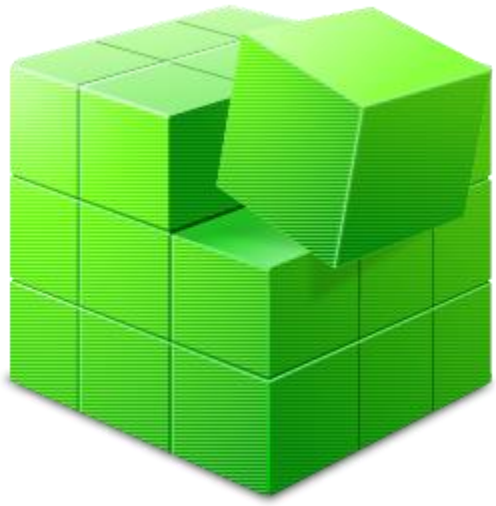


Registry Explorer

RECmd

Eric R. Zimmerman
saericzimmerman@gmail.com
501-313-3778

Revision history

07/01/2015 Rev. 1 – Initial release

06/08/2016 Rev. 2 – Updated for v0.8.1.0

05/19/2017 Rev. 3 – Updated for v0.9.0.0

Table of Contents

Requirements

Registry Explorer and RECmd require Microsoft .net framework version 4.6 full runtime or greater to be installed. It is available at https://www.microsoft.com/en-us/download/details.aspx?id=49982.

Why another Registry tool?

The need for Registry Explorer and RECmd rose 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 of 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.

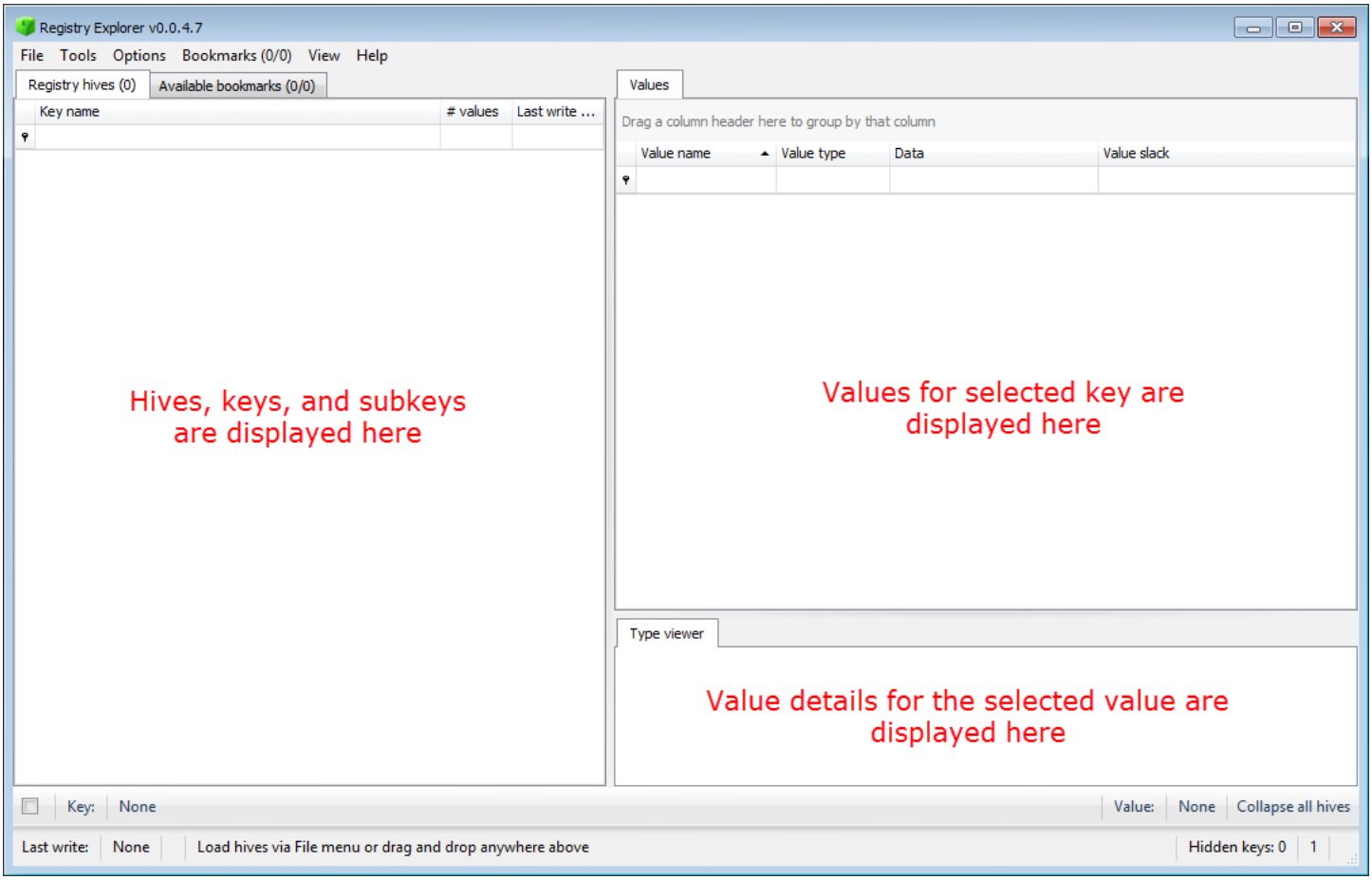
In summary, the capabilities of Registry Explorer and RECmd allows for quickly examining multiple hives at once and they can be leveraged to find new places where currently understood data is located in an easy to use and systematic way. It can be used in educational settings to not only understand the Registry from a functional level, but also from a deeply technical perspective.

Registry Explorer

Registry Explorer is a GUI based tool used to view the contents of offline Registry Hives. It has the ability to load multiple hives at once, search across all loaded hives using strings or regular expressions, exporting of data, and much more.

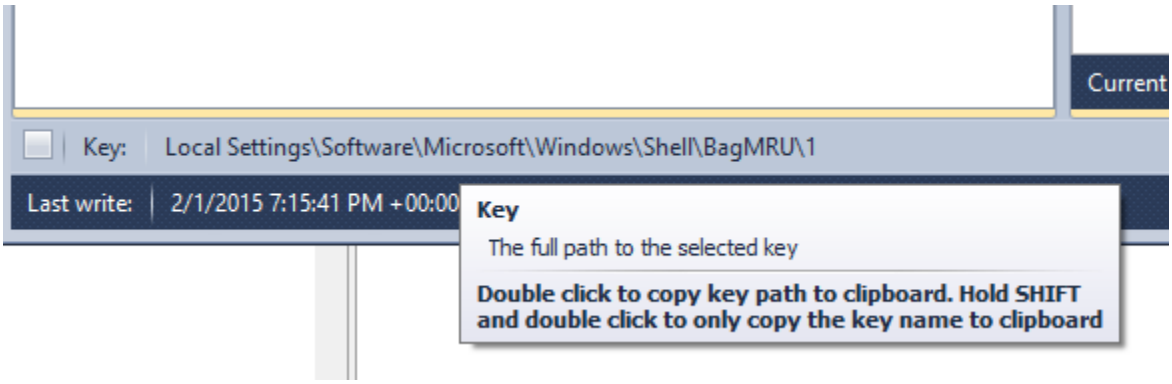
Getting started

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 .layout 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 of 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.

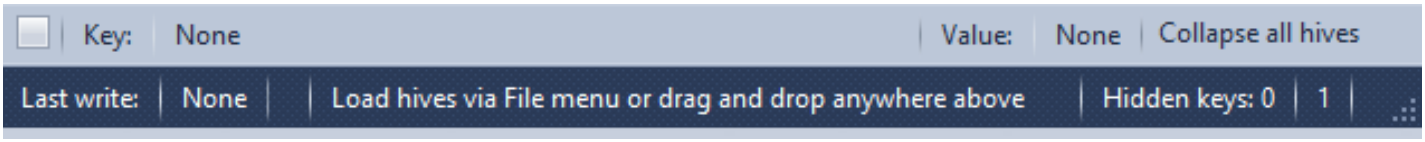
Value details

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.

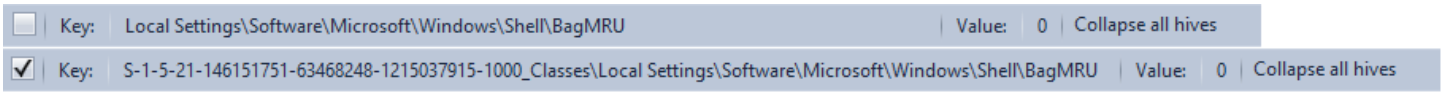
Status bars

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.

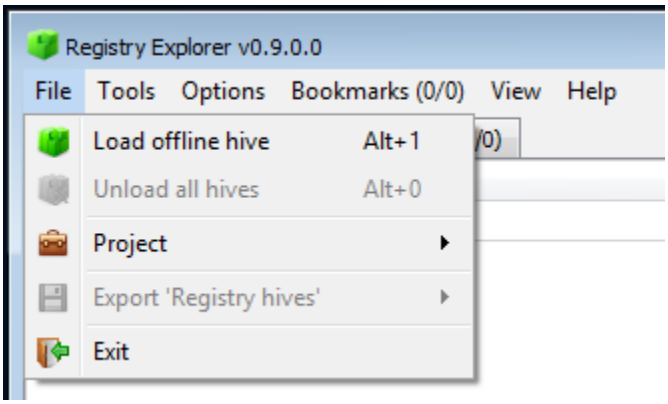
Main menu

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.

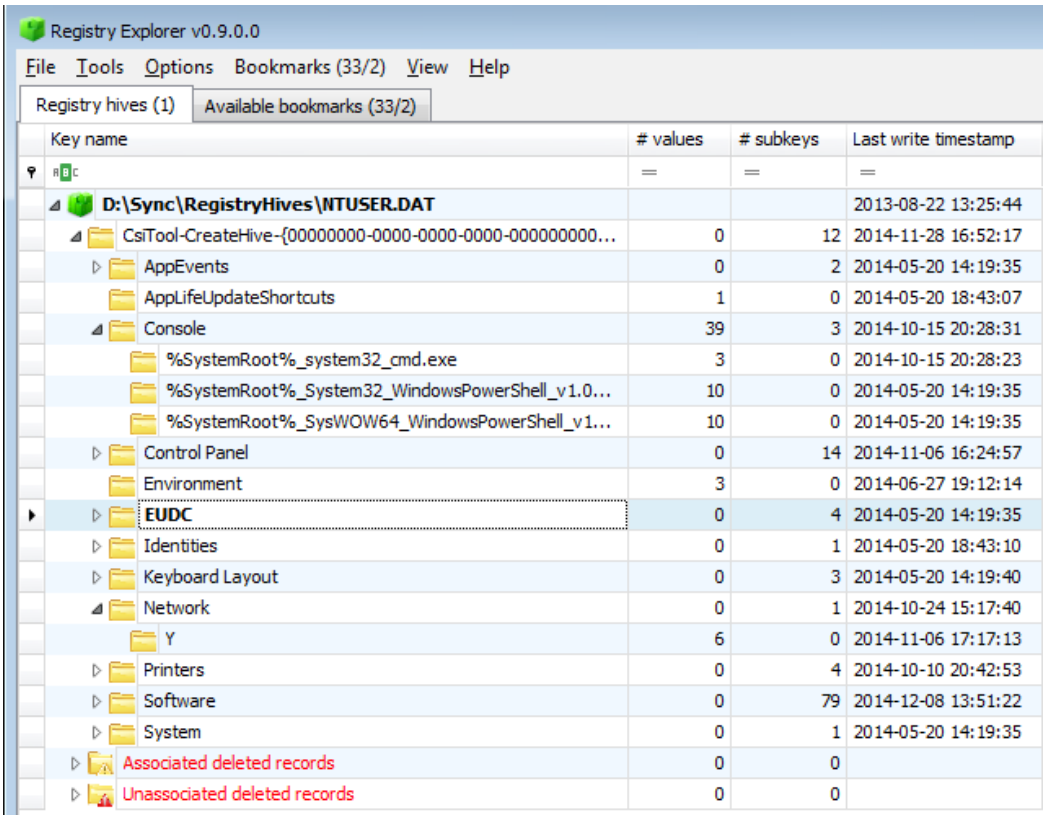
The various sections below will explain these submenus. Where things are obvious (like File | Exit), no additional information will be provided.

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:



Key name	# values	# subkeys	Last write timestamp
D:\Sync\RegistryHives\NTUSER.DAT			2013-08-22 13:25:44
CsiTool-CreateHive-{00000000-0000-0000-0000-000000000...	0	12	2014-11-28 16:52:17
AppEvents	0	2	2014-05-20 14:19:35
AppLifeUpdateShortcuts	1	0	2014-05-20 18:43:07
Console	39	3	2014-10-15 20:28:31
%SystemRoot%_system32_cmd.exe	3	0	2014-10-15 20:28:23
%SystemRoot%_System32_WindowsPowerShell_v1.0...	10	0	2014-05-20 14:19:35
%SystemRoot%_SysWOW64_WindowsPowerShell_v1...	10	0	2014-05-20 14:19:35
Control Panel	0	14	2014-11-06 16:24:57
Environment	3	0	2014-06-27 19:12:14
EUDC	0	4	2014-05-20 14:19:35
Identities	0	1	2014-05-20 18:43:10
Keyboard Layout	0	3	2014-05-20 14:19:40
Network	0	1	2014-10-24 15:17:40
Y	6	0	2014-11-06 17:17:13
Printers	0	4	2014-10-10 20:42:53
Software	0	79	2014-12-08 13:51:22
System	0	1	2014-05-20 14:19:35
Associated deleted records	0	0	
Unassociated deleted records	0	0	

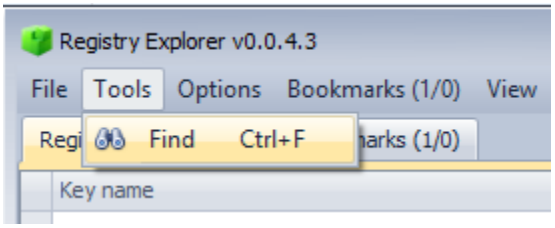
Exporting to PDF would generate a PDF file that contains the following:

Key name	# values	# subkeys	Last write timestamp
D:\Sync\RegistryHives\NTUSER.DAT			2013-08-22 13:25:44
CsiTool-CreateHive-{00000000-0000-0000-0000-000000000000}	0	12	2014-11-28 16:52:17
AppEvents	0	2	2014-05-20 14:19:35
AppLifeUpdateShortcuts	1	0	2014-05-20 18:43:07
Console	39	3	2014-10-15 20:28:31
%SystemRoot%_system32_cmd.exe	3	0	2014-10-15 20:28:23
%SystemRoot%_System32_WindowsPowerShell_v1.0_power	10	0	2014-05-20 14:19:35
%SystemRoot%_SysWOW64_WindowsPowerShell_v1.0_pow	10	0	2014-05-20 14:19:35
Control Panel	0	14	2014-11-06 16:24:57
Environment	3	0	2014-06-27 19:12:14
EUDC	0	4	2014-05-20 14:19:35
Identities	0	1	2014-05-20 18:43:10
Keyboard Layout	0	3	2014-05-20 14:19:40
Network	0	1	2014-10-24 15:17:40
Y	6	0	2014-11-06 17:17:13
Printers	0	4	2014-10-10 20:42:53
Software	0	79	2014-12-08 13:51:22
System	0	1	2014-05-20 14:19:35
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

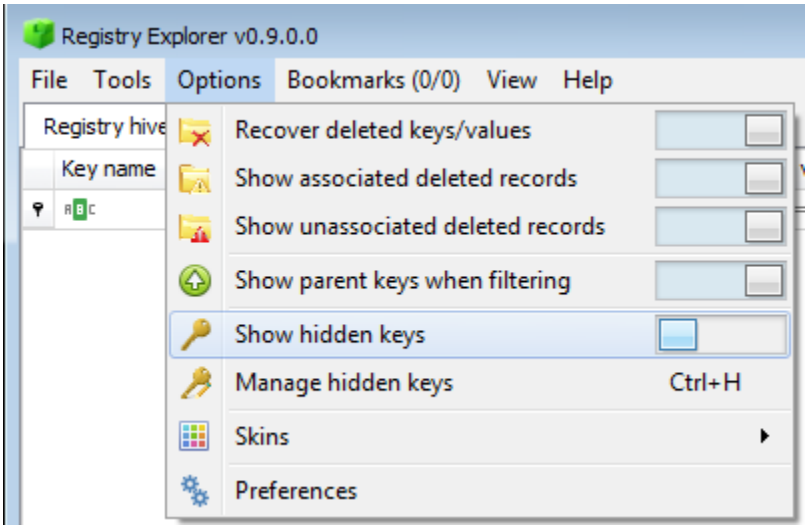
The Tools menu contains a single item, Find.



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**: Similar to 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.

Registry hives | Available bookmarks (1/1)

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...
0	2	2/1/2015 7:14:41 PM +0...
0	2	2/1/2015 7:14:41 PM +0...
0	2	2/1/2015 7:15:41 PM +0...
0	3	2/1/2015 7:15:41 PM +0...
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.

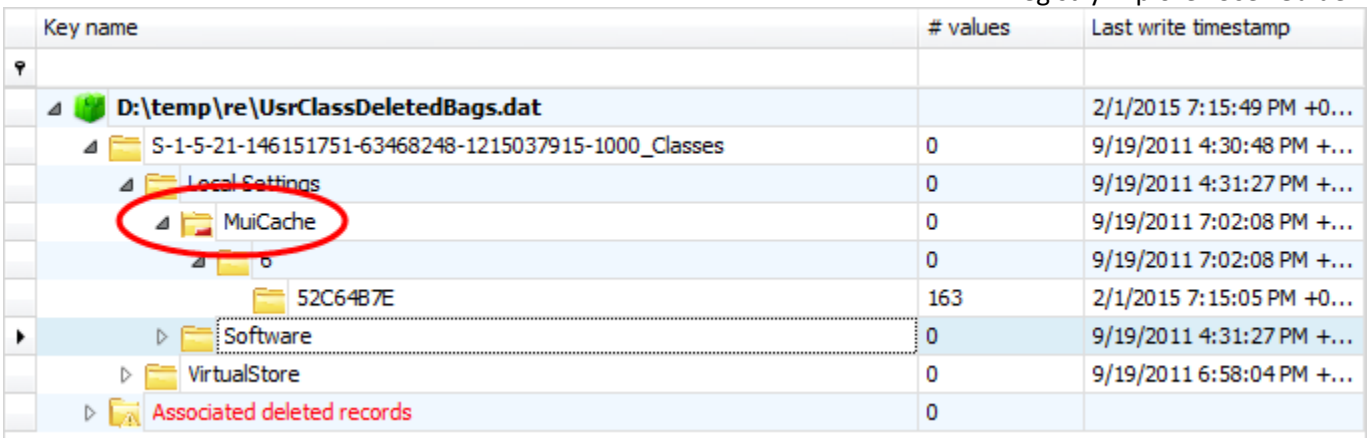
Key name	# values	Last write timestamp
D:\temp\re\UsrClassDeletedBags.dat		2/1/2015 7:15:49 PM +0...
S-1-5-21-146151751-63468248-1215037915-1000_Classes	0	9/19/2011 4:30:48 PM +...
Local Settings	0	9/19/2011 4:31:27 PM +...
MuiCache	0	9/19/2011 7:02:08 PM +...
6	0	9/19/2011 7:02:08 PM +...
52C64B7E	163	2/1/2015 7:15:05 PM +0...
Software	0	9/19/2011 4:31:27 PM +...
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 from view, 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	

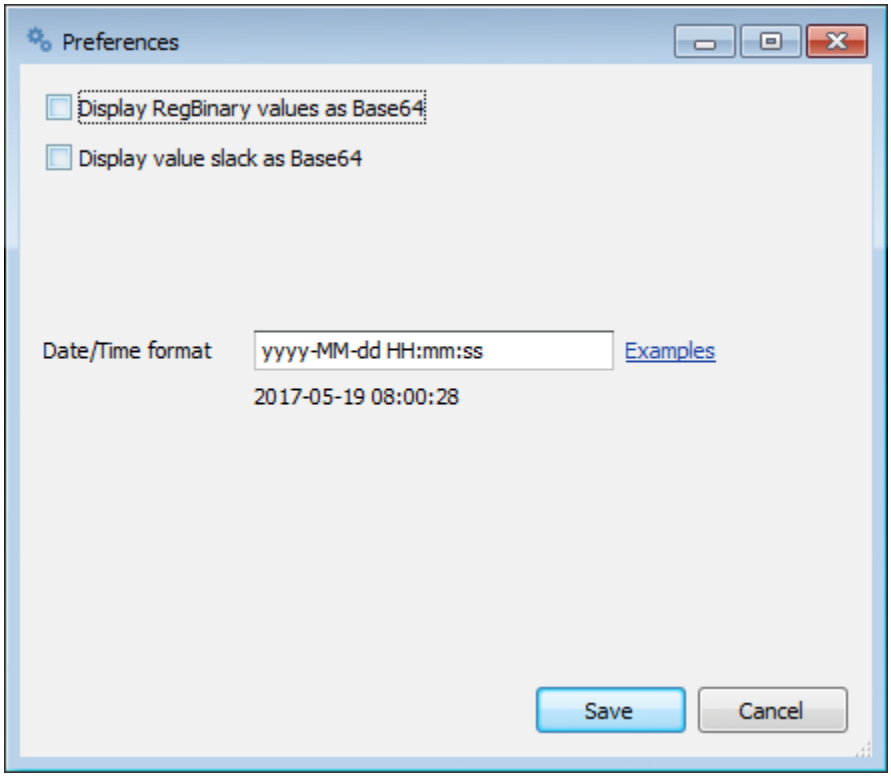
Notice the MuiCache key is no longer visible (assuming the Show hidden key option is off). If we enable this option, the MuiCache key will be shown in its original place, but the icon is different to show that it is in fact a hidden key.



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.



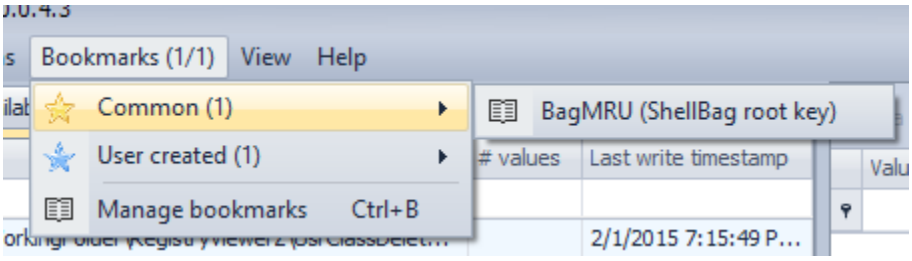
Bookmarks

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, that you can push your Bookmarks to.



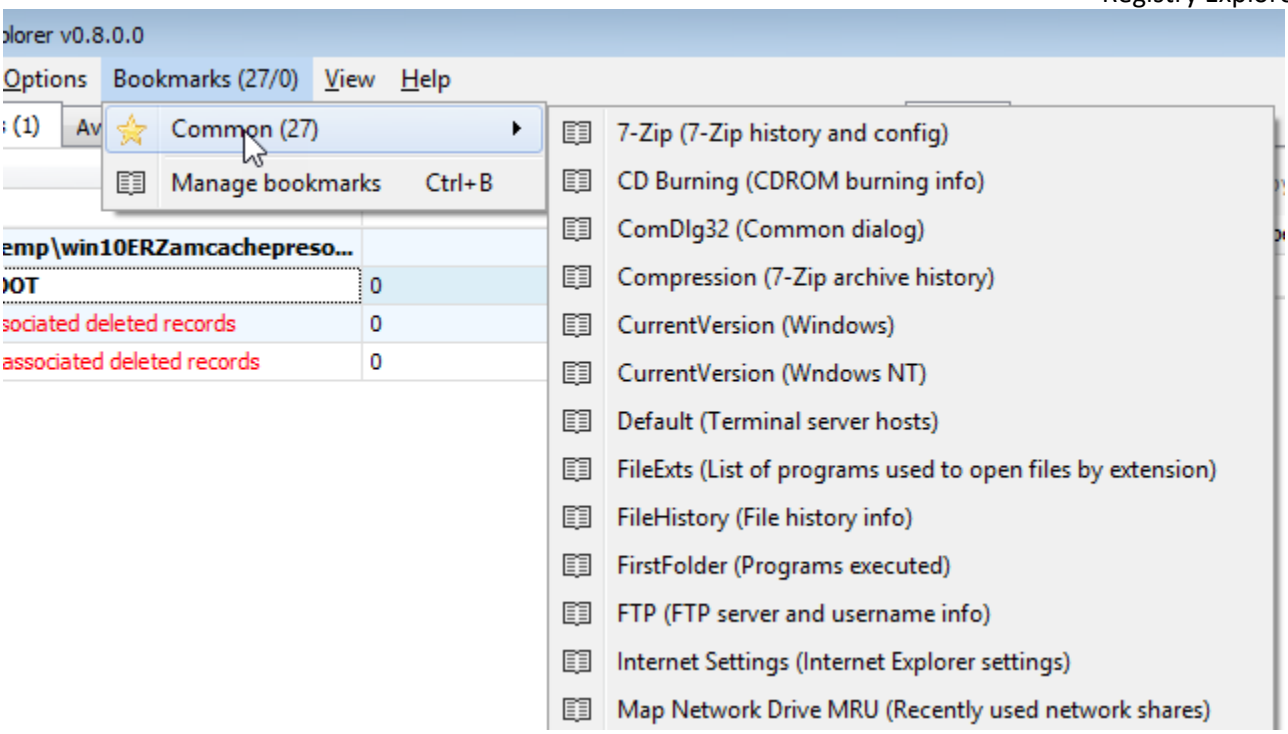
The main Bookmarks menu contains two numbers at the end. The first number is the total number of Common bookmarks ***that exist in the selected hive*** and the last number is the number of User bookmarks **that exist in the selected hive**. Clicking on any of the bookmarks will cause Registry Explorer 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).

