is not done behind the scenes. This allows you to look for specific byte patterns without Registry Explorer converting binary data to strings.

Here is an example where the string 'las' was found in value slack:

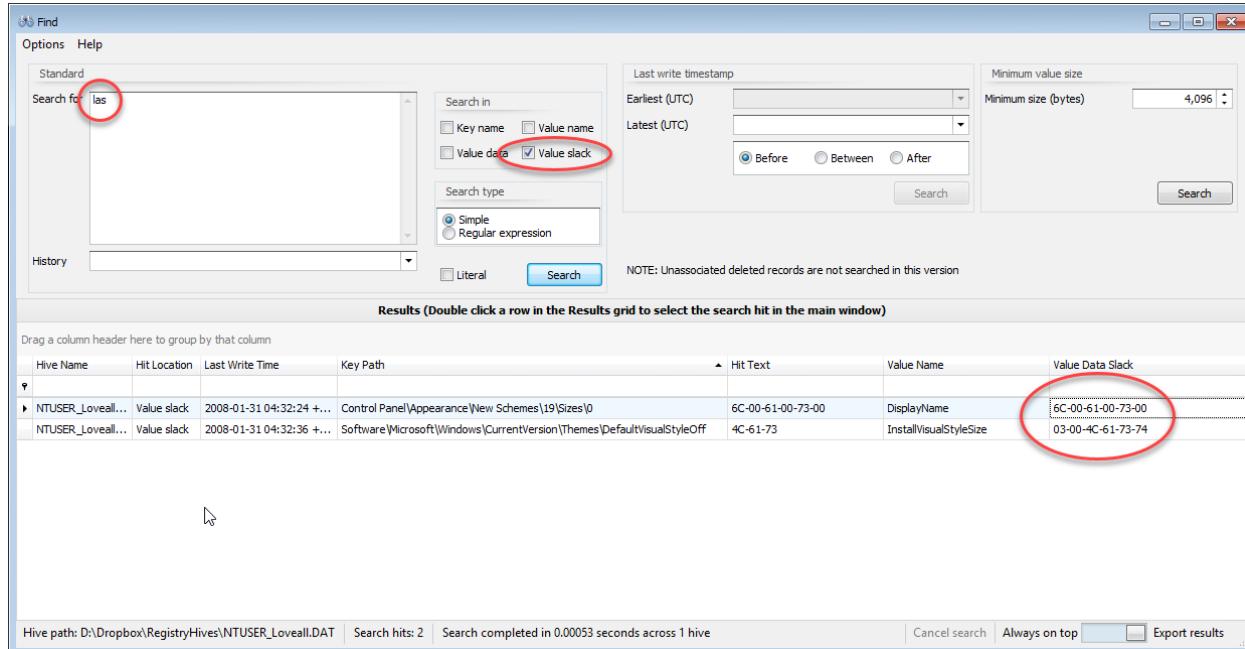


In the screen shot above, notice the hit in value slack was found in two different encodings.

## Viewing search results

To view the search hit in the main Registry Explorer window, simply double click on the result you wish to view. The Registry hive containing the hit will be selected along with the key where the hit

was found. If the Hit location is in a value name or in value data, the corresponding value will be selected under the key.



For all simple searches, the search hit will be highlighted (or, in the case of a RegBinary hit, the bytes that make up the hit will be selected). A few more examples of this are shown below.

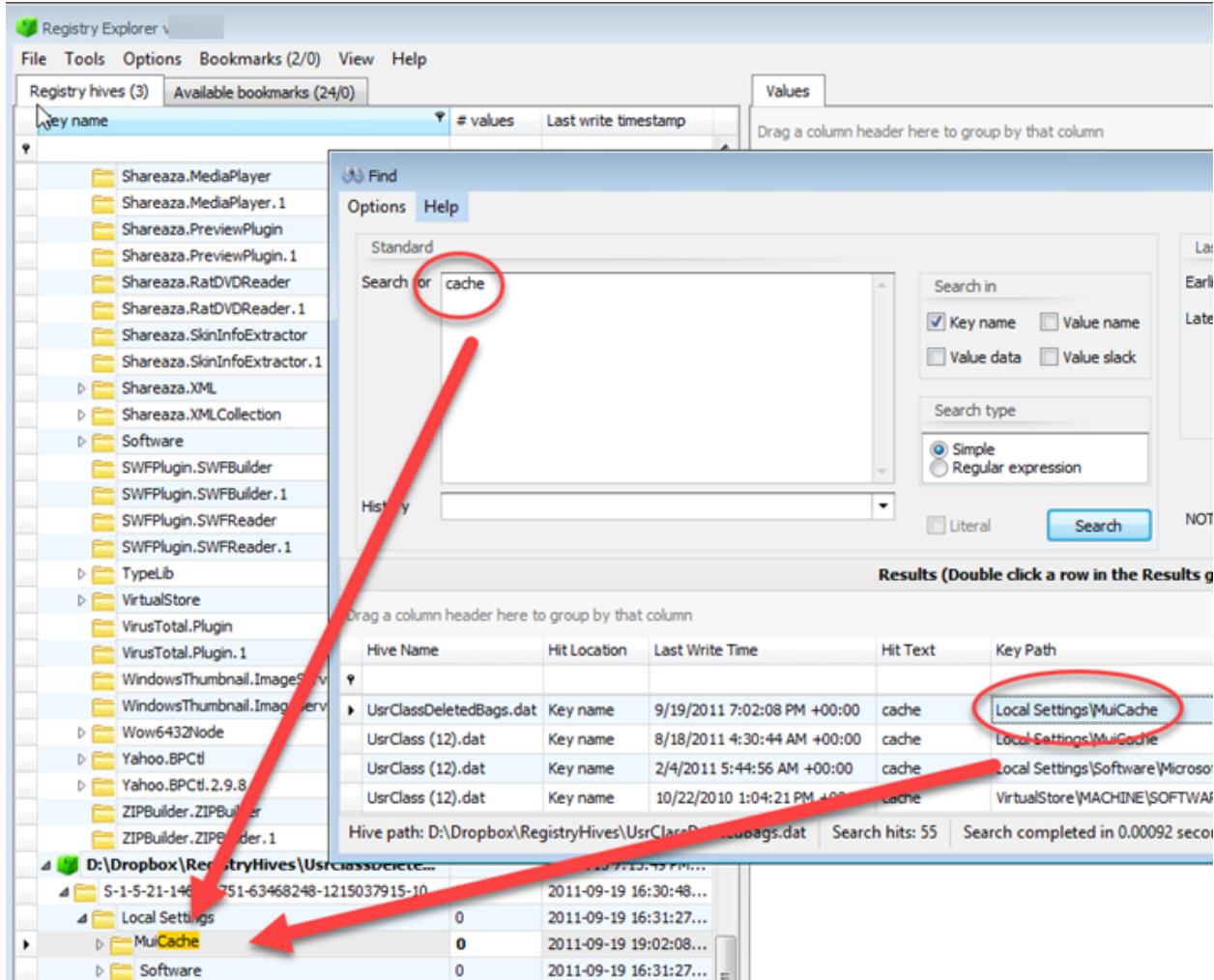
For key name hits, the matching part of the key name is highlighted.

The screenshot shows the Registry Explorer interface with a search dialog open. The search term 'cache' is entered in the 'Search for' field. The 'Key name' checkbox is checked. The search results grid shows several entries across three hives. One entry in the 'Software\Alexa Internet\Alexa9\Amazon\Cache' key path has its 'Key Path' column highlighted with a red oval, and the entire row is selected. A red arrow points from the search term 'cache' in the search dialog to the highlighted 'Key Path' in the results grid.

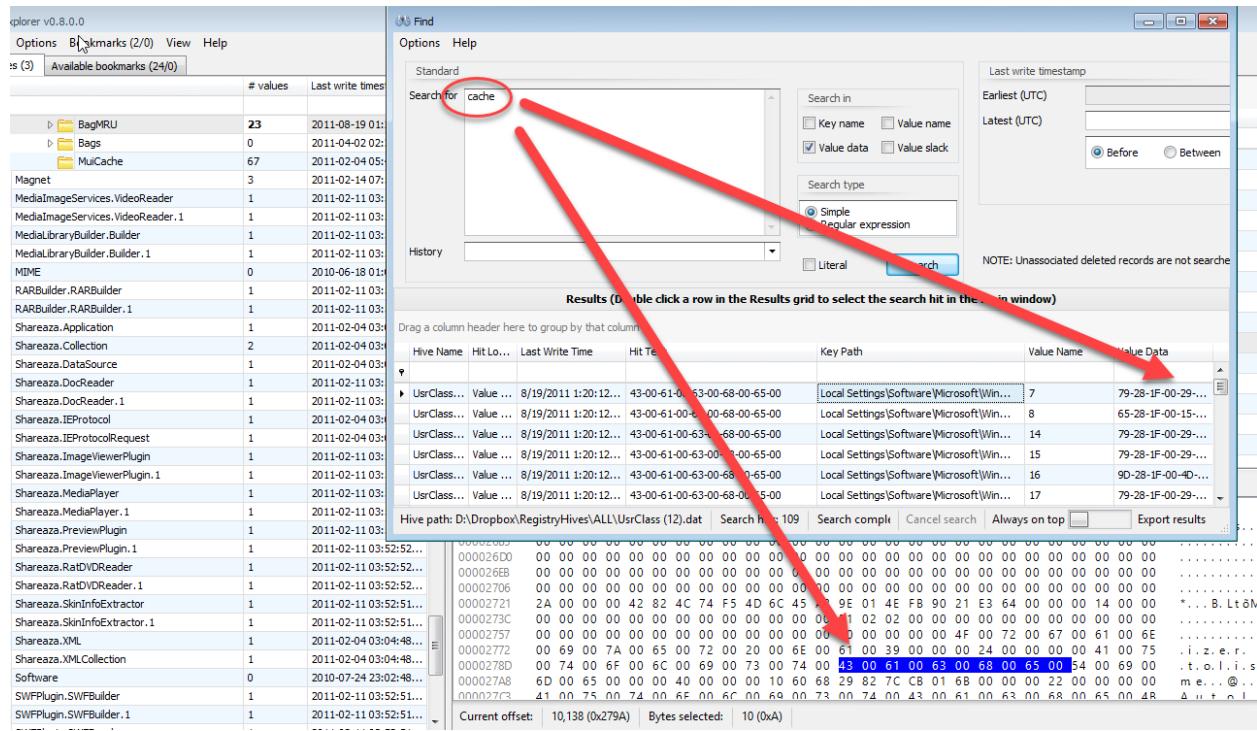
Hive Name	Hit Location	Last Write Time	Hit Text	Key Path	Value Name
UoRClass (12).dat	Value data	4/2/2011 2:20:15 AM +00:00	43-00-61-00-63-00	Local Settings\Software\Microsoft\Windows\Shell\BagMRU\1\0	10
NTUSER (.DAT)	Value data	9/23/2013 7:17:46 PM +00:00	cache	Software\Alexa Internet\Alexa9\Amazon\Cache	hOdQCSRivf2eJg+Dm63bg==
NTUSER (.DAT)	Value data	9/23/2013 7:17:46 PM +00:00	cache	Software\Alexa Internet\Alexa9\Amazon\Cache	FQdm61mYCrV
NTUSER (.DAT)	Value data	9/23/2013 7:17:46 PM +00:00	cache	Software\Alexa Internet\Alexa9\Amazon\Cache	qvw/qWkJJ2hyd

For value data (when the value type is RegBinary) and value slack, the bytes that make up the search

hit are selected in the hex viewer.



For value name and non-RegBinary value data hits, all instances of the search term are highlighted.



## Search tips

The fastest searches are against key names. Searching against value names will be slower than key names only. Searching value data/value slack is slower still. This is because every value of every key must be looked at in order to search for value names or value data/slack across all loaded hives.

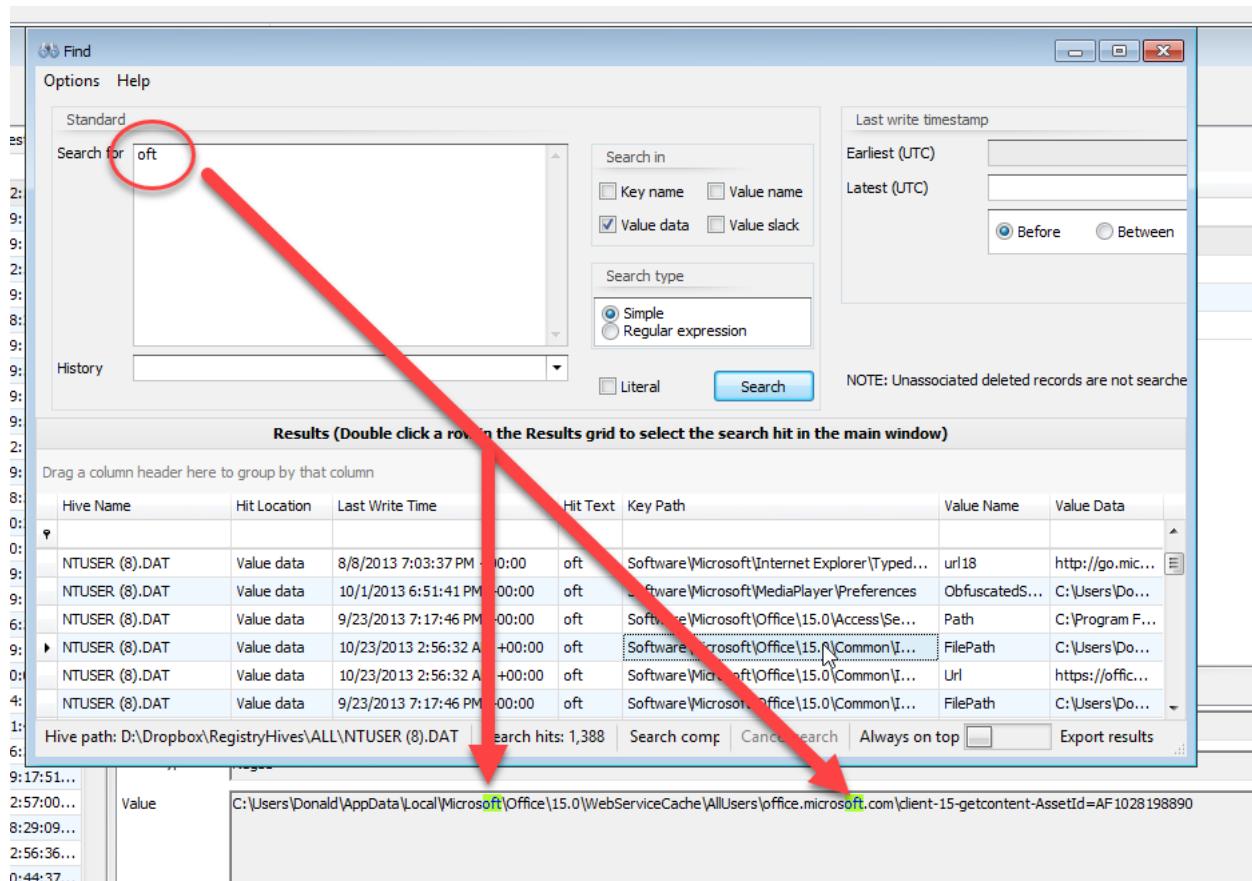
Do not let this stop you from searching against value names and data however. Even with these options selected, Registry Explorer can still search multiple hives very quickly (often under a second), but this depends on the number of keys and values in the loaded hives.

You can export the search results to Excel via the button in the lower right.

## Technical details in depth

One unique feature of Registry Explorer is the ability to view the technical details of any key, its values, security information, etc. This feature bridges the gap between a hex editor and other viewers in that Registry Explorer can be used to validate itself as to its interpretation of Registry data.

To view the technical details of a key, select the key you are interested in, then right click and select 'Technical details' from the context menu. F5 can also be used as a shortcut.

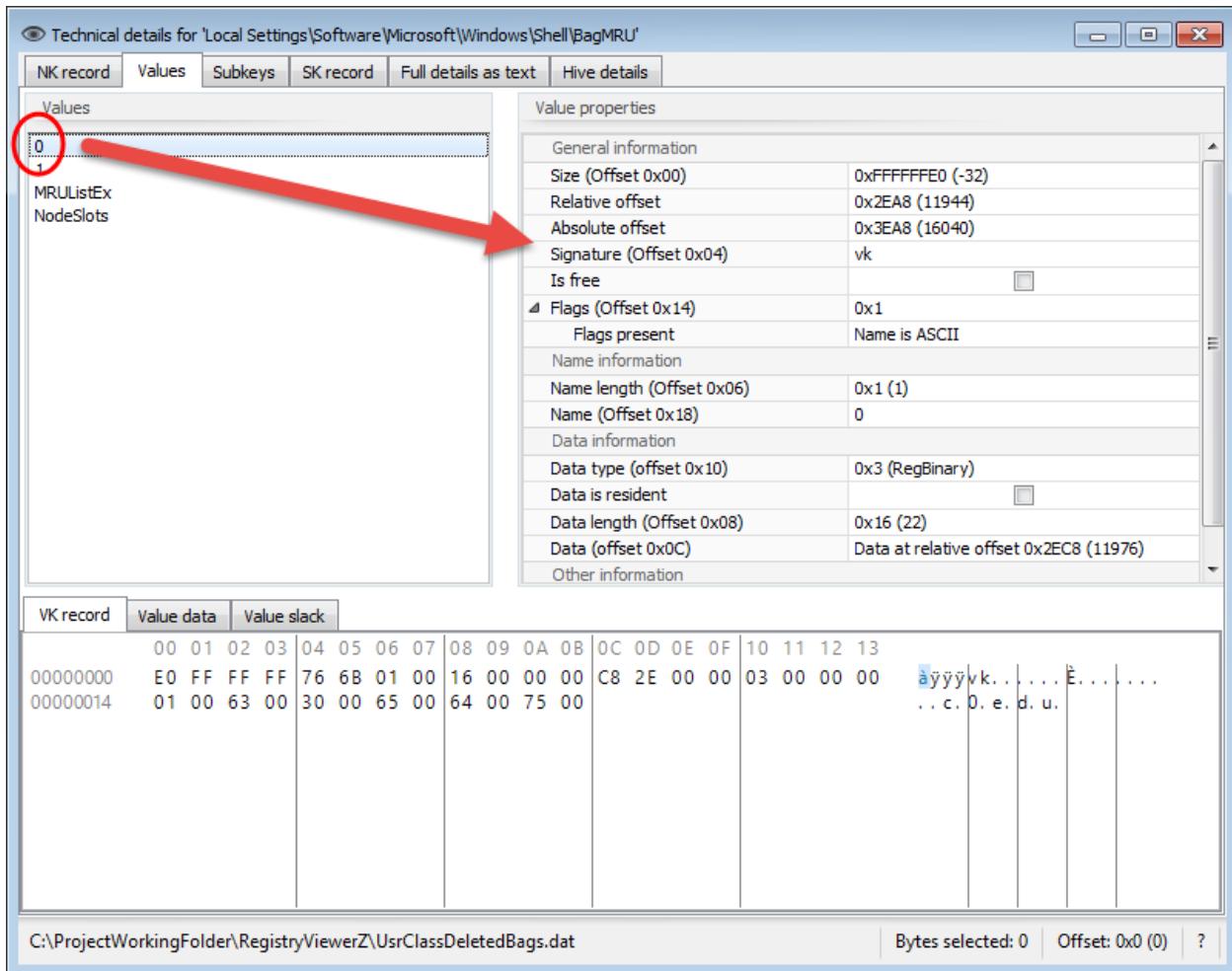


In the example above, the Technical details for the ‘Local SettingsSoftwareMicrosoftWindowsShell-BagMRU’ key are shown. The bytes at the bottom of the details form are the bytes for the NK record as they are found in the Registry hive as viewed in a hex editor.

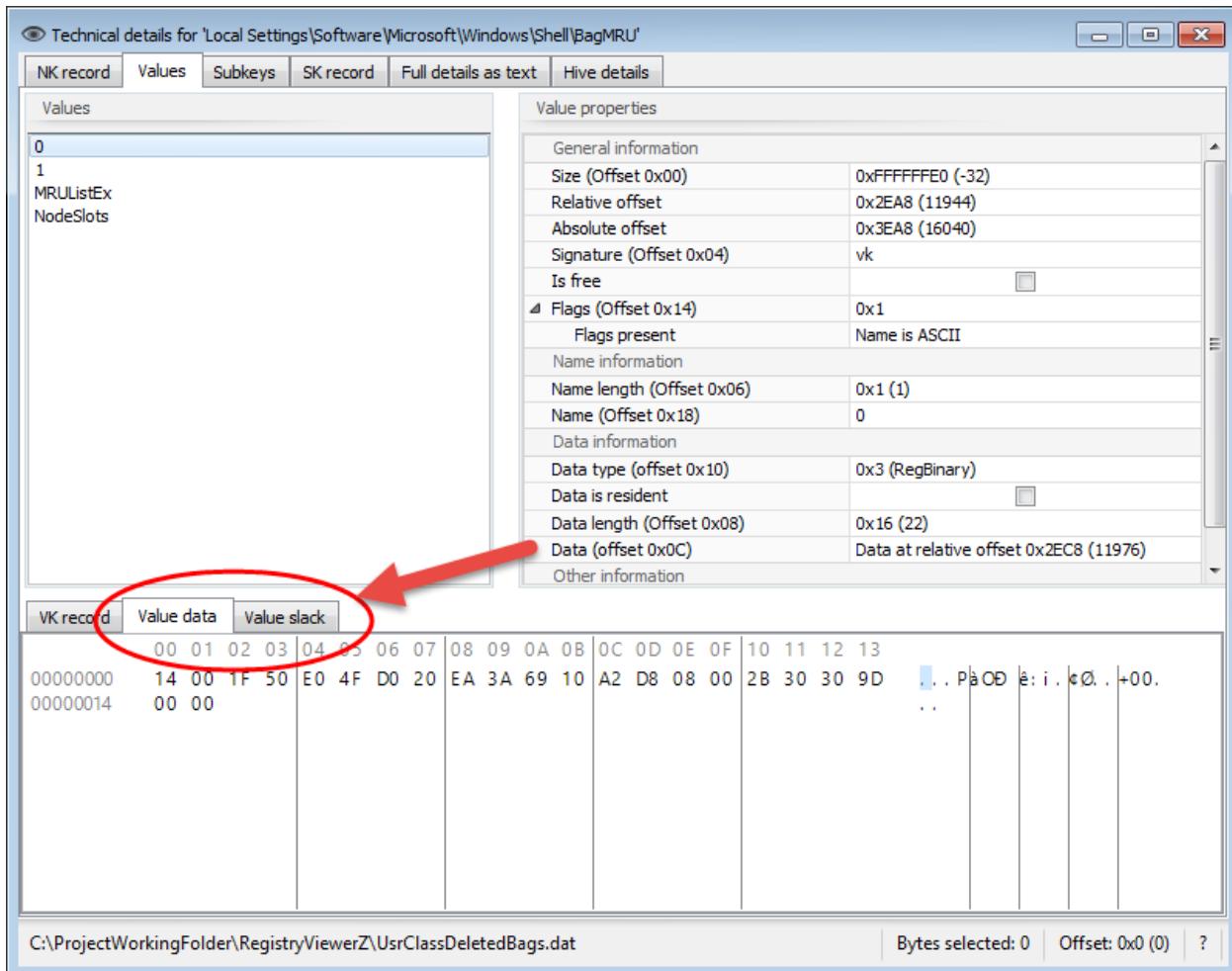
As different properties are selected, the highlighted bytes change to reflect the location in the raw data where that property lives. The Last write timestamp property is selected, as are the bytes that this property is derived from.

The selected bytes can be copied via Ctrl+C. Hold Ctrl+Alt+C to copy both the property name and the value to the clipboard.

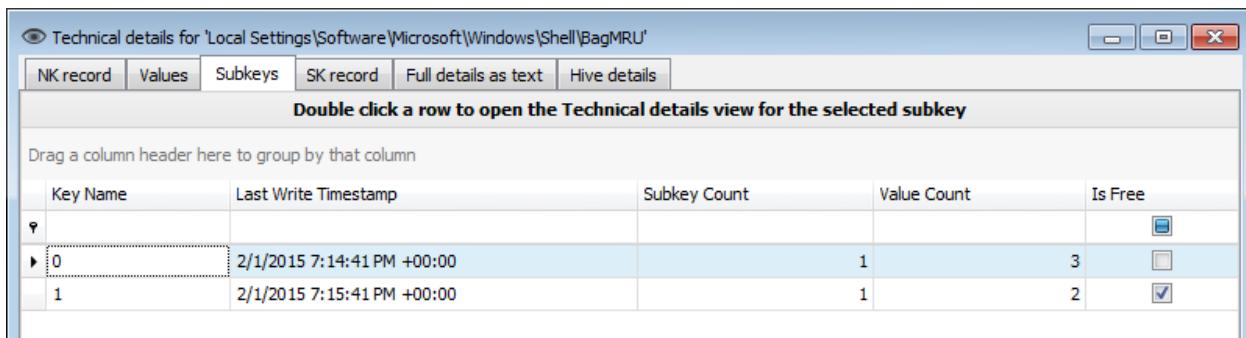
If a key contains one or more values, the Values tab is visible and contains a list of all the key’s values. Selecting a value will display the VK record’s properties and raw data as we saw with the NK record above.



At the bottom of the Values tab, the raw VK record is shown. A hex viewer for the value data and value slack (if the value has slack) is also shown. This allows you to see both the VK records and the data in one place. The value data/slack is the data that is available at the Data offset.



If a key contains subkeys, the Subkeys tab is visible and contains a list of the current key's subkeys. Double clicking on a subkey will open the Technical details report for that key in its own window.



Keys found in the active Registry (in other words, not deleted) will have an SK record tab that contains the security key information for the NK record. The SK record tab works the same as the NK and VK tab in that the hex editor updates when properties are selected, etc.

The screenshot shows the 'Technical details' window for the key 'Local Settings\Software\Microsoft\Windows\Shell\BagMRU'. The window has tabs at the top: NK record, Values, Subkeys, SK record, Full details as text, and Hive details. The 'Full details as text' tab is selected.

**General information**

Size (Offset 0x00)	0xD0 (208)
Relative offset	0x2E0 (736)
Absolute offset	0x12E0 (4832)
Signature (Offset 0x04)	sk
Forward link (Offset 0x08)	0x128 (296)
Backward link (Offset 0x0C)	0x1D8 (472)
Reference count (Offset 0x10)	0x1E (30)
Descriptor length (Offset 0x14)	0xB8 (184)

**Descriptor data**

Revision (Offset 0x18)	0x1 (1)
Control flags (Offset 0x1A)	0x00008004
Flags present	SeDadPresent, SeSelfRelative
Offset to owner (Offset 0x1C)	0x80 (128)
Offset to group (Offset 0x20)	0x9C (156)
Offset to SACL (Offset 0x24)	0x0 (0)
Offset to DACL (Offset 0x28)	0x14 (20)

**DACL**

Revision (Offset 0x00)	0x2 (2)												
00 01 02 03	04 05 06 07	08 09 0A 0B	0C 0D 0E 0F	10 11 12 13	14 15 16 17	18 19 1A 1B	1C 1D 1E 1F	20 21 22 23	24 25 26 27	28 29 2A 2B	2C 2D 2E 2F	28 29 2A 2B	2C 2D 2E 2F
00000000 30 FF FF FF	73 6B 00 00	28 01 00 00	D8 01 00 00	1E 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000014 B8 00 00 00	01 00 04 80	80 00 00 00	9C 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000028 14 00 00 00	02 00 6C 00	04 00 00 00	00 03 24 00	3F 00 0F 00	00 03 24 00	3F 00 0F 00	00 03 24 00	3F 00 0F 00	00 03 24 00	3F 00 0F 00	00 03 24 00	3F 00 0F 00	00 03 24 00
0000003C 01 05 00 00	00 00 00 05	15 00 00 00	47 19 B6 08	D8 72 C8 03	15 00 00 00	47 19 B6 08	D8 72 C8 03	15 00 00 00	47 19 B6 08	D8 72 C8 03	15 00 00 00	47 19 B6 08	D8 72 C8 03
00000050 DB 01 6C 48	E8 03 00 00	00 03 14 00	3F 00 0F 00	01 01 00 00	00 03 14 00	3F 00 0F 00	01 01 00 00	00 03 14 00	3F 00 0F 00	01 01 00 00	00 03 14 00	3F 00 0F 00	00 03 14 00
00000064 00 00 00 05	12 00 00 00	00 03 18 00	3F 00 0F 00	01 02 00 00	00 03 18 00	3F 00 0F 00	01 02 00 00	00 03 18 00	3F 00 0F 00	01 02 00 00	00 03 18 00	3F 00 0F 00	00 03 18 00
00000078 00 00 00 05	20 00 00 00	20 02 00 00	00 03 14 00	19 00 02 00	20 00 00 00	20 02 00 00	00 03 14 00	19 00 02 00	00 03 14 00	20 02 00 00	00 03 14 00	19 00 02 00	00 03 14 00
0000008C 01 01 00 00	00 00 00 05	0C 00 00 00	01 05 00 00	00 00 00 05	00 00 00 00	01 05 00 00	00 00 00 00	00 00 00 00	01 05 00 00	00 00 00 00	00 00 00 00	01 05 00 00	00 00 00 00
000000A0 15 00 00 00	47 19 B6 08	D8 72 C8 03	DB 01 6C 48	E8 03 00 00	15 00 00 00	47 19 B6 08	D8 72 C8 03	DB 01 6C 48	E8 03 00 00	15 00 00 00	47 19 B6 08	D8 72 C8 03	DB 01 6C 48
000000B4 01 05 00 00	00 00 00 05	15 00 00 00	47 19 B6 08	D8 72 C8 03	01 05 00 00	15 00 00 00	47 19 B6 08	D8 72 C8 03	01 05 00 00	15 00 00 00	47 19 B6 08	D8 72 C8 03	01 05 00 00
000000C8 DB 01 6C 48	01 02 00 00	00 00 00 00	00 03 14 00	19 00 02 00	00 00 00 00	00 03 14 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

C:\ProjectWorkingFolder\RegistryViewerZ\UsrClassDeletedBags.dat Bytes selected: 4 Offset: 0x10 (16) ?

The 'Full details as text' tab contains a textual representation of the selected key including the NK record, all VK records, and the SK record. This can be copied and pasted into reports as needed.

Finally, the Hive details tab contains information about the hive where the key was found. This includes the sequence numbers, timestamp, length, root key name, checksum, and so on.

Technical details for 'Local Settings\Software\Microsoft\Windows\Shell\BagMRU'	
NK record	Values
Subkeys	SK record
Full details as text	Hive details
General information	
Signature	regf
Sequence 1	0x15 (21)
Sequence 2	0x15 (21)
Timestamp	2/1/2015 7:15:49 PM +00:00
Version	1.3
Length	0x34000 (212,992)
Embedded file name	\Microsoft\Windows\UsrClass.dat
Root cell index	0x20 (32)
Root key name	S-1-5-21-146151751-63468248-1215037915-1000_Classes
CheckSum	0x8CB71E6C (-1934156180)
Calculated CheckSum	0x8CB71E6C (-1934156180)
CheckSum match	<input checked="" type="checkbox"/>

## Plugins

Plugins provide a means to process a key and/or value in the Registry. They are primarily intended for binary or otherwise obfuscated keys and values to be decoded to a more user-friendly manner. The plugin architecture is open source and easy to implement.

Each release of Registry Explorer will contain all available plugins, but the hope is that others will also contribute to the Registry Explorer project, located at <https://github.com/EricZimmerman/RegistryPlugins>.

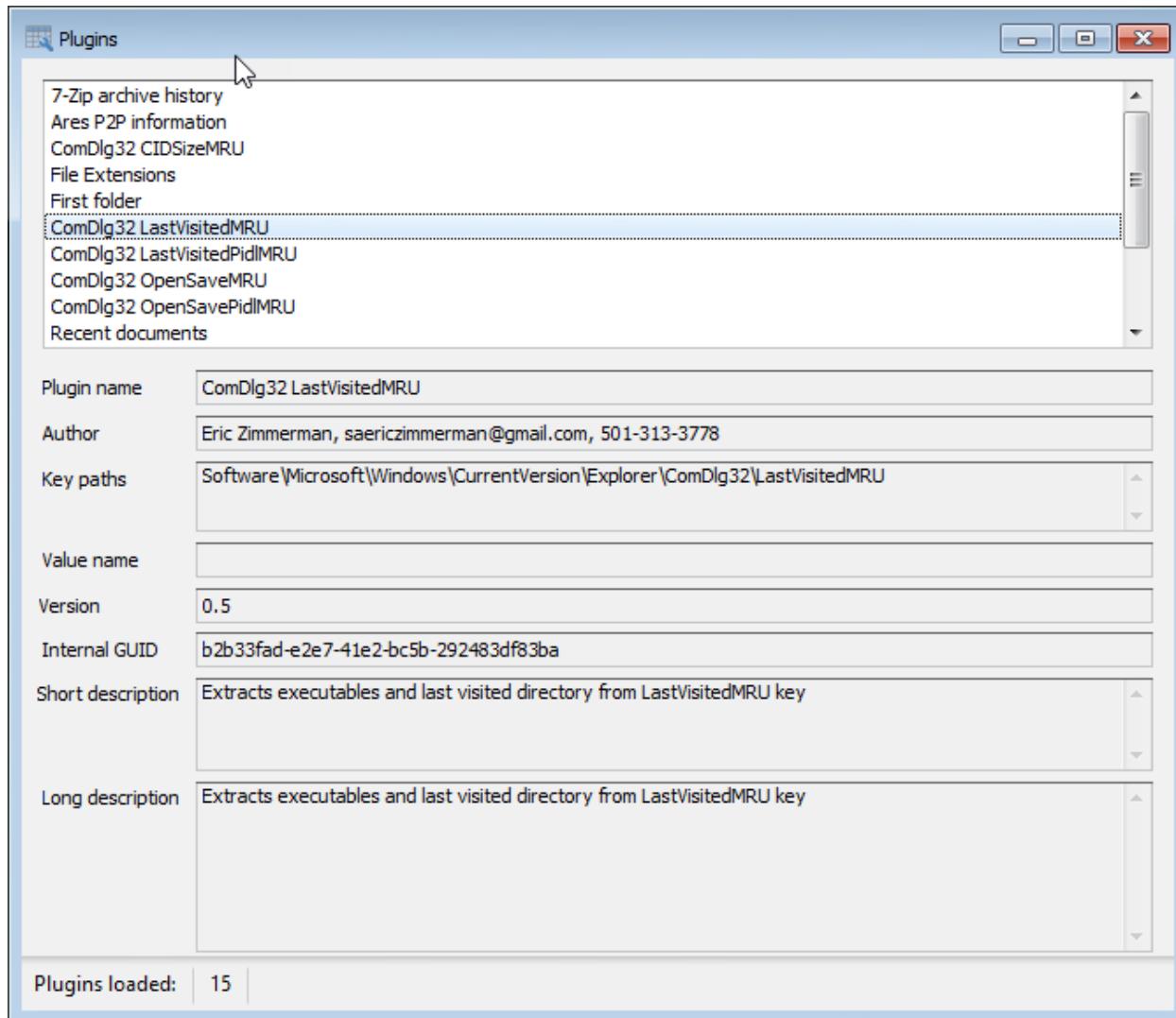
Plugins live under the main Registry Explorer directory in a subdirectory called ‘Plugins’ which can also contain other directories as needed. Plugins are named according to the format:

RegistryPlugin\*.dll

Where the \* is a description of what the plugin is for. Any subdirectories under the Plugins directory are also checked for files matching the above specification.

When Registry Explorer is started, it looks for all files matching that pattern. It then verifies that each file found is indeed a plugin. If it is, the plugin is made available to Registry Explorer.

To view all available plugins, use the View | Plugins menu option. When this is selected, the following dialog is displayed:



As different plugins are selected, the properties for the plugin are updated. In the above example we can see the name of the plugin and the exact key path (or paths) a plugin will handle.

Plugins can be tied to a key name or a key name and a value name. When a plugin processes particular value, that value name will be shown in the appropriate field.

The Internal GUID is an identifier Registry Explorer uses to make sure each plugin is unique. In this way you can have multiple plugins for a given key and things would still work properly (vs. basing uniqueness off of a name or similar).

The short and long descriptions are used to explain in more detail what a plugin is doing, why it is relevant, etc. The long description can also include even more details including links to blogs, etc.

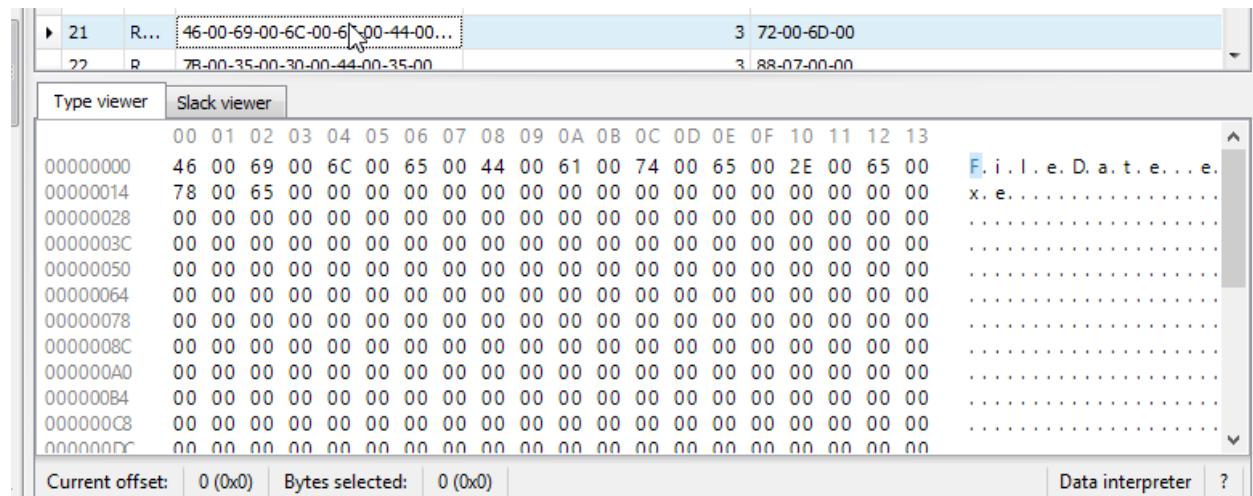
## Using plugins

There is no requirement on a user's part other than clicking on a given key and/or value. As Registry Explorer is used to navigate around a hive, Registry Explorer checks if any plugins have registered an interest in the selected key/value. If any plugins are found, the key is passed to the plugin for processing and the plugin then returns results to Registry Explorer which it then displays.

For example, say a user clicks on the SOFTWAREMicrosoftWindowsCurrentVersionExplorerComDlg32CIDSsizeMRU key. This key's values look like this:

Values			
Drag a column header here to group by that column			
Value Name	Value Type	Data	V
54	RegBinary	46-00-54-00-4B-00-20-00-49-00-6D-00-61-00-67-00-65-00-72-00-2E-00-65-00-...	
55	RegBinary	62-00-69-00-74-00-6D-00-61-00-70-00-63-00-61-00-63-00-68-00-65-00-76-00-...	
56	RegBinary	70-00-65-00-67-00-61-00-73-00-75-00-73-00-67-00-75-00-69-00-2E-00-65-00-...	
57	RegBinary	70-00-6F-00-77-00-65-00-72-00-73-00-68-00-65-00-6C-00-6C-00-70-00-6C-00-...	
58	RegBinary	70-00-75-00-74-00-74-00-79-00-67-00-65-00-6E-00-2E-00-65-00-78-00-65-00-...	
59	RegBinary	76-00-6D-00-77-00-61-00-72-00-65-00-2E-00-65-00-78-00-65-00-00-00-00-00-...	
6	RegBinary	7B-00-31-00-42-00-46-00-41-00-45-00-46-00-38-00-42-00-2D-00-31-00-37-00-...	
60	RegBinary	53-00-65-00-74-00-75-00-70-00-48-00-6F-00-73-00-74-00-2E-00-45-00-78-00-...	
61	RegBinary	53-00-68-00-65-00-6C-00-6C-00-42-00-61-00-67-00-73-00-45-00-78-00-70-00-...	
62	RegBinary	4D-00-66-00-74-00-32-00-43-00-73-00-76-00-36-00-34-00-2E-00-65-00-78-00-...	
63	RegBinary	4D-00-66-00-74-00-32-00-43-00-73-00-76-00-2E-00-65-00-78-00-65-00-00-00-...	
7	RegBinary	65-00-78-00-70-00-6C-00-6F-00-72-00-65-00-72-00-2E-00-65-00-78-00-65-00-...	
8	RegBinary	64-00-65-00-76-00-65-00-6E-00-76-00-2E-00-65-00-78-00-65-00-00-00-00-00-...	
9	RegBinary	6E-00-61-00-76-00-69-00-63-00-61-00-74-00-2E-00-65-00-78-00-65-00-00-00-...	
MRUListEx	RegBinary	16-00-00-00-05-00-00-00-00-00-00-00-3E-00-00-00-3F-00-00-00-2A-00-00-00-...	

There are many RegBinary values and an MRUListEx value that tracks the order each of the values. Clicking on a value would display all the binary data in the hex viewer, like this:



A Unicode string can be seen in the data, but it is difficult to see all the data at once. This is where the plugin steps in and presents a much easier to use presentation of the values under this key. After a plugin processes a key, a new tab is displayed next to the Values tab at the top of the right side of Registry Explorer:

Values ComDlg32 CIDSizeMRU			
Drag a column header here to group by that column			
Executable	MRU ...	Opened On	
{50D526C4-E50C-4138-A032-29EB9DE9EC35}	0	5/3/2016 4:41:26 PM +00:00	
PickerHost.exe	1		
EditPadPro7.exe	2		
Mft2Csv64.exe	3		
Mft2Csv.exe	4		
{33BE88C0-71D4-49E5-9891-64DB26171AF2}	5		
ShellBagsExplorer.exe	6		
xwforensics.exe	7		
{1BFAEF8B-17E3-4D03-BAD2-68853E050708}	8		
snagiteditor.exe	9		
chrome.exe	10		
{21EB7BD4-C127-401D-9751-5D221AEF0307}	11		
postbox.exe	12		
vlc.exe	13		

Total rows: 64 | Export | ?

