



**VERSION 3.2**

# **USER MANUAL**

Copyright © 2002 by the author, Joël François

PageSucker Launcher for Windows written by Eric J. François

PageSucker was originally conceived at the New Media Group, a division  
of the Henri Tudor Public Research Center, Luxembourg.

# Table Of Contents

<b>1. WHAT IS IT? - AN INTRODUCTION AND SOME BACKGROUND INFORMATION</b>	<b>3</b>
<b>2. GETTING STARTED – A BASIC TUTORIAL</b>	<b>5</b>
2.1    EXAMPLE 1 – DOWNLOADING A COMPLETE WEB SITE	5
2.2    EXAMPLE 2 – DOWNLOADING SELECTED DATA FILE TYPES ONLY	7
<b>3. THE MAIN CONTROLS</b>	<b>12</b>
<b>4. WHEN TO STOP? - THE END OF A DOWNLOAD PROCESS</b>	<b>17</b>
<b>5. THE SETTINGS MENU</b>	<b>19</b>
5.1    THE FILE TYPES WINDOW	19
5.1.1 <i>Some General Usage Notes</i>	20
5.1.2 <i>The Table Columns</i>	21
5.1.3 <i>The Checkbox Options</i>	23
5.1.4 <i>Adding, Editing and Removing File Types</i>	24
5.2    THE HTML FILES WINDOW	25
5.3    THE DATA FILES WINDOW	29
5.4    THE REALAUDIO FILES AND MPEG LAYER 3 FILES WINDOWS	31
5.5    THE SCRIPT WINDOW	35
5.6    THE LOCAL FILES WINDOW	37
5.7    THE PATTERN MATCHING WINDOW	41
5.7.1 <i>Some words on DOS Patterns</i>	42
5.7.2 <i>Some Words On Regular Expressions</i>	43
5.8    THE MISCELLANEOUS WINDOW	44
5.9    THE AUTHENTICATION WINDOW	48
5.10    THE ERROR HANDLING WINDOW	50
5.11    SAVING AND RESTORING SETTINGS	51
<b>6. HOW PAGESUCKER APPLIES FILTERS</b>	<b>53</b>
<b>7. THE FILE MENU</b>	<b>55</b>
7.1    REGISTERING PAGESUCKER	55
7.1.1 <i>Registering online</i>	56
7.1.2 <i>How to use your registration code</i>	57
7.2    CHECKING FOR A NEW RELEASE	57
7.3    ADJUSTING THE APPLICATION'S PREFERENCES	59
7.3.1 <i>General Preferences</i>	59
7.3.2 <i>Proxy And Firewall Settings</i>	60
<b>8. THE WINDOW MENU &amp; THE LOG WINDOW</b>	<b>62</b>
<b>9. KNOWN BUGS AND LIMITATIONS</b>	<b>63</b>
<b>10. CONTACT INFORMATION</b>	<b>65</b>
<b>11. THE LEGAL STUFF</b>	<b>66</b>
<b>12. INDEX</b>	<b>67</b>

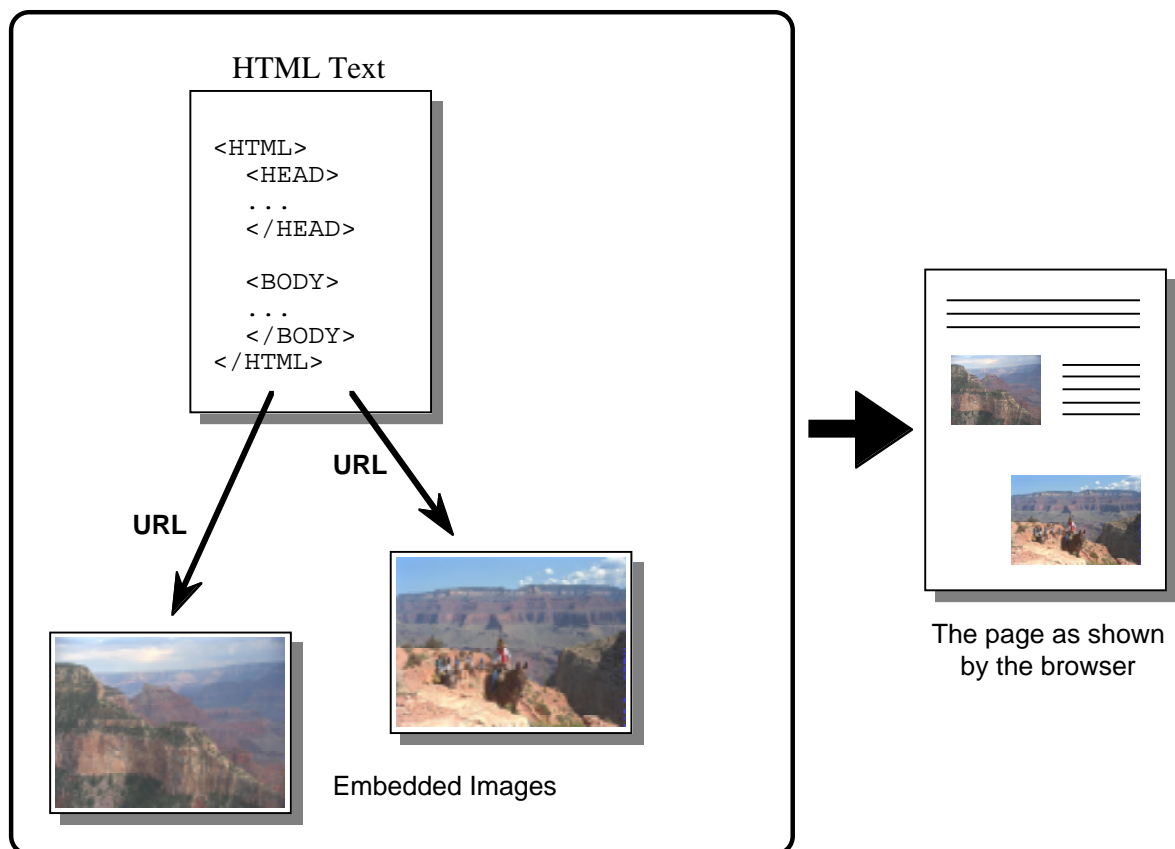
# 1. WHAT IS IT? - AN INTRODUCTION AND SOME BACKGROUND INFORMATION

PageSucker is a utility to download entire Web page hierarchies automatically to a local machine, so that they can be viewed later on while offline.

Let's take a look at how Web pages are organized on a Web server and how they can be located via URLs. A Uniform Resource Locator (URL) is a string that uniquely identifies one single object on the World Wide Web, be it an HTML page, a graphic image or some data file. A typical URL is composed of four parts: the protocol specification (e.g. HTTP, FTP etc.), the server address (e.g. "www.pagesucker.com"), a file path in the server's filesystem, and a file name.

http://	www.example.com	/pics/color/	tree.jpg
<i>protocol</i>	<i>server address</i>	<i>file path</i>	<i>file name</i>

To view a Web page, the user enters the page's URL into her browser. The browser then constructs the page's display out of various components: the page's HTML text (the skeleton of the page), possibly some embedded images, a background image and maybe even sound or MIDI files for background music. While all of these components seem to be part of one entity (the page), they are in fact separate files on the server, each of which has its own URL. These external components are connected to the page via their URLs, which are contained inside the page's HTML text.



Furthermore, a Web page usually contains links to other Web pages (usually shown by the browser as underlined text items that can be clicked by the user). Once again, these other Web pages are referred to by their URLs, even though this happens internally and the user doesn't have to enter the URLs manually to access these pages.

Most browsers allow the saving of a Web page being viewed to a local disk. However, most of the currently available browsers only save the HTML text (the skeleton) of the page, omitting embedded images as well as other pages linked to the base page. When viewed offline, these saved pages then lack their images, and links to other pages won't work as expected. This is what PageSucker was created for: although it will not graphically display Web pages (as a browser would), it is capable of downloading complete Web pages (the HTML text plus all embedded images) and even continue this process recursively for linked Web pages.

## 2. GETTING STARTED – A BASIC TUTORIAL

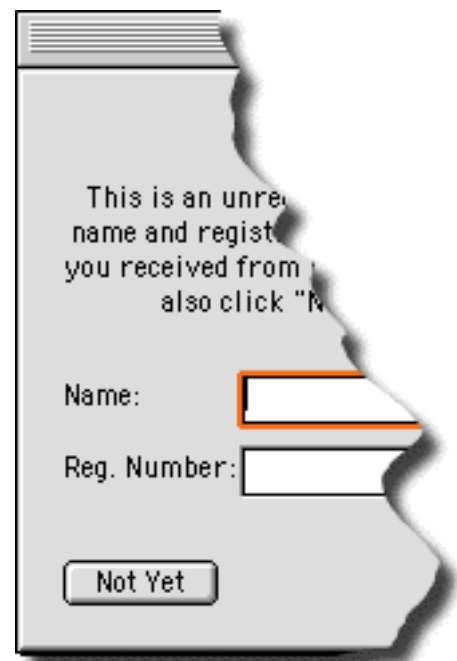
The remaining sections of this manual are to be considered a technical reference and may thus be quite difficult to understand to the novice user, due to the technical details they contain. To help you get an easy start, even if you don't understand all the details in those sections, the following tutorial has been included. It describes without much technical fuss the steps to take to perform some common types of downloads. Note: all screenshots are from the Macintosh version; the Windows version may look slightly different, but the functionality is identical.

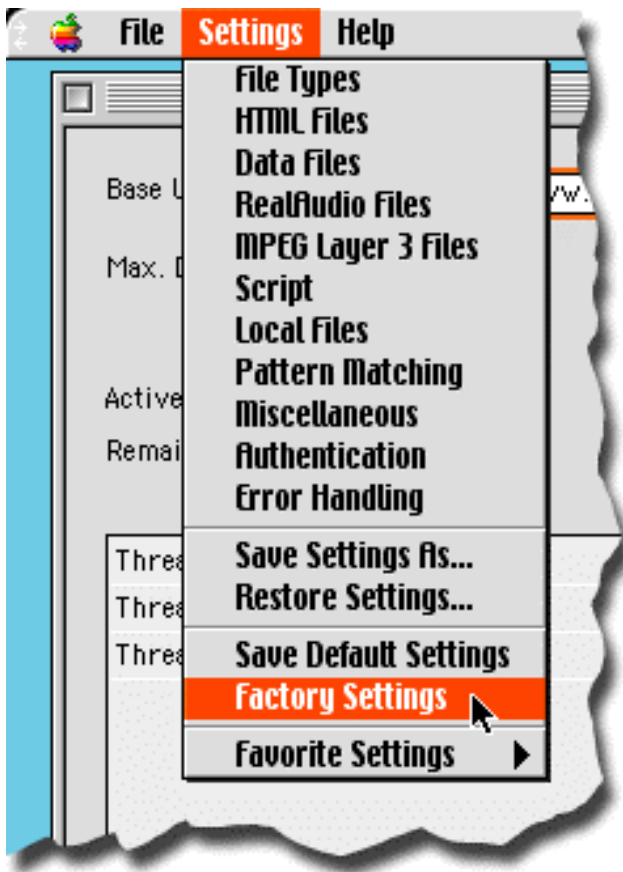
### 2.1 Example 1 – Downloading A Complete Web Site

Ok, let's get started. Locate the PageSucker icon and double-click it to launch PageSucker:



After a few seconds (depending on your system's speed) you should see the main window. If you're using an unregistered version of PageSucker (you haven't paid the shareware fee yet), you will be asked to enter a registration code. For the moment, just click "Not Yet" to use the demo version:





The following step is not necessary if you have never saved your own default settings, but let's do it anyway, just to be on the safe side: select "Factory Settings" from the "Settings" pull-down menu. This makes sure you are using the recommended default settings.

Next, enter the URL to download into the "Base URL" field in the main window:



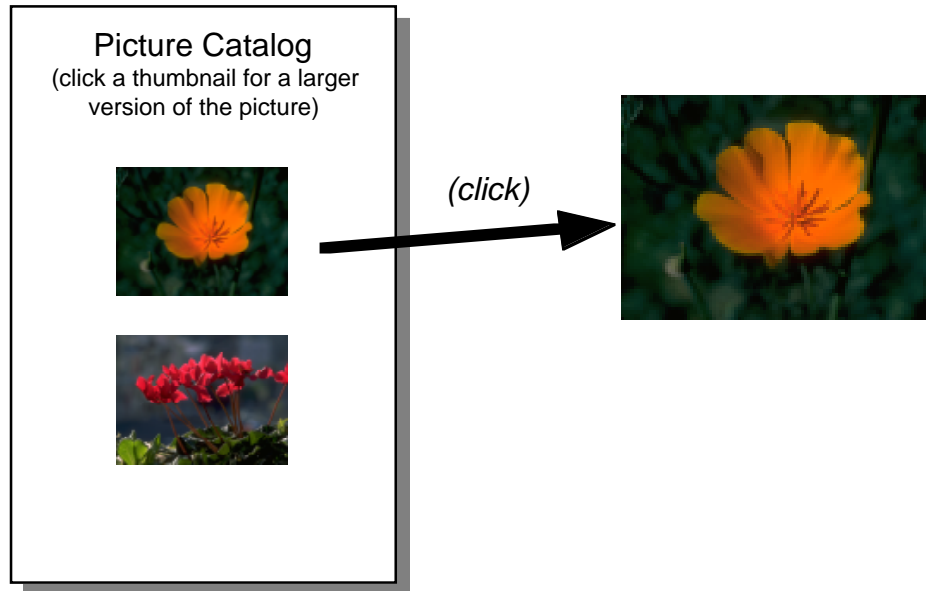
The URL displayed here is only an example which will not work for real. So choose a URL of a real server you are interested in. Then click the "Start Download" button.

PageSucker will ask for a download directory via a standard system file dialog. Navigate to an empty directory on your disk, open it and click "Save". PageSucker will now start the download, saving the page hierarchy starting with the file "index.html" in the directory "text" on the server "www.example.com" to the directory you selected on your local disk. The entire Web pages will be saved, including embedded pictures and most data files linked to the pages, provided they reside on the same server, i.e. "www.example.com".

