VirusDetective® Help
Search Strings Dialog Box

TABLE OF CONTENTS…

I) PREFACE… DEFINITIONS…

Before you get into this Help file in any depth, a couple of definitions and explanations are in order:

DEFINITION: “Hexually-transmitted disease” (HTD for short): any virus, Trojan Horse or worm; a file that has an HTD is an “infected” file.

DEFINITION: “A search string has matched the contents of a file”: this is technical jargon for when VirusDetective has found an HTD (virus, Trojan Horse, or worm). In technical terms, a file on the Macintosh is made up of either a resource part (called the resource fork) or a data part (called the data fork), or both. (NOTE: the majority of viruses infect the resource fork.) As a non-technical VirusDetective user, you need NOT know about resource forks and data forks.

Whenever you see something like “A search string matched the resource part of a file or data part of a file” or simply “A search string matched a resource”, know that VirusDetective has found an infected file. It is the same as saying “A search string found an infected file”.

In the event that a search string matches a file but it turns out NOT to actually be an HTD, it is a “false alarm” (a “ false positive”). Statistically, false positives cannot be prevented – they happen occasionally no matter what you do to prevent them.

A) CURRENT SEARCH STRING LIST (IN THE UPPER SCROLLABLE WINDOW)…

This is the list of search strings as they exist today. You can keep the search string list the same as when you originally got the VirusDetective desk accessory, you can add search strings, and you can delete search strings. When you want to copy a search string onto the Clipboard, highlight the desired search string, making sure nothing is highlighted in the Search String Entry box, then select Copy from the menu bar, or press Command-C.

B) SEARCH STRING ENTRY BOX…

The Search String Entry box is the long box and it is where you add search strings. Copy, cut and paste works in this box via Command-C, Command-X and Command-V, respectively.

Blank spaces may be used between search string parts to improve readability. A search string must be 255 characters or less.

C) “ADD”, “REMOVE”, “READ FROM FILE”, AND “WRITE FROM FILE” BUTTONS…

C.1 “Add” button…
When you click the Add button, VirusDetective adds what is in the Search String Entry box into the list of search strings. (Be aware that the Add button has nothing to do with the Clipboard.) You are notified of any errors in the search string after you click the Add button.

C.2 “Remove” button…
When you click the the Remove button, VirusDetective removes the search string that is highlighted. (Be aware that the Remove button has nothing to do with the Clipboard.)

C.3 “Read from File” and “Append to List” button…

C.3.1 “Read from File” button…
When you click the Read from File button, VirusDetective presents a directory dialog box that lets you read a text file (we repeat, TEXT file) of search strings, one per line, into the slot where the search strings belong REPLACING the entire current search string list.

It is the Read from File button you would use to read either the “VD Search Strings” or “Alternate Search Strings” files into VirusDetective. We have provided two files, “VD Search Strings” and “Alternate Search Strings” both of which are OPTIONAL. VirusDetective comes to you with its search string list stored ready to go.

The file “VD Search Strings” is an exact copy of the search string list that comes built-in to VirusDetective. This list is designed to check files for all known active (“INFECTIOUS”) viruses in a SPEEDY manner. We have put them into a separate text file just in case you accidentally remove or change a search string, and you want to input the original list into VirusDetective once again.

The file “Alternate Search Strings” contains the same search string list as the built-in list EXCEPT that the SPEED ENHANCEMENTS that are in the built-in list HAVE BEEN REMOVED. The “Alternate Search Strings” results in SLOWER scans than the built-in search string list. However, the trade-off is that the “Alternate Search Strings” gives you more PROTECTION because this set detects “AS-OF-YET UNDISCOVERED” mutations of existing viruses also.

VirusDetective recognizes text files generated from any Macintosh word processing application. Just make sure it is a TEXT file with one search string per line. The newly read-in list (or partial list) of search strings is NOT permanent until you click the Save button.

C.3.2 “Append to List” button: hold down the Option key and the “Read from File” button transforms into the “Append to List” button…

When you want to ADD search strings TO THE END of the existing search string list, hold down the option key and you will see the Read from File button changes to the Append to List button. The Append to List button allows you to ADD TO the current search string list instead of replacing the entire list.

C.4 “Write to File” button…
When you click the Write to File button, VirusDetective presents a directory dialog box that lets you save

a separate text file identical to VirusDetective's current search string set as displayed, but not necessarily saved, in the search string list. The default file name is "VD Search Strings"; change it if you like.