Each value is processed, and the results are displayed in a grid which can then be sorted on, filtered,

or exported as needed

For plugins that handle a key and a value, like the AppCompatCache plugin, the results of the plugin are displayed on a tab next to the Type viewer at the bottom part of the interface (below the values tab).

As an example, the 7-Zip archive history value is a NULL terminated list of archives opened in 7-Zip.

Type viewer															
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E
00000000	44	00	3A	00	5C	00	44	00	72	00	6F	00	70	00	62
0000000F	00	6F	00	78	00	5C	00	21	00	4B	00	72	00	6F	00
0000001E	6C	00	6C	00	5C	00	45	00	78	00	70	00	65	00	6E
0000002D	00	73	00	65	00	20	00	72	00	65	00	70	00	6F	00
0000003C	72	00	74	00	73	00	5C	00	41	00	70	00	72	00	69
0000004B	00	6C	00	20	00	32	00	33	00	20	00	32	00	30	00
0000005A	31	00	36	00	5C	00	41	00	70	00	72	00	69	00	6C
00000069	00	20	00	32	00	33	00	20	00	32	00	30	00	31	00
00000078	36	00	2E	00	7A	00	69	00	70	00	00	00	44	00	3A
00000087	00	5C	00	6D	00	49	00	2E	00	65	00	78	00	65	00
00000096	00	00	43	00	3A	00	5C	00	55	00	73	00	65	00	72
000000A5	00	73	00	5C	00	65	00	5C	00	44	00	65	00	73	00
000000B4	6B	00	74	00	6F	00	70	00	5C	00	46	00	52	00	4F
000000C3	00	4D	00	20	00	68	00	69	00	74	00	73	00	20	00
000000D2	53	00	51	00	4C	00	2E	00	37	00	7A	00	00	00	43
000000E1	00	3A	00	5C	00	54	00	65	00	6D	00	70	00	5C	00
000000F0	6F	00	73	00	54	00	72	00	69	00	61	00	67	00	65
000000FF	00	32	00	5F	00	63	00	75	00	73	00	74	00	6F	00
0000010E	6D	00	2E	00	65	00	78	00	65	00	00	00			

Current offset: 0 (0x0) Bytes selected: 0 (0x0) Data interpreter ?

When this value is selected, the 7-Zip plugin processes the value and returns a much nicer list, like this:

The screenshot shows the Registry Explorer interface. On the left, the registry tree is displayed with several keys under 'SOFTWARE'. One key, 'Compression' under '7-Zip', is highlighted with a red oval and has a red arrow pointing to it from the left. To the right of the tree is a grid titled 'Values' showing five entries. Another red arrow points from the bottom of the 'Values' grid to the 'Type viewer' tab at the bottom, which is also highlighted with a red oval. The 'Type viewer' tab displays a list of archive names.

Value Name	Value Type	Data	Value Type Raw	Value Slack
ArchHistory	RegBinary	44-00-3A-00-5C-00...	3	00-00
Archer	RegSz	zip	1	2C-39-29-00
EncryptHeader	RegDword	0		4
Level	RegDword	5		4
ShowPassword	RegDword	1		4

Type viewer    7-Zip archive history

Archive Name
D:\Dropbox\JKroll\Expense reports\April 23 2016\April 23 2016.zip
D:\nI.exe
C:\Users\el\Desktop\FROM hits SQL.7z
C:\Temp\osTriage2_custom.exe

The data returned can now be filtered, sorted, and exported as we saw earlier.

Some plugins can return a vast amount of information which can make displaying it all in a grid overwhelming. In these cases, Registry Explorer will, by default, hide the details from the plugin view (but this column can easily be unhidden if you like). As an example, let's look at the LastVisitedPidlMRULegacy plugin. When this key is selected, all values are processed, and the results are displayed as we saw before. This plugin has a property named 'Details' which contains just that, a very detailed listing of information extracted from a value. When a value is selected a new tab is shown next to the Type viewer tab that contains the details of the selected value, like this:

The screenshot shows the Registry Explorer interface with two panes. The left pane displays a tree view of registry keys under 'Available bookmarks (29/0)'. One key, 'LastVisitedPidMRULegacy', is highlighted with a red oval and has a red arrow pointing to the right pane. The right pane has tabs for 'Values' and 'ComDlg32 LastVisitedPidMRU'. It shows a table with columns: Value Name, Mru Position, Executable, Absolute Path, and Opened On. A red arrow points from the selected row in this table to a detailed viewer window. The viewer window is titled 'Type viewer ComDlg32 LastVisitedPidMRU selected row details' and contains several sections of decoded data for the selected value.

Value Name	Mru Position	Executable	Absolute Path	Opened On
0	2	MozBackup.exe	Unmapped GUID: e31eb727-12ed-4702-820c-4b6445f28e1a	
4	4	phpDesigner.exe	My Computer U:	
5	0	Wireshark.exe	dc\Host PCAP	4/29/2016 5:48:52 PM +00:00
1	3	WinHex.exe	My Computer\C:\Temp\testdump\la52b0784bd667468,	

**Type viewer ComDlg32 LastVisitedPidMRU selected row details**

Type: Directory, Value: dc

Extension blocks found: 1  
----- Block 0 (Beef0004)-----

Long name: dc  
Created: 2/19/2016 4:31:10 PM +00:00  
Last access: 2/19/2016 4:31:10 PM +00:00  
MFT entry/sequence #: 86129/116 (0x15071/0x74)  
File system hint: NTFS

Short name: dc  
Modified: 2/19/2016 4:31:10 PM +00:00

Type: Directory, Value: Host PCAP

Extension blocks found: 1  
----- Block 0 (Beef0004)-----

Long name: Host PCAP  
Created: 2/19/2016 4:33:00 PM +00:00  
Last access: 2/19/2016 4:33:00 PM +00:00  
MFT entry/sequence #: 401605/10 (0x620C5/0xA)  
File system hint: NTFS

Short name: HOSTPC~1  
Modified: 2/19/2016 4:33:00 PM +00:00

This particular plugin decodes all of the shell items found in the binary data and places this decoded data into the Details property. This is what is displayed when a value is selected.

When exporting plugin results for plugins that have a Details property, this column will be exported as well, like this:

The screenshot shows a software interface with a menu bar at the top. The menu items include Normal, Page Break, Page, Custom, Preview, Layout, Views, Gridlines, Headings, Zoom, 100%, Zoom to Selection, Window, New, Arrange, Freeze, Unhide, Reset Window Position, and Switch Windows. Below the menu is a toolbar with icons for Show, Zoom, and Window. The main area displays a grid of data in a tabular format. The columns are labeled A, B, C, D, E, and F. Column A contains the value '5'. Column B contains the name 'Wireshark.exe'. Column C contains the executable path 'dc\Host PCAP'. Column D contains the absolute path 'dc\Host PCAP'. Column E contains the date '4/29/2016 5:48:52 PM +00:00'. Column F contains detailed file information. It includes sections for 'Type: Directory, Value: dc', 'Extension blocks found: 1', 'Long name: dc', 'Created: 2/19/2016 4:31:10 PM +00:00', 'Last access: 2/19/2016 4:31:10 PM +00:00', 'MFT entry/sequence #: 86129/116 (0x15071/0x74)', and 'File system hint: NTFS'. Another section for 'Type: Directory, Value: Host PCAP' follows, with similar details. A third section for 'Short name: HOSTPC~1' and 'Modified: 2/19/2016 4:31:10 PM +00:00' is also present.

A	B	C	D	E	F	
1	Value Name	Mru Position	Executable	Absolute Path	Opened On	Details
5		0	Wireshark.exe	dc\Host PCAP	4/29/2016 5:48:52 PM +00:00	<p>Type: Directory, Value: dc</p> <p>Extension blocks found: 1</p> <p>----- Block 0 (Beef0004) -----</p> <p>Long name: dc</p> <p>Created: 2/19/2016 4:31:10 PM +00:00</p> <p>Last access: 2/19/2016 4:31:10 PM +00:00</p> <p>MFT entry/sequence #: 86129/116 (0x15071/0x74)</p> <p>File system hint: NTFS</p> <p>-----</p> <p>Short name: dc</p> <p>Modified: 2/19/2016 4:31:10 PM +00:00</p> <p>Type: Directory, Value: Host PCAP</p> <p>Extension blocks found: 1</p> <p>----- Block 0 (Beef0004) -----</p> <p>Long name: Host PCAP</p> <p>Created: 2/19/2016 4:33:00 PM +00:00</p> <p>Last access: 2/19/2016 4:33:00 PM +00:00</p> <p>MFT entry/sequence #: 401605/10 (0x620C5/0xA)</p> <p>File system hint: NTFS</p> <p>-----</p> <p>Short name: HOSTPC~1</p> <p>Modified: 2/19/2016 4:33:00 PM +00:00</p>

The Details column seen in the Excel sheet above is the same that is available after unhiding the details column in Registry Explorer.

## Creating plugins

NOTE: THIS MAY NOT BE 100% CURRENT AS YOU ARE READING THIS! USE THE EXAMPLES IN THE GITHUB REPOSITORY FOR WORKING SAMPLES!!

To create a new plugin, download the RegistryPlugins project from Github, open the project in Visual Studio, and create a new project that follows the correct naming convention similar to what we saw earlier, RegistryPlugin.<something>. Once the project is created, add a reference to the RegistryPluginBase project as this project will contain the base types and classes needed for a plugin. Once this is done, the actual coding can begin. By default, a generic class is created in the new project. Rename this to something more meaningful. This main class will contain the “brains” of the plugin. Next, add a new class to the project and call it ValuesOut. While you can name this class anything, most other plugins use this same convention.

The ValuesOut class defines the objects the plugin will return for display in Registry Explorer. For example, the 7-Zip plugins ValuesOut class looks like this:

```
1  namespace RegistryPlugin._7_ZipHistory
2  {
3      4 references | Eric Zimmerman, 6 days ago | 1 author, 2 changes
4          public class ValuesOut
5          {
6              1 reference | Eric Zimmerman, 6 days ago | 1 author, 2 changes
7                  public ValuesOut(string archiveName)
8                  {
9                      ArchiveName = archiveName;
10                 }
11             }
12 }
```

The important things to remember is to add read only properties and define a constructor that allows for setting up the object. By doing this way we ensure our objects are immutable.

With the ValuesOut class done, we can code the primary class. Using the 7-Zip project as a reference again, let's take a look at the top section of the class:

```
namespace RegistryPlugin._7_ZipHistory
{
    public class SevenZip : IRegistryPluginGrid
    {
        private readonly BindingList<ValuesOut> _values;

        public SevenZip()
        {
            _values = new BindingList<ValuesOut>();
            Errors = new List<string>();
        }

        public string InternalGuid => "6b1296a2-d3fb-441f-89c1-fd3706855acc";

        public List<string> KeyPaths => new List<string>(new[]
        {
            @"Software\7-Zip\Compression"
        });

        public string ValueName => "ArcHistory";
        public string AlertMessage { get; private set; }
        public RegistryPluginType.PluginType PluginType => RegistryPluginType.PluginType.Grid;
        public string Author => "Eric Zimmerman";
        public string Email => "saericzimmerman@gmail.com";
        public string Phone => "501-313-3778";
        public string PluginName => "7-Zip archive history";

        public string ShortDescription =>
            "Extracts archive history from ArcHistory key"
            ;

        public string LongDescription => ShortDescription;

        public double Version => 0.5;
        public List<string> Errors { get; }
    }
}
```

After the class name we can see a reference to an Interface, IRegistryPluginGrid. This interface contains the ‘rules’ the class must follow as it relates to properties. The interface definition looks like this:

```
namespace RegistryPluginBase.Interfaces
{
    16 references | EricZimmerman, 317 days ago | 1 author, 2 changes
    public interface IRegistryPluginGrid : IRegistryPluginBase
    {
        /// <summary>
        ///     Gets the values after processing by a plugin.
        /// </summary>
        /// <remarks>The underlying class should extend ProcessedValue</remarks>
        /// <value>The values.</value>
        45 references | EricZimmerman, 318 days ago | 1 author, 1 change
        IBindingList Values { get; }
    }
}
```

Which in turn references another interface, IRegistryPluginBase, that looks like this:

```

1 reference | Eric Zimmerman, 308 days ago | 2 authors, 5 changes
public interface IRegistryPluginBase
{
    /// <summary>
    /// Gets the internal unique identifier.
    /// </summary>
    /// <remarks>
    /// Set this to a static GUID value in plugin's constructor. This is used to make sure plugins aren't loaded more
    /// than once.
    /// </remarks>
    /// <value>The internal unique identifier.</value>
16 references | Eric Zimmerman, 308 days ago | 2 authors, 2 changes
    string InternalGuid { get; }

    /// <summary>
    /// The path to the key this plugin handles.
    /// </summary>
    /// <remarks>Do not include the root key in the key path</remarks>
    /// <value>The key path.</value>
16 references | Eric Zimmerman, 310 days ago | 1 author, 1 change
    List<string> KeyPaths { get; }

    /// <summary>
    /// The value name this plugin handles
    /// </summary>
    /// <value>The name of the value.</value>
17 references | Eric Zimmerman, 318 days ago | 1 author, 1 change
    string ValueName { get; }

    /// <summary>
    /// Gets the alert message.
    /// </summary>
    /// <remarks>Optional message to display to user (an interesting value, missing info, etc)</remarks>
    /// <value>The alert message.</value>
34 references | Eric Zimmerman, 308 days ago | 2 authors, 2 changes
    string AlertMessage { get; }

    16 references | Eric Zimmerman, 317 days ago | 1 author, 2 changes
    RegistryPluginType.PluginType PluginType { get; }
    16 references | Eric Zimmerman, 317 days ago | 1 author, 1 change
    string Author { get; }
    16 references | Eric Zimmerman, 317 days ago | 1 author, 1 change
    string Email { get; }
    16 references | Eric Zimmerman, 317 days ago | 1 author, 1 change
    string Phone { get; }
    16 references | Eric Zimmerman, 317 days ago | 1 author, 1 change
    string PluginName { get; }
    21 references | Eric Zimmerman, 317 days ago | 1 author, 1 change
    string ShortDescription { get; }
    16 references | Eric Zimmerman, 317 days ago | 1 author, 1 change
    string LongDescription { get; }
    16 references | Eric Zimmerman, 317 days ago | 1 author, 1 change
    double Version { get; }
    85 references | Eric Zimmerman, 317 days ago | 1 author, 1 change
    List<string> Errors { get; }

    /// <summary>
    /// Process raw values into plugin specific format
    /// </summary>
    /// <remarks>This method should populate the 'Values' property for plugins implementing this interface</remarks>
    /// <param name="key">The key where inValues originated. Also contains all subkeys, values, etc</param>
25 references | Eric Zimmerman, 308 days ago | 1 author, 1 change
    void ProcessValues(RegistryKey key);
}

```

These interfaces define what properties must exist in a class that implements a given interface.

Looking back at our SevenZip class, we can see several of the properties from the interface definitions along with a few other fields and variables. First, we see a read-only collection named \_values which will hold each of our ValueOut objects we create. Next is the constructor which initializes the class. The internal GUID is nothing more than a unique GUID that can be generated via the C# Interactive window (or any other means using Guid.NewGuid.ToString() method).

The next two properties define the key that this plugin is interested in and optionally, a value. This particular plugin does contain a value name and as such, a combination of the key name and value name is used.

The PluginType defines the how the resulting data from the plugin will be displayed. As of 0.8.0.0, Grid is the only valid option.

The Author, Email, and Phone properties contain information about who made the plugin and how to get a hold of them.

The PluginName, descriptions, and Version properties are next and are self-explanatory.

The Errors collection will contain a list of any errors encountered by the plugin as it processes a key and/or value.

The remaining part of the class looks like this:

```
85 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
public List<string> Errors { get; }

25 references | Eric Zimmerman, 6 days ago | 1 author, 2 changes
public void ProcessValues(RegistryKey key)
{
    _values.Clear();
    Errors.Clear();

    try
    {
        var arcHist = key.Values.SingleOrDefault(t => t.ValueName == "ArcHistory");

        if (arcHist != null)
        {
            var arcs = Encoding.Unicode.GetString(arcHist.ValueDataRaw).Split('\0');

            foreach (var arc in arcs)
            {
                if (arc.Trim().Length == 0)
                {
                    continue;
                }
                var v = new ValuesOut(arc);
                Values.Add(v);
            }
        }
        catch (Exception ex)
        {
            Errors.Add($"Error processing 7-Zip archive history: {ex.Message}");
        }

        if (Errors.Count > 0)
        {
            AlertMessage = "Errors detected. See Errors information in lower right corner of plugin window";
        }
    }
}

1 reference | Eric Zimmerman, 6 days ago | 1 author, 1 change
public IBindingList Values => _values;
```

The ProcessValues function is called by Registry Explorer when it is determined a given key should be processed by a plugin. This is where a key should be looked at and processed into ValuesOut objects. It is very important to properly handle any possible errors by use or Try/Catch blocks. This keeps the plugin from crashing and allows for reporting errors to a user in a consistent manner.

The AlertMessage can be anything a plugin author wishes and will be displayed in Registry Explorer below the grid containing plugin results.

The Values property at the bottom is the property Registry Explorer will use to display the data returned by the plugin.

Creating plugins is very simple in that it is some basic plumbing code (name, email, key path, etc.) and a single function to process things. While this example showed a simple plugin, there are other, more complicated examples in the Github project you can use as templates for new plugins.

## RECmd

RECmd is a command line tool used to access offline Registry hives. It includes many of the same features as Registry Explorer including searching, looking at keys and values, and exporting data. Version 1.2 added batch mode and plugin support for automated searching and extracting of data to CSV.

RECmd uses the same back end as Registry Explorer to process Registry hives. RECmd is open source and the source code is available [here](#).

## Getting started

Running RECmd.exe without any arguments displays a list of command line options as shown below.

```
RECmd version 1.2.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/RECmd

Note: Enclose all strings containing spaces (and all RegEx) with double quotes

d          Directory to look for hives (recursively). -f or -d is required.
f          Hive to search. -f or -d is required.

kn         Display details for key name. Includes subkeys and values
vn         Value name. Only this value will be dumped
bn         Use settings from supplied file to find keys/values. See included sample file for examples
csv        Directory to save CSV formatted results to. Required when -bn is used.
csvf       File name to save CSV formatted results to. When present, overrides default name
saveTo     Saves --vn value data in binary form to file. Expects path to a FILE
json       Export --kn to directory specified by --json. Ignored when --vn is specified

details    Show more details when displaying results. Default is FALSE

Base64    Find Base64 encoded values with size >= Base64 (specified in bytes)
MinSize   Find values with data size >= MinSize (specified in bytes)

sk         Search for <string> in key names.
sv         Search for <string> in value names
sd         Search for <string> in value record's value data
ss         Search for <string> in value record's value slack
literal   If true, --sd and --ss search value will not be interpreted as ASCII or Unicode byte strings
nd         If true, do not show data when using --sd or --ss. Default is FALSE
regex     If present, treat <string> in --sk, --sv, --sd, and --ss as a regular expression. Default is FALSE

dt         The custom date/time format to use when displaying time stamps. Default is: yyyy-MM-dd HH:mm:ss.ffffffff
nl         When true, ignore transaction log files for dirty hives. Default is FALSE
recover   If true, recover deleted keys/values. Default is TRUE

debug     Show debug information during processing
trace     Show trace information during processing

Example: RECmd.exe --f "C:\Temp\UsrClass 1.dat" --sk URL --recover false --nl
          RECmd.exe --f "D:\temp\UsrClass 1.dat" --StartDate "11/13/2014 15:35:01"
          RECmd.exe --f "D:\temp\UsrClass 1.dat" --RegEx --sv "(App|Display)Name"
```

There are several groups of command line options for RECmd. They are not case sensitive.

## Source

- f: The full path to the hive to process. If the path contains spaces, include them in double quotes.
- d: The full path to a directory to recursively search for hives to process. If the path contains spaces, include them in double quotes.

## Batch

- bn: Path to batch configuration file (see section below for details).
- csv: The directory to save batch results to
- csvf: File name to save CSV formatted results to. When present, overrides default name

## Query

If either the key or value has spaces in them, be sure to enclose them in quotes.

When passing in key names, the root key name is optional. This is because most of the time you will not even know the root key name in order to be able to include it.

To get default values, use a value name of “(default)”.

- kn: The key name to look for. If used without –vn, displays all subkeys and values.
- vn: Display only the value specified
- SaveTo: Saves –vn value data in binary form to a file
- json: Export –kn to directory specified by –json. Ignored when –vn is present
- details: Show more details when displaying results. Default is FALSE

## Search

This is a particularly useful feature to locate data across hives in key names, value names, and perhaps most importantly, in value data.

Searching is broken down into four types, by last write timestamp, value data minimum size, simple string searches (and regular expression (RegEx) based searches when –RegEx is present).

- MinSize: Find values with value data size greater than or equal to the specified size (in bytes).
- Base64: Find Base64 encoded values greater than or equal to the specified size (in bytes).
- sk: Search for <string> in key names
- sv: Search for <string> in value names
- sd: Search for <string> in value record’s value data. The value data will be converted to its equivalent in ASCII and Unicode and searched/compared to <string> unless the –Literal switch is used
- ss: Search for <string> in value record’s value slack. The value slack will be converted to its equivalent in ASCII and Unicode and searched/compared to <string> unless the –Literal switch is used
- RegEx: If present, treat <string> in –sk, –sv, –sd, and –ss as a regular expression
- literal: If present, the –sd and –ss search value is not interpreted as ASCII and Unicode strings
- nd: When true, suppress showing data when –sd or –ss is used. Default is FALSE.

## Other

- recover: If true, recover deleted keys and values. Default is TRUE
- nl: When true, ignore any transaction logs for dirty hives. Default is FALSE
- dt: Custom date/time format for displaying timestamps
- debug: Enable debug messages
- trace: Enable trace messages (VERY verbose)

## Simple searches

The two letter search options starting with ‘s’ are string search options. These options look for matches via ‘contains’ logic rather than ‘begins with’ or similar. For example, if you search for ‘cache’, the following keys would match if they existed in the Registry hive:

- Muicache
- Cache items
- UnCACHeD

Simple searches are not case sensitive.

To search for binary data in value data, simply string the hex characters you want to find together, separated by dashes (04-00-EF-BE for example).

```
.\RECmd.exe -f D:\Temp\UsrClassDeletedBags.dat --sd "04-00-EF-BE"
RECmd version 1.2.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/RECmd

Note: Enclose all strings containing spaces (and all RegEx) with double quotes

Processing hive 'D:\Temp\UsrClassDeletedBags.dat'

Header length is smaller than the size of the file.
hbin header incorrect at absolute offset 0x35000!!! Percent done: 82.81%
Extra, non-zero data found beyond hive length! Check for erroneous data starting at 0x35000!

Found 1 search hit in 'D:\Temp\UsrClassDeletedBags.dat'
Key: 'Local Settings\Software\Microsoft\Windows\Shell\BagMRU\0\0', Value: '0', Data: '74-00-31-00-00-00-00-0
0-33-3F-D9-83-11-00-55-73-65-72-73-00-60-00-08-00-04-00-EF-BE-EE-3A-85-1A-33-3F-D9-83-2A-00-00-00-B5-01-00-0
0-00-00-01-00-00-00-00-00-00-00-00-36-00-00-00-00-00-55-00-73-00-65-00-72-00-73-00-00-00-40-00-73-00-68-0
0-65-00-6C-00-6C-00-33-00-32-00-2E-00-64-00-6C-00-2C-00-2D-00-32-00-31-00-38-00-31-00-33-00-00-00-14-0
0-00-00'
```

This allows you to find signatures for common data structures ANYWHERE in the Registry. The binary signature used above is that of a BEEF0004 extension block, commonly used in ShellBags. It contains information such as MAC dates/times, MFT info, etc.

When using the sd and ss switches, the value data or slack will be converted to its equivalent ASCII and Unicode representation from the raw bytes. For example, if you searched for Ask, three searches would actually happen:

1. For the ASCII string itself
2. Raw data converted to ASCII string. A case insensitive search against this string is performed. If found, the position of the hit is used to extract the exact string that was hit on. This string is then converted back to bytes and reported as a hit.
3. Raw data converted to Unicode string. The rest happens as in step 2.

This allows string searches to find data regardless of encoding or case. If data is found in encoded form, the exact bytes making up the hit are highlighted. These bytes may differ from the searched for string if the capitalization was different.

If the –Literal switch is used with sd or ss, then only the first search is done behind the scenes. This allows you to look for specific byte patterns without RECcmd interpreting raw data or slack to ASCII or Unicode.

## Regular expression searches

When the –RegEx switch is present, the search term used is treated as a regular expression. Regular expression searches offer much more powerful capabilities to find things at the cost of having to follow a more complex set of rules when building search terms. Another tradeoff is that it can be slower depending on how complicated your RegEx is.

Enclose the RegEx in quotes to make sure the shell does not try to interpret anything in there.

As with simple searches, regular expression-based searches are case insensitive.

## Regular Expression examples

### Finding keys

To find all keys that contain ‘Microsoft.Bing’ followed by an F, H, or a W, then an o, use the following search:

```
RECcmd.exe -f "D:\temp\re\UsrClass 1.dat" --RegEx --sk "Microsoft.Bing[FHW]o"
```

```
Found 11 search hits in 'D:\Temp\UsrClass 1.dat'
Key: 'ActivatableClasses\Package\Microsoft.BingFoodAndDrink_3.0.4.212_x64_8wekyb3d8bbwe'
Key: 'Extensions\ContractId\Windows.BackgroundTasks\PackageId\Microsoft.BingFoodAndDrink_3.0.4.212_x64_8wekyb3d8bbwe'
Key: 'Extensions\ContractId\Windows.Launch\PackageId\Microsoft.BingFoodAndDrink_3.0.4.212_x64_8wekyb3d8bbwe'
Key: 'Extensions\ContractId\Windows.Protocol\PackageId\Microsoft.BingFoodAndDrink_3.0.4.212_x64_8wekyb3d8bbwe'
Key: 'Local Settings\Software\Microsoft\Windows\CurrentVersion\AppContainer\Storage\microsoft_bingfoodanddrink_8w
Key: 'Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Families\Microsoft.BingFoodAnd
Key: 'Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Families\Microsoft.BingFoodAnd
Key: 'Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Packages\Microsoft.BingFoodAnd
Key: 'Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Packages\Microsoft.BingFoodAnd
Key: 'Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\SystemAppData\Microsoft.BingFoodAndDrink_
Key: 'Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\SystemAppData\Microsoft.BingFoodAndDrink_
```

### Finding values

To find all values with names that contain either ‘AppName’ or ‘DisplayName’, use the following search:

```
RECcmd.exe -f "D:\temp\re\UsrClass 1.dat" --RegEx --sv "(App|Display)Name"
```

326 results were found, but due to the length of the output, only a few are shown below.

```

Key: 'Local Settings\Software\Microsoft\Windows\Shell\MuiCache', Value: 'C:\Program Files (x86)\VMware\VMware Workstation\vmware.exe.FriendlyAppName'
Key: 'Local Settings\Software\Microsoft\Windows\Shell\MuiCache', Value: 'C:\Program Files (x86)\Common Files\Microsoft Shared\VSEnv\VSLauncher.exe.FriendlyAppName'
Key: 'Local Settings\Software\Microsoft\Windows\Shell\MuiCache', Value: 'C:\Program Files (x86)\Microsoft Office\Office15\WINWORD.EXE.FriendlyAppName'
Key: 'Local Settings\Software\Microsoft\Windows\Shell\MuiCache', Value: 'C:\Program Files (x86)\Windows Media Player\wmplayer.exe.FriendlyAppName'
Key: 'Local Settings\Software\Microsoft\Windows\Shell\MuiCache', Value: 'C:\Program Files\Windows NT\Accessories\WORDPAD.EXE.FriendlyAppName'
Key: 'Local Settings\Software\Microsoft\Windows\Shell\MuiCache', Value: 'C:\Program Files\Altova\XMLSpy2015\XMLSpy.exe.FriendlyAppName'
Key: 'Local Settings\Software\Microsoft\Windows\Shell\MuiCache', Value: 'C:\Program Files (x86)\Just Great Software>EditPad Pro 7>EditPadPro7.exe.FriendlyAppName'
Key: 'Extensions\ContractId\Windows.Protocol\PackageId\Microsoft.BingSports_3.0.4.212_x64_8wekyb3d8bbwe\ActivatableClassId\AppeSports.AppYrrx553t7kt8vfxk9py4qdw3'
Key: 'Extensions\ContractId\Windows.Launch\PackageId\Microsoft.BingSports_3.0.4.212_x64_8wekyb3d8bbwe\ActivatableClassId\AppeSports.wwe', Value: 'DisplayName'
Key: 'Extensions\ContractId\Windows.BackgroundTasks\PackageId\Microsoft.BingSports_3.0.4.212_x64_8wekyb3d8bbwe\ActivatableClassId\Windows.Networking.BackgroundTrans
Key: 'Extensions\ContractId\Windows.BackgroundTasks\PackageId\Microsoft.BingSports_3.0.4.212_x64_8wekyb3d8bbwe\ActivatableClassId\Windows.Networking.BackgroundTrans
Key: 'Extensions\ContractId\Windows.BackgroundTasks\PackageId\Microsoft.BingSports_3.0.4.212_x64_8wekyb3d8bbwe\ActivatableClassId\Windows.Networking.ContentPrefetch

```

Notice that some of the hits are in red. This means they were recovered keys/values.

## Finding data

To find all values whose data contains ‘URL:bing’ followed by either an m, h, or s, use the following search:

```
RECcmd.exe -f "D:\temp\re\UsrClass 1.dat" --RegEx --sd "URL:bing[mhs]"
```

```

Found 3 search hits in 'D:\Temp\UsrClass 1.dat'
Key: 'binghealthnfitness', Value: '(default)', Data: 'URL:binghealthnfitness'
Key: 'bingmaps', Value: '(default)', Data: 'URL:bingmaps'
Key: 'bingsports', Value: '(default)', Data: 'URL:bingsports'

```

For more examples, run RECcmd.exe without any command line arguments.

All regular expressions must of course be valid .net regular expressions. Different flavors of RegEx providers allow for different syntax, so be sure to use the proper syntax.  
RegEx tutorials for .NET.

- \* <https://msdn.microsoft.com/en-us/library/az24scfc%28v=vs.110%29.aspx>
- \* <http://regexhero.net/reference/>
- \* <https://msdn.microsoft.com/en-us/library/hs600312%28v=vs.110%29.aspx>
- \* <http://www.codeproject.com/Articles/9099/The-Minute-Regex-Tutorial>
- \* <http://www.systemtextregularexpressions.com/help>

RegExBuddy is an awesome tool for building and testing RegEx against data sets.

## Batch mode

Batch mode works by crafting a YAML formatted document and passing it into RECcmd via the –bn switch. Data will be saved to the directory specified by –csv. Batch mode also allows for the use of plugins. These plugins are the same plugins as found in Registry Explorer and work the same way. When used by RECcmd, the data from the plugin will be normalized into a standard format for CSV output. With that said, when a plugin is used to process a key or key/value, the data generated by the plugin is also saved out to a CSV. In this way, it is very similar to exporting the data from Registry Explorer (albeit to Excel vs CSV).

Several example batch file specifications ship with RECcmd, but let’s take a look at one in more detail.

```
.....10.....20.....30.....40.....50.....60.....70.....80.....90
1 Description: Sample RECcmd batch file
2 Author: Eric Zimmerman
3 Version: 1
4 Id: ab13eb5f-31db-5cdc-83df-88ec83dc7a
5 Keys:
6 -
7     Description: Typed URLs
8     HiveType: NTUSER
9     Category: Browser history
10    KeyPath: Software\Microsoft\Internet Explorer\TypedURLs
11    Recursive: false
12    Comment: "some comment"
13 -
14     Description: WordWheelQuery
15     HiveType: NTUSER
16     Category: User searches
17     KeyPath: Software\Microsoft\Windows\CurrentVersion\Explorer\WordWheelQuery
18     Recursive: true
19     Comment: "search comment"
20 -
21     Description: Network MRU
22     HiveType: NTUSER
23     Category: Network shares
24     KeyPath: Software\Microsoft\Windows\CurrentVersion\Explorer\Map Network Drive MRU
25     ValueName: MRUList
26     Recursive: false
27     Comment: "value only"
28 -
29     Description: UserAssist
30     HiveType: NTUSER
31     Category: Execution
32     KeyPath: Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist
33     Recursive: true
34     Comment: "user assist"
```

The specification for a batch file is as follows:

## Header

- **Description:** A general description of what this batch file is going to find
- **Author:** Name of made this batch file (can me more too, like contact info)
- **Version:** A version number that should be incremented as changes happen
- **Id:** A unique (across all other batch files) GUID that identifies this batch file
- **Keys:** A list of things to look for

## Keys collection

Each entry consists of:

- **Description:** A user-friendly description of what this key will find. Can be anything from the key name to a friendlier description of what it means, etc.

- **HiveType:** The type of hive this entry corresponds to. Valid choices are: NTUSER, SAM, SECURITY, SOFTWARE, SYSTEM, USRCLASS, COMPONENTS, BCD, DRIVERS, AMCACHE, SYSCACHE
- **KeyPath:** The path to the key to look for
- **ValueName:** OPTIONAL value that, when present, is looked for under KeyPath
- **Recursive:** Whether to process KeyPath recursively or not
- **Comment:** Like Description in that you can add various things here that end up in the CSV

While the example above only includes NTUSER, you can mix and match as many different hive types as you like (again, see the examples) so you can extract data from NTUSER hives, SYSTEM, and SOFTWARE for example.

To use batch mode, supply the file to the -bn switch, along with -csv to tell RECmd where to save results:

```
RECmd.exe --bn D:\Code\RECmd\RECmd\BatchExampleUserAssist.reb -f C:\Temp\NTUSER_dblake.DAT --nl --csv C:\Temp
```

We only searched a single hive here, but if we used -d vs -f, we could search entire directories for hives to process.

```
Found key 'ControlSet001\Services'!
Found 655 key/value pairs across 1 file
Total search time: 0.949 seconds

Saving batch mode CSV file to 'C:\Temp\20190109202053_RECmd_Batch_BatchExampleServices_Output.csv'
```

This results in a few files being created, one for the batch data, which is named after the name of the batch file passed in, and the other is named after the Plugin that was executed (Services) along with the full path to the hive for that particular file.

Name	Size
20190109202053_RECmd_Batch_BatchExampleServices_Output.csv	256 KB
20190109202053_Services_C_Temp_SYSTEM_loneWolf.csv	157 KB
SYSTEM_loneWolf	12.2 MB

This lets you know exactly what was found (Services) and where it was found (C:TempSYSTEM\_loneWolf). If you had 10 NTUSER.DAT hives and 3 plugins ran on each, you would end up with 31 files generated, 1 for the main batch mode CSV, and 30 for the plugins.

The main batch mode CSV also contains a pointer to the detailed plugin CSV so if you do need to drill down you can do so easily.

## Version changes

For versions after 1.2.0.0, see ChangeLog at <https://ericzimmerman.github.io>

### Version 1.2.0.0

NEW: Updated controls and nuget

NEW: Display deleted values with red gradient in values list

NEW: Display deleted values with non-resident data with purple gradient when the data record that value points to has been reallocated to another cell somewhere else.

NEW: RECmd completely rewritten, adding support for plugins and batch mode

CHANGE: Updated back end Registry parser with tweaks and new features

CHANGE: Updated plugin interfaces to support batch mode in RECmd

FIX: Handle fringe errors related to save paths, loading bookmarks, etc

### Version 1.1.0.0

NEW: Updated controls, move to Fody vs LibZ for single exe

CHANGE: Show slack tab when a plugin is used

FIX: Fixes for handle leaks

FIX: Handle NK record referencing deleted value per @errnofail's research

FIX: Various tweaks and fixes

### Version 1.0.0.0

NEW: Display warning when Header sequence 1 != sequence 2. this means the hive is dirty and there are transactions in the log file(s) which has not been committed to the hive.

NEW: Updated controls and dependencies

NEW: For non-RegBinary values, display 'Binary viewer' tab which contains the value's raw data.

NEW: When grouping in a grid, show the total number of items in the group in the header

NEW: Added plugins for Bluetooth, Bam, Dam, RecentApp, etc.

NEW: Detect dirty hives and offer to replay transaction logs (new format only) so the hive contains all data contained in the transaction log(s).

CHANGE: RegNone values are treated like RegBinary and are shown in a hex viewer which allows for interacting with bytes more easily

CHANGE: Merge some changes from other users to plugins (SAM)

FIX: Update existing plugins to support new cases (AppCompatcache), timestamp to UTC (Office), etc.

### Version 0.9.0.0

NEW: Added Raw Value property to non-RegBinary values that contains the bytes that make up the value. This is useful for copying out into other programs like DCode, etc.

NEW: Plugins added for Known networks (SOFTWAREMicrosoftWindows NTCurrentVersionNetworkList), WordWheelQuery, TypedURLs (including TypedURLsTime), Services, Terminal services client (RDP history), DHCPNetworkHint,

NEW: Added Options | Convert selected | To ROT-13 in Find window. This allows for searching for things ROT-13 encoded like UserAssist, etc without having to rely on a plugin

NEW: Added '# subkeys' column to Registry Hives and Available bookmarks trees

NEW: Added 'Selected hive' to left side of status bar that tracks the name of the hive currently selected. Double clicking copies full path of hive to clipboard

NEW: More bookmarks

NEW: Add indicator for ‘Deleted’ in search results

NEW: Added ‘Data interpreter’ option to Values context menu. This allows you to view and decode the raw value data in a wide variety of formats (integer to EPOCH date, etc.)

NEW: Much better filtering options in trees and grid including Excel like filtering

NEW: Updated controls

NEW: Holding CTRL while right clicking a node in Registry hives tree will automatically expand all child nodes (saves time over using context menu)

NEW: Project support added. You can now create projects based on currently loaded hives and reload projects as needed

NEW: Add File | Unload all hives option

NEW: More data interpreter conversions

CHANGE: Allow for cell selection vs entire rows in Values grid

CHANGE: Allow for scrollbar on tree so all columns can be seen

CHANGE: User created bookmarks now show up in the Available bookmarks tab in Blue (bold) font to differentiate them from Common bookmarks

CHANGE: Absolute path to active Registry hive is now prepended to Key path on Copy via context menu in trees and to Value summary in Values grid

CHANGE: Add group membership and password hints to SAM plugin

FIX: Plugins updated based on test data

FIX: Save Datetime format and load it on subsequent starts

FIX: Bug fixes

Version 0.8.1.0

NEW: Change to .net 4.6

NEW: Added exporting of values to Excel, TSV, PDF, and HTML via key context menu (under Export | Values). Data is exported exactly as shown in Values grid (this lets you hide columns, reorder, sort, etc. before export)

NEW: Plugin support added.

NEW: Added View | Plugins to explore available plugins

NEW: Added Base64 to data interpreter under Strings section

NEW: Added Tools | Preferences

NEW: Option to show (and therefore export) RegBinary values as Base64 strings (enabled in Tools | Preferences)

NEW: Option to show (and therefore export) value slack as Base64 strings (enabled in Tools | Preferences)

NEW: Option to set custom date/time format for timestamps

NEW: For RegUnknown value type, show the actual value of the Registry Type in hex and decimal.

NEW: Ability to double click offset in hex viewers to jump to the offset in either decimal or hex

NEW: When a plugin is added for a key or value, make it the active tab

NEW: Hex viewer allows for selecting bytes and copying as hex, ANSI string (Windows 1252 code page), or Unicode string

NEW: When exporting value data, offer exporting in binary or string format

NEW: Allow for searching for many terms at once vs one at a time in Find dialog

NEW: Change messages count background color to yellow when there are warning messages and red when there are error messages. This color will be cleared when the Messages window is viewed.

CHANGE: Disable Bookmarks menu when on Available bookmarks tab

CHANGE: Clear any active filters before selecting bookmarked key

CHANGE: Set focus to last used search type on Find form

CHANGE: Sort bookmarks by name

CHANGE: Load hives when they do not have an nk record with a HiveRootEntry flag set. When this happens, an alternate method is used to find the root key

CHANGE: Put the newest search history items at the top of the list

CHANGE: Don't trust Header length when looking for hbins as sometimes Header length is wrong

CHANGE: Values grid filters use 'contains' vs 'starts with' as default

FIX: Add missing tooltip to Literal checkbox on Find form

FIX: Update hex position in hex type viewer when moving up and down rows vs only left and right

FIX: Correct issue when selecting hits in Find panel if the Registry keys tree was sorted when a virtual key existed (Associated Registry keys for example)

FIX: Handle rare issue when building virtual keys for 'Associated deleted records' where there is an active key and a recovered deleted key with the same name

FIX: Lots of tweaks and miscellaneous fixes

#### Version 0.7.1.0

##### RECmd changes

New: Added -Dir switch. This recursively searches for hives in a given directory and searches each of them

##### Registry Explorer changes

New: Registry Explorer can now function as a "default application" in that you can associate RE with \*.dat and then double click hives. This also allows for setting up RE in other apps like X-Ways as an external viewer, dragging and dropping hives onto RE shortcut/executable, etc.

New: Added Check for updates to About menu

#### Version 0.7.0.0

As of 0.7.0.0, Registry Explorer and RECmd are included together.

##### RECmd changes

NEW: Added -Literal switch. When present, -sd and -ss switches will not be interpreted

NEW: Added -ss switch for searching Value slack space

NEW: Search terms are now highlighted in search results. Edit nlog.config to adjust colors for foreground and background

NEW: Added -RegEx switch. When present, treat <string> in -sk, -sv, -sd, and -ss as a regular expression

NEW: If nlog.config is missing, add default config and warn user

CHANGE: Switches are NOT case sensitive any more

CHANGE: Remove RegEx specific switches (See -RegEx above)

CHANGE: Tweak command line option descriptions

**CHANGE: Updated nlog**

See here for changes in version 0.6.1.0.

