

Binary Pump 2.0

By: Eric Shieh

©1994 by Eric Shieh. All Rights  
Reserved

# Table of Contents

Legalities.....	iii
Distribution.....	iii
Disclaimer.....	iii
Shareware.....	iv
What is Shareware?.....	iv
Why Charge?.....	v
Shareware and Binary Pump.....	v
How Much?.....	vi
What do you get?.....	vi
How to Reach the Author	

.....

vii

## About This Manual

.....

viii

Installation.....1

    Requirements.....1

    Installing.....1

What Binary Pump Does....3

    Identifying Binary Files..3

    Filing.....3

Using Binary Pump.....5

    Before Using Binary Pump

    .....5

    Let's Go.....5

    Disabling.....7

    The Binary Pump Log File

    .....7

*Using Binary Pump With Communications Software8	
Setting Up Binary Pump.....	12
Setting up File Identification.....	12
*Type and Creator Codes .....	14
*The Tag Window Buttons.....	15
Setting Up The Filer.....	16
Terminology.....	16
Creating a New Set....	18
Editing Sets.....	20
Deleting Sets.....	20
*Other Set Attributes...	21

Manipulating Rules.....	22
Rule Types.....	22
Logical Statements.....	27
Preferences.....	33
Configuring Keys.....	33
Placing Aliases at the Original Location.....	34
Preventing the Filing of Unidentified Files.....	38
Choosing Which Files To Identify.....	39
How File Identification Works .....	40

How the Filer Works.....	41
Passes.....	41
Sets.....	41
Applications.....	43
A Simple File Filter.....	43
Resetting File Creators....	45
Archiving.....	46
A File Distributer.....	48
Shuttling Newly received Files.....	49
Tips and Tricks.....	53
Aborting.....	53
Filer Keys.....	53
Set List.....	53
Date Rules.....	53

Known Problems.....	54
Appendix.....	56

# Legalities

## Distribution

Binary Pump may be non-commercially distributed, provided that the archive remains intact. The archive should contain:

Binary Pump 2.0  
Binary Pump User's Guide  
[Read This](#)

Commercial distribution, such as the selling of this program on computer disks (e.g., floppy or CD-ROM), requires prior written permission from the author.

This manual may be copied for use with Binary Pump. Any other use (e.g., reproduction in one's article) is prohibited without prior written permission.

## Disclaimer

Use this software at your own risk. I, the author, take no responsibility for any damage caused inadvertently or implicitly, by this program.



## Shareware

### What is Shareware?

Shareware is a term coined many years ago for software which is chiefly distributed through non-commercial channels. This allows small companies, or individual programmers to market their programs for virtually nothing.

Shareware is commonly mistaken to be “free” software. Shareware is not free. It allows you to try the software for a certain period of time, so that you may decide whether or not you really need its functionality. If so, you should pay the shareware fee which typically ranges from \$5 – \$40, depending on the size and functionality of the program.

With shareware, you get a great bargain. You get to test drive the software first, to make sure you really need it without having any sort of bill placed in front of you. In addition, because no money needs to be spent on marketing or bureaucracy, shareware is often cheaper than comparable software from commercial developers.

Admittedly, there is a down side to shareware. The whole reason behind shareware is that the author has limited resources compared to commercial software companies. Thus, there is no technical support to call, the manual is often unbound (if it even exists), and the potential for upgrades is very foggy.

For most authors technical support is provided through either e-mail, or standard “snail” mail. (e.g., US Postal Service), by one person — the authors themselves. So, depending on the

volume of their mail, technical support can vary drastically. From experience, major software companies typically have robust technical support with many knowledgeable people available to answer questions. There are a few, though, that still provide horrid technical support if they provide it at all.

An important factor in whether the product gets upgraded or not depends on its popularity. Generally, the more successful the product, the more likely the author with upgrade it. So, in paying for a shareware product, you will increase the chances of it being upgraded. Upgrades include added functionality as well as bug fixes.

## Why Charge?

Many programmers often give away software for free. They retain copyrights and such, but they do not require any sort of payment; although they sometimes request a post card or some sort of response.

Some groups exist which provide many useful, robust, utilities for free. In reality, they receive donations in the form of new hardware and software.

Unfortunately, not everyone is in the same position. Many of the free software products are developed using tools and resources of other people. For example, universities give units to programmers who write programs using university resources. This is where a bulk of free software comes from.

Another part is from trivial software — software which takes only several days to write and has limited functionality.

Developing software is becoming more and more expensive. Software prices seem to have risen dramatically over the past few years. For example, a typical compiler (the most important tool for developers) for the Macintosh costs anywhere from \$200 to \$600. Student discounts are substantially cheaper at around \$70–\$150. In addition to the compiler, there are other programming tools which facilitate the programming process tremendously. These include subscriptions to technical magazines (which are not cheap), new hardware (almost everyone would benefit from a new computer every 3–4 years), and other software packages (e.g., Microsoft Word for writing documentation!).

## Shareware and Binary Pump

In the early days of Binary Pump, when it only performed file identification, it was given away free. The original file identification mechanism took approximately 11 hours to complete and source code was also available.

After many more hours (and about half a year) the Filer was added with several additional features and Binary Pump is now taking a turn into shareware.

## **How Much?**

Binary Pump's recommended price is \$20 US.

Please send to:

Eric Shieh  
P.O. Box 1235  
Danville, CA 94526

If you have an e-mail account, and would like upgrades sent to it, please include it with your payment. Note that some e-mail accounts have size limitations, and the total size of Binary Pump is fairly large.

What is meant by "recommended"? I basically do not want you to feel that "It's a great program, but not worth \$20". If you really think Binary Pump is useful, I would rather you use it, and simply pay what you feel it is worth. I also understand that not all people have money. So, since most people have a skill or trade, I will take goods or services instead: if you are a photographer, send me one of your photographs, if you are a writer, send me one of your books, or if you are an account, well...you can do my taxes or something.

## **What do you get?**

What you get by registering Binary Pump:

- If you have an e-mail account, I will put your address on a mailing list so that you receive upgrades as soon as they are released.
- Technical support via snail mail or e-mail.
- New disks (containing the latest version of Binary Pump) or a printed and bound manual for the cost of materials and shipping only.
- You can complain and holler about problems and I will listen.

I will always fix bugs. But complaints such as "I don't like..." or "Why doesn't it do ..." will be given priority if they are from registered users. You give to me, and I will give to you, that is the way it is supposed to work.

## **How to Reach the Author**

E-mail is usually the best way to reach me.

Internet:     eshieh@cory.berkeley.edu  
Compuserve:   76164,765  
Snail Mail:

Eric Shieh  
P.O. Box 1235  
Danville, CA 94526

## About This Manual

Throughout this manual **bold** text is used to indicate names of items on the screen, such as menus or buttons. Whenever possible try to identify these items before moving on to the next step.

In addition, the `Courier` font is used to indicate literal text. For example, an instruction might say: Type `open sesame` into the dialog.

Section names which are preceded with an asterisk (\*) contain more advanced material which are not necessary for casual use.

Before you go through this manual, please take the time to browse through the Apple Macintosh user's documentation and make sure you are familiar with terminology like "drag and drop". This manual assumes you know how to navigate through interface elements such as these.

## Installation

### Requirements

Binary Pump 2.0 requires System 7.0, and about 140K of free memory in order to run. In addition, Binary Pump uses about 200K of disk space.

Binary Pump 2.0 is a fat binary. This means it runs natively on both the PowerPC and standard 68K Macintosh platforms. Special “non-fat” versions of Binary Pump are available if you are tight on disk space. The “non-fat” versions use around 100K of disk space.

### Installing

Binary Pump is compressed with a popular compression program called Compactor. This compression program is widely available from user’s groups and on-line services such as American Online.

#### **To decompress Binary Pump in Compactor:**

1. Open the Binary Pump archive in Compactor.
2. Choose **Select All** from the **Edit** menu.
3. Choose **Extract** from the **Archive** menu.  
A standard file dialog box will appear
4. Choose a location to extract to.  
If necessary, create a new folder.
5. Click on the **Extract** button.  
A status window will then appear. When it disappears, Binary Pump will have been installed.



Another program called Stuffit Expander can decompress Binary Pump. To decompress Binary Pump using Stuffit Expander, drag the Binary Pump archive to the Stuffit Expander icon and release the mouse button. Stuffit Expander should immediately decompress the file into a folder, located in the same directory as the Binary Pump archive.

If you used a previous version of Binary Pump, it may also be necessary to rebuild the desktop file. If you do not do this icons may not appear

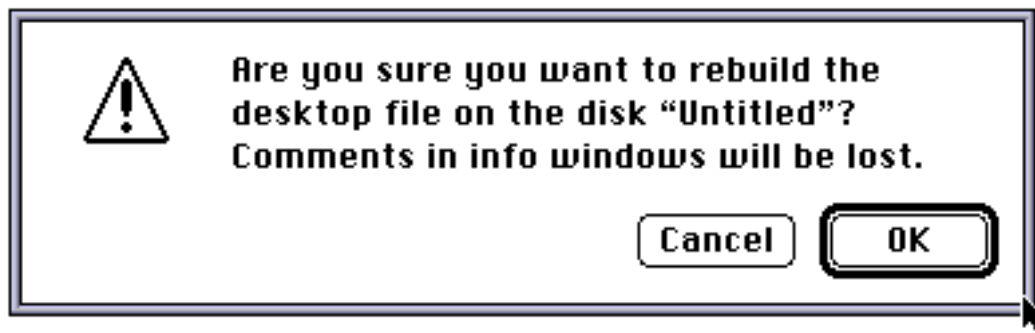
correctly, and drag–dropping folders and disks to Binary Pump may not work.

### To rebuild the desktop file:

1. Restart the Macintosh.



2. Hold down the option and command (Open Apple) keys.  
A dialog box will eventually appear, asking whether or not to rebuild the desktop file.



3. Click on **OK**.

A status window will appear, showing the progress of rebuilding the desktop.

## What Binary Pump Does

Binary Pump is a Macintosh utility which performs two basic functions: Identifying binary files and filing files into user defined folders.

### Identifying Binary Files

The first major function of Binary Pump is the identification of binary files. Although this function is rather trivial for most Macintosh users, it is extremely useful for people who frequently exchange files with IBM PC, Unix, VMS, or any other non-Macintosh computer. Among computers, the Macintosh is fairly unique in that each file has a creator and that Macintosh files generally do not use (or require for that matter) file name extensions (e.g., .com, .exe, .Z, .zip, etc...). Therefore, there is a disparity between the Macintosh and these other systems.

When transferring files between other machines, Macintosh users will often find themselves faced with the generic document icon. This is an indication the Macintosh is confused about what kind of file it is and what program created it. Thus, if you were to double-click on the icon (to try and open the file), you will more than likely get an error message stating that the correct application could not be found.

Binary Pump attempts to solve these problems by analyzing the contents of these files and setting their file type and creator correctly.

Binary Pump, is *not* a file format converter: it does not modify the file contents in any way, shape or form.

Binary Pump can also recognize and decode Macbinary files.

### Filing Away

In the process of organizing (or re-organizing) their files, people often designate certain folders for certain files. For example, creating a folder named **Finance** which *should* contain all files related to accounts, taxes, and payroll. Or, for the graphics artist, a folder named **Flowers** which *should* contain all files containing pictures of roses. The key word here is *should*; it is very easy to have several files, that *should* be in the same folder, scattered throughout a hard disk.

Binary Pump memorizes all those *shoulds* so that when you give it a file, it puts it in the correct folder. In this sense, Binary Pump is similar to a clerk: give it all your files and it will (hopefully) put them in the correct place. Binary Pump's file identification feature is also analogous to a clerk's labeling of files, or re-labeling of files (in case the label does not match the file's contents).

## Using Binary Pump

### Before Using Binary Pump

Binary Pump comes pre-configured to identify popular file formats and set their file types and creators to match popular Macintosh programs. Thus, it is possible to use Binary Pump's file identification function without having to go through the setup procedure.

Binary Pump's Filer, on the other hand, requires that you go through the setup procedure first. Otherwise, Binary Pump will not know where you want your files placed. See the section *Setting up the Filer* for more information.

### Let's Go

File identification and Filing are performed through the same mechanism. This allows files which have been identified, to be automatically filed.

## To process files with Binary Pump under System 7.0 or higher:

1. Go to the Finder

This is the program with various icons, from which you run applications.

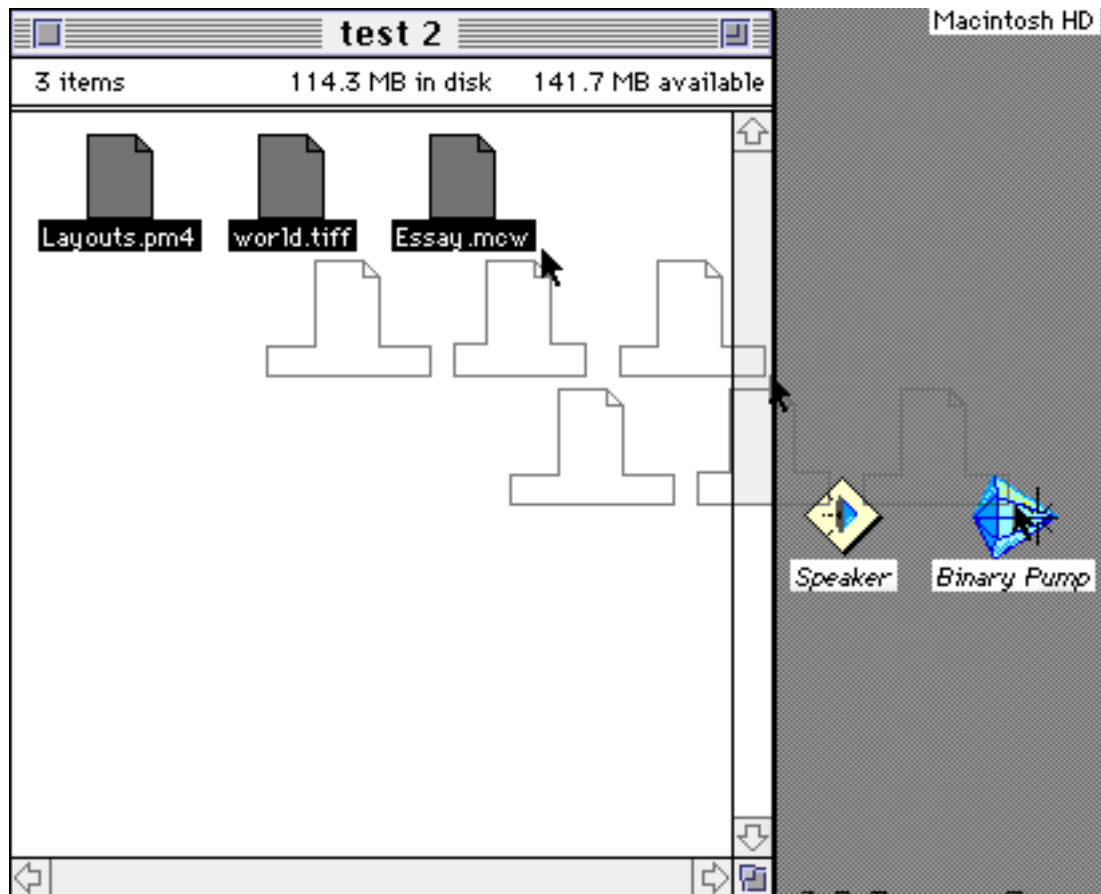
2. Place Binary Pump (or an alias) in an accessible location.