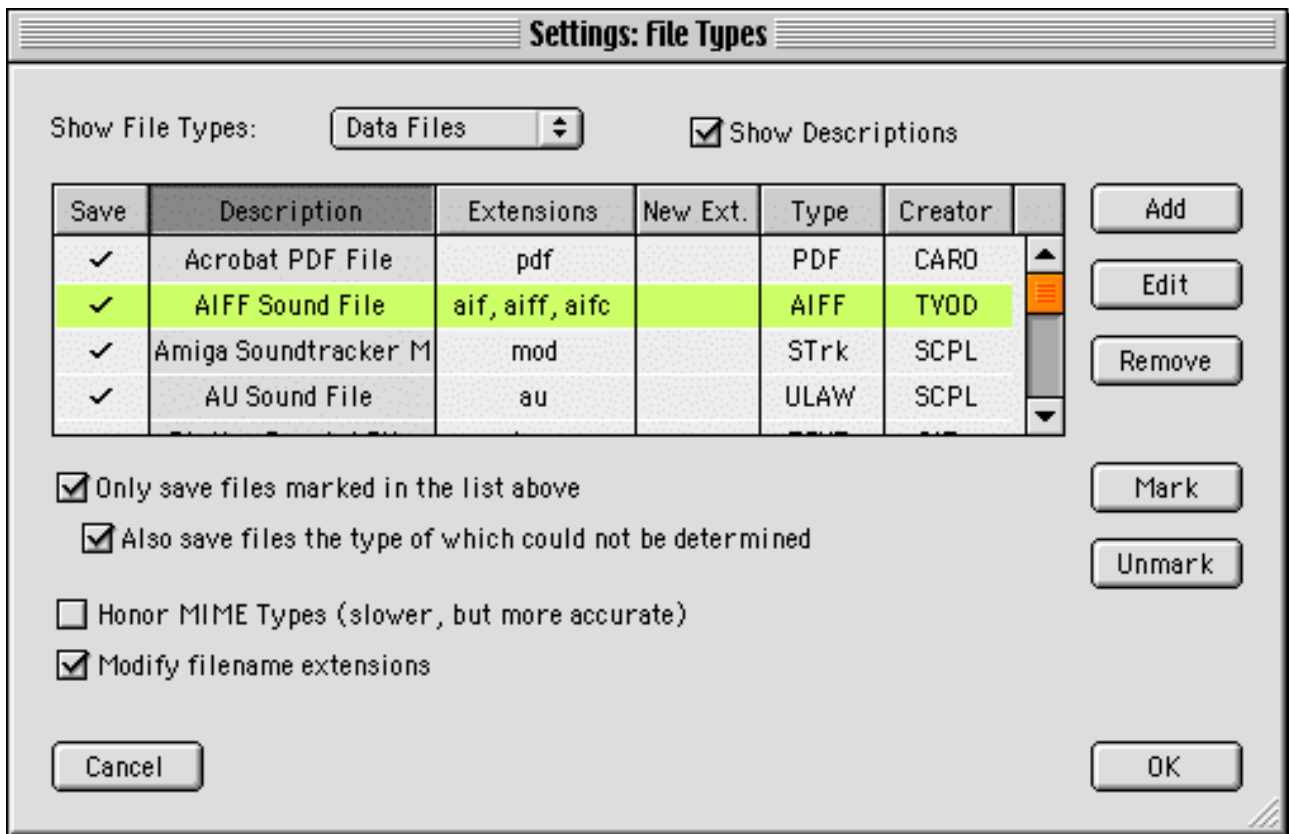
When the download is complete, PageSucker will emit a beep and display a dialog informing you of the end of the download process. You will then find a directory hierarchy on your local disk which reflects the directory structure of the remote server. At the top level of the download folder on your local disk, there will be a file called "index.html". You can open that document from your favorite Web browser to view the downloaded Web site.

## 2.2 Example 2 – Downloading Selected Data File Types Only

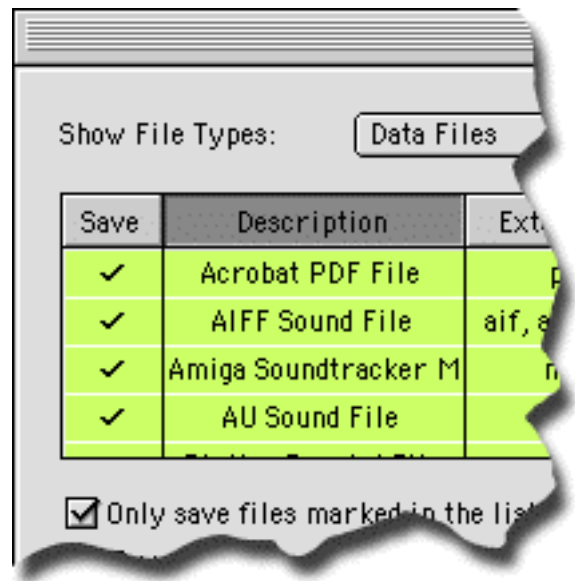
Let's suppose you have the URL of a Web page containing a picture catalog in the form of picture thumbnails (small versions of the actual pictures). You can download a bigger version of each picture by clicking the corresponding thumbnail. All pictures are of type GIF or JPG.



Let's suppose further that you'd like to download only the big pictures, but neither the thumbnails nor the HTML page itself. Moreover, let's assume that you don't want to download other pages that might be linked to this page. You can do all this with PageSucker; there are just a couple of settings that need to be adjusted. To begin with, select "File Types" from the "Settings" menu to open the *File Type Settings* window:



This is the list of data file types known to PageSucker<sup>1</sup>. We need to tell the application to only save GIF and JPEG files and to ignore other file types. To do this, first select all of the types in the table: click on some table row to highlight it, then press the keys Ctrl-A (on Windows) or Command-A (on Macintosh; the Command key is the one with the Apple and clover leaf symbols on it). Alternately, you could click on the top row, then scroll to the bottom of the list and click the bottom row while holding down the Shift key.

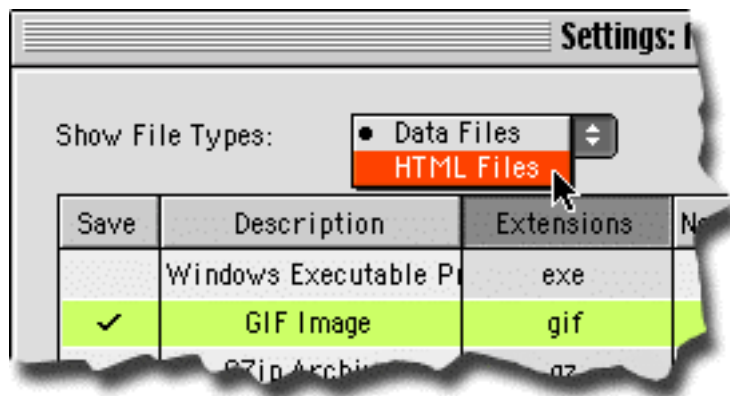
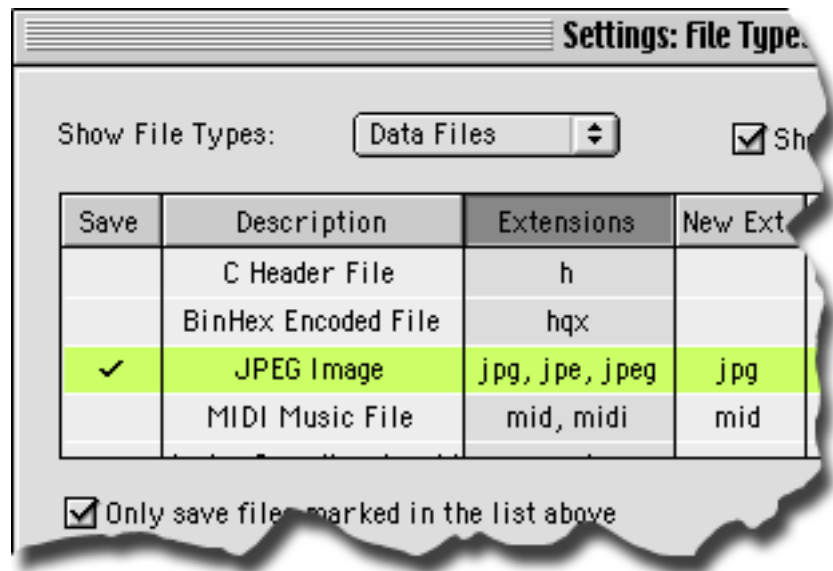


Once all rows are highlighted, click the “Unmark” button to remove the checkmarks in the “Save” column. Ok, now that all types have been unchecked, we need to find the JPEG and GIF types in the list in order to turn them back on. Click the “Extensions” header of the table; this will sort the table by filename extensions. Quickly type the letters “jpg” on your keyboard to make the table jump to the entry for JPEG files and automatically select the corresponding row. Of course you could also simply scroll down manually to find and select that row.

<sup>1</sup> The table columns called “Type” and “Creator” are only visible on Macintosh computers, as these concepts are Mac specific.

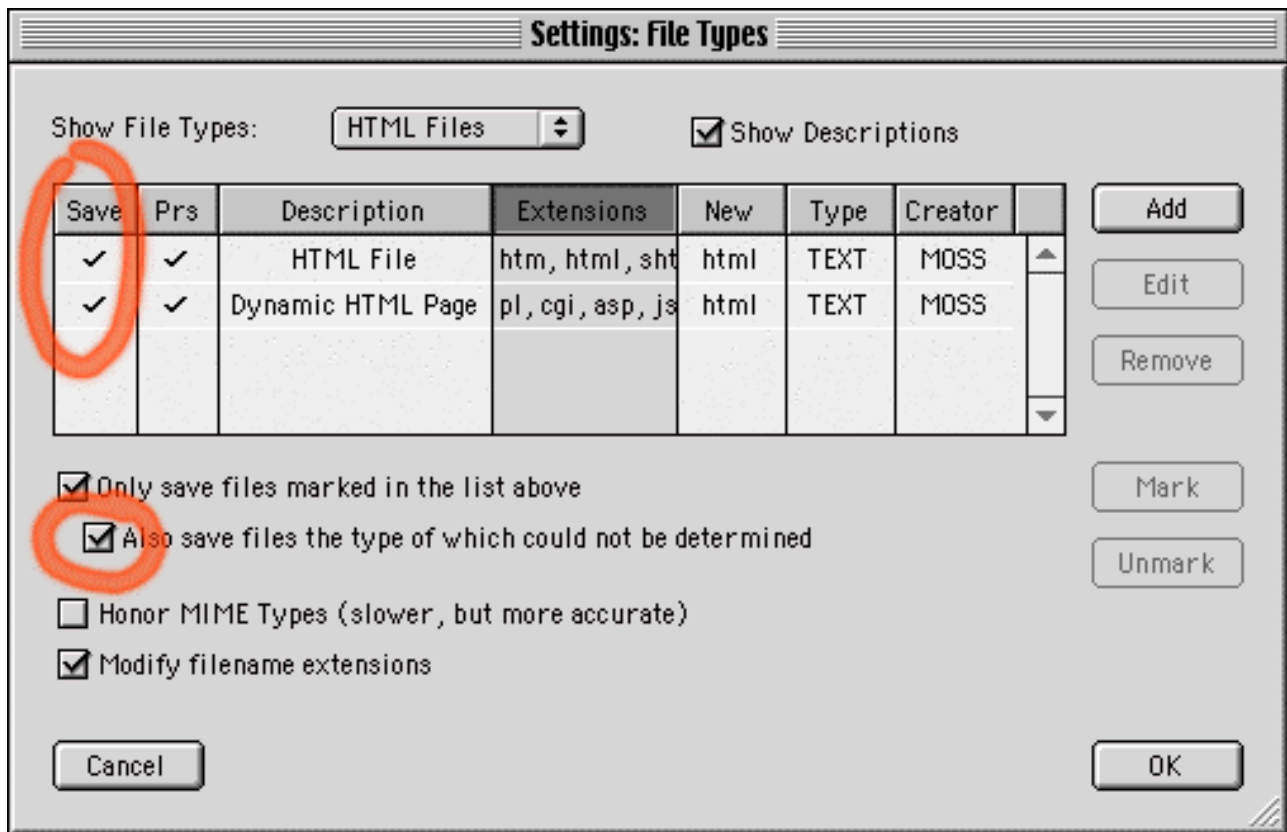


With the row selected, click the “Mark” button to place a checkmark in the “Save” column. Alternately, you could move your mouse pointer to the center of the “Save” column for the JPEG row and click the left mouse button to turn the checkmark on or off. Don’t worry, all of this sounds more complicated than it is, once you get used to it. Now repeat this procedure for the GIF file type to place a checkmark there too.



Ok, now PageSucker knows that we want to save only JPEG and GIF files, and no other data files. However, we still need to instruct the application to also skip HTML files, as those files are special and thus not shown in the data files list. To view the HTML file types, select “HTML Files” from the menu at the top of the window.

The list will update to show those file types considered to represent HTML pages:

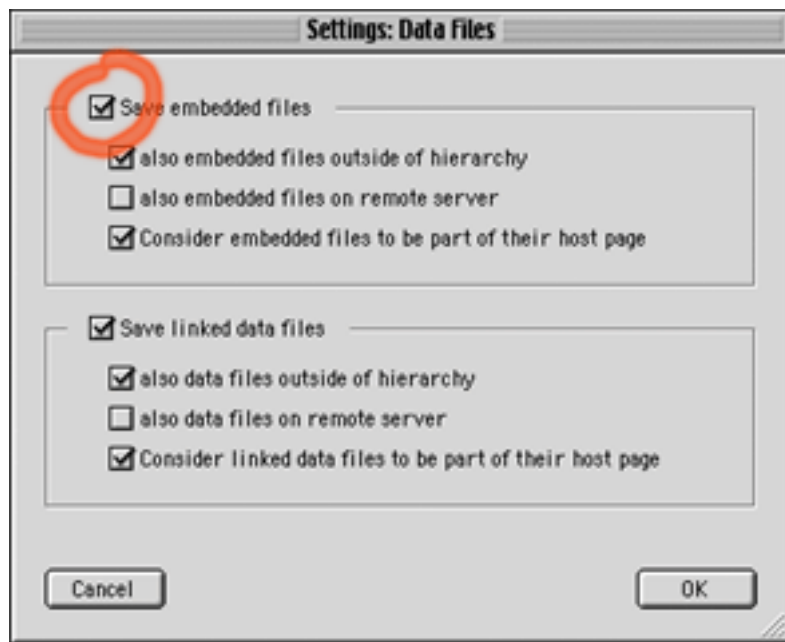


Click the checkmarks in the “Save” column to remove them (but don’t touch the checkmarks in the “Parse” column!). Finally, in order to make sure that PageSucker doesn’t save any files of types it doesn’t know, uncheck to the option “Also save files the type of which could not be determined”. You can now click the “OK” button to close the *File Types* window.

Now that we have defined which types of files to download, let’s make sure the thumbnail pictures will not be downloaded along with the big pictures. It is easy to distinguish the thumbnails from the big images, as they are embedded in the page, rather than available via a click. So let’s open the *Data Files* window: Select “Data Files” from the “Settings” menu.

Here, you need to uncheck the option called “Save embedded files”. This prevents the thumbnails from being downloaded. Furthermore, the “Save linked data files” option must be checked (but it probably already is, as this is the default).

Close the *Data Files* window by clicking the “OK” button.



In the main window, you’ll have to enter the URL of the picture catalog page:



Moreover, you must enter a “0” (zero) in the “Max. Depth To Dig” field. This prevents PageSucker from jumping to other linked pages as it specifies that only the components of the base page should be downloaded.