**Registry Explorer changes**

This version is pretty much a complete rewrite under the hood. This was done to address performance issues due to initial (bad) design decisions.

Hive processing is fully asynchronous, but very large hives can take a few seconds to display once the hive is loaded. This is due to the need to load all

**NEW:** Full support for searching including key names, value names, and value data, both with simple searches and RegEx. Searching based on last write timestamps is supported as well

**NEW:** Fully asynchronous loading of hives which keeps the GUI responsive, even when loading 100+ MB hives (I am looking at you SOFTWARE hive)

**NEW:** Tech details hex editors now update with offset and selection length when bytes are selected

**NEW:** Added value context menu to copy value summary (a combination of name, type, and data), name, type, data, and slack to clipboard

**NEW:** Add value context menu to export data and slack to a file

**NEW:** Settings for things persist

**NEW:** Search strings are remembered and autopopulate when typing on the Find form. Use the Tools menu to clear

**NEW:** Added Convert | To hex ASCII and To hex Unicode to Find. This allows you to look for encoded strings in value data without having to manually convert strings to hex

**NEW:** Allow deleting of user created bookmarks in Bookmark Manager via Ctrl-Delete

**NEW:** Added context menu to Available bookmarks (Copy, expand/collapse, tech details) that work the same as the 'Registry hives' context menu options

**NEW:** Added 'Jump to key' context menu item on Available bookmarks tab that will select the hive's key on the 'Registry hives' tab

**NEW:** More hot keys added to main/context menus

**NEW:** Added 'Root key name' to Tech details | Hive details properties and strip root key from Tech details window title to save space

**NEW:** Added Export 'Registry hive' menu to File menu. This exports the tree exactly as it is shown to the selected format

**NEW:** Enable/disable expand/collapse subkey options depending on the expanded state of the selected key

**NEW:** Save positions of vertical and horizontal splitters

**NEW:** Trees and grids all save settings (sorting, filtering, conditional formatting rules) between sessions

**NEW:** Save size of main form

**NEW:** Improved hex editor control for RegBinary keys. Added offset, selection length, and data interpreter

**NEW:** Added 'Show associated deleted records' and 'Show unassociated deleted records' to Options menu

**NEW:** Added 'Slack viewer' tab for values that have slack space

**NEW:** Added 'Show parent keys when filtering' to options menu. Turning this OFF shows only the

keys that match the filter. When ON, parent keys to keys matching the filter are also shown  
NEW: Added a Total messages counter to lower status bar (far right) that indicates the total number of messages available on the Messages form

NEW: Added skinning support. Active skin can be changed from the Options menu

NEW: Added icon for Registry hive in the Registry hives tree to visually separate it from keys

NEW: Make hive name bold to make it stand out from keys

NEW: Tech details info can be copied via Ctrl+C (just the value) or Ctrl+Alt+C (Name: Value)

NEW: All hex viewers now support Ctrl+C to copy selected bytes to clipboard

NEW: Search for minimum value sizes added

NEW: Search in value slack added

CHANGE: Allow resizing of window below 800x600

CHANGE: Drag and dropping of hives supported on any of the 3 main sections of Registry Explorer

CHANGE: Status bars adjusted. Added options to hold Shift when double clicking in order to copy different parts of the key/value

CHANGE: Add vertical scroll bar to Technical details hex editors

CHANGE: Rename tree context menus from 'child nodes' to 'subkeys'

CHANGE: Hide Messages form by default since things load and process faster when its hidden

CHANGE: Icon for existing key placeholder in Associated deleted records updated

CHANGE: Icon for Associated deleted records updated

CHANGE: Made legend icons bigger

CHANGE: Bookmarks manager now allows editing/deleted both common and user created bookmarks

FIX: Bug fixes in Registry parser (yay unit tests)

FIX: Show SK record in Technical details form

#### Version 0.2.0.0

NEW: Added new tab in upper left, Available bookmarks, that shows all available bookmarks across all loaded Registry hives

NEW: Added 'Technical details' option to context menu. Use this to view all the down and dirty details about a key including its bytes, its security key, subkeys, values, etc. This provides an easy to use way to explore and validate Registry tools

NEW: Added several hotkeys for commonly used key context menu items

NEW: Allow exporting of keys either individually or recursively to .reg format via the context menu

NEW: Add 'Collapse all hives' button to status bar.

NEW: Added more bookmarks

CHANGE: Prevent illegal file name characters in category names (\, /, |, and so on). Any illegal characters will be replaced with an underscore

CHANGE: Nlog logging added

CHANGE: Registry parsing is now ~150% faster and memory usage reduced by 40-80%

CHANGE: Prevent the same hive from being loaded more than once

CHANGE: Expand the top level node after loading a hive

CHANGE: Hide or unhide all matching keys in all open hives vs only the active hive

FIX: When removing keys from auto hide list, remove any hidden keys in the tree that match as well

FIX: Actually export the timestamp when exporting Messages

FIX: GUI polish

Version 0.1.8.0

Initial release

#### Appendix A – Contributors

The following people have contributed in one way or another during the development and refinement of SBE

- Devon P. Ackerman @aboutdfir
- David Cowen @HECFBlog
- Dan Pullega @4n6k
- Jerod Alexander @jerod
- Willi Ballenthin @williballenthin
- Maxim Suhanov @errno\_fail

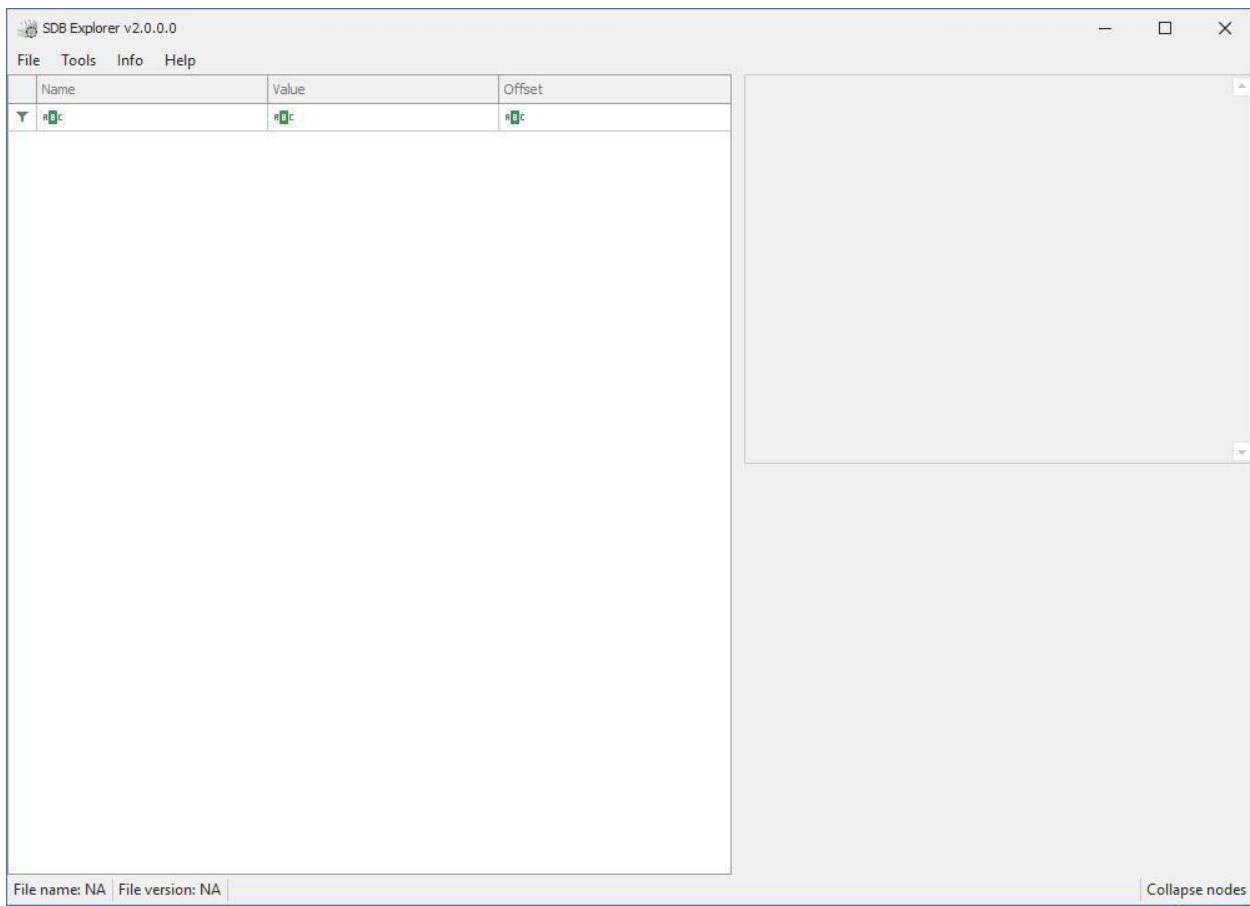
#### Appendix B – Additional resources

- <http://binaryforay.blogspot.com/>: Contains detailed postings on the internal workings of the Registry
- <https://github.com/EricZimmerman/Registry>: Source code for the back-end parser used in Registry Explorer
- <https://github.com/EricZimmerman/RECmd>: Source code for RECmd
- <https://github.com/EricZimmerman/RegistryPlugins>: Source code for all Registry Explorer plugins

# SDB Explorer

## SDB Explorer Introduction

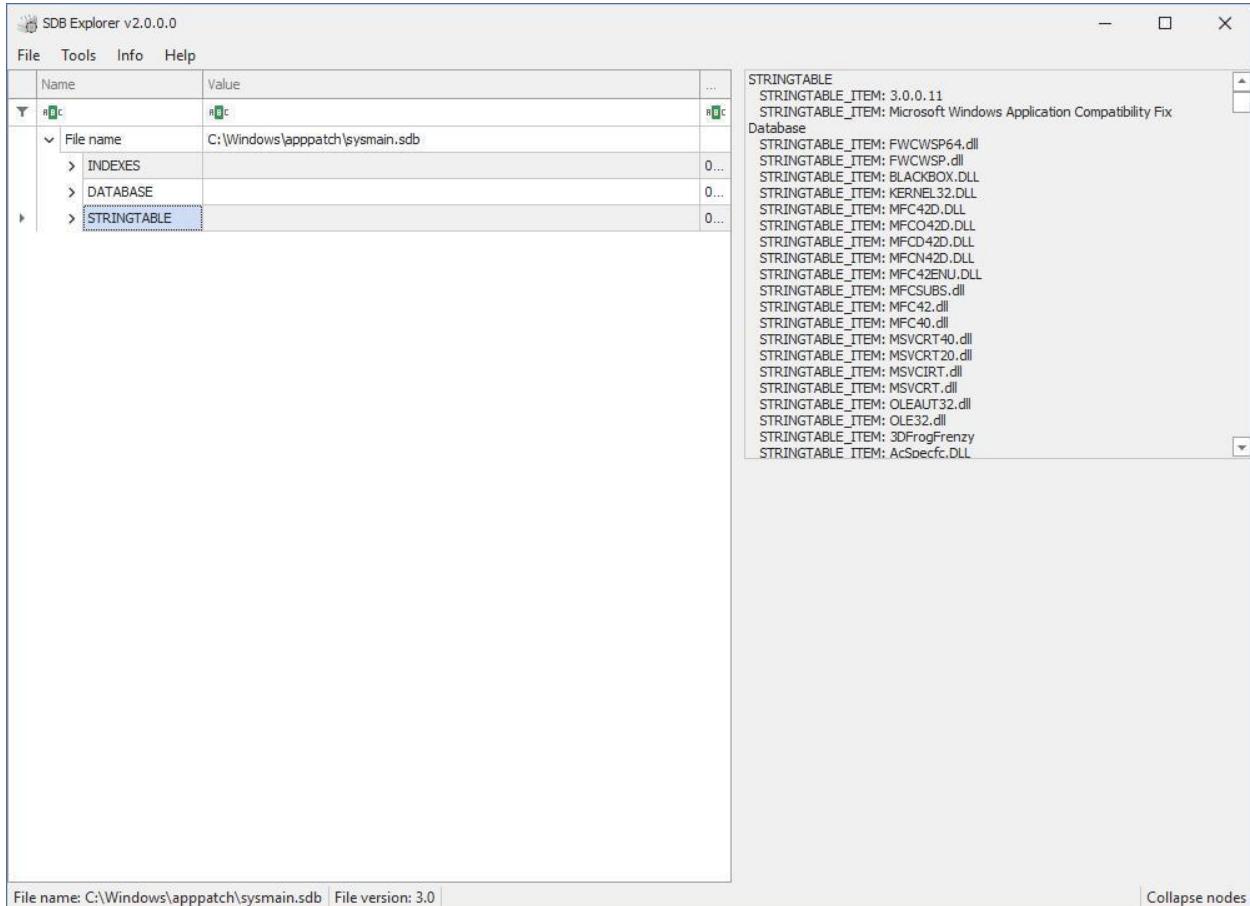
SDB Explorer is a tool created by Eric Zimmerman that can be used by forensic examiners to examine shim database files (.sdb). Please check out Microsoft's official documentation on Application Compatibility Databases [here<sup>159</sup>](#).



SDB Explorer

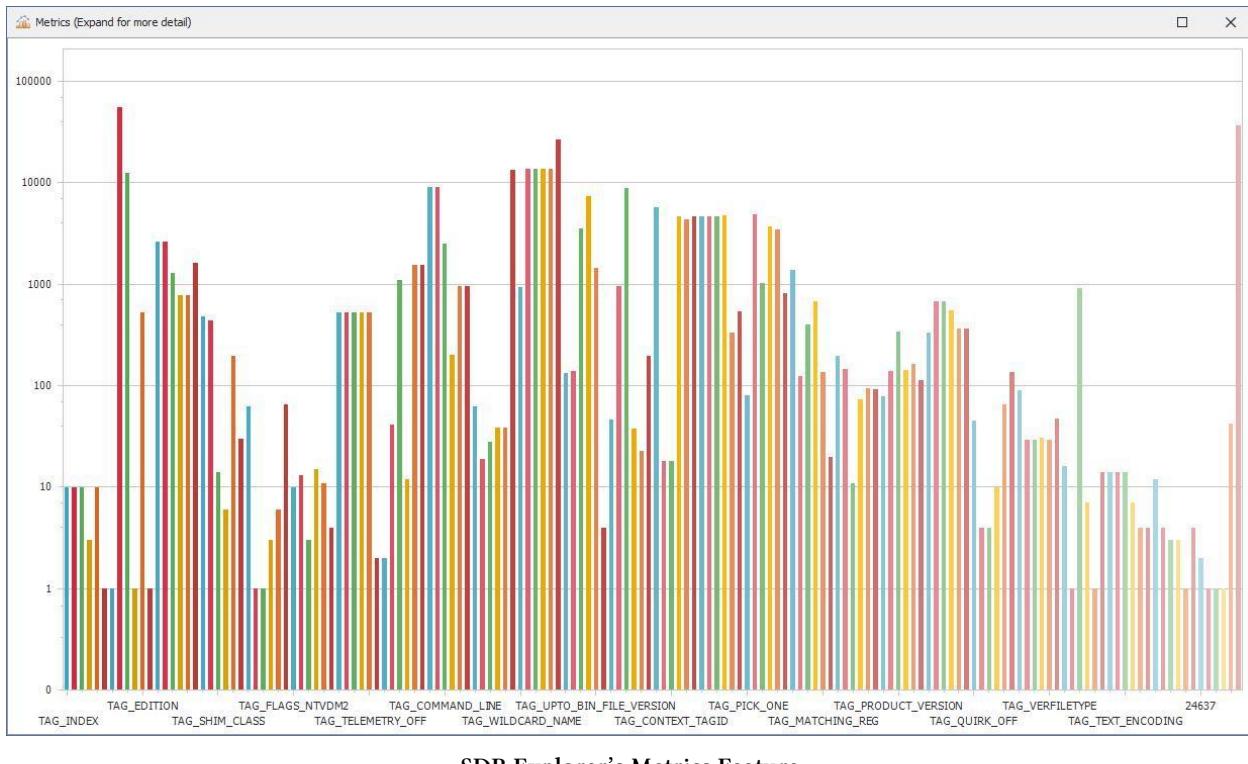
<sup>159</sup><https://learn.microsoft.com/en-us/windows/win32/devnotes/application-compatibility-database?redirectedfrom=MSDN>

Below is a screenshot of SDB Explorer with a shim database file loaded.



SDB Explorer with a loaded Shim Database

Using the Info -> Metrics menu, examiners can look at a visual representation of the data present within a loaded shim database.



SDB Explorer's Metrics Feature

## SDB Explorer References

### Blog Posts

#### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- Introducing SDB Explorer<sup>160</sup>

### Download SDB Explorer

SDB Explorer can be downloaded from <https://ericzimmerman.github.io/#!index.md>

<sup>160</sup><https://binaryforay.blogspot.com/2018/02/introducing-sdb-explorer.html>

# Shellbags Explorer



## Revision history

2014-11-21 Rev. 1 – Initial release

2022-07-25 Rev. 2 - Leanpub release

## Requirements

ShellBags Explorer requires Microsoft .net framework version 4.5.1 full runtime or greater to be installed. It is available at <http://www.microsoft.com/en-us/download/details.aspx?id=40779>.

## What are ShellBags?

ShellBags are of great forensic value from Intrusion investigations to Malware Causality examinations to timeline generation in that they are a perfect example of Locard's exchange principle - “a perpetrator will bring something into a crime scene and leave with something from it.”

By design, Microsoft® Windows may track a cyber intruder’s actions on a host system after compromise if the actor uses RDP or other Remote Connection controls and/or Windows Explorer to drop binaries onto the system, to access network resources, browse compressed archives, etc. Additionally, all directory traversal done with Windows Explorer is tracked and maintained in the registry. This data includes multiple timestamps and other pieces of information that provide context

and can be used to show knowledge and intent when it comes to accessing directories or other resources on a computer.

From the anti-forensics standpoint, an absence of ShellBag entries may suggest system cleaning or overt action by an end-user and potential sophistication of the actor. Lastly, bad actors are becoming smarter every day - not every forensic examination will contain evidence in plain sight/allocated space. ShellBags, along with other artifacts, may point to evidence that existed at one point in time or may assist the examiner in looking at the broader picture when only a piece is known.

ShellBags are a set of Windows Registry keys located in NTUser.dat and USRClass.dat registry hives (primarily usrclass.dat) that maintain view, icon, position, size (and other attributes) of folders when using Windows Explorer. They are likely to persist information even when the original directories, files, and physical devices have been removed from the system and due to this, can serve as a “history” of sorts into data that was previously on a system but may have since been removed. When combined with volume shadow copies (specifically, older copies of registry hives), significant insight into a user’s activity can be gleaned.

## ShellBags location in the registry

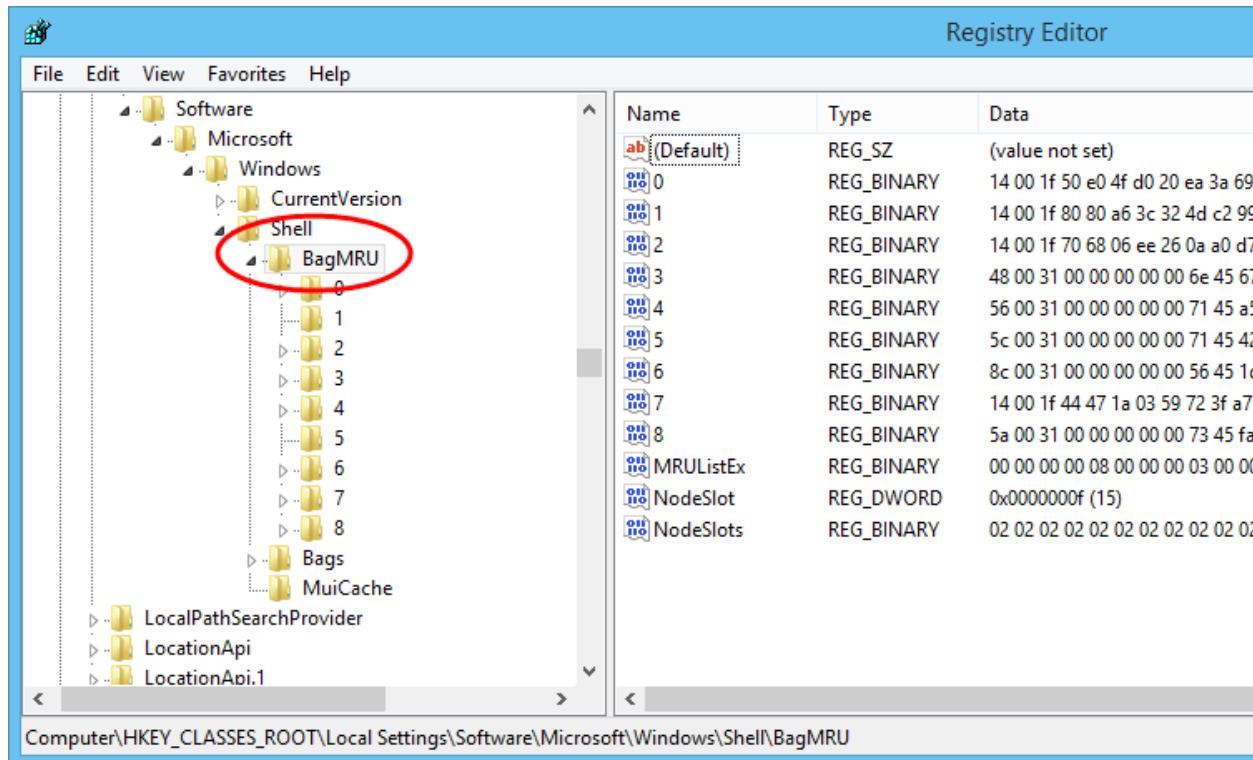
In Windows Vista and newer (including server operating systems based on the same technology), ShellBag data is located in the following registry keys and subkeys:

- \* HKEY\_CURRENT\_USERSoftwareMicrosoftWindowsShellBags
- \* HKEY\_CURRENT\_USERSoftwareMicrosoftWindowsShellBagMRU (ntuser.dat)
- \* HKEY\_CURRENT\_USERSoftwareMicrosoftWindowsShellNoRoamBags
- \* HKEY\_CURRENT\_USERSoftwareMicrosoftWindowsShellNoRoamBagMRU
- \* HKEY\_CURRENT\_USERSoftwareClassesLocal SettingsSoftwareMicrosoftWindowsShellBagMRU (usrclass.dat) (Found on disk at C:\Users<profile\_name>\AppData\Local\Microsoft\Windows)
- \* HKEY\_CURRENT\_USERSoftwareClassesLocal SettingsSoftwareMicrosoftWindowsShellBags

SBE currently looks at the bold keys. The “Bags” keys are not looked at as they have been determined to relate to window position, size, etc,

## Using RegEdit to view ShellBag data

The regedit utility included with Windows can be used to view ShellBag data. After starting regedit, navigate to the HKEY\_CURRENT\_USERSoftwareClassesLocal SettingsSoftwareMicrosoft-WindowsShellBagMRU key (shown below).

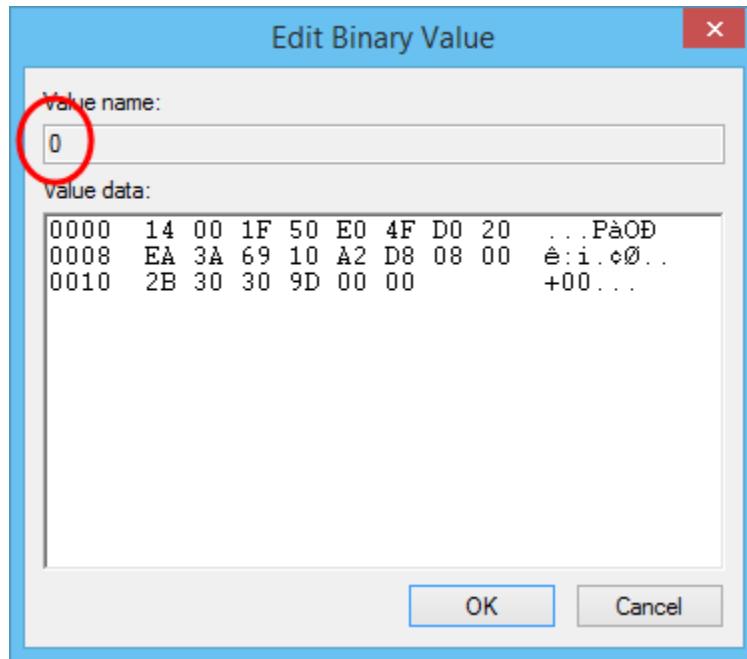


On the left, registry keys are displayed. After selecting a key, the values for that key are displayed in the right pane.

The values on the right correspond to ShellBags that are children of the selected key. In the screen shot above, it is the BagMRU key. The BagMRU key represents the “Desktop” in Windows and all ShellBags will exist under the Desktop. In other words, the Desktop is the topmost Shellbag.

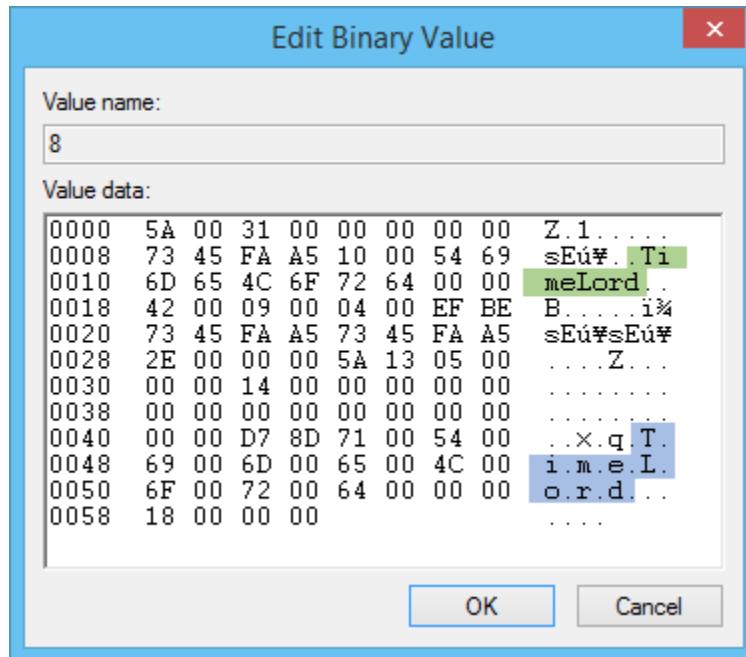
## ShellBag binary contents

Each ShellBag contains binary data that defines what the ShellBag represents. Double clicking on a value will bring up an editor showing the binary data as seen below.



In the example above, the “0” value has been opened. The contents of the ShellBag are shown in hexadecimal. The example above contains a GUID which corresponds to the “My Computer” folder. There are hundreds of GUIDs that map to directories, control panel items and categories, etc. SBE contains hundreds of GUID mappings to human readable formats.

Some ShellBags will contain strings (both ANSI and Unicode) representing things such as directory names or UNC paths. In the image below, a directory named “Timelord” was accessed. The ‘short name’ is highlighted in green and the ‘long name’ is highlighted in blue.

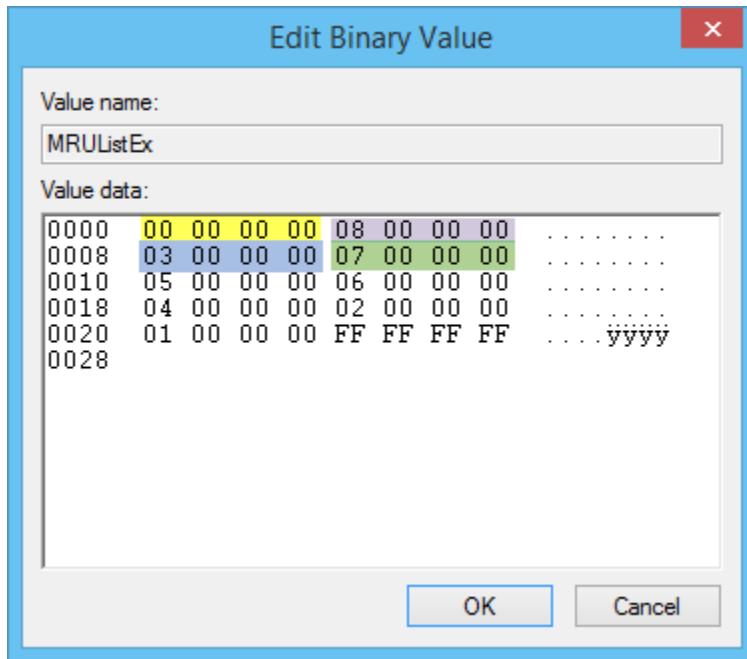


While not obvious, there are also several dates and times embedded in the binary data as well as MFT entry and sequence numbers. It is also possible to determine the file system this directory existed on based on the contents of the ShellBag. In the example above, the directory existed on an NTFS file system with MFT Entry Number 332634 and MFT Sequence Number 20 and was last accessed on 11/19/2014 at 8:47:52 PM +00:00!

## MRUListEx

The MRUListEx value is the Most Recently Used list and reflects the order the ShellBags were opened with the most recently opened bag being listed first. As ShellBags are opened, the MRUListEx values are shifted to the right and the most recently opened value is added to the leftmost position.

Double clicking the MRUListEx value brings up an editor showing the binary data contained in the value.



Each entry in the MRU list is 4 bytes long (little endian). In the image above, the first 4 MRU positions have been highlighted with different colors.

Based on what is in the MRUListEx key above, the order the ShellBags under BagMRU were opened is 0, 8, 3, 7, 5, 6, 4, 2, 1. The last entry in MRUListEx is always FF FF FF FF.

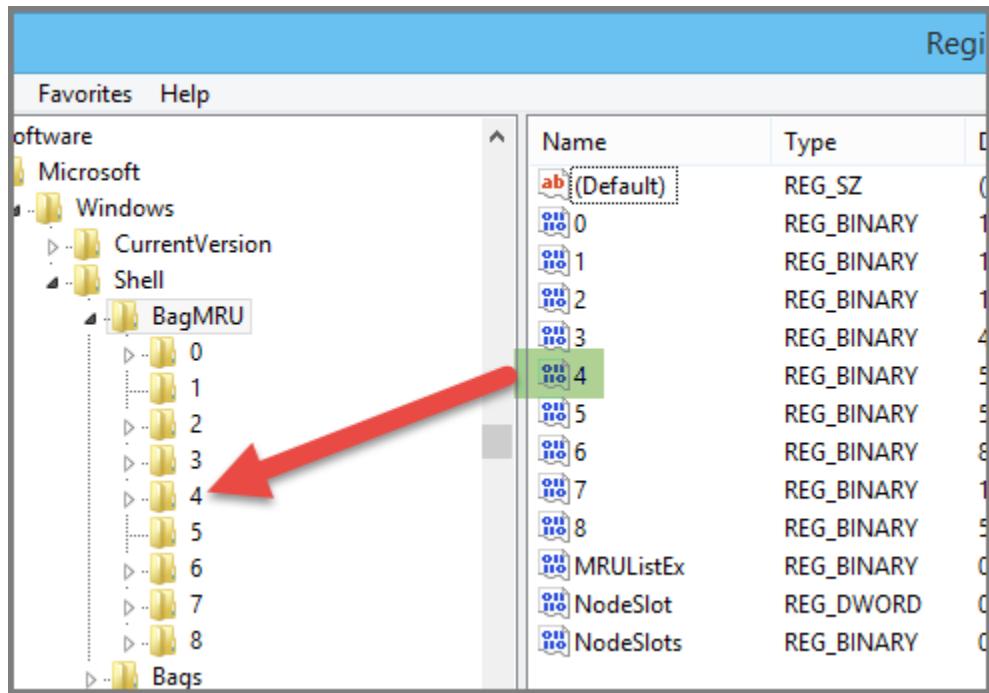
## NodeSlot

There is also a NodeSlot value that links to a subkey under the Bags key. This subkey contains such things as the sorting, icon size, and other properties for a given directory. The NodeSlot value is for the currently selected key and not the ShellBags present under a given key.

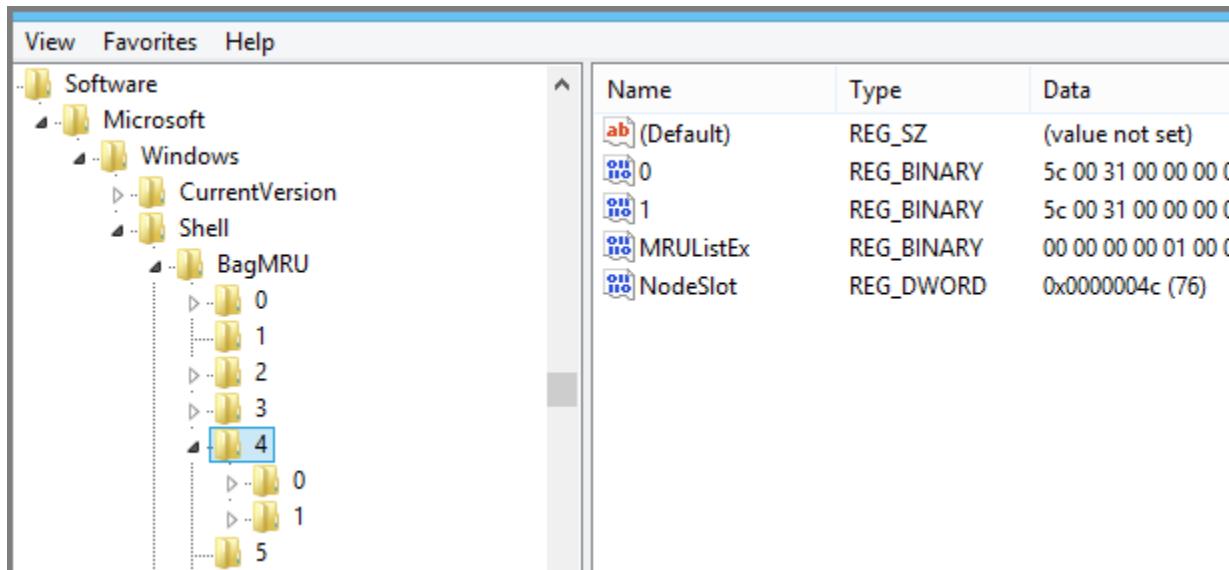
## Subkeys and their relation to values

In the screen shot on page 6, the BagMRU key has been expanded and circled in red. Under it are subkeys 0-8. Notice there are also values with the same name on the right.

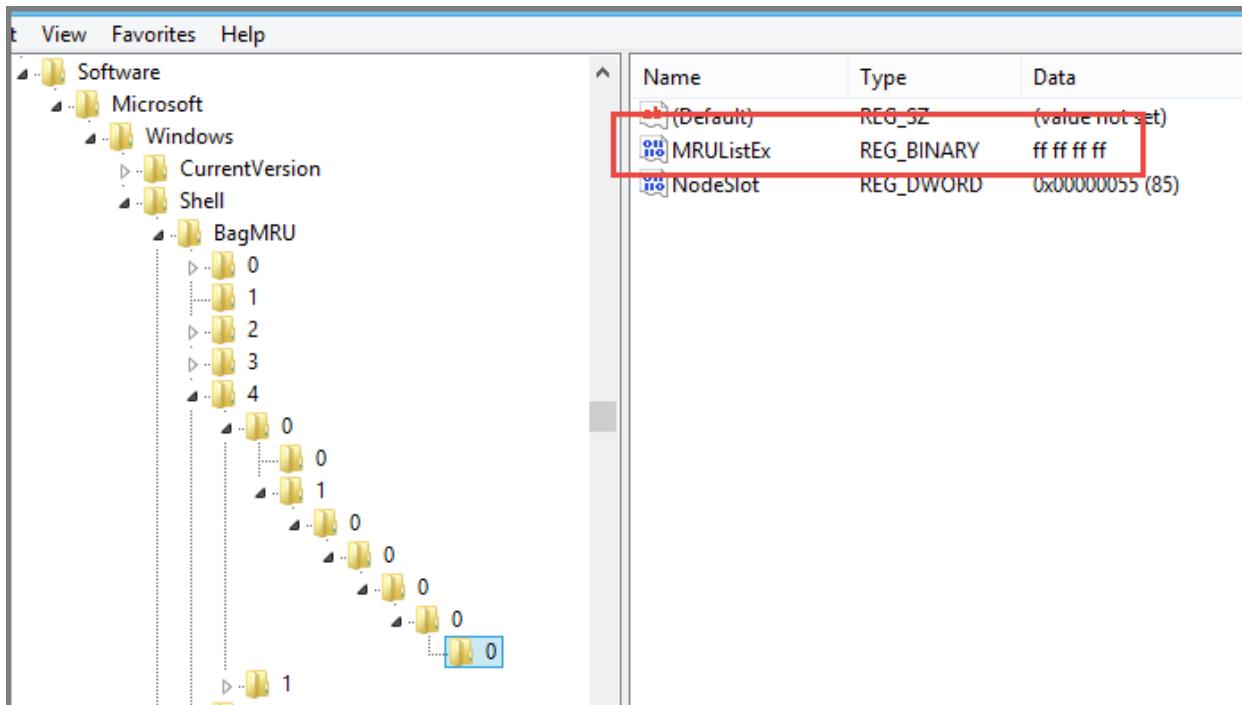
Subkeys with the same name as the value will contain child objects for that value. Recall each value represents a ShellBag. A ShellBag can be the Desktop item, a Control Panel Category, a Control Panel item, a drive letter, or a directory (there are other possibilities as well).



In the image above, the ShellBag with a value of 4 is the “parent” ShellBag for any ShellBags found in the key with the same name. Selecting the key with name 4 results in the following.



As you can see, the pattern continues as we drill down into child keys. If we continue to drill down we will eventually run out of child keys as seen in the next screen shot.



We know we are at the “bottom” because there are no more child keys and the MRUListEx value is FF FF FF FF.

## LastWrite times and their value in ShellBags analysis

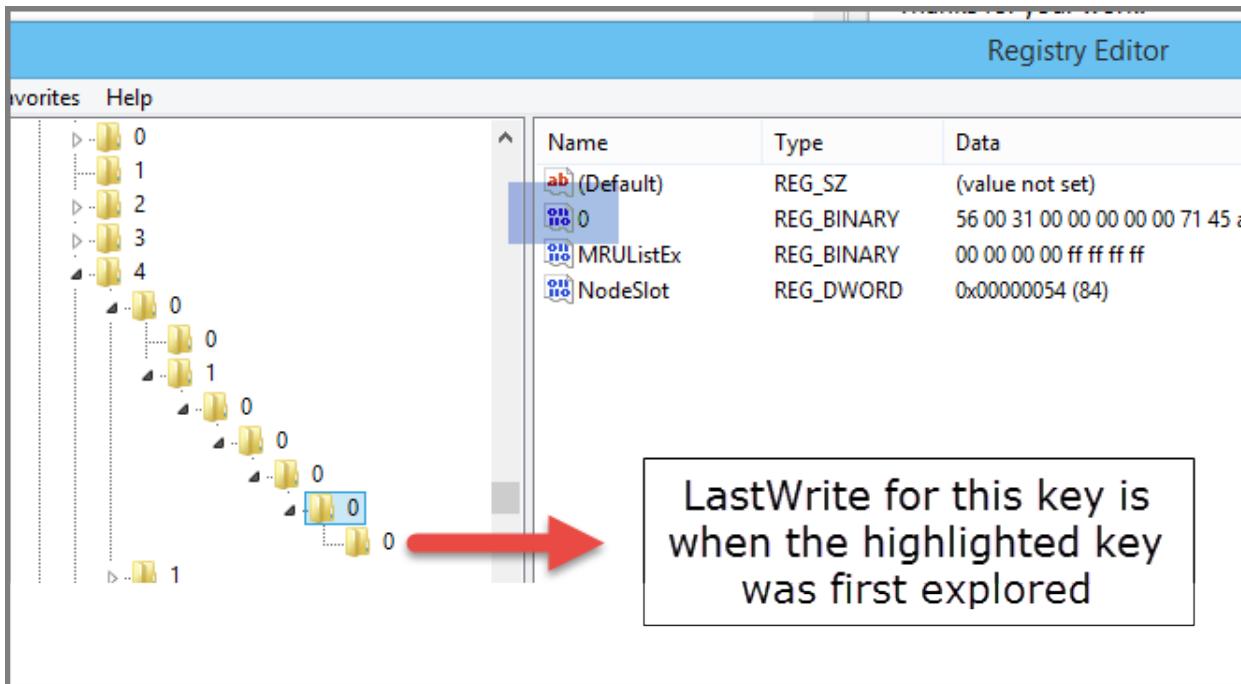
Whenever a value is changed in a registry key, the key’s LastWrite timestamp is updated. Regedit does not display these timestamps, but other forensic programs such as X-Ways Forensics or TZWorks yaru can. When using yaru to look at the same key as shown above, we can see the key was last written on 11/17/2014 at 18:14:40 UTC.