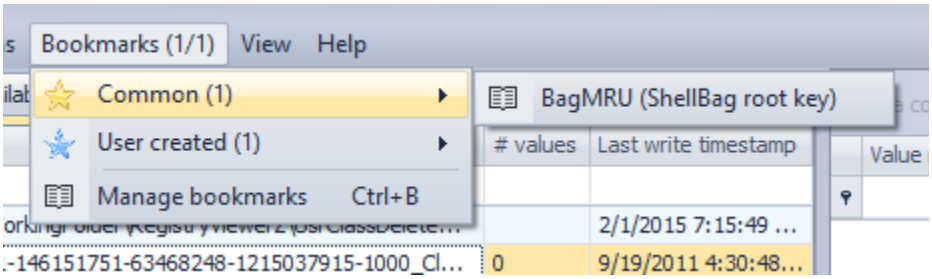
The Bookmarks menus dynamically adjust as hives are loaded and selected. For example, suppose you have the following bookmarks by hive type:

- NTUSER.DAT: 40 bookmarks
- USRCLASS.DAT: 8 bookmarks

You then load an NTUSER and USRCLASS hive. The NTUSER hive contains 27 out of the 40 key paths as defined in the NTUSER.DAT related bookmarks (27 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).

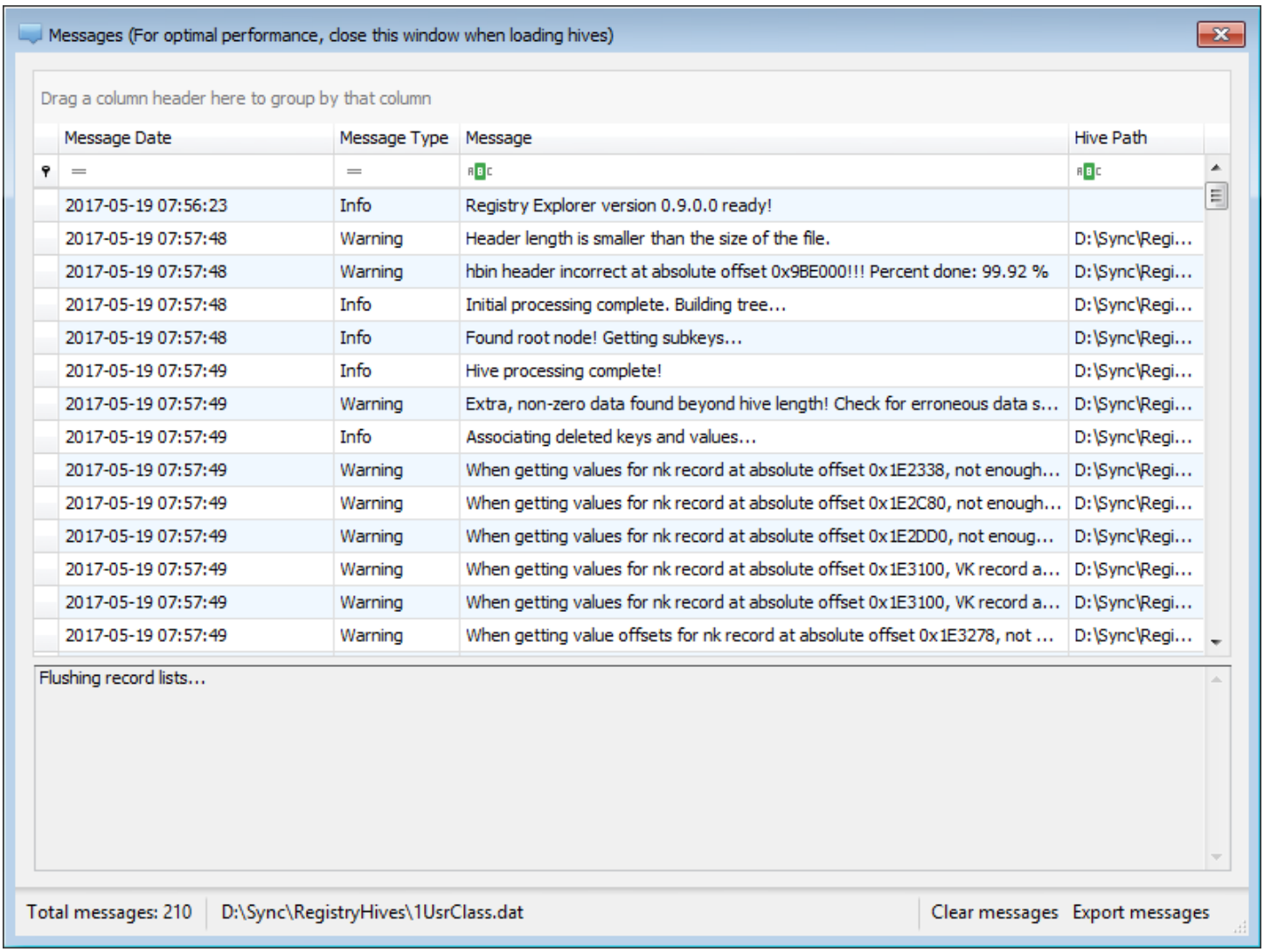
The bookmark names are sorted alphabetically as well so it's easy to find the bookmark you are interested in.

View

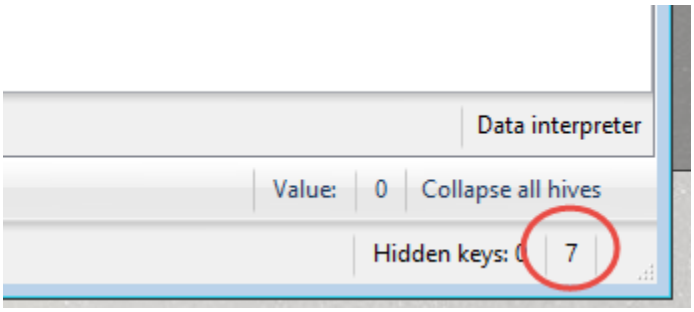
The View menu contains two options: Messages and Plugins.

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.

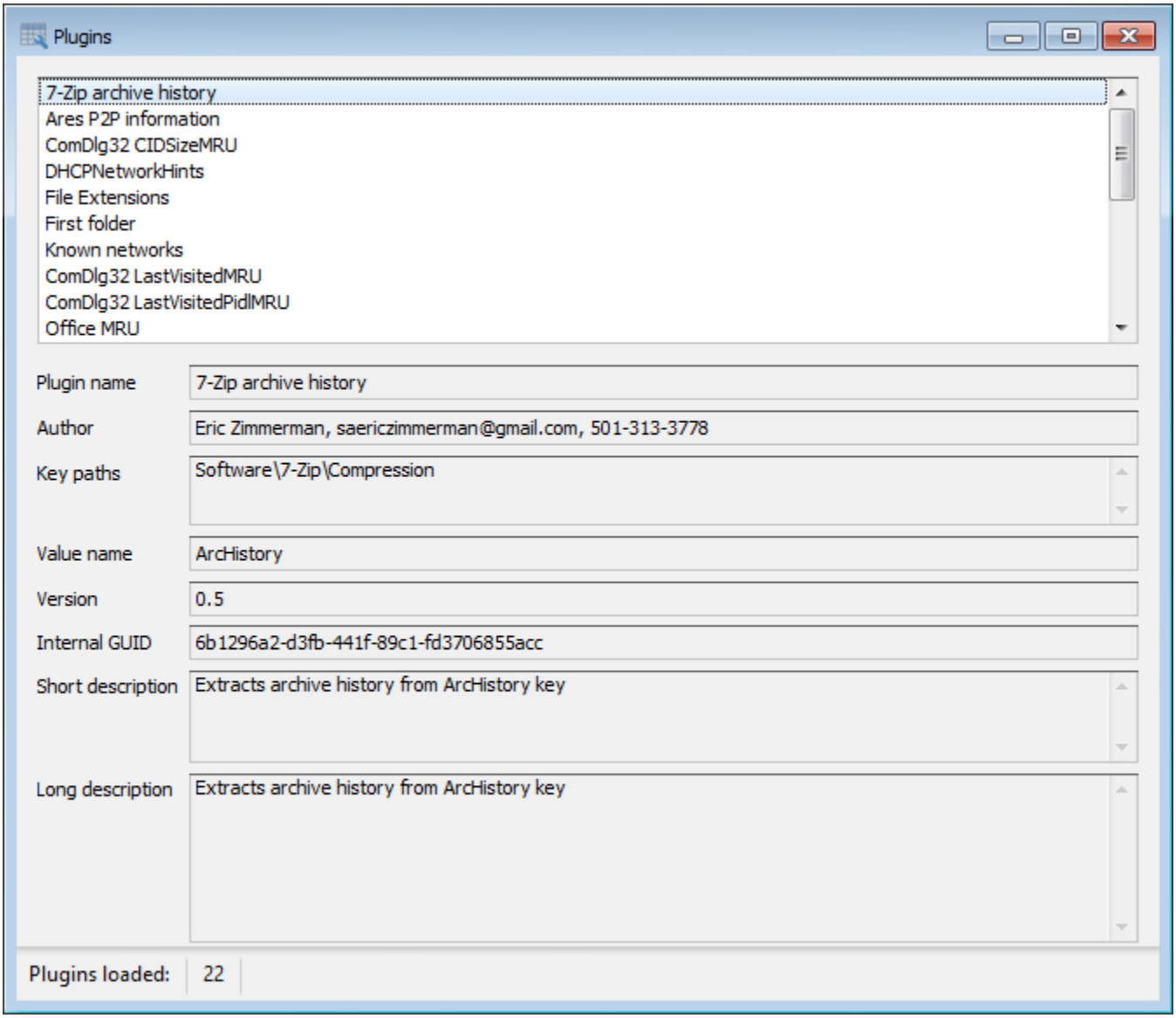


The total number of messages is also shown on the main window's bottom status bar to the far right. Double clicking the message count will show the Messages form.



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.

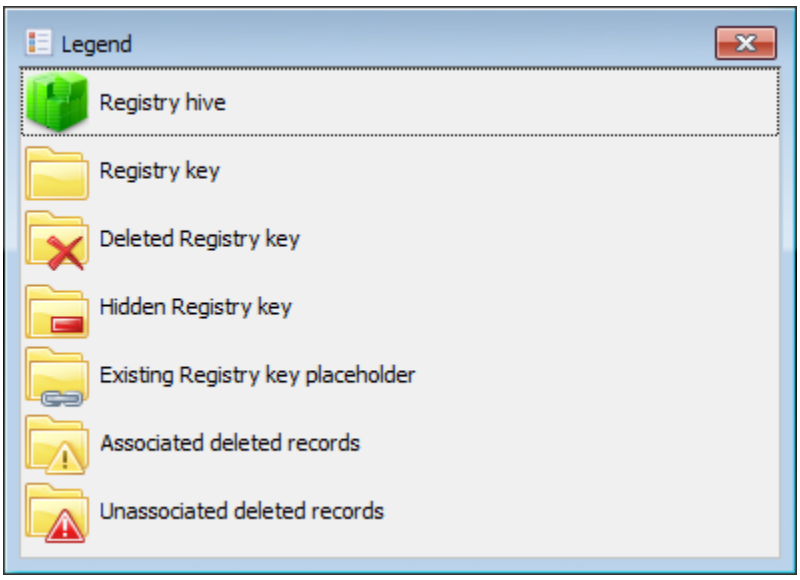


Plugins will be discussed in more detail in a dedicated section of this manual.

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.



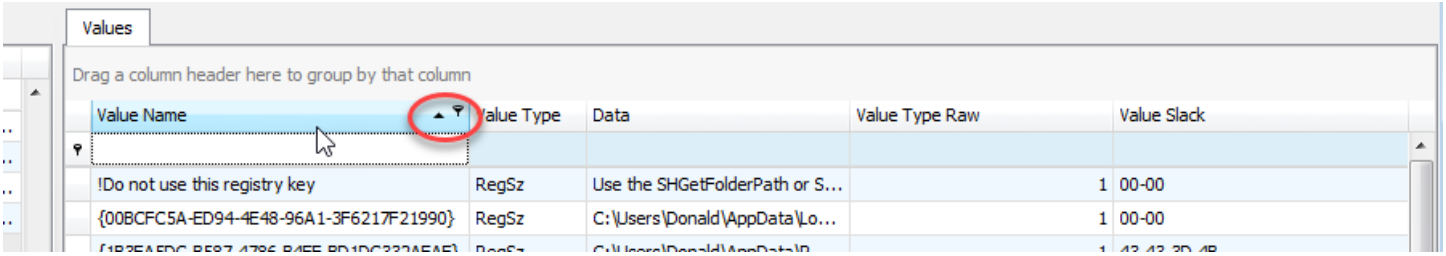
Using Registry Explorer

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

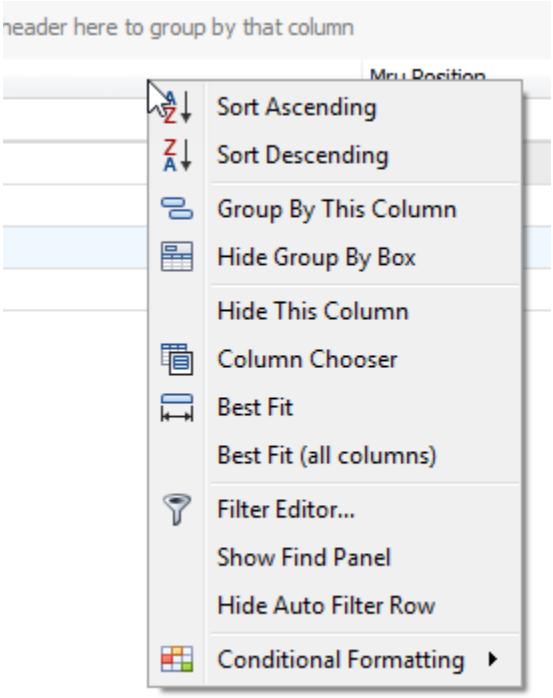
Sorting works like most every other program in that you can click on a column header to sort that column. Here the Value Name column has been sorted.



Notice the small arrow indicating the sort order.

Other options

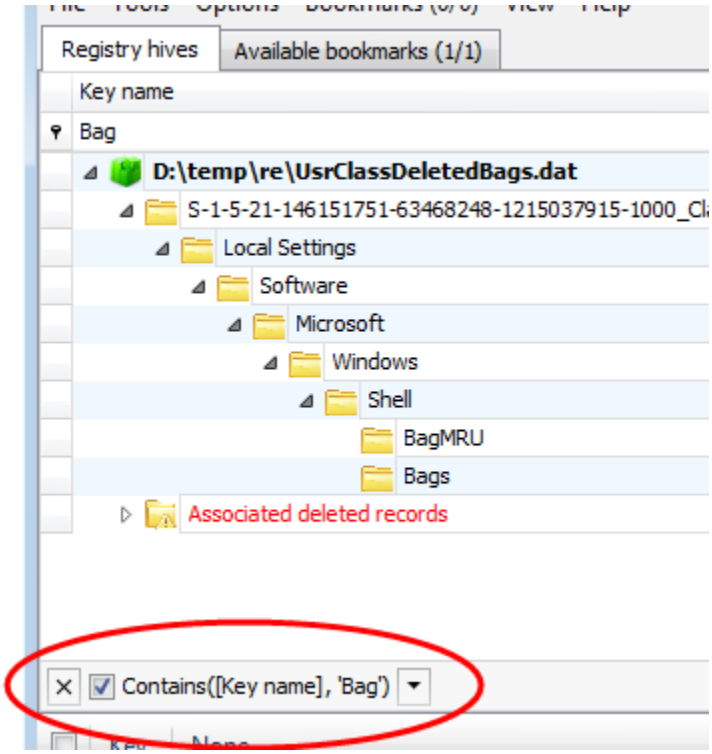
Right clicking on a column header will bring up a context menu that allows for sorting (as well as removing any existing sorting), grouping, and customization of columns including hiding or showing any columns.



Filtering

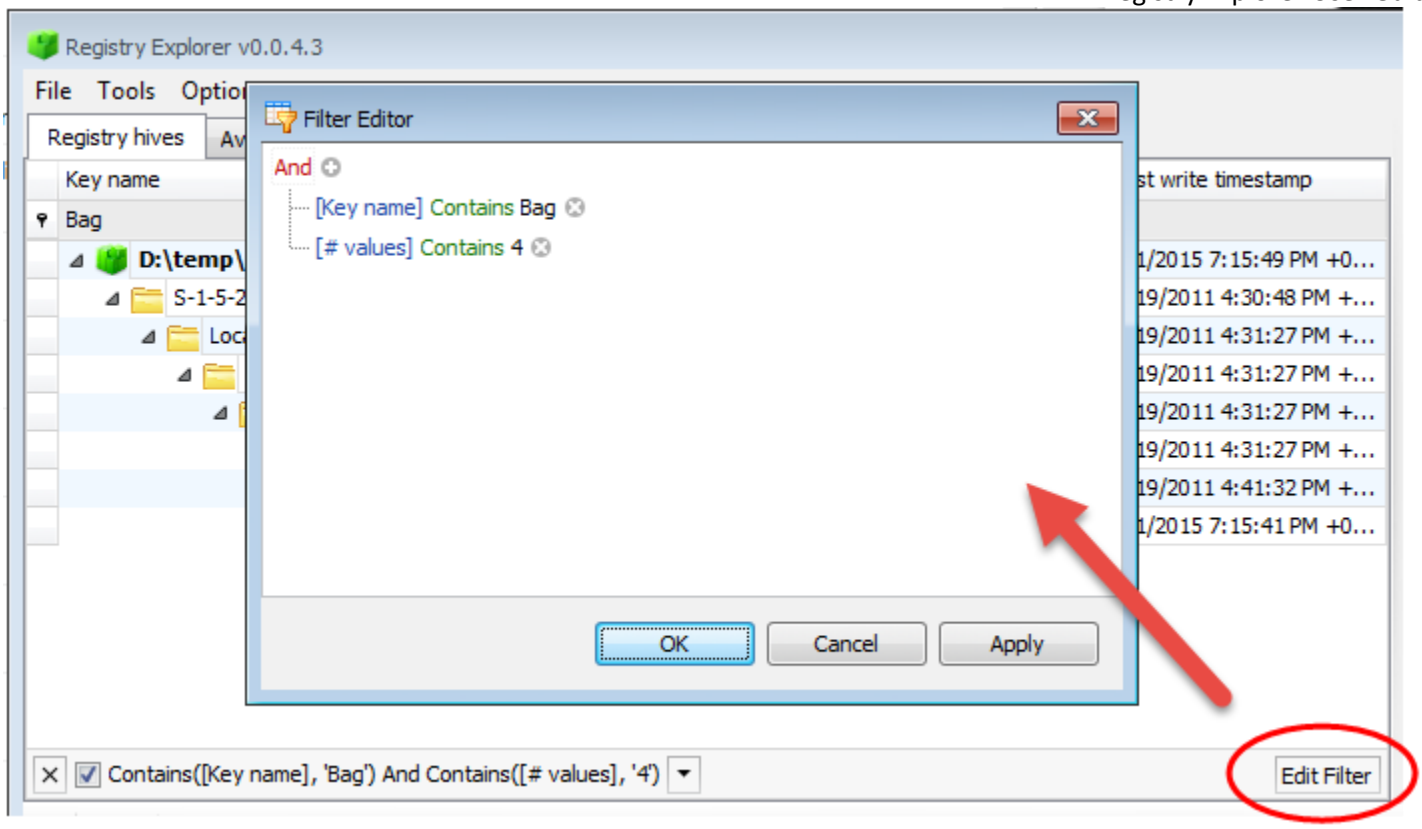
A column can be filtered by clicking in the blank space below the column header and entering something to filter by. The data will be filtered in real time and the bottom status bar will indicate how many things have been filtered out.

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.



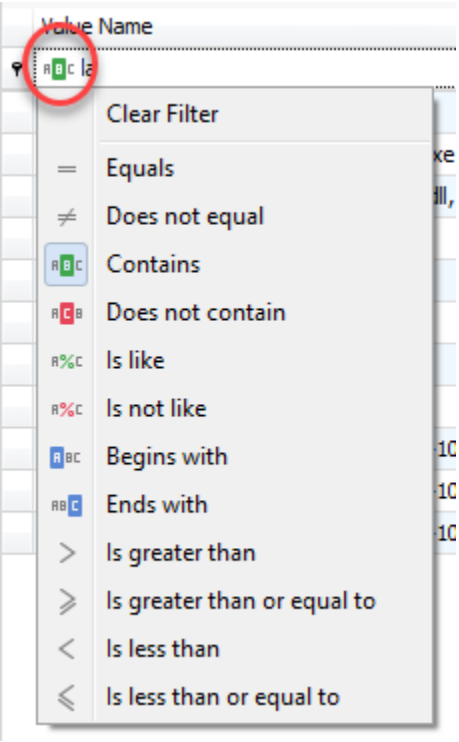
The leftmost X can be used to clear the active filter, the checkbox can be used to disable the filter without clearing it, and the down arrow on the right side contains a history of the different filters that have been recently used.

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.

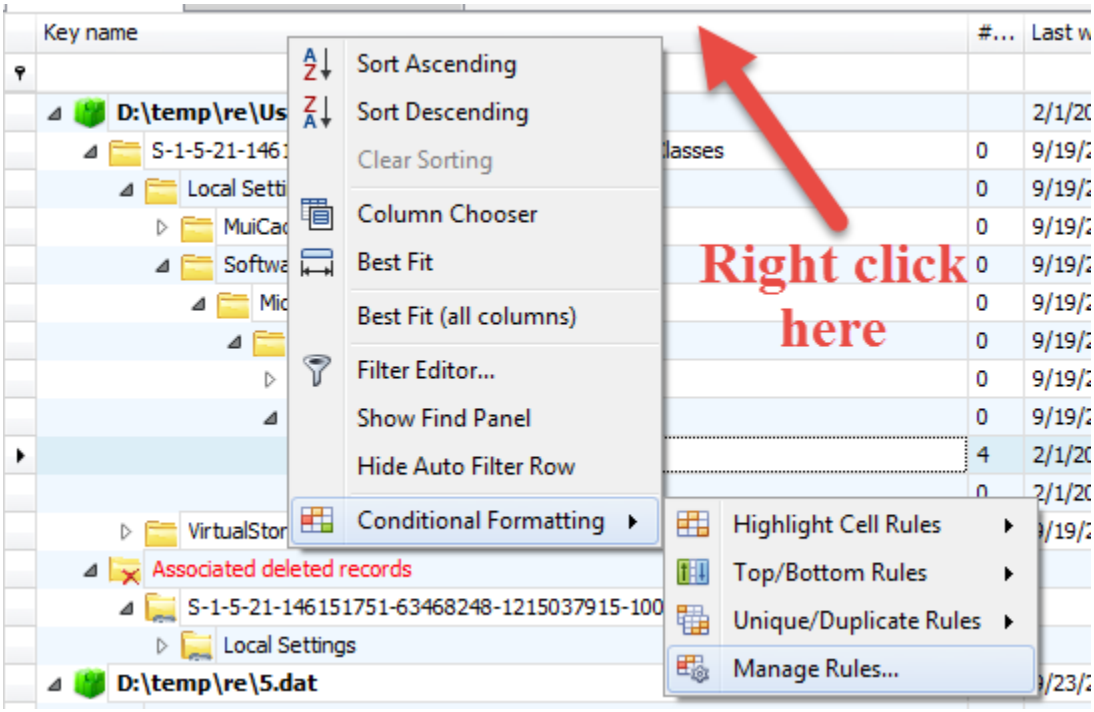
Clicking the icon in the left side of the filter area allows for changing the filter criteria as well. The default is 'Contains'



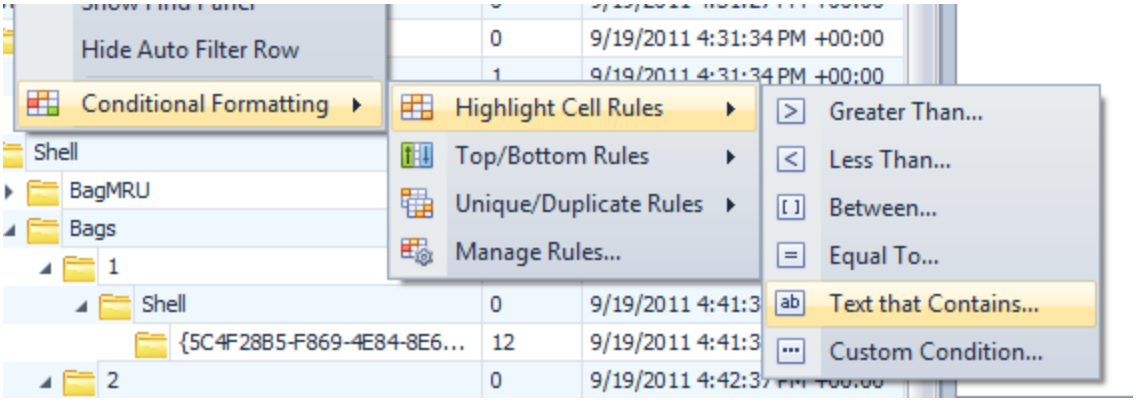
Conditional formatting

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.