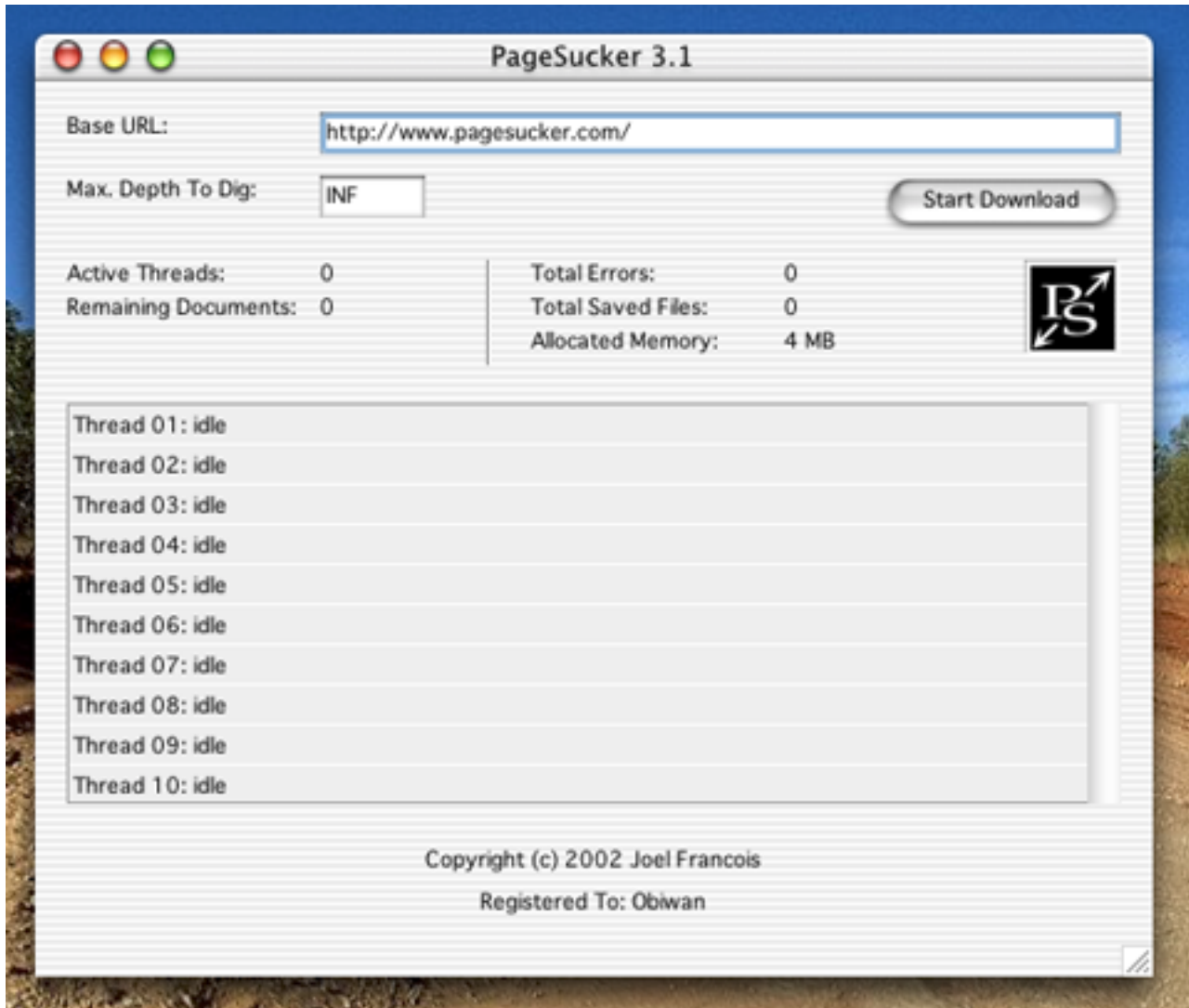
Once all these settings are established, you may press the “Start Download” button to start the download. When it is complete, you’ll find a mirror of the server directory structure on your local disk, but only the big versions of all linked JPEG and GIF pictures will be present. There will be no HTML files or other kinds of data files.

This is a good example of how PageSucker’s filters can be used to control exactly what should be downloaded. It also shows that it is always helpful to know the structure of the page(s) to download in order to set PageSucker’s options. Sometimes, it may even be necessary to take a look at the HTML source code to determine exactly why a download didn’t work as expected. Also, it may be necessary to try a few times to get all the options absolutely right.

As mentioned before, the rest of the manual is structured like a reference manual. It is thus not necessary to read it entirely before using PageSucker, but it can provide detailed answers to specific questions when needed.

### 3. THE MAIN CONTROLS

When PageSucker is launched, the main window appears (the picture shows the window as it looks on a Macintosh running MacOS X; on a other platforms it will look slightly different, but the functionality is exactly the same):



The top section contains the main controls and some status information (detailed below), the bottom section contains the thread status view.

#### ◆ **Base URL:**

The URL of the initial page. This page will be analyzed first. Pages linked to it will then be treated in the same way, until no more new links liable to be downloaded can be found or the maximum recursion depth is reached. This URL also determines the hierarchy to be scanned (see section 4 below for more details).

The URL must be a fully qualified URL, i.e. it must contain the protocol specification and the server name (“http://www.example.com/file.html” instead of just

“www.example.com/file.html”).

URLs pointing to a file on the local disk can also be entered. These are URLs starting with the protocol specification “file:”. On Windows systems, which commonly use drive letters followed by a colon to designate local disks (e.g. “C:”), these URLs take a slightly different form for PageSucker than for certain other browsers. The correct URL syntax for PageSucker is

**file:/C:/some\_path/some\_file.html**

as an example to designate the file “some\_file.html” in the directory “some\_path” on the disk “C:”.

◆ **Max. Depth To Dig:**

Sets the maximum number of links to follow in a row. This value is also sometimes called the maximum recursion depth. See section 4 below for more details on the recursion depth. The default value for this field is the special value “INF”, which specifies an infinite maximum depth.

Some examples: To download nothing but the base page, you would specify a maximum depth of zero. A depth of one (1) would download the base page plus all pages directly linked to it. Indirectly linked pages (i.e. for example pages linked to a page which is itself linked to the base page) would not be downloaded in that case though. The special value “INF” (for “infinite”) effectively disables recursion depth checking.

◆ **Active Threads:**

The number of “threads” currently active, i.e. the number of downloads currently happening in parallel. This field cannot be edited – it is intended as a status display for the application. You could consider a thread to be a kind of robot whose task it is to download or analyze exactly one URL. PageSucker contains a number of such robots, who can all work independently from each other. The number of threads is limited and can be set via the *Miscellaneous* settings window (see section 5.8 below).

◆ **Remaining Documents:**

The number of pages scheduled to be analyzed (and maybe saved) as soon as a thread becomes available. This field cannot be edited – it is intended as a status display only.

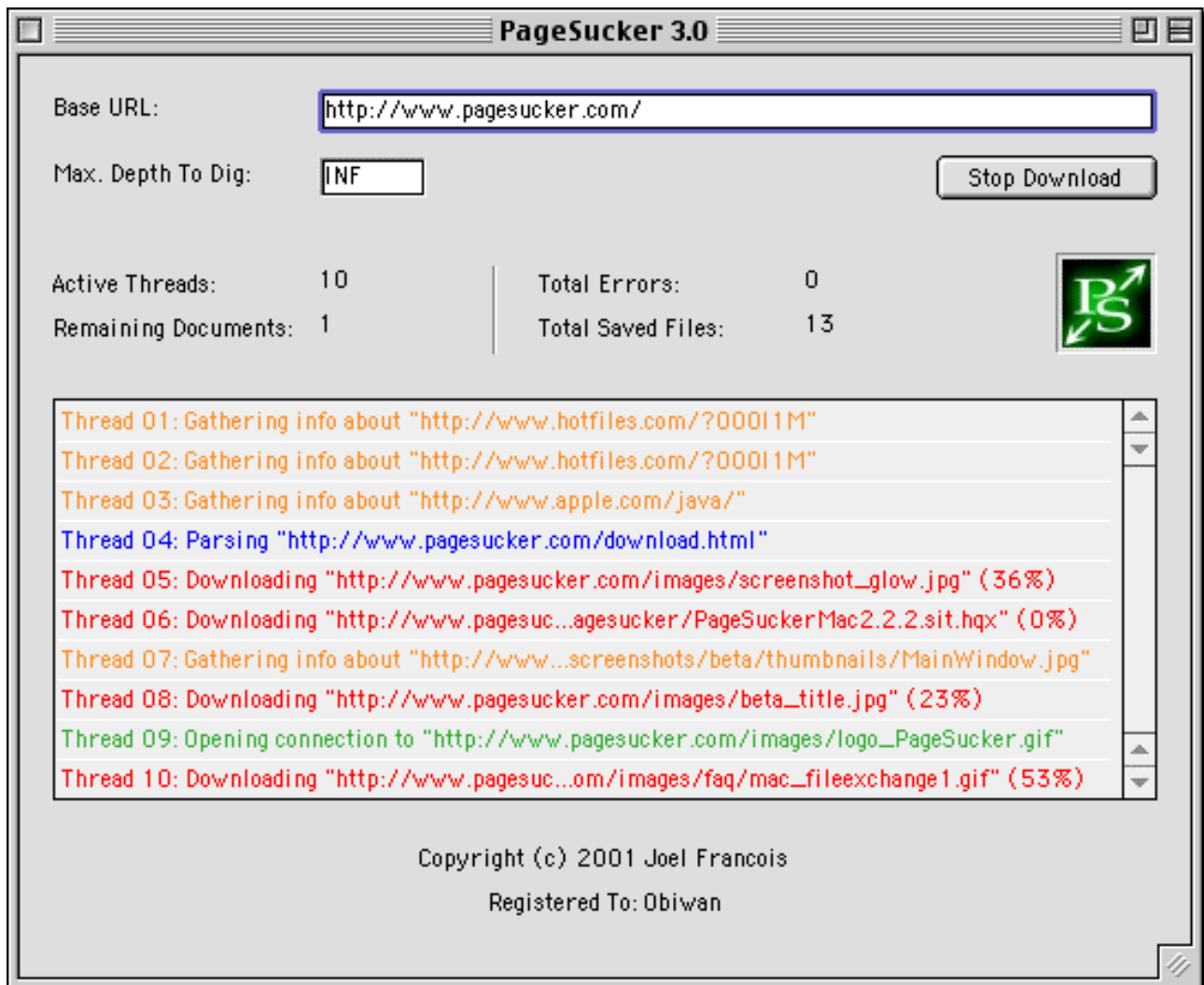
◆ **Total Errors:**

The total number of errors that have occurred since the download has been started. This field cannot be edited – it is intended as a status display only.

Common errors are the failure to find a linked page, the detection of an invalid URL on some parsed page or the lack of authorization to access a protected page. When an error occurs, it is announced in the log window (see section 8) and is inscribed in the log file. To see the history of all the errors which occurred

	<p>during a particular download, you can open the log file with a standard text editor. By default, this file is created inside the PageSucker window and has the name “PageSuckerLog.txt”. The name and location can be set via the <i>Miscellaneous</i> settings window however (see section 5.8).</p>
◆ <b>Total Saved Files:</b>	<p>The number of files actually saved to the local disk during the latest download process. Not all downloaded objects are saved; this can be controlled via various filters (see section 5 below for more details).</p>
◆ <b>Allocated Memory:</b>	<p>The amount of memory PageSucker allocated internally for its operation. Note that in reality more of the system's memory may actually be used by the application, as PageSucker is a Java application, and the Java engine also needs some memory for its own use.</p>
◆ <b>Start / Stop Download:</b>	<p>The <i>Start Download</i> button starts a download process. Once the download is running, that button changes to <i>Stop Download</i>. Normally, a download stops automatically when complete. To abort the process before it is complete, the <i>Stop Download</i> button can be clicked. PageSucker will then try to cancel all running threads in order to stop the download. The <i>Start</i> button will become available again only after the download currently in progress has terminated or has been successfully cancelled.</p> <p>Please note that if a thread is stuck (its connection has blocked due to some problem with the server being contacted) it usually cannot be interrupted. In that case, the application can't abort the download process and must be quitted and restarted in order to be used again.</p> <p>After clicking the <i>Start Download</i> button, PageSucker will ask for a download directory. This is the local directory the downloaded pages are saved in. Ideally, it should be an empty directory on a local disk. When downloading a page hierarchy, PageSucker tries to recreate the server's file system structure. This may lead to lots of subdirectories being created on the local disk. However, no files or directories will be created outside of the directory you choose here.</p>
◆ <b>The PageSucker Logo:</b>	<p>Directly below the Start button can be seen the PageSucker (PS) logo. It is an indicator of the application's overall status. When the application is idle (no download is running), the logo is simply white on black, as shown in the screenshot above. When a download is launched, the logo will start to glow green. A red glow will be visible when PageSucker is trying to stop a download in progress (after the <i>Stop</i> button has been clicked).</p>

Below these controls in the main window can be seen the list of threads (and what they're currently doing). You will find a summary of the various thread status displays below. Here's an example screenshot of a running download:



- \* Idle  
(shown in black)  
A thread marked "idle" isn't currently doing anything; it is waiting for some work to do. Threads are always "idle" when no download is running. During a download, idle threads may show up when there are no more URLs waiting to be handled in the queue at that time.
- \* Parsing URL  
(shown in blue)  
The thread is currently analyzing an HTML page to find embedded links.
- \* Downloading URL  
(shown in red)  
The thread is currently downloading a file. A percentage enclosed in parentheses shows the progress of the download.
- \* Opening/closing connection to URL  
(shown in green)  
This indicates that the thread is trying to open, resp. close a connection to a remote server. Both operations can take more or less long, depending on how busy the server and the network in general is.

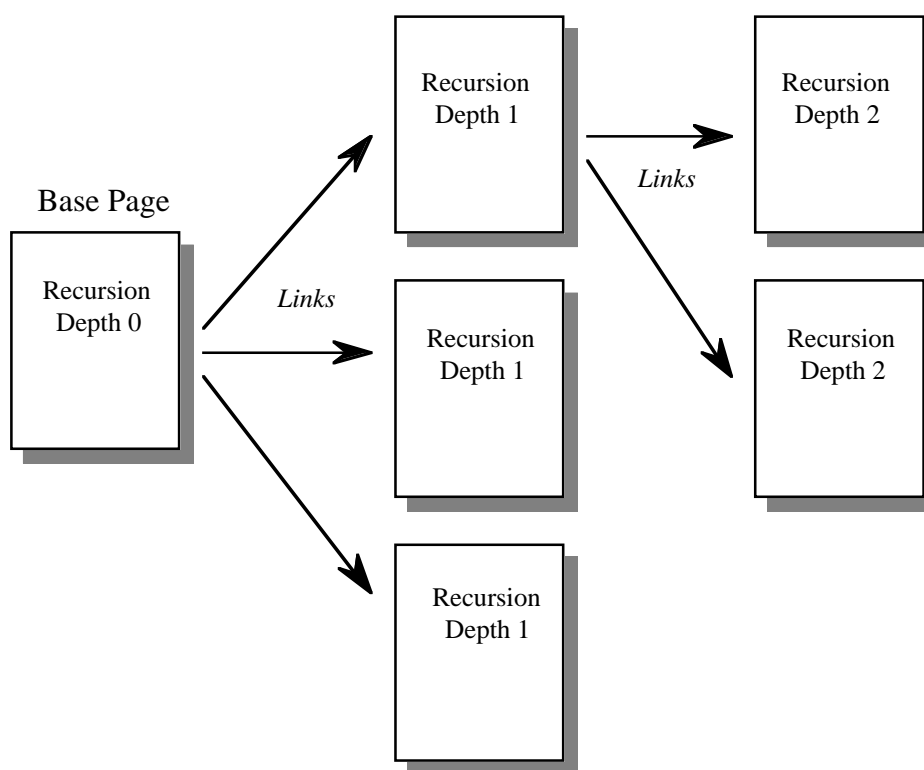
- \* Gathering info about *URL* (shown in orange) A particular URL is being looked at in detail, in order to decide if it should be downloaded or not. You will see this status only if certain filter settings are enabled (such as JavaScript or MIME type support).



## 4. WHEN TO STOP? - THE END OF A DOWNLOAD PROCESS

A download process stops **if no more files liable to be downloaded can be found**. Two global rules determine if a file should be downloaded:

1. A file is not downloaded if its **recursion depth** is greater than the **maximum recursion depth**. The recursion depth is a number attributed to the pages as they are downloaded. The initial page (specified by the user via the *Base URL* field in the main window) has a recursion depth of 0. All pages found through links on the initial page then have a recursion depth of 1, and so on. The recursion depth mechanism works somewhat like a generation counter: each time a link is resolved, the counter is incremented by one. If a maximum recursion depth is given in the main window (see “The Main Controls” above), PageSucker doesn’t download pages with a recursion depth which is greater than that value. Only pages with a depth less than or equal to the maximum depth will be analyzed. Example: If zero is entered as maximum recursion depth, only the initial page will be downloaded, whereas a value of one will download the initial page plus all pages linked from that page.