4	4	2	2014-11-17 18:18:03
0	4	2	2014-11-17 18:18:03
0	2	0	2014-11-17 18:13:46
1	3	1	2014-11-17 18:14:33
0	3	1	2014-11-17 18:14:38
0	3	1	2014-11-17 18:14:39
0	3	1	2014-11-17 18:14:40
0	3	1	2014-11-17 18:14:40
0	2	0	2014-11-17 18:14:40
1	3	1	2014-11-17 18:17:26

## Using LastWrite values to determine First Explored and Last Explored dates and times

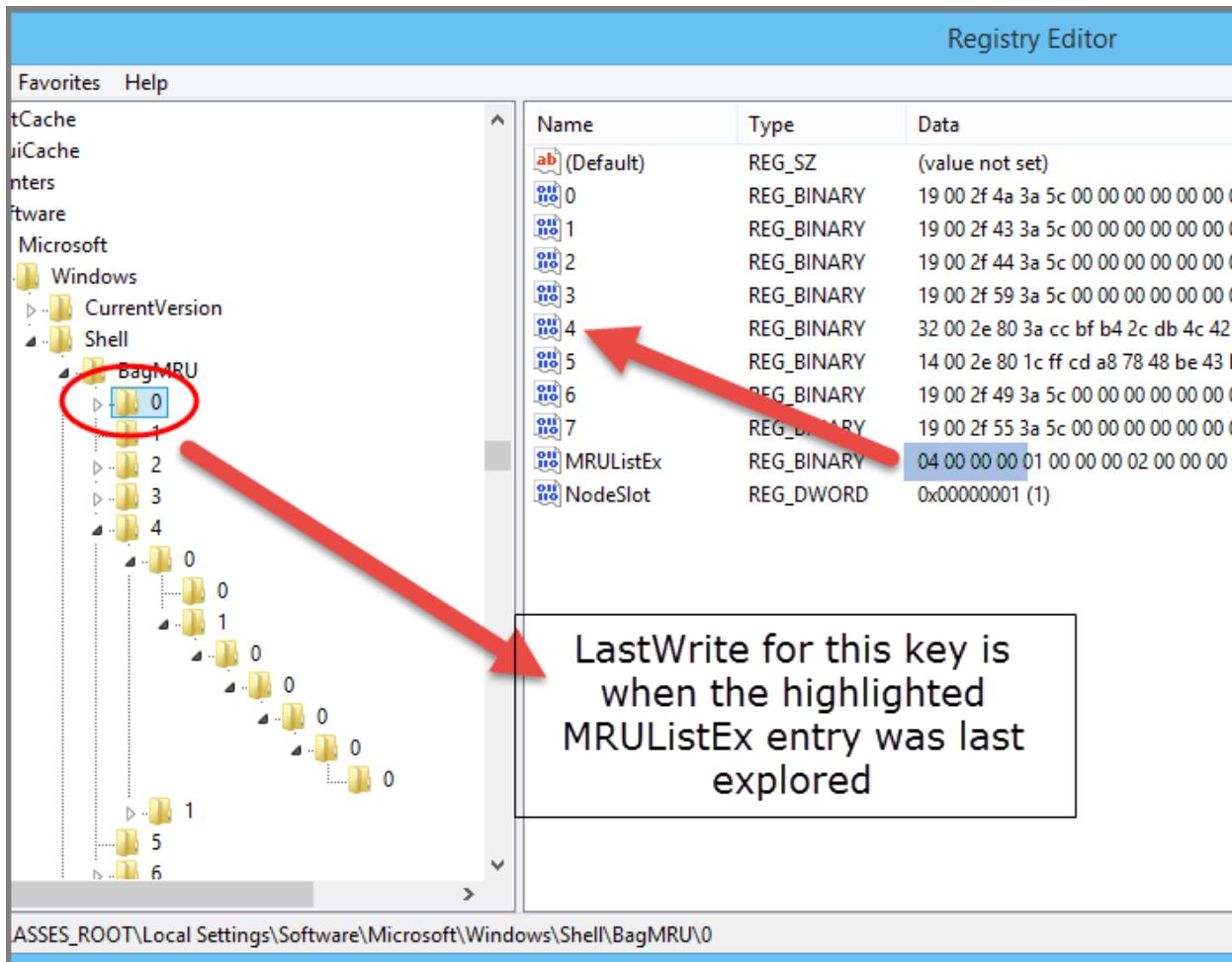
Keeping in mind how LastWrite timestamps work, we can now determine when a given ShellBag was accessed.

To determine the first time a ShellBag was explored, we can look for the “bottom” most keys and use their LastWrite timestamps for the ShellBag in the parent key. Recall in the example above, the key is “0” and the last write value is 11/17/2014 at 18:14:40.553 UTC. If we go “up” a level and select the parent key of the bottommost key, we see a ShellBag value named “0.”



Note: This only works for the bottommost key and its corresponding value in the key's parent. If a child directory is browsed underneath the highlighted key above, it is not possible to determine the first explored date for the intermediate folder anymore (but we would now know when the newly browsed folder was viewed, assuming it was the bottommost directory navigated to).

To determine the last time a ShellBag was explored, we use the LastWrite timestamp in conjunction with the MRUListEx value. Recall the first entry in MRUListEx is the most recently viewed ShellBag in a given key. By looking at the LastWrite timestamp for a key, we know when the first entry in the MRUListEx was last explored.



Since MRUListEx has its first entry set to a value of “4”, the ShellBag with the same value was last explored when the “0” key was last written to. Since the MRUListEx value is updated as ShellBags are accessed, the LastWrite timestamp is also updated.

More information on these concepts is available at <http://www.4n6k.com/2013/12/shellbags-forensics-addressing.html>.

## Why another ShellBags program?

Tools like RegRipper and sbags from TZWorks have processed ShellBags for quite some time now, but ShellBags Explorer (SBE) is different in that it presents a visual representation of what a user’s directory structure looked like. Additionally SBE exposes various timestamps such as First Explored and Last Explored for a given folder, the file system where a directory existed, and so on. SBE also contains a command line version which can produce output (while still including the additional relevant timestamps and file system info, etc.) similar to sbags and RegRipper.

## Capabilities overview

SBE is meant to be an all-inclusive tool for ShellBag artifacts. It negates the need for laborious manual steps, decoding of data, and determining contextual relationships between directories, etc.

- \* Included support for all known Extension blocks and auto-detection of unknown blocks, unknown ShellBag types, etc.
- \* Data interpreter to Hex view. As hex values are selected, the values update from the cursor's position.
- \* Support for NTUser.dat and USRClass.dat
- \* Consistent display of data for bags
- \* Ability to view all bags recursively to easily sort, filter, etc.
- \* Ability to ingest multiple registry hives and remove duplicate ShellBags. This allows for a comprehensive view of directory access spanning the range of data in all registry hives.
- \* Ability to show what directories were accessed on CD and DVD media (and therefore showing what drive letters were optical readers)
- \* And MUCH more

## How does it work?

- \* The basis for SBE is Joachim Metz's document, Windows Shell Item format specification, which is available here <http://goo.gl/rJXmLY>. The specification outlines all known shell bag types and layouts and is continually updated by Metz as new information is discovered.
- \* Certain shell bags contain other properties with complex layouts. These specifications are linked in the document above.

SBE has been designed with features and capabilities such as unknown GUID detection, extension block mismatches, etc. and has been used to contribute significant, previously unknown information to the format specification for shell items. By automatically reporting anomalies in the decoding process, these anomalies can be reported to the program developer in order to improve the capabilities of SBE and thusly the forensic process in which SBE is used.

## Why use ShellBags Explorer and not ShellBag (sbag) Parser by TZWorks, MiTeC Registry Analyser or RegRipper?

Each of the three other tools are summary tools and none show you the contents of all (or nearly all) bytes in ShellBags. In fact, some of them do not show certain bags at all. SBE parses hives that crash other tools and report far more data than others as well. Additionally, only SBE (outside of a hex editor) can truly be used to study what is inside ShellBags while at the same time representing data in such a way that is visually similar to how the end user would have seen their file systems. By representing ShellBags in a hierarchical fashion vs. a flat list, an examiner can more easily explore and see directory structure and layout as the subject did. Additionally, SBE allows for filtering, sorting and grouping far beyond the capabilities of Excel or similar tools.

## Getting started

If you are reading this, it is assumed you already have ShellBags Explorer installed somewhere on your machine. A default installation of ShellBags Explorer includes the following files:

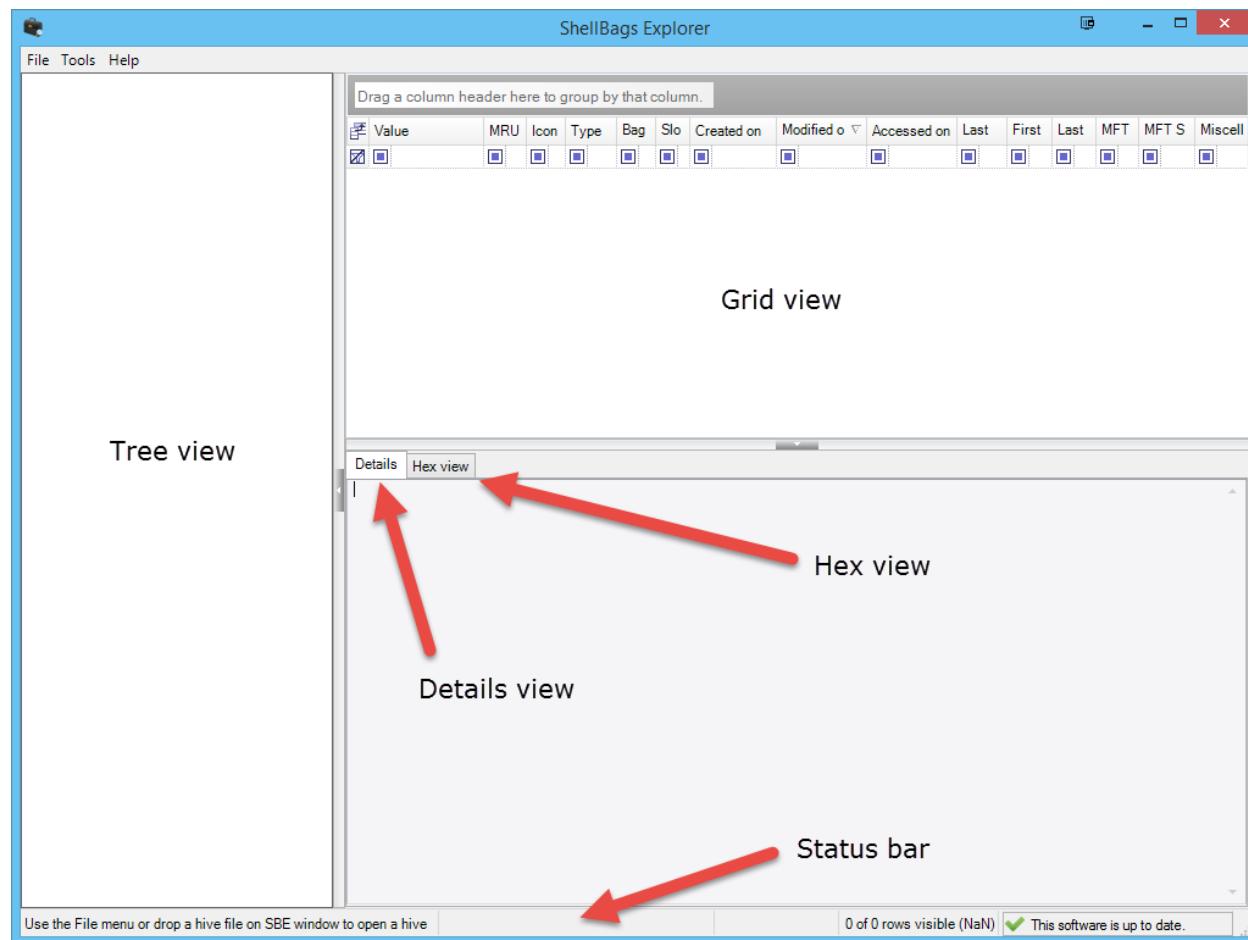
- \* SBECmd.exe: The command line version of ShellBags Explorer
- \* ShellBagsExplorer.exe: The GUI version of ShellBags Explorer

Either version of ShellBags Explorer (SBE) can be used to process and examine registry hives. One program does not rely on the other for SBE to function.

## ShellBagsExplorer.exe

This is the GUI version of SBE. It provides deep insight into shellbag data in an easy to use interface that resembles Windows Explorer.

To start SBE, open the SBE folder and double click on ShellBagsExplorer.exe.



Each of the sections above will be explored further in subsequent sections below.

## Tree view

The tree view displays a Windows Explorer like representation of ShellBag data. There are menu options under Tools to automatically expand and contract all nodes.

### Left clicking a tree node

Left clicking a node will display details of the selected node in the Details view pane. Additionally, all child bags of the selected node are displayed in the grid view.

### Right clicking a tree node

Right clicking a node in the tree will recursively load all child nodes of the selected node. The nodes are displayed in the grid view.

## Grid view

The grid view displays all ShellBags located below the selected node in the tree, or, in the case of recursive viewing, a list of all ShellBags located below the selected node and all child nodes of those nodes.

### Left clicking a grid entry

Left clicking a node in the grid view will display the details of the ShellBag in the Detail view. The hex view will also be updated to reflect the binary contents of the ShellBag.

### Right clicking a grid entry

Right clicking a node in the grid view will select that node in the tree view. This is handy when viewing nodes recursively to see where in the directory structure a particular node lives.

## Details view

After a ShellBag is selected, the details view pane displays all known data about the selected ShellBag. This includes basic information as well as extension blocks, MFT information, file system hints, raw hex content, and so on.

## Hex view

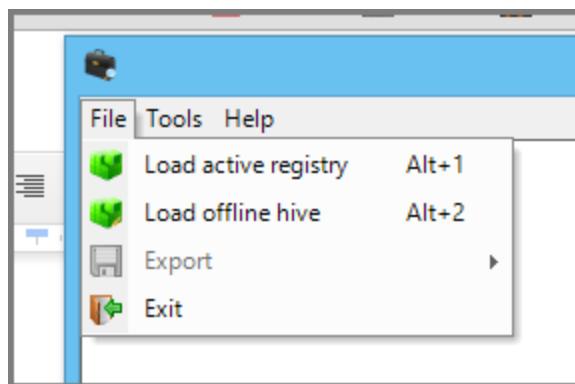
After a ShellBag is selected, the hex view will contain the raw hex content of the selected ShellBag. To the right of the hex display is a data interpreter that is updated in real time as data is selected in the hex editor. This allows for verification of data in both the grid and details view.

## Status bar

The status bar on the bottom contains various details about the loaded hive, whether or not a filter is active (including the number of visible rows vs. the total) and if there are any updates available to SBE.

## Menus

### File



The File menu allows for loading either the live registry information or an offline hive. When loading an offline hive, the user will be prompted for the location of the hive file to load.

More information is available when browsing an offline hive than browsing the live registry. This is because as of SBE version 0.5.0.0, the LastWrite timestamp for registry keys is not available when looking at the live registry. The live registry option can be used to explore ShellBags from the live machine during training, research, and so on.

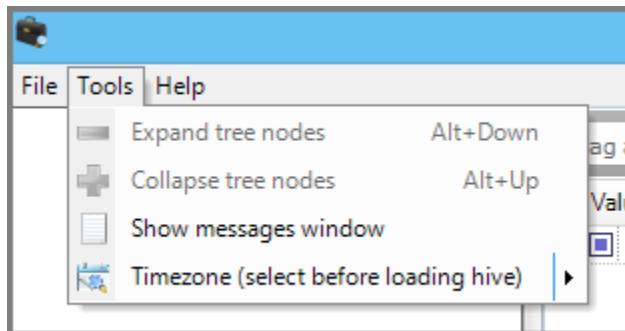
### Export submenu

Once SBE has loaded a hive, the File menu also allows for exporting of the data in one of several formats.

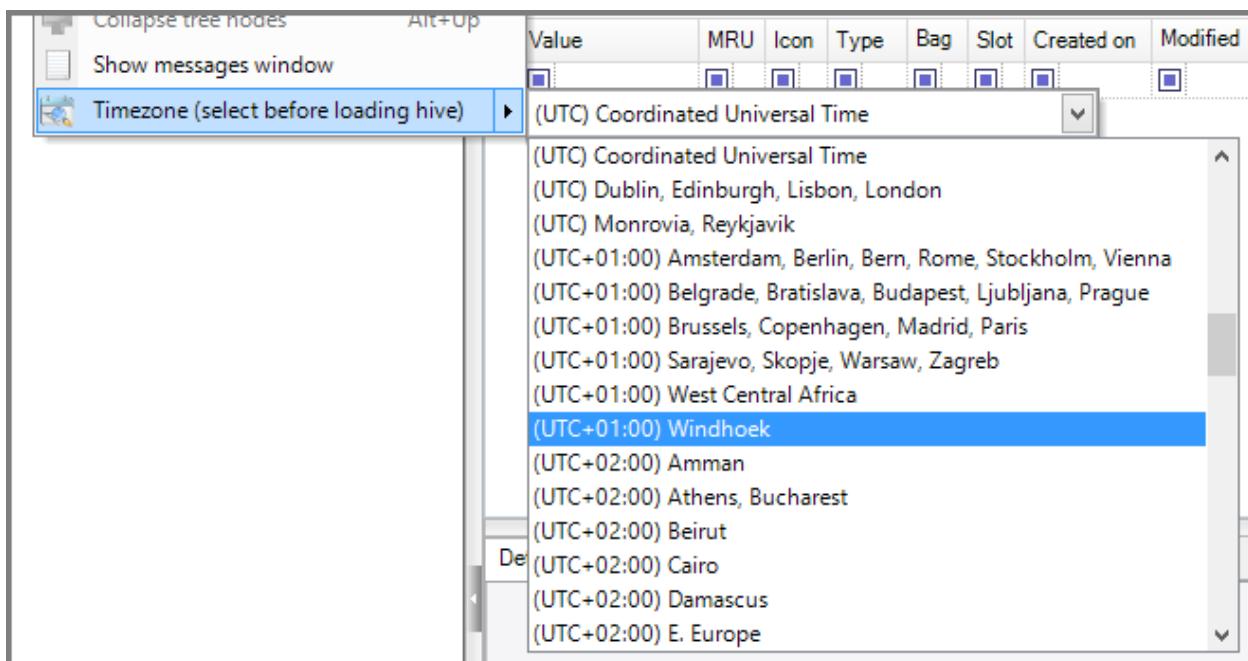
- \* CSV: All available columns will be exported
- \* Excel: The visible columns in the Grid view are exported.
- \* json: All data is exported in json format.

NOTE: You can customize the columns shown in the grid view by clicking the field chooser icon in the upper left corner of the grid (this is explained in more detail in an upcoming section). This allows you to hide or show whatever columns you choose to make your work easier.

## Tools

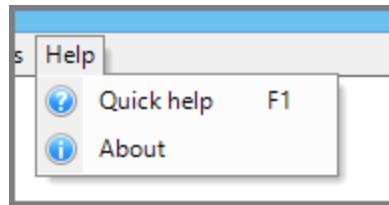


The tools menu contains options to expand and contract tree nodes, show the messages window, or set the time zone to be used when displaying dates and times. The messages window is displayed automatically as messages are generated.



A time zone must be selected before selecting a source for ShellBags (the live registry or an offline hive). By default, UTC is used for all dates and times.

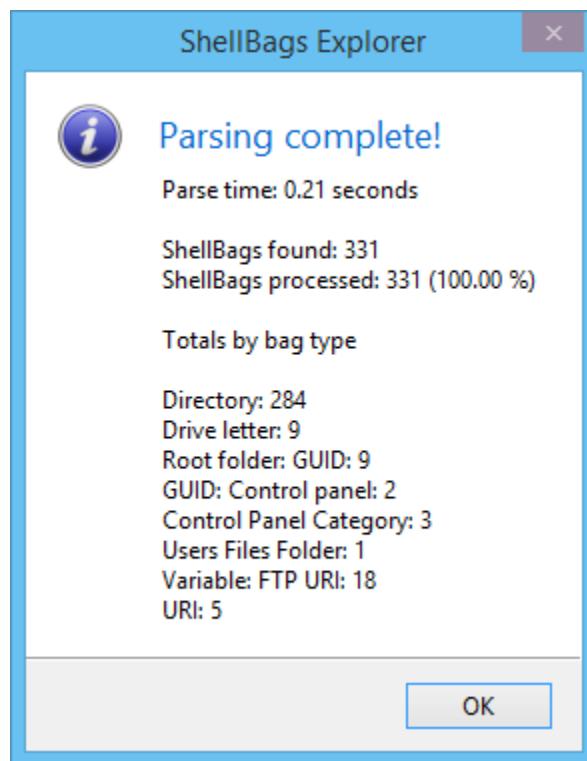
## Help



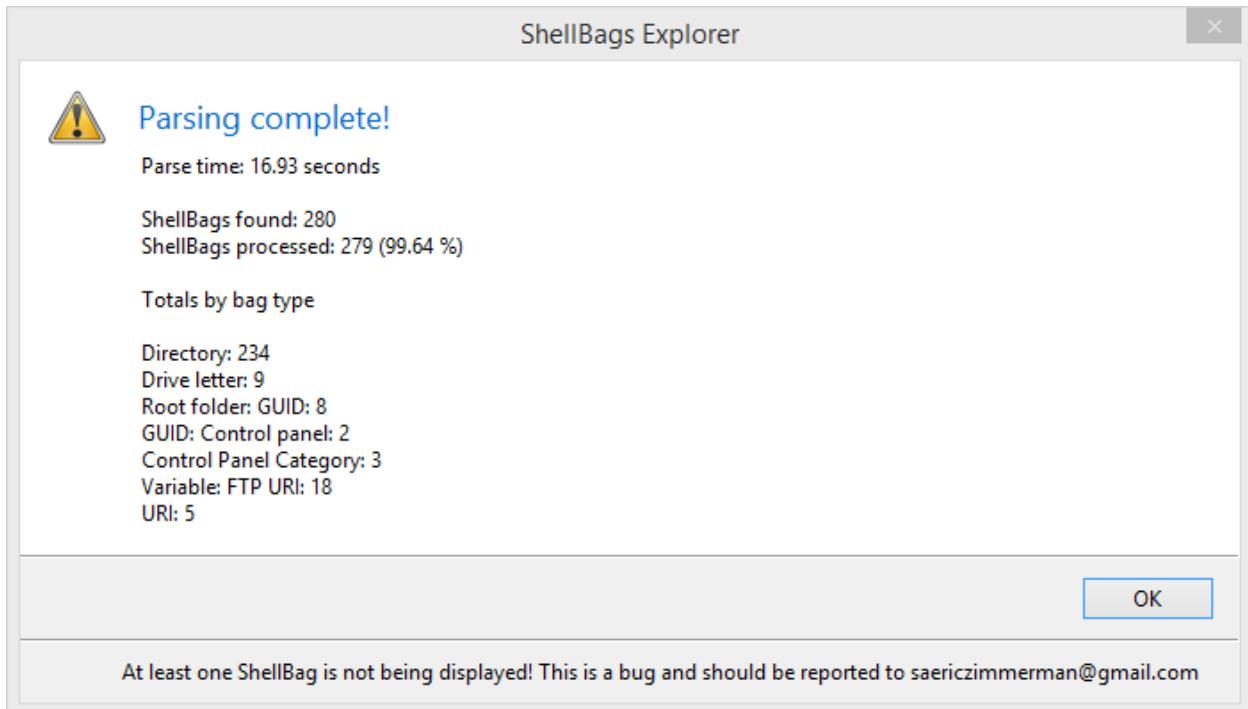
Quick help provides basic information on how to use SBE. About contains version information and contact information for the developer.

## Workflow overview

After selecting either the live registry or an offline hive, SBE will display a summary of the types of ShellBags that were found as seen below.

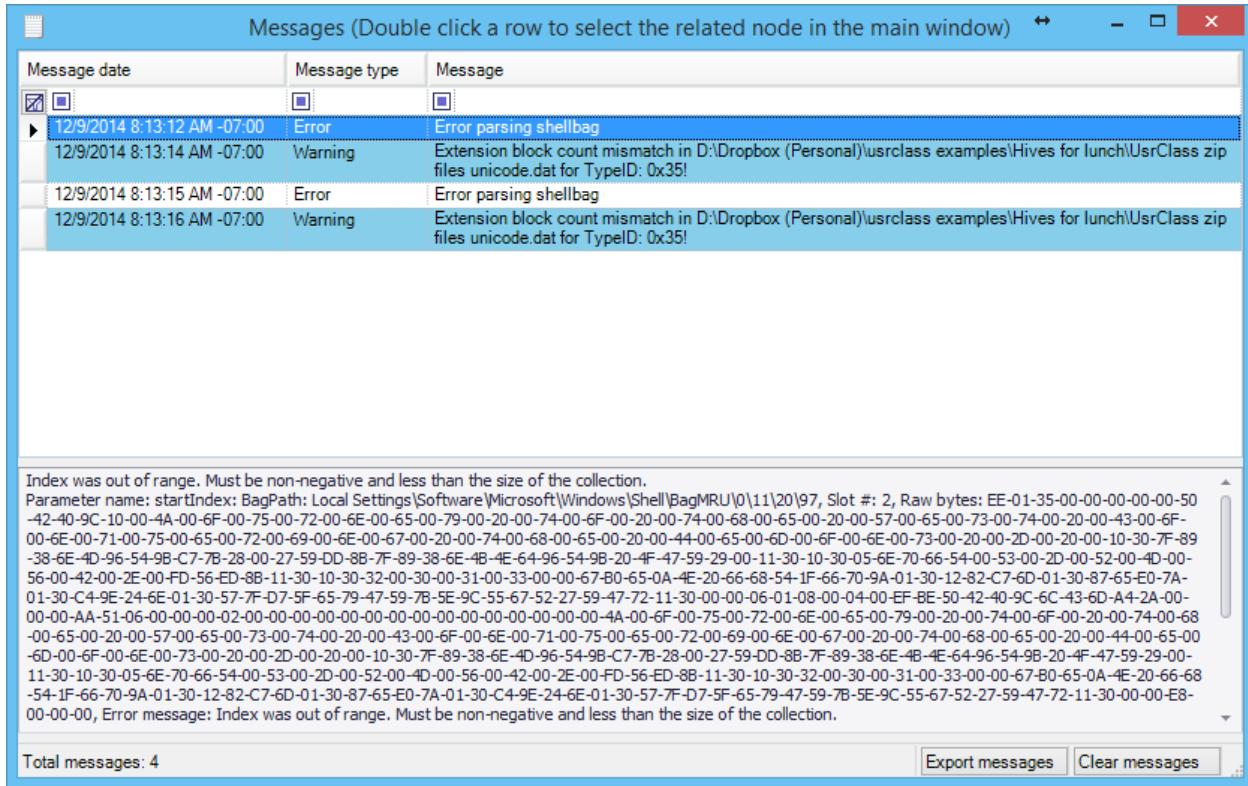


In the image above, notice 331 ShellBags were found in the registry and 331 ShellBags were processed. Should there be a situation where the number of ShellBags found is not equal to the number of ShellBags displayed by SBE, the summary screen will be different to reflect this fact.



## Messages window

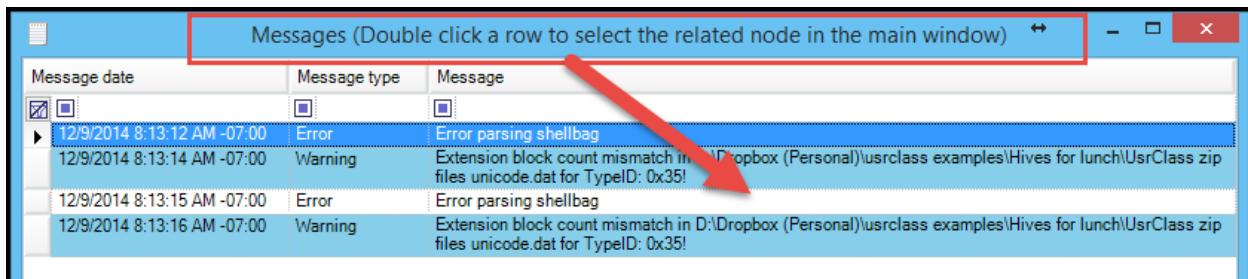
If there are any errors when processing ShellBags, the Messages window will be displayed.



Selecting a message will show the details for that message in the lower pane. All messages can be exported or cleared using the buttons in the lower right corner.

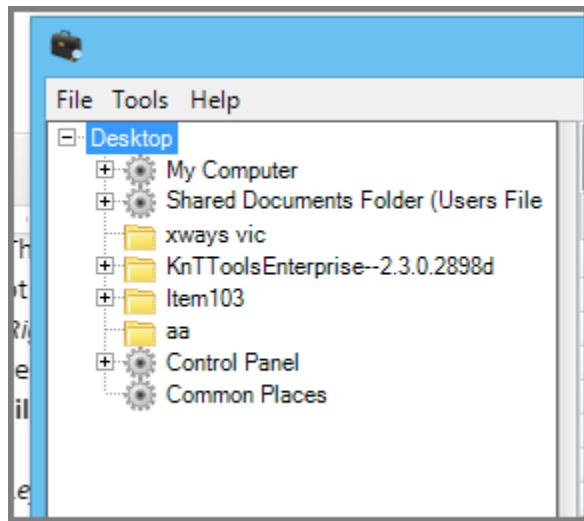
The Messages window can contain both errors and informational messages. Informational messages will be displayed when unknown GUIDs are found, new extension blocks are discovered, or there is a mismatch between the number of extension blocks in a ShellBag and the number of parsed extension blocks (these concepts will be elaborated on soon).

NOTE: You can quickly jump to the ShellBag related to the message by double clicking anywhere on a row. This will select the ShellBag in the tree so it can be reviewed as needed.

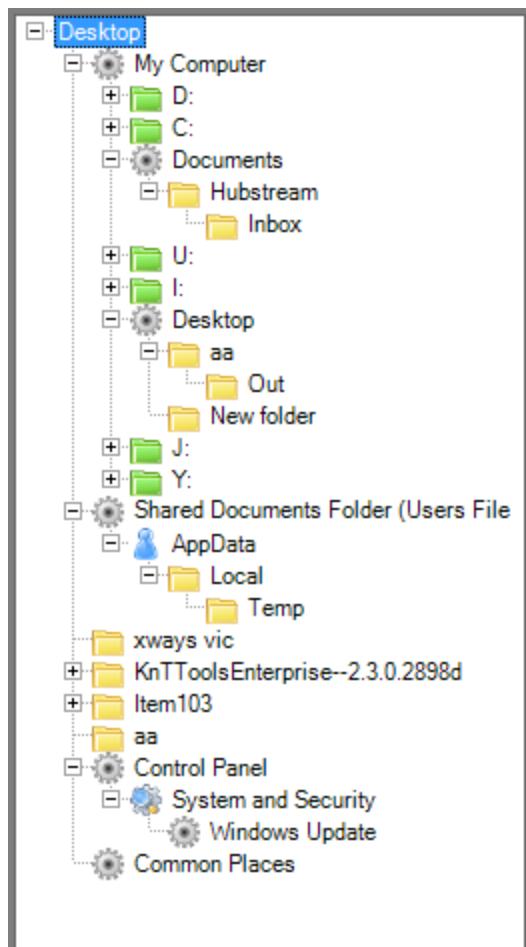


## Tree view

The tree view will also be updated. As seen earlier, the starting point for ShellBags is the Desktop folder.



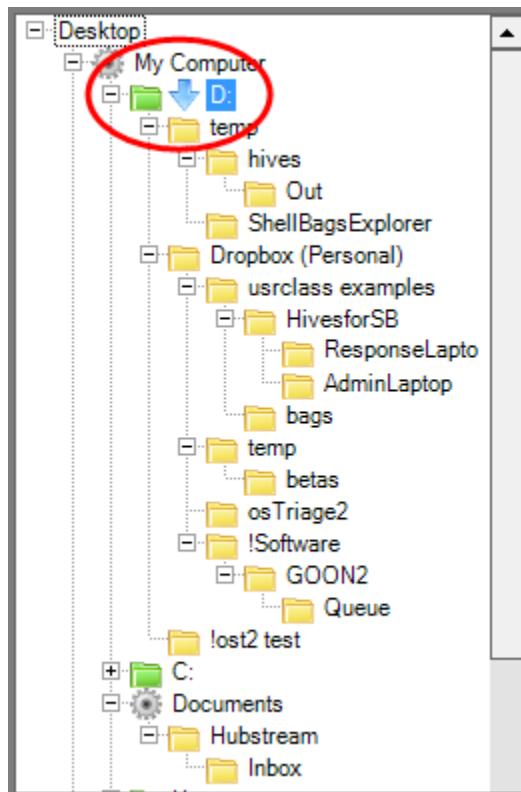
Expanding a node displays its children. In the image below, several child objects have been expanded.



Each different kind of ShellBag is represented by a different icon. In the example above we see icons

for directories, GUIDs, control panel categories, drive letters, and user application data. This helps you quickly visualize what kind of ShellBags are available.

As mentioned above, left clicking selects a given node in the tree and displays child bags in the grid view. Right clicking a node will select that node and expand all child nodes and display all bags under the selected node in the grid view.

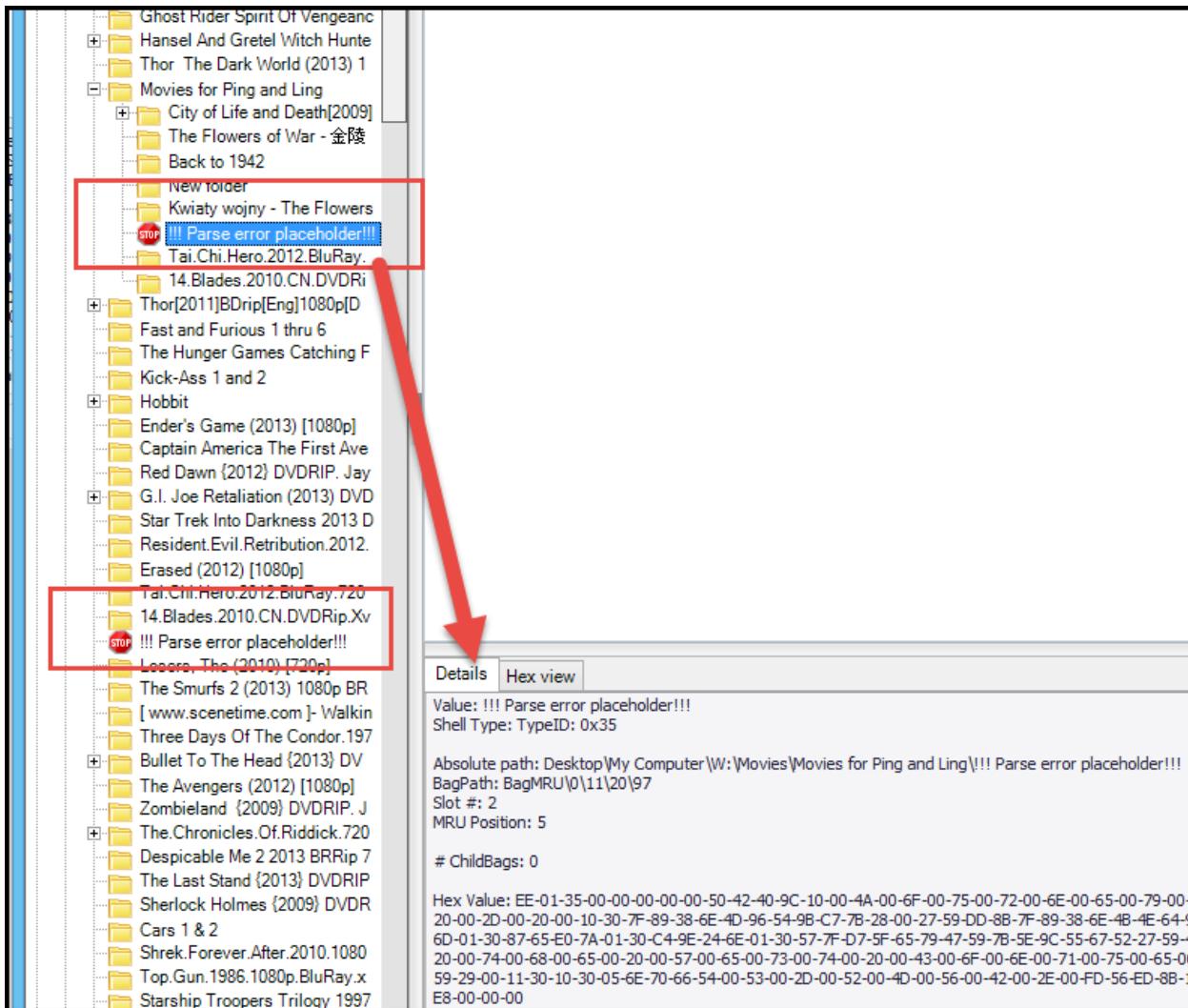


In the example above, D: has been recursively explored. Notice the icon now shows a blue down arrow indicating the node is being explored recursively. Exploring recursively lets you drill down into specific areas of interest and, as we will soon see, easily search for things across a given set of ShellBags.

## ShellBag parsing errors

Should any errors occur when parsing ShellBags, a placeholder for the ShellBag will be added as shown in the image below. Some details for the unparsed ShellBag will be displayed in the Details pane. The Hex view allows for reviewing of these ShellBags for relevant data and a message is added to the Messages window when this occurs.

By adding a placeholder for ShellBags with parse errors, any child ShellBags of the unparsed ShellBag will be still be visible in the proper context.



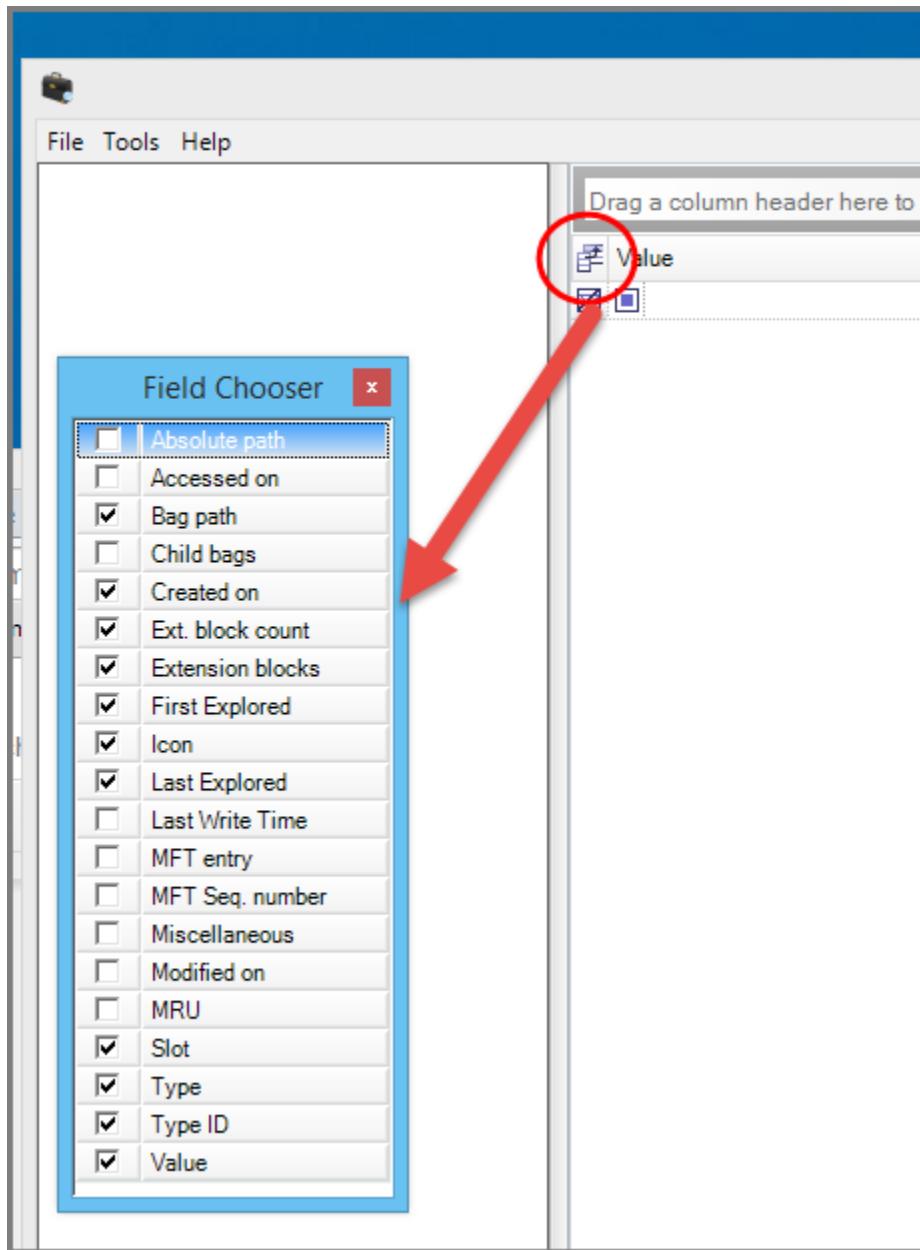
## Grid view

As nodes are selected in the tree view, the grid view is updated to show the child ShellBags of the selected node. In the case of recursive viewing, all child bags from the selected node down are displayed in the grid view.

The grid view serves as a way to view Shellbags in a consistent manner. Not all bags have data for all of the columns, but for the most part the columns available in the grid view provide a homogeneous view of ShellBags. Because of this, common features of different bag types can be grouped on, filtered, sorted by, etc.

## Customizing grid view columns

Clicking in the upper left corner of the grid view allows for customizing the fields displayed in the grid.



## Column definitions

SBE supports the following columns as of v0.5.0.0:

- \* Absolute path: The absolute path from the Desktop to the location of the shell bag
- \* Accessed on: The access date as stored in the ShellBag
- \* Bag path: The path to the ShellBag relative to the BagMRU key
- \* Child Bags: Number of child bags of the parent bag key
- \* Node Slot: The NodeSlot value for the selected ShellBag
- \* Created on: The created date as stored in the ShellBag (usually based on MFT data)

- \* Ext. block count: The total number of extension blocks found in this shell bag
- \* Extension blocks: The names of unique extension blocks found in the shell bag
- \* First Explored: When available, the timestamp a folder was first explored
- \* Icon: A visual identifier for the shell bag
- \* Last Explored: When available, the timestamp a folder was last explored
- \* Last Write Time: Only available when loading an offline hive, this is the timestamp the ShellBag KEY was last updated.
- \* MFT entry
- \* MFT Seq. number
- \* Miscellaneous: Used to report additional items of interest about shell bags as needed. This is primarily used to report the type of file system an item was located on
- \* Modified on: The modified date as stored in the ShellBag
- \* MRU: The Most Recently Used position of the bag
- \* Slot: The shell bag's numerical identifier in its parent key
- \* Type: The human readable description of what kind of data the shell bag represents (file, folder, etc.)
- \* Type ID: The hexadecimal identifier for the shell bag. This is found in offset 0x02 in the hex that makes up a shell bag.
- \* Value: The name of the file, folder, property view, etc. for the shell bag. This is the primary identifier for a shell bag.