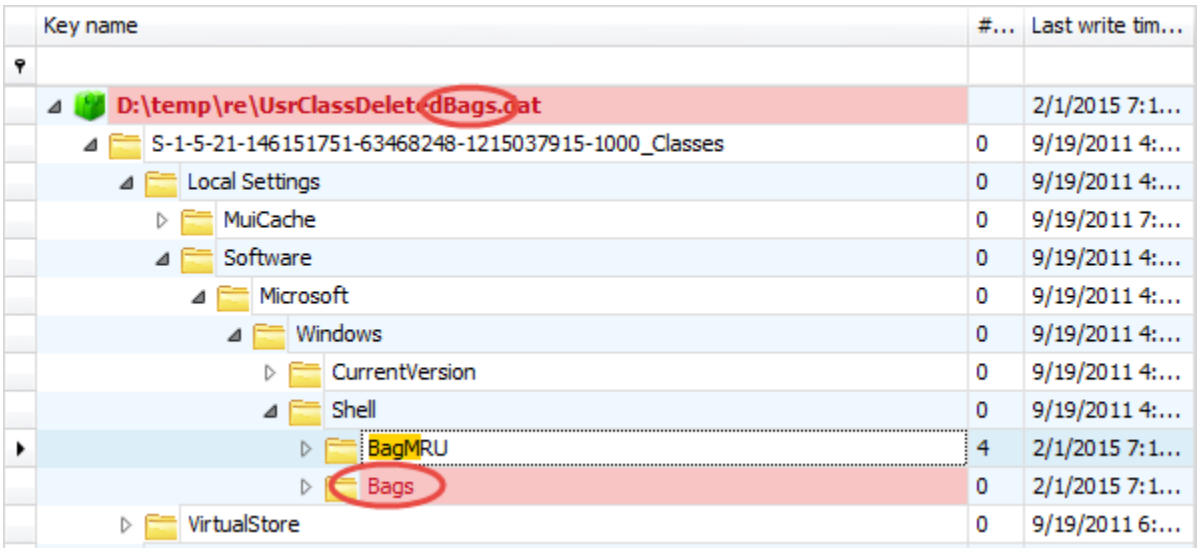


There are options to format things on a variety of conditions, as seen below.



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.



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.

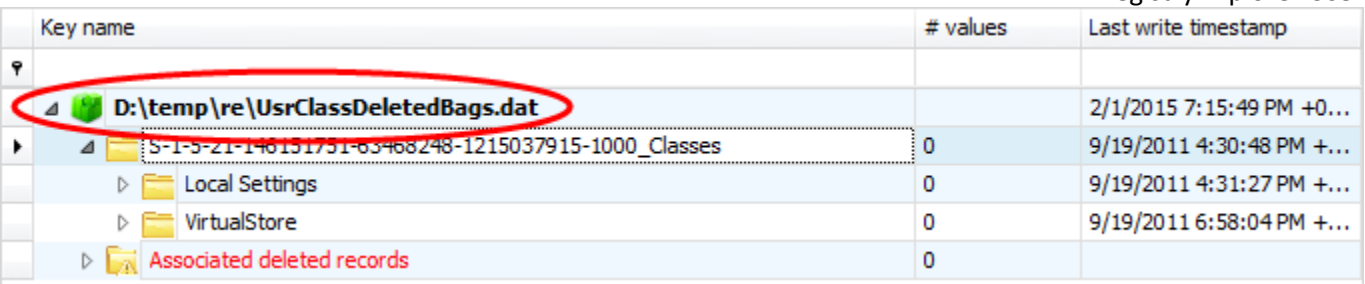
Use the same conditional formatting menu to edit rules.

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.

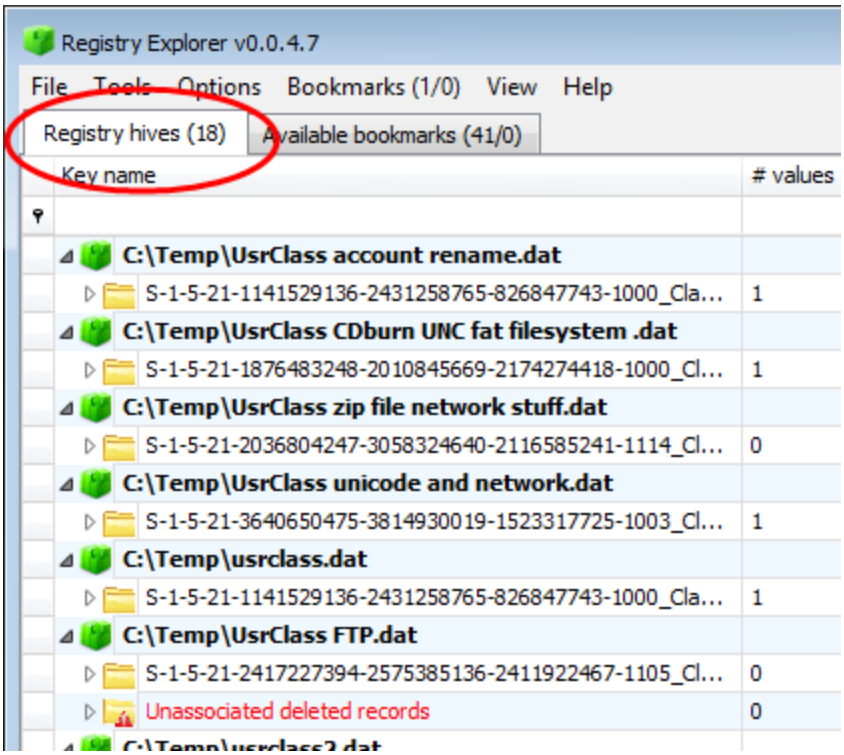


The last write timestamp for the hive is the timestamp value from the header of the hive.

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.



Projects

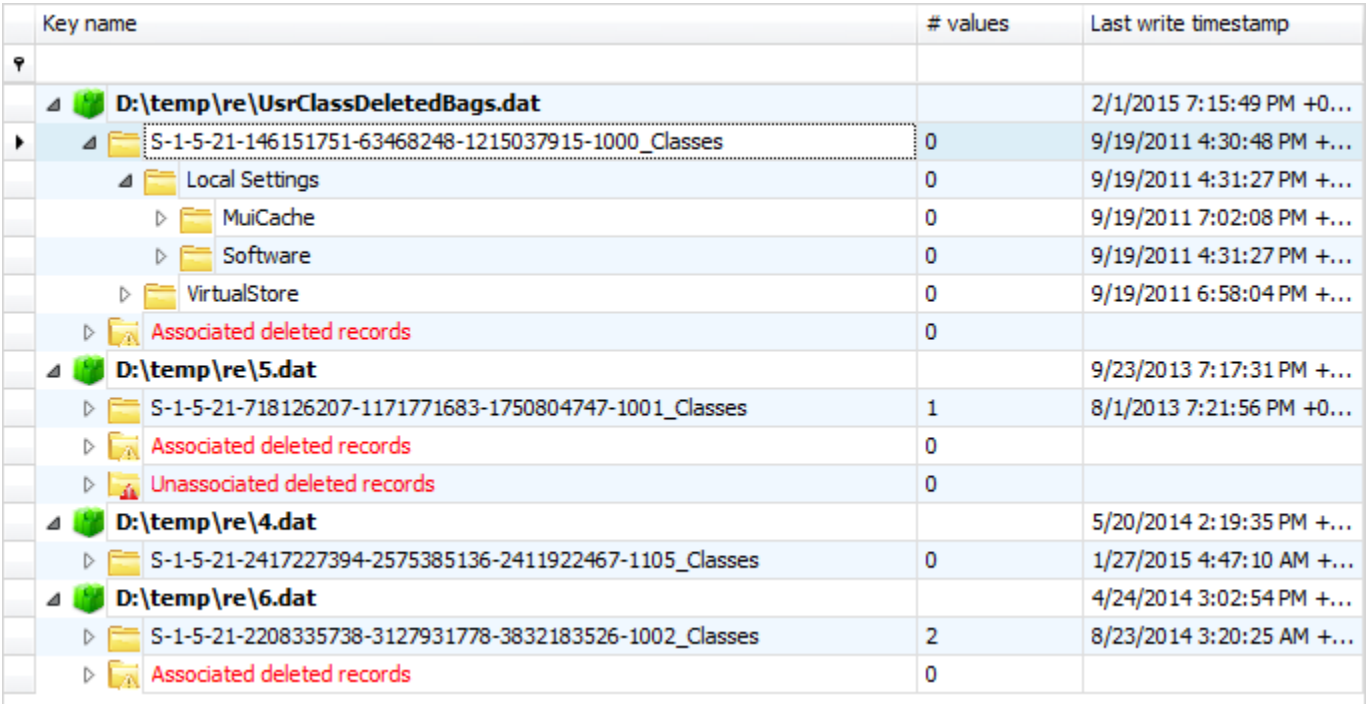
Projects allow you to load one or more hives into Registry Explorer and save the currently loaded hives into a project file. This allows you quickly load the same hives for a particular case quickly vs having to load a bunch of hives individually. You can also drag and drop Registry Explorer project 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.

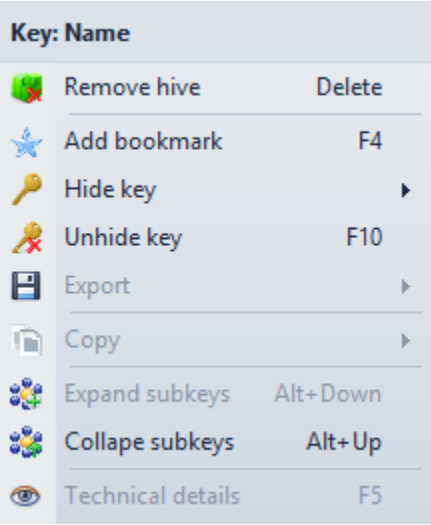
The **Options | Show parent keys when filtering** option affects what is shown when filtering keys. See the Options section for a full discussion on how this option works.

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:



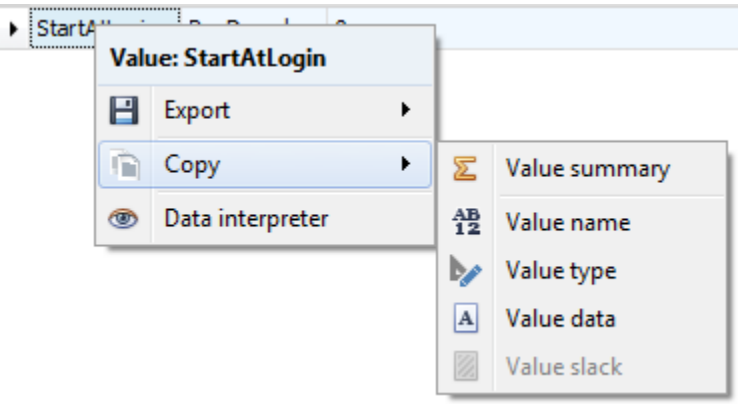
The name of the currently selected key is shown at the top. Most options also have shortcuts which can be used in lieu of using the mouse.

- **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
    - **Key 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:

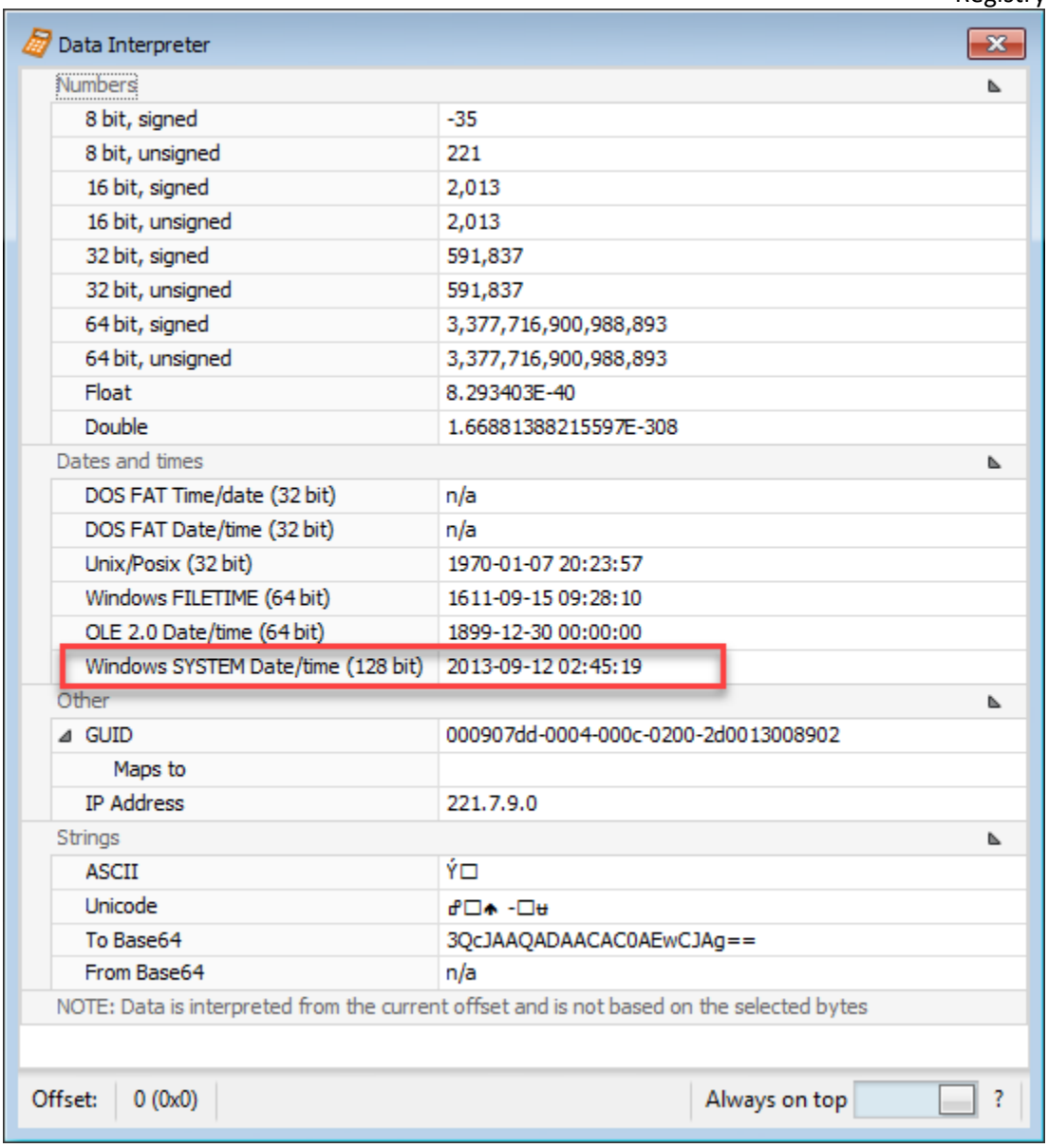


The name of the currently selected value is shown at the top.

- 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.

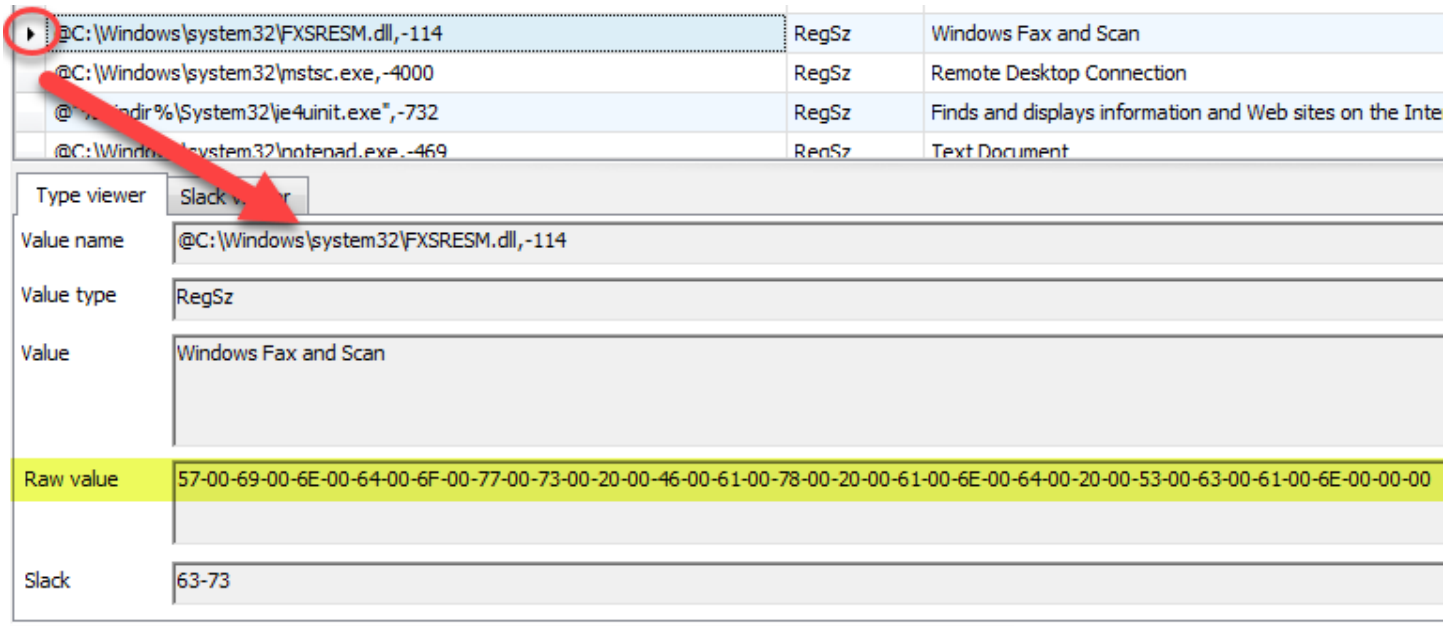


Value details

The Value details area will change depending on the type of value selected.

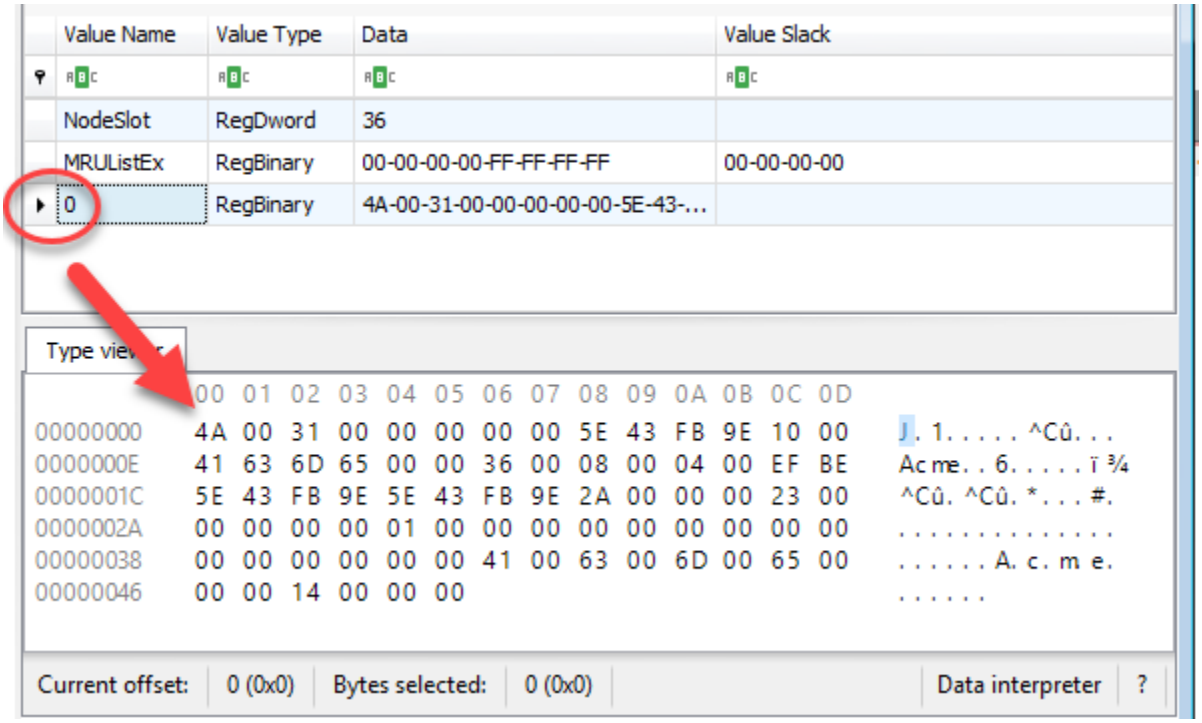
Type viewer

For all values except RegBinary values, a simple string representation of the value is shown as seen below.



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.



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
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.....ï¾
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: 62 (0x3E) | Bytes selected: 8 (0x8) | Data interpreter | ?

Slack viewer

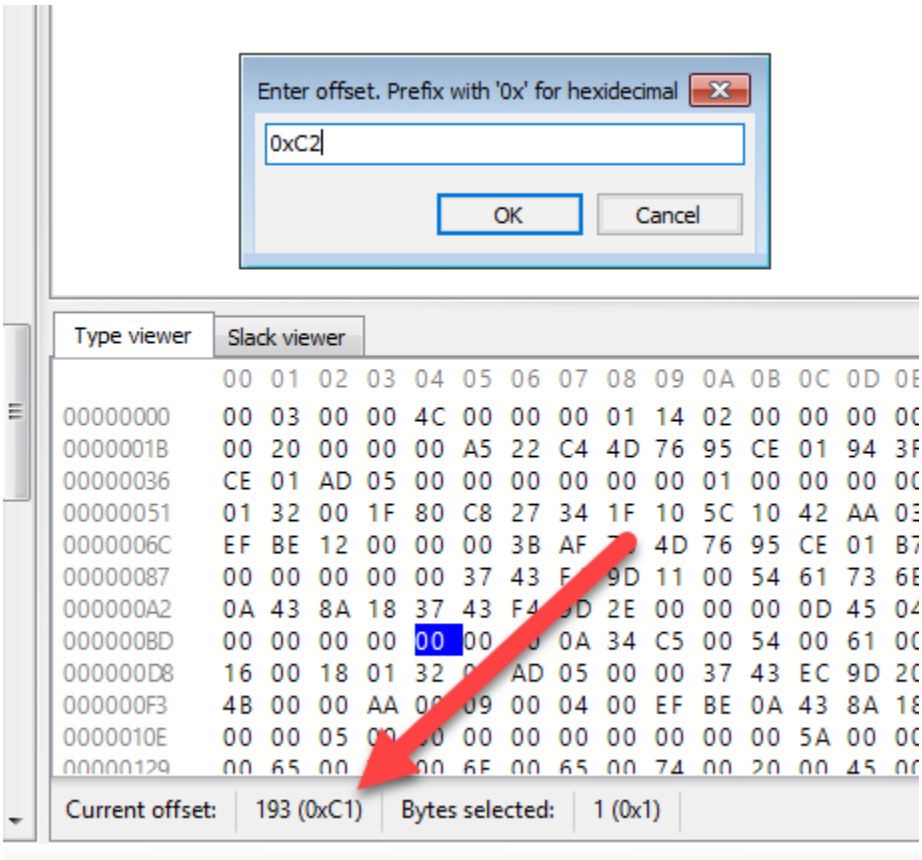
For values that have value slack, a Slack viewer tab will be added. This viewer works the same as the Type viewer for RegBinary values.