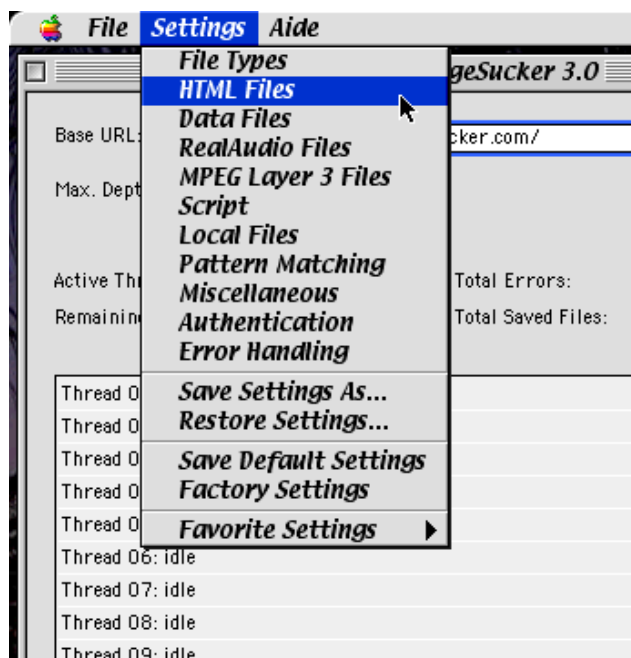
2. Only files the URL of which passes all of PageSucker’s URL filters will be downloaded and saved. More information on filters can be found in the following sections.

After the download has completed, if some HTML files have been saved to the local disk, PageSucker will have created a small index file called “index.html” inside the local download directory. This file can be opened with a Web browser to automatically load the downloaded base

page (the one the URL of which has been typed into the “Base URL” field in the main window). No index file is created if PageSucker has been set to only download data files.

## 5. THE SETTINGS MENU

PageSucker allows very detailed control on which files should be downloaded and which ones should be ignored. This is done via various filter settings available from several settings windows. To access a settings window, choose it from the “Settings” pull-down menu<sup>2</sup>:



Here's a quick overview:

The top items of the menu each open up a different settings window. The settings are grouped according to their type. Each settings window will be explained in detail in the following sections.

The *Save Settings As*, *Restore Settings* and *Save Default Settings* menu items allow you to save, resp. restore the current settings state to/from a file. More on this later.

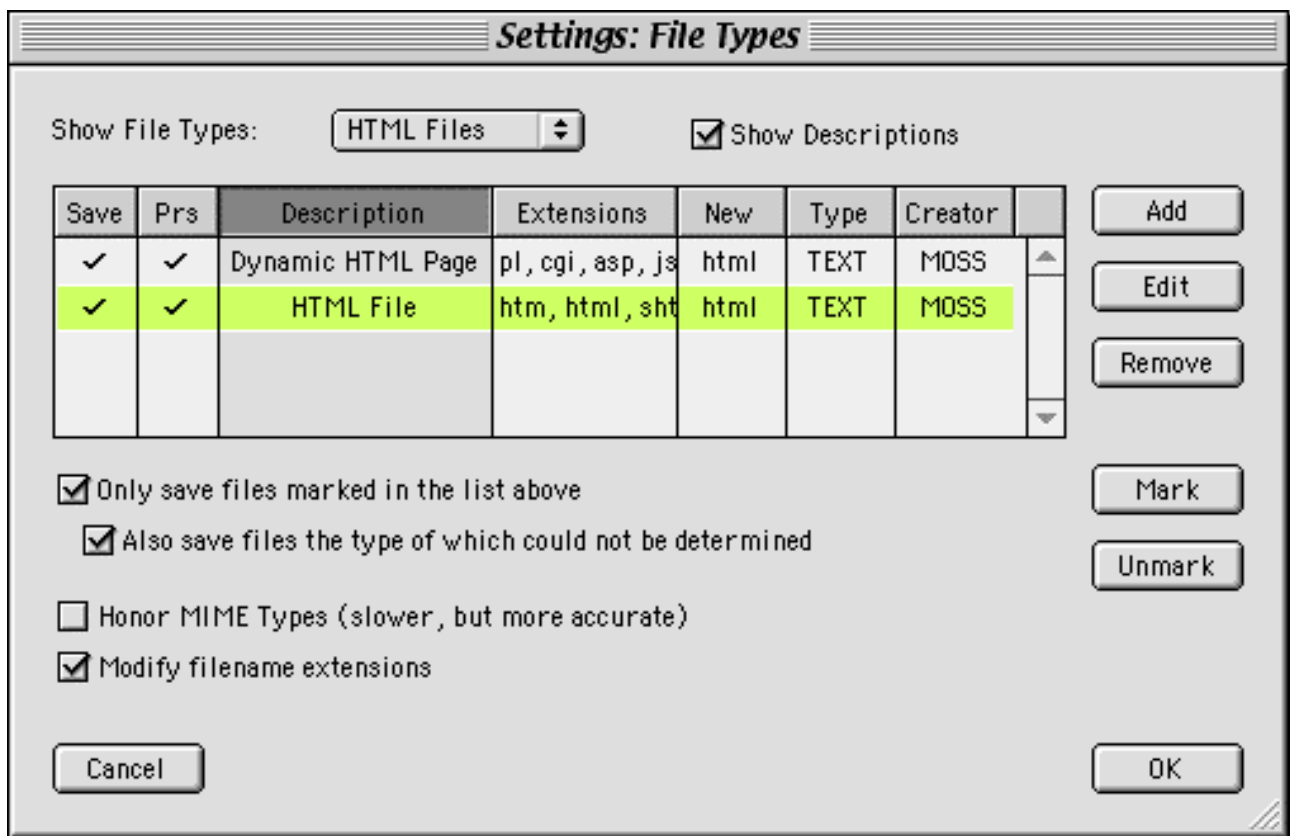
*Factory Settings* restores the default settings for the entire application. Finally, the *Favorite Settings* submenu can be used to locate those settings which are used regularly. See below on how to use this menu.

Please note that you can change settings even while a download is in progress. In that case, the new settings will be used for the following download only. They will not affect the download that is currently running. Let's take a look at each of the settings windows in detail.

### 5.1 The File Types Window

A first step in telling PageSucker which files to download is to select the types of files that we're interested in. The types of files known to the application are all listed in the *File Types* window, which might remind you of similar controls found in some commercial Web browsers.

<sup>2</sup> In the Windows version the pull-down menus appear only in the main window. In the Mac version, they are always visible at the top of the screen. The screenshot was taken on a Mac running MacOS 9.1.



### 5.1.1 Some General Usage Notes

One very important thing to note about this window is that it actually has two pages: one for data (non HTML) and one for HTML file types. At the top of the window is a pulldown menu, which allows to switch between the two pages. It is marked “Show File Types”. The screenshot above shows the “HTML Files” page.

Dominating this window is the file type table, which lists the known file types in the chosen category (either data file types or HTML file types). Here are a few hints on how to handle the table:

- The table can be sorted according to one of its columns by clicking the appropriate header tab.
- Finding a specific line is very easy: simply type the first few letters of the desired entry, and the matching row will be displayed and selected. How the typed letters are interpreted depends on the active column. For example, in the screenshot above, we could type the letter “D” to select the first row when the “Description” column is active. If the “Extensions” column were active though, we would have to type the letter “P” to select the same row.
- If a row is already visible on the screen, it can be selected by clicking in any of its fields.
- All rows can be selected together by pressing Command-A (on Macintosh) resp. Control-A (on Windows).

- Several contiguous rows can be selected by clicking a row while the SHIFT key is being held down. To make a non contiguous selection, hold down the Command key (on Macintosh) resp. the Control key (on Windows) while clicking a row.
- The “Save” and “Parse” columns are special, so-called checkmark columns. If the mouse is clicked on a row within a checkmark column, the row will not be selected, but instead a checkmark symbol (✓) will be toggled in the clicked cell.
- An alternate way to set a checkmark in the “Save” column for one or several rows is to select those rows and click the *Mark* button. Similarly, the *Unmark* button can be used to remove the checkmarks.

### 5.1.2 The Table Columns

---

The meaning of the various columns is explained below. The abbreviated column name is used when there is not enough space available on the screen to show the full name.

❖ <b>Save:</b> (abbreviated “Sav”)	If this column is checked, files of the associated type will be downloaded and saved to the local disk. Please note that files not marked to be saved may still be downloaded (for example to be parsed), but no local file will be created to store them.
❖ <b>Parse:</b> (abbreviated “Prs”)	This column is available only on the HTML file types page. It can be checked to specify that PageSucker should parse (analyze) a certain type of files in order to find new links to download. Usually, this column will remain checked for most HTML file types, but you might want to temporarily uncheck it for a specific type of files (for example dynamic HTML files) to suppress the parsing of those files.
❖ <b>Description:</b> (abbreviated “Desc”)	A simple plain text description or name for the file type. It is not used by PageSucker but can be useful for a human viewer to identify a type’s meaning. The description column can be hidden by unchecking the <i>Show Descriptions</i> checkbox near the top of the window.
❖ <b>Extensions:</b> (abbreviated “Ext”)	<p>The list of filename extensions associated with the file type. It is common practice to append a short extension (preceded by a dot) to filenames, in order to specify their type. For example, a JPEG image file typically would be called “SomeName.jpg”, whereas a GIF image file would have the extension “gif”.</p> <p>In some cases, several different extensions can be used to specify similar types of files. Typically, on the Internet, URLs having the extensions “htm”, “html”, “pl”, “cgi”, “asp”, etc. usually all designate HTML documents. (Technically speaking, there is a difference between those types, but it is usually only visible on the server side; so we don’t have to worry about that when downloading such documents.)</p>

❖ **Local Ext:**  
(abbreviated “Loc”)

PageSucker uses the list of filename extensions to associate one of the file types from the table to each URL it discovers.

A single filename extension to be used for the local file created to hold a downloaded file of this type. As explained above, various filename extensions can sometimes be used to designate the same type of files. In those cases, PageSucker can use the local extension to ensure that the local files of that type use one common extension.

That is particularly useful with dynamic HTML files, which commonly have extensions like “pl”, “php”, “asp” or “jsp”. Some commercial browsers don’t understand those filename extensions when they are used with local files. It is thus a good idea to rename those files so that they always have the extension “html”, which is known by the browsers.

Filename extensions are only changed if the *Modify filename extensions* check box is selected, though. The local extension column will not be visible if that option is unchecked.

❖ **MIME Type:**  
(abbreviated “MIME”)

This column lists the file type’s MIME type. It is visible only if the *Honor MIME Types* option is checked. A MIME type is an official, standardized textual name for each type of file. For example, the MIME type for JPEG files is “image/jpeg” while HTML files have the type “text/html”. As you can see, the MIME type defines that a JPEG file is an image and an HTML file is a text document. But it goes further and also specifies the kind of image resp. of text each file contains.

PageSucker can ask a Web server to get the MIME type for a given URL. This is a very accurate way of identifying the type of a file (better than using a filename extension, actually), but it has the disadvantage that the server has to be contacted in order to find out the type of a certain URL. The problem here is that opening a connection to a Web server is always a slow operation, and as such PageSucker tends to work noticeably slower if the *Honor MIME Types* options is enabled.

By default that option is off, and the recommendation is to turn it on only when downloading a Web site which uses URLs that either have no filename extensions or inconclusive ones. For example, if a Web site uses URLs like

`http://www.example.com/file.cgi`

to interchangeably designate HTML or data files, the type of each file cannot reliably be determined by using a filename extension approach only. In that case, PageSucker’s MIME type support should be used.

❖ **Mac Type:**  
(abbreviated “Type”)

This column, like “Mac Creator” below, is only visible when PageSucker is being used on a Macintosh computer. A Mac type is a four letter code, which uniquely identifies each type of files to the Macintosh operating system. Examples are “TEXT” for text files (including HTML files) and “GIF” for GIF image files.

If a Mac type is given for one of PageSucker’s file types, it will be applied to the local file when it is created.

❖ **Mac Creator:**  
(abbreviated “Creator”)

This column, like “Mac Type” above, is only visible when PageSucker is being used on a Macintosh computer. A Mac creator is a four letter code which uniquely defines the application that supposedly created a given Macintosh file. That way, the operating system knows which application to open when a file is double-clicked in the Finder.

If a Mac creator is given for one of PageSucker’s file types, it will be applied to the local file when it is created.

### 5.1.3 The Checkbox Options

---

◆ **Show Descriptions**

When this option is checked, the “Descriptions” column of the file type table is visible. Uncheck the option to hide that column.

◆ **Only save files marked in the list above**

This is the general switch which determines if the file type filter is applied or not. If this option is unchecked, PageSucker will download all types of files, regardless if they are marked to be saved in the table or not. If you would like to restrict your download to certain types of files though, you should check this option.

◆ **Also save files the type of which could not be determined**

This option is available only when the *Only save files marked in the list above* option is checked. It applies to those files for which PageSucker fails to determine a specific file type. That can happen if a file is encountered which is of a type that is not listed in the table. If this option is checked, such files will be saved locally, otherwise they will be ignored.

◆ **Honor MIME Types**

If this option is checked, PageSucker will try to get a MIME type from the server for each URL it encounters and will use it to identify the type of the file. If this option is not checked, the file type identification process will rely only on the filename extensions, which is less accurate, but much faster. You should thus enable this option only if needed. See the description of the “MIME Type” table column above for more info on MIME types and how PageSucker uses them.

◆ **Modify filename extensions**

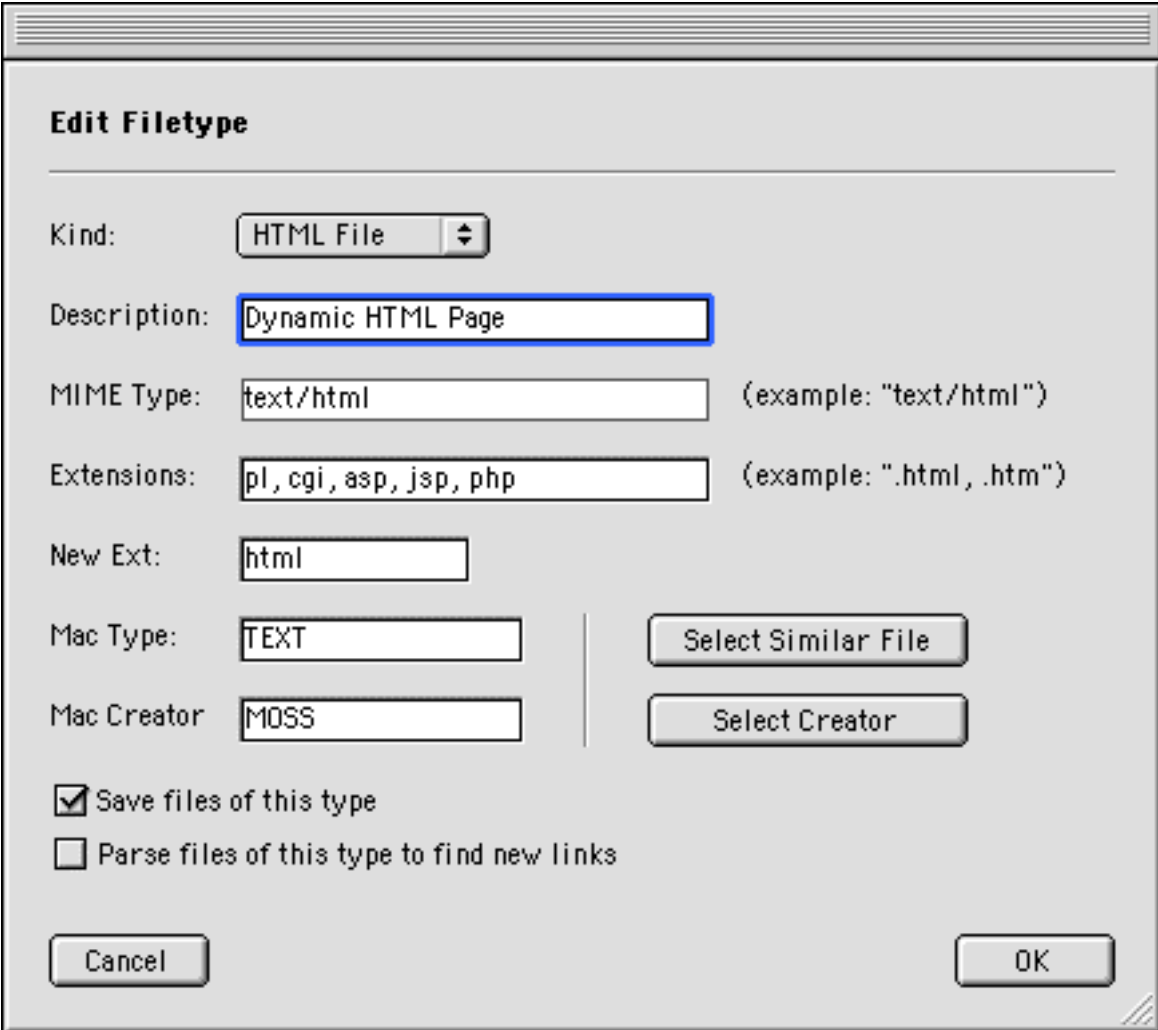
Check this option to have PageSucker replace the filename extension with a new one, defined for each file type. See the

documentation for the “Local Ext” table column above for more info on this option.

