Apple® IIGS

ByteBagger Reference Manual

Brian Wiser “Bites the bag”
Photo: Courtesy of Mike Maginnis

Dedicated to the memories of Joe Kohn 1947-2010
and Ryan Suenega 1967-2011

Credits

My thanks go to all who have helped me develop ByteBagger,
and especially for their extensive beta testing, many suggestions,
and error corrections:
David Schmidt, Antoine Vignau, Kim Howe, Andrew Roughan
The Installer and 2mg disk version by:
Antoine Vignau

Preface

Welcome to ByteBagger

Background

Way back, in the dim mists of the early days of the Apple II, I wrote a sector editor for 5.25" floppies. Since then, as a programmer, I have from time to time needed to be able to look into files, and if necessary, alter their bytes on the fly.

ByteBagger arose from having added a feature recently to SAFE2, that allowed a user to click on a file in the Home folder, and be able to view its content or data. This was achieved by having the file details sent out as a FinderSaysBeforeOpen IPC call, and then, depending on the file type, various installed NDAs, such as Balloon™ or Hermes™ might intercept that call, and open up appropriately.

The idea was simple, but I soon realised there were some file types that could not be handled this way. For instance, if you were to send out the name of an application, or some other System files, nothing would appear to happen till you Quit SAFE2, when a whole multitude of applications might suddenly open up without warning!

To resolve this problem, I built in an exclusion list of file types in SAFE2, and for those specific file types, send out an alternate IPC call directly to ByteBagger instead, which was then able to safely handle them.

Thus the need for the ByteBagger NDA arose...

The ByteBagger NDA

Requirements

ByteBagger is an NDA. Drop it into the System:Desk.Accessories folder.

Once the file is copied over, reboot your IIgs for the NDA to load.

What is ByteBagger

ByteBagger lets you view the raw data of either the data or resource fork of a file, search for target strings within that fork, using either Hex or ASCII data, and optionally allow you to edit the data using either the Hex or ASCII data fields, and write the changed data back to the file. You can also change the filetype and auxtype of a file.

Warning: ByteBagger will not let you alter the length of a file, as with many IIgs files, this can be a very dangerous operation. You should also know exactly what you are doing before you alter the data in a file, as you could easily corrupt an application and make it unusable! ByteBagger will not be held responsible for any damage that you may cause to your files...