Type viewer | Slack viewer

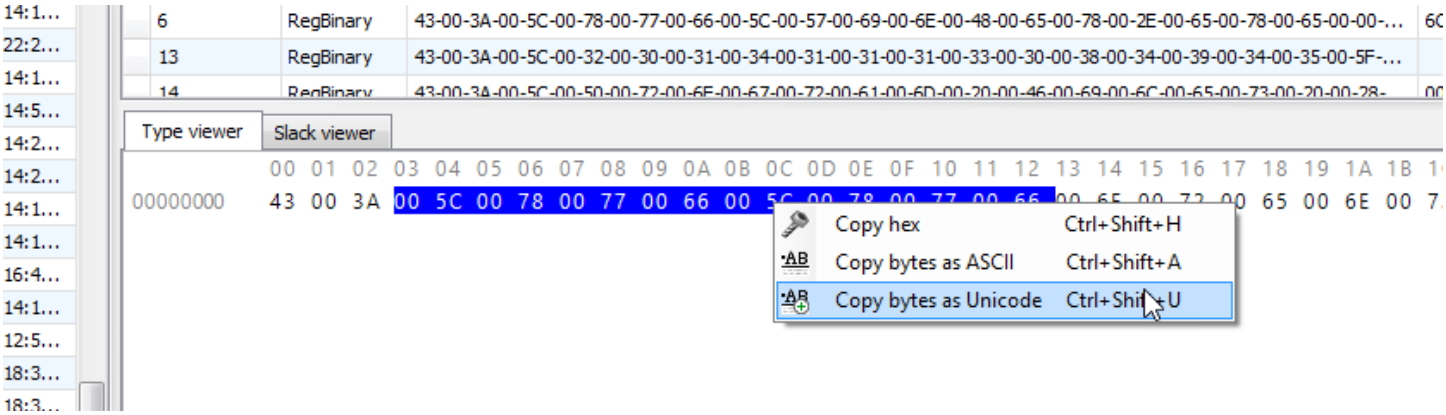
```
            00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
00000000    5C 57 69 6E 64 6F                            \Windo
```

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.

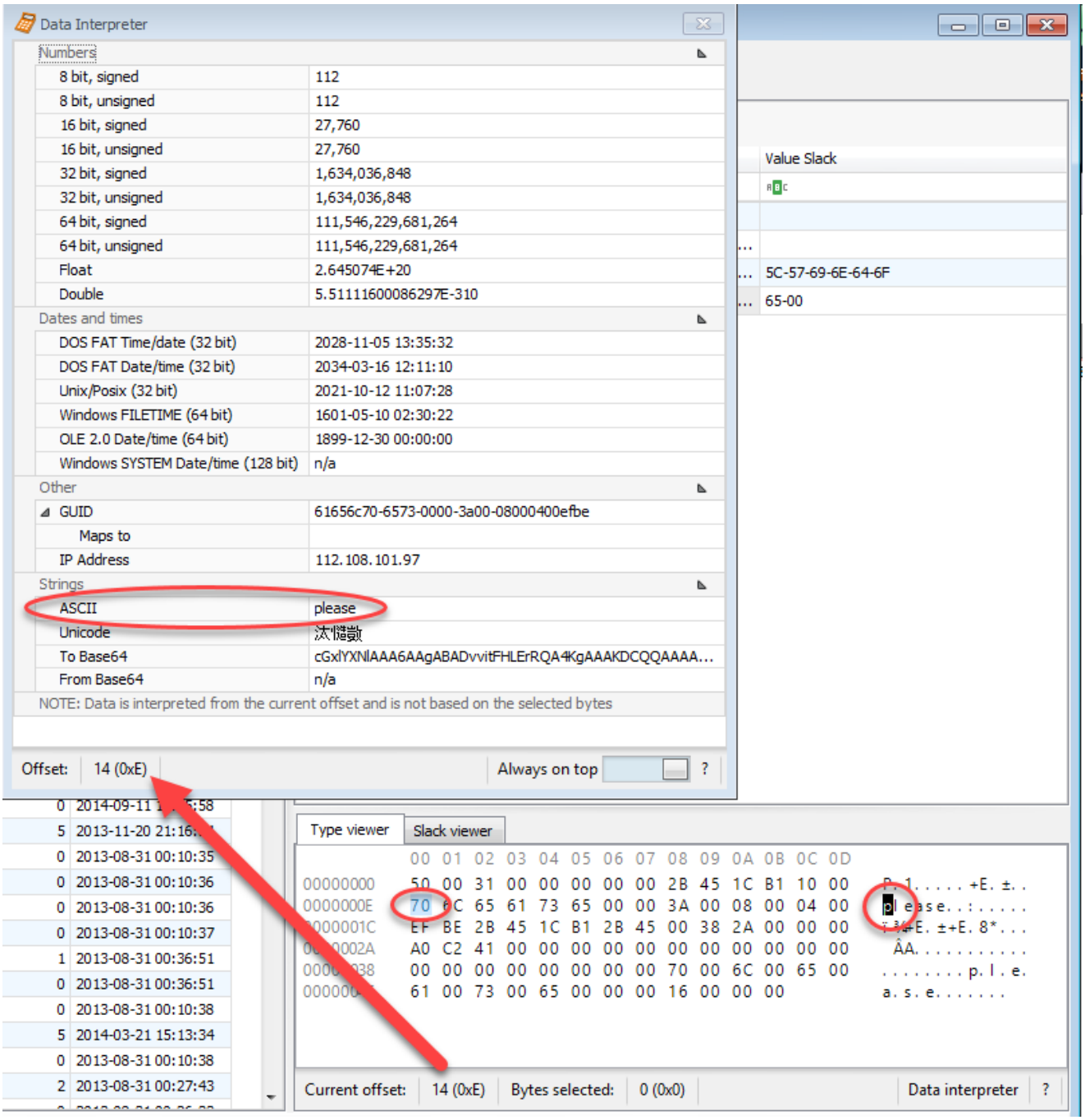


When viewing binary data, you can copy the selected bytes to the clipboard as either hex, ASCII, or Unicode via the context menu:



Data interpreter

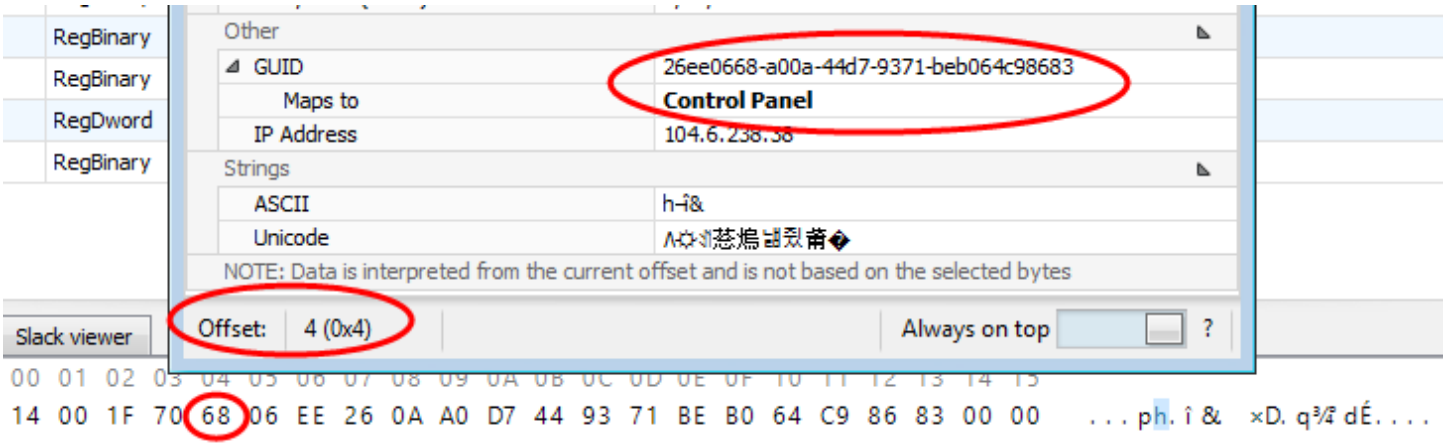
In the lower right corner of the hex viewer is a Data interpreter button. Clicking this button will bring up the Data interpreter that converts the raw hexadecimal data into a variety of formats including dates and times, GUIDs, IP addresses, and more. The Data interpreter window is shown below.



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.



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

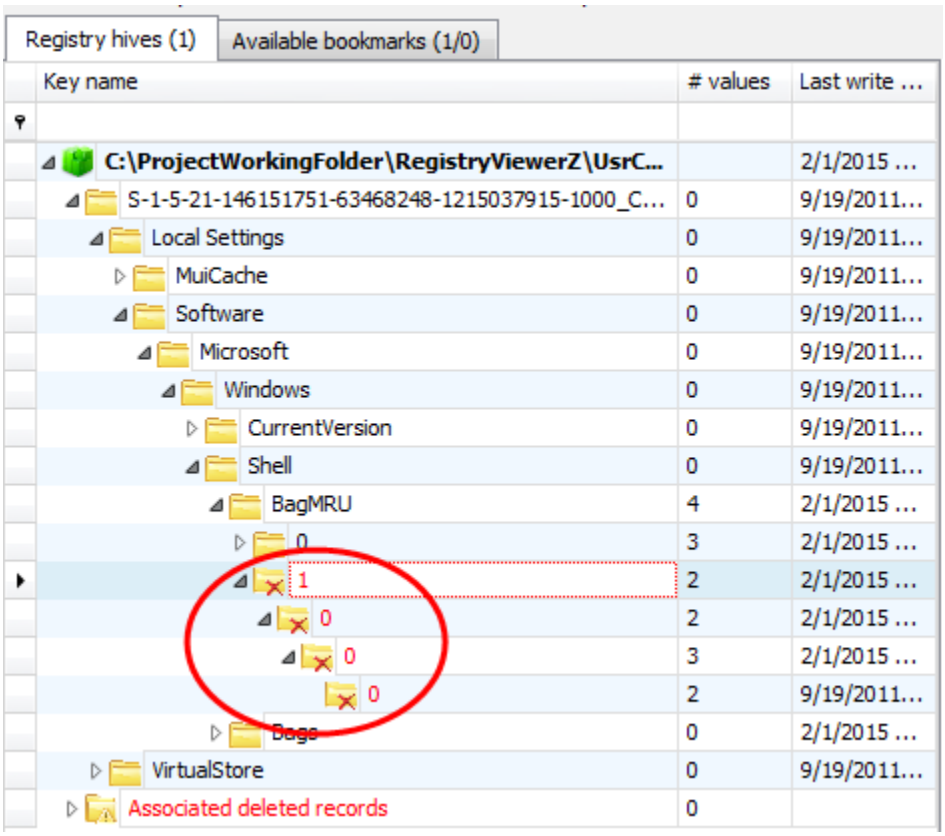
Registry Explorer can recover both deleted Registry keys and values. It also reassociates deleted values with their parent keys and subkeys to their parent keys.

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).

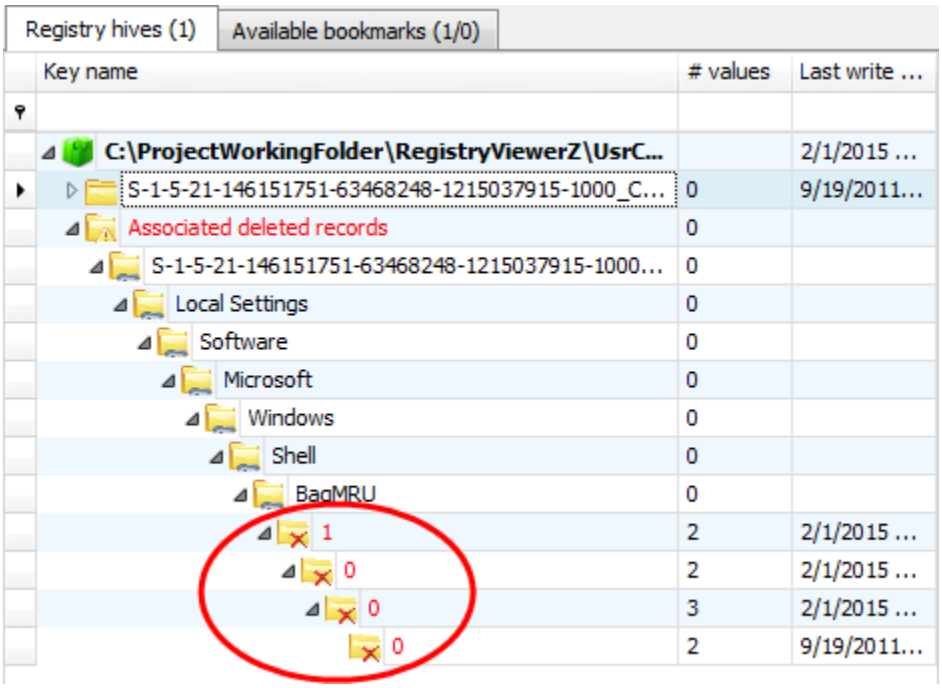
Inlined with existing keys

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.



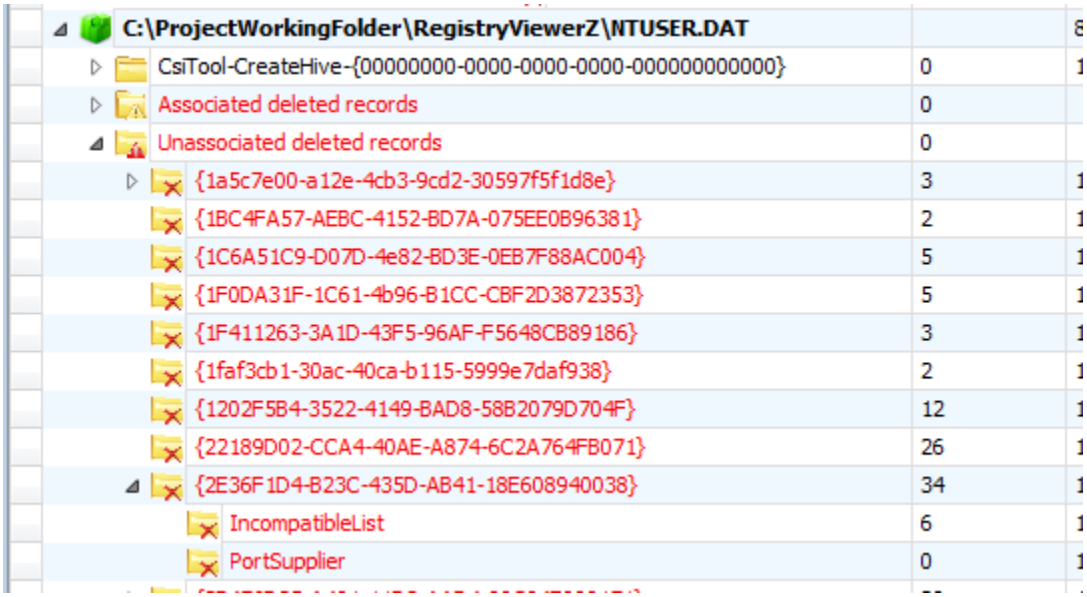
Associated deleted records

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.



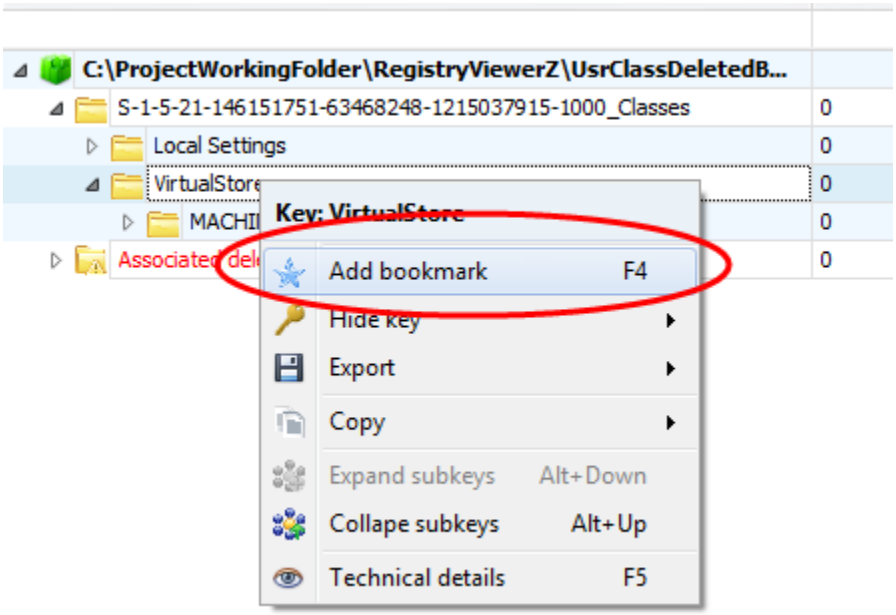
Unassociated deleted records

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.).

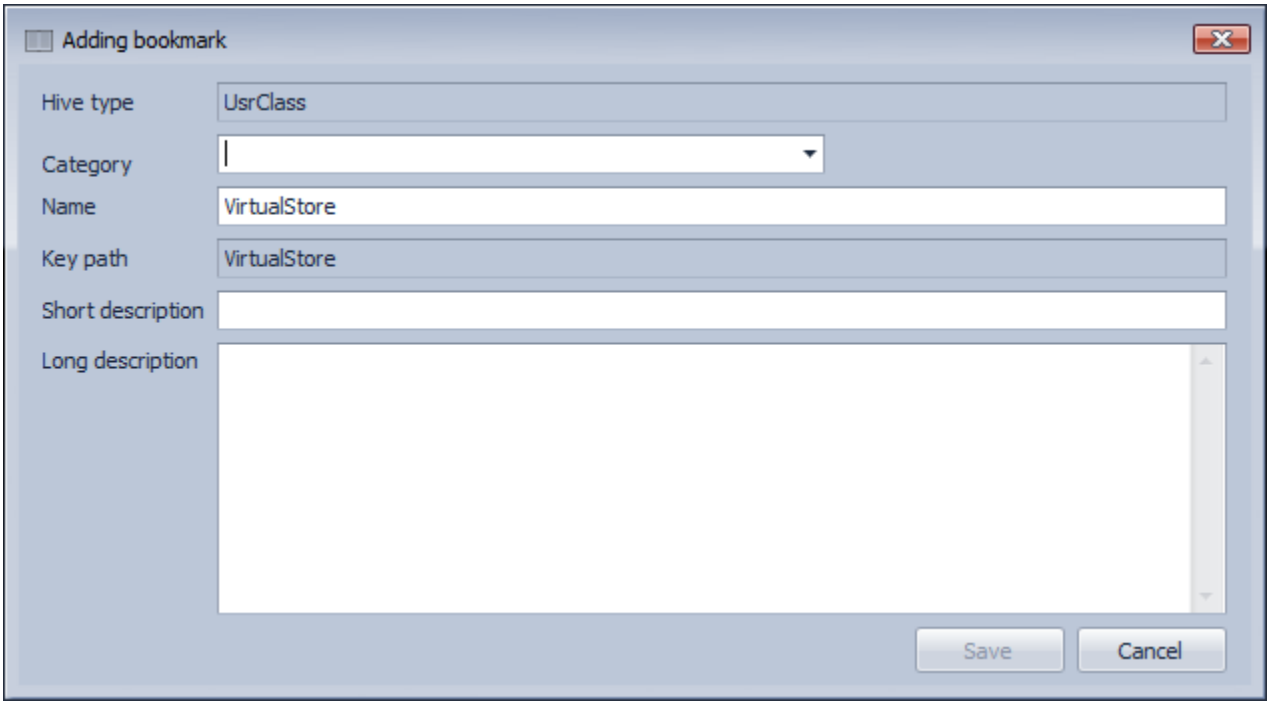


Creating bookmarks

To create a bookmark, right click on a key and select Add bookmark.



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.

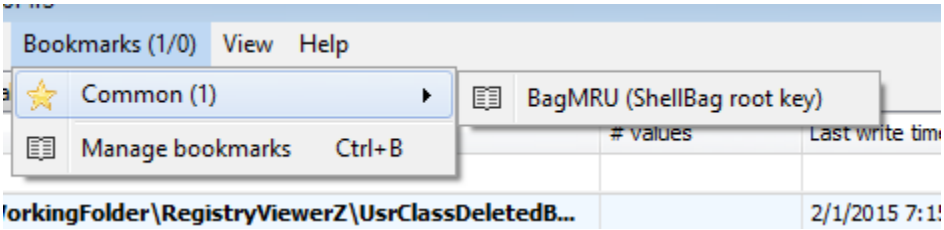


The Category field allows you to place this particular Registry key into a high-level group. This will eventually be used for reporting. Several preexisting categories are included, but typing a new Category will add it to the list.

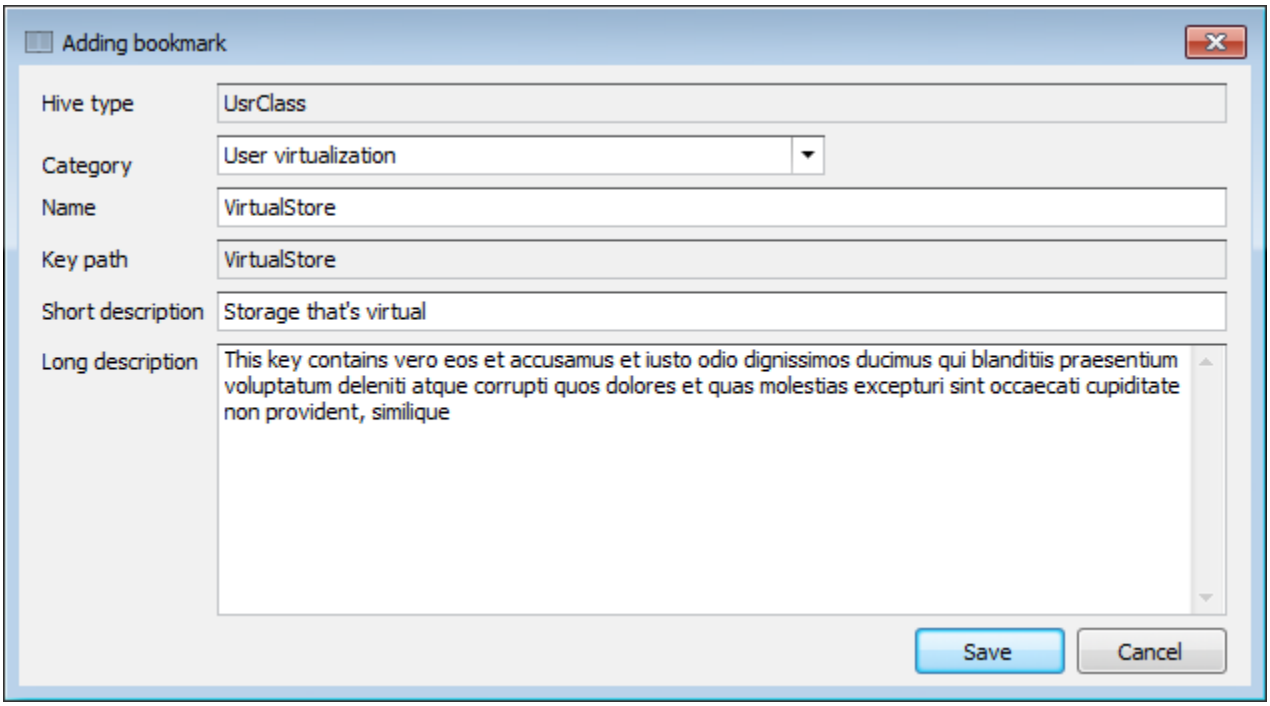
The Short description serves as a summary for what the key means or why it is relevant. The value entered for Short description will show up after the name of the bookmark in the bookmarks menu.

The Long description should contain technical information, links to web pages with more information, or any other information you want to convey.