#### 5.1.4 Adding, Editing and Removing File Types

New file types can be added to the table by clicking the *Add* button. To edit an already created file type, find its row in the table and either double-click the row, or select it and then click the *Edit* button. Finally, the *Remove* button can be used to delete the selected rows from the table.

When creating or editing file types, the following dialog will be used:

The image shows a dialog box titled "Edit Filetype". It contains several input fields and checkboxes. The "Kind" field is a pulldown menu set to "HTML File". The "Description" field is a text box containing "Dynamic HTML Page". The "MIME Type" field is a text box containing "text/html" with an example "(example: 'text/html')". The "Extensions" field is a text box containing "pl, cgi, asp, jsp, php" with an example "(example: '.html, .htm')". The "New Ext" field is a text box containing "html". The "Mac Type" field is a text box containing "TEXT" next to a "Select Similar File" button. The "Mac Creator" field is a text box containing "MOSS" next to a "Select Creator" button. At the bottom, there are two checkboxes: "Save files of this type" (checked) and "Parse files of this type to find new links" (unchecked). "Cancel" and "OK" buttons are at the bottom left and right respectively.

All of the file type attributes present in the file type table can be edited in this window. The attributes will not be discussed in detail in this section, as this has already been done in section 5.1.2 above. Still, a few usage notes shall be added:

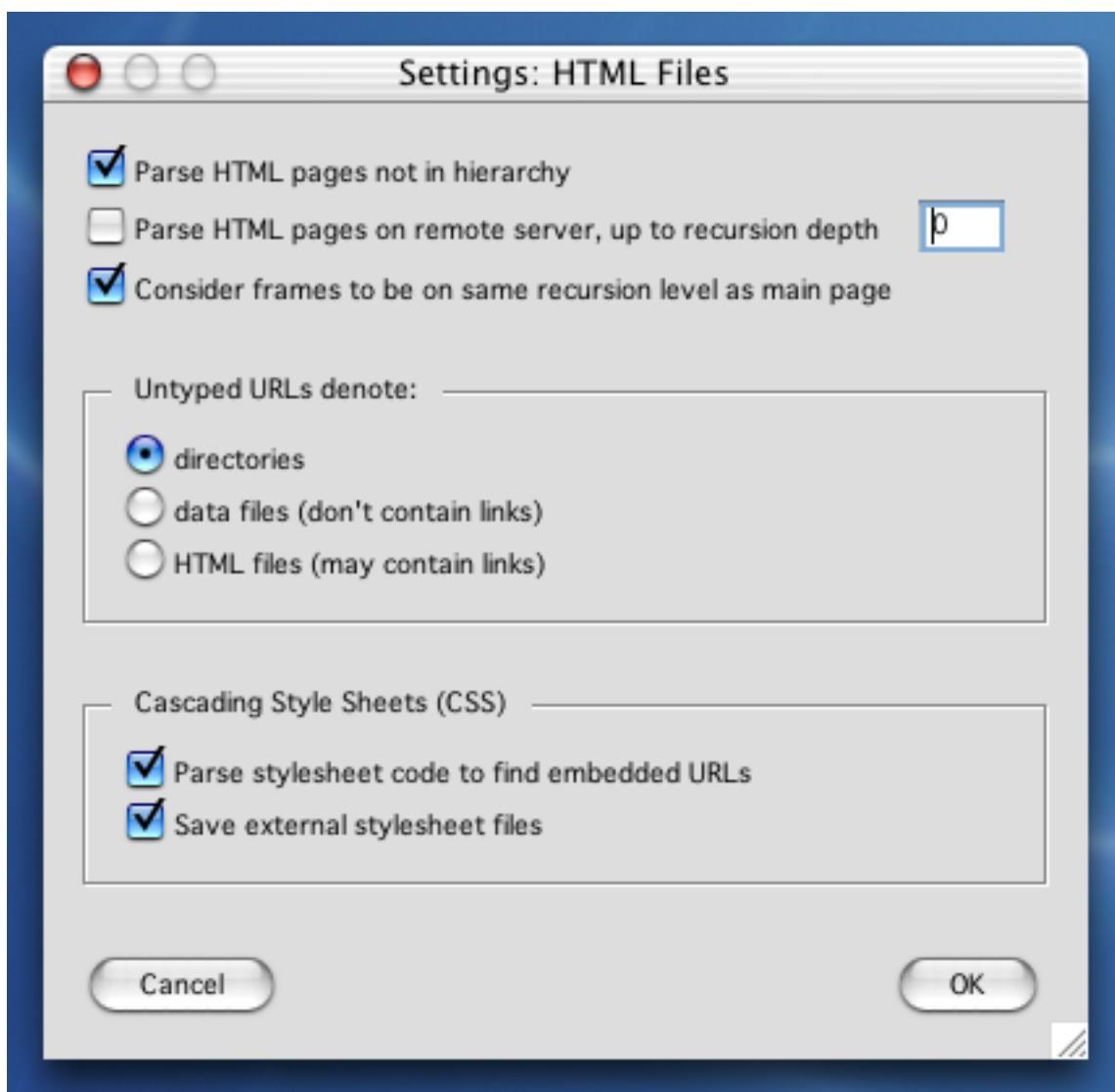
- Use the *Kind* pulldown menu to choose between a data file and an HTML file. This specifies in which page of the file types window the type will be listed.



- The *MIME Type* text field will be locked and will always show “text/html” if an HTML file type has been selected in the *Kind* menu. To edit the MIME type, first select “Data File” from the *Kind* menu.
- Several extensions, separated by commas, can be entered into the *Extensions* text field. It is not necessary to prefix each extension with a dot.
- The *Mac Type* and *Mac Creator* fields and the associated buttons are present only if PageSucker is run on a Macintosh. Use the *Select Similar File* button to choose a file the type and creator of which you wish to use. This will replace both the *Mac Type* and *Mac Creator* fields’ contents. Likewise, the *Select Creator* button can be used to select an application the signature of which will be used as creator.

## 5.2 The HTML Files Window

HTML file filters handle various issues related to HTML files, i.e. those files that define the layout of a Web page. When you choose *HTML Settings* from the Settings menu, the following window appears:



◆ **Parse HTML pages not in hierarchy**

If this option is unchecked, PageSucker ignores all HTML files not contained inside the hierarchy defined by the base URL (the URL entered in the main window – see section 3). This includes all files the URL of which does not start with the base URL stripped of its file name. An example should elucidate this concept. If we set the base URL to be

```
http://www.example.com/text/french/paris.html
```

PageSucker will ignore all files the URL of which does not start with

```
http://www.example.com/text/french/
```

If the *Parse HTML pages not in hierarchy* option is enabled however, PageSucker isn't concerned about the hierarchy: all HTML pages on the base server will be considered for downloading.

◆ **Parse HTML pages on remote server**

Normally, PageSucker ignores all HTML pages not on the server specified in the base URL. This is generally what you want, as most of the time the page hierarchy you're trying to download will be contained on one server. Sometimes however, this is not the case. That is why the *Parse HTML pages on remote server* option has been included: when it is checked, PageSucker also considers pages not on the base URL's server. An example: let's assume the base URL is

```
http://www.example.com/text/french/paris.html
```

If the *Parse HTML Pages On Remote Server* option is unchecked, PageSucker will ignore all files the server (host) of which is not

```
www.example.com
```

To include files on other servers as well, check this option. But be careful! While this may seem very useful, it could also lead to the problem that too many pages are downloaded: most pages on the Internet are linked to other, remote pages. PageSucker might jump to these remote servers and from there to yet other linked servers and so on. This would result in downloading lots of pages you probably don't want to download. Even worse, it might lead to a download process that just never seems to stop (until PageSucker runs out of system memory).

To counter that problem, you have the possibility to limit the download from remote servers to a fixed recursion depth<sup>3</sup>. It's the

---

<sup>3</sup> For an explanation of the term "recursion depth", see section 4.

◆ **Consider frames to be on same recursion level as main page**

number in the text field behind the *Parse HTML pages on remote server* option. The default value is zero, which means that remote pages are downloaded, but all links inside those pages are ignored. This insures that the download process doesn't get out of control. You may set this value to "INF" (for an infinite recursion depth), but be very careful when doing so – you wouldn't want to download the entire Internet to your hard drive, would you?

Wow! That's one complicated name for an option. To understand what it means, you must know that HTML frames (independently scrolling panes in the browser window) are actually implemented by referencing individual files via URLs from one main HTML page (the "frameset"), i.e. they are **not contained** inside the main page, but **linked** to it, just as those clickable links that appear as underlined words in the page. Now, as explained in section 4 above, whenever a link is resolved, the recursion depth (or recursion level) counter is incremented. Moreover, a linked page is ignored whenever its recursion depth is higher than the allowed maximum recursion depth. This then means that certain pages would have their *frameset* downloaded, but not the individual *frames* that make up the page, as these frames would have too high a recursion depth. Well, to make sure this **does not happen**, just check the *Consider frames to be on same recursion level as main page* option. In fact you'll rarely need to uncheck it.

Don't worry if you don't understand any of the above technobabble; just leave this option checked (it is checked by default).

◆ **Untyped URLs denote**

When reading the discussion about file types and filename extensions in section 5.1 above, you might wonder what PageSucker does with filenames that have no extension at all. For example, what does the following URL refer to?

```
http://www.example.com/text
```

Well, normally a URL of that type refers to some file which is contained inside a directory that has the URL's filename. The name of the file itself is defined on the server side and is not visible to PageSucker. Usually it is "index.html" or "home.html". In the example case, the URL would thus refer to the file "index.html" in a directory called "text" on the server "www.example.com".

Sometimes however, this is not true. Some servers use filenames with no extension at all. Thus, the above example could also refer to the file "text" on the server "www.example.com". As PageSucker is not able to find out if a name without an extension denotes a file or a directory, you must decide this up front. You can do this with one of the three radio buttons:

- Choose "**directories**" if you don't expect to find files without

filename extensions. This is the default setting and the one you will need most of the time.

- Choose “**data files**” if you know that the server contains files without filename extensions but these files contain only generic data (like pictures etc.), not HTML code that needs to be analyzed in a special way.
- Choose “**HTML files**” if you know for sure that the server you are downloading uses files without filename extensions that contain HTML code.

Please note that your choice between “data files” or “HTML files” may be overruled by other settings, such as the *Honor MIME Types* option in the File Type settings window (see section 5.1.2). If PageSucker knows for sure that a file contains HTML code, it will treat it as an HTML file, even if the filename doesn’t have an extension and the “data files” option is selected. If however the nature of a file is not certain, PageSucker will rely on the chosen settings in the *HTML Files* window.

◆ **Parse stylesheet code to find embedded URLs**

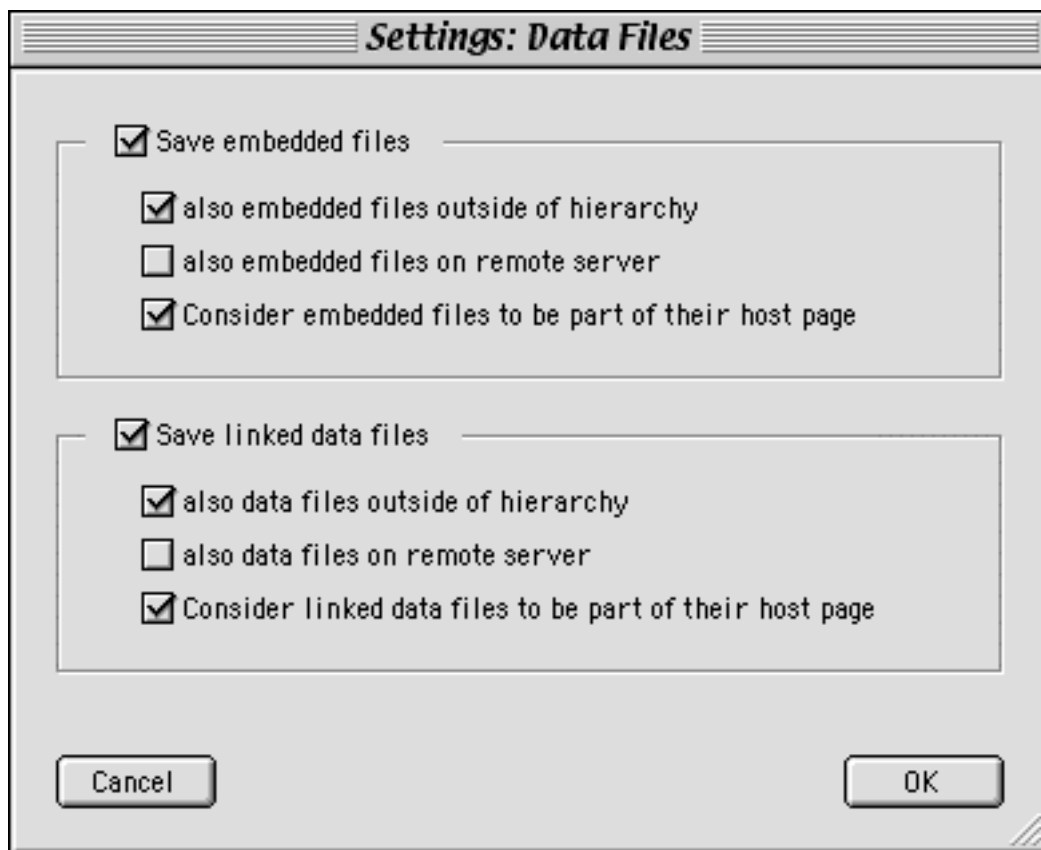
When this option is enabled, cascading stylesheet code is analyzed in order to find possibly embedded URLs which would otherwise not be found. Cascading stylesheets (short: CSS) are a way used influence the looks of Web pages, and are nowadays widely used. Often CSS are used to set text colors and fonts only, but they can also be used for example to cause a picture to be displayed as the background of a Web page. If PageSucker should detect and download such pictures, the “Parse stylesheet code” option needs to be checked. This option is checked by default (in registered copies of PageSucker) and normally you should leave it that way.

