Recovering Apple Safari History Binary PList (Property List) Files

Introduction

In the Mac OS X Cocoa, NeXTSTEP, and GNUstep programming frameworks, property list files are files that store serialized objects. Property list files use the filename extension .plist, and are therefore often referred to as plist files. Property list files are often used to store a user's settings. They are also used to store information about bundles and applications, a task served by the resource fork in the old Mac OS.

MAC OS X

In Mac OS X 10.0, an XML format plist was introduced, with a public DTD defined by Apple. The XML format supports non-ASCII characters and storing NSValue objects (which, unlike GNUstep's ASCII property list format, Apple's ASCII property list format does not support). Since XML files, however, are not the most space-efficient means of storage, Mac OS X 10.2 introduced a new format where property list files are stored as binary files. Starting with Mac OS X 10.4, this is the default format for preference files.

Apple Safari Browser

The Apple Safari browser stores a history of visits to web pages in a plist. These plists come in two flavours depending on which Safari version was present on the system. In older versions of the browser, the history is stored in an XML formatted property list which is reasonably easy to parse and recover from unallocated clusters. The individual history records are stored in dictionary objects which have a nice structure and are easy to recover using RBE (Record Based Extraction) techniques.

In newer releases of the browser (for example Safari v4 for Windows) the plist is stored in a proprietary binary plist structure which is more difficult to parse and recover from unallocated clusters. With the binary structure, recovery is only possible using FBE (File Based Extraction).

As NetAnalysis v1.50 (and above) can parse and read Safari binary plist data, we will create a Blade Recovery Profile to attempt to recover these files.

Creating a Blade Recovery Profile

In this example, we are going to create a Blade recovery profile to extract binary plist files. To create a profile, select Personal Profile Database from the Tools menu. Select Add New to create a blank profile (as shown in Figure 1).