Note: The MFT entry and sequence number can be used to determine the file system a particular bag came from:

- \* If entry number > 0 and sequence number > 0, then the file system is NTFS.
- \* If entry number > 0 and sequence number == 0, then the file system is FAT. (Further checks against the accuracy of Last Access is then done to determine FAT vs. exFAT)

When applicable, the Miscellaneous column will indicate which file system the ShellBag target originated from. Thanks to David Cowen for this idea and for testing.

## Changing column order

Column order can be adjusted by dragging and dropping columns to new positions in the grid. Any changes are persisted.

## Interacting with the grid

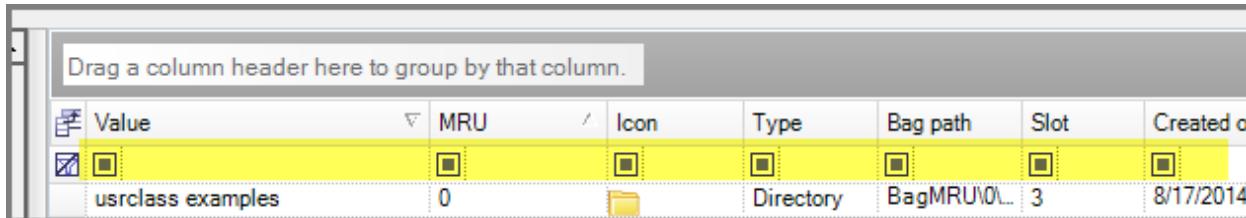
The grid view can be used to sort, filter, and group ShellBags.

## Sorting

To sort, simply click a column header. Click it again to sort in reverse order. To sort by more than one column, click the initial column to sort by, then hold SHIFT and click on another column. The second column will be sorted while the first column retains its initial sorting.

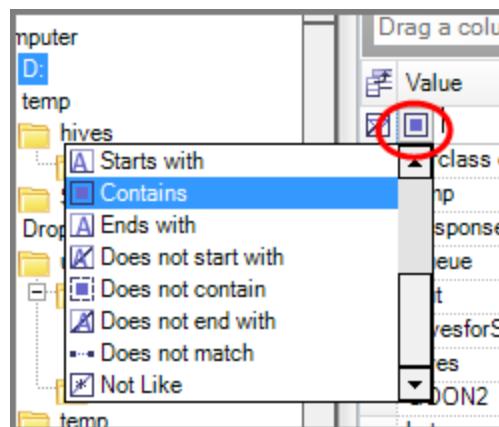
## Filtering

At the top of each column is a filter cell. In the image below it is highlighted in yellow.



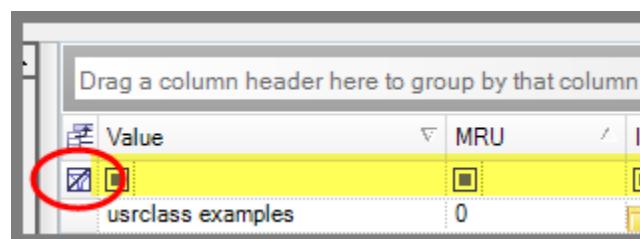
Value	MRU	Icon	Type	Bag path	Slot	Created o
usrclass examples	0	Directory	BagMRU\0L	3		8/17/2014

Clicking on this cell and typing will immediately filter that column to only items containing the entered. To the left of the cell is a button with options on how the filter should work. The default is “contains.”



There are also dropdowns which contain all unique entries in the column plus options to add custom filters to the list. To the right of each cell is a button that can be used to clear that column's filter.

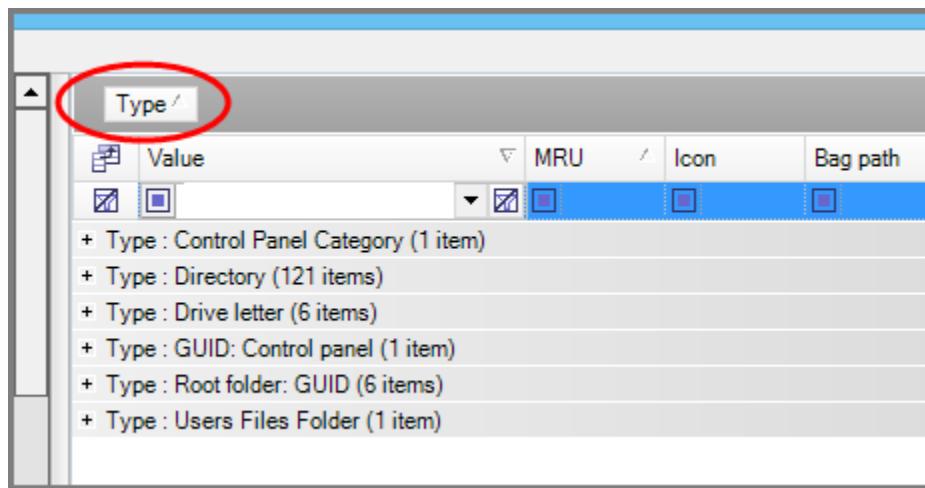
To the far right of the filter row is a button that can be used to clear all filters.



Value	MRU	Icon
usrclass examples	0	Directory

## Grouping

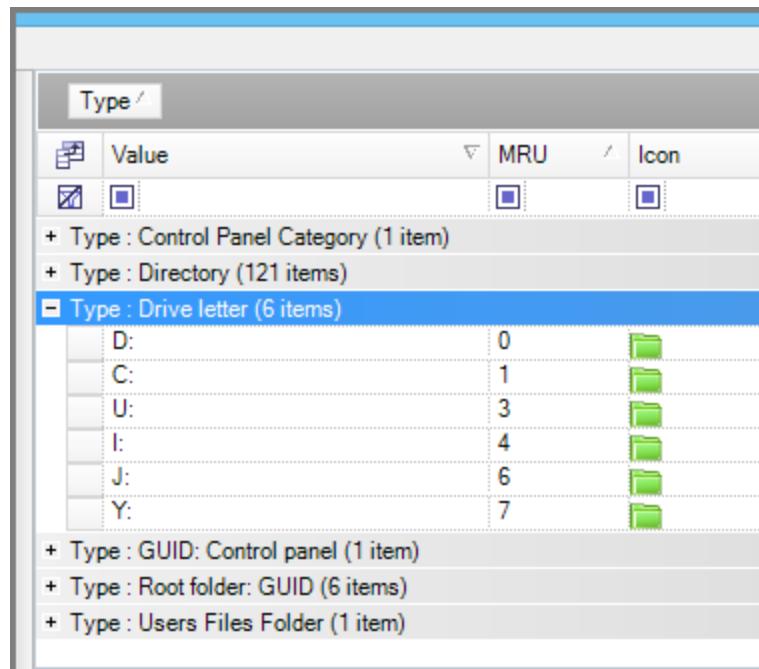
At the top of the grid is an area where one or more column headers can be dragged. As columns are dragged and dropped, each unique value in the selected column will be used to create a group.



The screenshot shows a Windows-style application window titled "Type /". Inside, there's a table with columns: Value, MRU, and Icon. A red circle highlights the "Type" column header. Below the header, there are several entries, each starting with a plus sign (+) followed by a category name and its count in parentheses. The first entry is "Type : Control Panel Category (1 item)".

Type	Value	MRU	Icon	Bag path
+	Type : Control Panel Category (1 item)			
+	Type : Directory (121 items)			
+	Type : Drive letter (6 items)			
+	Type : GUID: Control panel (1 item)			
+	Type : Root folder: GUID (6 items)			
+	Type : Users Files Folder (1 item)			

In the example above, the “Type” column was grouped. Clicking on a plus sign expands that group.



This screenshot shows the same application window after expanding the "Type : Drive letter (6 items)" group. The expanded group is highlighted with a blue background. It contains six entries, each with a drive letter (D:, C:, U:, I:, J:, Y:) and a corresponding number (0, 1, 3, 4, 6, 7) in the adjacent column. The "Type" column header is no longer circled.

Type	Value	MRU	Icon	Bag path
+	Type : Control Panel Category (1 item)			
+	Type : Directory (121 items)			
-	Type : Drive letter (6 items)			
D:	0		Folder	
C:	1		Folder	
U:	3		Folder	
I:	4		Folder	
J:	6		Folder	
Y:	7		Folder	
+	Type : GUID: Control panel (1 item)			
+	Type : Root folder: GUID (6 items)			
+	Type : Users Files Folder (1 item)			

Clicking the minus sign collapses it.

After grouping, columns can be sorted as we saw earlier, by clicking on the column header.

To group more than one column, drag them to the group area.

Type	Value	Icon	Bag
MRU			
+ Type : Control Panel Category (1 item)			
- Type : Directory (8 items)			
+ MRU : 0	(68 items)		
+ MRU : 1	(25 items)		
+ MRU : 2	(11 items)		
+ MRU : 3	(8 items)		
- MRU : 4	(4 items)		
Program Files (x86)		Folder	Bag
Item103		Folder	Bag
GOON		Folder	Bag
1		Folder	Bag
+ MRU : 5 (3 items)			
+ MRU : 6	(1 item)		
+ MRU : 7	(1 item)		
+ Type : Drive letter (6 items)			
+ Type : GUID: Control panel (1 item)			
+ Type : Root folder: GUID (6 items)			
+ Type : Users Files Folder (1 item)			

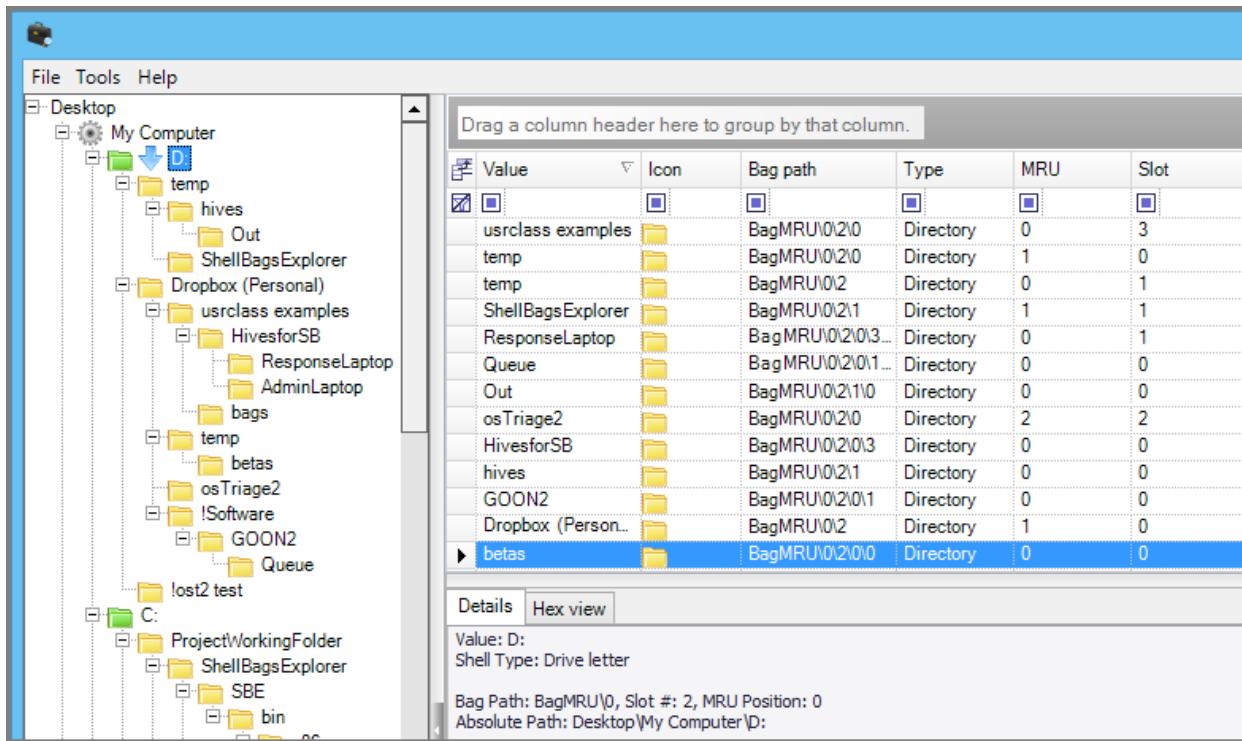
In the example above, we first grouped by type, then by MRU. This can be useful to see a list of all the most recently accessed directories (i.e. all the ShellBags of type directory where MRU == 0).

To “undo” grouping, drag the columns back into the main grid area at any location.

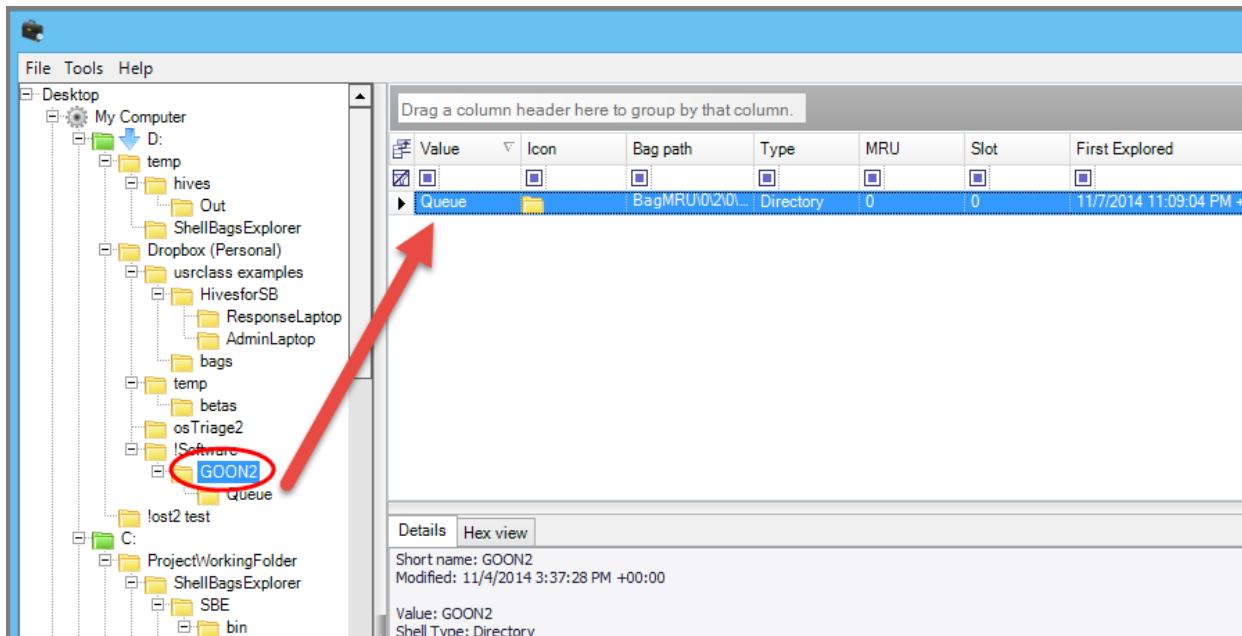
## Selecting bags in grid view

Left clicking a bag will select that bag and display that bag’s details in the details view (described below). After selecting a bag via left clicking it, you can jump to that bag by right clicking it.

In the image below, we are recursively viewing the bags at and below D:



Notice the D: drive is highlighted in the tree view and the details view reflects this. If the bag with value “GOON2” is left clicked, then right clicked, notice the tree view is updated and the “GOON2” bag is highlighted and the grid view is updated to show the child bags of the GOON2 bag.



The details pane is also updated with the newly selected bag in the tree view. Clicking the “Queue” bag in grid view will update the details view to the “Queue” bag.

The screenshot shows the Shellbags Explorer interface. At the top, there is a header bar with the title 'Shellbags Explorer'. Below it is a toolbar with various icons. The main area consists of two panes. The left pane is a grid view with columns labeled 'Value', 'Icon', 'Bag path', 'Type', 'MRU', and 'Slot'. A row in this grid is highlighted with a red circle around the 'Value' column, which contains the text 'Queue'. A large red arrow points from this highlighted row down to the right pane. The right pane is titled 'Details' and displays the following information:

Short name:	Queue
Modified:	10/17/2014 4:28:50 PM +00:00
Value:	Queue
Shell Type:	Directory
Bag Path:	BagMRU\0\2\0\1\0, Slot #: 0, MRU Position: 0
Absolute Path:	Desktop\My Computer\D:\Dropbox (Personal)\Software\GOON2\Queue
# Child Bags:	0
First explored:	11/7/2014 11:09:04 PM +00:00
Last explored:	11/7/2014 11:09:04 PM +00:00

## Details view

The details view contains all known ShellBag information in human readable (and hopefully consumable) format. Binary data is converted to timestamps, strings, numbers, and other structures.

ShellBags in general have the following structure (For an exhaustive breakdown of ShellBag binary layouts, including extension blocks, see the Metz document referenced above (<http://goo.gl/rJXmLY>)):

- \* Offset 0-1: Size of the ShellBag
- \* Offset 2: Type indicator
- \* Type specific data

The type specific data contains things as timestamps, file attributes, strings, and extension blocks. Extension blocks contain additional data relevant to a ShellBag which can be used in different ShellBags. Because of this, extension blocks have their own formatting that is consistent across bags.

Extension blocks can be identified by a unique signature, BE EF 00 XX, where XX varies on the type of extension block. In the image below, the beginning of a BEEF0004 block is highlighted. Since the data is stored in little endian format it has to be read “backwards.”

00	01	02	03	04	05	06	07	08	09	0A	0B	
60	00	31	00	00	00	00	43	45	F1	72		.. 1..... CEñr
10	00	41	44	4D	49	4E	4C	7E	31	00	00	.. ADM NL~1..
48	00	09	00	04	00	EF	BE	43	45	EC	72	H.....T%CEñ r
45	45	74	A0	2E	00	00	00	2C	00	00	00	EEt .....,..
00	00	3B	00	00	00	00	00	00	00	00	00	...;.....
00	00	00	00	00	00	88	A2	18	00	41	00	.....ç.. A.
64	00	6D	00	69	00	6E	00	4C	00	61	00	d. m i . n . L . a.
70	00	74	00	6F	00	70	00	00	00	18	00	p. t . o . p . .....
00	00											..

Extension blocks in general have the following format:

- \* Offset 0-1: Size of extension block
- \* Offset 2-3: Extension version
- \* Offset 4-7: Extension signature
- \* Offset 8+: Extension specific data

In the image above, we can see:

- \* Size = 0x48, or 72 decimal
- \* Version = 0x9
- \* Signature = 0x0400EFBE, or when converted BEEF0004 (start from the right and read left)
- \* After the signature the extension block specific data begins. BEEF0004 denotes a file entry extension block and as such we can recover creation and last access timestamps (FAT format), the version of Windows (Vista, 7, 8, etc.), long and short directory names, and MFT related information.

As SBE processes ShellBags, all of this binary data is interpreted and stored in different objects that make up the ShellBag, so rather than seeing a string of hexadecimal characters, SBE represents bags using common structures like Value, type, and so on (the fields, in general, as reflected in the grid view).

All available information about a bag is visible in details view, even when a bag contains data that cannot be abstracted into a grid view. For example, in some cases a bag may have 4 BEEF0004 blocks (and therefore 4 sets of timestamps, MFT info, etc.) and the details view allows you see all that data.

An example of the full details of a ShellBag representing a directory is shown below.

**Details** **Hex view**

Short name: ADMINL~1  
Modified: 10/3/2014 2:23:34 PM +00:00

Value: AdminLaptop  
Shell Type: Directory

Bag Path: BagMRU\0\2\0\3\0, Slot #: 0, MRU Position: 1  
Absolute Path: Desktop\My Computer\D:\Dropbox (Personal)\usrclass examples\HivesforSB\AdminLaptop

# Child Bags: 0

First explored: 11/17/2014 9:25:46 PM +00:00

Extension blocks found: 1  
----- Block 0 -----  
Signature: Oxbeef0004  
Size: 72  
Version: 9  
Version Offset: 0x18

Identifier: 2E (Windows 8.1)

Created On: 10/3/2014 2:23:24 PM +00:00  
Last Access: 10/5/2014 8:03:40 PM +00:00

Long Name: AdminLaptop

MFT Entry Number: 44  
MFT Sequence Number: 59

File system hint: NTFS

---

Last Write Time: 11/17/2014 9:26:46 PM +00:00

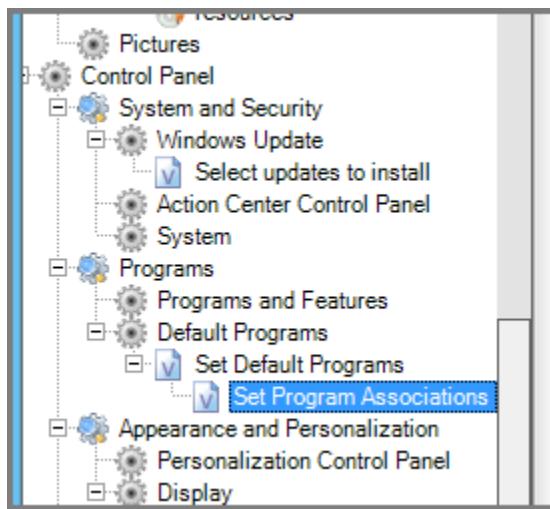
Hex Value: 60-00-31-00-00-00-00-43-45-F1-72-10-00-41-44-4D-49-4E-4C-7E-31-00-00-48-00-09-00-04-00-EF-BE-43-45-EC-72-45-45-74-A0-2E-00-00-02-C0-00-00-00-00-3B-00-00-00-00-00-00-00-00-00-00-00-88-A2-18-00-41-00-64-00-6D-00-69-00-6E-00-4C-00-61-00-70-00-74-00-6F-00-70-00-00-00-18-00-00-00

Here we can see the short name, MAC dates and times, full bag path, the absolute path, and the first explored date.

Note: The Last Write Time of the PARENT key (BagMRU\0\2\0\3\0) this ShellBag lives in is 11/17/2014 9:26:46 PM +00:00, but the First explored timestamp is 11/17/2014 9:25:46 PM +00:00. This is because First explored is determined by the last write date of the “bottom” key corresponding to BagMRU\0\2\0\3\0, Slot 0. The key for this bottom slot is not shown but the Last Write timestamp is collected and applied to the relevant bag.

Additionally we can see that the file system is NTFS. This can be determined since we have both an MFT entry and sequence number.

Some ShellBags contain embedded property sheets. These are another data structure commonly seen in various Windows artifacts and essentially are key/value pairs. In the example below, the bags related to the Control Panel have been expanded.



After selecting the “Set Program Associations” bag, its details view contains:

Here we can see the various property sheets and the data contained therein. Notice that now it is possible to tell not only WHEN program associations were changed, but WHAT program was used to take over those associations.

Finally, the bottom of the details view contains the raw hexadecimal data.

## Hex view

The hex view contains the raw hexadecimal data of the selected Shellbag. It is displayed in traditional hex editor style in read-only mode.

	Details	Hex view	
		00 01 02 03 04 05 06 07 08 09 0A 0B 60 00 31 00 00 00 00 00 43 45 F1 72 10 00 41 44 4D 49 4E 4C 7E 31 00 00 48 00 09 00 04 00 EF BE 43 45 EC 72 45 45 74 A0 2E 00 00 00 2C 00 00 00 00 00 3B 00 00 00 00 00 00 00 00 00 00 00 00 88 A2 18 00 41 00 64 00 6D 00 69 00 6E 00 4C 00 61 00 70 00 74 00 6F 00 70 00 00 00 18 00 00 00 00	. 1..... CEñr . ADM NL~1.. H. .... i %CEi r EEt ..... ...; ..... ..... c.. A. d. m i . n. L. a. p. t. o. p. .... ..
00000000			Signed 16-bit: 96 Unsigned 16-bit: 96 Signed 32-bit: 3,211,360 Unsigned 32-bit: 3,211,360
0000000C			FILETIME: 1/1/1601 12:00:00 AM +00:00
00000018			DOS Date: n/a
00000024			GUID: 00310060-0000-0000-4345-f1721000 4144
00000030			
0000003C			
00000048			
00000054			
00000060			
			Bytes selected: 0 (Hex: 0x0) Offset: 0 (Hex: 0x0)

To the right of the hex data is a rough interpretation of the data. A few strings can be seen in both ASCII and Unicode format. As the position of the cursor is changed by clicking on hex values, the data interpreter updates in real time.

## Data interpreter

The data interpreter assists in decoding the contents of ShellBags when manual verification is desired. As the cursor is moved to different offsets, the interpreter updates in real time and shows different interpretations of the data FROM THE POSITION OF THE CURSOR IRRESPECTIVE OF ANY SELECTED BYTES.

Details	Hex view	
	00 01 02 03 04 05 06 07 08 09 0A 0B	
00000000	5A 00 31 00 00 00 00 00 98 44 24 69	Z. 1..... D\$!
0000000C	10 00 4E 45 57 46 4F 4C 7E 31 00 00	.. NEWFOL~1..
00000018	42 00 08 00 04 00 EF BE 98 44 24 69	B..... i ¾ D\$!
00000024	98 44 24 69 2A 00 00 00 36 92 02 00	. D\$! *.... 6...
00000030	00 00 07 00 00 00 00 00 00 00 00 00	.....
0000003C	00 00 00 00 00 00 4E 00 65 00 77 00	..... N. e. w.
00000048	20 00 66 00 6F 00 6C 00 64 00 65 00	. f. o. l. d. e.
00000054	72 00 00 00 18 00 00 00	r.....
		Signed 16-bit: 17,560 Unsigned 16-bit: 17,560 Signed 32-bit: 1,763,984,536 Unsigned 32-bit: 1,763,984,536 FILETIME: n/a DOS Date: <b>4/24/2014 1:09:08 PM +00:00</b> GUID: 69244498-0010-454e-5746-4f4c7e31000  Bytes selected: 0 Offset: 8 (Hex: 0x8) (Double click here to copy values)

In this example the cursor is at offset 0x8 and we see the DOS date is showing 4/24/2014 1:09:08 PM +00:00 which corresponds to the Modified timestamp.

In the next example we can see the data interpreter has found a GUID for the Control Panel at offset 0x4.

Details	Hex view	
	00 01 02 03 04 05 06 07 08 09 0A 0B	
00000000	14 00 1F 70 08 06 EE 26 0A A0 D7 44	... ph. i & xD
0000000C	93 71 BE B0 64 C9 86 83 00 00	. q¾ dÉ....
		Signed 16-bit: 1,640 Unsigned 16-bit: 1,640 Signed 32-bit: 653,133,416 Unsigned 32-bit: 653,133,416 FILETIME: n/a DOS Date: 3/8/1983 4:55:28 AM +00:00 GUID: <b>26ee0668-a00a-44d7-9371-beb064c98683 (Maps to: Control Panel)</b>  Bytes selected: 0 Offset: 4 (Hex: 0x4) (Double click here to copy values)

By using the data interpreter to explore ShellBags you can begin to see common patterns including timestamps, extension blocks, property sheets, and so on.

NOTE: The data interpreter will show certain fields in bold based on certain conditions:

-A GUID is found that maps to a known location

-A timestamp where the calculated datetime > Now - 10 years AND calculated datetime < Now + 5 years.

These provide visual cues that a valid timestamp or GUID may have been found.

Finally, the data from the data interpreter and its corresponding ShellBag details can be copied to the clipboard by double clicking the designated area at the bottom of the data interpreter. An example is shown below of what this data may look like once copied into a text editor.

```
0.....10.....20.....30.....40.....50.....60.....70.....  
0 Offset: Offset: 4 (Hex: 0x4)  
1  
2  
3 Signed 16-bit: 1,640  
4 Unsigned 16-bit: 1,640  
5 Signed 32-bit: 653,133,416  
6 Unsigned 32-bit: 653,133,416  
7 FILETIME: n/a  
8 DOS Date: 3/8/1983 4:55:28 AM +00:00  
9 GUID: 26ee0668-a00a-44d7-9371-beb064c98683 (Maps to: Control Panel)  
10  
11  
12 Value: Control Panel  
13 Shell Type: Root folder: GUID  
14  
15 Bag Path: BagMRU, Slot #: 2, MRU Position: 11  
16 Absolute Path: Desktop\Control Panel  
17  
18 # Child Bags: 3  
19  
20 Hex Value: 14-00-1F-70-68-06-EE-26-0A-A0-D7-44-93-71-BE-B0-64-C9-86-83-00-00  
21  
22  
23 |
```

## SBECmd.exe

SBECmd.exe is a command line version of ShellBags Explorer that will automatically process hive files and export the results to TSV. The command line options are shown below.

```
SBECmd 0.5.0.0
Copyright (C) 2014 Eric R. Zimmerman <saericzimmerman@gmail.com>

Usage: SBECmd -d <directory>

-d          Required. Directory file to look for registry hives
--dedupe    <Default: False> When true, SBECmd processes all hives and
            removes duplicate shellbag items. Deduplication is based on
            BagPath, Slot, MRUPosition, Value, CreatedOn, ModifiedOn,
            AccessedOn, MFTEntry, MFTSequenceNumber, FirstExplored, and
            LastExplored
--timezone  <Default: GMT Standard Time> The timezone to use when
            displaying dates and times <Default = GMT Standard Time>.
            Enclose in quotes. Use --timezone="" to see a list of all
            available timezones
--help      Display this help screen.

Using -d, SBECmd will process each file in <directory> and create one TSV
report per file found in <directory>\Out.

Using -d and --dedupe, SBECmd will process each file in <directory> and create
one report containing the information from ALL hives with duplicated shellbags
removed.
```

## Processing multiple, unrelated hives

After exporting one or more hives to a directory, execute SBECmd.exe as follows:

```
SBECmd.exe -d <directory>
```

A new directory underneath <directory> will be created called ‘Out’ which will contain the TSV reports generated by SBECmd.exe. An example run is shown below.

```
C:\temp>SBECmd.exe -d c:\Users\eric\Desktop\aa
SBECmd.exe version 0.5.0.0
Directory to process: c:\Users\eric\Desktop\aa
Export format: TSV
Deduplication: False
Timezone: GMT Standard Time

Processing 'c:\Users\eric\Desktop\aa\UsrClass 1.dat'
Parse time: 0.33 seconds

    Total Shell Bags found: 65

    Totals by bag type

        Directory: 56
        Drive letter: 4
        Root folder: GUID: 3
        GUID: Control panel: 1
        Control Panel Category: 1

Parse time: 0.33 seconds

    Total Shell Bags found: 65

    Totals by bag type

        Directory: 56
        Drive letter: 4
        Root folder: GUID: 3
        GUID: Control panel: 1
        Control Panel Category: 1

Finished processing 'c:\Users\eric\Desktop\aa\UsrClass 1.dat'
Exported to: 'c:\Users\eric\Desktop\aa\Out\UsrClass 1.dat.tsv'

-----
Processing 'c:\Users\eric\Desktop\aa\UsrClass 2.dat'
Parse time: 0.60 seconds

    Total Shell Bags found: 136

    Totals by bag type

        Directory: 121
        Drive letter: 6
        Root folder: GUID: 6
        GUID: Control panel: 1
        Control Panel Category: 1
        Users Files Folder: 1

Parse time: 0.60 seconds

    Total Shell Bags found: 136

    Totals by bag type

        Directory: 121
        Drive letter: 6
        Root folder: GUID: 6
        GUID: Control panel: 1
        Control Panel Category: 1
        Users Files Folder: 1

Finished processing 'c:\Users\eric\Desktop\aa\UsrClass 2.dat'
Exported to: 'c:\Users\eric\Desktop\aa\Out\UsrClass 2.dat.tsv'

-----
Processing complete!
Processed 2 files in 0.93 seconds!
Total Shell Bags found: 201
```

If SBECmd.exe encounters any errors processing hives, they will be reported to the console and to '!SBECmd\_Messages.txt' in <directory>\Out

## Processing multiple, related hives

In certain cases you will have multiple registry hives from the same user. These hives may be found in volume shadow copies, carving, etc.

SBECmd can also deduplicate ShellBags across multiple registry hives via the –dedupe switch. This is useful when processing many hives from a given user and eliminating duplicate shellbag entries. Uniqueness is determined from the following data in a ShellBag: BagPath, Slot, MRUPosition, Value, CreatedOn, ModifiedOn, AccessedOn, MFTEntry, MFTSequenceNumber, FirstExplored, and LastExplored. These values are combined and then SHA-1 hashed. Only one copy of a ShellBag with a given SHA-1 hash is reported and the rest are discarded.

After exporting one or more hives to a directory, execute SBECmd.exe as follows:

```
SBECmd.exe -d <directory> --dedupe
```

A new directory underneath <directory> will be created called 'Out' which will contain the TSV report named 'Deduplicated.tsv'. An example run is shown below.

```
C:\temp>SBECmd.exe -d c:\Users\eric\Desktop\aa --dedupe
SBECmd.exe version 0.5.0.0
Directory to process: c:\Users\eric\Desktop\aa
Export format: TSV
Deduplication: True
Timezone: GMT Standard Time

Processing 'c:\Users\eric\Desktop\aa\UsrClass 1.dat'
Processing 'c:\Users\eric\Desktop\aa\UsrClass 2.dat'
Unique Shell Bags found: 162 (19.40 % duplicates)
Results saved to 'c:\Users\eric\Desktop\aa\Out\Deduplicated.tsv'

Processing complete!
Processed 2 files in 0.93 seconds!
Total Shell Bags found: 201
```

## Time zone support

Like its GUI counterpart, SBECmd can adjust all dates and times to a user selected time zone. The default time zone is UTC, but a different time zone can be selected via the –timezone switch. Enclose the timezone in quotes. An example is shown below.

```
C:\temp>SBEcmd.exe -d c:\Users\eric\Desktop\aa --dedupe --timezone="Mountain standard time"
SBEcmd.exe version 0.5.0.0
Directory to process: c:\Users\eric\Desktop\aa
Export format: TSU
Deduplication: True
Timezone: Mountain standard time
Processing 'c:\Users\eric\Desktop\aa\UsrClass_1.dat'
-----
Processing 'c:\Users\eric\Desktop\aa\UsrClass_2.dat'
Unique Shell Bags found: 162 (19.40 % duplicates)
Results saved to 'c:\Users\eric\Desktop\aa\Out\Reduplicated.tsv'
Processing complete!
Processed 2 files in 0.93 seconds!
Total Shell Bags found: 201
```

The TSV file(s) will now reflect the selected time zone, including any adjustments for daylight savings time.

Count	FirstExplored	LastExplored	Misc
0	11/7/2014 10:29:55 AM -07:00		
0		11/13/2014 8:42:44 AM -07:00	
0		11/8/2014 9:54:49 AM -07:00	
0			
1	11/7/2014 10:03:32 AM -07:00		NTFS
1			NTFS
1	11/7/2014 3:25:07 PM -07:00		exFAT
1	11/7/2014 3:25:08 PM -07:00	11/7/2014 3:25:08 PM -07:00	FAT fi
1	11/7/2014 10:02:59 AM -07:00	11/7/2014 10:02:59 AM -07:00	NTFS
1	11/7/2014 3:22:48 PM -07:00		exFAT
1	11/7/2014 3:23:51 PM -07:00	11/7/2014 3:23:51 PM -07:00	exFAT
1		11/8/2014 9:28:00 AM -07:00	NTFS
1			NTFS
1			NTFS
1		11/7/2014 12:30:46 PM -07:00	NTFS
1			NTFS
1			NTFS
1	11/7/2014 12:30:33 PM -07:00		NTFS
1	11/7/2014 12:30:38 PM -07:00		NTFS
1	11/7/2014 12:30:43 PM -07:00		NTFS
1			NTFS
1			NTFS
1		11/8/2014 3:19:56 PM -07:00	NTFS
1	11/7/2014 10:33:31 AM -07:00	11/7/2014 3:17:04 PM -07:00	NTFS
1	11/7/2014 11:03:09 AM -07:00		NTFS
1	11/7/2014 11:03:12 AM -07:00		NTFS
1	11/7/2014 12:29:56 PM -07:00	11/7/2014 12:29:56 PM -07:00	

## General usage tips and tricks

(misc stuff here) cdburn etc

## Version changes

### Version 0.6.0.0

NEW: Added NodeSlot info. This column is hidden by default in the grid, but will always be shown in Details view

CHANGE: Updated registry parser code (~150% faster and significant memory reduction)

FIX: Correct FAT vs exFAT file system hint in BEEF0004 blocks

FIX: Correctly detect end of ShellBag in C3 bags

### Version 0.5.0.4

NEW: Build for AnyCPU vs forcing x86.

NEW: Add icon for “History folder” ShellBag type

NEW: Allow hives to be dropped on Hex view

NEW: Added millisecond precision to timestamps that have that level of resolution. These timestamps are visible in the details pane

NEW: Added support for http URLs in variable blocks.

NEW: Added ability to double click on Data Interpreter to copy values on Interpreter plus ShellBag details to Clipboard

NEW: FILETIME and DOS Date fields in data interpreter will be bolded if the calculated date > Now - 10 years AND calculated date < Now + 5 years. This provides a visual cue a valid timestamp may have been found.

NEW: GUID field in the data interpreter will be bolded if the calculated GUID maps to a known value

NEW: Count all ShellBag registry values seen and compare to the number of ShellBags processed. If seen != processed, show warning on summary screen as data is missing

NEW: Added Placeholder extension block which is used to “pad” main ShellBags when they contain additional ShellBag items (Bags containing bags). Placeholder bags are not printed to the details pane

NEW: Added detection of Beef0010 blocks in 0x71 ShellBags.

NEW: Added detection of Beef0010 blocks in 0x2f ShellBags.

NEW: Added ability to double click on a row in the Messages window which will select the related node in the tree on the main window

NEW: Add GUID for OneDrive on Windows 10

NEW: Detect drive letters in 0x1F ShellBags.

CHANGE: Detect optional Beef0004 extension block in CDBurn ShellBags

CHANGE: Look for common signatures in several bags and act accordingly

FIX: Detect several fringe cases and set value to “!!! Unable to determine value !!!”. These need more research for full support

FIX: Remove early return in 0x4d bag which resulted in the ShellBag not displaying properly

### Version 0.5.0.3

NEW: Add new ShellBag type, ShellBagParseError, that is used as a placeholder when ShellBag contents aren't parsed properly.

### Version 0.5.0.2

NEW: Added support for 0x61 bags (FTP). Connection date and username (when saved) are reported  
NEW: Added support for Variable: FTP URI ShellBags. These show directories browsed on an FTP server and in many cases contains the timestamp on the FTP folder

CHANGE: More support for contents in BEEF000e extension blocks

CHANGE: Improve message when BagMRU key isn't found in a hive.

CHANGE: Updated drive letter icon to a hard drive vs green folder

CHANGE: Expand and collapse nodes now affects the selected node vs all nodes.

FIX: Some GUI fixes (when using expand/collapse nodes remove recursive icon)

### Version 0.5.0.1

FIX: Remove recursive icon when right clicking an entry in the grid

FIX: Minor manual updates

### Version 0.5.0.0

NEW: Completely rewritten manual

NEW: Added Time zone setting to Tools menu. Change the time zone BEFORE processing a hive to have all dates automatically adjusted for the selected timezone

NEW: Added -timezone command line argument to SBECmd.exe. Use -timezone="" for a list of valid time zones

NEW: Added -dedupe option to SBECmd. This will process all hives in a directory and remove any duplicate bags. See -help for details

FIX: Adjust Beef0025 version offset to always pull the right offset.

### Version 0.4.3.0

CHANGE: Switch from CSV extension to TSV for easier processing in Excel without having to import manually

FIX: Minor fixes to command line version if source directory is missing

### Version 0.4.2.0

FIX: Correct file system hint in details pane misreporting file system. The value in Misc column was correct but in some cases the details pane misreported the file system type

### Version 0.4.1.0

NEW: A shiny manual is now included

NEW: CSV output has been added. You can optionally include the raw hex from bags as well

NEW: Added SBECmd.exe which is the command line version of SBE.