Save Files in the Form of… (in the Options dialog box) controls what file format VirusDetective saves the file as. For example, if you select the WriteNow button in Save Files in the Form of…, then VirusDetective writes the search string text file as WriteNow text format.

D) SEARCH STRING HELP FILE (IN THE LOWER SCROLLABLE WINDOW)…
<You are in this file now.> Scroll up or down to find the information you require.

E) "HELP->CLIPBOARD", "CANCEL" AND "SAVE" BUTTONS…

E.1 "Help->Clipboard" button…
When you click the Help->Clipboard button, you can copy the contents of the Help file to the Clipboard. Paste the contents of the Clipboard wherever you want (probably to your favorite word processing program).

E.2 "Cancel" button…
When you click the Cancel button:

1) if you made any changes, VirusDetective presents an alert box that asks "Save changes to name of the dialog box?". You have the choice between: Don't Save, Cancel (this cancel cancels the first cancel) or Save;

2) if you didn't make any changes, VirusDetective brings you back to the Main Window.

E.3 "Save" button…
If you click the Save button, VirusDetective saves any changes and then closes the dialog box.

APPENDIX 1) HOW VIRUSDETECTIVE'S SEARCH STRINGS WORK…
How VirusDetective works – it provides you with the means to search for the "fingerprints" an HTD leaves behind AFTER it infects a file but BEFORE it can get a chance to infect your files. In technical jargon, these fingerprints are called "resources."

VirusDetective searches various resources by means of various matching criteria. These matching criteria are called "search strings." The search strings are programable – you can change them, add them and delete them.

APPENDIX 2) WHEN AN HTD THREATENS THE MACINTOSH COMMUNITY, HOW WE DISTRIBUTE THE NEW SEARCH STRINGS…
One strong point of VirusDetective is that you don't have to purchase a new update of VirusDetective every time a new HTD appears on the scene. This is because of its search string programability.

We mail to all VirusDetective REGISTERED users (at no additional charge) postcards or flyers that contain the new search string as each new virus is discovered in a timely manner.

APPENDIX 3) SEARCH STRING SYNTAX: HOW TO CREATE AND CUSTOMIZE YOUR OWN SEARCH STRINGS…

There are only two things you really should know about search strings:

1) what role the search strings play in relation to how VirusDetective works, and

2) when a new HTD threat appears, if would be good if you were able to do one of these two things: a) add a search string to the search string set you already have, or b) add an entire new file of search strings to the VirusDetective you have. Either way, your search string list is the most recent.

You need NOT be concerned if you do not understand VirusDetective's search string syntax nor ever want to learn. You can get along just fine without knowing the nitty-gritty of the search string syntax!

We include this search string syntax primarily for advanced Macintosh users. Go for it!

THE SEARCH STRING SYNTAX IS AS FOLLOWS:

3.1 LEGEND TO THE SEARCH STRING SYNTAX…

An item within <> means that this item is further defined as SOMETHING ELSE;

:= means "is defined as one of the following," one possibility per line. Is defined as 'that' OR 'that' OR 'that' , each 'that' being on a different line;

An item within braces means that the item is OPTIONAL;

An item within «» means that item can be REPEATED one or more times;

An item in bold means that item is a KEYWORD. A keyword must be typed EXACTLY as shown.

3.2 SEARCH STRING FORMATS…

<search-string> :=

<file-string> ; Comment
<file-string> & <resource-string> ; Comment
<file-string> & <data-string> ; Comment
<resource-string> ; Comment
<data-string> ; Comment

A search string is defined as one of five formats: 1) a <file-string>, 2) a <file-string> followed by the character '&' followed by a <resource-string> or <data-string>, 3) a <file-string> followed by the character '&' followed by a <data-string>, or 4) a <resource-string> by itself, or 5) <data-string> by itself.

All of the five can be followed by an optional comment which is a semi-colon followed by what you want to say in the comment. The braces mean the comment is optional.

<file-string>'s can actually follow most <resource-string>'s anywhere in the <search-string> (not just at the beginning) and they MUST precede a <data-string>. In general, you want <file-string>'s before anything else in order to speed up the search process.

3.3 FILE STRING FORMATS…

<file-string> :=

Creator <op> <CID>
Filetype <op> <FID>

<op> :=
=
≠
>
<

<CID> := 4 character file creator

<FID> := 4 character file type

A <file-string> starts with either the keyword Creator or the keyword Filetype followed by one of four comparison operators: '=' (is equal to), '≠' (is not equal to) , '>' (is greater than), or '<' (is less than) followed by a four-character file creator or four-character file type. (Type Option-= to get the '≠' character.) For example,

Filetype = APPL

matches all applications.