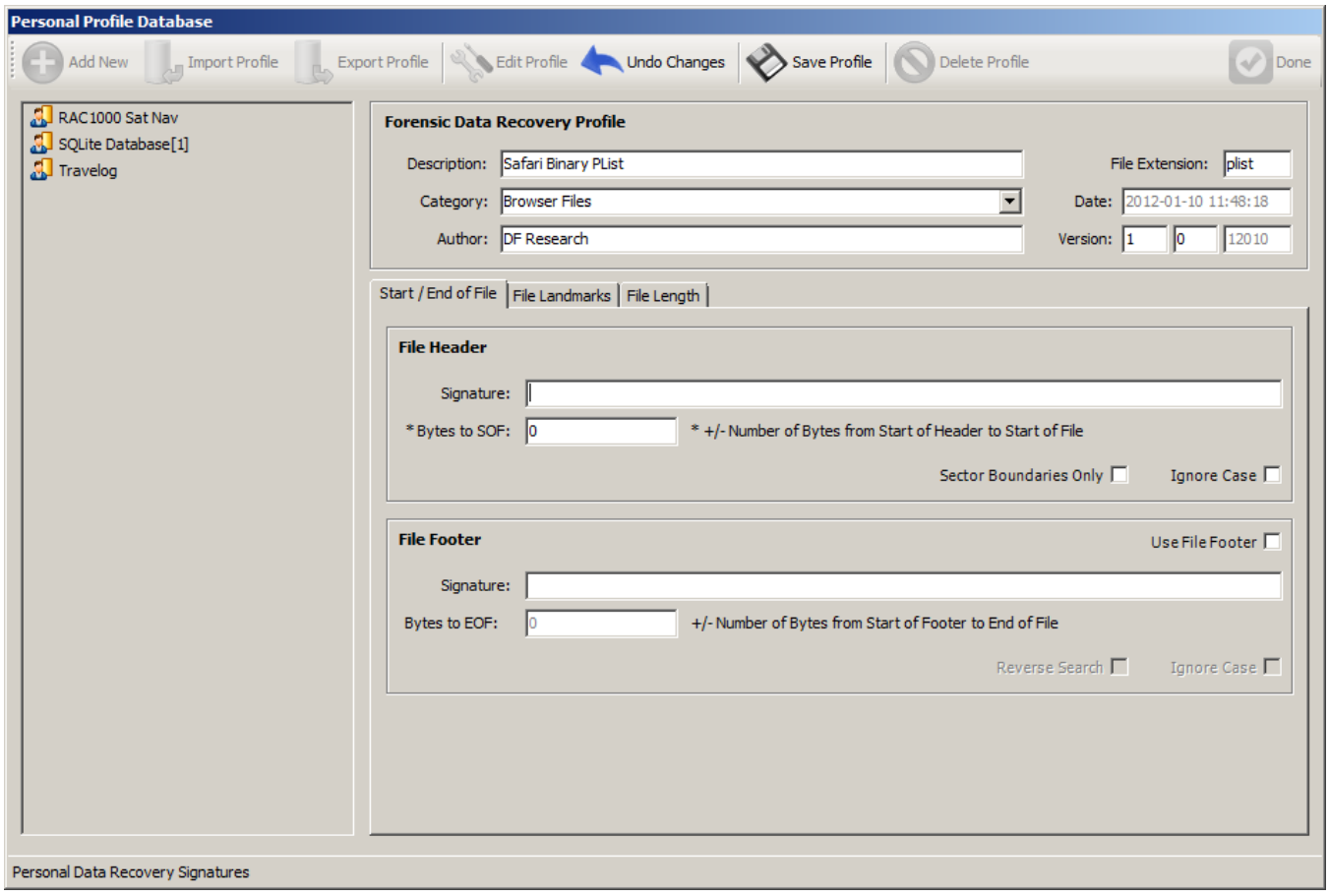

Figure 1

In the Description Field, we have entered “Safari Binary Plist”, in the Category field we have entered “Browser Files” and in the Extension field we entered “plist”. The Author and version numbers are automatically generated for you.

File Header Section

The next section to complete is File Header Section. The signature field holds the string regular expression which identifies that pattern of bytes at the start of a file (or segment of data). This is sometimes referred to as the “file signature” or “magic number”. This section contains information about the start of the file. To identify an appropriate file header signature, we will need to examine the structure of the binary plist. To do this, I have loaded a binary plist file into a hex viewer. Examination of the file (and with reference to the binary file specification) shows that the header contains the lower case string “bplist”. See Figure 2.

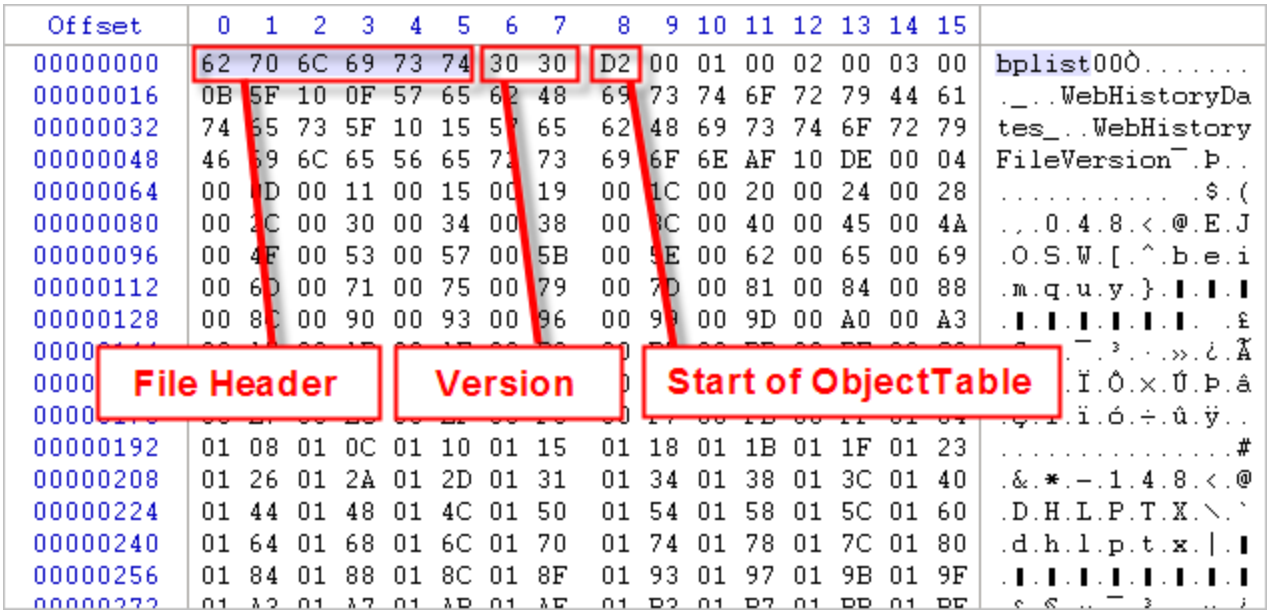

Figure 2

With this information, we can now enter the signature “bplist” for the start of the file as shown in Figure 3.