The Desktop (the gray area) is usually the best place.

3. Locate and Select (highlight) the files you wish to identify.

4. Drag the files or folders to Binary Pump.

Binary Pump will then process the files.



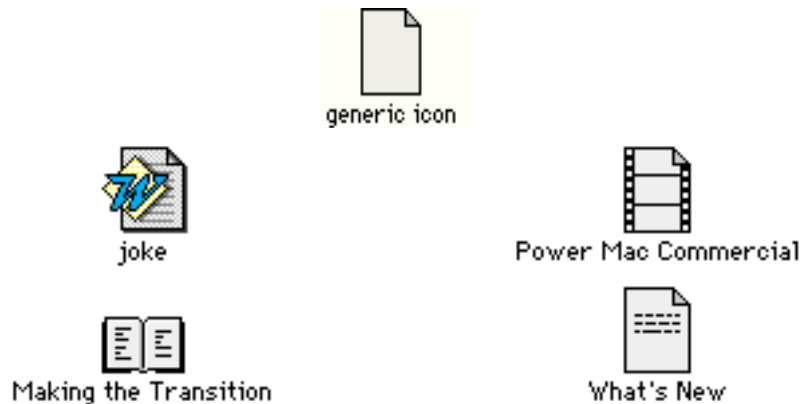
Dragging and Dropping files onto an alias of Binary Pump

You may continue working while Binary Pump processes files in the background. To place Binary Pump in the background, click on the desktop (the gray, background area of your screen). Be careful not to use the files which you gave Binary Pump to process.

Two situations arise under which your files may not be correctly identified:

- Binary Pump does not recognize the file at all.
- Binary pump recognizes the file, but does not set the creator to the desired application.

You will be able to immediately recognize these situations by the icon of these file types: the generic document icon doesn't change. To remedy this situation, see the section *Setting up File Identification*.



Generic icon at top, with examples of non-generic icons below it.

## Disabling Filing or File Identification

There may be times when you want to have a set of files identified, but not filed, and vice-versa. Binary Pump can be configured, so that only the Filer or only File Identification runs when certain keys are pressed. See the *Preferences* section for more information.

## The Binary Pump Log File

To keep track of where files have been moved, and what actions have been performed on them, Binary Pump creates a file called “Binary Pump Log”. The Log file tries to be as specific as possible, stating whether it moves, duplicates, or aliases files and in the case of moving, whether it



moved the original or an alias. Any sort of errors are also written to the file.

If you are ever lose a file, or are puzzled by Binary Pump's behavior, the log file should be the first place to look. Also, if you send me bug reports concerning files, the log relevant log entries would be extremely helpful.

Here is a sample from a log file:

```
5:27 PM Mon, Jun 6, 1994 - Set Type and Creator of file 'blab' to 'pZIP' and 'pZIP'.
5:27 PM Mon, Jun 6, 1994 - Moved original file 'blab' to folder 'test 2'.
5:27 PM Mon, Jun 6, 1994 - Set Type and Creator of file 'test 5' to 'pZIP' and 'pZIP'.
5:27 PM Mon, Jun 6, 1994 - Moved original file 'test 5' to folder 'test 2'.
5:31 PM Mon, Jun 6, 1994 - An error occured while duplicating file 'untitled folder' to 'test'.
```

## \*Using Binary Pump With Communications Software

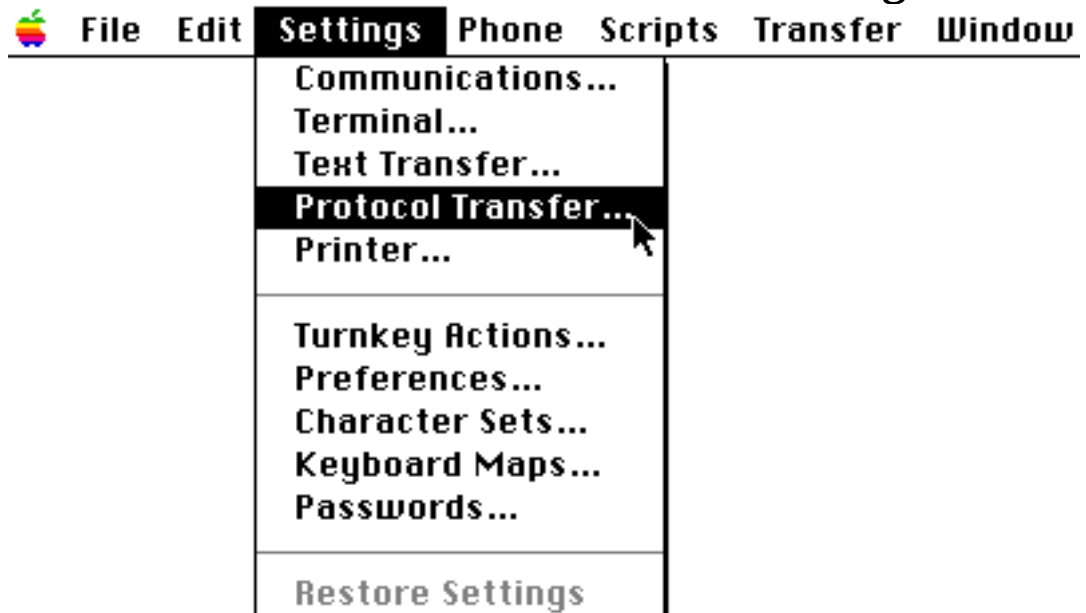
Unidentified files are often received through communications software. In this situation, there is an alternate way to use Binary Pump. Once set up, it will be possible to double-click your unknown documents (as opposed to dragging and dropping them onto Binary Pump), and have Binary Pump identify them. It really boils down to your preference between drag-and-drop and double-clicking.

This method is dependent on your communications software. The only packages described here are Microphone and Zterm; if you don't have either of these packages, you should consult your Communications software documentation. Most communications software have file transfer settings which allow you to set the 4 letter type and creator codes of unknown binary files (as opposed to text files). When you find out how to do this, change the creator signature to BINP.

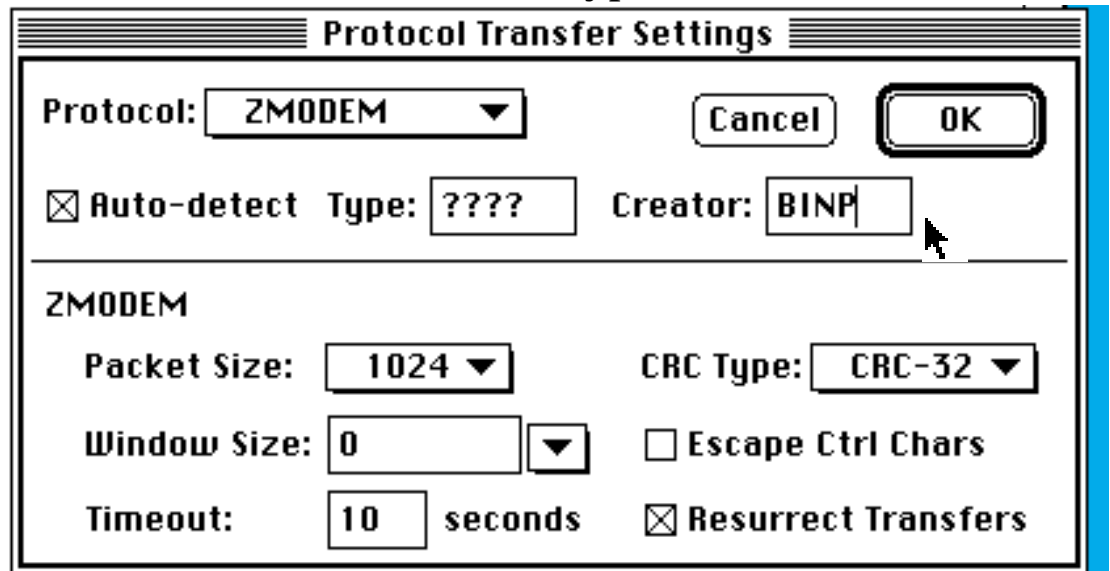
If you setup your communications software correctly, the next time you download an unidentifiable file (i.e., a file with a generic icon) you can switch to the Finder and double-click on it. Binary Pump will automatically launch and identify the file. If you get an error message stating that “The Application that created this document could not be found,” you should re-check your communications setup. Also make sure you are double-clicking on the correct file! This method will only work on unidentified (e.g., non-macbinary) files transferred after you have set up your communications software.

## To use Binary Pump with Microphone II 3.0 (or higher) or Microphone Pro:

1. Run Microphone and open the desired settings file.  
A settings file is where Microphone stores your communications settings. The default file is Microphone Settings. If you run Microphone and get a big window titled Microphone Settings, you are OK (continue with these instructions). If you run Microphone and don't get any windows, consult your Microphone user's guide on how to create/open a settings file.
2. Choose **Protocol Transfer...** from the **Settings** menu.



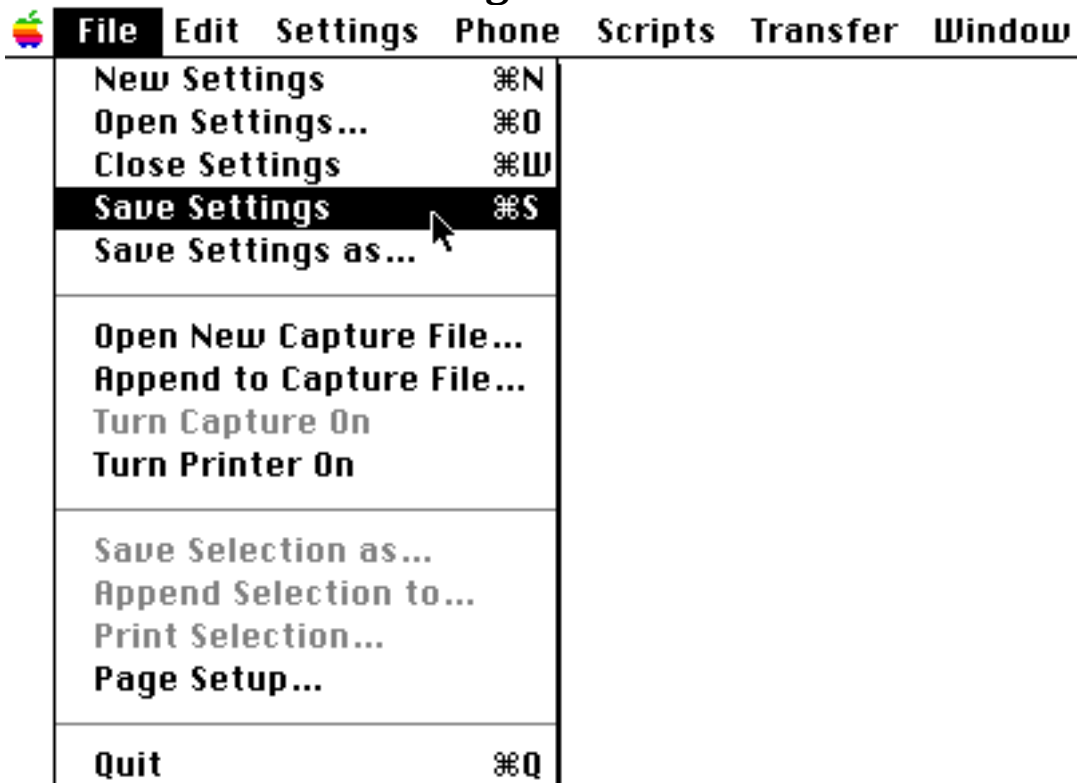
3. In the area labeled 'Creator', type BINP



The image shows a 'Protocol Transfer Settings' dialog box. At the top, there is a title bar with the text 'Protocol Transfer Settings'. Below the title bar, there are several fields and buttons. The 'Protocol:' field is set to 'ZMODEM'. To the right of this field are 'Cancel' and 'OK' buttons. Below the 'Protocol:' field, there is a checked checkbox for 'Auto-detect', followed by a 'Type:' field set to '????' and a 'Creator:' field set to 'BINP'. A mouse cursor is pointing at the 'Creator:' field. Below these fields, there is a section titled 'ZMODEM' which contains several more settings: 'Packet Size:' set to '1024', 'CRC Type:' set to 'CRC-32', 'Window Size:' set to '0', 'Timeout:' set to '10 seconds', and two checkboxes: 'Escape Ctrl Chars' (unchecked) and 'Resurrect Transfers' (checked).