3.4 RESOURCE STRING FORMATS…
<resource-string> :=
Executables
<resource-selector> «& <resource-comp>»

A <resource-string> is defined as one of two things: 1) the keyword Executables, or 2) a <resource-selector> followed by one or more <resource-comp>'s each separated by an '&'.

Executables matches any executable resource (such as CODE, WDEF, etc.) that can exist in a file. Certain files such as the Finder's "Desktop" file do NOT normally contain these resources and these resource's existence in that file MAY very well mean you have a virus. The only thing that can come after the Executables keyword is a comment.

3.5 RESOURCE SELECTOR FORMATS…
<resource-selector> :=
Resource Start
Resource StartPos
Resource <RID>

<RID> := 4 character resource type

A <resource-selector> is defined as one of three things: 1) Resource Start which means the first executed CODE resource, or 2) Resource StartPos which means the first executed instruction of the first executed CODE resource, or 3) Resource XXXX where XXXX is some resource type like nVIR.

NOTE: <RID> MUST BE EXACTLY FOUR (4) CHARACTERS LONG, INCLUDING ANY SPACES. ONE, AND ONLY ONE, SPACE MUST EXIST BETWEEN THE WORD Resource AND <RID> OR BETWEEN Resource AND Start OR BETWEEN Resource AND StartPos.

3.6 RESOURCE COMP FORMATS…

<resource-comp> :=
Any
Data <pattern>
WData <pattern>
LData <pattern>
Pos <snum> & Data <pattern>
Pos <snum> & WData <pattern>
Pos <snum> & LData <pattern>
ID <op> <snum>
Name <sep><string><sep>
Size <op> <num>

<snum> :=
<num>
-<num>

<num> := unsigned number (in Base 10)

<sep> := any single character; each pair has to be the same character

<string> := string of up to 255 characters

3.6.1 Any

Any matches any <resource-selector> resource. For example,

Resource nVIR & Any

matches any nVIR resource.

3.6.2 Data <pattern>

Data matches any <resource-selector> resource containing <pattern> (described below). You can specify an optional starting offset position with the Pos keyword. Positive offsets add to the beginning; negative offsets subtract from the end. For example,

Resource Start & Pos -1344 & Data 060CA9#643E9

starts searching the first executed CODE resource for the pattern 060CA9#643E9 located 1344 bytes from the end of the CODE resource.

Data must be the last keyword in a search string.

3.6.3 Pos <snum>

The Pos keyword (if present) can occur anywhere before the Data, WData or LData keywords (you cannot, however, specify the Pos keyword if you already have specified Resource StartPos because the starting position is already determined.) <snum> specifies the location in the resource the <pattern> matching should start at. Positive offsets add to the beginning; negative offsets subtract from the end.

3.6.4 WData <pattern> and LData <pattern>

WData and LData work just like the Data keyword except that WData starts on 16-bit ‘word’ boundaries, and LData starts on 32-bit ‘long-word’ boundaries. Once the first byte of the pattern matches, the rest of the pattern is still compared byte by byte.

NOTE: WHEN YOU USE WData OR LData AND SPECIFY A Pos THAT DOES NOT FALL ON THE APPROPRIATE BOUNDARY, SCANNING WILL ACTUALLY START AT THE PREVIOUS POSITION WHICH SATISFIES THE BOUNDARY CONDITION. FOR EXAMPLE, IF YOU WERE TO WRITE

Pos 10 & LData 0210

THE PATTERN MATCHING WILL ACTUALLY START AT POSITION 8 BECAUSE POSITION 8 IS THE FIRST LONG-WORD BOUNDARY BEFORE POSITION 10.

Like Data, both WData and LData must be the last keyword in a search string. Also, note you can only specify ONE of these three keywords.

Where you have a search string that contains either Data, WData or LData, the offset location from the beginning of the resource fork where the search string matched displays in the Show Info dialog box as the “Data Search Offset” number. The Data Search Offset number displays as both a decimal and hexadecimal number (hexadecimal in parentheses).

3.6.5 ID <op> <snum>

ID matches any <resource-selector> resource where the resource ID satisfies the given relationship. For example,

Resource CODE & ID > 10

matches any CODE resource where the ID is greater than 10.