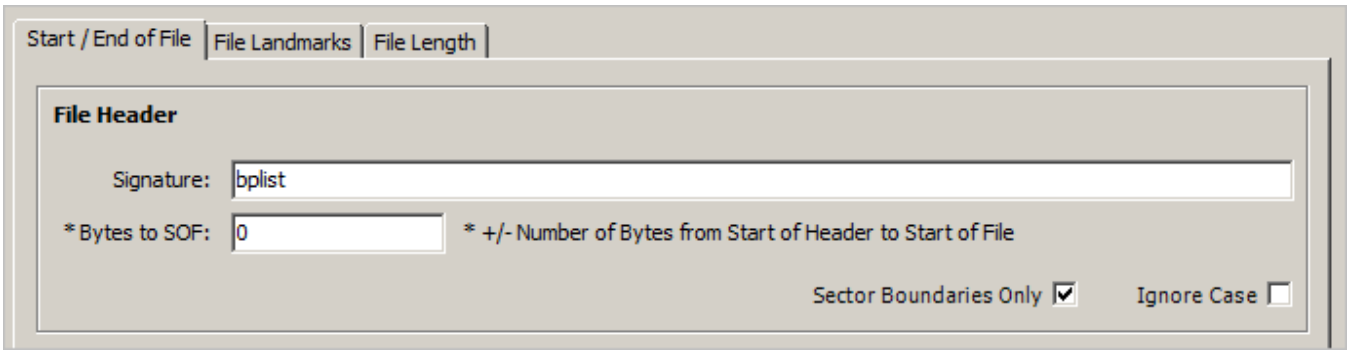

Figure 3

As we only want to recover plists which were originally history files, and not binary plist data embedded in other files, we will check “Sector Boundaries Only”. The signature is also case sensitive as we only want to recover “bplist” when all the characters are lower case, so the “Ignore Case” option is left unchecked.

File Landmark Section

The file landmark section allows you to improve the recovery capability even further. If you think of the file header and footers as bookends, the file landmark section refers to any data which can be found within the two boundaries. The landmark can be found at a specific offset, or at any position within the file. The landmark also uses regular expression patterns, and you can also select Unicode data.

Examination of the Safari History plist shows that the history records are stored in a dictionary where the key is “WebHistoryDates” and the data stored inside an array of dictionaries inside this object. The string “WebHistoryDates” can therefore be used as a Landmark within the file.

Check the “Use File Landmark” option in the File Landmark section, and enter the text “WebHistoryDates” in the Signature field. This string is also case sensitive so leave “Ignore Case” unchecked. In this case, as the exact location of the landmark can change, we will leave the Location field set to “Floating”. See Figure 4.

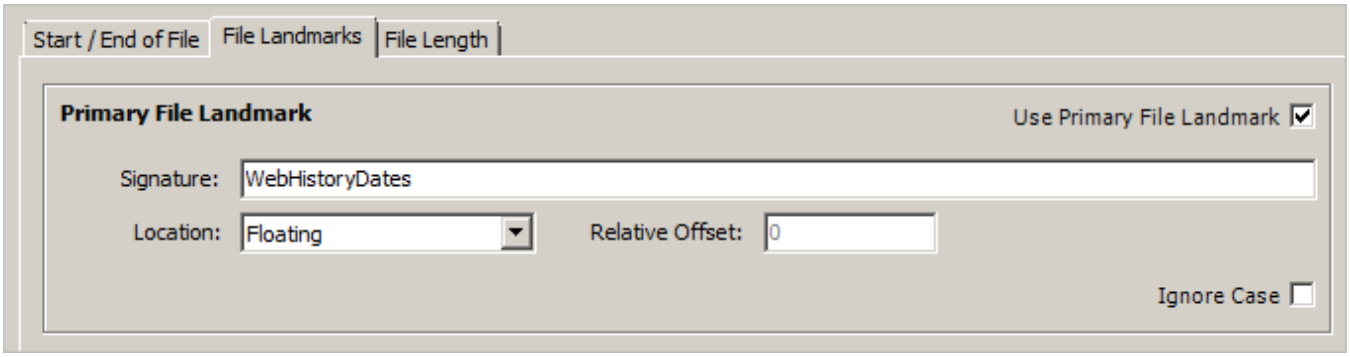

Figure 4

File Footer Section

The file footer section contains information to allow the end of the file to be found. By selecting the Use File Footer check box, the file footer fields will be activated. As with the other signature sections, you have the ability to use a regular expression pattern for this field.