4. Click on **OK**.

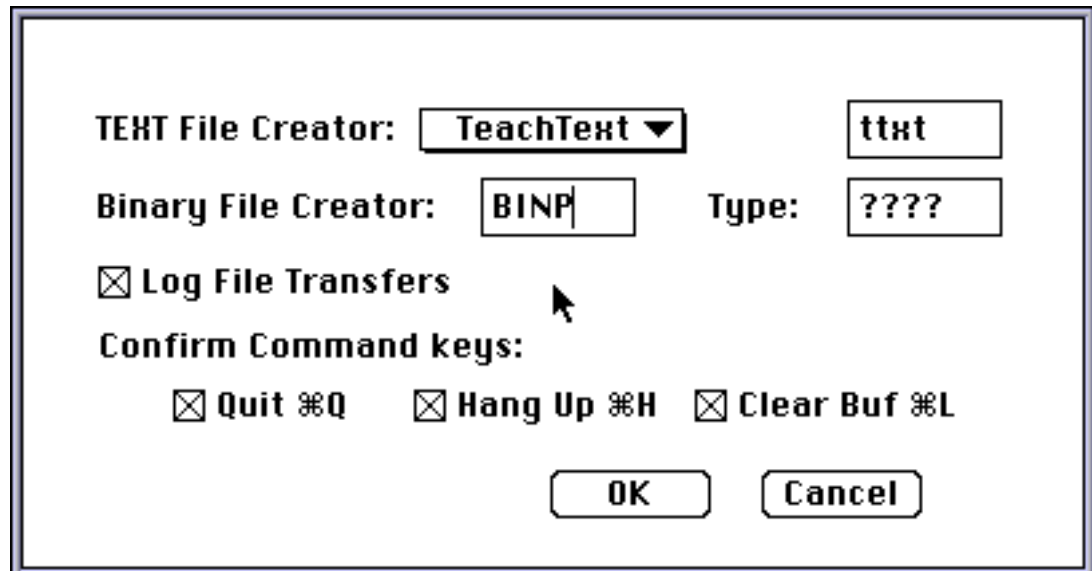
5. Choose **Save Settings...** from the **File** menu.



Microphone relies on the settings file for this configuration information. If it is ever replaced or deleted (i.e. you re-install Microphone), you should run through the above procedure again.

### Using Binary Pump with Zterm:

1. Run Zterm.
2. Choose **General Preferences...** from the **Settings** menu.
3. In the area labeled 'Binary File Creator' type BINP



4. Click on **OK**.
5. Choose **Save Setup** from the **Dial** menu.



## Setting Up Binary Pump

There may be times, when the File Identification mechanism either cannot identify your files, or incorrectly identifies them. In this case it is necessary to configure Binary Pump.

A minor facet: Not all files use binary tags. One such case is Photoshop 2.0 files—the Photoshop 2.5 format corrects this. The only way Binary Pump can identify these files is through file name extensions.

A list of all the types of files Binary Pump recognizes and their default type and creator codes can be found in the Appendix.

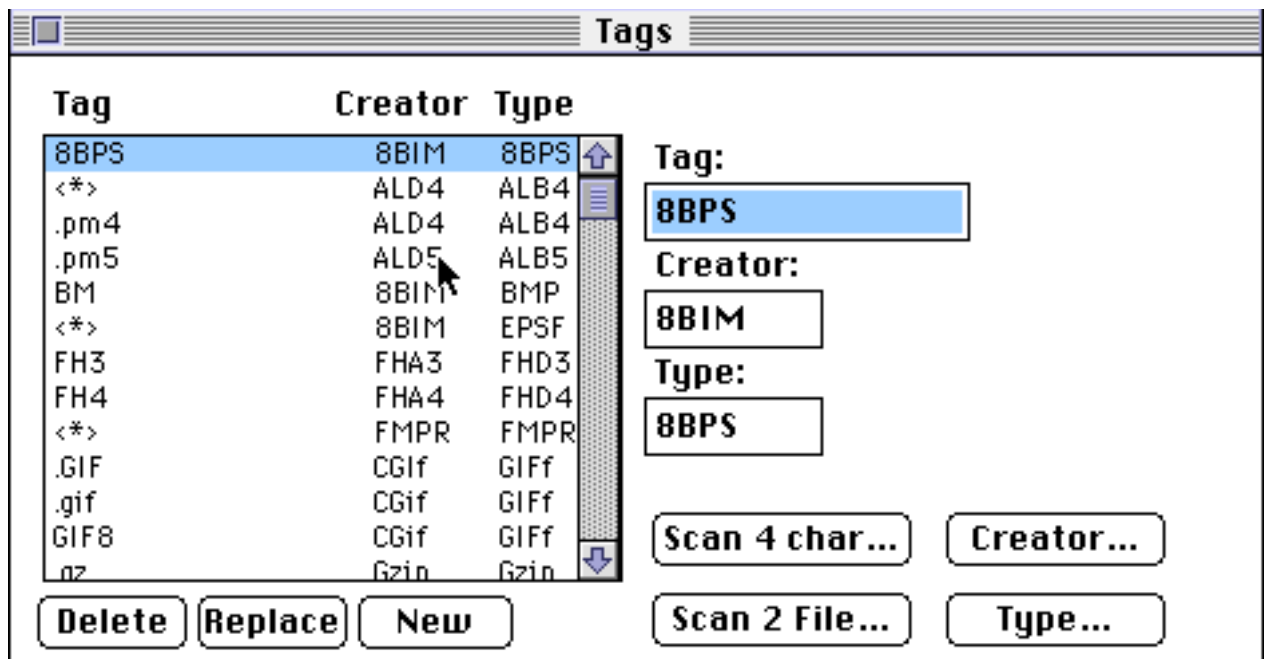
In order to use the Filer, it is necessary to set it up first. The setup procedure entails telling Binary Pump where to place files, and which files to place at that location.

## Setting up File Identification

One of the easiest ways to configure Binary Pump is to have two different “model” files of the same type; files which already have the correct Macintosh attributes. Files created on a Macintosh have the correct Macintosh attributes. For example, choose two Photoshop PICT files, two standard Photoshop 2.5 (Photoshop’s own file format) files, or two MS Word files.

### **To set up File Identification:**

1. Choose **Edit Tags...** from the **File** menu.  
This will bring up the Tags window.



2. Click on the **Type...** button and choose one of your “model” files.  
This is not necessary if you already know the four letter type code.
3. Click on the **Creator...** button and choose the same file.  
Again, this is not necessary if you already know the four letter creator code.
4. Click on the **Scan 2 Files...** button and choose your two different “model” files, one after the other.  
This will automatically place a binary tag into the Tag field.

If you do not have two files to use, but know a file name extension (e.g., .jpg) instead, type it directly into the Tag field, and include the period (.) in the extension.

Note: File name extensions are case sensitive. For

example, a distinction is made between .jpg and .JPG

5. Click on the **New** button.

This will create a new tag entry. Binary Pump will now recognize the new file format.

If Binary Pump beeps, the tag entry already exists. See below on how to edit existing entries.



### To edit a tag entry:

1. Locate the existing entry in the tag list.  
The tag list is sorted in ascending order according to the Type code.
2. Click on the entry.  
The tag entry information will be loaded and displayed in the Tag, Type, and Creator fields.
3. Change the information in the Tag, Type, or Creator fields.  
The **Scan 2 Files...**, **Type...**, and **Creator...** buttons may be useful in this step if you have a “model” file to use them on.
4. Click on the **Replace** button.  
The old entry will be replaced by the modified entry.

Note: NOT ALL FILE TYPES HAVE BINARY TAGS! If you click on **Scan 2 Files...** and, after choosing two different files, the tag field is left empty, the file type does not have a binary tag. (Make sure the two files you chose are meant to be the same file type.)

## **\*Type and Creator Codes**

A type code represents file contents; what kind of data is stored in the file. A program like Photoshop, which supports many file types, uses many different type codes. For example, JPEG files have the 'JPEG' type code. Some type codes are not so straight forward. Targa files usually use the extension .TGA — only three letters. Instead of making some new four letter type code like 'TARG', Photoshop uses the original three letters. Because type codes are required to have four characters, a space is inserted at the end. Thus the type code would look like 'TGA '.

A creator code represents an application which can open the file. Again, we will use a hypothetical JPEG file. We will set the creator code of this file to 'JVWR' — the creator code for JPEG View. Thus, whenever we double click on the file, JPEG View will open it. But let's say we want to edit the file — something JPEG View can't do. So, we now want Photoshop (which also supports JPEG) to open the file. One method is to set the creator code to '8BIM' — the creator code for Photoshop (don't ask me what it is supposed to stand for, I don't know!). Now, when we double-click on the file, Photoshop will open it.

## **\*The Tag Window Buttons**

This section discusses the function of each button in the Tag window.

### **New**

Creates a new entry, with the attributes contained in the Tag, Type, and Creator fields.

It will not work (and subsequently beep) if the tag entry already exists, or the Type and Creator codes are invalid—they must be four characters.

If you believe your codes are valid, try using spaces. For example, `IFF<space>`

### **Replace**

Replaces the attributes of the currently selected entry with those contained in the Tag, Type, and Creator fields.

The same restrictions that apply to the **New** button, apply here as well.

### **Delete**

Deletes the currently selected entry.

The Tag, Type, and Creator fields are not updated. Thus, if you want to undo the deletion, click on the New button.

### **Scan 4 Char...**

Scans the selected file, and places the first four bytes into the tag field. Some files have a 4 byte code, similar to a Macintosh's Type code, at their beginning. The Photoshop 2.5, GIF, and IFF utilize tags in this manner.

This method of finding tags is generally not recommended for novices.

### **Scan 2 Files...**

Scans two files, looks for similarities, and then creates a tag. The two files must be different: comparing the same file to itself would yield a tag as long as the file length!

### **Type...**

Scans the Macintosh Type code from a file and places it in the Type field. The supplied file must have a valid type code.

### Creator...

Scans the Macintosh Creator code from a file and places it in the Creator field. The supplied file must have a valid Creator code.

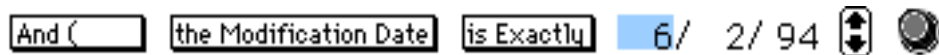
## Setting Up The Filer

### Terminology

The Filer is used to place files with user defined attributes, into user defined folders. Some of the attributes available are: File Name, Type, Creator, Modification Date, Creation Date, File Size, Finder Comments, Finder Label, and whether or not the file is locked.

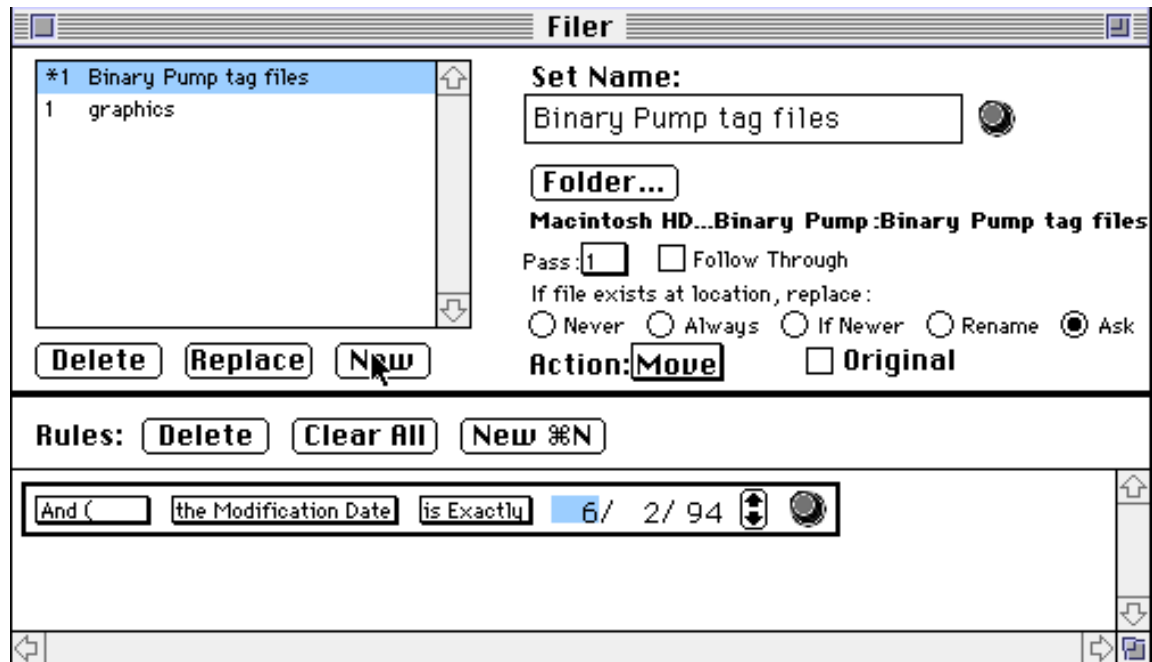
Using logical statements, attributes can be combined to form logical combinations. For example, if the file name is “Fireworks” and the modification date is July 4, 1994 place the file into the folder “Fourth of July Celebration”.

Logical statements combined with attributes are known in Binary Pump as Rules.



An example of a Modification Date rule in Binary Pump.

A rule by itself doesn't do much. A destination folder, for example, would be extremely useful (and in Binary Pump's case, necessary). While we are at it, a type of action to perform would be nice also. Thus, in Binary Pump, Sets are amalgamations of rules, a destination folder, an action, and other user options.



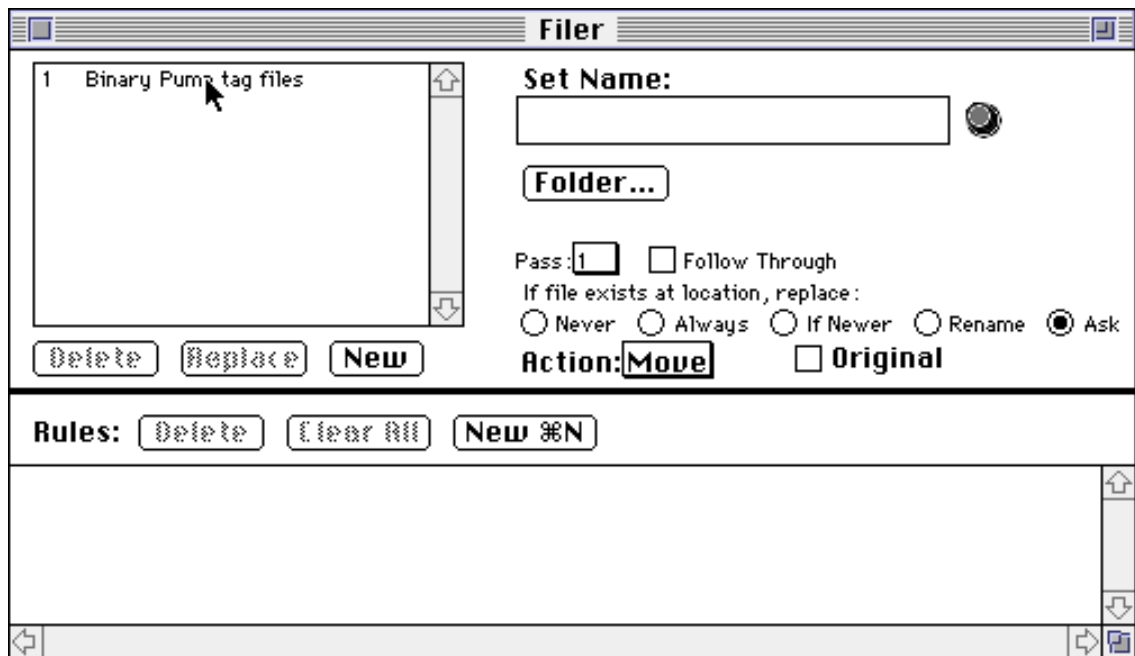
The Filer window with a loaded set.