NEW: Added FirstExplored column to grid and details section (when available)

NEW: Added LastExplored column to grid and details section (when available)

NEW: More GUIDs (Windows 10, RealPlayer Cloud, etc)

NEW: Added support for ShellBags withTypeID 0x01 special case

NEW: You can now drag and drop an offline hive on the grid, tree, or details pane to load

NEW: Added note to Miscellaneous column denoting what file system a bag came from based on MFT information (Thanks to David Cowen for idea and testing)

### Version 0.3.9.0

NEW: Added icon for CDBurn bags  
NEW: Added CTRL-A and CTRL-C functionality to main grid. This lets you quickly select all rows and copy selected rows to clipboard  
NEW: SBE now remembers its size and will restore the last size on startup  
CHANGE: Add detection of certain special signatures earlier in the process as these types of bags can occur in different places (CDBurn, etc)  
CHANGE: Rename “Variable: CDBurn” to “CDBurn” since they can be seen in different places

### Version 0.3.8.0

NEW: Added operating system information for beef0004 blocks based on Identifier property. 0x14 ⇒ Windows XP, 2003 | 0x26 ⇒ Windows Vista | 0x2a ⇒ Windows 2008, 7, 8.0 | 0x2e ⇒ Windows 8.1  
NEW: Report unmapped GUIDs to messages window  
NEW: Added support for 0x35 identifiers, which contain Unicode strings vs ascii  
NEW: Added support for ShellBag with type ID 0x64 and 0x65. These refer to browser history folders  
NEW: Even MORE GUIDS!!!!1  
CHANGE: Improvements to message form details pane  
CHANGE: Hide InternalID and HexView from grid field chooser

FIX: Deal with improperly terminated Unicode strings so a ? isn't displayed as the last character  
FIX: Handle embedded property stores in 0x71 (Control panel bags) vs reporting unknown GUID  
FIX: Lots of fringe case cleanup

### Version 0.3.7.0

NEW: Report unknown type IDs to Messages window so they can be reported  
NEW: Added support for Beef0010 extension block. This block contains property store data. Vector data in one of the sheets looks to contain additional property stores with additional data. More work to be done here  
NEW: Added support for Beef0021 extension block. This block has been seen to contain the URL where files have been downloaded from  
NEW: Added support for Beef0016 extension block. This looks to contain what was searched for (ie \*.inf)  
CHANGE: Do not force ‘Last write time’ column visible when loading an offline hive  
CHANGE: Added more GUIDs

FIX: Don't crash when grouping and a group is selected

### Version 0.3.6.0

NEW: Pull extension blocks out of property sheets in 0x00 and 0x1F ShellBag types. These appear to be the results of searches. More research is needed to confirm  
CHANGE: Clean up some messages

### Version 0.3.5.0

NEW: Added ‘Absolute path’ to grid which contains the full path from the Desktop to the bag item  
NEW: Added a few dozen more GUIDs

NEW: Alternate row colors in Messages window

NEW: Added exporting of messages to Excel via button on Messages window

NEW: Added 'Clear messages' button to Messages window

CHANGE: Made column headers look nicer by adding spaces (it's the little things sometimes)

CHANGE: Tweak Messages window to allow for message type, message, and details view to unclutter the display

CHANGE: Recoded Root folder shell item ShellBag type (0x1f) to find and add multiple extension blocks

CHANGE: Expand hex view to fill more space and move data interpreter to far right

CHANGE: Show wait cursor while doing things vs arrow

CHANGE: Add detection of bags related to zip file contents early vs inside each known bag type. This prevents a lot of future "unknown" types going forward

#### Appendix A – Contributors

The following people have contributed in one way or another during the development and refinement of SBE

\* SA/FE Devon P. Ackerman devon.ackerman@ic.fbi.gov, sadevonackerman@gmail.com

\* David Cowen @HECFBlog

\* Harlan Carvey @keydet89

\* Dan Pullega @4n6k

\* Mark Woan @woanware

#### Appendix B – Additional resources

\* <http://msdn.microsoft.com/en-us/library/aa965725%28v=vs.85%29.aspx>

\* <https://code.google.com/p/libfwsi/wiki/ShellFolderIdentifiers>

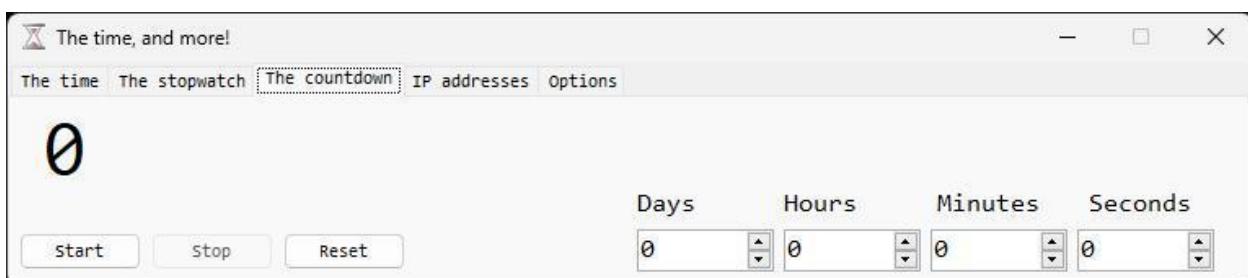
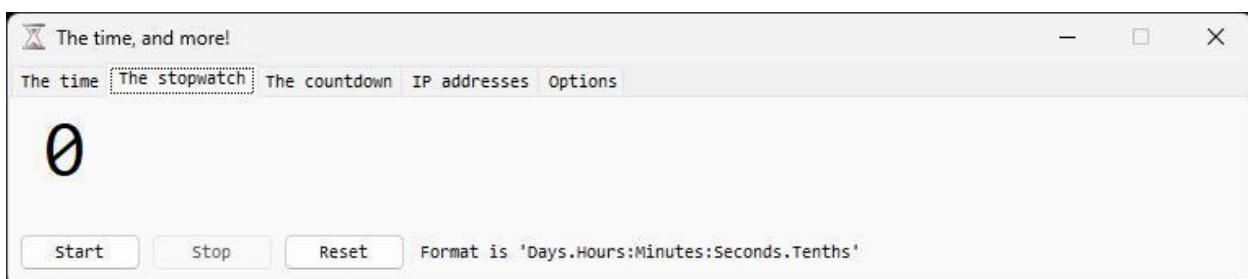
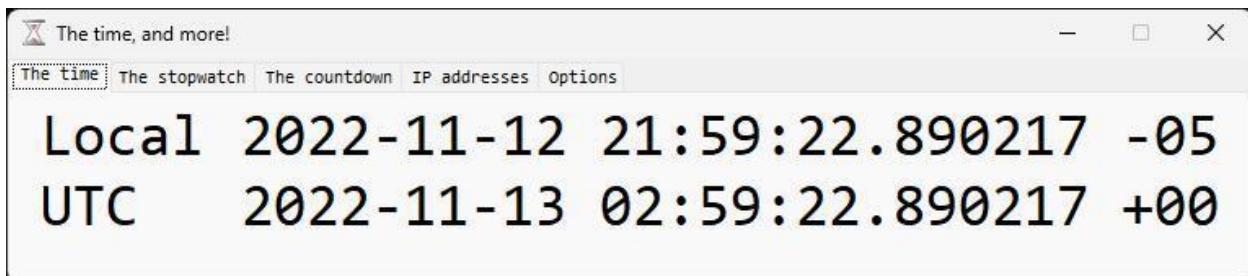
\* <https://docs.google.com/file/d/0B-VYGsDJPtZIVDNJQ3pWX0M1b1k/edit>\* <http://www.4n6k.com/2013/12/shellbags-forensics-addressing.html>\* <http://www.williballenthin.com/forensics/shellbags/index.html>

# TimeApp

## TimeApp Introduction

TimeApp is a tool created by Eric Zimmerman that provides multiple functions: current time, stopwatch, countdown timer, and current IP address.

## TimeApp Screenshots





## TimeApp References

### Associated GitHub Repositories

- <https://github.com/EricZimmerman/timeapp> is the GitHub repository for TimeApp

### Download TimeApp

TimeApp can be downloaded from <https://ericzimmerman.github.io/#!index.md>

# Timeline Explorer

## Timeline Explorer Introduction

Timeline Explorer is a tool created by Eric Zimmerman that can be used by forensic examiners (or anyone) to ingest CSV files (and a few other filetypes). Timeline Explorer can handle large files (over 14GB) and provides a user friendly experience for dealing with data contained within CSV files. Timeline Explorer can have multiple files open at a time (over 300) and contains many familiar keyboard shortcuts to move through analysis more efficiently.

Timeline Explorer uses [Plugins<sup>161</sup>](#) that will allow Timeline Explorer to ingest multiple filetypes. The benefit of these Plugins allows for the Line and Tag columns to be populated for the various supported file types as well as other enhancements to boolean (True/False) and timestamp values.

## Supported File Types

Timeline Explorer supports the following filetypes:

- .csv
- .tsv
- .xlsx
- .txt

Timeline Explorer only runs on Windows due to the [.NET Desktop Runtime<sup>162</sup>](#) currently only being available for Windows and not macOS or Linux.

## Ingesting Excel Spreadsheets (.xlsx)

Please note, when you ingest an Excel spreadsheet with multiple sheets, only the first sheet will load within Timeline Explorer. All other sheets will be ignored.

---

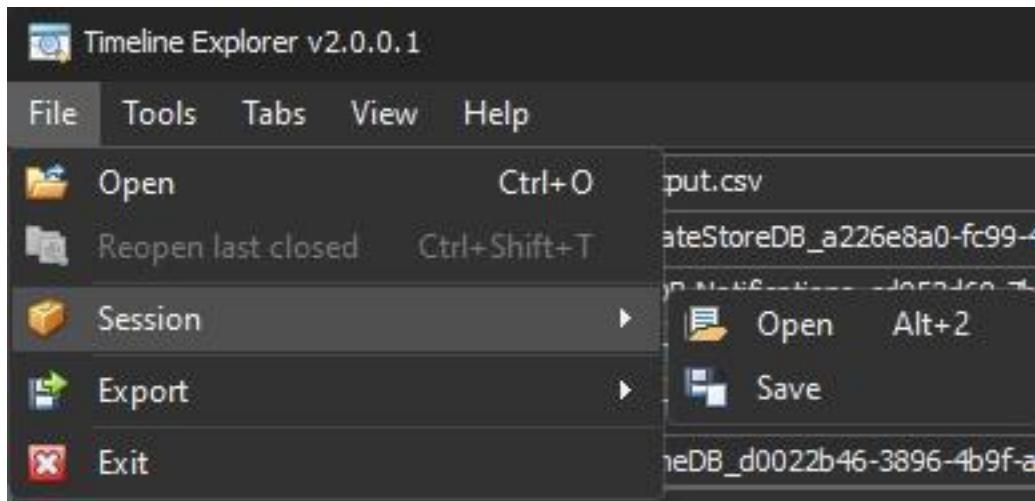
<sup>161</sup><https://github.com/EricZimmerman/TLEFilePlugins>

<sup>162</sup><https://dotnet.microsoft.com/en-us/download/dotnet/6.0>

# Timeline Explorer Features

## File Menu

The File menu has a couple of noteworthy features that we'll cover in this section



Timeline Explorer's File Menu

## Sessions

When you have multiple CSVs ingested into Timeline Explorer, you can save a session which will open all the same files that are opened at the time of session save. Timeline Explorer will create a `.tle_sess` file which is simply a JSON file that will look similar to the example below:

```
{  
    "SessionFiles": {  
        "C:\\\\temp\\\\20220913030423_RECcmd_Batch_Kroll_Batch_Output.csv": [  
            ],  
        "C:\\\\temp\\\\20220913030450_RECcmd_Batch_Kroll_Batch_Output.csv": [  
            ]  
    }  
}
```

Opening this particular session file will instruct Timeline Explorer to open the files specified. Additionally, any tagged rows will persist when saving and reloading sessions.

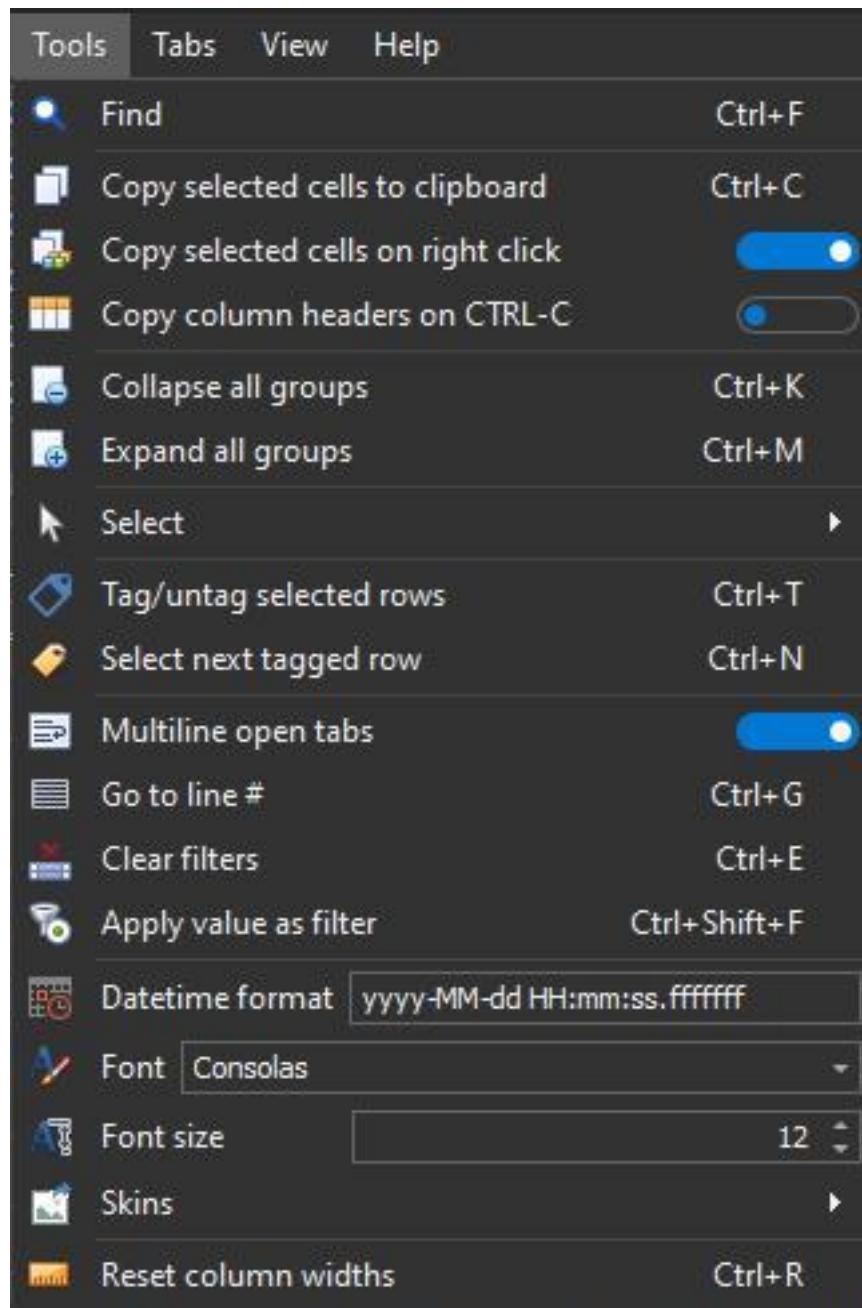
## Export

Timeline Explorer can export an opened file as CSV or XLSX. This may be helpful where if you have a CSV file ingested but want to export it as XLSX, or vice-versa. Please note, the Export function

is WYSIWYG, so whatever data is visible due to active layouts, filters, sorting, etc will be what is exported.

## Tools Menu

Timeline Explorer offers multiple useful features within the Tools menu.



Timeline Explorer's Tools Menu

## Find

Timeline Explorer provides an incredibly useful Find window where examiners can search for terms amongst all open files. The below example shows the term `microsoft` and the amount of hits within each open file. Double-clicking on a search result will warp to the file in question and automatically filter on the search term.

The screenshot shows the Timeline Explorer interface with a 'Find' window open. The 'Find' window has a search bar with 'microsoft' typed in, a 'Clear terms' button, and a 'Find' button highlighted with a blue border. Below the window, the main pane displays a table titled 'Search results (double click to view)'. The table has columns for 'File Name', 'Term', and 'Hits'. The data shows multiple CSV files from various dates containing the term 'microsoft' with varying hit counts. A status bar at the bottom indicates '95. Found 3,870 results in 0.08544 seconds'.

File Name	Term	Hits
20220913030423_RECcmd_Batch_Kroll_Batch_Output.csv	microsoft	1876
20220913030450_RECcmd_Batch_Kroll_Batch_Output.csv	microsoft	1876
20220916023431858235_Windows_WindowsUpdateStoreDB_a226e8a0-fc99-499d-8797-ba80179f349f.csv	microsoft	12
20220916023434822104_Windows_WindowsUpdateStoreDB_a226e8a0-fc99-499d-8797-ba80179f349f.csv	microsoft	11
20220916023435039963_Windows_NotificationsDB-Notifications_cd952d69-7b3e-4d13-9810-8d987155bc58.csv	microsoft	8
20220916023435231499_Windows_NotificationsDB-Notifications_cd952d69-7b3e-4d13-9810-8d987155bc58.csv	microsoft	6
20220916023435926946_Windows_ActivityPackageId_d0022b46-3896-4b9f-a9e5-499241eba806.csv	microsoft	26
20220916023436092201_Windows_ActivityPackageId_d0022b46-3896-4b9f-a9e5-499241eba806.csv	microsoft	10

Timeline Explorer's Find Window

## Right-Click to Copy

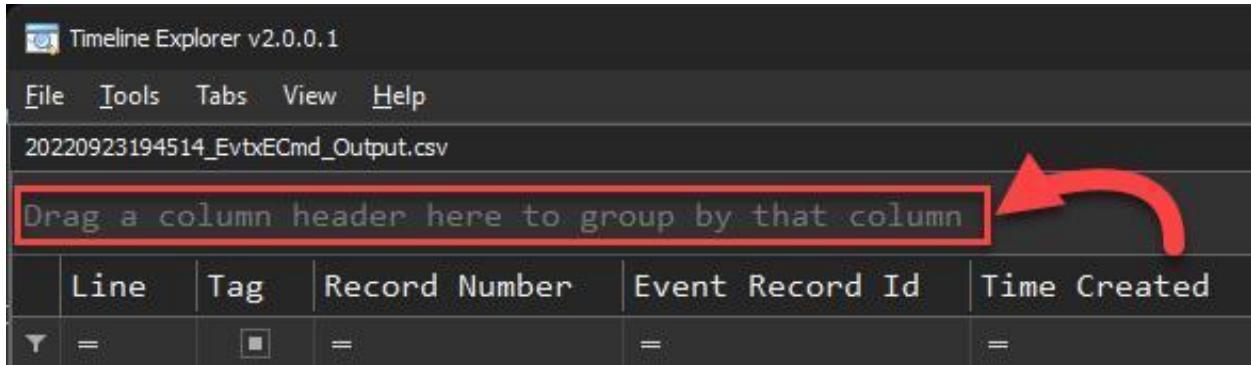
One of the most convenient features Timeline Explorer offers is the ability to carry out copy functions with a simple right-click of the mouse. This allows for minimal, if any, interaction with a keyboard when copying data from a file opened in Timeline Explorer. Every click saved matters!

## Copy Column Headers When Copying

This can be useful for when headers need to be copied out of a file that's open within Timeline Explorer.

## Groups

Timeline Explorer allows examiners to grab a column header and drag above to the box which contains the text Drag a column header here to group by that column, as seen below.



Line	Tag	Record Number	Event Record Id	Time Created
=	■	=	=	=

An Example of the Grouping by Column Header Workflow in Timeline Explorer

Grabbing the Time Created column header and dragging it into the blank area above the column headers will create a grouping that can be very helpful during everyday analysis.

## RECmd

Below is an example of grouping RECmd output by the Hive Type column.

The screenshot shows the Timeline Explorer interface with two tabs open: '20220913030423\_RECcmd\_Batch\_Kroll\_Batch\_Output.csv' and '20220913030450\_RECcmd\_Batch\_Kroll\_Batch\_Output.csv'. The left tab is active. The main pane displays a table with columns: Line, Tag, Hive Path, Description, Category, Key Path, Value Name, and Value Type. A dropdown menu labeled 'Hive Type' is open, showing collapsed sections for 'UsrClass' (Count=17) and 'System' (Count=861). The 'UsrClass' section contains rows from line 2698 to 2714, all showing 'MuiCache' under 'Description' and 'Program Execu...' under 'Category'. The 'System' section contains rows from line 369 to 370, showing 'Windows Boot...' and 'ControlSet C...' under 'Hive Path' respectively, and 'System Info' under both 'Description' and 'Category'. The bottom status bar shows 'Total lines 2,732 | Visible lines 2,732 | Open files: 2 | Search options ...'.

Line	Tag	Hive Path	Description	Category	Key Path	Value Name	Value Type
<b>Hive Type:</b> UsrClass (Count=17)							
2698	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	LangID	RegBinary
2699	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Program Files\Internet...	RegSz
2700	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Program Files\Internet...	RegSz
2701	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\system32\explo...	RegSz
2702	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\system32\explo...	RegSz
2703	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\system32\shell...	RegSz
2704	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\system32\shell...	RegSz
2705	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\System32\fsquia...	RegSz
2706	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\System32\fsquia...	RegSz
2707	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\System32\WFS.e...	RegSz
2708	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\System32\WFS.e...	RegSz
2709	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\Explorer.exe.F...	RegSz
2710	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\Explorer.exe.A...	RegSz
2711	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Program Files\VMware\V...	RegSz
2712	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Program Files\VMware\V...	RegSz
2713	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\System32\cmd.e...	RegSz
2714	□	D:\DFIRArtifactMuseum\W...	MuiCache (Vi...	Program Execu...	S-1-5-21-2212929841-27309...	C:\Windows\System32\cmd.e...	RegSz
<b>Hive Type:</b> System (Count=861)							
369	□	D:\DFIRArtifactMuseum\W...	Windows Boot...	System Info	ROOT\Setup	SystemPartition	RegSz
370	□	D:\DFIRArtifactMuseum\W...	ControlSet C...	System Info	ROOT\Select	Current	RegDword

### Grouping Columns in Timeline Explorer

When pushing CTRL+K, the Hive Type groups will collapse which can then be expanded with CTRL+M.

## EvtxECmd

Below is an example of grouping EvtxECmd output by the Map Description column.

## Timeline Explorer with Multiline Tabs Enabled

Below is an example of stacking column headers to form a complex grouping.

Line	Tag	Record Number	Event Record Id	Time Created	Event Id	Level	Process Id	Computer	User Id
<b>Channel</b> ▾									
<b>Provider</b> ▾									
Enter text to search... Find									
> Channel: PowerBiosServerLog (Count: 764)									
< Channel: Security (Count: 30,086)									
> Provider: Microsoft-Windows-Eventlog (Count: 7)									
> Provider: Microsoft-Windows-Security-Auditing (Count: 30,034)									
> Provider: VSSAudit (Count: 45)									
> Channel: SentinelOne/Operational (Count: 726)									
> Channel: Setup (Count: 89)									
< Channel: System (Count: 27,310)									
> Provider: Application Popup (Count: 10)									
> Provider: BTHUSB (Count: 317)									
> Provider: disk (Count: 3)									
> Provider: EventLog (Count: 286)									
> Provider: i8042prt (Count: 6)									
> Provider: MEIx64 (Count: 45)									
> Provider: Microsoft-Windows-BitLocker-Driver (Count: 40)									
> Provider: Microsoft-Windows-Dhcp-Client (Count: 279)									
< Channel: Win32-Kernel (Count: 201)									
D:\Tutorial\mount\EventLogs\20220923194514_EvtxECmd_Output.csv									
Total lines 356,801   Visible lines 356,801   Open files: 1   Search options ...									

Timeline Explorer with Multiline Tabs Enabled

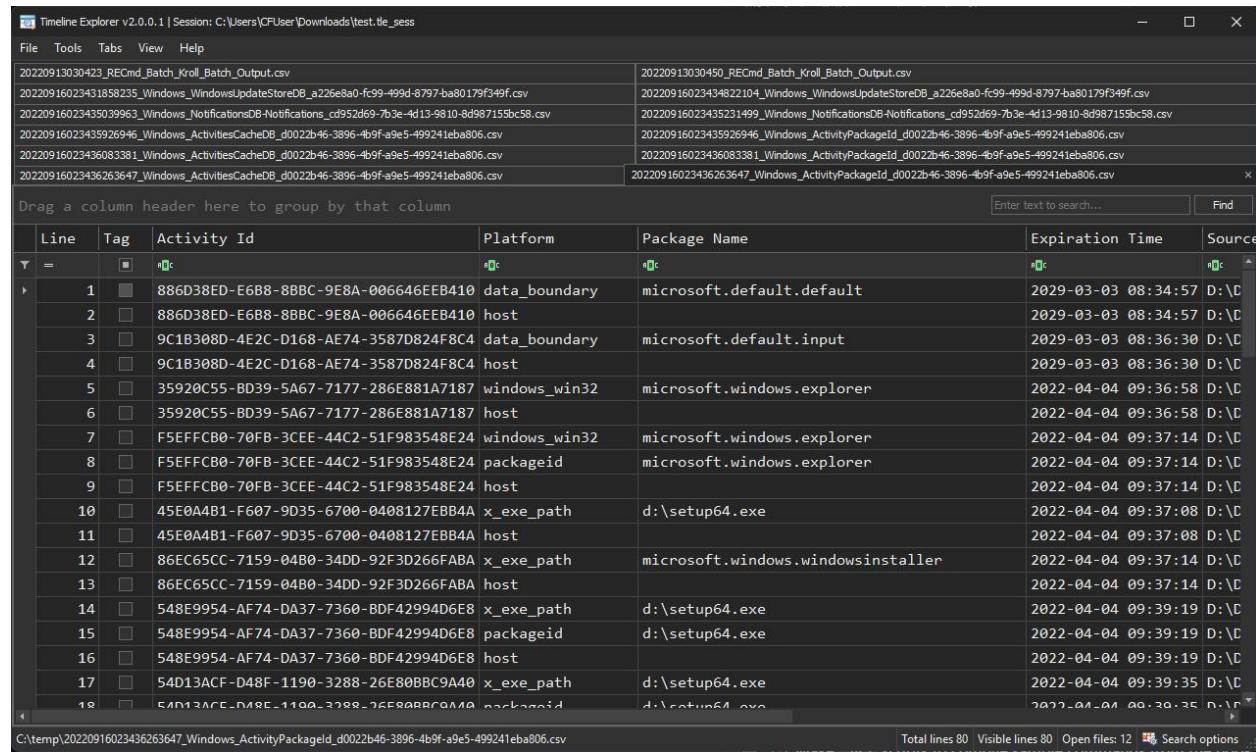
With this particular grouping, examiners can see how many overall events resides in the Security.evt file as well as how many events exist within each Provider that logs to that particular .evt file.

## Tagging

As long as there are Line and Tag columns visible for a file ingested into Timeline Explorer, examiners can take advantage of the Tag feature. Rows that contain data of interest can be tagged by checking the box in the Tag column. This can be useful for when analysis is complete on a given file to where an examiner is ready to filter on only the rows of interest. This can be done by filtering on the Tag column for checked boxes only. This will allow for all untagged rows to disappear and only rows that were tagged by the examiner will appear.

## Multiline Open Tabs

Below is an example of Timeline Explorer with Multiline Tabs enabled.



The screenshot shows the Timeline Explorer interface with several tabs open at the top. The tabs are:

- 20220913030423\_RECcmd\_Batch\_Kroll\_Batch\_Output.csv
- 20220916023431858235\_Windows\_WindowsUpdateStoreDB\_a226ea8af-f99-49d8-8797-ba80179f349f.csv
- 20220916023435039963\_Windows\_NotificationsDB-Notifications\_cd952d69-7b3e-4d13-9810-8d987155bc58.csv
- 20220916023435926946\_Windows\_ActivitiesCacheDb\_d0022b46-3896-4b9f-a9e5-499241eba806.csv
- 20220916023436083381\_Windows\_ActivitiesCacheDb\_d0022b46-3896-4b9f-a9e5-499241eba806.csv
- 20220916023436263647\_Windows\_ActivitiesPackageId\_d0022b46-3896-4b9f-a9e5-499241eba806.csv

The main pane displays a table of activity data with columns: Line, Tag, Activity Id, Platform, Package Name, Expiration Time, and Source. The table contains approximately 18 rows of data, mostly from host processes like Microsoft Default and Windows Explorer.

Timeline Explorer with Multiline Tabs Enabled

Below is an example of Timeline Explorer with Multiline Tabs disabled.

Line	Tag	Activity Id	Platform	Package Name	Expiration Time	Source
1		886D38ED-E6B8-8BBC-9E8A-006646EEB410	data_boundary	microsoft.default.default	2029-03-03 08:34:57	D:\D
2		886D38ED-E6B8-8BBC-9E8A-006646EEB410	host		2029-03-03 08:34:57	D:\D
3		9C1B308D-4E2C-D168-AE74-3587D824F8C4	data_boundary	microsoft.default.input	2029-03-03 08:36:30	D:\D
4		9C1B308D-4E2C-D168-AE74-3587D824F8C4	host		2029-03-03 08:36:30	D:\D
5		35920C55-BD39-5A67-7177-286E881A7187	windows_win32	microsoft.windows.explorer	2022-04-04 09:36:58	D:\D
6		35920C55-BD39-5A67-7177-286E881A7187	host		2022-04-04 09:36:58	D:\D
7		F5EFFCB0-70FB-3CEE-44C2-51F983548E24	windows_win32	microsoft.windows.explorer	2022-04-04 09:37:14	D:\D
8		F5EFFCB0-70FB-3CEE-44C2-51F983548E24	packageid	microsoft.windows.explorer	2022-04-04 09:37:14	D:\D
9		F5EFFCB0-70FB-3CEE-44C2-51F983548E24	host		2022-04-04 09:37:14	D:\D
10		45E0A4B1-F607-9D35-6700-0408127EBB4A	x_exe_path	d:\setup64.exe	2022-04-04 09:37:08	D:\D
11		45E0A4B1-F607-9D35-6700-0408127EBB4A	host		2022-04-04 09:37:08	D:\D
12		86EC65CC-7159-04B0-34DD-92F3D266FABA	x_exe_path	microsoft.windows.windowsinstaller	2022-04-04 09:37:14	D:\D
13		86EC65CC-7159-04B0-34DD-92F3D266FABA	host		2022-04-04 09:37:14	D:\D
14		548E9954-AF74-DA37-7360-BDF42994D6E8	x_exe_path	d:\setup64.exe	2022-04-04 09:39:19	D:\D
15		548E9954-AF74-DA37-7360-BDF42994D6E8	packageid	d:\setup64.exe	2022-04-04 09:39:19	D:\D
16		548E9954-AF74-DA37-7360-BDF42994D6E8	host		2022-04-04 09:39:19	D:\D
17		54D13ACF-D48F-1190-3288-26E80BBC9A40	x_exe_path	d:\setup64.exe	2022-04-04 09:39:35	D:\D
18		54D13ACF-D48F-1190-3288-26E80BBC9A40	packageid	d:\setup64.exe	2022-04-04 09:39:35	D:\D
19		54D13ACF-D48F-1190-3288-26E80BBC9A40	host		2022-04-04 09:39:35	D:\D
20		575EE26E-05BC-5EC3-BFB9-EA301C22F174	windows_universal	microsoft.windowsterminal_8wekyb3d8bbwe	2022-04-04 09:40:29	D:\D
21		575EE26E-05BC-5EC3-BFB9-EA301C22F174	host		2022-04-04 09:40:29	D:\D
22		9FE5479D-0990-8E20-4A94-EC0AD2050B85	windows_universal	microsoft.windowsterminal_8wekyb3d8bbwe	2022-03-12 08:42:55	D:\D

Timeline Explorer with Multiline Tabs Disabled

## Clear Filters

CTRL+E is one of the most useful keyboard shortcuts in Timeline Explorer. Regardless of how many filters are applied by an examiner, this shortcut will clear them all.

## Applying Value as a Filter

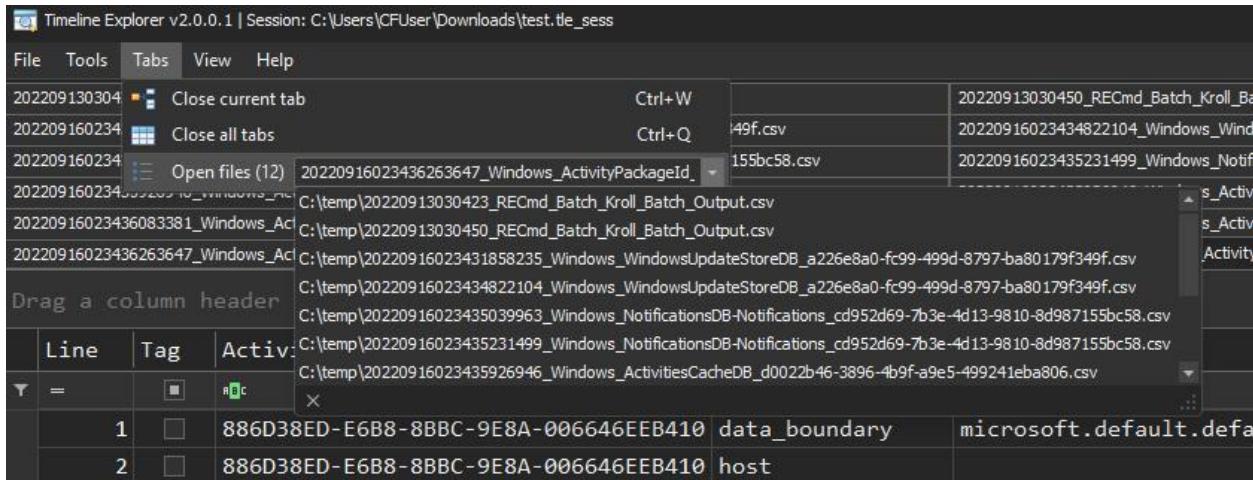
CTRL+SHIFT+F is another useful keyboard shortcut available in Timeline Explorer. When a given cell is highlighted in Timeline Explorer, this keyboard shortcut will allow for the value within that cell to be applied as a filter in the column it resides in. The filter type can be changed once the value is populated in the column header filter box.

## Reset Column Widths

As examinations progress, columns get resized and it can be a pain to manually resize the columns back to a reasonable size. CTRL+R provides the quickest method to reset all columns to their default width.

## Tabs Menu

The Tabs menu provides examiners with an overview of all files opened within a Timeline Explorer instance.

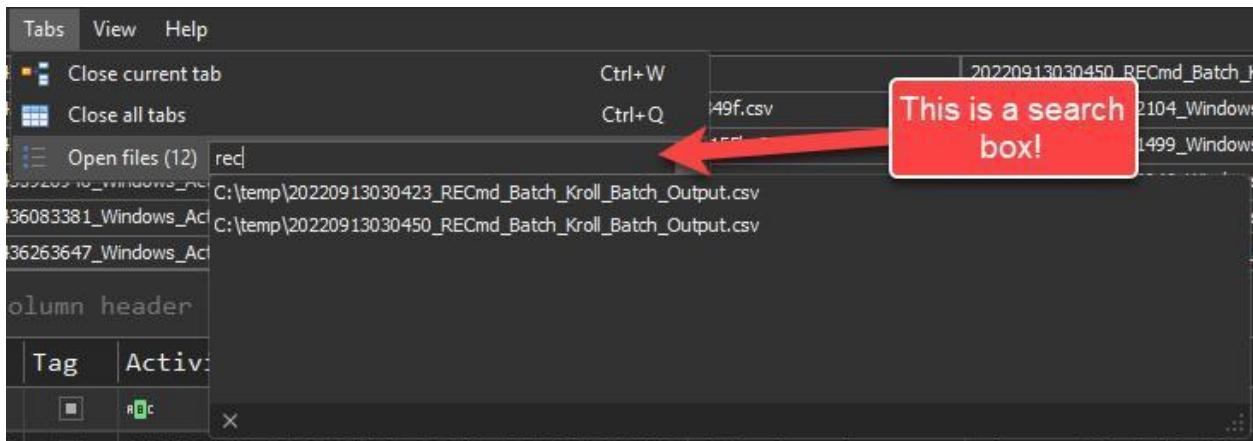


Timeline Explorer v2.0.0.1 | Session: C:\Users\CFUser\Downloads\test.tle\_sess

File	Tools	Tabs	View	Help
		Close current tab		Ctrl+W
		Close all tabs		Ctrl+Q
		Open files (12)		20220913030450_RECcmd_Batch_Kroll_Batch_Output.csv
202209130304				149f.csv
202209160234				20220916023434822104_Windows_Wind
202209160234				155bc58.csv
202209160234				20220916023435231499_Windows_Notify
202209160234				s_Activ
202209160234				s_Activ
202209160234				Activity
Drag a column header				
	Line	Tag	Activ:	
	=	■	abc	x
1	□	886D38ED-E6B8-8BBC-9E8A-006646EEB410	data_boundary	microsoft.default.defa
2	□	886D38ED-E6B8-8BBC-9E8A-006646EEB410	host	

Timeline Explorer's Tabs Menu

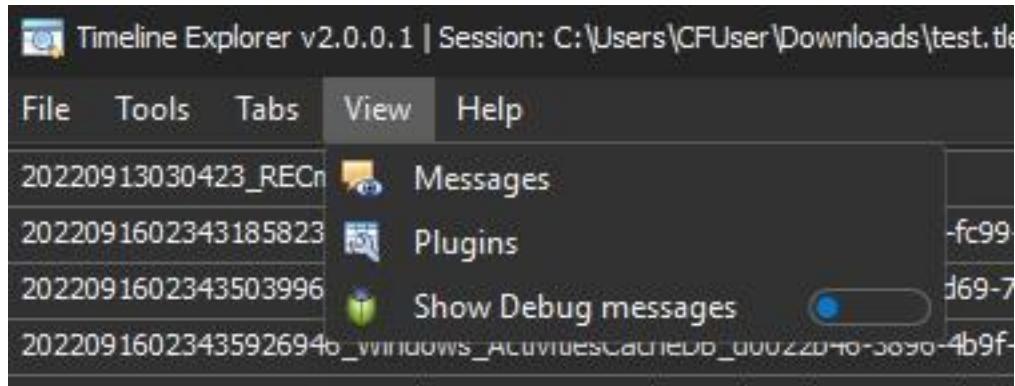
In addition to providing an easy way to see a massive amount of tabs open within Timeline Explorer, this box also serves as a search box so you can filter on a specific open file that you want to analyze.



Searching within Timeline Explorer's Tabs Menu

## View Menu

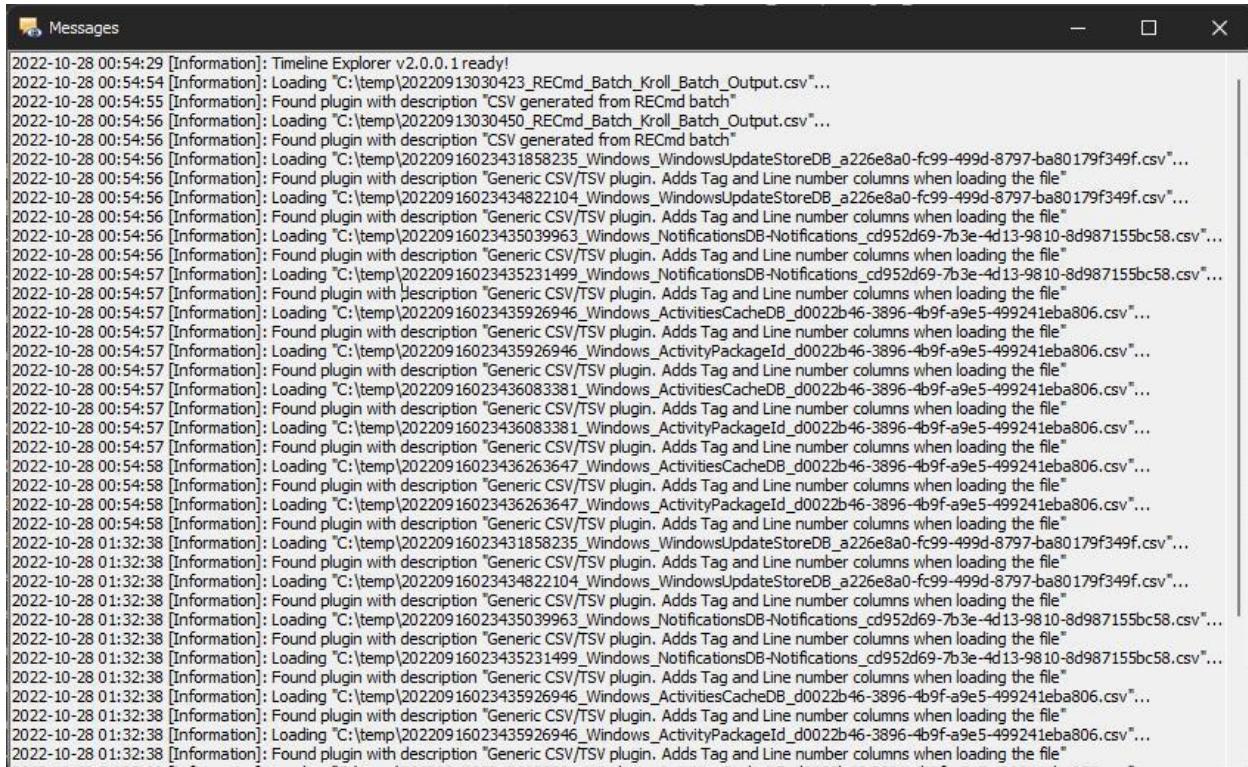
The View menu provides helpful information when trying to troubleshoot issues with Timeline Explorer.



Timeline Explorer's View Menu

## Messages

Timeline Explorer offers a Messages window that can provide insight as to how long a file takes to ingest into Timeline Explorer, which plugin was used to process a file during ingestion, and error messages that may come up while ingesting unsupported output. As always, toggling Debug Messages will provide much more verbose messaging in this window but it will also likely drastically slow the ingestion of a file.