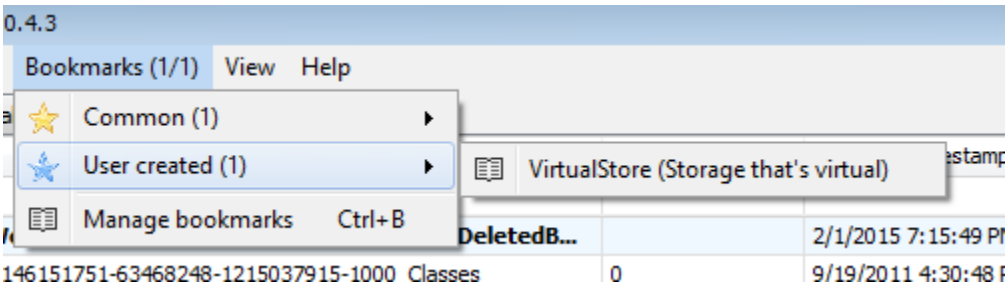
Recall the bookmarks menu is dynamic and will update according to the keys that are available for the selected hive. If, before adding the bookmark, the Bookmarks menu looked like this:



And our new bookmark looks like this:



The Bookmarks menu will look like this once the Save button is clicked:

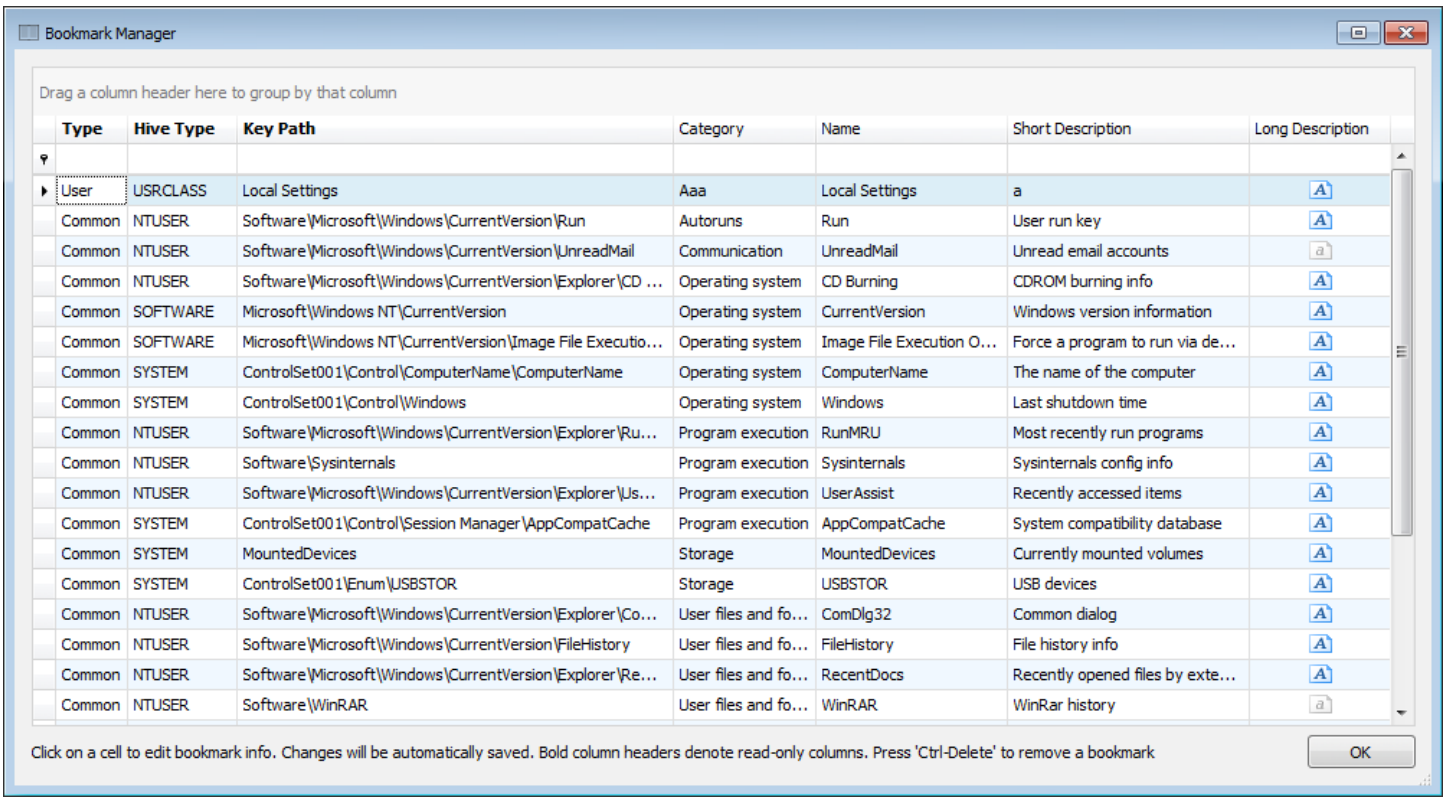


A 'User created' menu is now visible as is our VirtualStore bookmark (with the short description shown in parenthesis after the key name).

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.



Type	Hive Type	Key Path	Category	Name	Short Description	Long Description
User	USRCLASS	Local Settings	Aaa	Local Settings	a	
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Run	Autoruns	Run	User run key	
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\UnreadMail	Communication	UnreadMail	Unread email accounts	
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Explorer\CD ...	Operating system	CD Burning	CDROM burning info	
Common	SOFTWARE	Microsoft\Windows NT\CurrentVersion	Operating system	CurrentVersion	Windows version information	
Common	SOFTWARE	Microsoft\Windows NT\CurrentVersion\Image File Executio...	Operating system	Image File Execution O...	Force a program to run via de...	
Common	SYSTEM	ControlSet001\Control\ComputerName\ComputerName	Operating system	ComputerName	The name of the computer	
Common	SYSTEM	ControlSet001\Control\Windows	Operating system	Windows	Last shutdown time	
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Explorer\Ru...	Program execution	RunMRU	Most recently run programs	
Common	NTUSER	Software\Sysinternals	Program execution	Sysinternals	Sysinternals config info	
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Explorer\Us...	Program execution	UserAssist	Recently accessed items	
Common	SYSTEM	ControlSet001\Control\Session Manager\AppCompatCache	Program execution	AppCompatCache	System compatibility database	
Common	SYSTEM	MountedDevices	Storage	MountedDevices	Currently mounted volumes	
Common	SYSTEM	ControlSet001\Enum\USBSTOR	Storage	USBSTOR	USB devices	
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Explorer\Co...	User files and fo...	ComDlg32	Common dialog	
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\FileHistory	User files and fo...	FileHistory	File history info	
Common	NTUSER	Software\Microsoft\Windows\CurrentVersion\Explorer\Re...	User files and fo...	RecentDocs	Recently opened files by exte...	
Common	NTUSER	Software\WinRAR	User files and fo...	WinRAR	WinRar history	

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.

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 ...

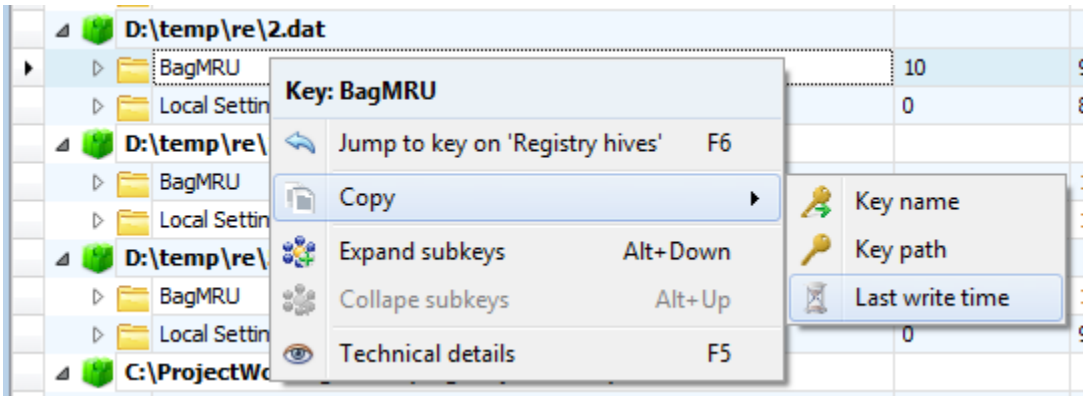
Bookmark information

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

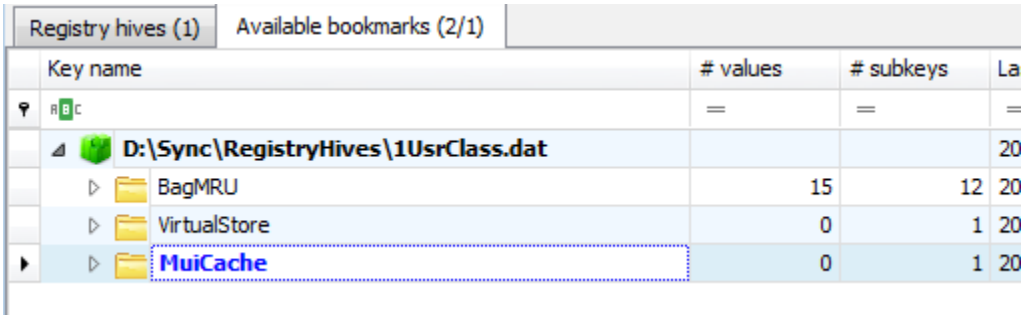
When the root folder for a bookmark is selected (BagMRU in the example above), information about the bookmark is shown at the bottom of the window in the Bookmark information section.

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.



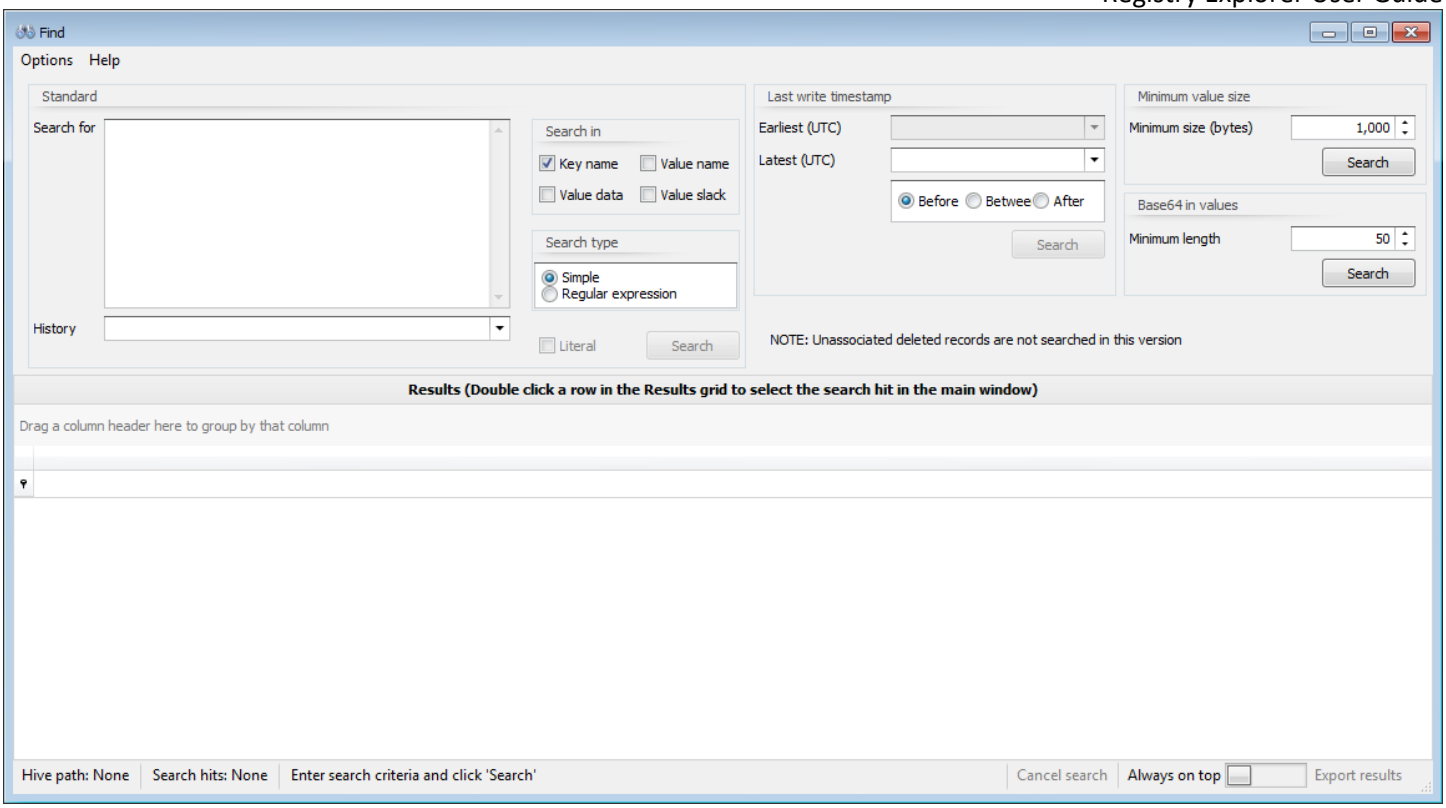
Finally, common bookmarks are differentiated from user created bookmarks by showing user created bookmarks in blue, as shown below:



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 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

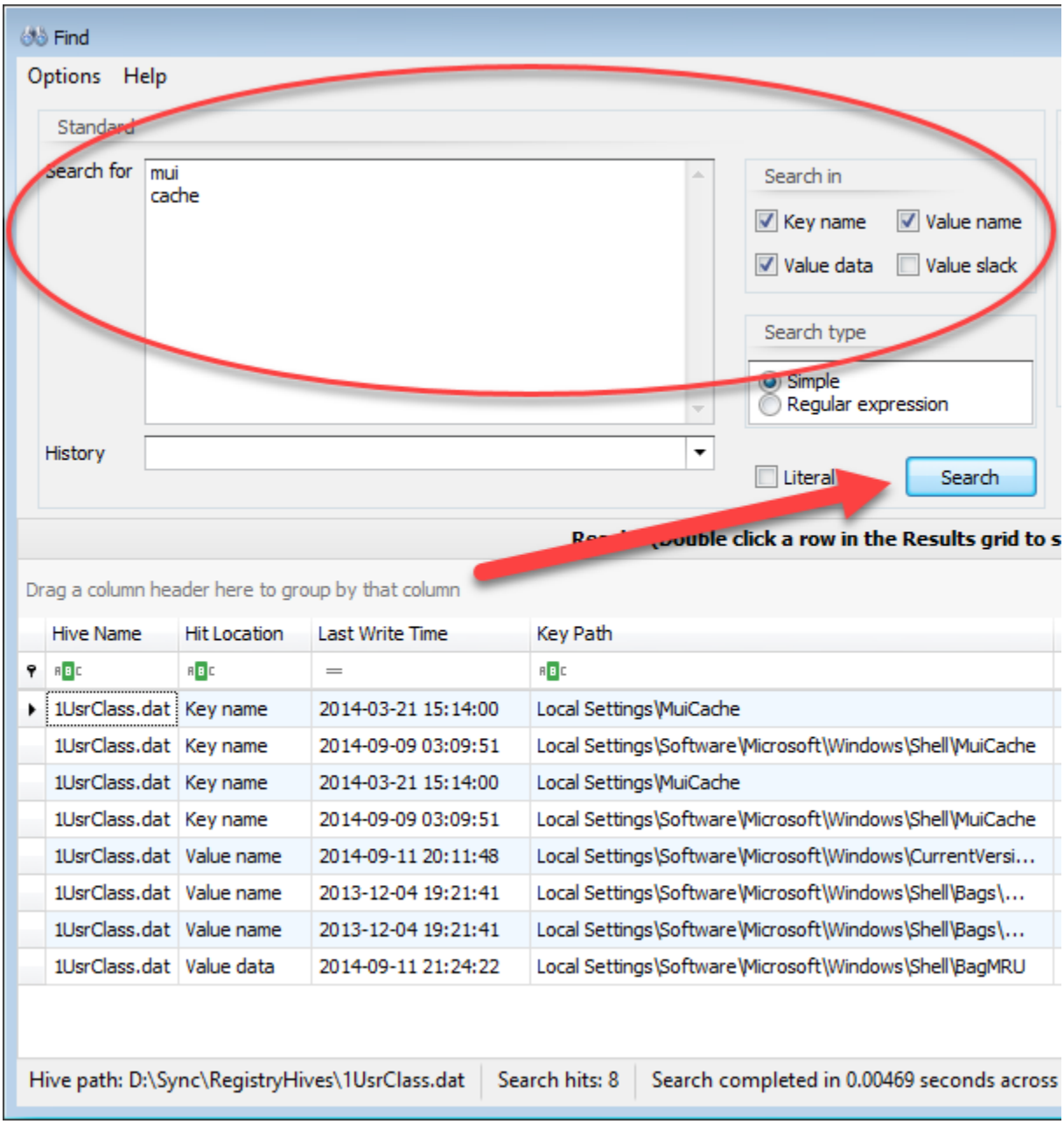
Shows several tips for different kinds of searches

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.



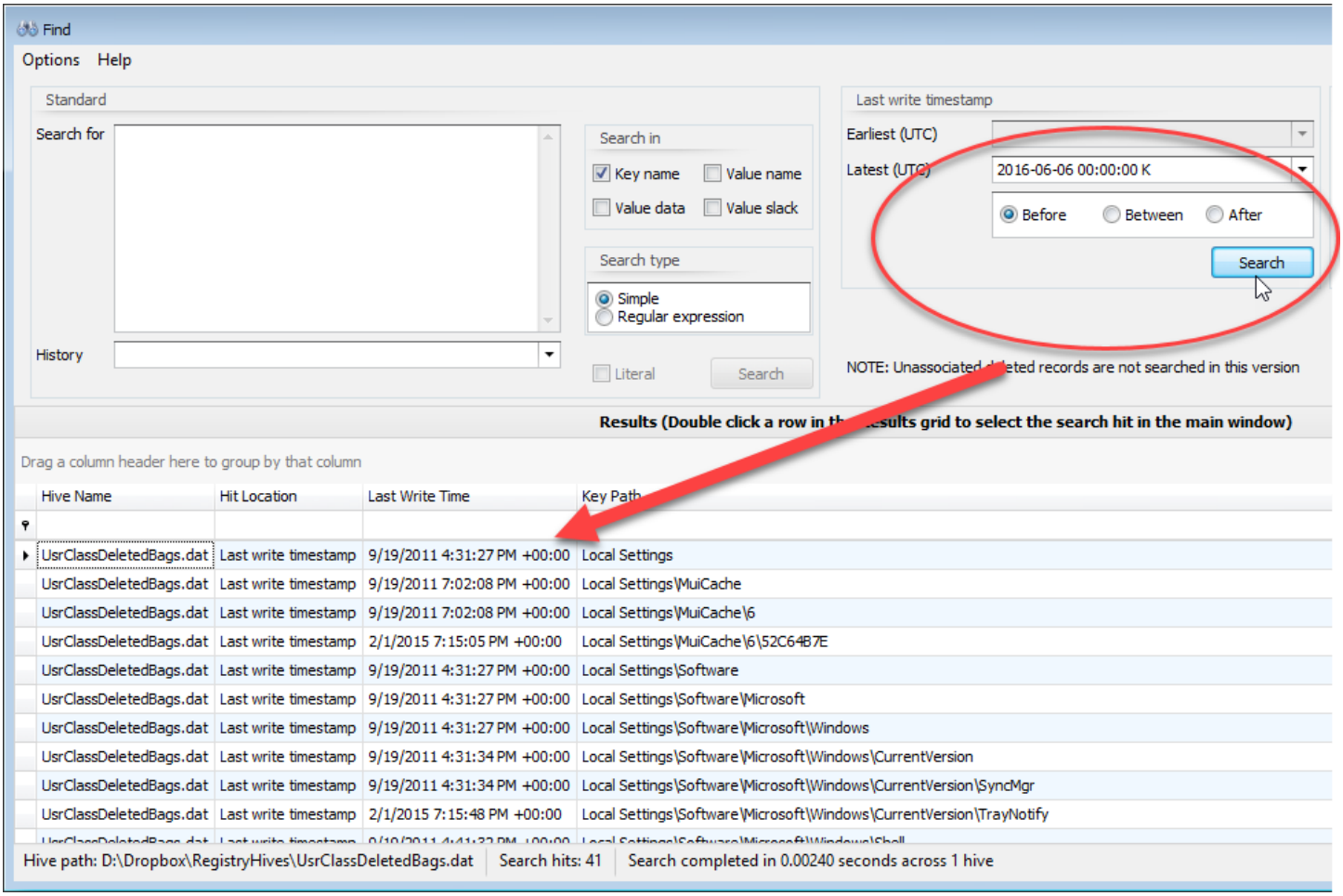
The Literal checkbox controls whether the term searched for is looked for in binary data when searching in value data and/or slack. This is explained in more detail below.

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

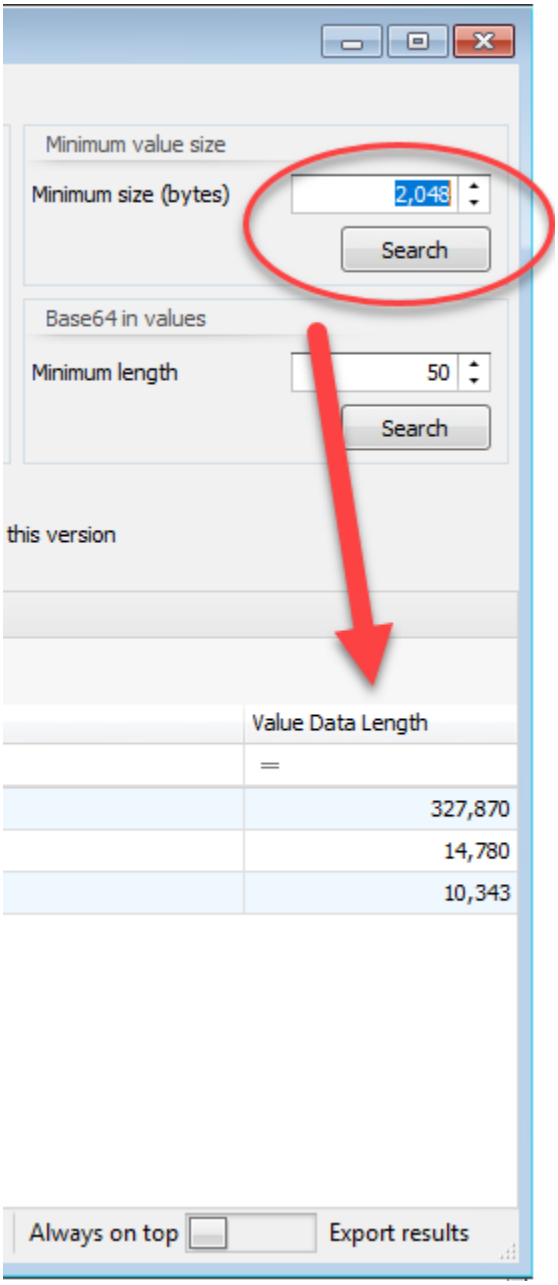
To conduct a last write timestamp search, choose the date range to search for via the radio buttons and enter the required time stamp values, and then click Search (or press Enter).



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.