## Creating a New Set

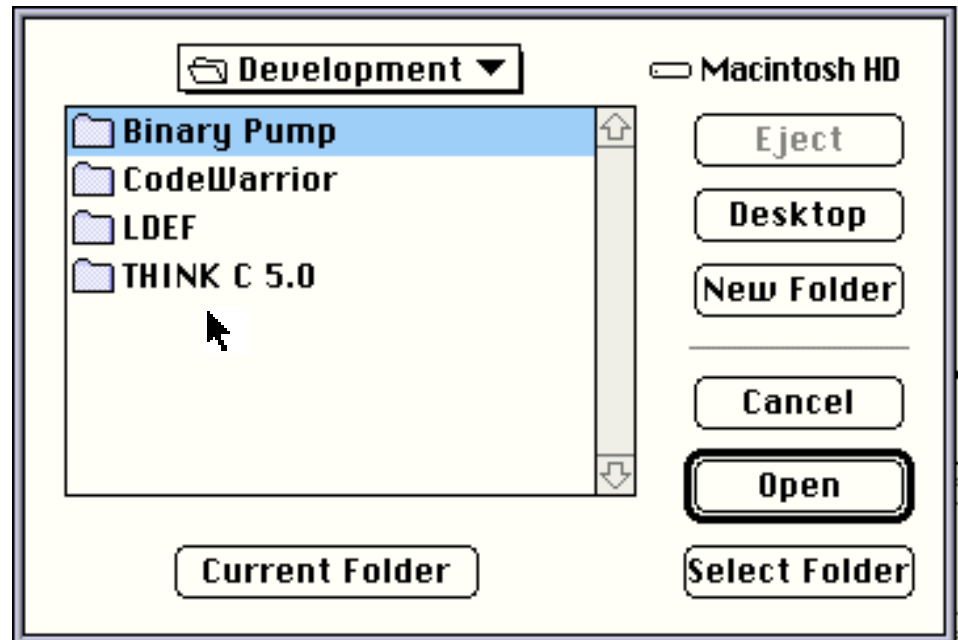
The Filer utilizes sets to figure out if a file should be moved, and where to move it.

### To Create a new Set:

1. Choose **Edit Sets...** from the **File** menu.  
This will bring up the Filer window.



2. Click on the **Folder...** button and choose a destination folder for the set.



3. Enter a Set name in the corresponding field, or click on the Helper Button to use the name of the folder.  
The Set name is the name that will appear in the set list.
4. Choose an action.  
Valid actions are:  
**Move**—move the file to the chosen destination.  
**Duplicate**—duplicate the file to the chosen destination.  
**Alias**—create an alias of the file at the chosen destination.
5. Click on the **New Rule** button to create a new rule.
6. Choose the rule type from the second menu and enter the required information.  
See the section *Manipulating Rules* for more information on rules.
7. Click on the **New Set** button located beneath the set list.  
A new set will be create an will appear in the Set list.

## Editing Sets

### To edit a set:

1. Locate the existing entry in the set list.  
The set list is sorted in ascending order according to pass number and set name.
2. Click on the set.  
The set information will be loaded.
3. Change the information.  
You may edit the name, edit the rules, add rules, or change any of the other set attributes.
4. Click on the **Replace** button.

The old entry will be replaced by the modified entry.

## Deleting Sets

### To Delete a Set

1. Locate and click on the desired entry in the set list.
2. Under the set list, click on the **Delete** button.



Note that the set information will remain loaded in the filer window. This allows you to “undo” the deletion by immediately clicking on the **New Set** button.



## **\*Other Set Attributes**

In addition to containing a destination folder and a type of action, sets contain several other properties.

### **New Rules**

The **New Rule** button behaves differently, depending on the currently active window item.

If a rule is selected, the **New Rule** button will insert a rule after the selected rule. If no rule is selected (e.g., the set list is selected) a rule will be inserted at the end of all the other rules.

Holding down the option key either inserts a rule before the selected rule, or before all other rules in the list if no rule is selected

### **Passes**

The Filer processes files in passes. Once a file has been processed, it is no longer available to passes of a higher number. Sets within the same pass will still have access to the file. (See the section *How The Filer Works* for more information.)

There is also a check box labeled **Follow Through**. When a set has this check box marked, files which are processed will be allowed to go on to latter passes. This allows the set to act transparently.

This option is useful in creating “monitoring” sets: sets which “monitor” the files passing through. It is thus possible to track all files going through Binary Pump without interfering with the passing of files from one pass to the next. If this option were not available, a set configuring in this way would “catch” all the files and not allow them to proceed to further passes.

Note that if another set processes the file, and does not have this check box marked, the file will still be removed from the reach of further passes. Remember, these types of sets act transparently.

### **Moving Originals**

The **Original** check box is only available when the set is performing the move action.

When Binary Pump moves files, the first set it comes across will obtain the original file, while all the others will receive aliases. This option specifies that this set is to receive the original file; not an alias.

In the event that two sets have the **Original** check box marked, the set which had the option marked most recently will receive the original.

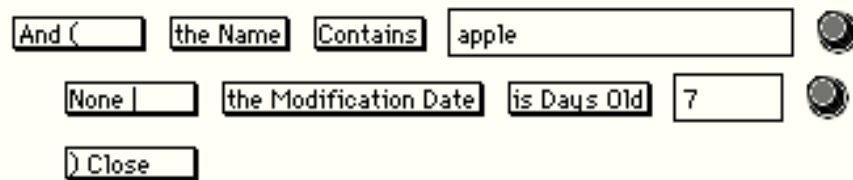
### Replacing

If a duplicate file is found at the destination folder, there are various action the Filer can take. The table below describes these actions.

Never	Never replace the duplicate.
Always	Always replace the duplicate.
If Newer	Replace the duplicate if the file we are moving is newer (the modification date is later) than the duplicate.
Rename	Renames the file we are moving by appending a serial number onto the end of the file.
Ask	If it comes across a duplicate, put up a dialog to ask whether or not to replace the file.

## Manipulating Rules

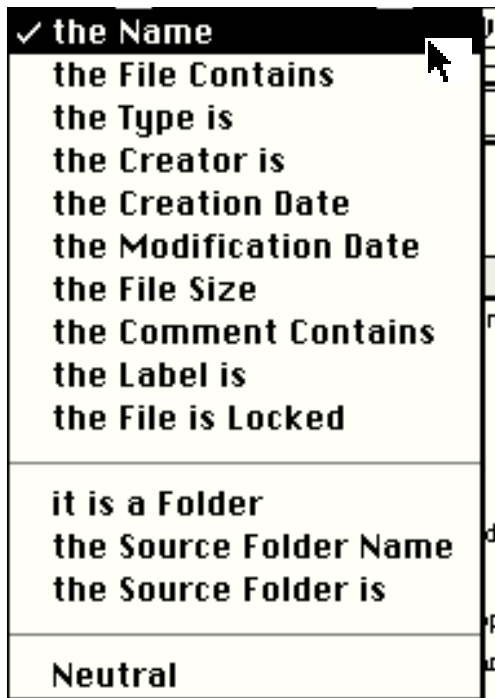
The main power behind the Filer lies in its comprehensive set of rules. These rules allow you to specify precisely which files you wish the set to be applicable to.



A list of rules from the Filer window

### Rule Types

Different rule types utilize different file properties. For example, the “File Name” rule requires that a file’s name fits a specific criteria and the “Modification Date” rule requires that a file’s modification date occurs during a specific time.



The rule menu.

Some rules have modifiers; slight variations of the rule. For example, whether a file's name is an exact match, or begins with certain letters, or, even more in general, merely contains certain letters anywhere within the file name.

In addition, some rules do not apply to folders. When given a folder, non-applicable rules will return False (or True if the Not flag is enabled).

Most rules have helper buttons. These helper buttons allow you to select a file to extract pertinent information from. For example, in a Type Rule, the helper button extracts the four letter type code from a selected file and automatically places the information into the rule.



A helper button

## Name and Source Folder Name Rules

Both the Name and Source Folder Name rules are very similar. The difference lies in where the Filer looks for the name. In the normal Name rule, the file's name is simply extracted. In the Source Folder Name rule, the Filer looks at the file's parent folder, and extracts its name.

The Name Rule deals with the various aspects of a files name. Except for the Greater Than and Less Than variations, the name rule is not case sensitive.

Modifiers:

Exact	The file name must match the given text exactly. This includes spaces, and any symbols.
Contains	The file name must contain the given text. The text may be at the beginning, in the middle somewhere, or at the end.
Starts With	The file name must begin with the given text.
Ends With	The file name must end with the given text.
Greater Than	A string comparison is done based upon the characters. For example, 'Zeppelin' is greater than 'ABC'. Think of it like names in a phone book: the farther into the list of names, the greater the magnitude. Capital letters are considered greater in magnitude than lower case. Thus, 'Ab' is considered greater than 'ab'. Not inclusive. This means that if two names are exactly the same, greater than will return False.
Less Than	Similar in principle to the Greater Than modifier, but has the opposite relation. Not Inclusive.

### Contains Rule

The Contains rule inspects a file's contents and looks for the given text. This rule does not apply to folders.

Modifiers:

Case Insensitive	There is no differentiation between upper and lower case.
Case Sensitive	The Filer takes note of the differences between upper and lower case.

### Type and Creator Rules

Type and Creator and creator codes are similar in that they are both four letter codes. For example, 'TEXT' or 'MSWD'. The difference lies in what they represent. See the section *Type and Creator Codes* for more information.

These rules are not applicable to Folders.

### Modification and Creation Date Rules

Because the modification and creation date rules deal with dates, they are almost exactly the same. Modification dates are changed whenever the file or folder is altered in some way. Folders are altered whenever new files are created within them (not when the files are modified). Creation dates are the dates of when files are created and normally never change.

In addition to date, time is also part of modification and creation dates. The dates specified in the Filer always have their time set to 12:00AM. When matching exact dates, the Filer ignores the time. But when comparing dates, time is taken into consideration. Thus, two dates may be equal, yet Greater Than will return True because of the time factor. One exception to this is the Minutes Old modifier.

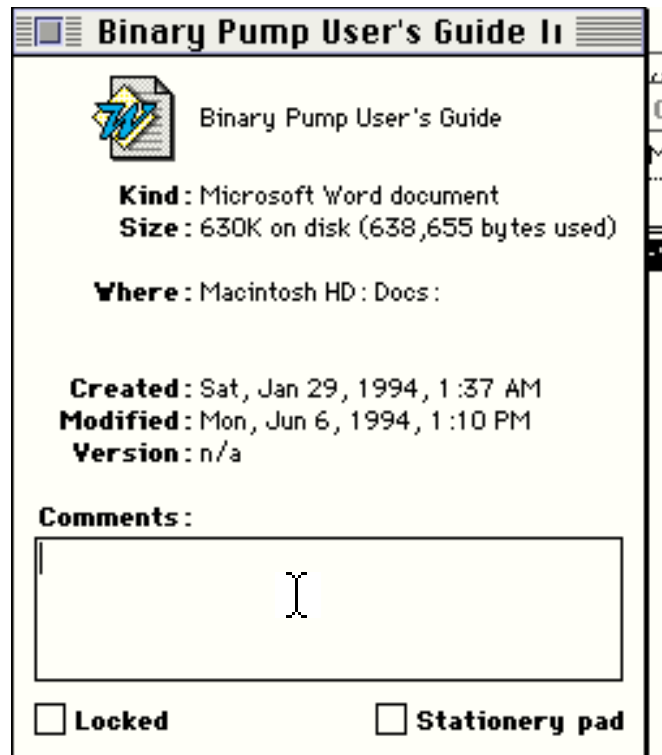
Modifiers:

Exact	The file's date must match the given date exactly.
Greater Than	The file's date must be greater than the given date. Not Inclusive.
Less Than	The file's date must be less than the given date. Not Inclusive.
Days Old	The current date and the file's date are subtracted, yielding the file's age in days. Checks if the file is <i>at least</i> the given number of days old.
Minutes Old	Similar to Days Old.

### File Size Rule

The File Size Rule performs a comparison with the given size in bytes, and the logical size of the file.

The logical size of the file, varies slightly from the physical size of a file. The Finder's Get Info window displays both.



The Finder's Get Info Window

In the above example, 630K is the physical size, while 638,655 is the logical size.

This rule does not apply to folders.

Modifiers:

Exact	The file's size must be exactly the given amount.
Greater Than	The file's size must be greater than the given size. Not Inclusive.
Less Than	The file's size must be less than the given size . Not Inclusive.

### Comment Rule

When you perform a **Get Info** from the **File** menu in the Finder, you may notice a section for comments. You may type up to two hundred characters (including spaces) for a comment. The Comment Rule will then look to see if the given text is in the comment of a file.

Comments have two disadvantages. The first is that if you re-build your desktop, all of your comments (on the disk which is being rebuilt) will go

into oblivion. A shareware utility exists which can re-build the desktop while retaining comments.

The second disadvantage is that comments currently cannot be extracted from files located on floppy disks.

### **Label Rule**

The Finder allows you to give your files eight different labels. Seven out of the eight are user-definable from the Views control panel.

The label rule looks at the labels of files and check to see if they match the given label.

Note: Labels are not actually updated until the folder containing the file is closed. See the section *Known Problems* for more information.

### **Locked Rule**

Files can be locked to prevent them from being altered, deleted, or renamed. This rule will check to see if files have this property.

### **Is Folder Rule**

This rule allows you to differentiate actual files from folders.

### **Source Folder Rule**

This rule checks to see if files are located directly within the given folder.

### **Neutral Rule**

There are times when it is desirable to have a logic statement by itself. In these instances, the neutral rule acts as a place holder that does not affect the chain of logic.