◆ **Save external stylesheet files**

Cascading stylesheet code does not always necessarily occur embedded inside an HTML document. Often, stylesheets reside as separate files on a server, which can then be included in Web pages with specific instructions. In order to enable PageSucker to also download such external stylesheet files, the “Save external stylesheet files” option must be checked. This option is checked by default (in registered copies of PageSucker) and normally you should leave it that way.

## 5.3 The Data Files Window

Choose “Data Files” from the Settings menu to open the following window:



While HTML file filters (see section 5.2) are concerned with all files that contain HTML code, data file filters handle all other kinds of files (i.e. files not containing HTML code). Here’s a detailed explanation of this window’s settings:

◆ **Save embedded files**

Files embedded in Web pages will be ignored unless this option is enabled. Technically speaking, this includes all images referenced via the `<IMG>` HTML tag, as well as files embedded using the `<EMBED>` tag<sup>4</sup>. Images and other files accessed by clicking on some link (referenced through the `<A>` HTML tag) are not considered embedded and are thus not affected by this option.

The following three options are only available if the *Save embedded files* option is checked.

◆ **(Save) also embedded files outside of hierarchy**

Embedded files that are located on the host server specified by the base URL, but not inside the hierarchy which the base URL specifies, will not be saved unless this option is checked. The concept of file hierarchy has been explained in section 5.2 above. You should generally leave this option checked (it is checked by

<sup>4</sup> There are other tags (for example `<OBJECT>` and `<APPLET>`) which are also used to embed files in Web pages. Those are not yet supported by PageSucker 3.0. Support for those tags will be added in a future release of the software.

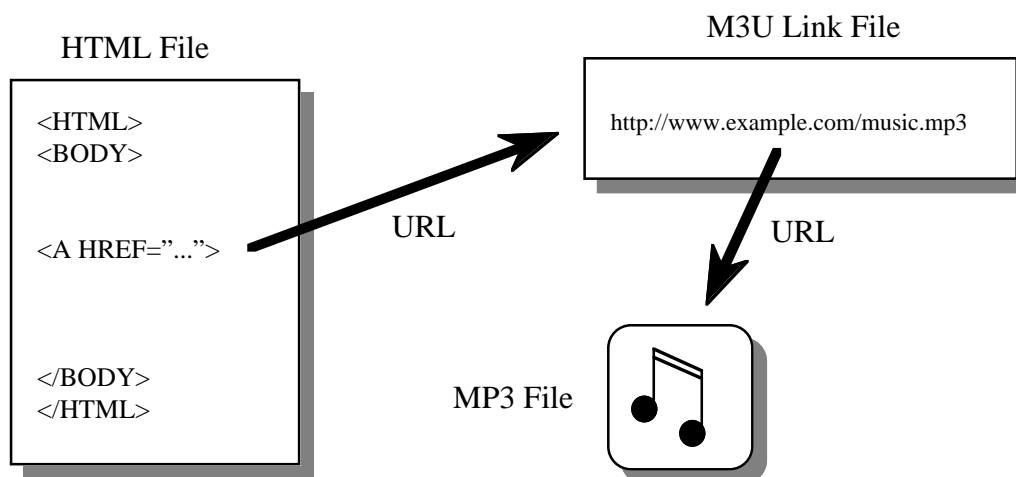
	<p>default), as it is common practice to store pictures in a hierarchy separate from the HTML files.</p>
<p>◆ <b>(Save) also embedded files on remote server</b></p>	<p>Embedded files that are located on a different host server than the one specified by the base URL will be ignored unless this option is checked.</p>
<p>◆ <b>Consider embedded files to be part of their host page</b></p>	<p>Check this option if embedded files should be treated as though they were a part of the page which references them. Technically speaking, this is related to the maximum recursion depth (which you can set in the main window).</p> <p>As explained in section 4 above, the recursion depth is incremented each time a URL found inside a page is analyzed. Given that an embedded file is a separate file and is also referenced via its own URL, it would be logical to consider it to have the recursion depth of the page it is embedded in, incremented by one. If this option is checked however, the recursion depth is <u>not</u> incremented for embedded files.</p> <p>For example, if the maximum recursion depth is set to zero, only the base page will be downloaded. All links contained in that page will be ignored, as they have a recursion depth of one, which is greater than zero. If the <i>Consider embedded files to be part of their host page</i> option is <u>unchecked</u>, none of the files embedded in the base page will be downloaded. With the option <u>checked</u>, those files will be saved, although all other links are ignored.</p> <p>Normally you should leave this option checked.</p>
<p>◆ <b>Save linked data files</b></p>	<p>This filter concerns all files that are not HTML files, and which are linked to a page (via an &lt;A&gt; HTML tag). When a Web page is viewed, linked files usually show up as underlined words which can be clicked to access those files. If this option is <u>unchecked</u>, all of those linked files will be ignored. Embedded files are not affected by this filter.</p> <p>The following three options are available only if this option is on.</p>
<p>◆ <b>(Save) also data files outside of hierarchy</b></p>	<p>Linked data files that are located on the host server specified by the base URL, but not inside the hierarchy that the base URL specifies, will not be saved unless this option is checked.</p> <p>HTML files or embedded files are not affected by this filter.</p>
<p>◆ <b>(Save) also data files on remote server</b></p>	<p>Linked data files that are located on a different host server than the one specified by the base URL will be ignored, unless this option is checked.</p> <p>HTML files or embedded files are not affected by this filter.</p>

◆ **Consider linked data files to be part of their host page**

Check this option if linked data files should be treated as though they were a part of the page which references them. Technically speaking, this means that the recursion depth counter will not be incremented when a linked data file is treated. See the discussion on the *Consider embedded files to be part of their host page* option in this section for more details on this concept.

## 5.4 The RealAudio Files And MPEG Layer 3 Files Windows

The settings in both of these windows are very similar. They handle the case of special data files used when referencing RealAudio (“.ra” or “.rm”) files and MPEG Layer 3 (“.mp3”) files. These special files contain nothing but one or more URLs referencing the real data files. In the case of a RealAudio file, the special file’s name ends in “.ram”. In the case of an MP3 file, the special file’s name ending is “.m3u”.



In both cases the reason for using an intermediate file is that this way a specialized application (not the Web browser) can be used to download the data files and play them at the same time (the playback is said to be “streaming”). PageSucker will of course need to handle this kind of link files in a special way, otherwise you would end up with only the intermediate file instead of the file containing the desired data when trying to download RealAudio or MP3 files.

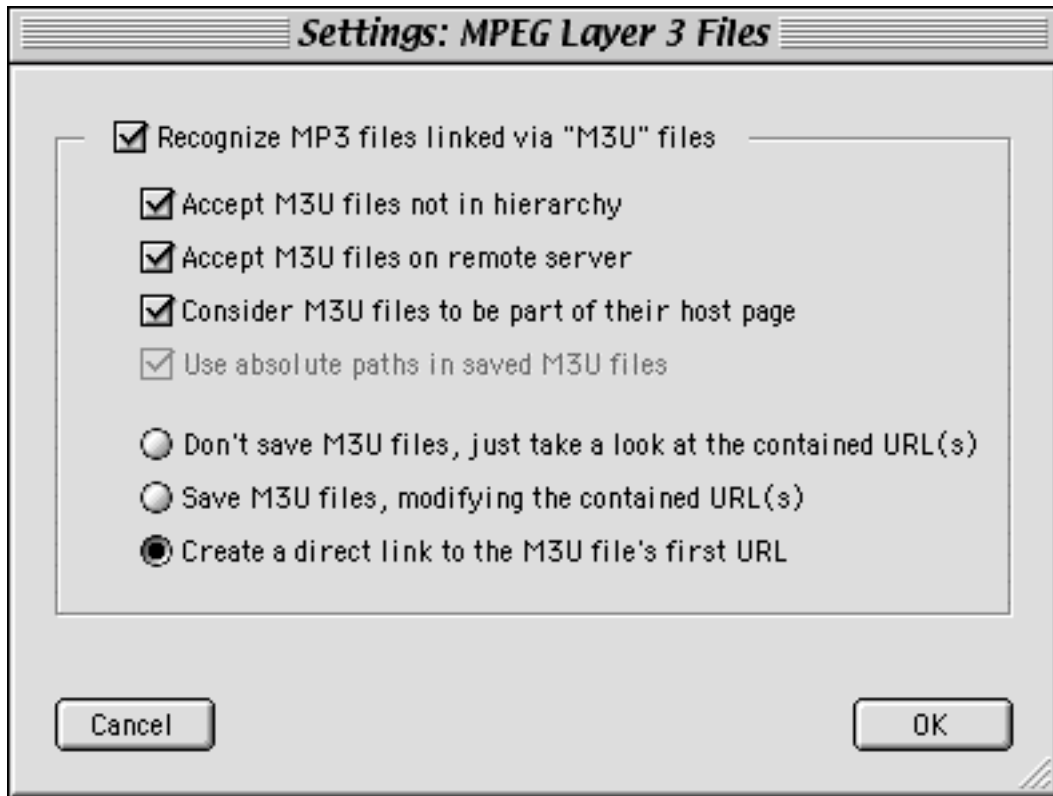


In the case of RealAudio files, the special player application will most probably be the RealAudio Player. A word of warning must be added up front in this context: Nowadays, most RealAudio streams that can be found on the Internet use special proprietary protocols (e.g. “pnm://”) instead of the standard “http://” protocol that was used some time ago. The “.ram” file will in that case contain a URL starting with “pnm://” or something similar. PageSucker is not able to understand that kind of protocol, so you might find that you cannot download a targeted RealAudio file although you told PageSucker to download it. As PageSucker silently skips URLs with unknown protocols, there will be no error message in that case.

One more thing to note is that the appropriate settings in the *File Types* and *Data Files* windows (see sections 5.1 and 5.3 above) are still used if indirectly linked RealAudio and MP3 files are to be downloaded. For example, if you specified that the only types of data files to download are JPEG

and GIF files, PageSucker will not download MP3 files referenced from an M3U file, even though you might have enabled the recognition of M3U files. By default, MP3, RA and RM files are already in the list of data files to download, so you needn't worry if you haven't changed those settings yourself.

Below is a screenshot of the MPEG Layer 3 Files window and a detailed description of each control. The RealAudio Files window is not discussed in detail as the MP3 window discussion applies to it as well.



◆ **Recognize MP3 files linked via “M3U” files**

This is the master switch for the entire window. All other options are disabled when this one is unchecked. In that case, M3U files will not be treated in any special way: they are normal data files that can be downloaded as any other data file (see the *File Types* and *Data Files* window descriptions in sections 5.1 and 5.3 for details on how to filter data files).

If this option is enabled, M3U files will be recognized as special files by PageSucker, the URL(s) contained therein will be extracted and handled.

◆ **Accept M3U files not in hierarchy**

PageSucker will ignore M3U files if they're not in the hierarchy specified by the base URL, unless this option is checked. You should generally leave this option checked (as it is by default).

◆ **Accept M3U files on remote server**

Check this option to also accept M3U files if they reside on a server that differs from the one set by the base URL. If this option is unchecked, such remote M3U files will be ignored.



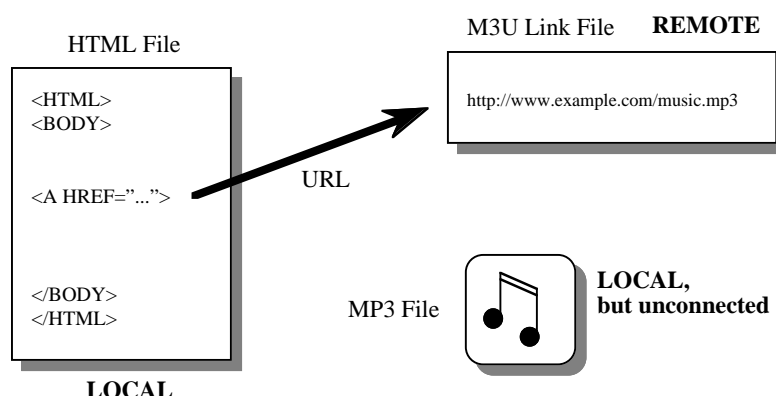
◆ **Consider M3U files to be part of their host page**

Check this option if M3U files should be treated as though they were a part of the page which references them. Technically speaking, this means that the recursion depth counter will not be incremented when an M3U file is encountered. See the discussion on the *Consider embedded files to be part of their host page* option in section 5.3 for more details on this concept.

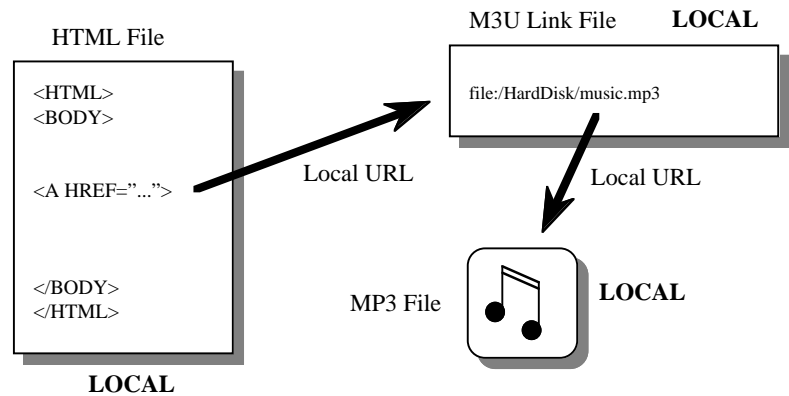
◆ **Use absolute paths in saved M3U files**

To explain this option, we must first take a look at the three possible ways to handle an M3U file. Which treatment is used is specified with the radio buttons at the bottom of the window:

1. ***Don't save M3U files, just take a look at the contained URL(s).*** If this treatment is chosen, the M3U file is opened, the URLs it contains are extracted and considered for downloading. The M3U file itself is not saved however. If the HTML page referencing the M3U file is saved, the link between the locally saved HTML page and the possibly saved MP3 files (referenced via the M3U file) will then be broken. The items marked “LOCAL” in the figure below have been downloaded to the local disk whereas those marked “REMOTE” have not been saved locally.

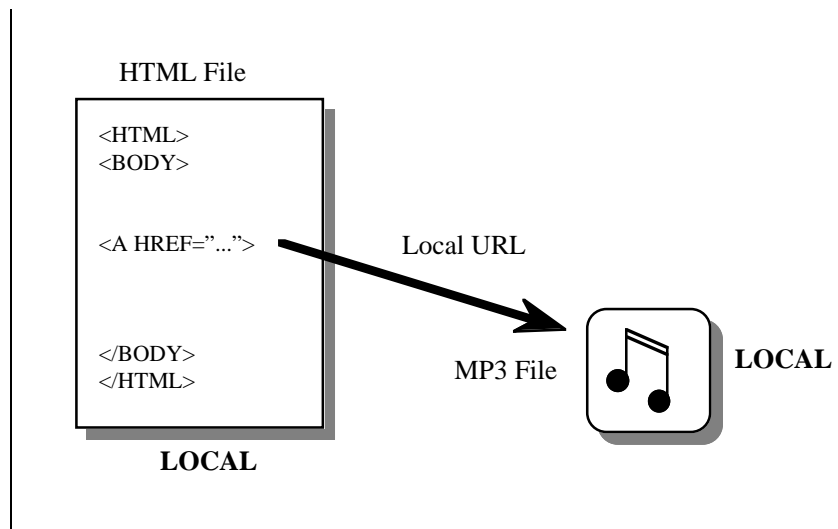


2. ***Save M3U files, modifying the contained URL(s).*** This saves the M3U file locally after extracting the contained URLs. The local M3U file is then updated to reflect the new local paths of the downloaded MP3 files.