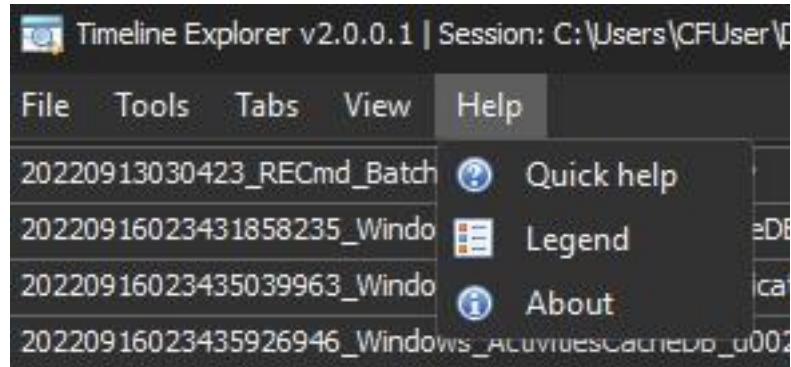


```
2022-10-28 00:54:29 [Information]: Timeline Explorer v2.0.0.1 ready!
2022-10-28 00:54:54 [Information]: Loading "C:\temp\20220916030423_RECmd_Batch_Kroll_Batch_Output.csv"...
2022-10-28 00:54:55 [Information]: Found plugin with description "CSV generated from RECmd batch"
2022-10-28 00:54:56 [Information]: Loading "C:\temp\20220916030450_RECcmd_Batch_Kroll_Batch_Output.csv"...
2022-10-28 00:54:56 [Information]: Found plugin with description "CSV generated from RECmd batch"
2022-10-28 00:54:56 [Information]: Loading "C:\temp\20220916023431858235_Windows_WindowsUpdateStoreDB_a226e8a0-fc99-499d-8797-ba80179f349f.csv"...
2022-10-28 00:54:56 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 00:54:56 [Information]: Loading "C:\temp\20220916023434822104_Windows_WindowsUpdateStoreDB_a226e8a0-fc99-499d-8797-ba80179f349f.csv"...
2022-10-28 00:54:56 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 00:54:56 [Information]: Loading "C:\temp\20220916023435039963_Windows_NotificationsDB-Notifications_cd952d69-7b3e-4d13-9810-8d987155bc58.csv"...
2022-10-28 00:54:56 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 00:54:57 [Information]: Loading "C:\temp\20220916023435231499_Windows_NotificationsDB-Notifications_cd952d69-7b3e-4d13-9810-8d987155bc58.csv"...
2022-10-28 00:54:57 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 00:54:57 [Information]: Loading "C:\temp\20220916023435926946_Windows_ActivitiesCacheDB_d0022b46-3896-4b9f-a9e5-499241eba806.csv"...
2022-10-28 00:54:57 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 00:54:57 [Information]: Loading "C:\temp\20220916023435926946_Windows_ActivityPackageId_d0022b46-3896-4b9f-a9e5-499241eba806.csv"...
2022-10-28 00:54:57 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 00:54:57 [Information]: Loading "C:\temp\20220916023436083381_Windows_ActivitiesCacheDB_d0022b46-3896-4b9f-a9e5-499241eba806.csv"...
2022-10-28 00:54:57 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 00:54:57 [Information]: Loading "C:\temp\20220916023436083381_Windows_ActivityPackageId_d0022b46-3896-4b9f-a9e5-499241eba806.csv"...
2022-10-28 00:54:57 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 00:54:58 [Information]: Loading "C:\temp\20220916023436263647_Windows_ActivitiesCacheDB_d0022b46-3896-4b9f-a9e5-499241eba806.csv"...
2022-10-28 00:54:58 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 01:32:38 [Information]: Loading "C:\temp\20220916023431858235_Windows_WindowsUpdateStoreDB_a226e8a0-fc99-499d-8797-ba80179f349f.csv"...
2022-10-28 01:32:38 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 01:32:38 [Information]: Loading "C:\temp\20220916023434822104_Windows_WindowsUpdateStoreDB_a226e8a0-fc99-499d-8797-ba80179f349f.csv"...
2022-10-28 01:32:38 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 01:32:38 [Information]: Loading "C:\temp\20220916023435039963_Windows_NotificationsDB-Notifications_cd952d69-7b3e-4d13-9810-8d987155bc58.csv"...
2022-10-28 01:32:38 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 01:32:38 [Information]: Loading "C:\temp\20220916023435926946_Windows_ActivitiesCacheDB_d0022b46-3896-4b9f-a9e5-499241eba806.csv"...
2022-10-28 01:32:38 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
2022-10-28 01:32:38 [Information]: Loading "C:\temp\20220916023435926946_Windows_ActivityPackageId_d0022b46-3896-4b9f-a9e5-499241eba806.csv"...
2022-10-28 01:32:38 [Information]: Found plugin with description "Generic CSV/TSV plugin. Adds Tag and Line number columns when loading the file"
```

Timeline Explorer's Messages Window

## Help Menu

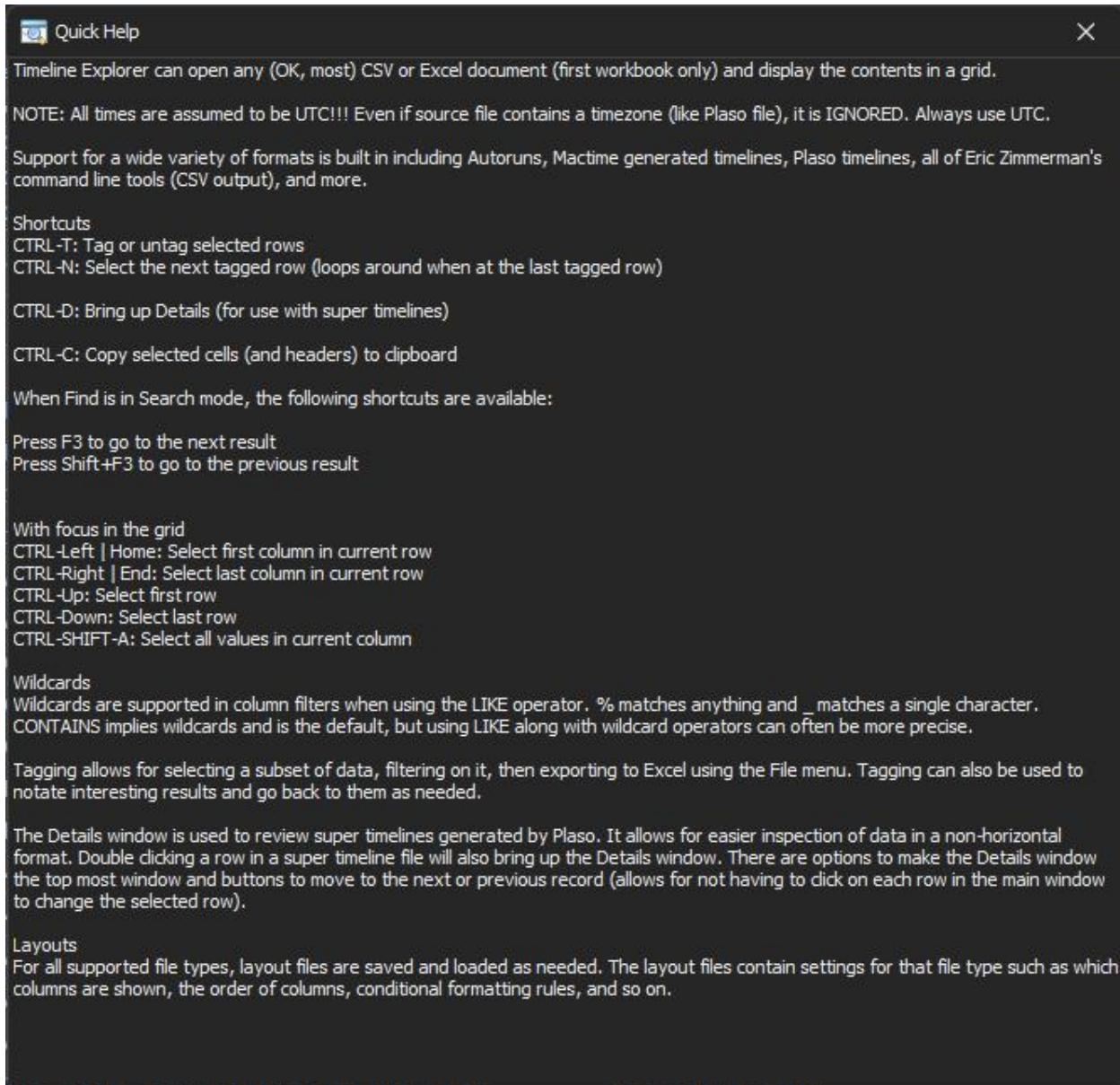
Timeline Explorer provides helpful resources within the Help menu.



Timeline Explorer's Help Menu

## Quick Help

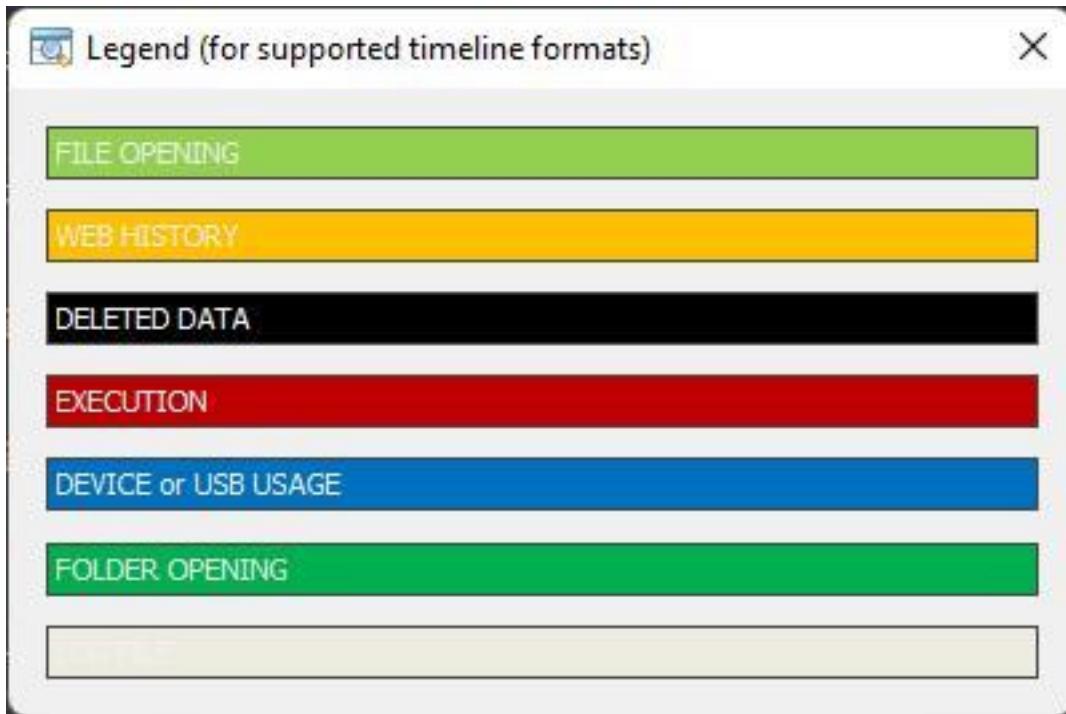
Below is the information located in Help -> Quick help.



Timeline Explorer's Quick Help

## Legend

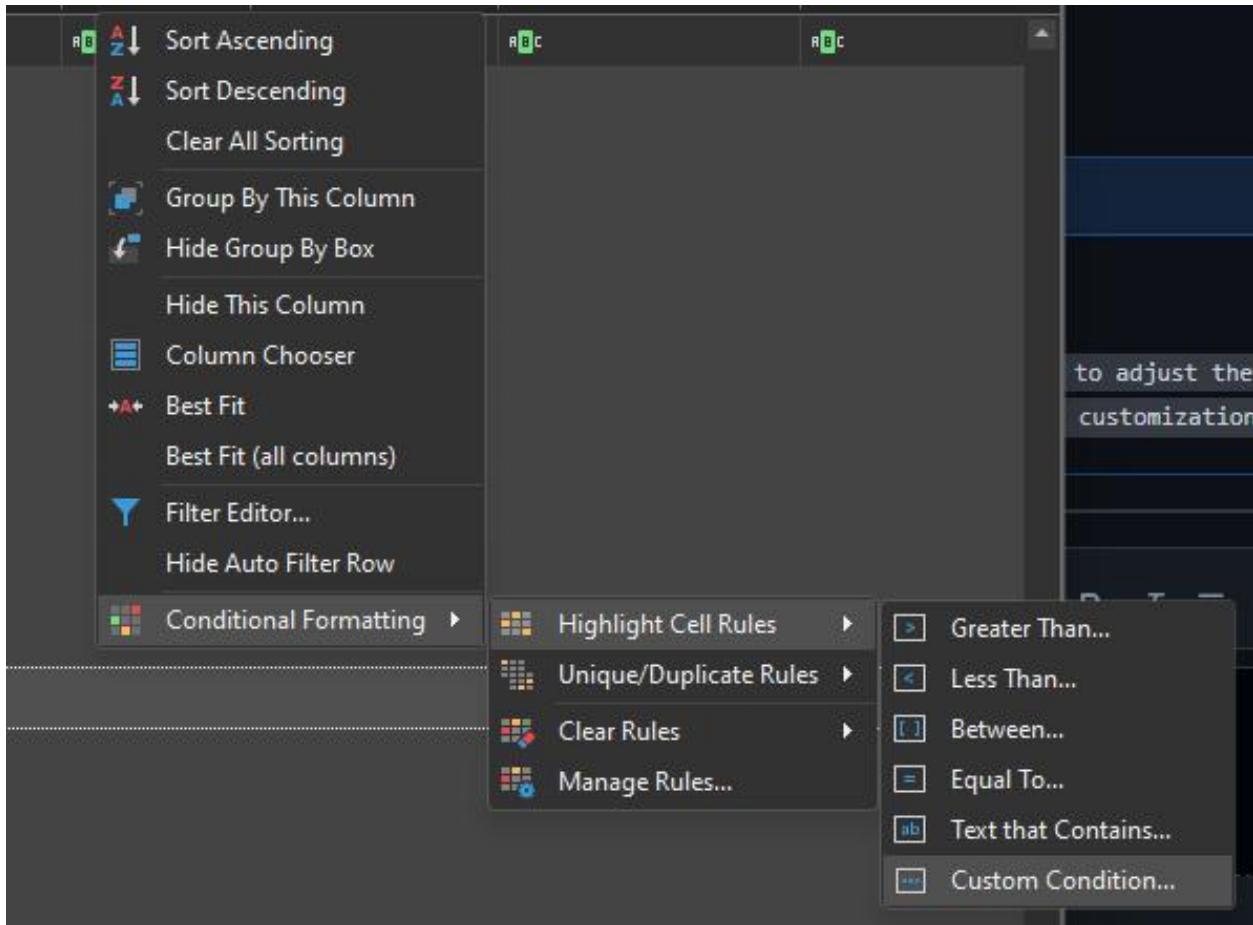
When ingesting Super Timelines, the following Legend can prove to be helpful.



Timeline Explorer's Legend for Super Timelines

## Conditional Formatting

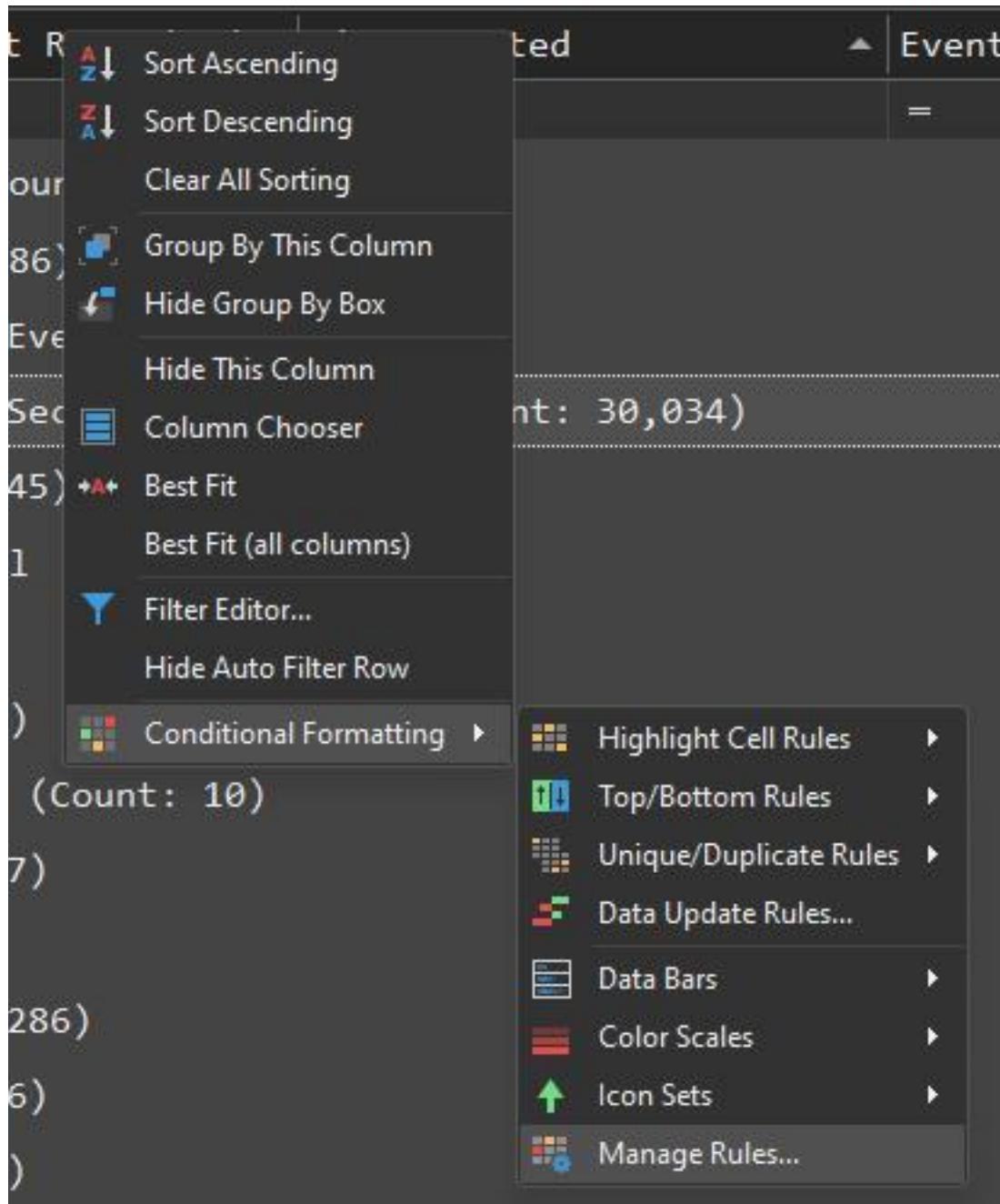
### Custom Conditions



Timeline Explorer's Conditional Formatting -> Manage Rules Location

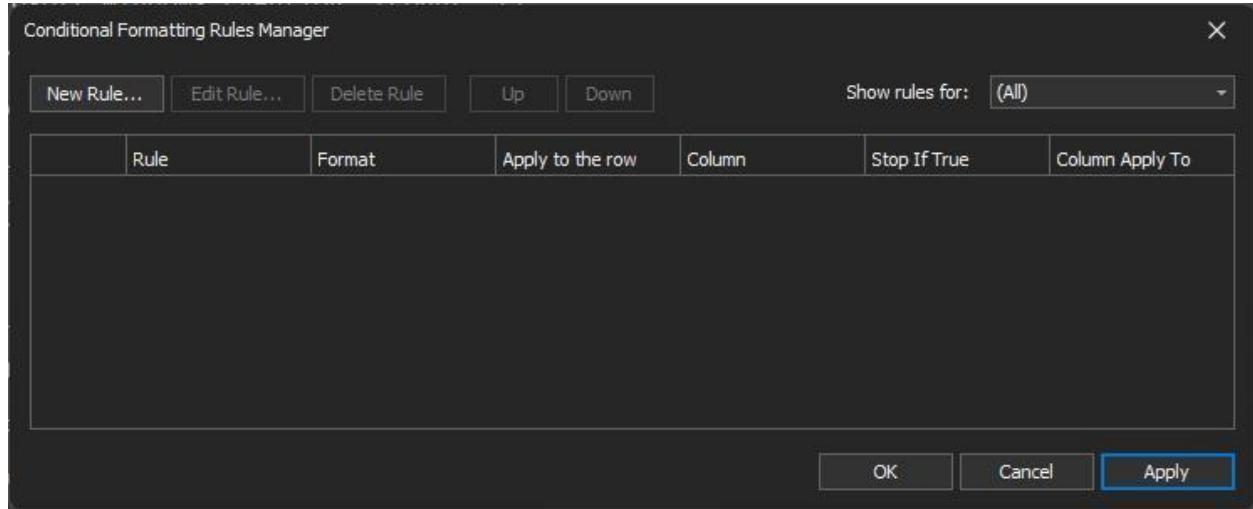
## Manage Rules

Timeline Explorer allows for examiners to set conditional formatting for certain criteria within rows, columns, and/or cells. This can be done using the Manage Rules menu.



Timeline Explorer's Conditional Formatting -> Manage Rules Location

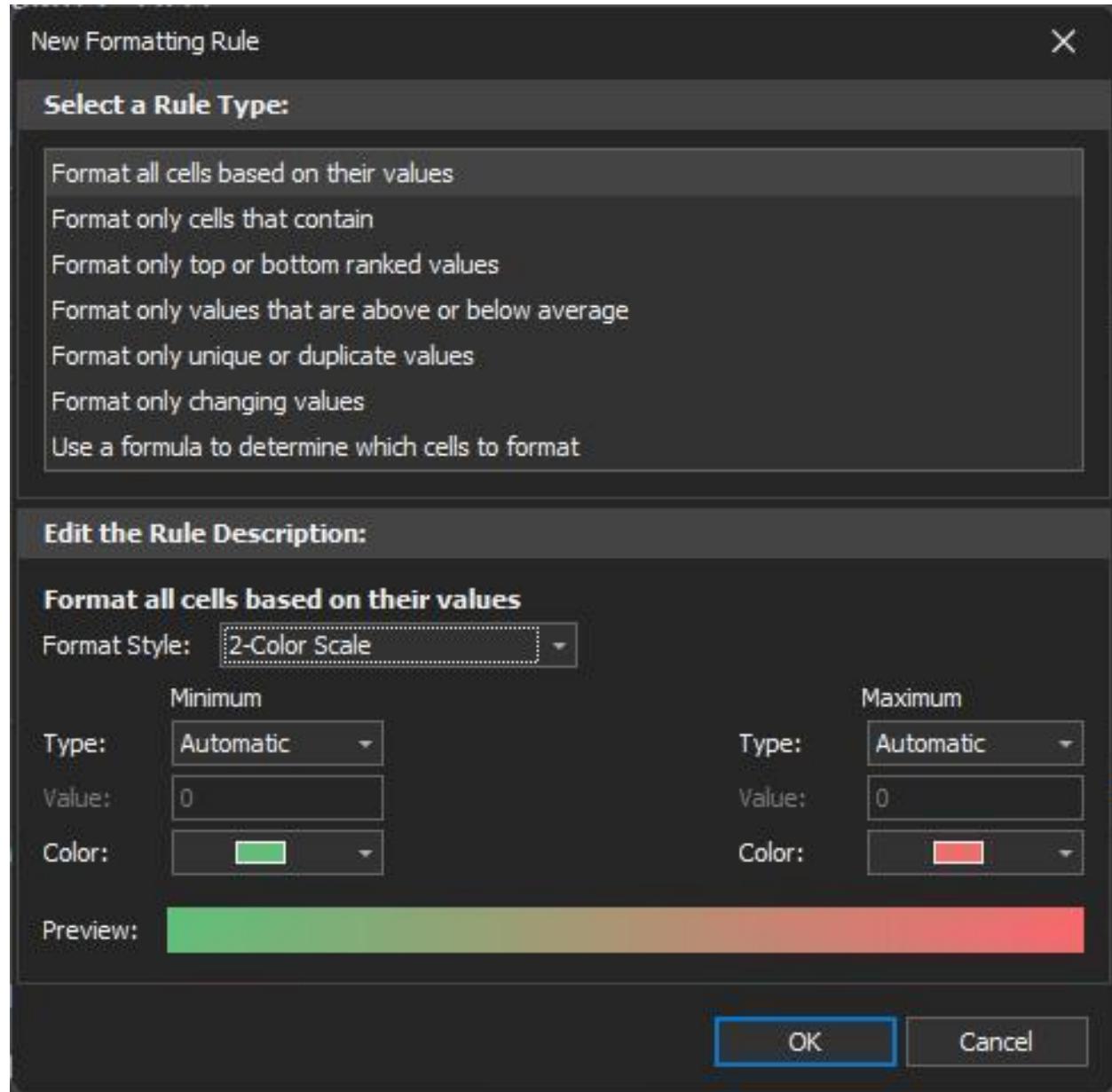
Clicking on Manage Rules will bring up this Rules Manager window.



Timeline Explorer's Conditional Formatting -> Manage Rules Location

## Example Rule

Clicking New Rule in the top left will bring up the following window.



Timeline Explorer's Conditional Formatting -> Manage Rules Location

Below is an example of what highlighting every row containing the string successful in the Map Description column looks like.

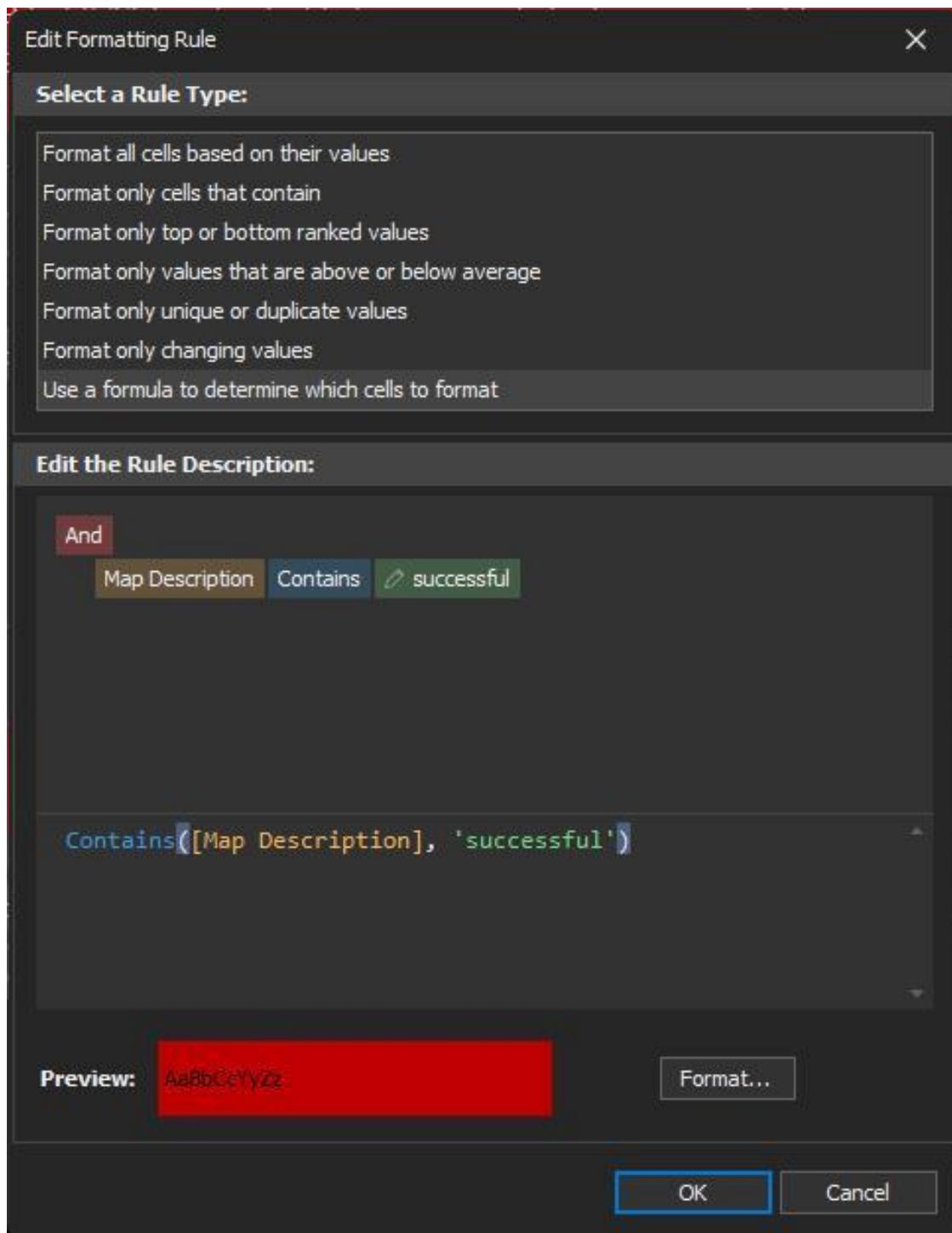
#### Highlighting Every Row That Contains “Successful”

Diving deeper into the Rules Manager will show the settings for applying this conditional formatting to every row containing the desired string.

The screenshot shows the 'Conditional Formatting Rules Manager' dialog box. At the top left is the title bar. Below it is a toolbar with buttons for 'New Rule...', 'Edit Rule...', 'Delete Rule...', 'Up', 'Down', and a dropdown menu 'Show rules for: (All)'. The main area contains a table with columns: Rule, Format, Apply to the row, Column, Stop If True, and Column Apply To. A single rule is listed: 'Formula: Contains([M-AaBbCc]YyZz)'. The 'Format' column for this rule is highlighted with a red background. The 'Column' column shows 'Map Description'. The 'Stop If True' and 'Column Apply To' columns both have '(None)' selected. At the bottom right are buttons for 'OK', 'Cancel', and 'Apply'.

## Example Rule Part 1

Using the bottom formula option allows the examiner to craft a filter similar to the filter editor used in the main Timeline Explorer window.



Example Rule Part 2

## Timeline Explorer Settings

A TLE\_settings.xml file is located within .\TimelineExplorer\Settings which stores user preferences in XML format. These are effectively storing the preferences for the options that can be modified by the user within the Tools menu. An example can be seen below:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <Settings xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.\ 
3 datacontract.org/2004/07/TimelineExplorer.Classes">
4     <CopyColumnHeaders>
5         false
6     </CopyColumnHeaders>
7     <CopyRightClick>
8         true
9     </CopyRightClick>
10    <DateTimeFormat>
11        yyyy-MM-dd HH:mm:ss.fffffff
12    </DateTimeFormat>
13    <DetailsOnTop>
14        false
15    </DetailsOnTop>
16    <FirstPinnedColumns xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/\ 
17 Arrays" />
18    <FontName>
19        Consolas
20    </FontName>
21    <FontSize>
22        12
23    </FontSize>
24    <MultilineTabs>
25        true
26    </MultilineTabs>
27    <Theme>
28        Office 2019 Black
29    </Theme>
30 </Settings>
```

A public copy of this exact TLE\_settings.xml file is hosted on Andrew Rathbun's GitHub [here<sup>163</sup>](#).

---

<sup>163</sup><https://github.com/AndrewRathbun/TimelineExplorerSettings>

## **DateTFormat**

It is very important that the .fffffff values are added to this whether in the Timeline Explorer GUI or directly into the XML file itself. Without these subseconds enabled, it will be impossible to detect timestamping<sup>164</sup> within MFTECmd output or leverage more precise timestamps that other EZ Tools provide.

# **Timeline Explorer Layout Files**

## **What Are Layout Files?**

Layout files are located at .\ZimmermanTools\net6\TimelineExplorer\Layouts\Default which provide the default layouts for output that's supported with a Plugin. If you move columns around, hide columns, or anything else, those preferences will be stored in layout files with the .layout.local file extension. If you ever want to revert to the default, delete your .layout.local files and Timeline Explorer will revert to the .layout files within the .\ZimmermanTools\net6\TimelineExplorer\Layouts\Default directory.

---

<sup>164</sup><https://www.kroll.com/en/insights/publications/cyber/anti-forensic-tactics/anti-forensics-tactics-timestamping>

# Timeline Explorer Plugins

Plugins are written in C# (as are all of Eric's tools) which aim to provide better support for ingested CSV output.

## Expected Headers

Each Plugin looks for headers within CSV output so it can know that a plugin matches with the tool output being ingested.

### Why does Timeline Explorer need plugins?

It doesn't NEED plugins but plugins will allow for a better experience when dealing with supported tool output.

For instance, [here<sup>165</sup>](#) is an excerpt from the EZ Tools plugin where the plugin is looking for certain headers within the CSV in order to ingest the output properly:

```
ExpectedHeaders = new HashSet<string>(StringComparer.OrdinalIgnoreCase)
{
    "entrynumber,sequencenumber,inuse,parententrynumber,parentsequencenumber,par
    th,filename,extension,filesize,referencecount,reparsetarget,isdirectory,hasads,isads\
    ,si<fn,useczeros,copied,siflags,nametype,created0x10,created0x30,lastmodified0x10,la\
    stmodified0x30,lastrecordchange0x10,lastrecordchange0x30,lastaccess0x10,lastaccess0x\
    30,updatesequencenumber,logfilesequencenumber,securityid,objectidfiledroid,loggeduti\
    lstream,zoneidcontents"
};
```

---

<sup>165</sup><https://github.com/EricZimmerman/TLEFilePlugins/blob/13055b37b5880c131e1cf6ae4d5ff1a57a537467/TLEFileEZTools/EZTools.cs#L368>

## Ingesting CSVs That Don't Have Plugin Support

If there are no headers that match, then Timeline Explorer will default to the GenericCSV plugin, which can be found [here<sup>166</sup>](#). Effectively, this plugin will treat every value as a string, regardless of whether it's a True/False value, a timestamp, etc.

## Examples of Benefits of Timeline Explorer Plugins

Here we will cover a few key examples of what value plugins provide to Timeline Explorer and supported output.

### Checkboxes

Excel is unable render checkboxes for True/False values when dealing with MFTECmd output.

C
InUse
TRUE
TRUE
TRUE
TRUE

MFTECmd's InUse column in Excel

Timeline Explorer will render the True/False values as a checkbox.

In Use
<input type="checkbox"/>
<input checked="" type="checkbox"/>
<input type="checkbox"/>
<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>

MFTECmd's InUse column in Timeline Explorer

### Timestamp Column Filters, Part 1

Excel is unable to handle the values within MFTECmd output (and other EZ Tools' output) without input from the user.

---

<sup>166</sup><https://github.com/EricZimmerman/TLEFilePlugins/tree/master/TLEFileGenericCsv>

T	U	V	W	X	Y	Z	AA
Created0x	Created0x	LastModif	LastModif	LastRecord	LastRecord	LastAccess	LastAccess
15:49.1		15:49.1		15:49.1		15:49.1	
15:49.1		15:49.1		15:49.1		15:49.1	
15:49.1		15:49.1		15:49.1		15:49.1	
15:49.1		15:49.1		15:49.1		15:49.1	
15:49.1		15:49.1		15:49.1		15:49.1	
31:02.9		10:26.7	16:23.6	10:26.7	16:23.6	10:26.7	16:23.6

MFTECmd's timestamp columns in Excel

When converting the timestamp manually in Excel to the specified format, it works, but why do all of that work? Timeline Explorer displays each timestamp as a DateTime object and allow the examiner to digest the data quicker.

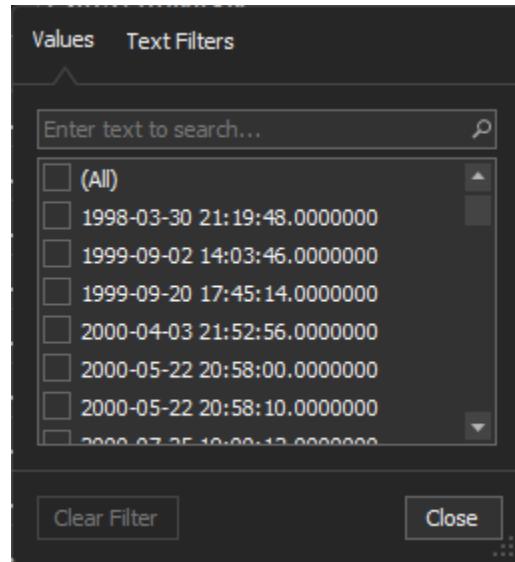
group by that column			
Created0x10	Created0x30	Last Modified0x10	Last Modified0x30
=	=	=	=
1998-03-30 21:19:48.00000...	2014-10-04 13:57:24.77825...	1998-03-30 21:19:48.00000...	2014-10-04 13:57:24.77825...
2019-07-13 13:48:08.28753...	2020-03-24 10:44:19.87244...	1999-03-18 04:00:00.00000...	
1999-09-02 14:03:46.00000...	2014-10-04 13:57:25.01827...	1999-09-02 14:03:46.00000...	2014-10-04 13:57:25.01827...
1999-09-20 17:45:14.00000...	2014-10-04 13:57:20.32025...	1999-09-20 17:45:14.00000...	2014-10-04 13:57:20.32025...
2019-07-13 13:48:08.17814...	2020-03-24 10:44:19.38803...	2000-02-23 17:46:34.00000...	
2000-04-03 21:52:56.00000...	2016-04-25 18:01:08.14343...	2000-04-03 21:52:56.00000...	2016-04-25 18:01:08.14343...
2000-05-22 20:58:00.00000...	2016-04-25 18:01:08.03406...	2000-05-22 20:58:00.00000...	2016-04-25 18:01:08.03406...
2000-05-22 20:58:00.00000...	2016-04-25 18:01:08.04968...	2000-05-22 20:58:00.00000...	2016-04-25 18:01:08.04968...
2000-05-22 20:58:10.00000...	2016-04-25 18:01:08.11218...	2000-05-22 20:58:10.00000...	2016-04-25 18:01:08.11218...
2000-07-25 19:00:12.00000...	2014-10-04 13:57:24.77825...	2000-07-25 19:00:12.00000...	2014-10-04 13:57:24.77825...
2000-07-25 19:09:54.00000...	2014-10-04 13:57:24.77825...	2000-07-25 19:09:54.00000...	2014-10-04 13:57:24.77825...
2000-08-10 11:44:12.00000...	2014-10-04 13:57:24.77825...	2000-08-10 11:44:12.00000...	2014-10-04 13:57:24.77825...
2000-08-10 11:48:32.00000...	2014-10-04 13:57:24.77825...	2000-08-10 11:48:32.00000...	2014-10-04 13:57:24.77825...

MFTECmd's timestamp columns in Timeline Explorer

## Timestamp Column Filters, Part 2

Now that we've established the power of plugins for timestamp values, let's further demonstrate the power of a plugin that's catered towards a specific tool's output compared to the GenericCsv plugin.

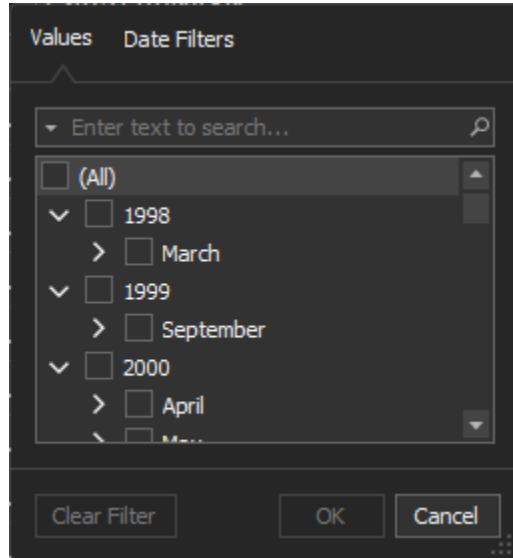
By removing the `TLEFileEZTools.dll` prior to ingesting MFTECmd output, we can see that Timeline Explorer won't know to convert the values within the timestamp-specific columns to `DateTime` objects.



MFTECmd's timestamps displayed as strings

To reproduce the above example, remove the `.\TimelineExplorer\Plugins\TLEFileEZTools.dll` file prior to opening Timeline Explorer and ingest output from the most current version of MFTECmd.

When we leverage the plugin built for EZ Tools output, we can see what value Timeline Explorer provides.



MFTECcmd's timestamps displayed as DateTime objects

The above provides a much more user friendly method of filtering on years, months, and days when conducting analysis.

# Timeline Explorer References

## Blog Posts

### Official Blog Posts

Blog posts from Eric Zimmerman's blog, Binary Foray:

- Introducing Timeline Explorer v0.4.0.0<sup>167</sup>
- Timeline Explorer 0.5.0.0 released<sup>168</sup>
- Timeline Explorer 0.6.0 released!<sup>169</sup>
- Updates to the left of me, updates to the right of me, version 1 releases are here (for the most part)<sup>170</sup>
- A fluery of updates!<sup>171</sup>
- Everything gets an update, Sept 2018 edition<sup>172</sup>

### Community Resources

- AboutDFIR - Timeline Explorer<sup>173</sup>
- Episode 87: Introducing and Using Timeline Explorer<sup>174</sup>

## Download Timeline Explorer

Timeline Explorer can be downloaded from <https://ericzimmerman.github.io/#!index.md>

---

<sup>167</sup><https://binaryforay.blogspot.com/2017/04/introducing-timeline-explorer-v0400.html>

<sup>168</sup><https://binaryforay.blogspot.com/2017/05/timeline-explorer-0500-released.html>

<sup>169</sup><https://binaryforay.blogspot.com/2017/10/timeline-explorer-060-released.html>

<sup>170</sup><https://binaryforay.blogspot.com/2018/03/updates-to-left-of-me-updates-to-right.html>

<sup>171</sup><https://binaryforay.blogspot.com/2018/05/a-fluery-of-updates.html>

<sup>172</sup><https://binaryforay.blogspot.com/2018/09/everything-gets-update-sept-2018-edition.html>

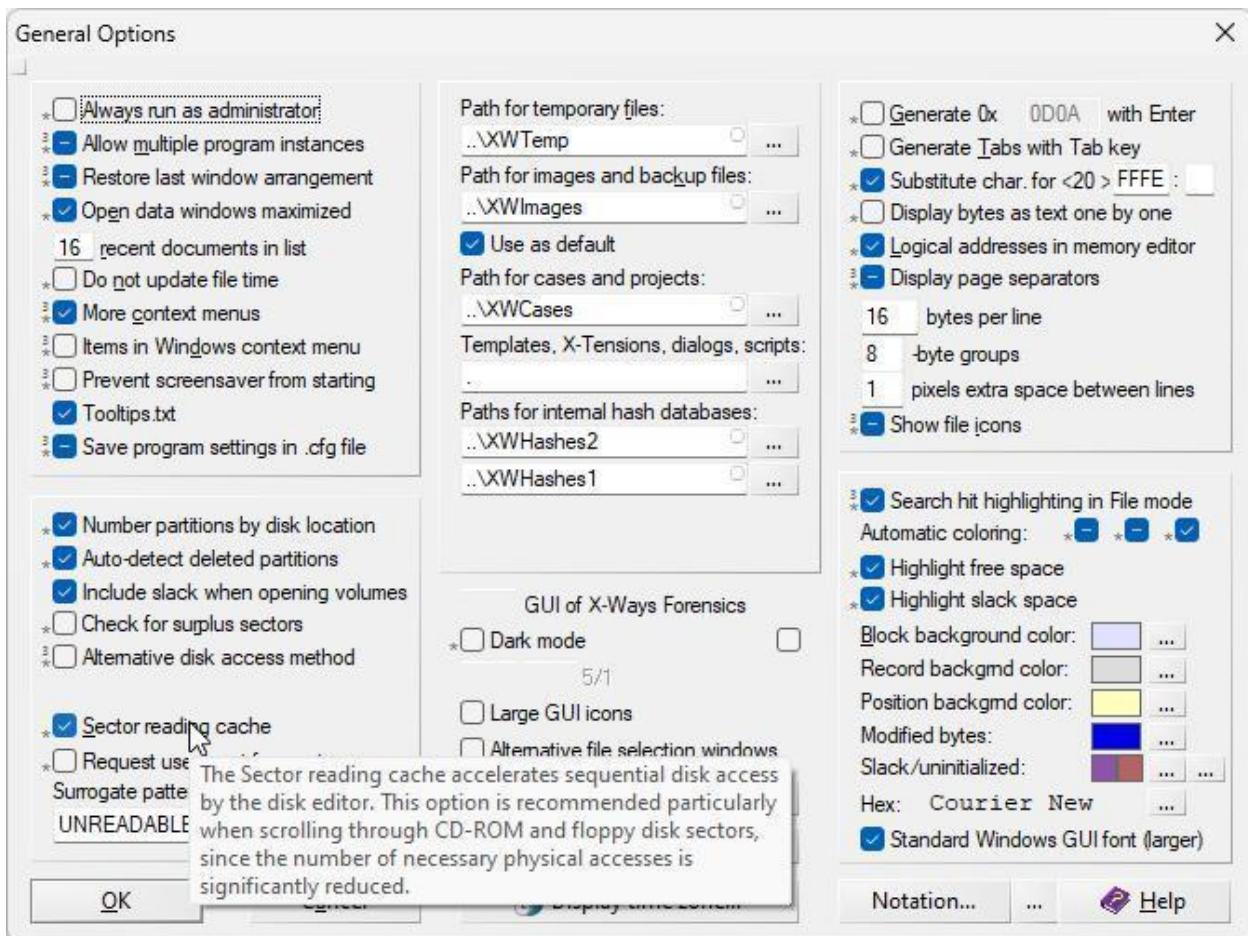
<sup>173</sup><https://aboutdfir.com/toolsandartifacts/windows/timeline-explorer>

<sup>174</sup><https://www.youtube.com/watch?v=Hy8ZIc86tCo>

# XWFIM

XWFIM is a tool created by Eric Zimmerman that assists with installing and managing a local instance of X-Ways<sup>175</sup>. XWFIM allows examiners to download everything necessary to begin using X-Ways Forensics (including BYOD) or X-Ways Investigator in a few clicks.

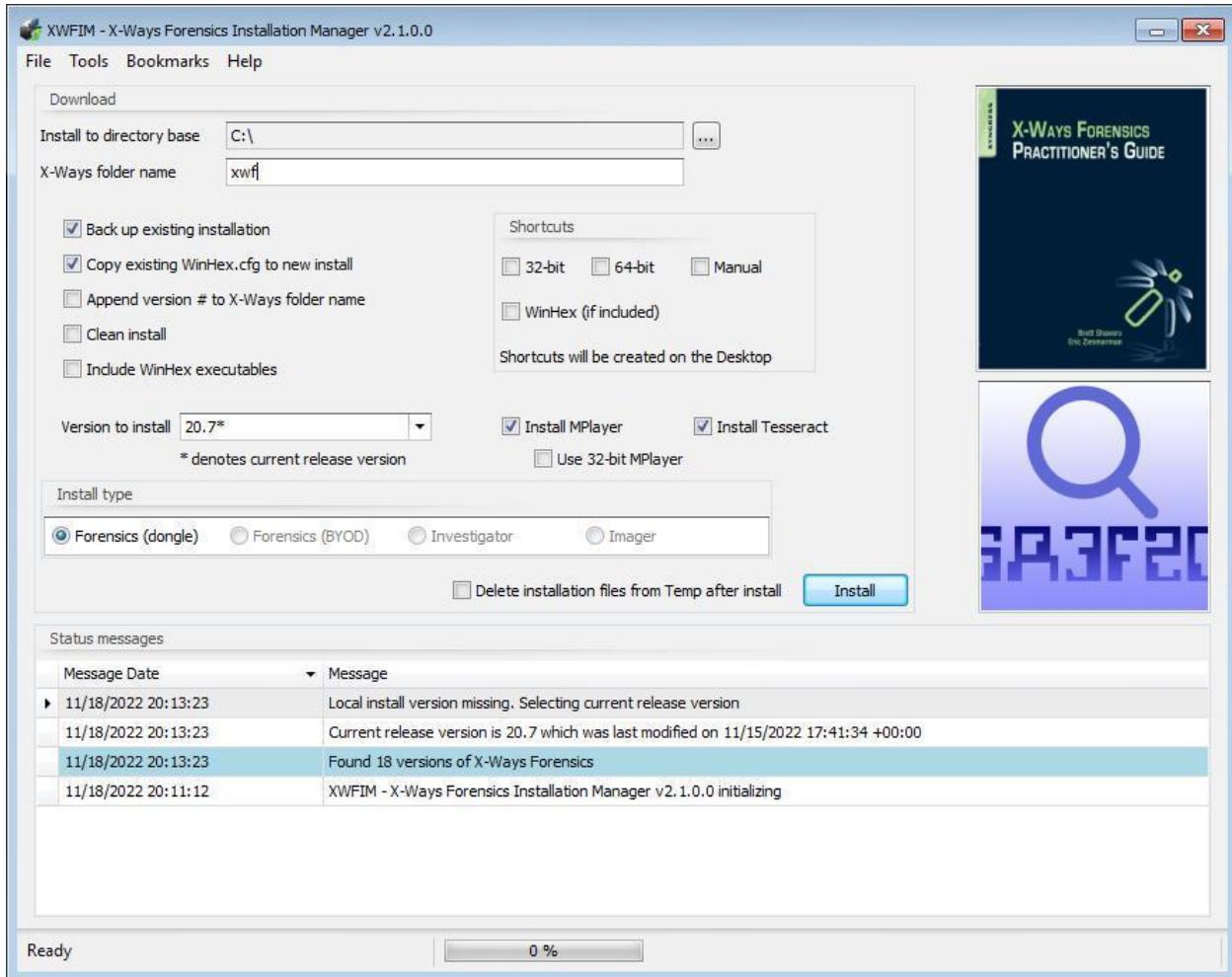
XWFIM also provides tooltips for X-Ways to leverage. These can be found by hovering the cursor over each option within X-Ways.



<sup>175</sup><https://www.x-ways.net/>

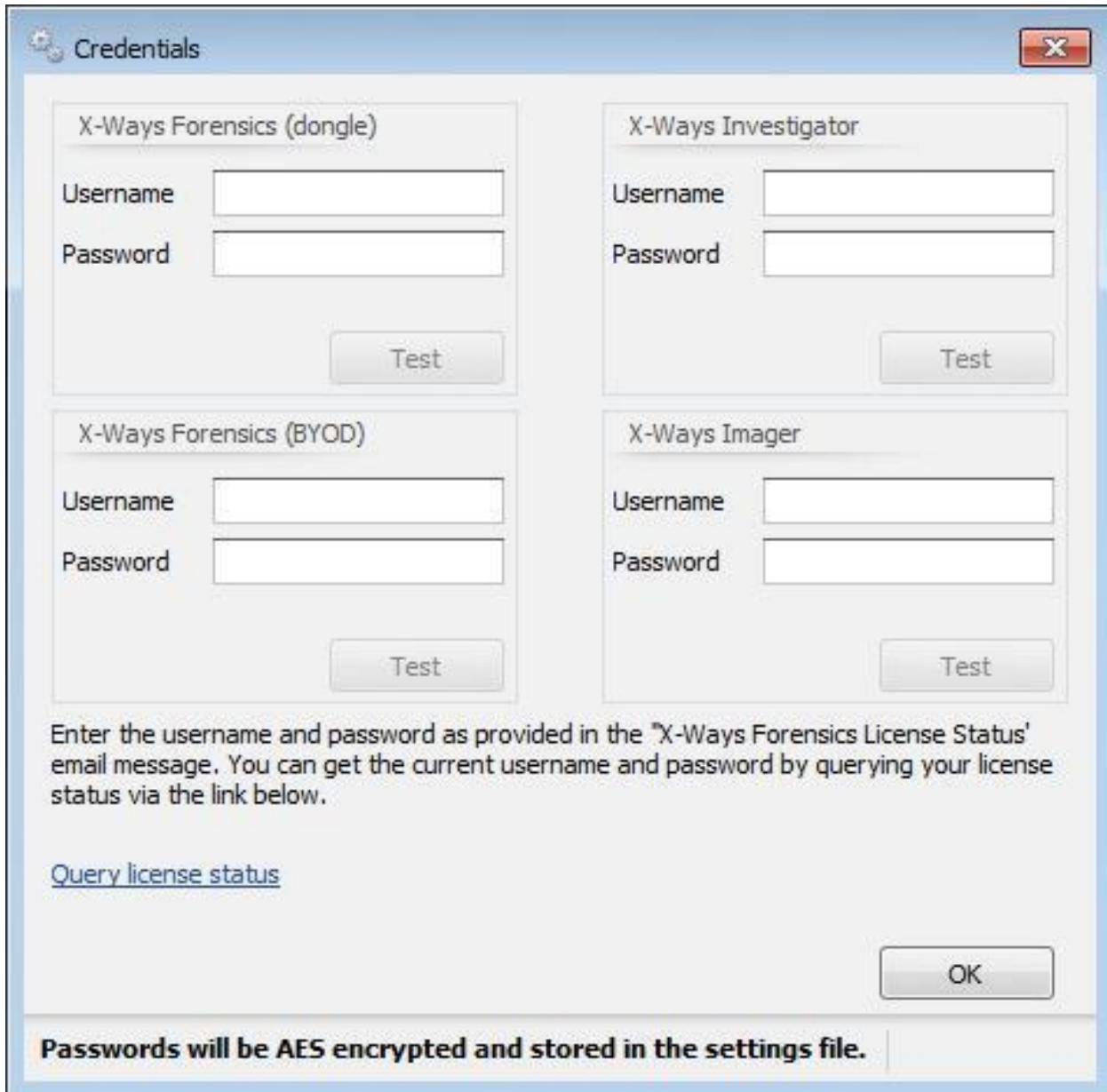
## Using XWFIM

XWFIM provides many features to assist with installing, managing, and verifying a local installation of X-Ways.



## X-Ways Credentials

Prior to using XWFIM, one must have valid credentials to download X-Ways using XWFIM.



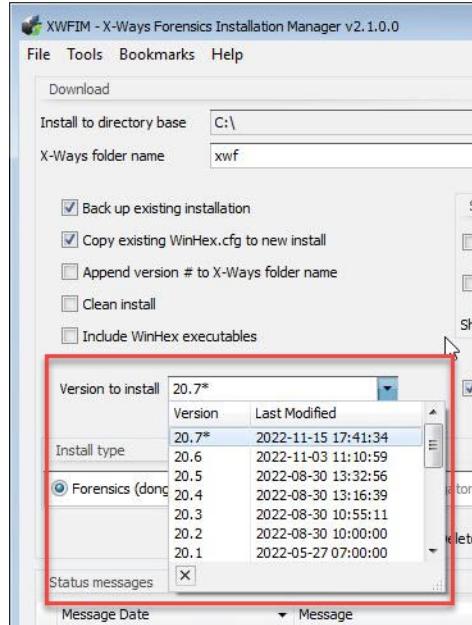
Once valid credentials are entered, XWFIM will provide a prompt similar to below.



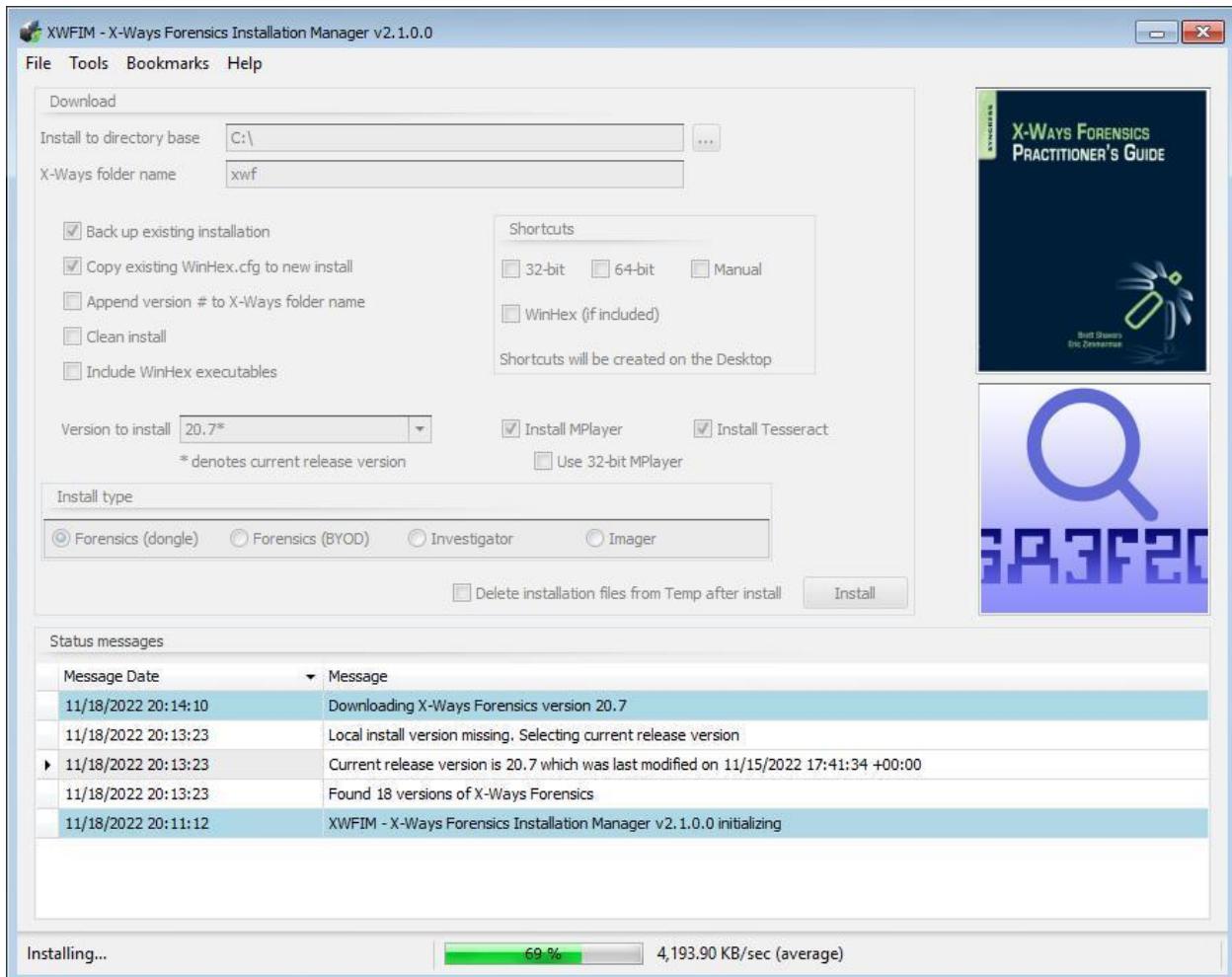
## Typical Installation

Installing X-Ways Forensics can be done following the steps below.

First, select the desired version using the dropdown, as seen below.



Once a version is selected, ensure the install directory and X-Ways folder name are populated and hit Install.



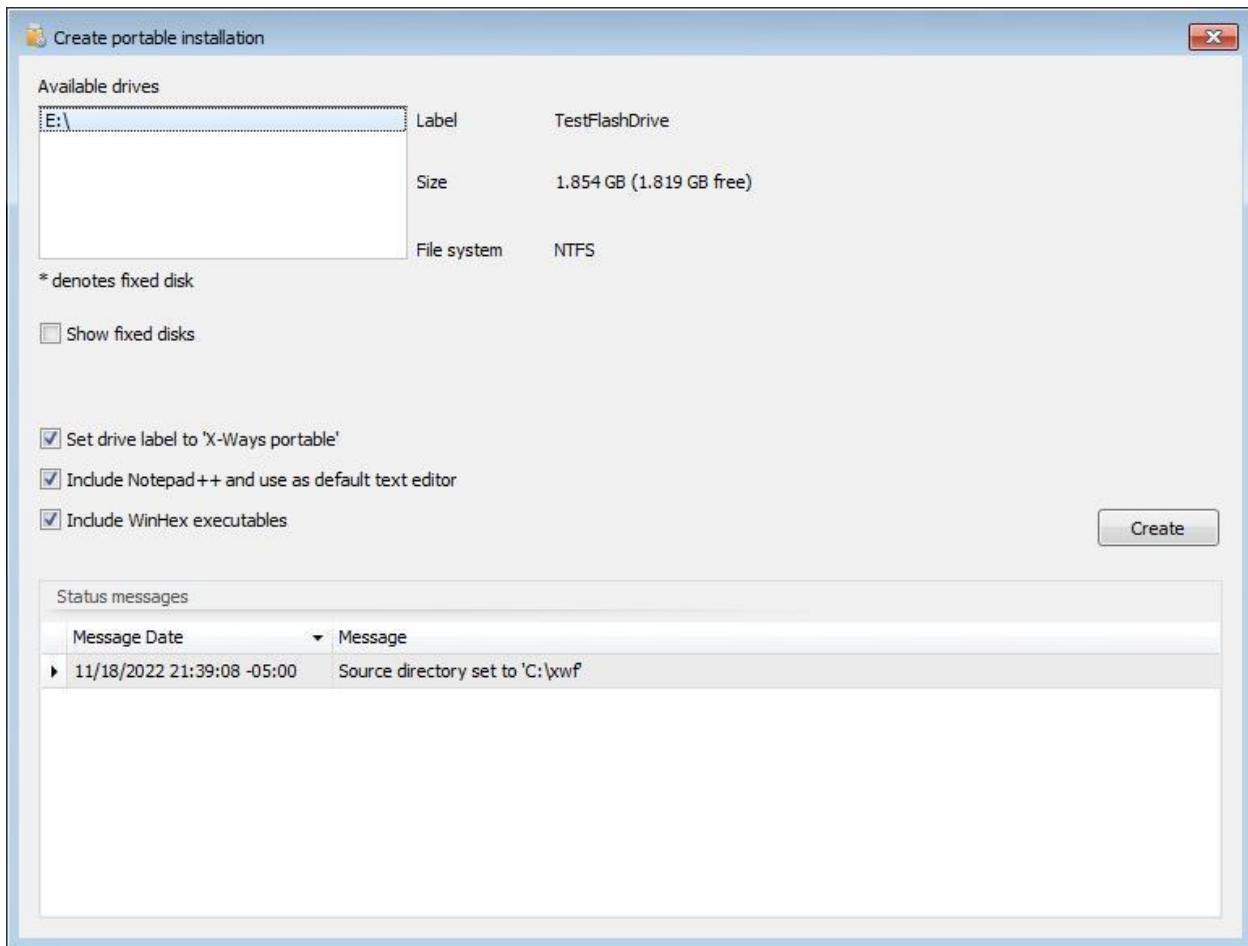
Once the installation is complete, XWFIM will display a prompt similar to below.



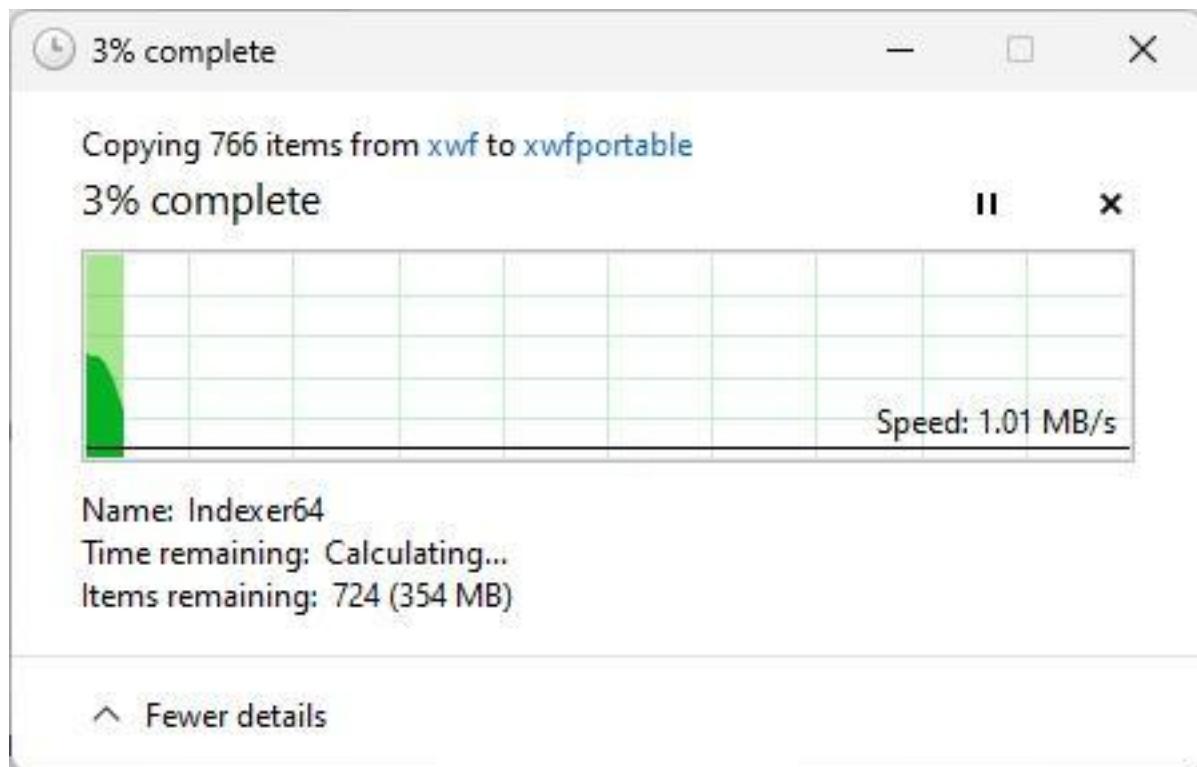
## Portable Installation

To create a portable installation of X-Ways using XWFIM, select the Tools -> Create portable installation feature. Please note, if you try to do this without having executed X-Ways successfully on the host system, a similar error message will likely display: WinHex.cfg is missing from 'C:\xwf'. Open X-Ways at least once to create it and try again. Aborting...

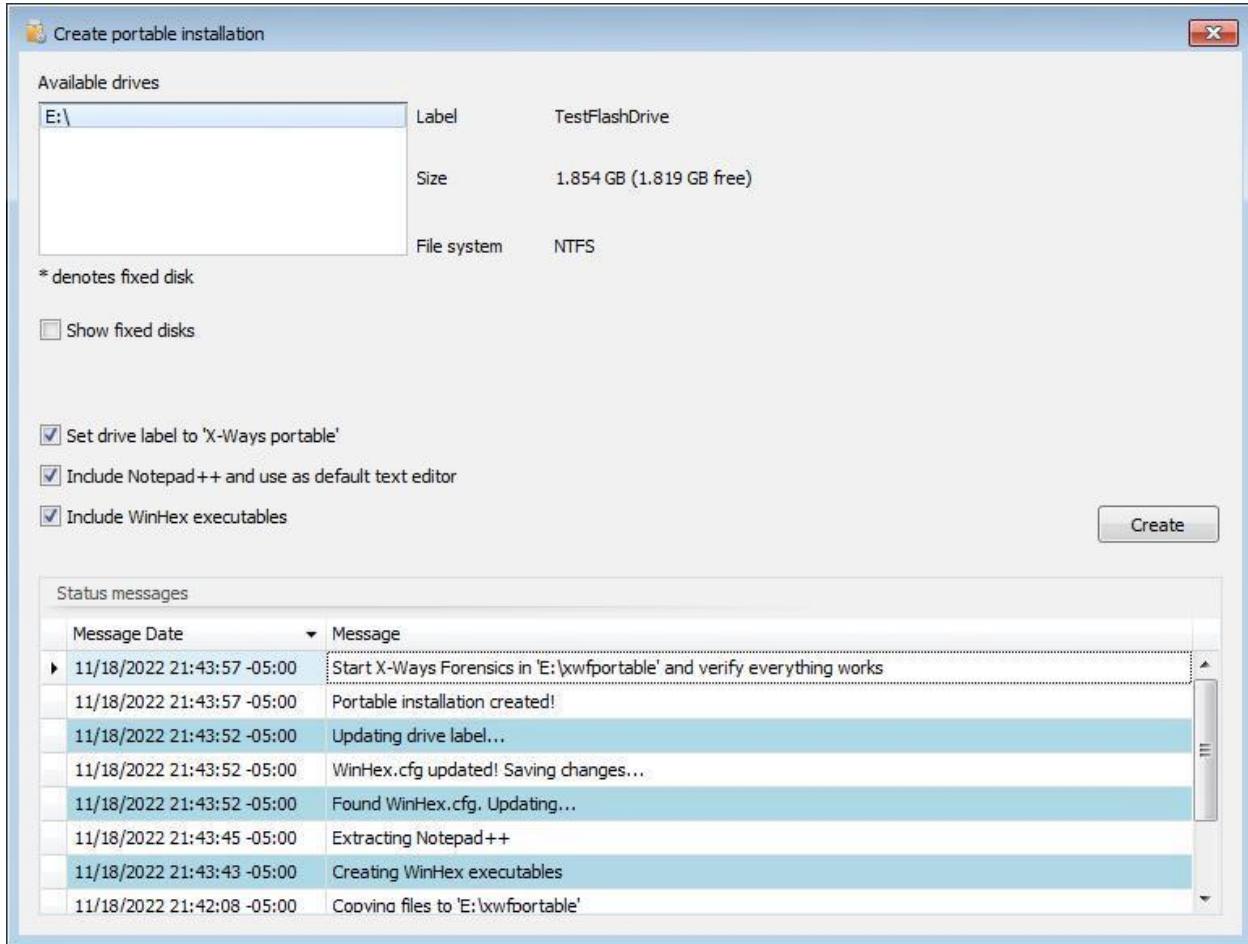
With a USB flash drive plugged in, one will see all attached USB flash drives available to create the portable installation on.



Selecting the E:\ as the destination, a progress window will emerge once the Create button is pushed.



When complete, a similar window will display.



On the destination USB flash drive, one can see similar files to what can be found in C:\xwf from the previous example.

The screenshot shows a file explorer window with the following details:

- Path: E: > xwfportable >
- File List:

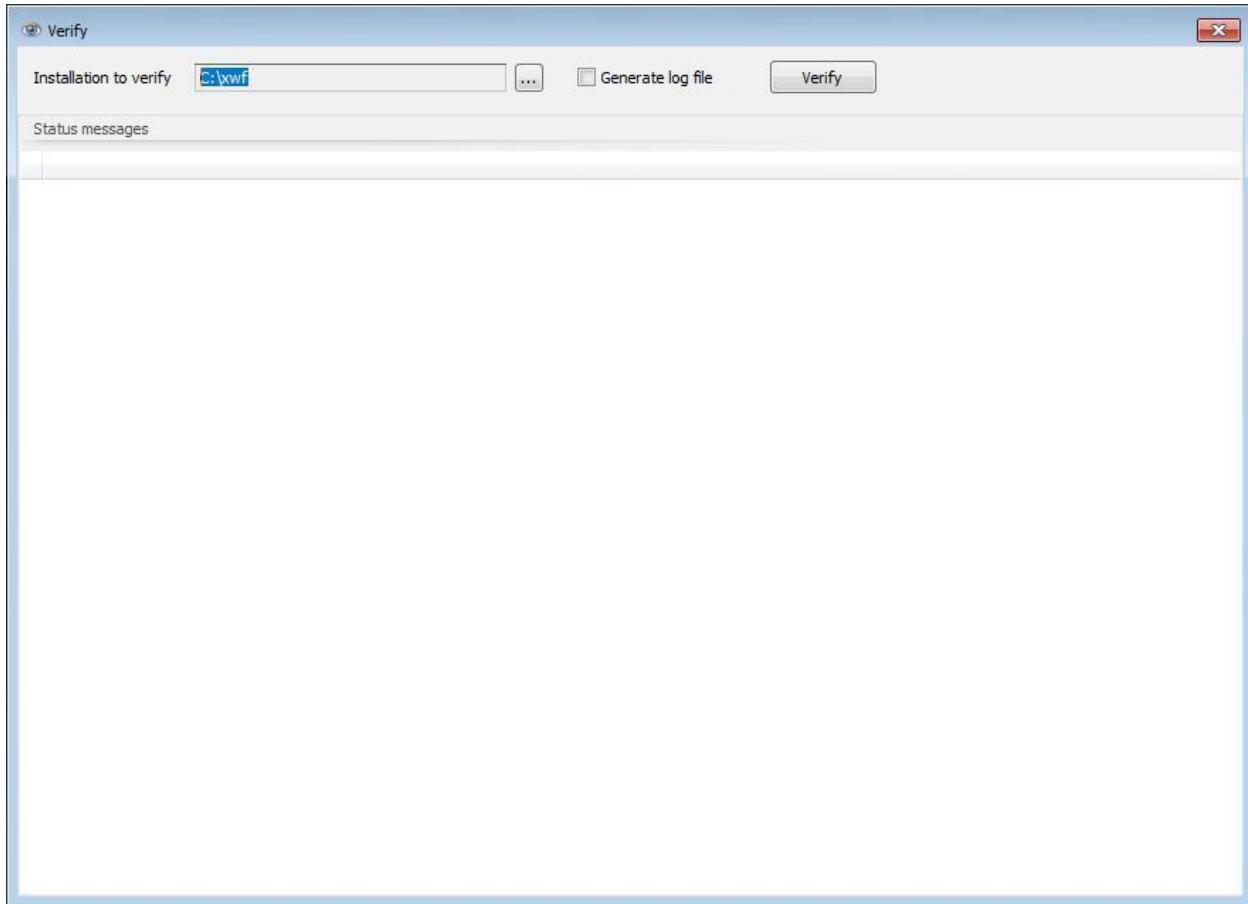
Name	Size	Type	Modified	Attr
MPlayer	32.9 MB	File Folder	Today 21:42	-----i
Tesseract	196 MB	File Folder	Today 21:42	-----i
viewer	8.47 MB	File Folder	Today 21:43	-----i
!readme.txt	2.91 KB	Text document	2022-08-16 23:59	-a-----
avcodec-57.dll	2.20 MB	mkape_auto_file	Tuesday 20:07	-a-----
avdevice-57.dll	453 KB	mkape_auto_file	Tuesday 20:07	-a-----
avfilter-6.dll	557 KB	mkape_auto_file	Tuesday 20:07	-a-----
avformat-57.dll	796 KB	mkape_auto_file	Tuesday 20:07	-a-----
avutil-55.dll	1.10 MB	mkape_auto_file	Tuesday 20:07	-a-----
Boot Sector FAT.tpl	1.18 KB	TPL File	Tuesday 20:07	-a-----
Boot Sector FAT32.tpl	1.40 KB	TPL File	Tuesday 20:07	-a-----
Boot Sector NTFS.tpl	1.59 KB	TPL File	Tuesday 20:07	-a-----
Case Report.css	6.38 KB	Cascading Style Sheet Document	Tuesday 20:07	-a-----
Case Report Classic.css	261 bytes	Cascading Style Sheet Document	Tuesday 20:07	-a-----

- Bottom Status Bar:

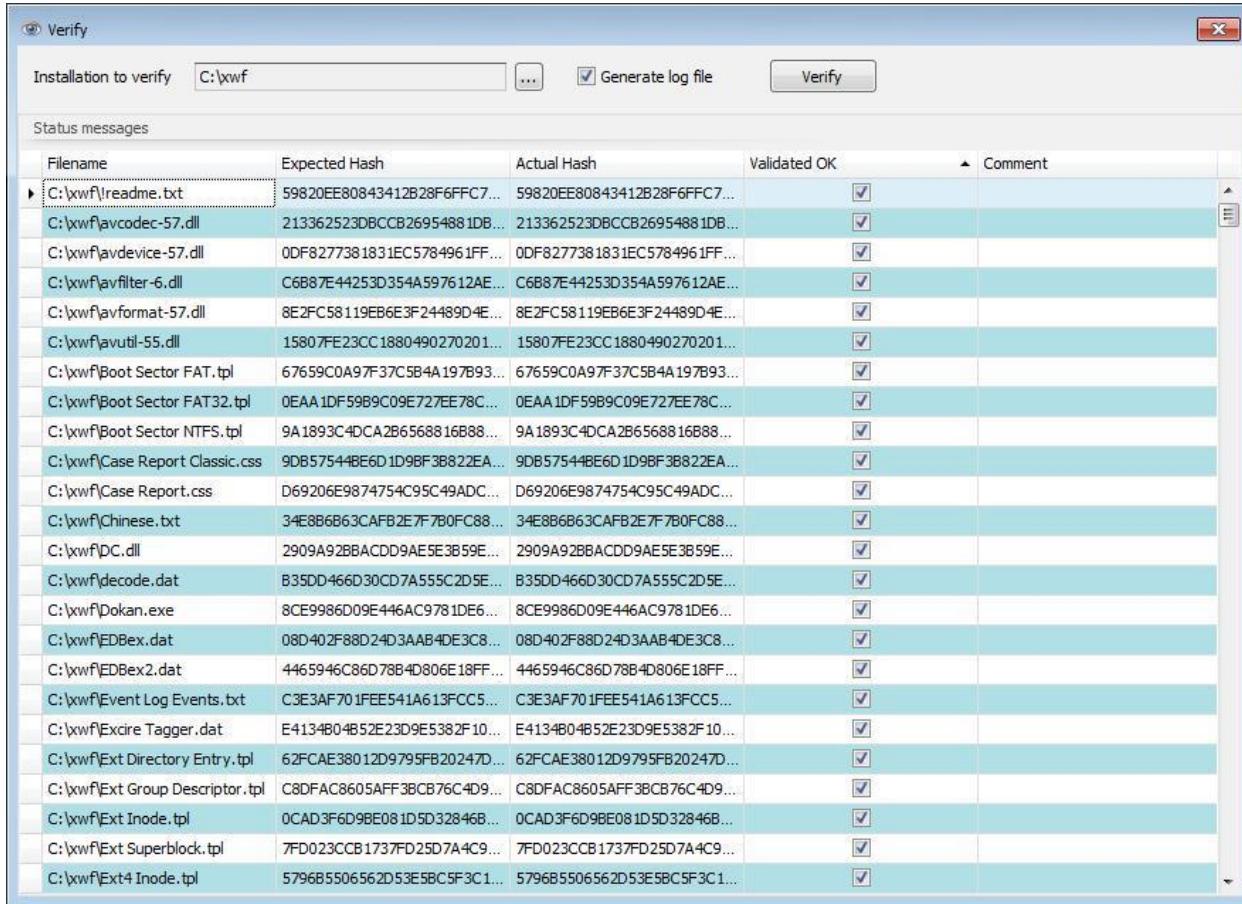
  - 91 folders, 3 files, 0 hidden, 0 bytes / 270 MB
  - Enter filter dropdown
  - Show everything button

## Verifying Installation

XWFIM can verify a local installation of X-Ways using the Tools -> Verify installation feature. First, specify which directory for XWFIM to verify the installation of. In this example, we'll use the C:\xwf directory as shown in previous screenshots.



Once a directory is specified, choose whether or not a log file should be generated and hit Verify.



The screenshot shows a software interface titled "Verify". At the top, there is a field labeled "Installation to verify" containing "C:\xwf", a "Verify" button, and a checkbox labeled "Generate log file" which is checked. Below this is a table titled "Status messages" with columns: "Filename", "Expected Hash", "Actual Hash", "Validated OK", and "Comment". The table lists numerous files from the "C:\xwf\" directory, such as "readme.txt", "avcodec-57.dll", "avdevice-57.dll", etc., comparing their expected and actual hash values. Most files show a green checkmark in the "Validated OK" column, indicating they passed the verification process.

Filename	Expected Hash	Actual Hash	Validated OK	Comment
C:\xwf\readme.txt	59820EE80843412B28F6FFC7...	59820EE80843412B28F6FFC7...	<input checked="" type="checkbox"/>	
C:\xwf\avcodec-57.dll	213362523DBCCB26954881DB...	213362523DBCCB26954881DB...	<input checked="" type="checkbox"/>	
C:\xwf\avdevice-57.dll	0DF8277381831EC5784961FF...	0DF8277381831EC5784961FF...	<input checked="" type="checkbox"/>	
C:\xwf\avfilter-6.dll	C6B87E44253D354A597612AE...	C6B87E44253D354A597612AE...	<input checked="" type="checkbox"/>	
C:\xwf\avformat-57.dll	8E2FC58119EB6E3F24489D4E...	8E2FC58119EB6E3F24489D4E...	<input checked="" type="checkbox"/>	
C:\xwf\avutil-55.dll	15807FE23CC1880490270201...	15807FE23CC1880490270201...	<input checked="" type="checkbox"/>	
C:\xwf\Boot Sector FAT.tpl	67659C0A97F37C5B4A197B93...	67659C0A97F37C5B4A197B93...	<input checked="" type="checkbox"/>	
C:\xwf\Boot Sector FAT32.tpl	0EAA1DF59B9C09E727EE78C...	0EAA1DF59B9C09E727EE78C...	<input checked="" type="checkbox"/>	
C:\xwf\Boot Sector NTFS.tpl	9A1893C4DCA2B6568816B88...	9A1893C4DCA2B6568816B88...	<input checked="" type="checkbox"/>	
C:\xwf\Case Report Classic.css	9DB57544BE6D1D9BF3B822EA...	9DB57544BE6D1D9BF3B822EA...	<input checked="" type="checkbox"/>	
C:\xwf\Case Report.css	D69206E9874754C95C49ADC...	D69206E9874754C95C49ADC...	<input checked="" type="checkbox"/>	
C:\xwf\Chinese.txt	34E8B6B63CAF82E7F7B0FC88...	34E8B6B63CAF82E7F7B0FC88...	<input checked="" type="checkbox"/>	
C:\xwf\DC.dll	2909A92BBACDD9AE5E3B59E...	2909A92BBACDD9AE5E3B59E...	<input checked="" type="checkbox"/>	
C:\xwf\decode.dat	B35DD466D30CD7A555C2D5E...	B35DD466D30CD7A555C2D5E...	<input checked="" type="checkbox"/>	
C:\xwf\Dokan.exe	8CE9986D09E446AC9781DE6...	8CE9986D09E446AC9781DE6...	<input checked="" type="checkbox"/>	
C:\xwf\EDBex.dat	08D402F88D24D3AAB4DE3C8...	08D402F88D24D3AAB4DE3C8...	<input checked="" type="checkbox"/>	
C:\xwf\EDBex2.dat	4465946C86D78B4D806E18FF...	4465946C86D78B4D806E18FF...	<input checked="" type="checkbox"/>	
C:\xwf\Event Log Events.txt	C3E3AF701FEE541A613FCC5...	C3E3AF701FEE541A613FCC5...	<input checked="" type="checkbox"/>	
C:\xwf\Exdire Tagger.dat	E4134B04B52E23D9E5382F10...	E4134B04B52E23D9E5382F10...	<input checked="" type="checkbox"/>	
C:\xwf\Ext Directory Entry.tpl	62FCAE38012D9795FB20247D...	62FCAE38012D9795FB20247D...	<input checked="" type="checkbox"/>	
C:\xwf\Ext Group Descriptor.tpl	C8DFAC8605AFF3BCB76C4D9...	C8DFAC8605AFF3BCB76C4D9...	<input checked="" type="checkbox"/>	
C:\xwf\Ext Inode.tpl	0CAD3F6D9BE081D5D32846B...	0CAD3F6D9BE081D5D32846B...	<input checked="" type="checkbox"/>	
C:\xwf\Ext Superblock.tpl	7FD023CCB1737FD25D7A4C9...	7FD023CCB1737FD25D7A4C9...	<input checked="" type="checkbox"/>	
C:\xwf\Ext4 Inode.tpl	5796B5506562D53E5BC5F3C1...	5796B5506562D53E5BC5F3C1...	<input checked="" type="checkbox"/>	

Generating a log file will output the same data in the above screenshot to an .XLSX file.

## XWFIM Status Messages

Below is an example of status messages that XWFIM displays during the installation process.

Message Date	Message
11/18/2022 20:14:44	Installation complete. Be sure to configure/verify the appropriate settings in Options   General and Options   Viewer Programs
11/18/2022 20:14:42	Generating validation data...
11/18/2022 20:14:42	Copying manual to 'C:\xwf'
11/18/2022 20:14:39	Unzipping Tesseract to 'C:\xwf'
11/18/2022 20:14:38	Unzipping MPlayer to 'C:\xwf'
11/18/2022 20:14:36	Unzipping viewer to 'C:\xwf'
11/18/2022 20:14:35	Unzipping X-Ways Forensics to 'C:\xwf'
11/18/2022 20:14:35	'WinHex.cfg' not found in 'C:\xwf'. Skipping
11/18/2022 20:14:34	Downloading manual
11/18/2022 20:14:23	Downloading Tesseract
11/18/2022 20:14:20	Downloading MPlayer
11/18/2022 20:14:14	Downloading viewer
11/18/2022 20:14:10	Downloading X-Ways Forensics version 20.7
11/18/2022 20:13:23	Local install version missing. Selecting current release version
11/18/2022 20:13:23 /2022 17:41:34 +00:00	Current release version is 20.7 which was last modified on 11/15\
11/18/2022 20:13:23	Found 18 versions of X-Ways Forensics
11/18/2022 20:11:12	XWFIM - X-Ways Forensics Installation Manager v2.1.0.0 initializing

## XWFIM References

### Download XWFIM

XWFIM can be downloaded from <https://ericzimmerman.github.io/#!index.md>

# **Errata**

## **Reporting Errata**

If you think you've found an error relating to spelling, grammar, or anything else that's currently holding this book back from being the best it can be, please visit the book's [GitHub repository<sup>176</sup>](#) and create an Issue detailing the error you've found. Anyone is also welcome to submit a Pull Request with new content, fixes, changes, etc.

---

<sup>176</sup><https://github.com/EZToolsManuals/EZToolsManuals/issues>