Using the Not flag in conjunction with this rule is not recommended. (The opposite of neutral, of course, is biased!)

For more information, see the following section, *Logical Statements*.

## **Logical Statements**

Logical statements are useful whenever you wish to create more than one criteria. In addition, there is a Not flag, which allows you to evaluate the opposite of your criteria.

## Prefix Notation

Binary Pump uses prefix notation in its logical statements. This means the type of operation is followed by some data. For example, for arithmetic, addition would look like + 2 5. Translated to infix notation this would be  $2 + 5$

Why use prefix notation instead of infix? Try to evaluate the following arithmetic problem:  $5+7*9-5$ . The rules of arithmetic state that you perform multiplication first, before addition or subtraction. Thus, the answer is 63.

The above example may seem trivial, but now let us apply it to logic statements with another problem: False Or True And False Or True. Because most people are unfamiliar with these types of statements, it is not as clear (as the above example), which to perform first, And or Or. This is a problem faced by computer programmers very frequently, and frequently they forget (or never bother to learn) which comes first. This ambiguity could be solved by parentheses. If the problem were (False Or True) And (False Or True) the solution becomes fairly clear.

In Prefix notation, there is never this sort of ambiguity. The problems:

And ( Or ( True, False), Or (True,False) )

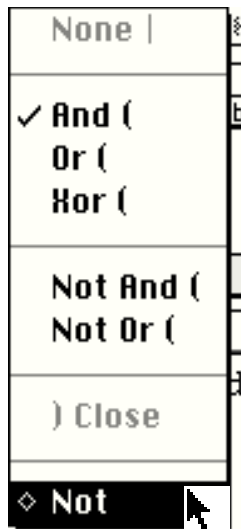
Or ( False, Or (And (True, False), True) )

are easy to figure out, given enough time. One could say, that prefix notation has the parentheses (of infix notation) built in.

## Types of Logic

Binary Pump supports the following logical operators: And, Or, Xor, Nand, and Nor. In addition, there is the Close operator which functions similarly to a close parentheses, and a Not operator which operates on the rule criteria.

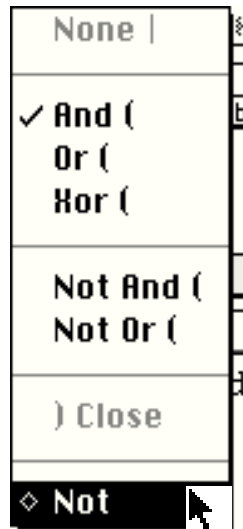




The Logic Menu

And	If all of its arguments are true, it returns true, otherwise, it returns false.	True And True = True False And True = False False And False = False
Or	If one of its arguments are true, it returns true, otherwise, it returns false.	True Or True = True False Or True = True False Or False = False
Xor	If one and only one of its arguments are true, it returns true, otherwise, it returns false.	True Xor True = False False Xor True = True False Xor False = False
Not And	Opposite of And.	True Nand True = False True Nand False = True False Nand False = True
Not Or	Opposite of Or	True Nor True = False False Nor True = False False Nor False = True
Not	Opposite of its argument (the criteria)	Not False = True Not True = False
None	No logic statement. The rule is a pure argument to other logical operators.	

## The Not Flag



Setting the Not Flag

The Not operator requires some explanation relative to Not And and Not Or. You may notice in the table above, that Not takes only one argument, whereas Not And and Not Or takes two. This is because the Not flag applies only to the criteria item — not to the logic statement. Logically, usage of the not flag would look like:

And (Not (True), True)

Which is different than the usage of Not Or, or Not And:

Nand(True, True)

Nor(False, True)

Binary Pump also displays this relation in its menus. A normal rule, without the Not flag looks as so:

And (  the Label is In Progress ☐ )



But when the Not flag is used, it looks like:

And (  the Label is not In Progress ☐ )





## Examples

And (  the Name Contains  

A simple rule which checks for file names containing 'apple'. Note that Binary Pump doesn't require corresponding parentheses; it automatically places imaginary close statements at the very end of all your rules.

And (  the Name Contains    
 the Modification Date is Days Old  

Another rule which now requires the file name contain 'apple' and also be at least seven days old. The close statement is added for clarity, but is not required.

And (  the Name Contains    
 the Modification Date is Days Old    
 the Source Folder is    
 the Type is  

A more complex rule which checks to see if a file's name contains apple, is at least seven days old, and that the file either comes from the 'Morefiles 1.1.1' folder, or it has the type code 'MMPR'.

And (  Neutral

Or (  the Type is JPEG ☐

None  the Name Ends With .jpg ☐

) Close

Or (  the Creator is JVWR ☐

None  the Comment Contains Picture

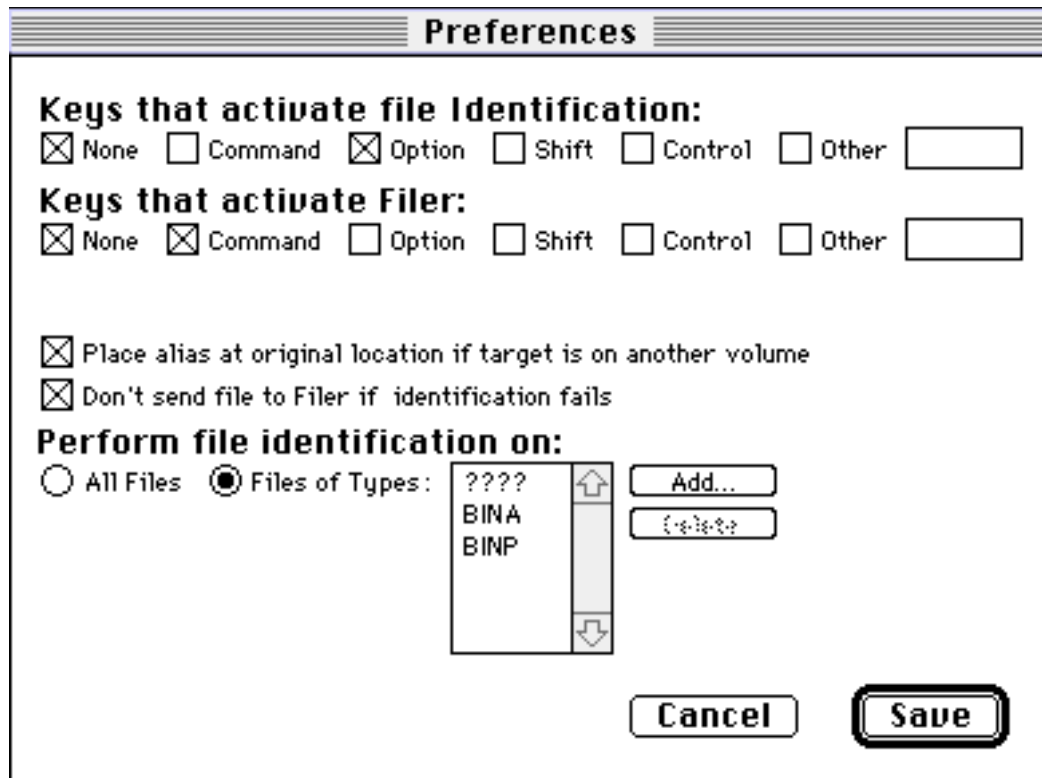
) Close

) Close

This rule demonstrates the usage of the Neutral rule. In this case the first Close rule is required; if we left it up to Binary Pump to close it for us, the Close rule would have been placed at the end, yielding different results.

## Preferences

The preferences window allows you to configure certain aspects of Binary Pump. From the preferences window, it is possible to turn File Identification or the Filer on or off. There are also several, more technical options you may change.



### Configuring Keys

Key configuration tells Binary Pump what to do with the files that you give it: Identify them, File them, or both. Binary Pump's default configuration is to do both.

The checkboxes in the preferences window have a different relationship than in other popular programs. Popular programs generally use an AND relationship. For example, if the option AND command keys are pressed, do some arbitrary action. Binary Pump, on the other hand, uses an OR relationship. For example, if either the option OR command keys are pressed, do something.

In the preferences window shown above, you will notice that File

Identification has checks in the None and Option boxes. and that the Filer has checks in the None and Command boxes. This means, that when no

keys are pressed, both File Identification and the Filer are active, when the option key is pressed, only File Identification is active, and when the command key is pressed, only the Filer is active. If both the command and option keys are pressed, both would be active.

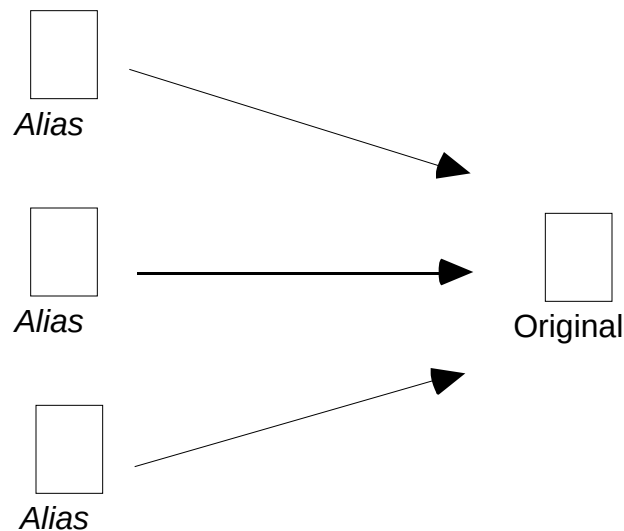
Note: Do not use the Other key option. The preference will be saved, but the option is currently ineffective.

## Placing Aliases at the Original Location

☒ Place alias at original location if target is on another volume

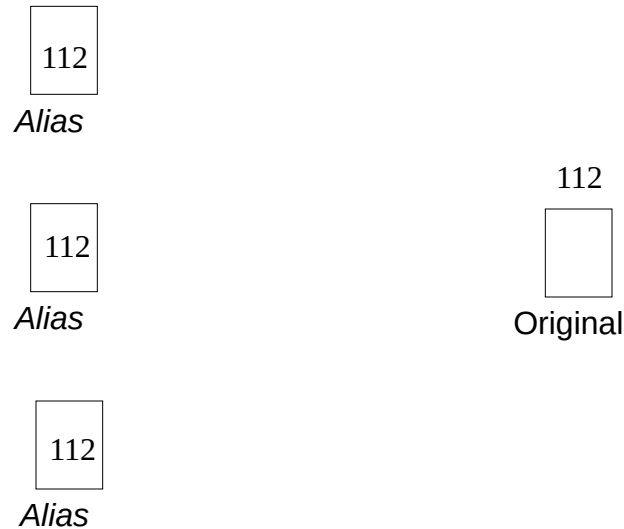
Note: This section becomes fairly technical. If it gets too dry, skip to the sections *Drawbacks* and *Alternatives*. These sections contains precautions and suggestions you should use when moving files between two different volumes.

Have you ever noticed that the Finder will not let you move files from volume to volume? This is because this type of action has major side effects. Unlike the Finder, Binary Pump allows you to move files between volumes, and tries to deal with these side effects.



Representation of Alias/Original relationship.

As shown above, aliases normally point to the original file. Aliases do this by remembering the File Identification number of the original file. Because File Identification numbers are unique within volumes, this works well.



Alias/Original relationship with a File ID of 112.

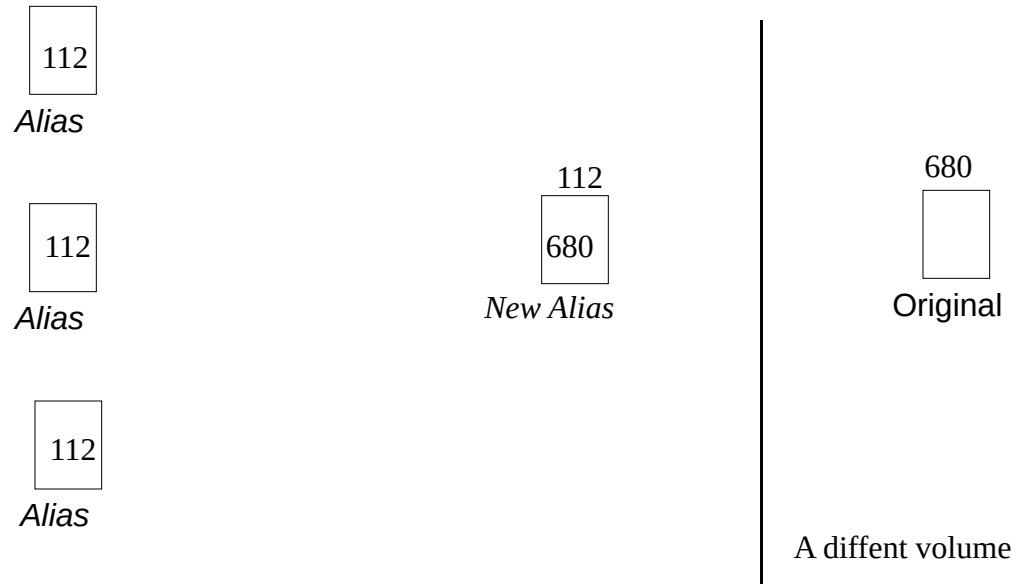
Because File ID numbers are only unique within volumes, files cannot be moved to other volumes; the File ID which the aliases have remembered would no longer match the new File ID.

One solution is to simply duplicate the file: this is what the Finder does.

Another solution is to update all the aliases. This is perhaps the best solution, but is an extremely laborious task, made even more complex by the fact that aliases may exist on multiple volumes, some of which may not be mounted (available). To update all the aliases on a one volume system could take anywhere between several minutes to a half an hour, depending on the number of files on the volume, for *each* file moved to a different volume, whether it in fact does have aliases pointing to it or not.

Yet another solution, and the one that Binary Pump uses, is to create an alias to the updated location, to which all the other aliases can point to.