This is where the *Use absolute paths in saved M3U files* option is used. If it is checked, the M3U file is updated with absolute local paths, otherwise relative local paths are used. An absolute local path is a URL that starts with “file:” and contains the complete path of the local MP3 file, starting from the hard drive’s root directory, while a relative path specifies the location of the MP3 file as compared to the location of the saved M3U file. Usually, relative paths are to be preferred as they remain valid even when the downloaded hierarchy is copied to another disk (this is not true for absolute paths). However, relative URLs sometimes seem to create problems when used in M3U files (probably due to a bug in certain player applications). That’s why you might want to use absolute paths. But remember that absolute paths become invalid when the downloaded hierarchy is moved around on the local disk.

3. **Create a direct link to the M3U file’s first URL.** This option reads the first URL contained in the M3U file, considers it for download, then uses the downloaded MP3 file’s URL in place of the URL referencing the M3U file when updating the potentially downloaded HTML file. The M3U file is thus shortcircuited. This is the default behavior for treating M3U files, but keep in mind that M3U files might contain more than one URL. This option considers the first URL only and skips the others.



## 5.5 The Script Window

This window, accessible via the “Script” item in the “Settings” menu, contains some options that enable PageSucker to detect certain types of links in HTML documents that would otherwise go unnoticed. This includes URLs contained inside JavaScript blocks and single JavaScript statements embedded in the HTML code, as well as URLs found in the definition of pulldown menus and lists.



Please note however that the detection of any of these types of URLs is mainly guesswork, i.e. it cannot be guaranteed that all included URLs will be found, nor that the downloaded pages will work as expected. Sometimes, PageSucker will even detect strings that are not URLs at all. This may then result in incorrectly downloaded pages. The reason for all this uncertainty is that JavaScript is a very complex procedural programming language. You would need some kind of artificial intelligence to find out what a JavaScript program really does (and even then there would be no guarantee that the artificial intelligence would understand the program’s functionality correctly). Now, without *understanding* how a given JavaScript routine works, it is of course not possible to predict all possible URLs it might produce.

Fortunately the problem is not always as tough in practice. Often, JavaScript routines contain complete URLs that are easy to spot. This is exactly what PageSucker tries to do: it extracts all strings that look like URLs, then it tries to open a connection to each of those would-be URLs. If the connection fails, PageSucker concludes that the considered piece of text was not really a URL, so it forgets about it. If the connection is successful however, the URL is added to PageSucker’s internal list of pages to be analyzed. Let’s consider two examples of JavaScript routines:

### **Routine #1:**

```
function generateURL (variant)
{
  var host = "http://www.example.com";
  var file = "picture";
  var url;

  if (variant == 1)
    url = host + file + ".gif";
  else
    url = host + file + ".jpg";

  return url;
}
```

### **Routine #2:**

```
function generateURL (variant)
{
  if (variant == 1)
    return "http://www.example.com/picture.gif";
  else
    return "http://www.example.com/picture.jpg";
}
```

}	
---	--

Both routines produce exactly the same URLs. Two different URLs are generated according to the “variant” parameter:

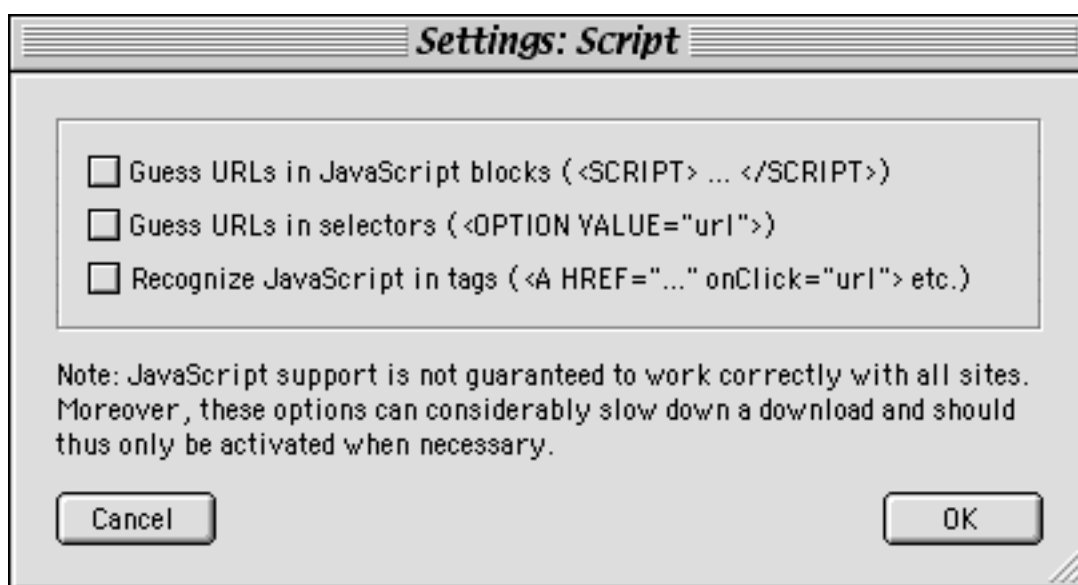
http://www.example.com/picture.gif  
and http://www.example.com/picture.jpg

While PageSucker will have no problem to detect both URLs in the second script, there are actually three problems when trying to extract the URLs from the first routine:

1. None of the two possible URLs is included literally in the first script, so PageSucker will find neither one correctly.
2. The string “http://www.example.com” looks like a complete URL to PageSucker, so it will consider it (incorrectly) for downloading.
3. The string “picture” is detected. This might be a relative URL, so PageSucker tries appending it to the URL of the HTML page that contains the JavaScript. Now, if the directory containing the HTML file also contains a directory or a file with the name “picture”, PageSucker will manage to open a connection to that file or directory and will thus consider it (incorrectly again) for downloading. This will not only cause unneeded downloads, but may also render the downloaded JavaScript program itself non functional.

So, to sum up we could say that PageSucker’s JavaScript support can be very handy in certain cases, but that it should be used with caution. However, you may of course always try if it works for your particular download. Moreover, due to the need to open additional connections in order to check the validity of potential URLs, the *Script* window options will considerably slow down PageSucker’s operation.

Below is a screenshot of the *Script* window and some words about each option it contains:



◆ **Guess URLs in JavaScript blocks**

This option enables PageSucker to look for URLs in blocks of JavaScript code. Typically such blocks of code can be found at the top of HTML pages. They are visible in the HTML source code as

being limited by the `<SCRIPT LANGUAGE="JavaScript">` and `</SCRIPT>` tags.

JavaScript include files (linked via the `SRC="url"` parameter in the `<SCRIPT>` tag) are handled by reading the include file's contents and embedding them into the HTML document that references them.

◆ **Guess URLs in selectors**

Check this to have PageSucker look for potential URLs in the definition of pulldown menus and scrollable lists. These user interface elements are defined in HTML forms with the `<OPTION VALUE="...">` tag. In some pages they are used with literal URLs embedded, hence the reason for adding this option to PageSucker.

While form elements are not really related to JavaScript, this option still has been included in the script support window, as in this case too PageSucker has to proceed by trial and error rather than knowing for sure that the value string is a URL.

◆ **Recognize JavaScript in tags**

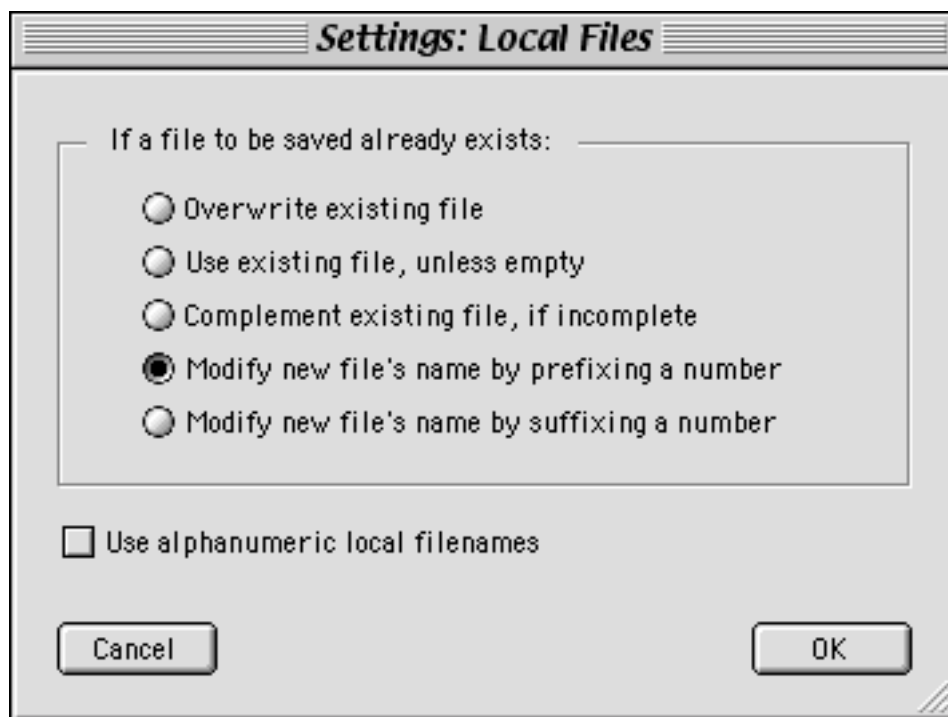
If you'd like PageSucker to recognize single lines of JavaScript embedded in various HTML tags, you'll need to check this option. For example the HTML code

```
<A HREF="url1" onMouseOver="showHelp('url2')">
```

defines a link to the URL "url1", but also calls the JavaScript routine *showHelp* while passing in a second URL (url2) as a parameter. This second URL will not be detected by PageSucker unless the *Recognize JavaScript in tags* option is checked.

## 5.6 The Local Files Window

Select "Local Files" from the "Settings" menu to show the following window:



◆ **If a file to be saved already exists**

These options let you specify what should happen if a file to be saved already exists on the local disk. There are five choices:

- ***Overwrite existing file*** – the remote file is downloaded, thereby overwriting the existing local file. The local copy (if any) of the HTML page that references the file being downloaded is updated so that it points to the overwritten file on the local disk.
- ***Use existing file, unless empty*** – the remote file is not downloaded. The local copy of the HTML page that references the file being downloaded is updated so that it points to the existing file on the local disk. If the existing file is empty, PageSucker falls back to the ***Overwrite*** option explained above.
- ***Complement existing file, if incomplete*** – PageSucker compares the lengths of the existing local file and the remote file. If the local file is smaller than the remote file, only the missing part is requested from the server in order to complement the file. This is a useful option when restarting an interrupted download, which may have left half downloaded files on the local disk. Please note though that this option will not work as expected if the server's files have been updated between two subsequent downloads. Also, if the first download was done using one of the “modify new file's name” options discussed below, it can be problematic to complement incomplete files, as the “complement” mode does not do any filename modifications.

- ***Modify new file's name by prefixing a number*** – the remote file is downloaded under a different name, which is constructed by adding a number and the '~' character to the beginning of the original file name (e.g. "picture.gif" becomes "0~picture.gif").
- ***Modify new file's name by suffixing a number*** – the remote file is downloaded under a different name, which is constructed by adding the '~' character and a number to the end of the original file name (e.g. "picture.gif" becomes "picture~0.gif").

The recommended (and default) choice is *Modify new file's name by prefixing a number*. Choosing to modify the new file's name is especially important if the site you are downloading may contain longer filenames than are permitted on the platform PageSucker is running on, as in that case the filenames will need to be shortened in order for the files to be saved locally. Although PageSucker takes care of the name shortening automatically, that operation may lead to name clashes, which can be resolved satisfactorily only if one of the "modify new file's name" options is selected.

A simple example should illustrate that problem: Let's suppose the local platform is limited to filenames no longer than 8 characters and let's further suppose that the following two URLs should be downloaded:

```
http://www.example.com/LongNameOfATextFile.zip
http://www.example.com/LongNameOfAPictureFile.zip
```

PageSucker will shorten both names to the local filename "LongName.zip", which is only 8 characters long (not including the filename extension). When the second file is to be created, the first one (having the same local name!) will already exist, and PageSucker will thus have to decide what to do: overwrite, reuse, complement or rename. Obviously, renaming the new file is the only good alternative in that case.

A similar case can happen with some servers that use a dynamic process to serve Web content. For example, consider the following two URLs, which actually refer to two different files via a generic server side engine:

```
http://www.example.com/getPicture.cgi?id=1
http://www.example.com/getPicture.cgi?id=2
```

These two URLs will lead to the creation of two local files, which will both have the name "getPicture.cgi" – again the ensuing name clash can only be solved if the second file is renamed by prefixing or suffixing a number.

◆ **Use alphanumeric local filenames**

This option is useful when files are downloaded the name of which contains special characters. Characters considered special in that regard are for example accented characters which appear in certain non-English languages, such as **é, à, ä, ø, ç, ñ** etc. (As a technical sidenote, we're talking about all those characters that are not defined by the ASCII standard.)

When such a filename is encountered, the special characters are automatically replaced by the letter 'x' if the *Use Alphanumeric Local Filenames* option is on. This prevents problems in the case the downloaded file is later copied to another platform (for example from a Macintosh to a Windows PC).

By default, this option is not checked. You might choose to enable it if cross platform compatibility is an issue.



## 5.7 The Pattern Matching Window

Select “Pattern Matching” from the “Settings” window to reveal the following window.

**Settings: Pattern Matching**

Pattern Type: DOS Pattern

☐ Ignore URLs if they match the following pattern:

☒ Parse URLs only if they don't match the following pattern:

☐ Save URLs only if they match the following pattern:

Cancel OK

The options in this window can be used to restrict the accepted URLs to only those which match certain patterns. Near the top of the window is the pattern type switch: use this pull down menu to select if your patterns are standard DOS like patterns or full fledged regular expressions. More details on those two types of patterns can be found below. Let's take a look for now at the other controls in the pattern matching window.

Three separate patterns can be entered, each of which will be used in a different way by PageSucker. All three patterns must be of the same type though. For each pattern you can indicate if you want a URL to pass the filter only if it matches or if it doesn't match the concerned pattern.

- **Ignore URLs if they match (resp. don't match) the following pattern.** This filter can be used to completely eliminate certain URLs, before other filters are applied to them. For example, you could instruct PageSucker to ignore all files pointing to a certain server or residing in a certain directory on the base server.
- **Parse URLs only if they match (resp. don't match) the following pattern.** This filter is used by PageSucker when it is trying to determine if a certain HTML file should be analyzed ("parsed") in an attempt to find more links to download. Please note that this does not affect the decision to save a file locally or not: HTML files which are not being parsed can still be saved.
- **Save URLs only if they match (resp. don't match) the following pattern.** Using this filter, you can control which files will be saved locally. In the case of HTML files it is of course still possible to have them parsed, even if this filter prevents them from being saved.

When composing patterns for PageSucker, just keep in mind that entire URLs will be matched against your patterns, not only file names or paths. Thus you need to include matching strings for the protocol (e.g. "http://") and server name. Also, remember that all other enabled filters will always be applied in addition to the pattern matching filters.