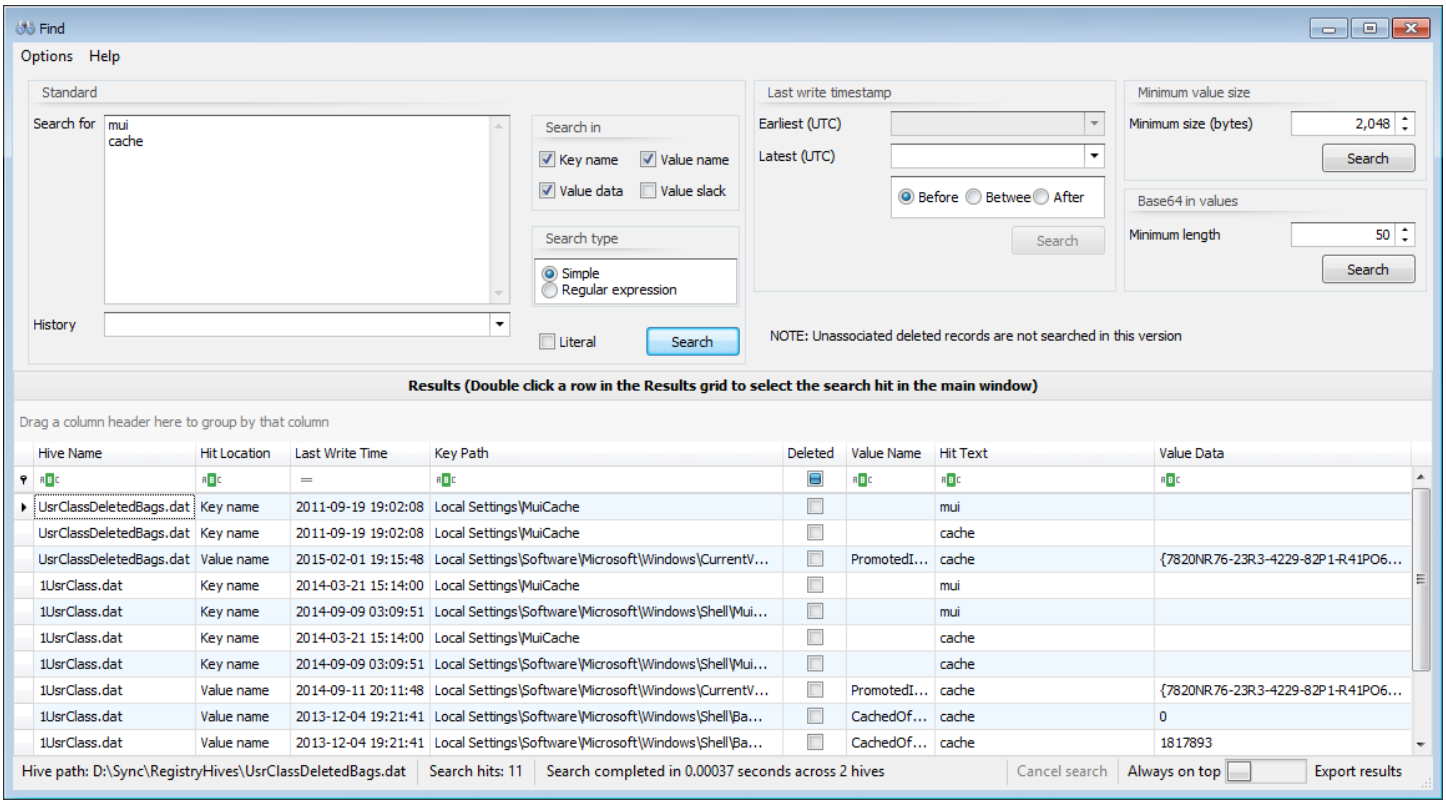


Base64 in values search

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 contains 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.

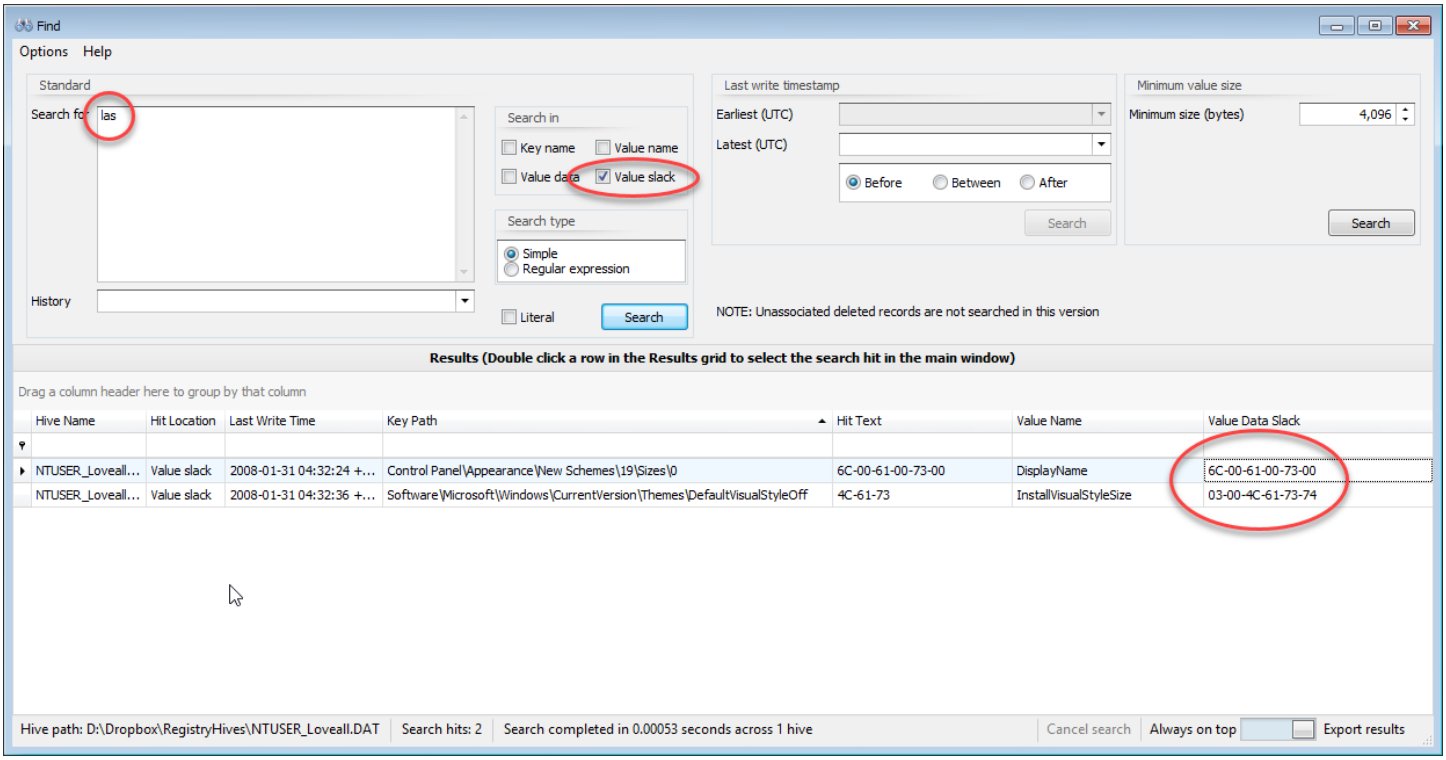
When searching in value data and/or value slack, the Search for term will be found regardless of case or encoding (Western 1252 and/or Unicode to be exact). This makes it easy to find strings that have been encoded in binary data.

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.

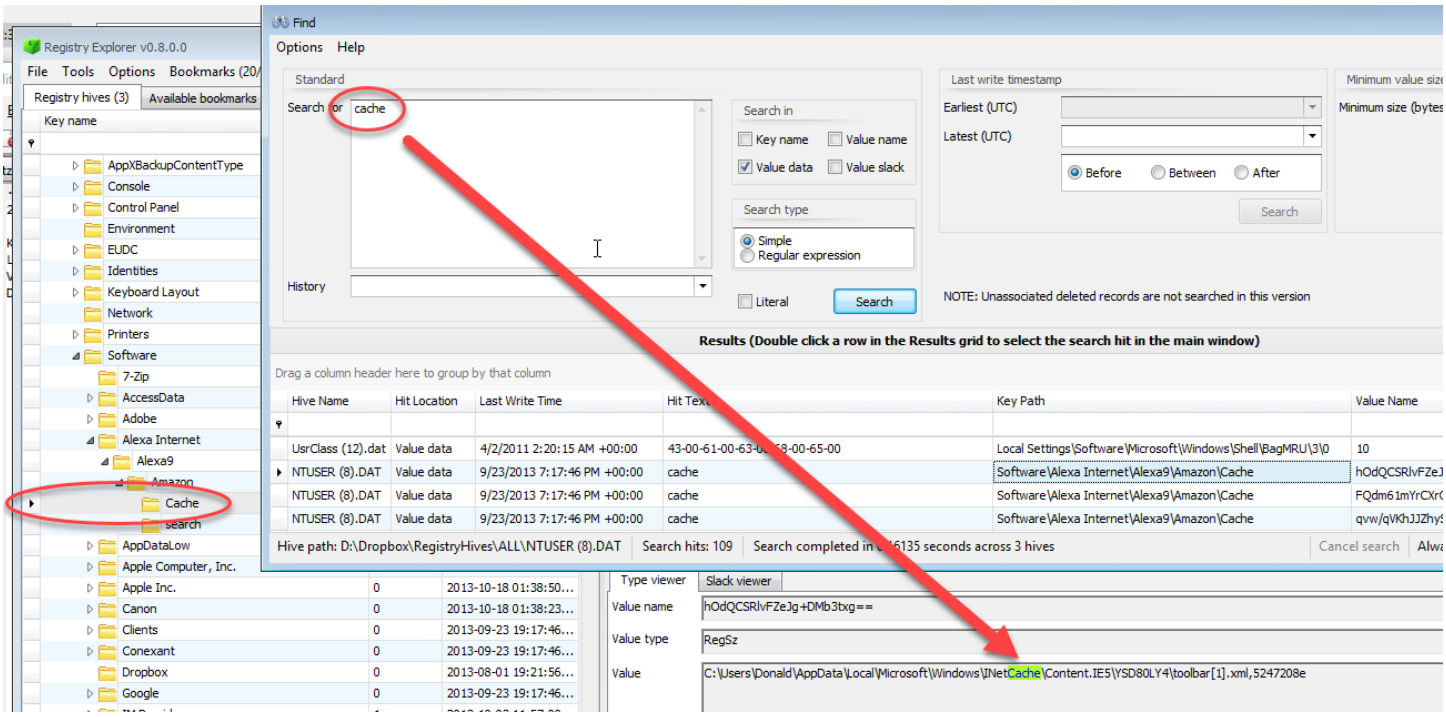
Here is an example where the string 'las' was found in value slack:



In the screen shot above, notice the hit in value slack was found in two different encodings.

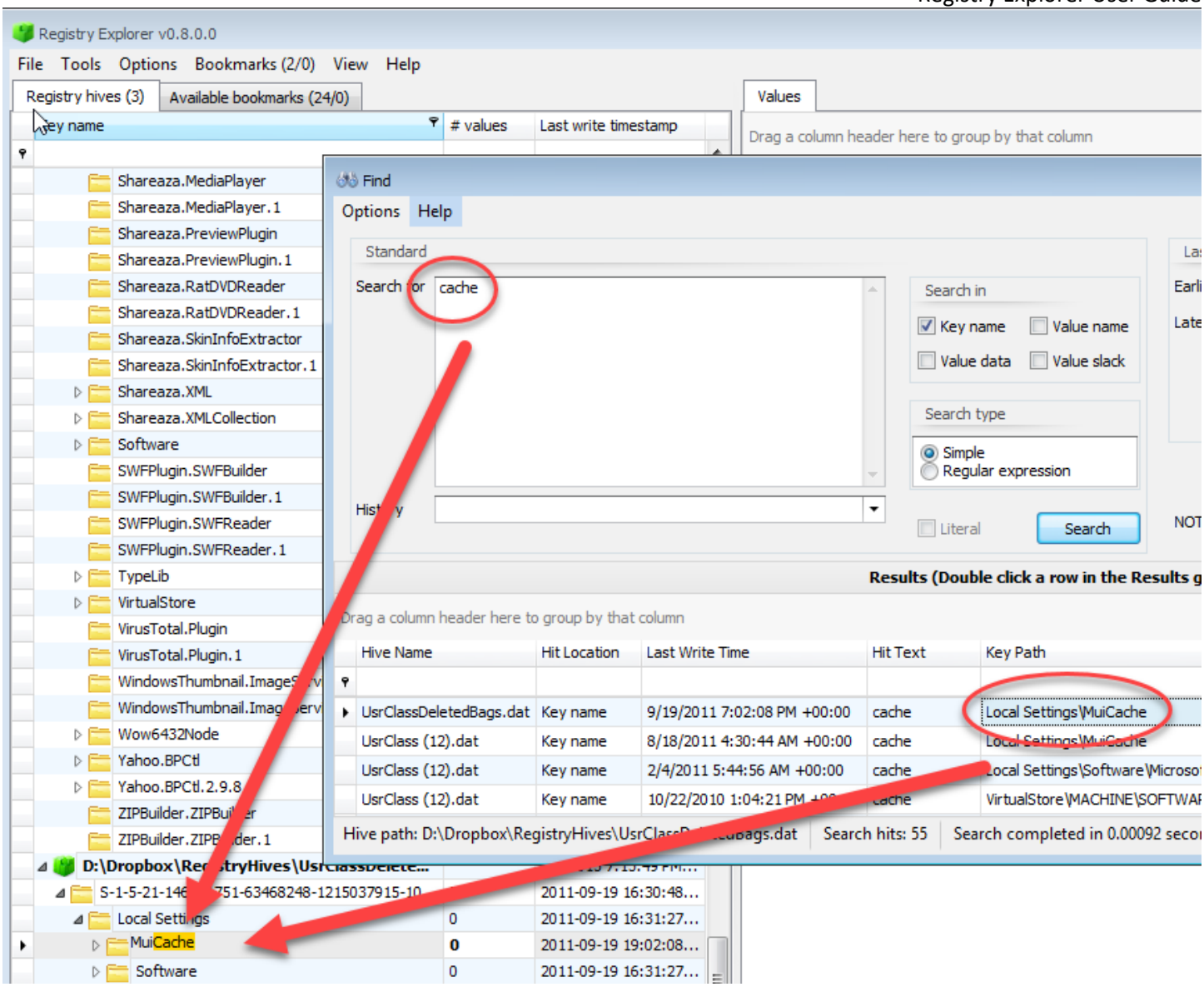
Viewing search results

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.

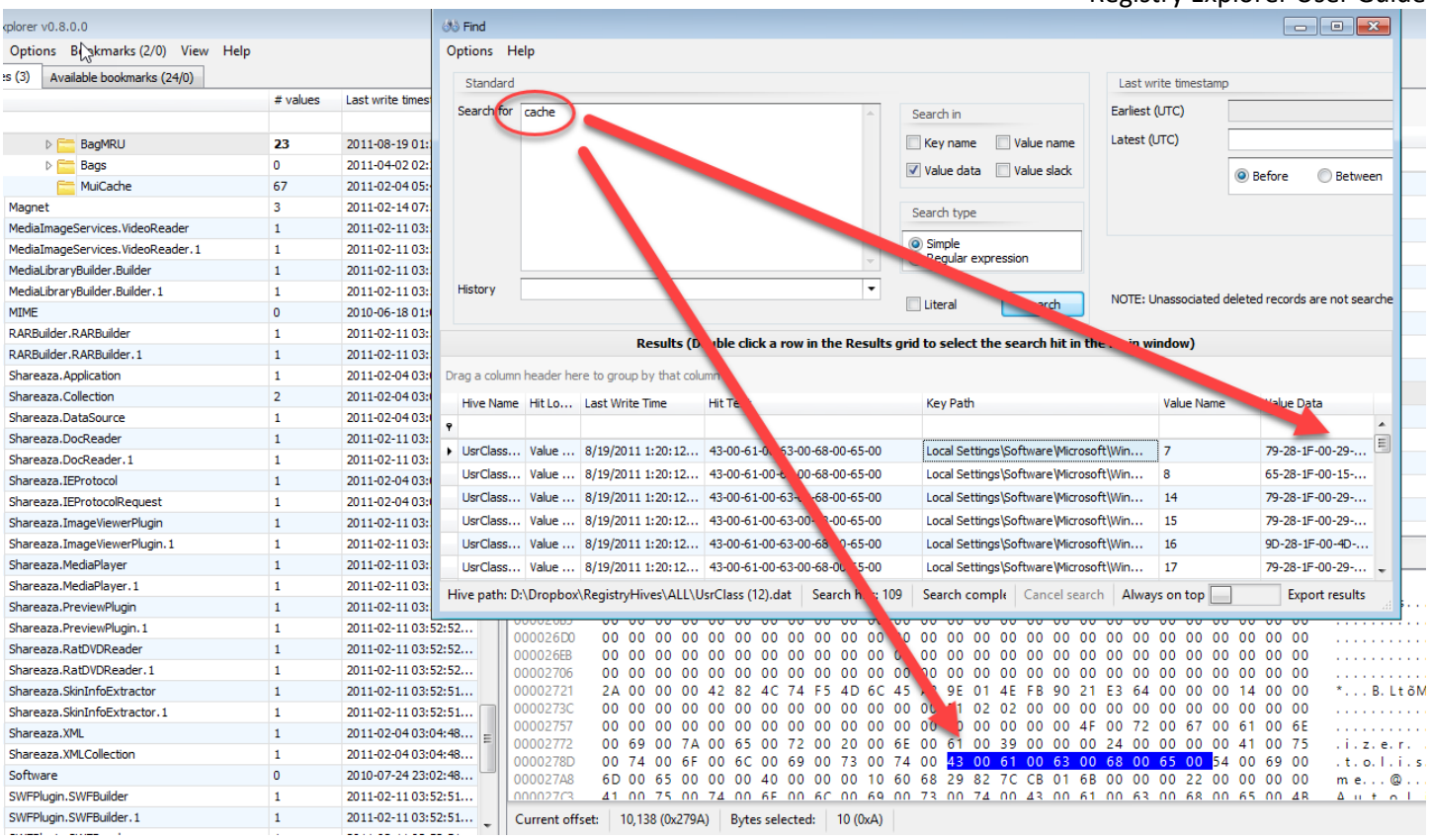


For all simple searches, the search hit will be highlighted (or, in the case of a RegBinary hit, the bytes that make up the hit will be selected). A few more examples of this are shown below.

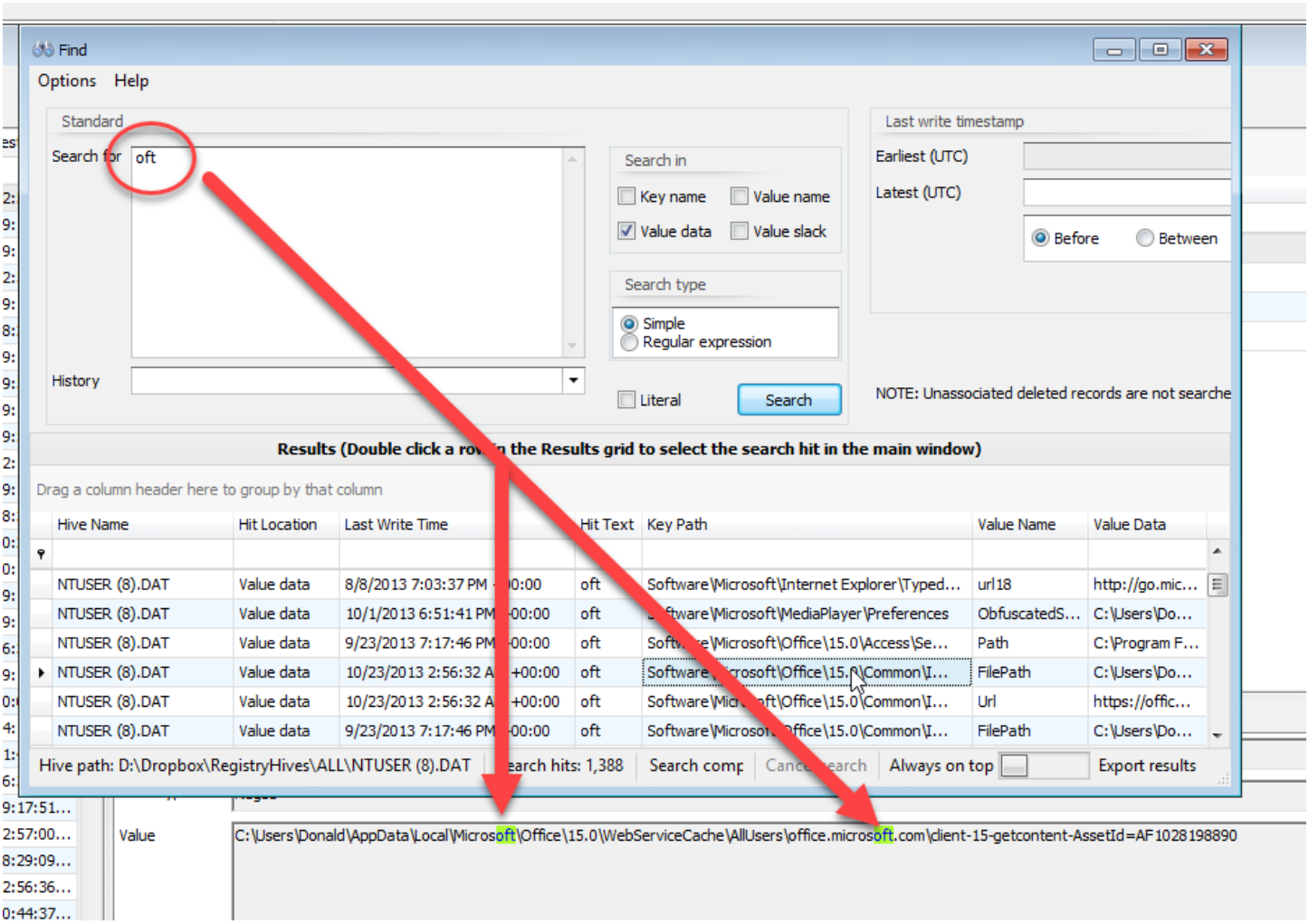
For key name hits, the matching part of the key name is highlighted.



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.


Find
Options Help
Standard
Search for cache
Search in
Key name
Value name
Value data
Value slack
Search type
Simple
Regular expression
History
Literal
Last write timestamp
Earliest (UTC)
Latest (UTC)
Before
Between
NOTE: Unassociated deleted records are not searche
Hive Name
Last Write Time
Key Path
Value Name
Hive path: D:\Dropbox\RegistryHives\ALL\UsrClass (12).dat
Cancel search
Always on top
Export results
Current offset:
10,138 (0x279A)
Bytes selected:
10 (0xA)

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 has to be looked at in order to search for value names or value data/slack across all loaded hives.

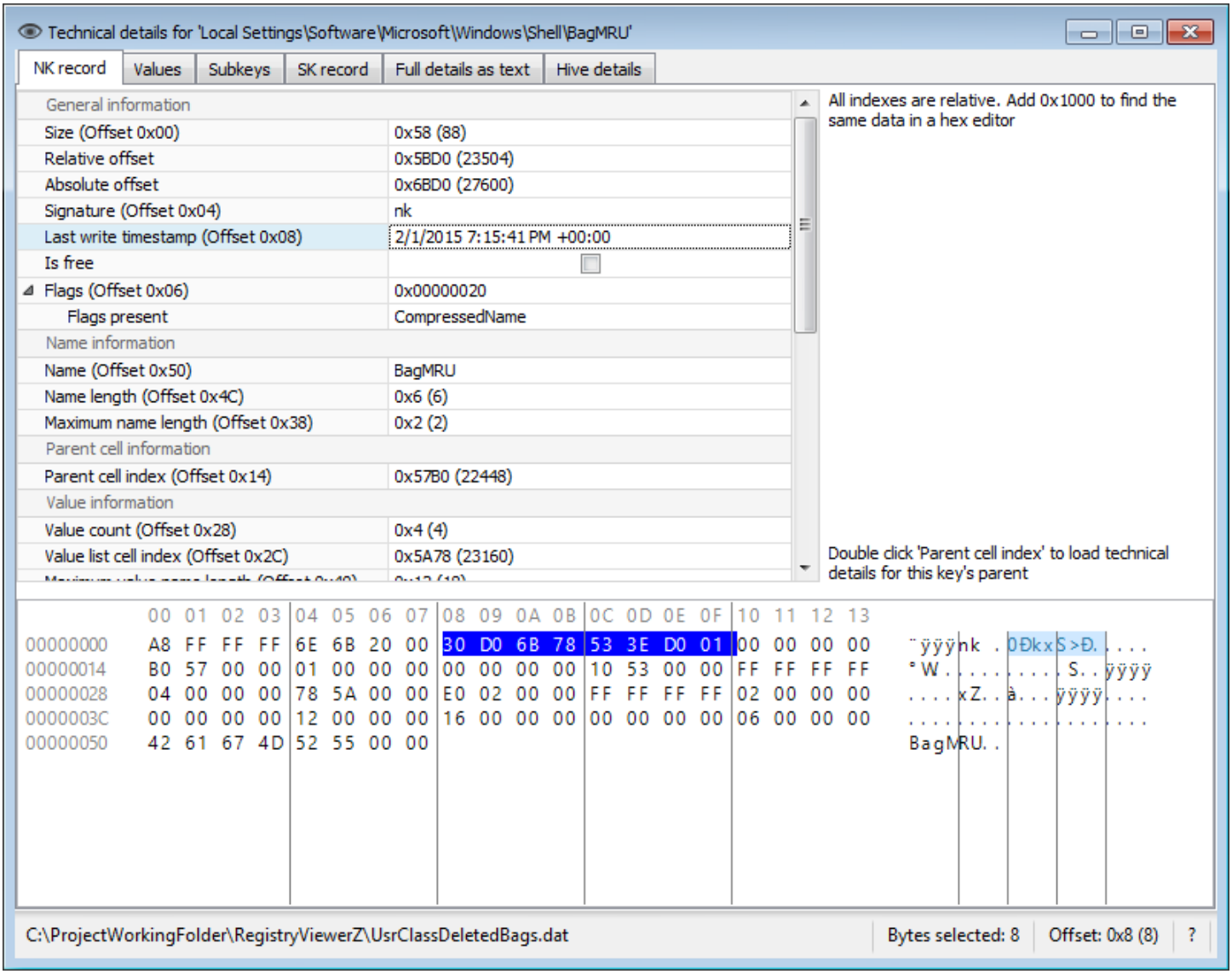
Do not let this stop you from searching against value names and data however. Even with these options selected, Registry Explorer can still search multiple hives very quickly (often under a second), but this depends on the number of keys and values in the loaded hives.