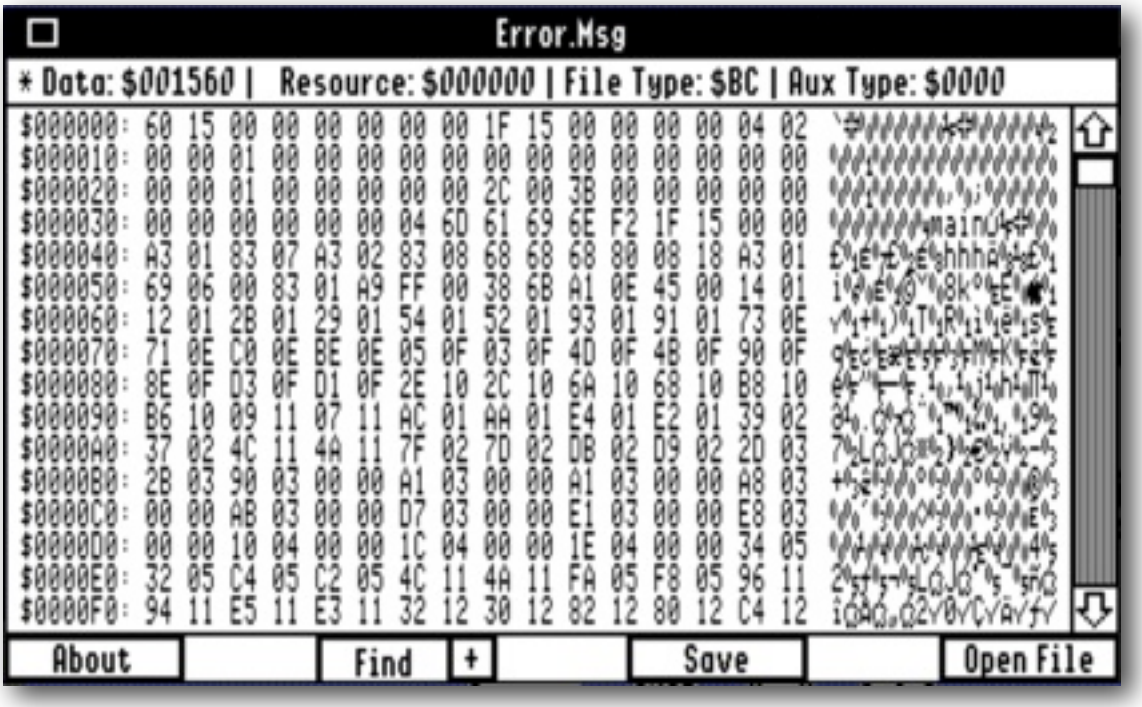
Legal

ByteBagger is Freeware, and may be distributed freely, provided the archive stays intact, no alterations are made to it, and full attribution is always made to the author Ewen Wannop. ByteBagger may not be sold in any form, but may be included in other distributed archives, provided you first seek my permission.

Using ByteBagger

Open ByteBagger and you will see this window:

When you open ByteBagger, there are only two active buttons, 'Open File' and 'About'. 'Click Open File' and choose a file. You can optionally open the Data, or if the file has one, the Resource fork. You will then see a window like this:



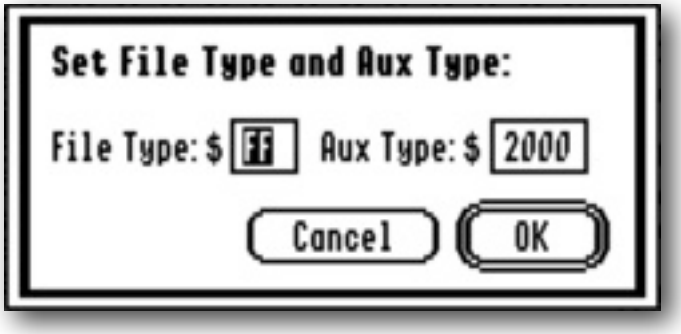
The window has a Title bar, showing the name of the current file, an 'Info' bar showing the 'Data' size, 'Resource' size, 'File Type', and 'Aux Type' for the file. An asterisk against the fork size shows which fork you are currently viewing.

The left column shows where in the file you are currently positioned. In the centre is a Hex data field giving a Hex representation of the data in the file, and to the right, an ASCII field, giving an ASCII representation of the data.

Note: Although the Hex data field values are correctly displayed, the ASCII data field has some control characters replaced with a period. Both fields have also been formatted with invisible CRs to break the lines.

To reduce memory overhead of the NDA, ByteBagger only opens a 256 byte window into a file. You can though move smoothly through the file using the scrollbar, or the Arrow keys, but if you have made any changes to one of the fields, and the changed data is valid, you will be prompted to save the changed data before you can scroll any further.

To switch forks, or jump to a fixed point in one of the forks, just click on either the 'Data:' or 'Resource:' fields in the Information bar. This will open a 'Go To:' dialog, where you can enter either a Hex value using a '$' sign, or a decimal value.



To change the File Type, or Aux Type of a file, click on the 'File Type:' field in the Information bar to open a 'Set File Type' dialog. You must enter Hex values.