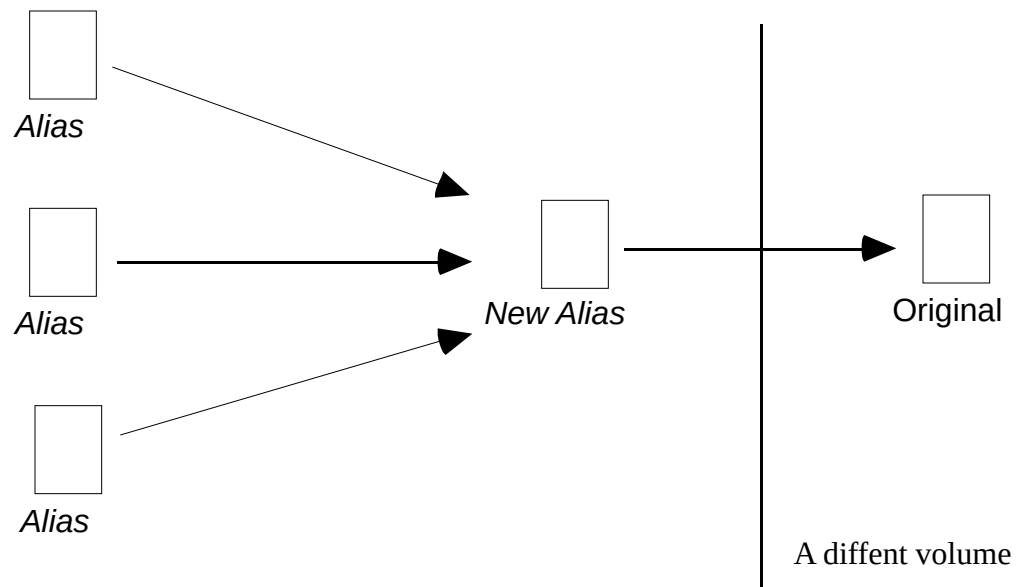




Re-assignment of file ID to an alias, which points to the new location of the file.

As you can see in the figure, the aliases are left untouched: The only change is the creation of a new alias file, which takes on the old file ID of the original file.

When the Macintosh resolves aliases, it's smart enough to follow the chain of aliases until it reaches the real file.



Note: All of the aliases which Binary Pump creates will always point to the new location.

### **Drawbacks**

With the creation of a new alias file (as can be seen in the figure on the previous page), there are several things that can break this new link. To prevent this from happening take in mind the following pointers:

- Do not delete or replace the newly created alias
- Do not repeatedly move files back and forth between the same volumes.
- Avoid moving files to removable media (MO Disks, floppy disks, removable hard disks).

### **Alternatives**

Because of the potential for problems, it would probably be wise to look for alternatives, before moving files to different volumes.

Some alternatives are:

- Instead of Moving the file, Duplicate or Alias it to the new location instead
- Make sure it is necessary to move the file to another volume.

Another alternative is to turn the “Place alias at original location” option off. Do this *only* if the files you are moving do not have aliases pointing to them. This is often difficult to discern because aliases can be anywhere. In addition, Binary Pump uses aliases to track the location of folders and to move files to multiple locations; if a file is to be moved to more than one location, the original is moved to one of them, and aliases are placed in the others (see the section *What the Filer Does* for more information). This alternative is only recommended for advanced users.

## Preventing the Filing of Unidentified Files

☒ Don't send file to Filer if identification fails

It is often desirable to not have files sent to the Filer if they are unidentified. There are a several reasons why this is so:

- Filer rules can depend on the file type or creator

Thus, the file may get moved to a location it is not supposed to be.

- If the file is moved, you may have to dig into several folders to retrieve the original in order to properly fix it.
- Aliases and Duplicates will have the wrong Creator and Type codes.

This could be horrendous if you are sending these files over a network for your boss to use.

- Files which cannot be identified may possibly be corrupted.

This point is only valid if Binary Pump is supposed to be able to identify it's particular file type. For example, if a file is a Pixar file, and you *know* it is supposed to be a pixar file, but Binary Pump cannot identify it, then something may be wrong with the file. (Pixar files are one of the file types Binary Pump is initially set up to recognize)

So why would you want to turn this option off?

- If you do not know how to fix file types/creators and do not use them in your rules.

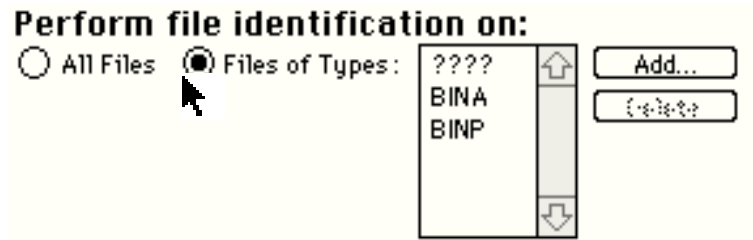
In this case, it may be better to simply send the files off anyways, and hope for the best.

- If you use the filer to screen out unidentified files (Advanced users!).

It would be possible to put these files in a folder like "Unidentified files" which you could dig through on your free time. You could also add the proper specifications to Binary Pump so that Binary

Pump would recognize the file type in the future.

## Choosing Which Files To Identify



Certain type codes are usually used for unidentified files that are received through communications or network programs. Therefore, it is desirable have Binary Pump attempt to identify these files only, and leave the others alone.

Binary Pump comes set up to identify files with the '????', 'BINA', or 'BINP' file types. One frequently used file type, which was purposely omitted is 'TEXT'. 'TEXT' is such a generally used type code for all types of files that it was not included in the list.

If the initial list of types is inadequate, you can edit the types through the **Add...** and **Delete** buttons which are adjacent to the type list. Before using the **Add** button, you should know the four digit type code you wish to add. It is possible to obtain this information through the tag window by using the **Type...** button; upon selecting the desired file, it will place the type code in the corresponding field.

So when would you ever want to use the All Files option?

- If you use the Filer and File Identification separately.
- If you know that the files you are giving Binary Pump are unidentified.
- To change the file type/creator of identified files.

See the *Applications* section for more information.

## How File Identification Works

This section is meant to be an overview of the File Identification mechanism so that you may use it more effectively; it does not go into technical detail unless necessary.

There are two methods File Identification uses to identify files:

- Look into the file and see if its contents can be recognized.
- Look at the file name extension and see if it can be recognized.

IBM PC systems use file name extensions as the standard way to identify files. Unix and other robust operating systems frequently use file name extensions also; but not always. So why bother looking into a file's guts?

The simple reason: It's better.

A file name can be easily altered, a file's identifiable contents cannot (without damaging the file). Many communications programs for example, ask the user to supply a file name in order to transfer a file. It's easy for a Macintosh user to leave the name extension off; after all, why should Mac users have to go through MS-DOS's file name nonsense?

Just the opposite of MS-DOS, Unix systems allow absurdly long file names. At times, it is convenient to rename these long file names to something shorter. This is a perfect time for the file name extension to be omitted, or mis-typed.

Therefore, when Binary Pump identifies files, it gives binary tags precedence over file name extensions. This means that even if you purposefully change the file name extension of your file to something else, Binary Pump may still insist on the file's actual file type.

**Note:** NOT ALL FILE TYPES HAVE BINARY TAGS. A perfect example of this is text files. In this case, Binary Pump is dependent on file name extensions.

During file identification, Binary Pump will check to see if any files are in the Macbinary format. If so, they are decoded and are not passed on for file identification; the entire purpose of Macbinary is to preserve the original Macintosh type codes information.

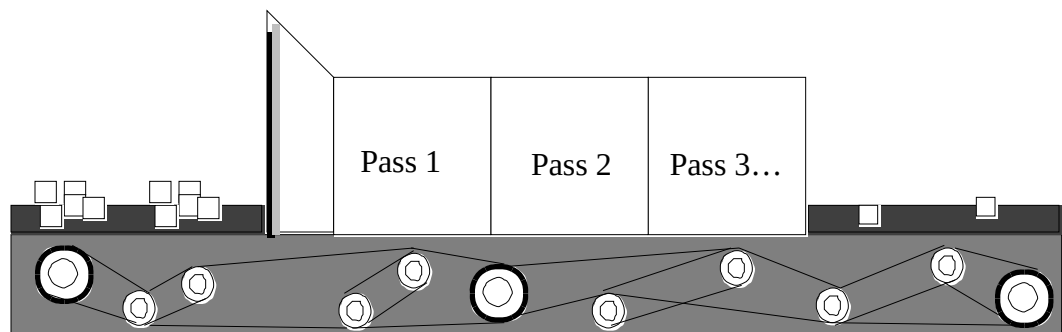
## How the Filer Works

When the Filer is given a set of files, it works somewhat like an assembly line, moving files through different stages.

### Passes

There are a total of nine passes the filer goes through. Each pass is like a separate stage which occurs one after the other. Within each pass, the same type of processing occurs.

A file which makes it through all nine passes is left undisturbed.



Example of the pass metaphor

To use an analogy: passes are similar to the editing process. A rough draft of a paper is created. Then, sections are added, moved and deleted, spelling is correct, and grammar mistakes are remedied; the first pass. Then, the process is repeated until the second pass, third pass, and so forth are completed. The outcome is usually a finely polished paper.

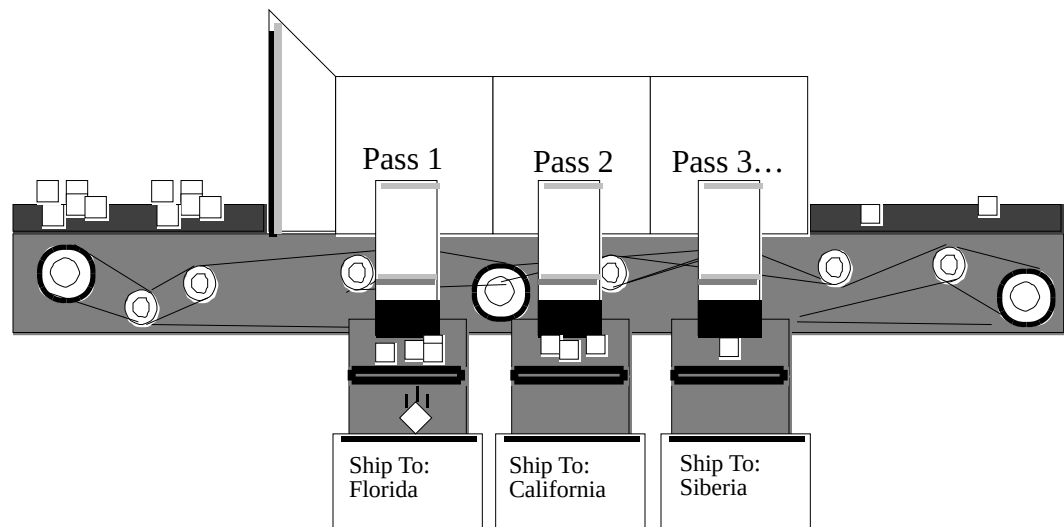
### Sets

Each pass moves through a list of applicable sets. Recall that an attribute of sets is a pass value.

Within the set, the first step is to see if the files fit the rules. The files that fit, then have the set's action performed on them (move, duplicate, or alias). In addition, these files are marked so that they do not go on to the next pass.

When moving files, the first applicable set the Filer comes across will receive the original. Subsequent sets (and thus, their destinations) will receive only an alias.

At this time, all moving and duplicating is done before aliasing, so that these aliases are always valid; even if the file is moved to a different volume.



Extended pass metaphor with sets.



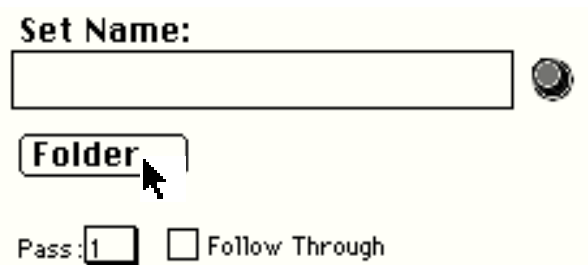
## Applications

The following sections are used to demonstrate the capabilities of Binary Pump. They are not necessarily the optimal solution to the given problems.

### A Simple File Filter/Selector

The simple file filter allows you to choose a subset of files with which to work. In this example, we want to find an appropriate graphic for a flyer. The flyer is for a new water slide park so we would like a picture of either water or slides. In addition, we have a lot of old low quality bitmap graphics mixed in with our higher quality postscript graphics. We only want to use the postscript graphics.

The first thing we do is to open the Filer window and click on the **Folder...** button.

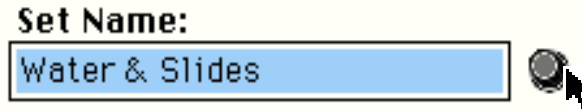


A dialog box will appear in which we can select a destination folder. Because our destination folder does not exist, we click on the **New Folder** button and enter a new folder named "Water & Slides".



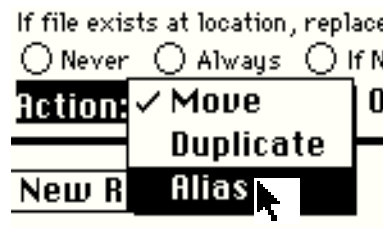
We can now select the newly created folder from the file list, and click on **Select Folder**. The destination folder has now been set up.

We will now use the name of the folder for our set, so we click on the helper button next to the set name.



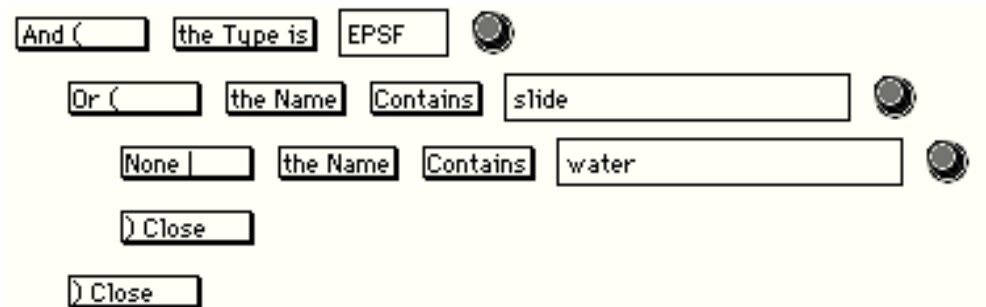
We do not want our graphic files to be processed by any other sets, so we leave the pass at its default value of 1.

Because a file filter screens out applicable files, and does not want to affect the original files, we should either duplicate or alias files. In this case, we will alias the files.



The first rule should be one to screen out all the postscript files. We use a Type rule, and click on the helper button to find that the type code of postscript files is 'EPSF'. The rule logic should be And, since we only want postscript files.

Next, we want files which contain either 'slide' or 'water' in their names. So we create two rules, one which uses the Or logic, to search for these items.