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.

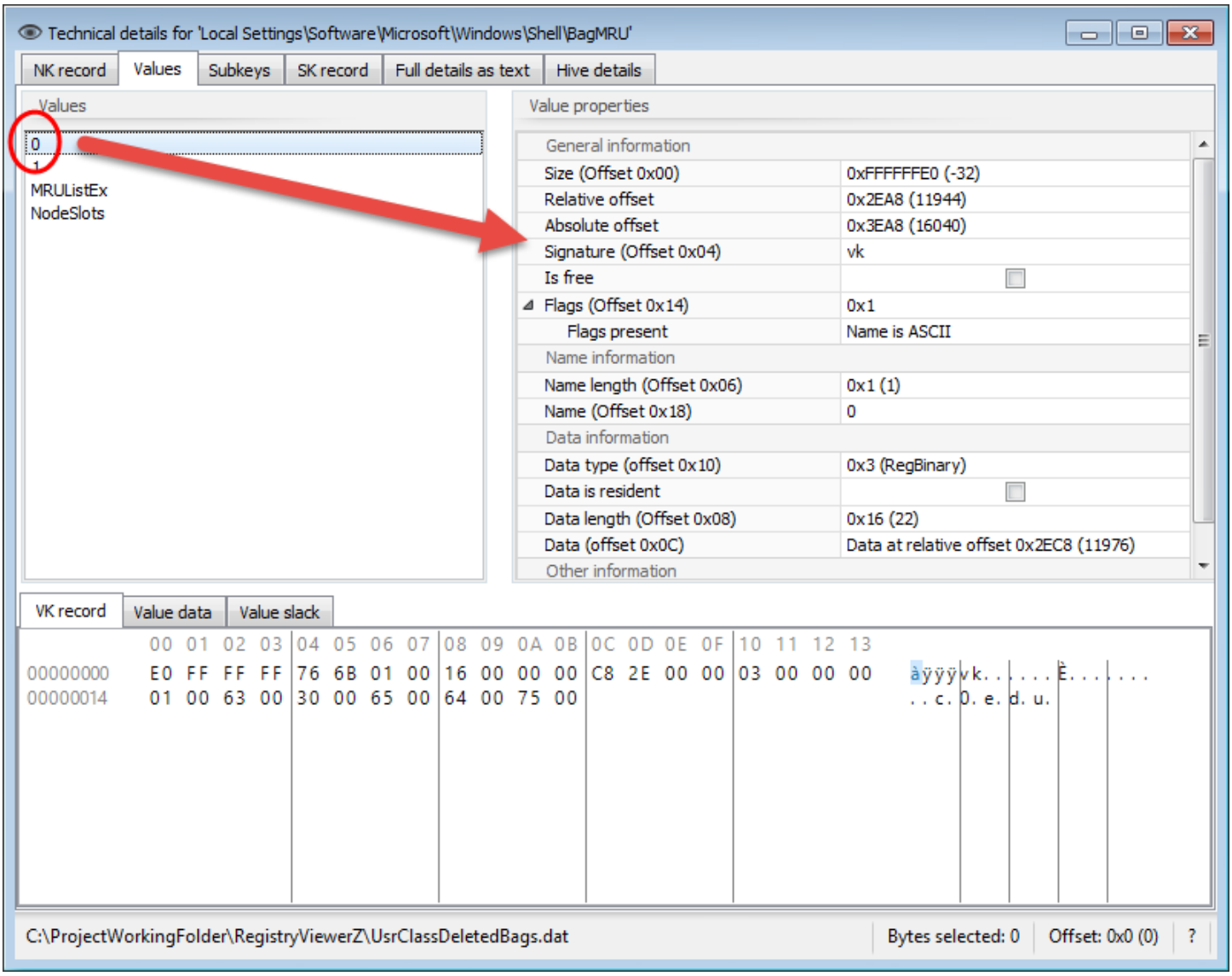


In the example above, the Technical details for the 'Local Settings\Software\Microsoft\Windows\Shell\BagMRU' key are shown. The bytes at the bottom of the details form are the bytes for the NK record as they are found in the Registry hive as viewed in a hex editor.

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.

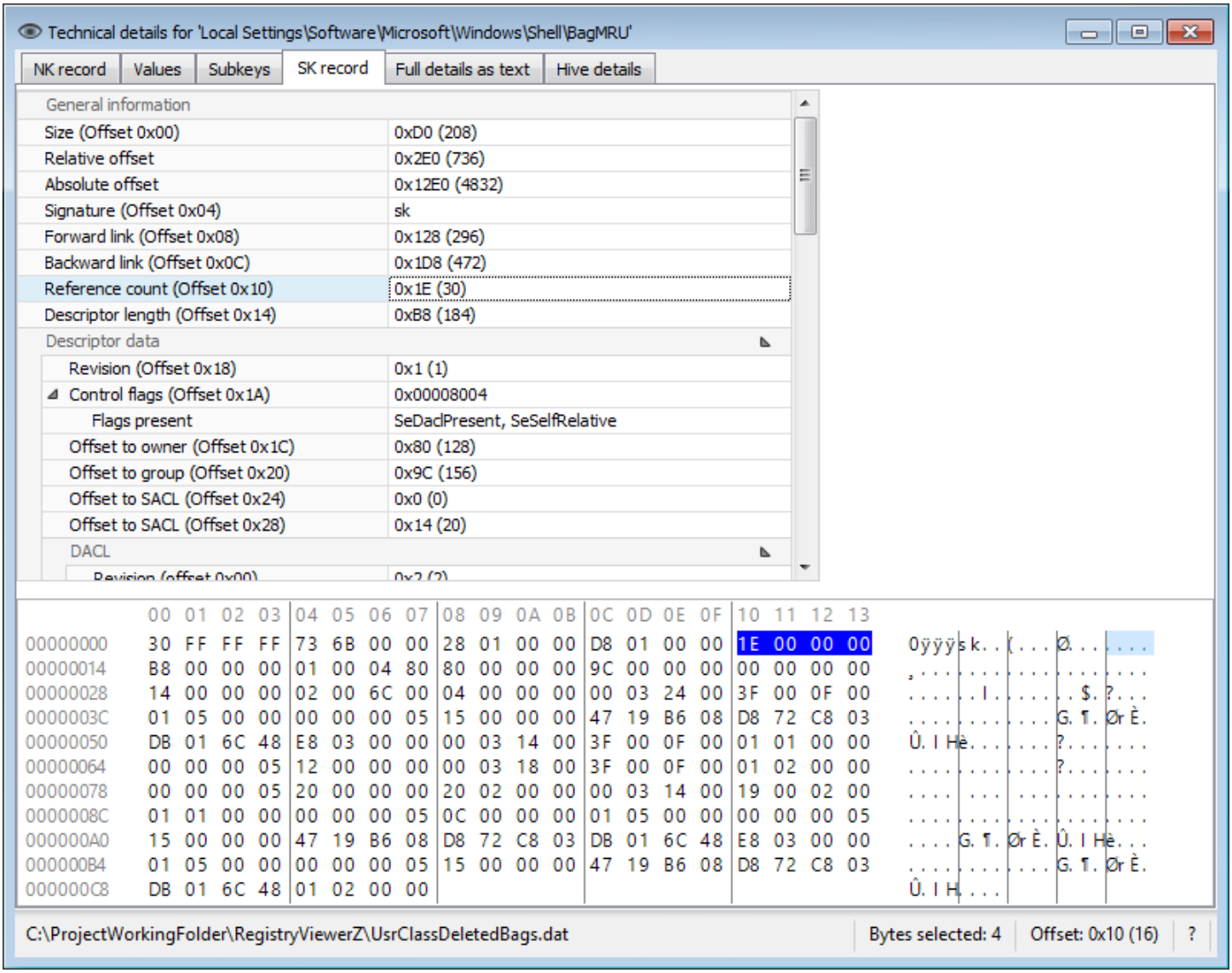
If a key contains one or more values, the Values tab is visible and contains a list of all the key's values. Selecting a value will display the VK record's properties and raw data as we saw with the NK record above.



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.

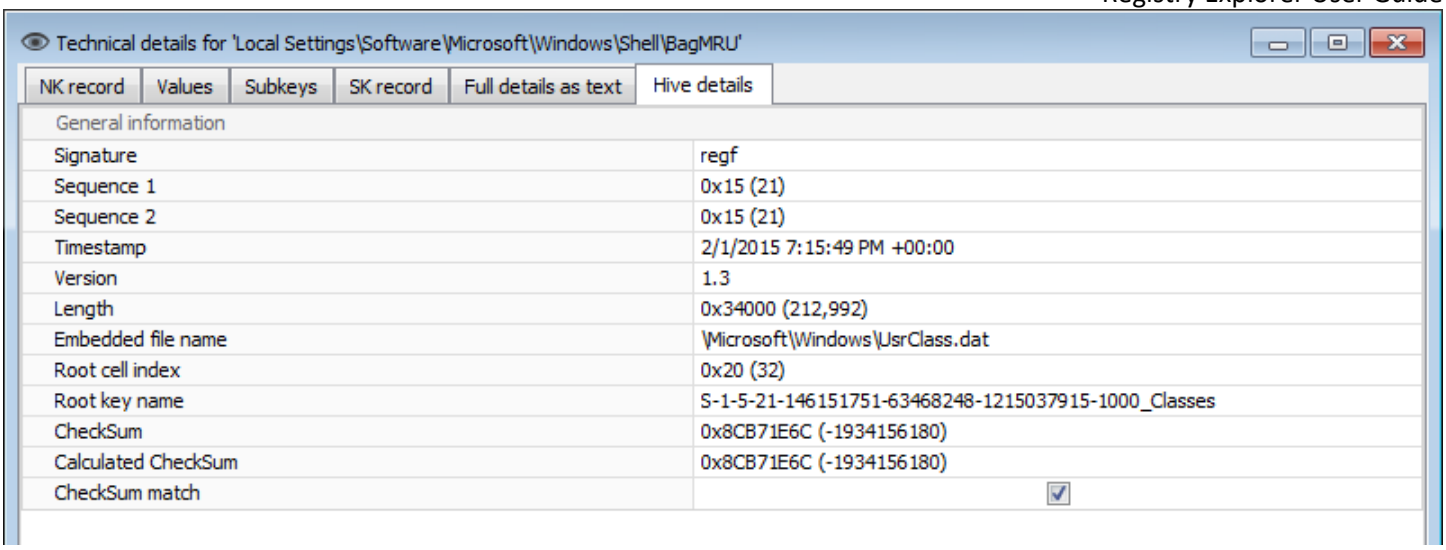
If a key contains subkeys, the Subkeys tab is visible and contains a list of the current key's subkeys. Double clicking on a subkey will open the Technical details report for that key in its own window.

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 '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.



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.

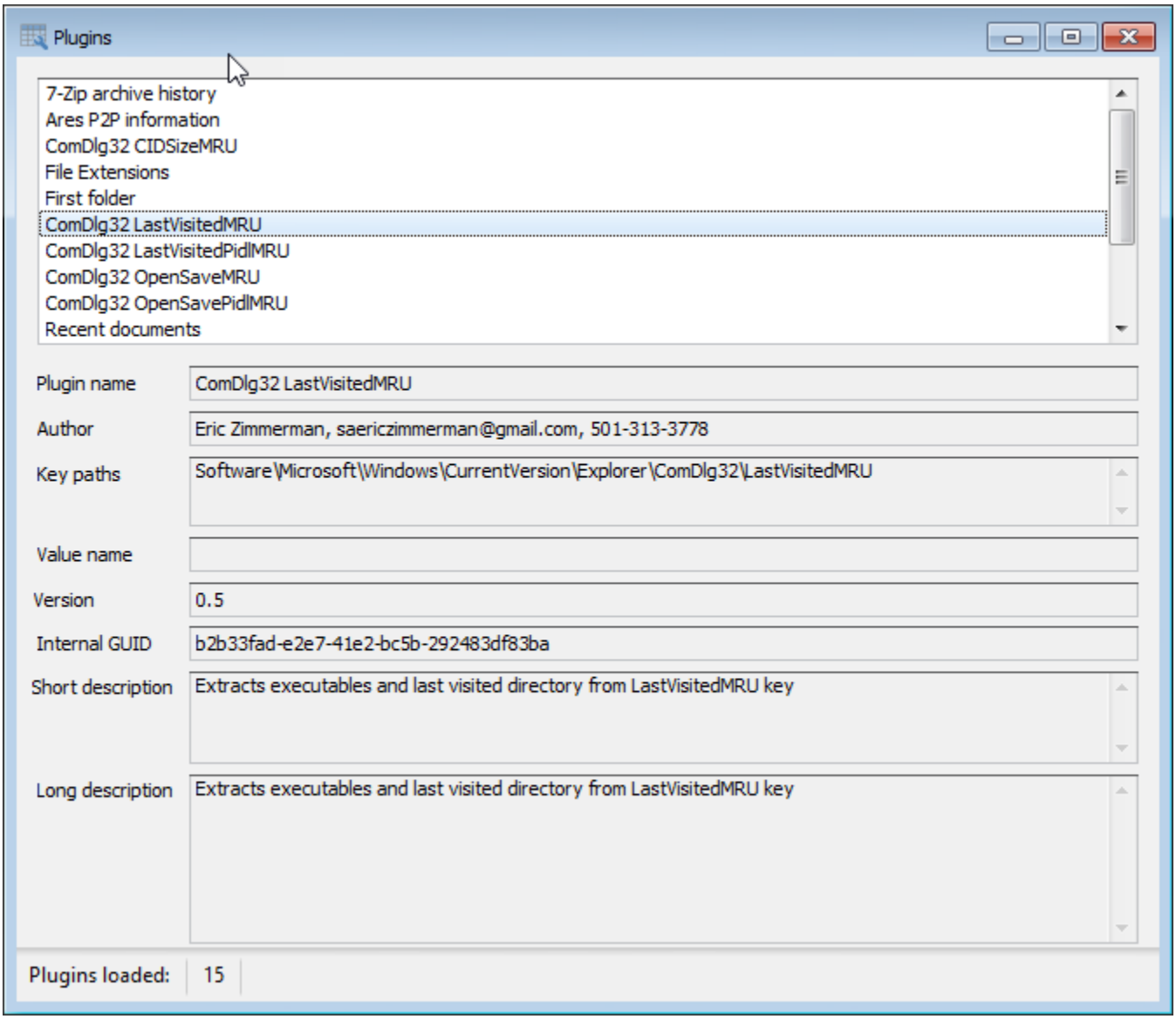
Plugins live under the main Registry Explorer directory in a subdirectory called 'Plugins' which can also contain other directories as needed. Plugins are named according to the format:

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.

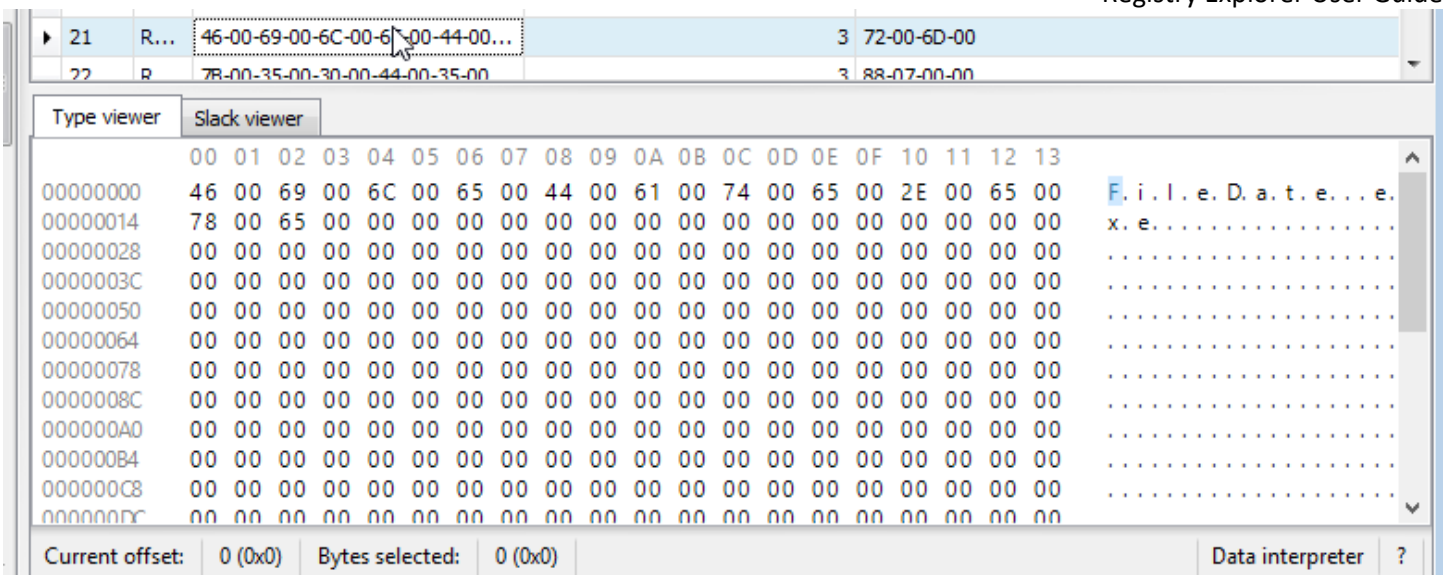
For example, say a user clicks on the *SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\CIDSizeMRU* key. This key's values look like this:

Values

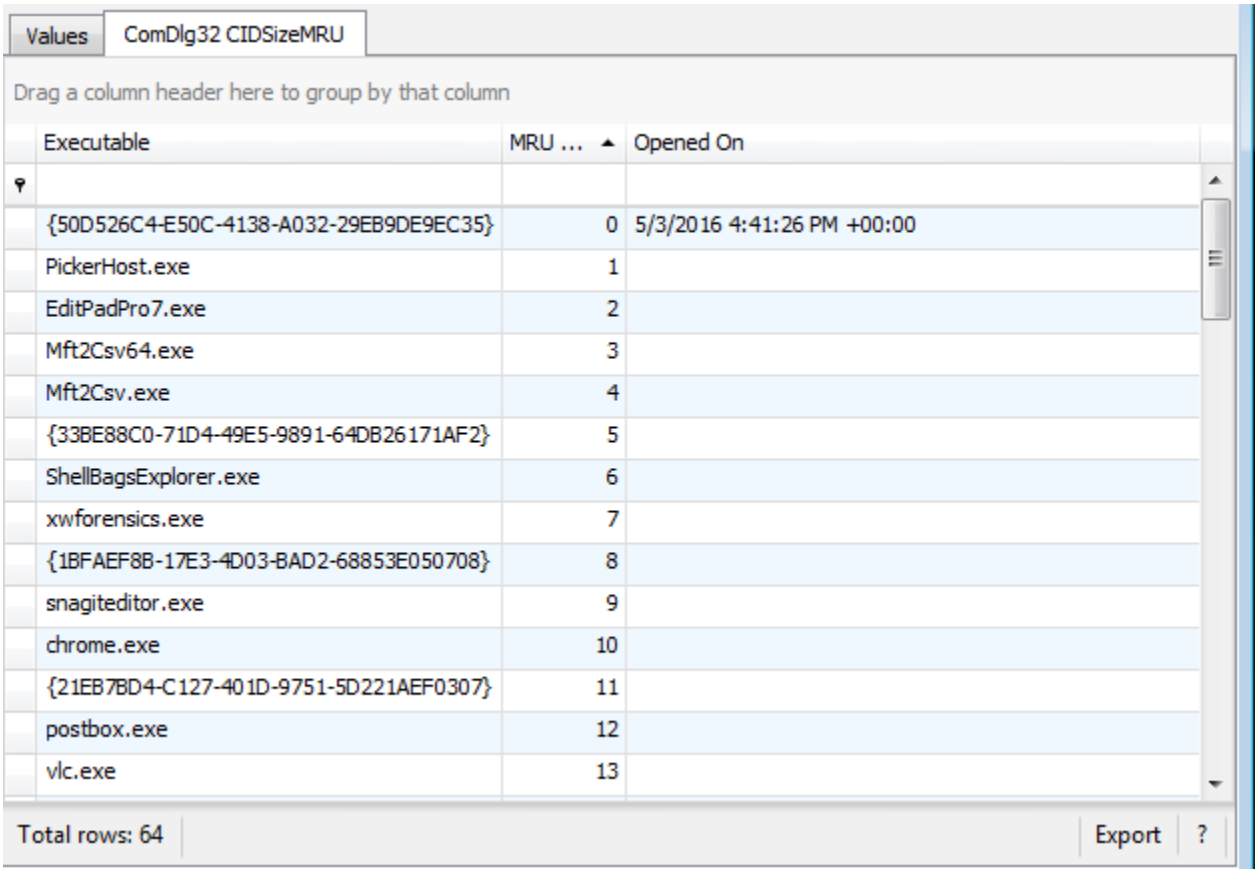
Drag a column header here to group by that column

Value Name	Value Type	Data
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 of the binary data in the hex viewer, like this:



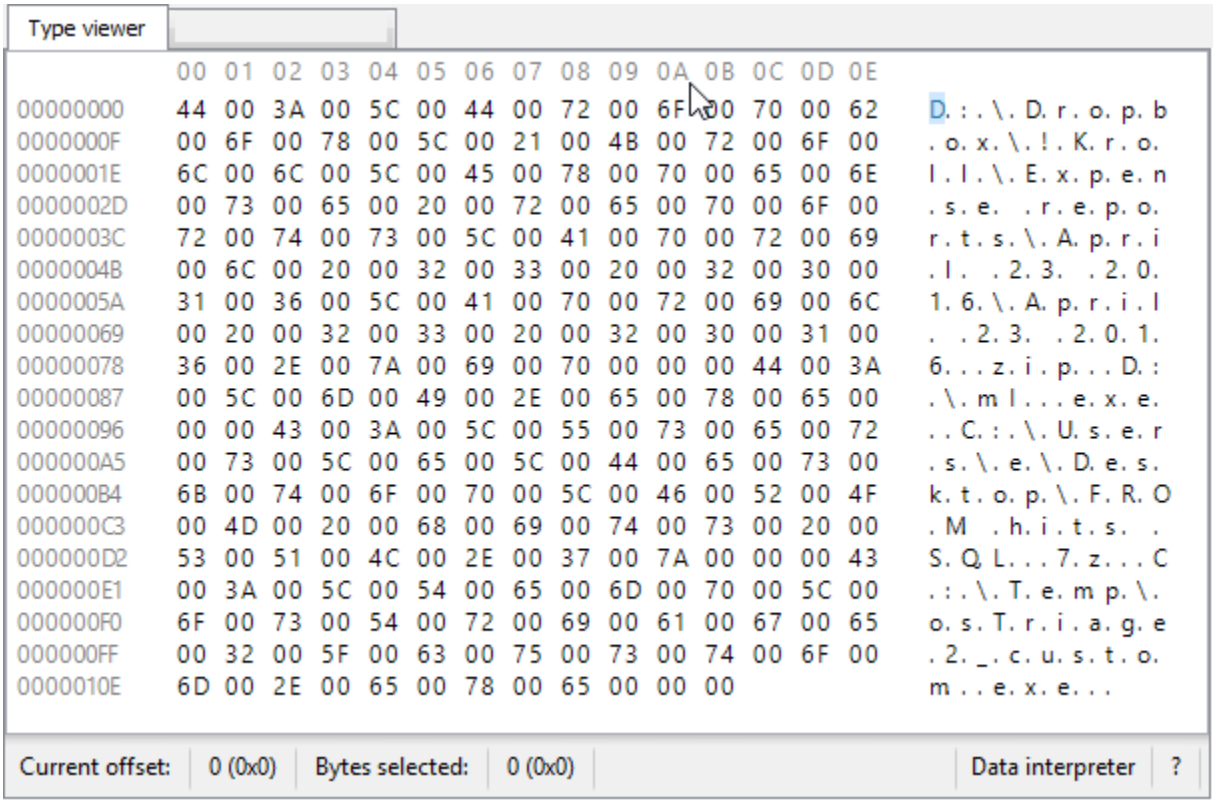
A Unicode string can be seen in the data, but it is difficult to see all of the data at once. This is where the plugin steps in and presents a much easier to use presentation of the values under this key. After a plugin processes a key, a new tab is displayed next to the Values tab at the top of the right side of Registry Explorer:



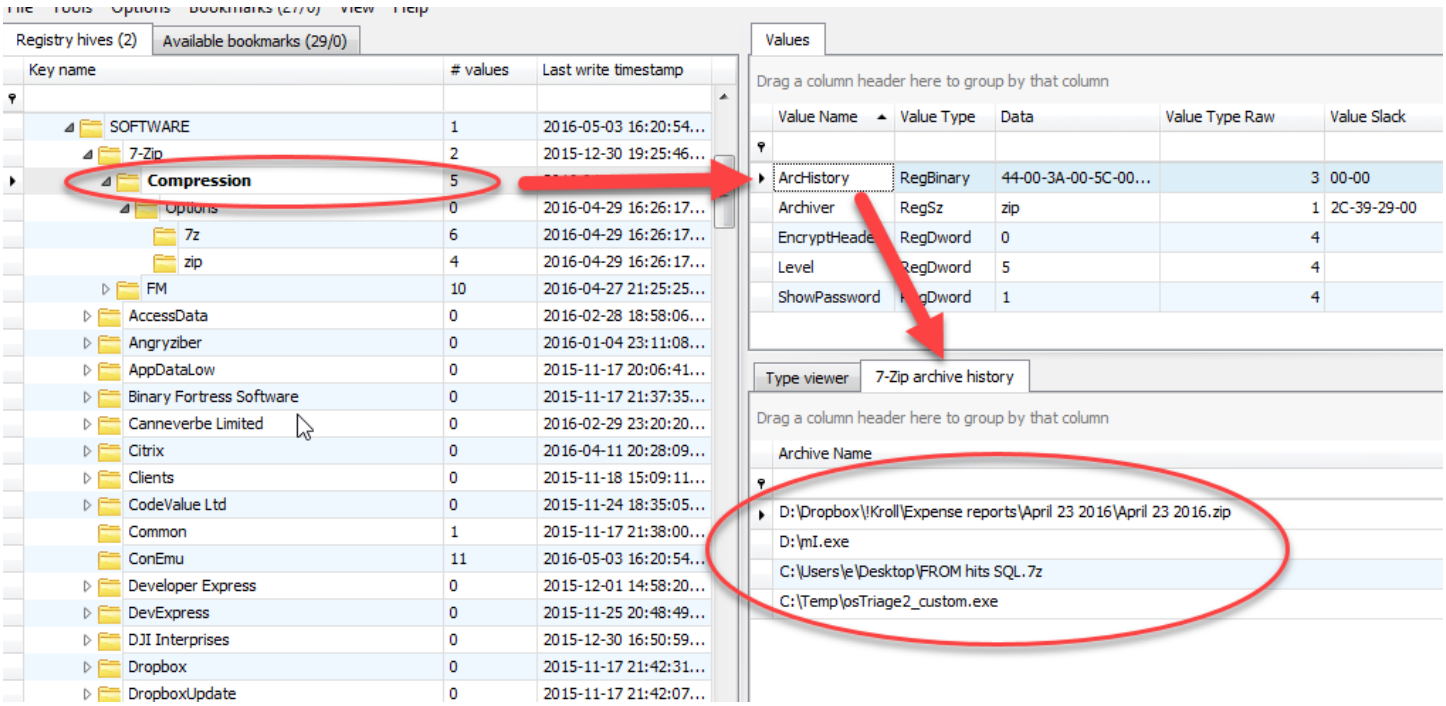
Each value is processed and the results are displayed in a grid which can then be sorted on, filtered, or exported as needed

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.