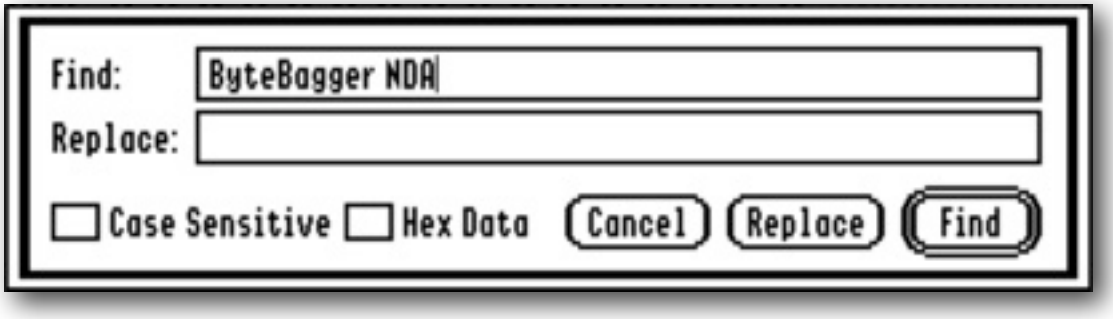
SAFE2 & Phoenix

From version 2.2.6 onwards, SAFE2 has a feature that allows you to double-click an entry in the Home folder. Phoenix also has this feature. They both send out a call to listening NDAs, so they can handle that file appropriately. For instance, double clicking a text file will open Hermes™ if it is available, and double-clicking a ShrinkIt file will open Balloon™ if that is available.

For applications, and some other files, which would not normally respond to that call, ByteBagger will open up instead, and display the data content of that file.

Note: If you hold down the Closed Apple, Option, or Alt key as you double click, if it exists, the Resource fork will be opened instead of the Data fork.

Find & Replace

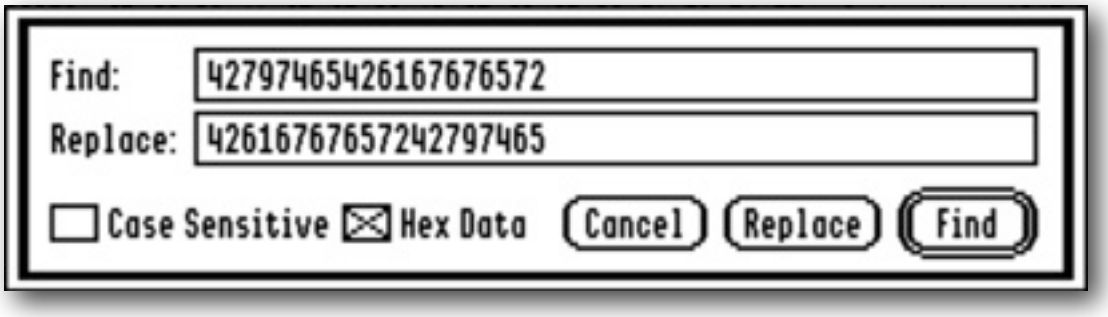


Now the more complicated stuff...

To find a target string within the file, click on Find, or press the OA-F keys. You can enter either a string representing Hex values and check the 'Hex Data' box, or enter ASCII text, with the 'Hex Data' box unchecked. Click 'OK' to search for that string, starting from the current position in the file. Optionally for ASCII text you can choose to make a 'Case Sensitive' search.

You can optionally choose to Replace with a second string. This string must be the same length as the first string, and must be different to it.

If the search string is found, it will be highlighted in the appropriate data field, and a small window opens. You can choose to Stop searching, Find Again, or if replacing, to Replace with the new string and find the next occurrence.