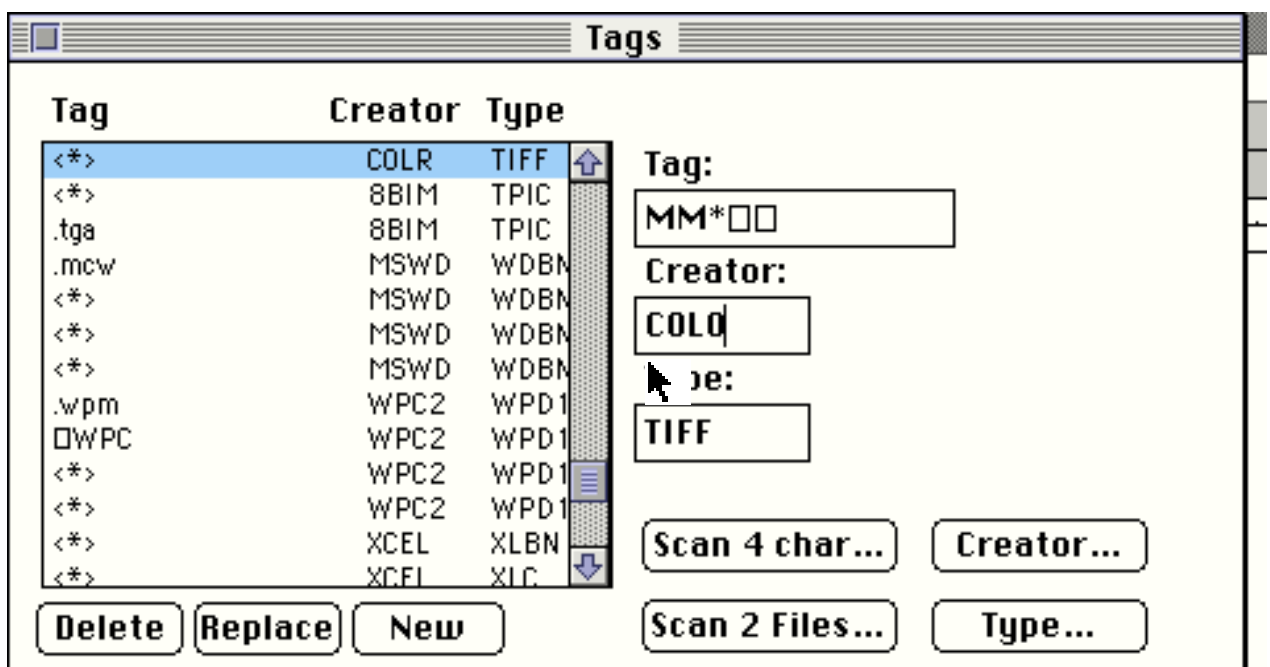
Binary Pump is now ready to filter the correct files out. We can drag our 5,000+ graphics files onto Binary Pump and receive a list of applicable files for our flyer.

## Resetting File Creators

We have a printing shop which receives many different types of files. Most of our customers like to use ColorStudio for final image corrections, whereas we use Photoshop for output. This example will concentrate on TIFF files.

Both Photoshop and ColorStudio recognize the TIFF format. But right now, if we double click on the ColorStudio files, ColorStudio will run, and open the files. We want Photoshop to do this instead.

So, we run Binary Pump, and immediately go to the Tags window. We then locate the 'TIFF' file type and click on it. The current tag information is immediately loaded. (For those of you familiar with ColorStudio, the creator type shown below is probably wrong: I don't know what it really is!)



We click on the **Creator...** button and choose Photoshop. This places the creator code '8BIM' into the creator field.

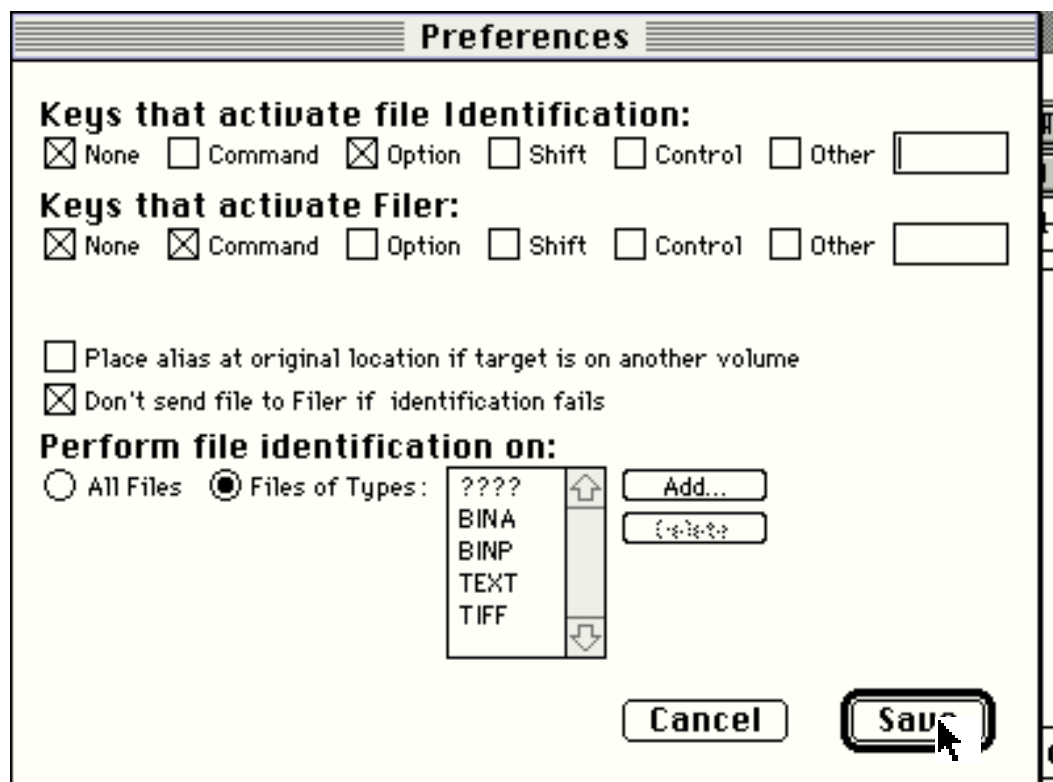
To save our changes, we click on the **Replace** button.

Note: Binary Pump actually contains *several* tag entries for the TIFF format. These other entries would also need to be changed using the same procedure.

The default setting for Binary Pump is to identify files of only certain types. Because 'TIFF' is not one of those types, we need to add it. To do this, we go into the Preferences window, and click on the **Add...** button. In the dialog that appears, we type 'TIFF' and click on **OK**.



We then click on the **Save** button.



Now, whenever 'TIFF' files are given to Binary Pump, their default application will be change to Photoshop.

## Archiving With Simple Versioning

In this example, we are computer programmers. Every once in a while, we make changes to our programs. Sometimes, in making changes, we



introduce errors, or accidentally remove something important. Thus, it is convenient to have a previous version of our program to fall back to. Once again, we venture into the Filer to accomplish this task.

As far as programmers are concerned, the main part of a program is called the source file and this is what we are going to archive.

In the Filer, we click on the **Folder...** button, create a new folder named “Source Archive”, and select it. We click on the helper button next to the set name to automatically name the set for us.

We only want a duplicate of our files to be archived, so we choose the duplicate action.

In order establish versioning, we will choose the **Rename** option by clicking on the Rename radio button.



We program in C++, so our source files will end in either .c, .cc, or .cpp. In addition, we only want the source files from a specific folder — we don’t want other peoples source files!

And (  the Source Folder is Macintosh HD...Binary Pump:Source ☐

Or (  the Name Ends With  ☐

the Name Ends With  ☐

the Name Ends With  ☐

We then click on the **New Set** button to save the set.

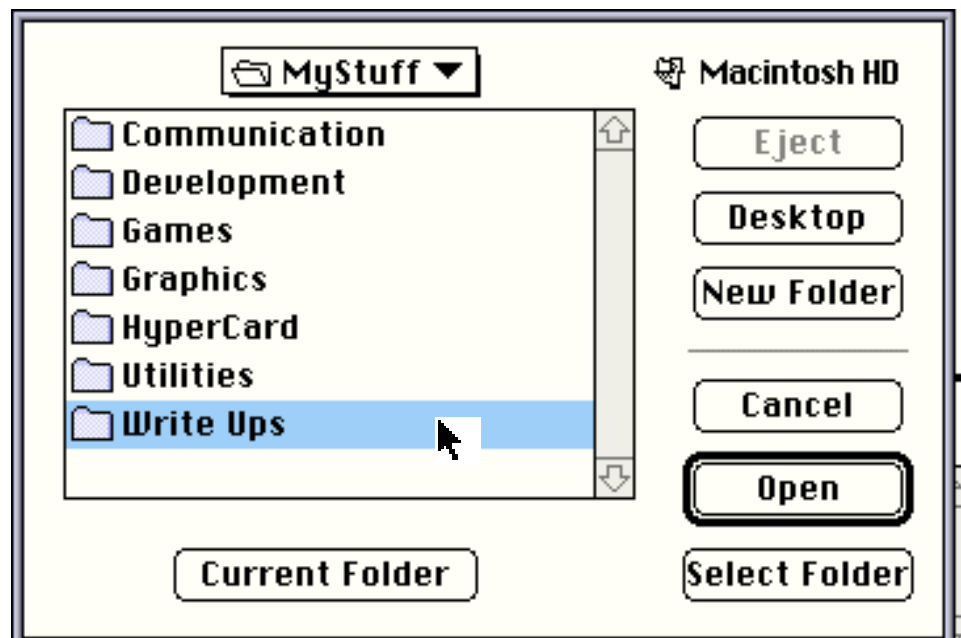
Now, whenever we give Binary Pump our source files, it will archive them and give each file a version number.

## A File Distributer

We have just finished a write up of our project, and now want to send copies to other people on the network. Everyone on this network has file sharing, so we could simply copy the files directly to their computers. But we need to do this after each revision, so after a while, it becomes an arduous task. So we will let Binary Pump take care of it.

Note: To set up Binary Pump, all destination volumes must be mounted beforehand. This is done from the Chooser. Consult your Apple documentation on mounting Appleshare/File sharing volumes.

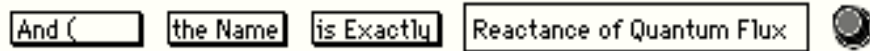
We run Binary Pump, go into the filer, and click on the **Folder...** button. We select the desired destination folder on the remote hard drive.



We only want to send a duplicate of our file, so we choose the **Duplicate** action.

Because we want everyone to have the latest copy, we choose **If Newer** under the replace options.

We only have one file we want to send, so we create a Name rule, which matches the name of our file exactly.



And now, we click on the **New Set** button.

Each destination requires a set. Thus, for each destination, we repeat only the first two steps, then create a **New Set**: all we need to do is change the destination folder and the set name.

Now, whenever we drag the file “Reactance of Quantum Flux” to Binary Pump, it will distribute a copy to all the chosen computers.

## Shuttling Newly received Files

With all the hype about the information highway, more and more people are discovering the Internet. This is a fabulous resource which has existed for many years. Experienced users of the Internet have probably discovered one thing: the Internet is predominately composed of Unix machines. If you read the section *What Does Binary Pump Do?* you would realize that we have a great potential for problems. Macbinary files should come across fine, but a great wealth of the Internet does not use Macbinary. These things include graphics , sounds , and movies. Just think, you might miss out on all the multimedia hype which ripples throughout the Internet! Fear not, because Binary Pump’s file identification capabilities will easily solve this problem.

But there is another, more subtle problem. With all these files you receive from the Internet, your hard drive will

become a jumble of incoherency. Want to hear that sound file you downloaded last month? After several hours of searching through, and opening the wrong files, you might find it. Organizing your files is what the Filer is for!

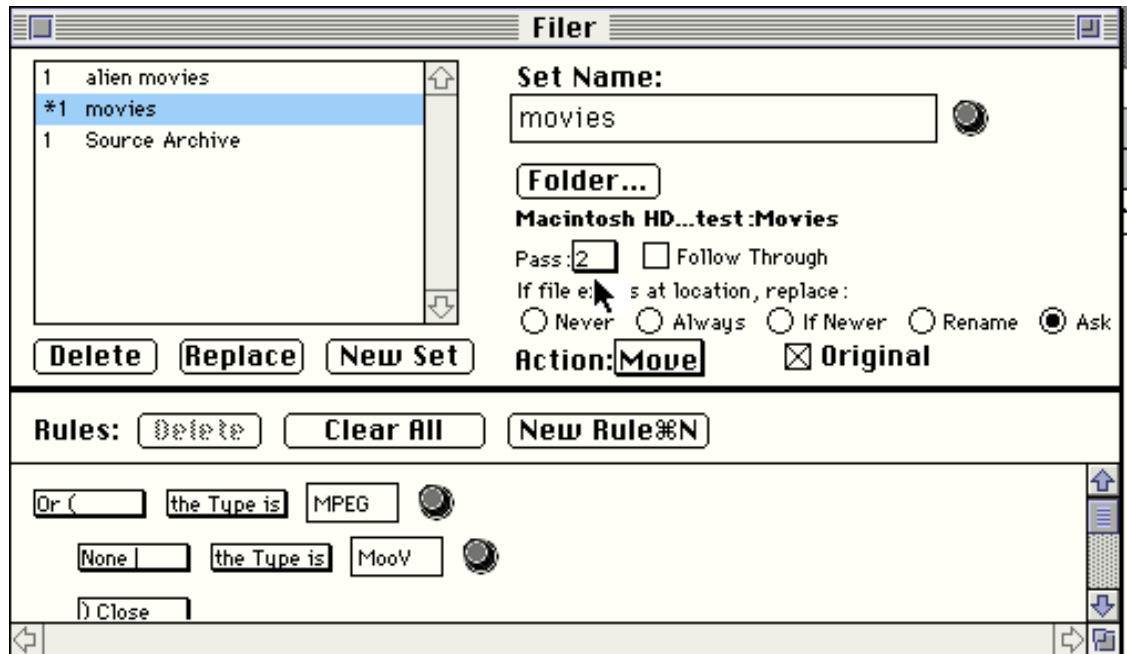
Thus, Binary Pump can identify your files and keep you organized, all in one fell swoop. Since the Binary Pump comes set up for file identification, we don't even need to set it up! The Filer requires a little work though.

This example, will place all graphics files in one folder, and all movie files in another (yes, I know I left out sound).