In the case of binary plist files, there is no recognised footer, so we will need to devise a way to identify the end of the file. Examination of the binary plist structure shows that it does have a standard structure for the end of the file. The trailer structure is shown below in Figure 5.

```
59 typedef struct BPListTrailer
60 {
61         uint8_t                         unused[6];
62         uint8_t                         offsetIntSize;
63         uint8_t                         objectRefSize;
64         uint64_t                        objectCount;
65         uint64_t                        topLevelObject;
66         uint64_t                        offsetTableOffset;
67 } BPListTrailer;
```

Figure 5

Examination of the file shows that it has an array of 6 unused UInt8 values in the BPListTrailer structure. As there does not appear to be any runs of data within a binary plist containing this pattern of bytes, we will be able to use this to our advantage. See Figure 6 for a trailer from a binary plist.

```
00042128  EA 9A 3F 9A 4B 9A 5C 9A  9E 9A AA 9A C3 9A D4 9B  ê▮?▮K▮\▮▮▮ª▮Ã▮Ô▮
00042144  1F 9B 2B 9B 51 9B 62 9B  A3 9B AF 9B FC 9C 0D 9C  .▮+▮Q▮b▮£▮¯▮ü▮.▮
00042160  55 9C 61 9C 84 9C 95 9C  E1 9C ED 9D 11 9D 22 9D  U▮a▮▮▮▮á▮í▮.▮"▮
00042176  74 9D 80 9D A2 9D B3 9D  CF 9D DB 00 00 00 00 00  t▮▮▮¢▮³▮Ï▮Û.....
00042192  00 02 02 00 00 00 00 00  00 03 64 00 00 00 00 00  ..........d.....
00042208  00 00 00 00 00 00 00 00  00 9E 03                 ...........▮.
```

Figure 6

With the structure above, we could create a regular expression pattern to identify the Unused[6], offsetIntSize and ObjectRefSize bytes. We know that there will be six 0x00 bytes and then two bytes which will not be 0x00. In most cases, these two bytes will be 0x02, 0x02 as the offset Integer and object reference size are normally 2 bytes; however, as it is possible for these values to change, we will leave it as a non-zero value. In the File Footer Section, add the signature as shown in Figure 7. As these bytes are found 32 bytes from the end of the file, Blade will need to know where this footer is in relation to the end of the file. Enter the value “32” in the “Bytes to EOF” field.

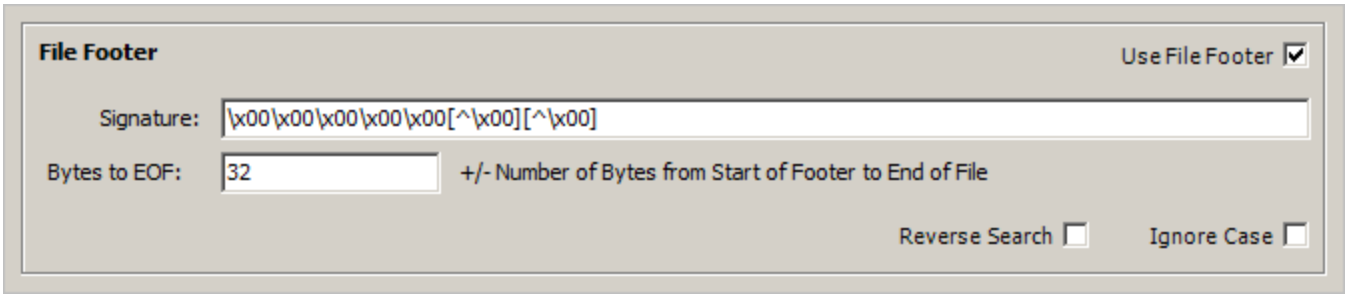

Figure 7

We will leave the other options for this recovery profile with their default values. Select “Save Profile” from the toolbar which takes you back to the main screen.

You can now select the newly created “Safari Binary Plist” profile and start recovering the data. The recovered files should load straight into NetAnalysis as long as they are intact and not corrupted.