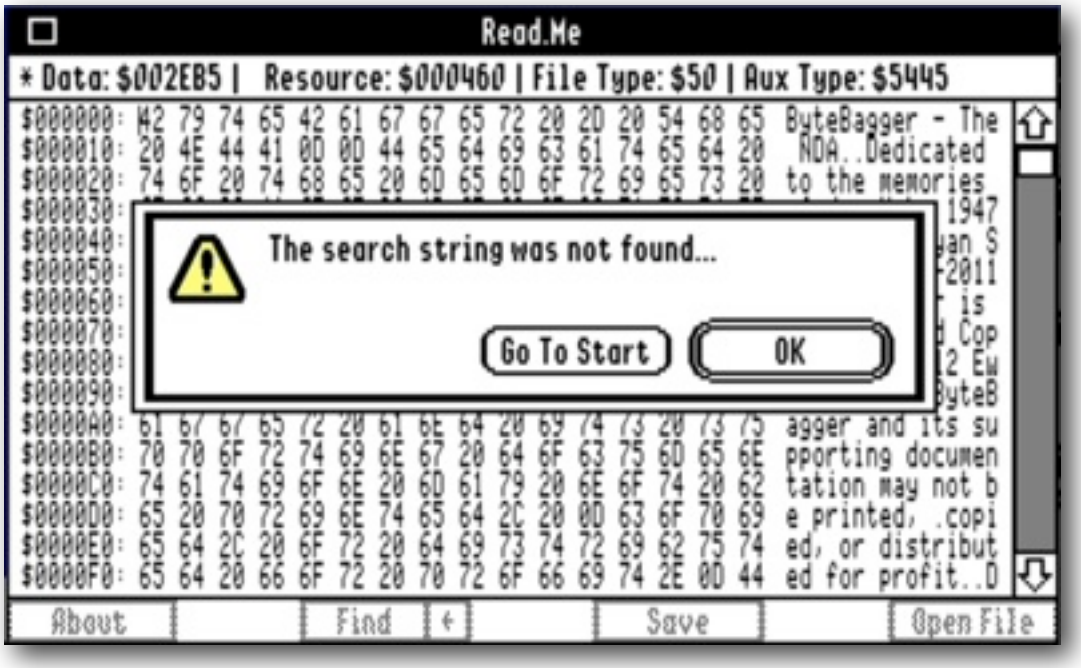


Note: If you copy data from the two data fields using the OA-C or Cmd-C keys, the data will be correctly formatted for a search. The data will be the true data, with any formatting spaces and invisible CRs removed. Because ASCII data may contain control characters and other hibit characters which the custom font displays, it may look different when you paste it into the Search box.

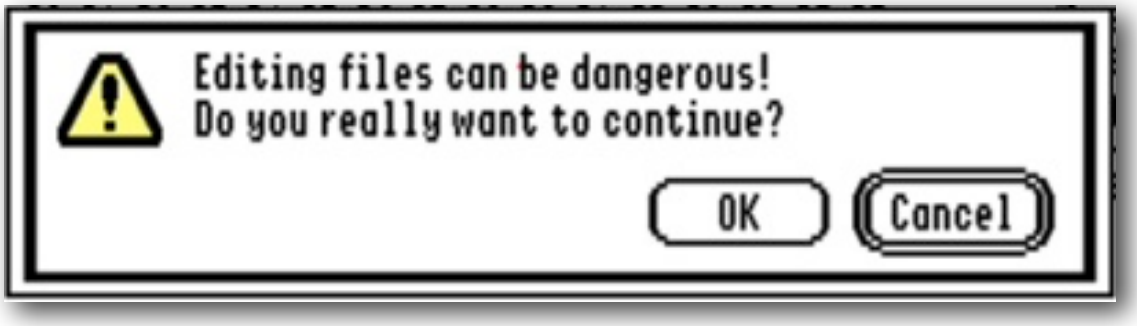
+ (Find Again)

Repeats the search, starting from the active cursor point, which unless you have changed it, will be where the last search left off. You can also press OA-G, instead of clicking the button.

If the search string was not found, you will get the chance to stop looking or start again from the beginning.



Editing



Warning: Be very careful if you go decide to edit the data in a file. You should know exactly what you are doing before you change anything at all, as modifying either the data, or resource fork of a file could have unpredictable consequences!

To keep the memory overhead of the NDA low, ByteBagger only displays a 256 byte 'window' into a file. Through scrolling, it will however appear as if you are moving normally through the entire file. If you make any changes to the data in any of the fields, this 256 byte 'window' must be saved before you can scroll further.

ByteBagger uses standard TextEdit windows to display the editable data fields. As ByteBagger will not let you change the size of the file when you Save, this 'window' must not change in size, or you will be unable to save the data back to the file. If the data field is of the wrong size, it will display in blue, and before you can scroll further, you must either edit back to the correct size, or it will revert to the original data, and you will lose any changes.

If the NDA is being closed, you will get a chance to Save any changes, but if the data fields are not of the correct size when closing, it is not possible to go back and edit the data. You will lose those changes as a result.

You can enter any byte from $00-FF into the Hex field, but as the ASCII field cannot display some control characters, this means you cannot enter these into the ASCII field. Any of those control characters that are in the file, will display as a period, and can be changed to normal printable characters. If you enter a period again in the positions where there was originally one of these control characters, the original data will be restored. To enter any control or other non-standard characters, it is best to edit the Hex data field.