3.6.6 Name <sep><string><sep>

Name matches any <resource-selector> resource where the name is enclosed in the separator characters. For example,

Resource INIT & Name "RR"

matches the INIT resource where the name is RR (" is the separator in this case).

3.6.7 Size <op> <num>

Size matches any <resource-selector> resource where the resource size satisfies the given <op> relationship to <num>. For example,

Resource MEV# & Size = 722

matches any MEV# resource where the size is equal to 722.

3.7 PATTERN FORMATS…

<pattern> :=
<segments>
#<hex-char><pattern>
<wild-pattern>
<white-pattern>
[<segments>]
[<white-pattern>]

<segments> :=
<hex-segment>
<ascii-segment>
<hex-segment><ascii-segment>
<ascii-segment><hex-segment>

<wild-pattern> :=
<segments>*<pattern>

<white-pattern> :=
<segments>%<pattern>

<hex-segment> :=
<hex-byte-word><hex-pattern>

<hex-byte-word> :=
<hex-char><hex-char>

<hex-char> := character 0 through 9 or A through F

<ascii-segment> :=
'<string>'
"<string>"

3.7.1 <pattern>

A <pattern> is defined as one of six things: 1) <segments>, 2) #<hex-char><pattern>, 3) <wild-pattern>, 4) <white-pattern>, 5) [<segments>], or 6) [<white-pattern>].

3.7.2 <segments>

<segments> are either: 1) a <hex-segment> which is a sequence of hexadecimal digits, two per byte, or 2) an <ascii-segment> which is an ASCII string enclosed in either single or double quotes. Both segment types can directly follow each other.

When a <ascii-segment> is in single quotes, it means that the <ascii-segment> pattern must match EXACTLY in order for it to be considered “a match.”

If it is enclosed in double quotes, then all alphabetic characters match either upper and lower case alphabetic characters. In other words, it is case-insensitive. For example,

Data "HELLO"

matches the string “Hello” in the file.

3.7.3 #<hex-char><pattern>

#<hex-char><pattern> is used when you want to skip an exact number of characters in a pattern. It skips a specific number of bytes by using the ‘#’ character followed by a single hexadecimal character, 0 through F (represents 0 through 15 bytes), between each segment. For example, the pattern

Data 3C#500

matches a file containing 3C12C9006A8000.

3.7.4 <wild-pattern>

<wild-pattern> is used when you want to skip ANY number of characters in a pattern. It skips any number of bytes (‘wild card’) by using the ‘*’ character between each segment. Any sequence of zero or more bytes matches the ‘*’ character.

3.7.5 <white-pattern>

<white-pattern> is used when you want to skip ANY number of ‘whitespace’ characters (space, tab, CR, LF, VT, and FF) by using the ‘%’ character between each segment. Any sequence of zero or more whitespace characters matches the ‘%’ character.

3.7.6 [<segments>] and [<white-pattern>]

If a pattern segment (not containing a wild card) is surrounded by ‘[]’s (an “any of” pattern), then any character in the pattern can match a character. Furthermore, matching continues within the same “any of” segment until a character NOT in the segment is encountered. Scanning then resumes with the first character after the closing ‘]’. At least one character has to match the “any of” segment. For example,

Data "TO"[%00]'or'

matches “to” (case insensitive), followed by any number of NULL’s (hex 00) and whitespace characters, followed by ‘or’ (case sensitive).

3.8 DATA STRING FORMATS…

<data-string> :=
dataFork
dataFork «& <data-comp>»

<data-comp> :=
Pos <snum> & Data <pattern>
Pos <snum> & WData <pattern>
Pos <snum> & LData <pattern>
Size <op> <num>
Data <pattern>
WData <pattern>
LData <pattern>

A <data-string> is defined as one of two things: 1) the keyword dataFork (NOTE the lowercase d) by itself, or 2) the keyword dataFork followed by one or more <data-comp>'s each separated by a '&'.

The dataFork keyword matches if the current file has more than 0 bytes in its data fork.

The <data-comp> portion of dataFork «& <data-comp>» has the same meaning as its <resource-comp> equivalents (defined under the RESOURCE COMP FORMATS section) but refer to the data fork instead of the resource.

The Data, WData and LData must be the last keywords in a search string. Also, note you can only specify ONE of these three keywords.

Where you have a search string that contains either Data, WData or LData, the offset location from the beginning of the data fork where the search string matched displays in the Show Info dialog box as the "Data Search Offset" number. The Data Search Offset number displays as both a decimal and hexadecimal number (hexadecimal in parentheses).

-the end -