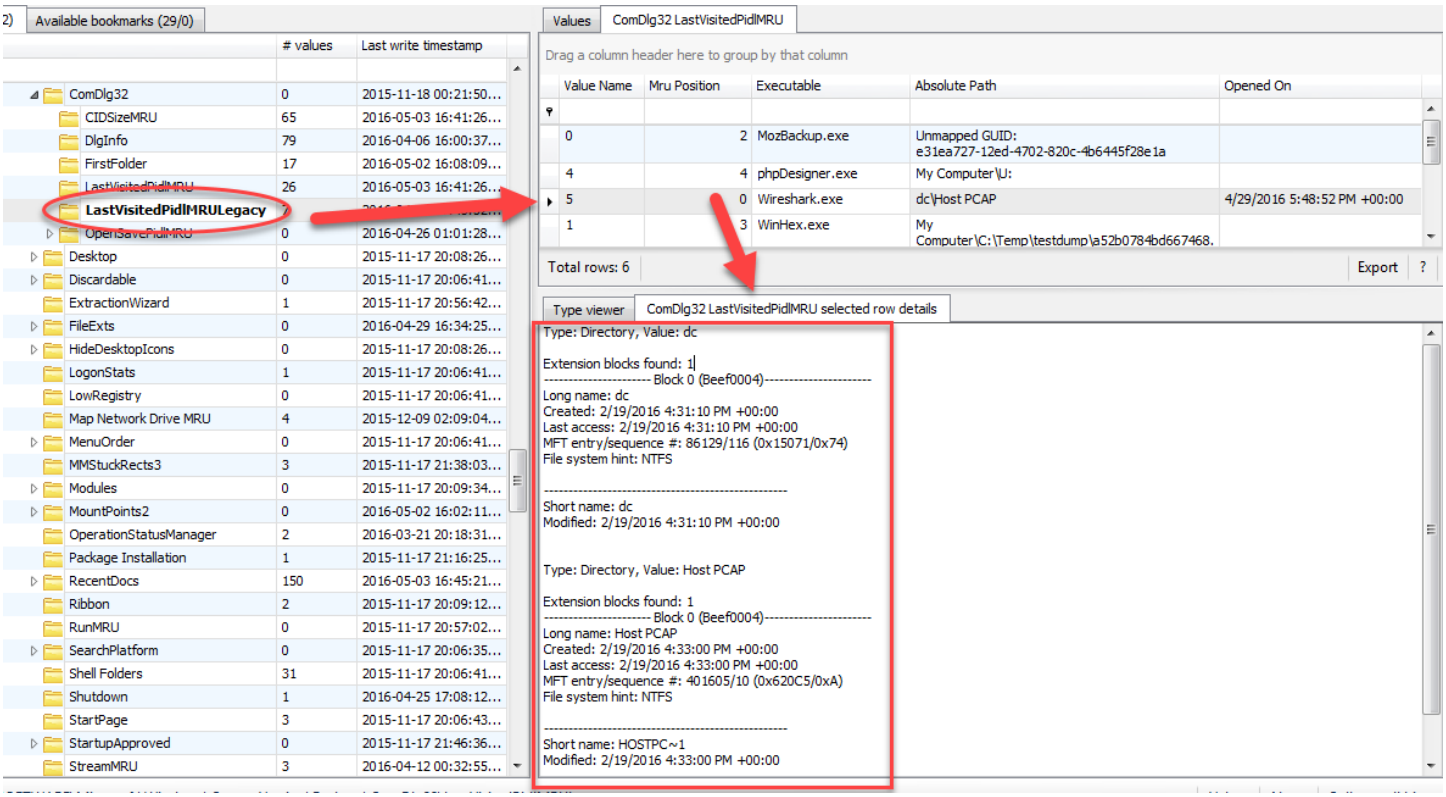


When this value is selected, the 7-Zip plugin processes the value and returns a much nicer list, like this:



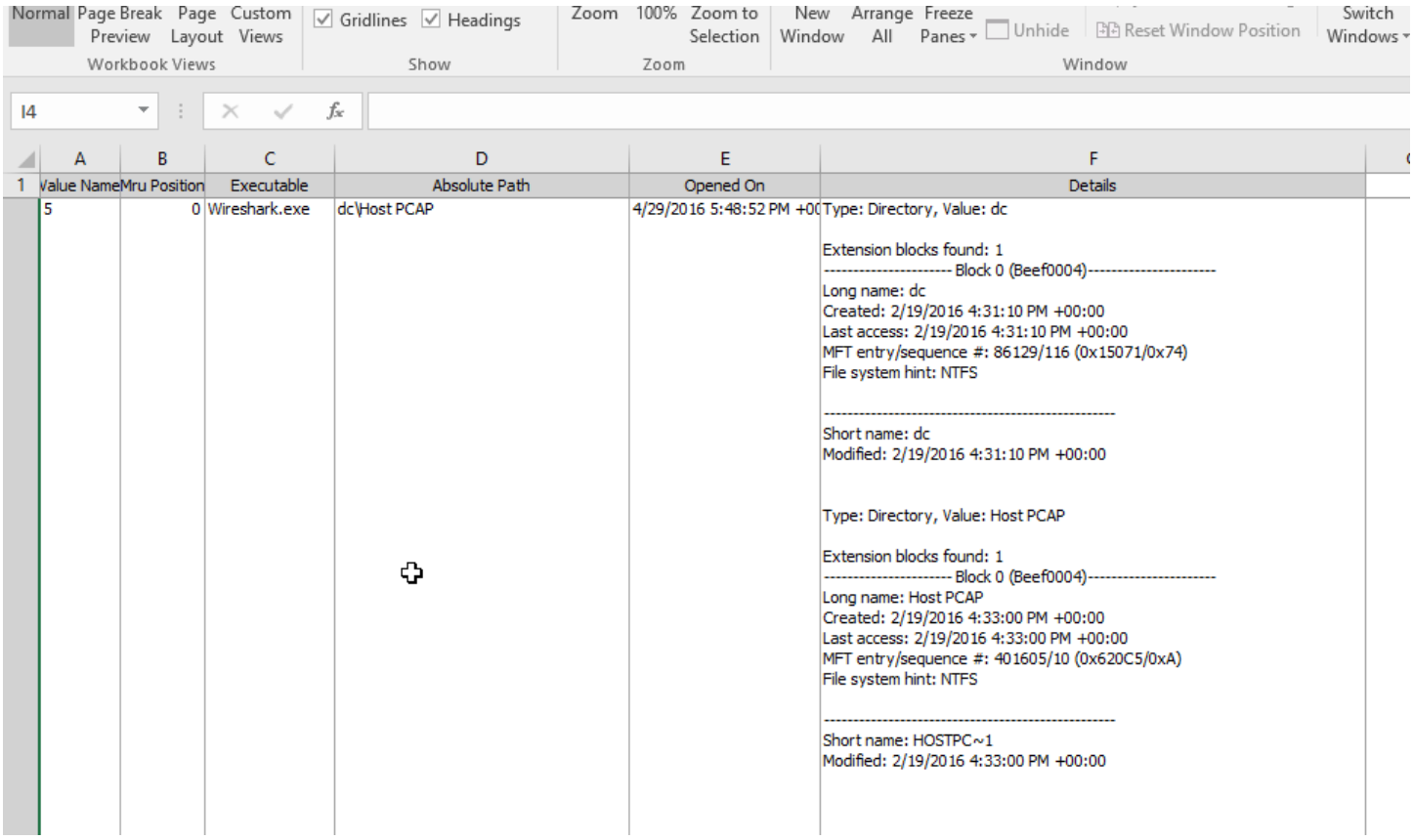
The data returned can now be filtered, sorted, and exported as we saw earlier.

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:



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:



	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	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

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

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   {
        4 references | Eric Zimmerman, 6 days ago | 1 author, 2 changes
3       public class ValuesOut
4       {
            1 reference | Eric Zimmerman, 6 days ago | 1 author, 2 changes
5           public ValuesOut(string archiveName)
6           {
7               ArchiveName = archiveName;
8           }
9
            1 reference | Eric Zimmerman, 7 days ago | 1 author, 1 change
10          public string ArchiveName { get; }
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, lets take a look at the top section of the class:

```
namespace RegistryPlugin._7_ZipHistory
{
    1 reference | Eric Zimmerman, 6 days ago | 1 author, 2 changes
    public class SevenZip : IRegistryPluginGrid
    {
        private readonly BindingList<ValuesOut> _values;

        0 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public SevenZip()
        {
            _values = new BindingList<ValuesOut>();

            Errors = new List<string>();
        }

        16 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public string InternalGuid => "6b1296a2-d3fb-441f-89c1-fd3706855acc";

        16 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public List<string> KeyPaths => new List<string>(new[]
        {
            @"Software\7-Zip\Compression"
        });

        17 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public string ValueName => "ArcHistory";
        34 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public string AlertMessage { get; private set; }
        16 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public RegistryPluginType.PluginType PluginType => RegistryPluginType.PluginType.Grid;
        16 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public string Author => "Eric Zimmerman";
        16 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public string Email => "saericzimmerman@gmail.com";
        16 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public string Phone => "501-313-3778";
        16 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public string PluginName => "7-Zip archive history";

        21 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public string ShortDescription =>
            "Extracts archive history from ArcHistory key"
            ;

        16 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public string LongDescription => ShortDescription;

        16 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        public double Version => 0.5;
        85 references | Eric Zimmerman, 7 days ago | 1 author, 1 change
        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 | EricZimmerman, 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 | EricZimmerman, 317 days ago | 1 author, 2 changes
    RegistryPluginType.PluginType PluginType { get; }
    16 references | EricZimmerman, 317 days ago | 1 author, 1 change
    string Author { get; }
    16 references | EricZimmerman, 317 days ago | 1 author, 1 change
    string Email { get; }
    16 references | EricZimmerman, 317 days ago | 1 author, 1 change
    string Phone { get; }
    16 references | EricZimmerman, 317 days ago | 1 author, 1 change
    string PluginName { get; }
    21 references | EricZimmerman, 317 days ago | 1 author, 1 change
    string ShortDescription { get; }
    16 references | EricZimmerman, 317 days ago | 1 author, 1 change
    string LongDescription { get; }
    16 references | EricZimmerman, 317 days ago | 1 author, 1 change
    double Version { get; }
    85 references | EricZimmerman, 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";
    }
}


7 references | Eric Zimmerman, 7 days ago | 1 author, 2 changes
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.

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.

```
λ .\RECmd.exe

RECmd version 0.7.0.0

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

Note: Enclose all strings containing spaces (and all RegEx) with double quotes

        Hive                    Hive to search.
        Literal                 If present, --sd and --ss search value will not be interpreted as ASCII or Unicode byte strings
        Recover                 If present, recover deleted keys/values
        Recursive               Dump keys/values recursively (ignored if --ValueName used). This option provides FULL details about keys and values
        RegEx                   If present, treat <string> in --sk, --sv, --sd, and --ss as a regular expression
        Sort                    If present, sort the output
        SuppressData            If present, do not show data when using --sd or --ss

        KeyName                 Key name. All values under this key will be dumped
        ValueName               Value name. Only this value will be dumped
        SaveToName              Saves ValueName value data in binary form to file

        StartDate               Start date to look for last write timestamps (UTC). If EndDate is not supplied, last writes AFTER this date will be returned
        EndDate                 End date to look for last write timestamps (UTC). If StartDate is not supplied, last writes BEFORE this date will be returned
        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

Example: RECmd.exe --Hive "C:\Temp\UsrClass 1.dat" --sk URL --Recover
         RECmd.exe --Hive "D:\temp\UsrClass 1.dat" --StartDate "11/13/2014 15:35:01"
         RECmd.exe --Hive "D:\temp\UsrClass 1.dat" --RegEx --sv "(App|Display)Name"
         RECmd.exe --Hive "D:\temp\UsrClass 1.dat" --StartDate "05/20/2014 19:00:00" --EndDate "05/20/2014 23:59:59"
         RECmd.exe --Hive "D:\temp\UsrClass 1.dat" --StartDate "05/20/2014 07:00:00 AM" --EndDate "05/20/2014 07:59:59 PM"
```

There are three groups of command line options for RECmd.

General

This category includes the Hive and Recover switches. Hive is always required.

- **Hive**: The full path to the hive to process. If the path contains spaces, include them in double quotes.
- **Literal**: If present, the --sd and --ss search value is not interpreted as ASCII and Unicode strings
- **Recover**: If present, recover deleted keys and values
- **Recursive**: If present, dump keys and values recursively from the key specified by KeyName. This option provides much more detail about keys and values.
- **RegEx**: If present, treat <string> in --sk, --sv, --sd, and –ss as a regular expression
- **Sort**: If present, sort the output in a meaningful way based on the type of data requested.

- **SuppressData**: If present, do not include the contents of value data when using the sd or nd switches. This is useful for seeing the key paths and value names without the potential noise of the value data itself.

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)".

- **KeyName**: The key name to look for. If used without ValueName, displays all subkeys and values.
- **ValueName**: Display only the value specified
- **SaveToName**: Saves ValueName value data in binary form to a file

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).

- **StartDate**: The earliest date to look for in Registry key last write timestamps. Timestamp should be in UTC.
- **EndDate**: The latest date to look for in Registry key last write timestamps. Timestamp should be in UTC.
- **MinSize**: Find values with value data size 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 also 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 also searched/compared to <string> unless the --Literal switch is used

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).

```
D:\temp\RegistryExplorer\RECmd
λ .\RECmd.exe --Hive "D:\temp\re\ALL\UsrClassDeletedBags.dat" --sd "04-00-EF-BE"
RECmd version 0.7.0.0

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\re\ALL\UsrClassDeletedBags.dat'

Initial processing complete. Building tree...
Found root node! Getting subkeys...
Hive processing complete!
Flushing record lists...

Root key name: S-1-5-21-146151751-63468248-1215037915-1000_Classes

Key: Local Settings\Software\Microsoft\Windows\Shell\BagMRU\0\0, Value: 0, Data: 74-00-31-
0-00-00-00-00-33-3F-D9-83-11-00-55-73-65-72-73-00-60-00-08-00-04-00-EF-BE-EE-3A-85-1A-33-3
-D9-83-2A-00-00-00-B5-01-00-00-00-00-01-00-00-00-00-00-00-00-00-00-36-00-00-00-00-00-55-00
73-00-65-00-72-00-73-00-00-00-40-00-73-00-68-00-65-00-6C-00-6C-00-33-00-32-00-2E-00-64-00-
C-00-6C-00-2C-00-2D-00-32-00-31-00-38-00-31-00-33-00-00-00-14-00-00-00

Found 1 value data hit

Search took 0.050 seconds
```

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 Ask 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 RECmd 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:

RECmd.exe --Hive "D:\temp\re\UsrClass 1.dat" --RegEx --sk "Microsoft.Bing[FHW]o"

```
λ .\RECmd.exe --Hive "D:\temp\re\ALL\UsrClass 1.dat" --RegEx --sk "Microsoft.Bing[FHW]o"
RECmd version 0.7.0.0

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\re\ALL\UsrClass 1.dat'

Initial processing complete. Building tree...
Found root node! Getting subkeys...
Hive processing complete!
Flushing record lists...

Root key name: S-1-5-21-2417227394-2575385136-2411922467-1105_Classes

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_8wekyb3d8bbwe
Key: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Families\Microsoft.BingFoodAndDrink_8wekyb3d8bbwe
Key: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Families\Microsoft.BingFoodAndDrink_8wekyb3d8bbwe\Microsoft.Bi
ngFoodAndDrink_3.0.4.212_x64__8wekyb3d8bbwe
Key: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Packages\Microsoft.BingFoodAndDrink_3.0.4.212_x64__8wekyb3d8bb
we
Key: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\Repository\Packages\Microsoft.BingFoodAndDrink_3.0.4.212_x64__8wekyb3d8bb
we\Applications\Microsoft.BingFoodAndDrink_8wekyb3d8bbwe!AppexFoodAndDrink
Key: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\SystemAppData\Microsoft.BingFoodAndDrink_8wekyb3d8bbwe
Key: Local Settings\Software\Microsoft\Windows\CurrentVersion\AppModel\SystemAppData\Microsoft.BingFoodAndDrink_8wekyb3d8bbwe\SplashScreen\Micro
soft.BingFoodAndDrink_8wekyb3d8bbwe!AppexFoodAndDrink

Found 11 keys (via RegEx)

Search took 0.369 seconds
```

Finding values

To find all values with names that contain either 'AppName' or 'DisplayName', use the following search:

RECmd.exe --Hive "D:\temp\re\UsrClass 1.dat" --RegEx --sv "(App|Display)Name"

320 results were found in 0.380 seconds, but due to the length of the output, only the first few are shown below.

```
λ .\RECmd.exe --Hive "D:\temp\re\ALL\UsrClass 1.dat" --RegEx --sv "(App|Display)Name"
RECmd version 0.7.0.0

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\re\ALL\UsrClass 1.dat'

Initial processing complete. Building tree...
Found root node! Getting subkeys...
Hive processing complete!
Flushing record lists...

Root key name: S-1-5-21-2417227394-2575385136-2411922467-1105_Classes

Key: Extensions\ContractId\Windows.AppointmentsProvider.AddAppointment\PackageId\microsoft.windowscommunicationsapps_17.5.9600.
3d8bbwe\ActivatableClassId\Microsoft.WindowsLive.Calendar.AppX8aw8zprmge6rpnpcf2pbb5xb4tm2kwee.wwa, Value: DisplayName
Key: Extensions\ContractId\Windows.AppointmentsProvider.RemoveAppointment\PackageId\microsoft.windowscommunicationsapps_17.5.96
kyb3d8bbwe\ActivatableClassId\Microsoft.WindowsLive.Calendar.AppXmdsmbnj96vbbsrtjy28k3k47cv542vte.wwa, Value: DisplayName
Key: Extensions\ContractId\Windows.AppointmentsProvider.ReplaceAppointment\PackageId\microsoft.windowscommunicationsapps_17.5.9
ekyb3d8bbwe\ActivatableClassId\Microsoft.WindowsLive.Calendar.AppXkaqyc69sy6we8d3wgez35dkc33h4dhmw.wwa, Value: DisplayName
Key: Extensions\ContractId\Windows.AppointmentsProvider.ShowTimeFrame\PackageId\microsoft.windowscommunicationsapps_17.5.9600.2
d8bbwe\ActivatableClassId\Microsoft.WindowsLive.Calendar.AppXtp1qs1d9f5y8jbjjmhsq3cg4kqg3fsqr.wwa, Value: DisplayName
Key: Extensions\ContractId\Windows.BackgroundTasks\PackageId\Microsoft.BingFinance_3.0.4.212_x64__8wekyb3d8bbwe\ActivatableClas
AppXt2b0qt8jwqketvnyx02s765gyw55jaq6.mca, Value: DisplayName
Key: Extensions\ContractId\Windows.BackgroundTasks\PackageId\Microsoft.BingFinance_3.0.4.212_x64__8wekyb3d8bbwe\ActivatableClas
rking.BackgroundTransfer.Internal.BackgroundTransferTask.ClassId.1, Value: DisplayName
Key: Extensions\ContractId\Windows.BackgroundTasks\PackageId\Microsoft.BingFinance_3.0.4.212_x64__8wekyb3d8bbwe\ActivatableClas
rking.BackgroundTransfer.Internal.NetworkChangeTask.ClassId.1, Value: DisplayName
Key: Extensions\ContractId\Windows.BackgroundTasks\PackageId\Microsoft.BingFinance_3.0.4.212_x64__8wekyb3d8bbwe\ActivatableClas
rking.ContentPrefetcher.Internal.ContentPrefetcherTask.ClassId.1, Value: DisplayName
Key: Extensions\ContractId\Windows.BackgroundTasks\PackageId\Microsoft.BingFoodAndDrink_3.0.4.212_x64__8wekyb3d8bbwe\Activatabl
dAndDrink.AppX63c7c5was2w5nzjj8e95fkzrz35wbngh.mca, Value: DisplayName
```

Finding data

To find all values whose data contains ‘URL:bing’ followed by either an m, h, or s, use the following search:

RECmd.exe --Hive "D:\temp\re\UsrClass 1.dat" --RegEx --sd "URL:bing[mhs]"

```
λ .\RECmd.exe --Hive "D:\temp\re\ALL\UsrClass 1.dat" --RegEx --sd "URL:bing[mhs]"
RECmd version 0.7.0.0

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\re\ALL\UsrClass 1.dat'

Initial processing complete. Building tree...
Found root node! Getting subkeys...
Hive processing complete!
Flushing record lists...

Root key name: S-1-5-21-2417227394-2575385136-2411922467-1105_Classes

Key: binghealthnfitness, Value: (default), Data: URL:binghealthnfitness
Key: bingmaps, Value: (default), Data: URL:bingmaps
Key: bingsports, Value: (default), Data: URL:bingsports

Found 3 value data hits (via RegEx)

Search took 0.494 seconds
```

For more examples, run RECmd.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.

Version changes

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 (SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList), 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

- SA/FE Devon P. Ackerman devon.ackerman@ic.fbi.gov, sadevonackerman@gmail.com
- David Cowen @HECFBlog
- Dan Pullega @4n6k
- Jerod Alexander @jerod
- Willi Ballenthin @williballenthin

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