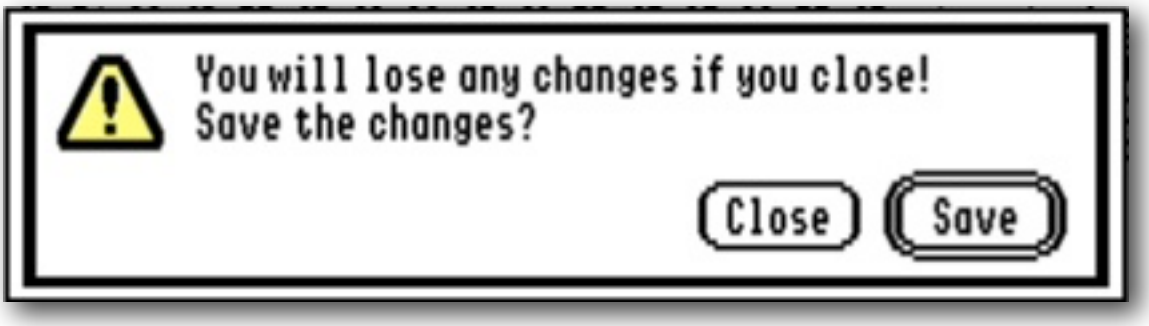
Note: Except for the last line of each field, there is an invisible CR at the end of every 16 byte line in both the Hex Data and ASCII fields. This CR must be preserved, or the field will be the incorrect length, and you will not be able to Save the edited data back to the file. If this CR is deleted, you should immediately see the lines change how they are displayed. You will need to put the CR line break back before you will be able to continue and Save the changes you have made. If you get in a mess, Revert to the original data, and try again.

Entering a character in the Hex data field will enter three, to maintain the correct length, and the second character will enter in the correct position within those three characters.

If you have changed any data, and try to scroll through the file, you will be warned, and have a chance to either Save, Revert, or Cancel. If you Cancel, you will get a chance to edit the data once more. If you Revert, it will restore the original Data from the file, and let you edit it once more. If you Save, the data will be Saved back to the file, and you will scroll as requested.

To save without scrolling, click the 'Save' button to save any modified data back to disk. Confirm that you wish to 'Save', or 'Cancel' without saving. You can also choose to 'Revert' to the original saved data.

If there have been no changes made to the data in either field, the Save button will do nothing.



IPC Call Interface

ByteBagger responds to inbound IPC calls directed to the 'bbRequestString'.

Three calls are currently supported:

```
$8900   goOpenFile     Opens a file in ByteBagger

        Format the txtDataIn block as shown below.

$8901   goOpenNDA      Opens the NDA

        The txtDataIn block is not required.

$8902   goCloseNDA     Closes the NDA

        The txtDataIn block is not required.
```

Send the calls in this format:

```
        pea   $8900          goOpenFile
        pea   $8001          stopAfterOne + sendToName
        PushLong #bbRequestString
        PushLong #txtDataIn
        PushLong #$00000000
        _SendRequest
```

Where txtDataIn is a data block. You must supply the Pointer to the pathname of the file, fill in the ftype and auxtype fields. Set option_key to zero to open the data fork, or non-zero to open the Resource fork:

```
txtDataIn anop
        dc   i'7'            parameter count
        dc   i4'PathName'    pointer to file
        dc   i4'0'
ftype   dc i'$0000'          filetype
auxtype dc i4'$00000000'     auxtype
option_key dc i'$0000'       option key modifiers
        dc   i4'0'
        dc   i'0'

bbRequestString str 'Speccie~ByteBagger~Wannop~'
```

Extras

Problems

Hopefully you will have none, but if you do, and they cannot be answered by reading these notes, please contact me on:

spectrumdaddy@speccie.uk

Other information

If you do not already know about Spectrum™, please drop by my home pages and read more. Apart from all the other wonderful things it does, Spectrum™ offers many useful tools for processing files, such as post processing text files that you have received that may have obstinate formatting.

Spectrum™ is now Freeware, and amongst my other applications, is available from my web site:

http://speccie.uk

Someone once said to me, 'Spectrum™ does everything!'