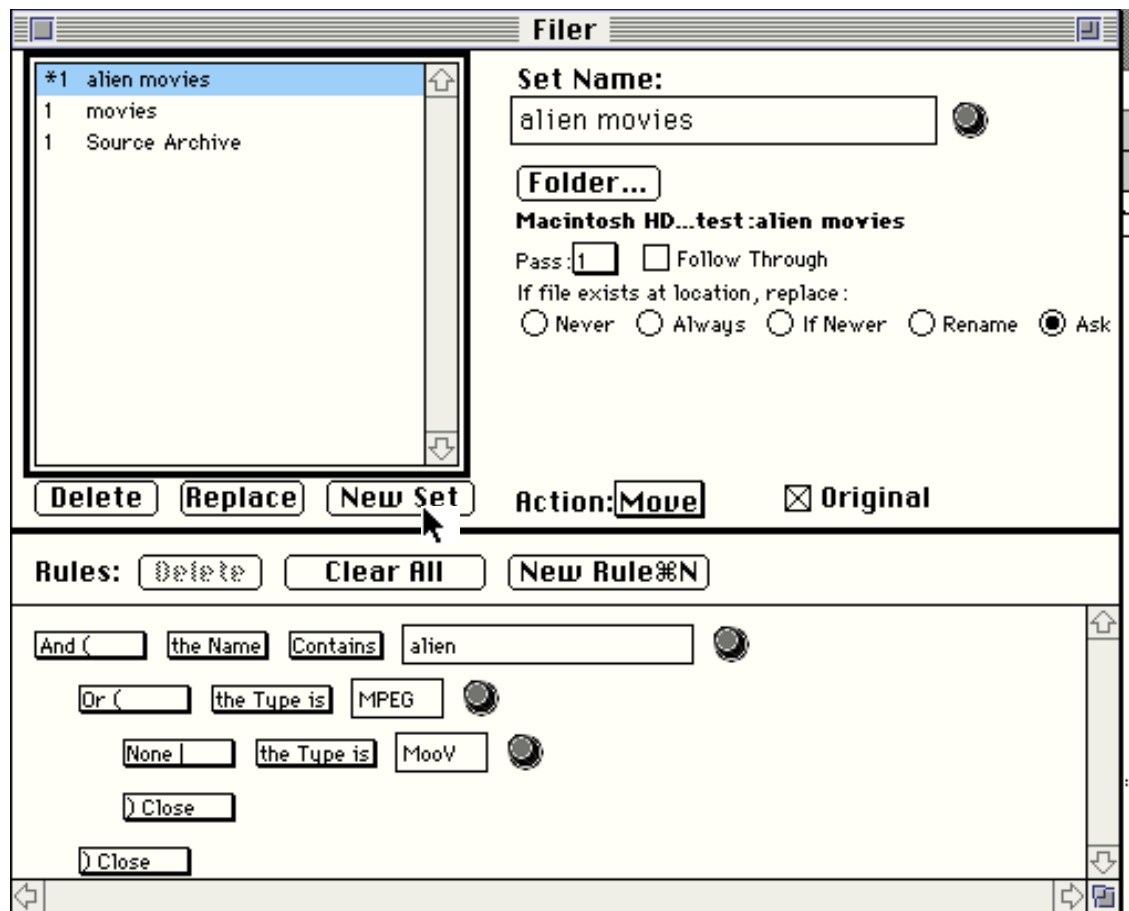
Through the **Folder...** button, we will create a new folder named "Movies", and select it. This is the same name we will use for the set.

Movies is a little general, so we will set its pass at 2.

Two standard movie formats are ‘MPEG’ and the quicktime format ‘MooV’. We will use these two types for rules and create a new set.

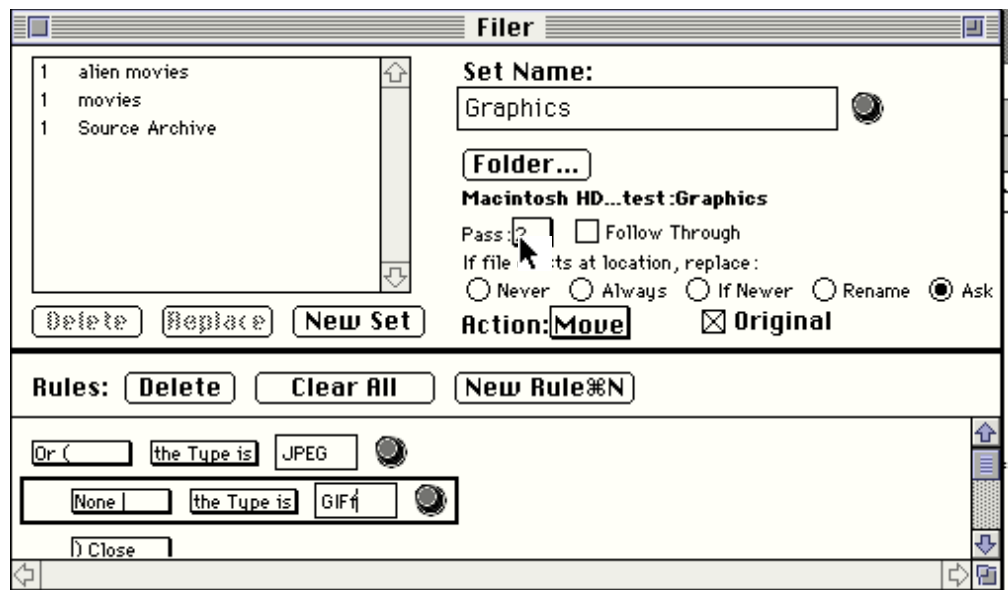


Let us say that there are special movies we want to be placed in their own folder. We love movies with aliens in them. So, we will create another set, similar to the previous, but instead, place the files in a folder called “Alien movies”. We also add the additional rule that the file name contain “alien” and we will set the pass value to 1.



For the graphics files, we will use the same procedure as with general movie files. The main different is that the popular types for graphics are 'JPEG' and 'GIFf'. These file types can easily be found by using the helper button within the Type rule and selecting the appropriate file.





Now, you can give Binary Pump your newly received files and it will 1) Identify them and 2) place your graphics files and movie files into their respective folders, giving special treatment on alien movies.

Note: Because newly received files rarely have aliases pointing to them, this is the perfect situation to disable to “Place alias at original location if target is another volume” preference.

## Tips and Tricks

### Aborting

To abort either file identification or Filer operations, hold down the command–. (period) keys simultaneously.

### Filer Keys

The filer support many key actions which replace many of s many keys may replace most mouse functions.

The tab key selects the next item in the Filer window. All other keys depend on which item is selected.

#### Set List

##### Up Arrow

Selects the previous set in the list.

Down Arrow	Selects the next set in the list.
a...z	Selects the first item in the list whose name matches the characters typed. Many character may be typed rapidly in succession to narrow the search.
Return	Selects the next item in the list which has the same name as the currently selected item.

#### Date Rules

Up Arrow	Increments the date by one day to the next valid date.
Down Arrow	Decrements the date one day.
Left Arrow	Cycles left through the date fields.
Right Arrow	Cycles right through the date fields.

## Known Problems

The following is a list of known quirks currently in Binary Pump. Programmatically speaking, these have no quick and easy solution.

- Finder comments from floppy disks cannot be read.

Avoid using the comment rule in the Filer if you plan to use Binary Pump to process files from floppy disks.

- Finder labels are not updated until the file's folder is closed then opened.

This means you cannot change the label of a file and immediately use the Filer to place the file according to that label.

You should avoid changing the label of files on the desktop.

- Duplicating/Moving original files over a Localtalk network is slow.

This has to do with Localtalk and Apple's implementation of file sharing. Ethernet should improve things.

It is still possible to place Binary Pump in the background, but your entire computer will run like molasses. I have moved several megabytes over a network with a PowerMac 7100, and things really slowed down. This has more to do with the speed of the network rather than the speed of the computer, so running things on a slower computer should yield approximately the same results.

- Updating icons over a network is slow.

Normally, when Binary Pump identifies a file, or places an alias somewhere, you will see the new icon immediately. However, this is not always the case over networks. It should take about thirty seconds to a minute for the icons to update. During this period, the files will still have all of their old properties (i.e., when you try to open them, you may still get the "Application could not be found..." error message). If you

are in a hurry, a quick way to update the icons is to close the folder that the file is in, and then open it.

- Binary Pump cannot mount Unmounted volumes.

When Filing, Binary Pump uses aliases to access destination folders and is therefore bound to the rules of aliases which cannot automatically mount volumes.

This restriction does not include removable media such as floppy disks, nor does it include Appleshare volumes.

- The Other Key options in the Preferences window does nothing.

This problem is being investigated. It is not considered a significant enough of a problem to bar Binary Pump's release.

## Appendix

<b>File Type</b>	<b>File Name Extension</b>	<b>Type Code</b>	<b>Creator Code</b>	<b>Application Name</b>
ARC	.arc	arc*	mArc	Mac Arc (?)
BinHex	.hqx	TEXT	SITx	Stuffit Expander
BMP	.bmp	BMP	8BIM	Photoshop
C header	.h	TEXT	MMCC	Metroworks 68K C++
C source	.c	TEXT	MMCC	Metroworks 68K C++
C++ source	.cc, .cpp	TEXT	MMCC	Metroworks 68K C++
Compactor	.cpt	PACT	CPCT	Compactor
Encapsulated Postscript	.eps	EPSF	8BIM	Photoshop
Excel Chart	.xlc	XLC	XCEL	Microsoft Excel
Excel Macro	.xlm	XLM	XCEL	Microsoft Excel
Excel Spreadsheet	.xls	XLS	XCEL	Microsoft Excel
Excel Text		TEXT	XCEL	Microsoft Excel
Filemaker Pro		FMPR	FMPR	Filemaker Pro
Freehand 3.0		FHA3	FHA3	Freehand 3.0
Freehand 4.0		FHA4	FHA4	Freehand 4.0
GIF	.gif	GIFf	CGIf	Cyber GIF
Gnu Zip	.gz	Gzip	Gzip	????
IFF	.iff	ILBM	8BIM	Photoshop
JPEG (JFIF)	.jpg	JPEG	JVWR	JPEG View
Microsoft Word	.mcw	WDBN	MSWD	Microsoft Word
MPEG	.mpg	MPEG	mMPG	Sparkle
Packit	.pak	PAK	PAK	Packit

PageMaker 4.0		ALD4	ALD4	Pagemaker 4.0
PageMaker 5.0		ALD5	ALD5	Pagemaker 5.0
PCX	.pcx	PCX	8BIM	Photoshop
Photoshop 2.5		8BPS	8BIM	Photoshop
Picture	.pct	PICT	8BIM	Photoshop
Pixar	.pxr	PXR	8BIM	Photoshop
Quicktime		MooV	TVOD	Simple Player
Stuffit	.sit	SIT!	SIT!	Stuffit
tar	.tar	TARF	TAR	????
Targa	.tga	TPIC	8BIM	Photoshop
Text	.txt	TEXT	ttxt	Teachtext
TIFF	.tif	TIFF	8BIM	Photoshop
Unix Compress	.Z	ZIVU	LZIV	????
WordPerfect		WPD1	WPC2	WordPerfect
ZIP	.zip	pZip	pZip	Unzip
Zoo	.zoo	ZOO	BOOZ	????



## Aborting 53

- Action 19
- Add... 39
- Alias 19
- Always 22
- And 29
- Applications 43
- Archiving 46
- Ask 22
- Background Operation 7
- Case Insensitive 24
- Case Sensitive 24
- Comment Rules 26
- Communications Software 8
- Compactor 1
- Contains 24
- Contains Rule 24
- Creators
  - Resetting 45
- Creator... 16
- Days Old 25
- Delete 15, 39
- Disclaimer iii
- Distributing Files 48
- Distribution iii
- Duplicate 19
- Edit Tags... 12
- Ends With 24
- Exact 24, 25, 26
- E-mail vii
- File Identification 3, 5, 40
- File Name Extensions 3
- File Size Rules 25
- Filer 41-42
  - Keys 53
  - Set List 53
- Filing 3, 5
- Filter 43
- Follow Through 21
- Get Info 26
- Greater Than 24, 25, 26
- Helper Buttons 23
- IBM PC 3
- If Newer 22
- Infix Notation 28
- Installing 1
- Is Folder Rules 27
- JPEG 14
  - JPEG View 14

- Keys 33
- Label Rules 27
- Less Than 24, 25, 26
- Locked Rules 27
- Log 7
- Logic
  - And 29
  - Combinations 16
  - Examples 31
  - None 29
  - Not 29
  - Not And 29
  - Not Or 29
  - Or 29
  - Statements 27
  - Types 28-32
  - Xor 29
- Macbinary 3
- Microphone 8
- Minutes Old 25
- Model files 12
- Modification Date Rules 25
- Modifiers 23, 24
  - Case Insensitive 24
  - Case Sensitive 24
  - Contains 24
  - Days Old 25
  - Ends With 24
  - Exact 24, 25, 26
  - Greater Than 24, 25, 26
  - Less Than 24, 25, 26
  - Minutes Old 25
  - Starts With 24
- Move 19
- Name Rule 23
- Neutral Rule 32
- Neutral Rules 27
- Never 22
- New 15
- New Rule 19
- New Rules 21
- None 29
- Not 29, 30
- Not And 29
- Not Or 29
- On 33
- Or 29
- Original 21
- Other Key 34

- Passes 21, 41
- Photoshop 14
- Placing Aliases 34
- Preferences 33-39
  - Keys 33
- Prefix Notation 28
- Preventing Filing 38
- Problems 54
- Rebuilding the desktop 1
- Rename 22
- Replace 14, 15
- Replacing 22
- Requirements 1
- Resetting File Creators 45
- Rules 16
  - Contains 24
  - Date Rule Keys 53
  - Manipulation 22
  - Types 22
- Scan 2 Files... 15
- Scan 4 Char... 15
- Selector 43
- Sets 17, 41
  - Attributes 21
  - Creating 18
  - Deleting 20
  - Editing 20
- Shareware iv
  - Fee vi
- Shuttling Files 49
- Source Folder Rules 27
- Starts With 24
- Stuffit Expander 1
- Tags
  - Creating 12
  - Editing 14
- Tags window 12
  - Buttons 15
- Time 25
- Type Codes 14, 39
- Type Rules 23, 24
- Type... 15
- Undo 20
- Unix 3
- Versioning 46
- VMS 3
- Xor 29

Zterm 8