Some URLs found on the Internet may contain illegal characters, such as spaces for example. In order to correctly work with those URLs (which are - strictly speaking - invalid), PageSucker has to encode the illegal characters. For example, a space would be replaced with the string "%20". This can lead to problems when applying the pattern matching filters though, as the patterns will need to include the encoded form of any invalid characters present in a URL in order to enable a successful match. For example, the DOS pattern

```
http://*/a name with spaces.html
```

will not be able to match the following (invalid) URL

```
http://www.example.com/a name with spaces.html
```

because internally, this URL will be converted into the following valid, encoded URL:

```
http://www.example.com/a%20name%20with%20spaces.html
```

Thus, in order to successfully match the invalid URL, the pattern must be changed to

```
http://*/a%20name%20with%20spaces.html
```

In the real world, this problem should only affect a small number of URLs though.

### 5.7.1 Some words on DOS Patterns

DOS patterns are simple patterns similar to those traditionally used on the MS-DOS platform to select multiple files. The rules are very simple:

- A question mark character (?) matches any one arbitrary character.

- The special asterisk character (\*) is a placeholder for any number of arbitrary characters (zero or more).
- If a character is prefixed with a backslash (\), that character is used literally, even if it usually has a special meaning (for example \\* and \?).

Here are some examples of DOS patterns:

This DOS pattern ...	... matches
<code>http://*/h*.*</code>	All files the file name of which starts with the letter 'h'. Their location (server and directory) does not matter. However, this pattern also matches URLs which have at least one path element starting with an "h".
<code>http://www.example.com/*</code>	Any file on the Web server "www.example.com".
<code>*/test/*</code>	Any file on any server the path of which somewhere contains a directory called "test".
<code>http://*/picture???.gif</code>	Any GIF files on any Web server having the name "picture" followed by exactly three arbitrary characters, e.g. "picture000.gif", "pictureABC.gif" etc. However, note that this pattern also matches the URL "http://www.example.com/picture/XY.gif" (!)
<code>*\?id=1</code>	Any URL which has an argument section equal to "id=1". The argument section of a URL is the part preceded by a question mark, near the end of the URL. Note the use of a literal question mark symbol here: this "?" is not a placeholder for an arbitrary character in this case, as it is preceded by a backslash.

DOS patterns are useful for a number of simple applications. If more complex pattern matching is required though, the pattern type "Regular Expression" should be used.

### 5.7.2 Some Words On Regular Expressions

Regular Expressions are a very powerful pattern matching system that can be found in various text editors (BBEdit, GNU Emacs ...), programming languages (Perl ...), and UNIX shell commands (egrep, sed ...). Their usage does require some technical knowledge though, and they are thus to be considered one of the more advanced features of PageSucker.

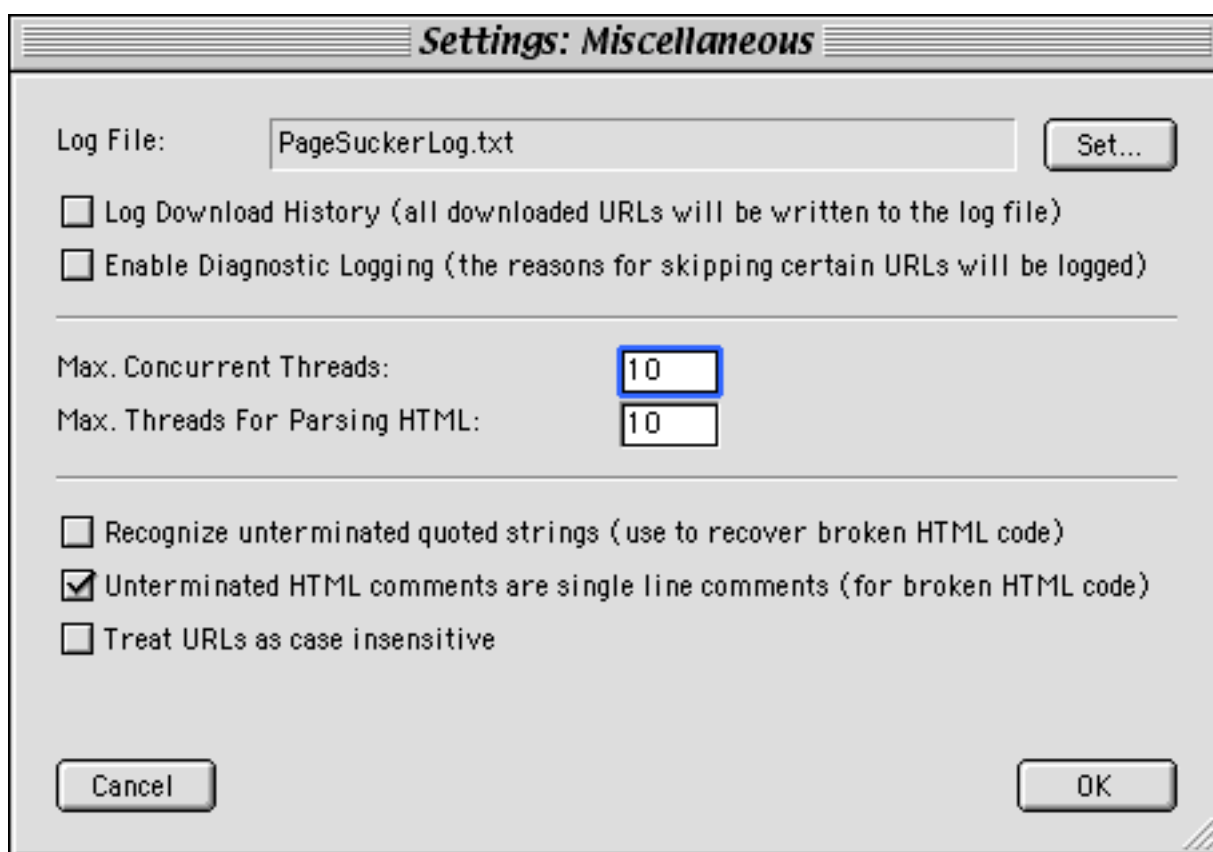
A complete documentation of regular expressions would be too vast to be included in this modest user manual. Let it therefore just be mentioned that PageSucker supports the Perl 5 regular expression flavor. For detailed explanations on the regular expression syntax, please consult some book on Perl, like "Programming Perl" by Larry Wall, Tom Christiansen & Randal L. Schwartz (published by O'Reilly & Associates). Another highly recommended O'Reilly book is "Mastering Regular Expressions" by Jeffrey E.F. Friedl. Various information can also be found on the Web, e.g. by using one of the standard Web search engines to look up the term "regular expression" or "regex".

Finally, here are some examples of regexes that are of direct use with PageSucker's URL matching filters:

This regular expression ...	... matches
<code>http://.+/?h[^\.]*\.jpg</code>	All JPEG files the file name of which starts with the letter 'h'. Their location (server and directory) do not matter.
<code>http://www\.\example\.com/.*</code>	Any file on the Web server "www.example.com".
<code>(http ftp)://.+/?test/.*</code>	Any file on any Web or FTP server the path of which somewhere contains a directory called "test".
<code>http://.+/?picture[0-9]{3}\.gif</code>	Any GIF files on any Web server having the name "picture" followed by exactly three decimal digits, e.g. "picture000.gif", "picture001.gif" etc.

## 5.8 The Miscellaneous Window

This window contains a collection of options which did not fit well into other settings windows. Select "Miscellaneous" from the "Settings" menu to open this window:



### ◆ Log File

This field specifies where to save the log file. It can be set by pressing the *Set* button next to the field. By default the log file is called "PageSuckerLog.txt" and is saved in the directory the PageSucker application resides in.

The log is a text file containing all kinds of information on the latest download process. Any error messages that may have

occurred during the download (and which have been displayed in the log window) will also be recorded in the log file. Please note that the log file will be overwritten each time a new download is started, unless you choose a new log file name between downloads.

When requesting technical support (see section 10 below) because of a problem related to downloading a certain site, it would be helpful if you sent the log file related to the problem download with your request.

#### ◆ Log Download History

Check this option if you want even more detailed information about a download to be included in the log file. PageSucker will list all encountered URLs in the log file. If this option is not checked, only those URLs that caused some kind of problem will be remembered in the log file. All URLs will be marked with one of the following terms when listed in the log file:

- ***SAVED*** – the file has been saved to the local disk
- ***PARSED*** – the URL has been scanned for new links but PageSucker has decided that there was no need to save the file locally
- ***ENCOUNTERED*** – the URL was seen by it was neither saved nor parsed
- ***ABORTED*** – the download was interrupted by the user before the file could be downloaded entirely
- ***ERROR*** – an error occurred while trying to download the file.

All log file entries will also be echoed to the console window.

#### ◆ Enable Diagnostic Logging

This is a very handy feature when you're trying to figure out why a certain file was not downloaded as expected during your last download. Did PageSucker not see the URL (possibly because the page's HTML code was invalid)? Or did you maybe forget to enable the right file types? Then again, perhaps the pattern you typed into the Pattern matching window is too restrictive ...

Sometimes it can be hard to figure out what is the exact reason why a download didn't work as expected. That's where diagnostic logging comes in. By enabling that feature, PageSucker will write an entry into the log file for all the URLs it encountered but decided not to parse or save. Typical entries you will find in the log file are of the form:

The URL "http://www.example.com/text.html" was neither parsed nor saved because it didn't match the global pattern matching filter.

You will then know that the specified URL was indeed found by PageSucker and why it was rejected. If the expected URL does not have such an entry in the log file and it was not saved to the local disk either, this means that PageSucker didn't encounter that URL, either because the page containing a link to that URL was not parsed or because the page containing the link was faulty somehow. To protect against faulty pages, you can try using the *Recognize unterminated quoted strings* option (explained below).

#### ◆ **Max Concurrent Threads**

To speed up file transfers, PageSucker can download and analyze more than one page at the same time. The *Max. Concurrent Threads* field sets the maximum number of downloads that can happen in parallel.

Usually the default value of 10 is a good setting, but the optimal value depends on the contacted Web server: if too many threads are used, the overall performance will be diminished, and some of the open connections may even fail or block. On the other hand, if too few threads are used, download times may also increase (for obvious reasons).

Please also note that the more threads are used, the more RAM is needed to run PageSucker.

#### ◆ **Max Threads For Parsing HTML**

PageSucker can only handle a certain number of links at any one time (set by the number of threads using the *Max Concurrent Threads* option explained above). Links that have been detected, but not yet downloaded are kept in a queue. The length of that queue is shown in the main window as *Remaining Documents*. When downloading a site that contains a great lot of links, it can happen that the queue grows very long with URLs not yet handled. This consumes a certain amount of RAM and as such tends to slow down the application.

PageSucker has two features that try to alleviate the above problem. First, data file downloads are always preferred over HTML file downloads, because data files (such as pictures and plain text files) never contain links that need to be handled, and which thus would have to be added to the queue. That's why PageSucker will always choose to download the data file first when it has to decide between a data file and an HTML file.

Second, PageSucker can be told to use less threads when downloading HTML files than when downloading data files. You can set the maximum number of threads to be used when downloading HTML files with the *Max Threads For Parsing HTML* option. The number must be less than or equal to the overall maximum number of threads.

◆ **Recognize unterminated quoted strings**

For most sites, you should leave this setting at its default value of 10 threads (the same as the overall maximum number of threads).

This option is useful only when downloading incorrect HTML code. A commonly overlooked error in HTML code is an unclosed quoted string. For example, you might find the following broken HTML code:

```
<A HREF="picture.jpg> Click here </A>
```

(note the missing double quote after “picture.jpg”). This code is accepted by older browsers (e.g. Netscape 1.0) but not understood by the more current browser versions (Netscape 3.0 or higher). Normally, PageSucker doesn’t understand such incorrect code. To counter this, you can check the *Recognize unterminated quoted strings* option, which will enable PageSucker to recognize unterminated strings, and even correct them in the downloaded file by adding the missing double quote.

However, enabling that option will also lead PageSucker to misinterpret correctly quoted strings that contain a ‘>’ character, which should not be a big problem though, as most strings will not contain any ‘>’ characters.

◆ **Unterminated HTML comments are single line comments**

The HTML language used to build Web pages allows the author of a page to include comments. The standard defines that all text enclosed by the markers `<!--` and `-->` is to be considered a comment. Now, it can happen that a certain Web page contains incorrectly terminated comments, i.e. parts of text which start with `<!--`, but for which there is no end marker (`-->`). A page containing this type of HTML error may cause PageSucker to skip links contained inside the page. By checking the *Single Line Comments* option, you can tell PageSucker to consider nothing but the first line of such incorrectly terminated comments.

By default this option is checked. Uncheck it only if you have a particular reason to do so.

◆ **Treat URLs as case insensitive**

While domain names are never case sensitive, this cannot always be said of URL paths and file names. For example, the following two URLs may not denote the same file:

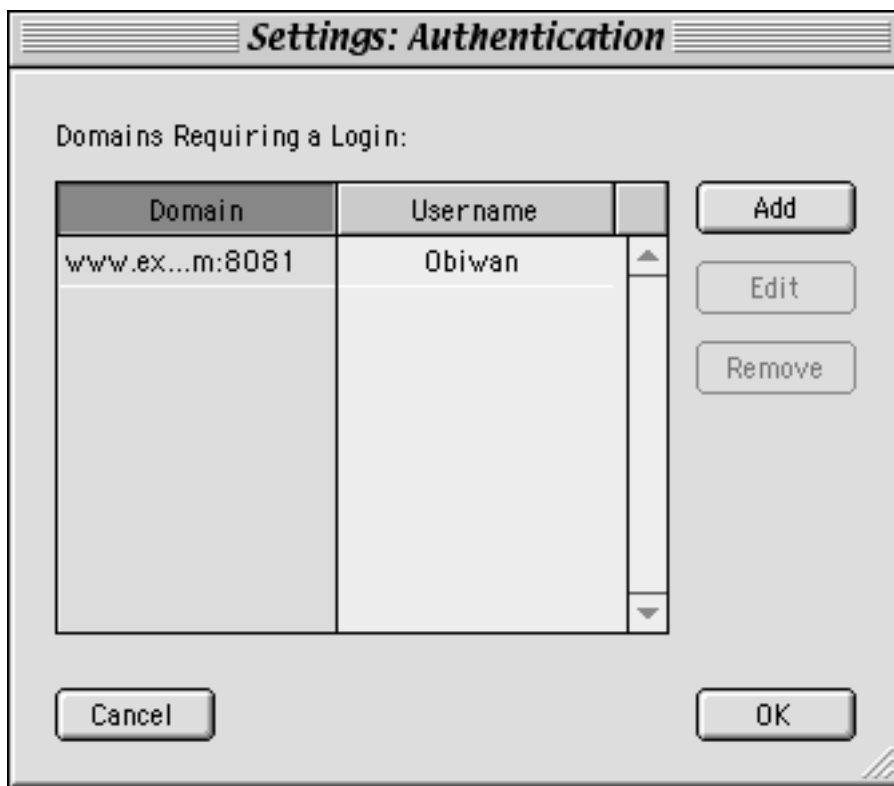
```
http://www.example.com/Picture.jpg  
http://www.example.com/pICTUrE.jpg
```

In practice however, both URLs point to the same file if the server on the “www.example.com” site is running an operating system which has case insensitive filenames, like Windows NT or the MacOS. If the server is running UNIX however, the two URLs are not equivalent.

That's why normally PageSucker considers URLs to be case sensitive (assuming a UNIX server). Thus, if both of the above URLs are detected in a Web page, they are both downloaded, possibly causing the same file to be downloaded twice (if the server is not running UNIX). To prevent that from happening, check the *Treat URLs as case insensitive* option.

## 5.9 The Authentication Window

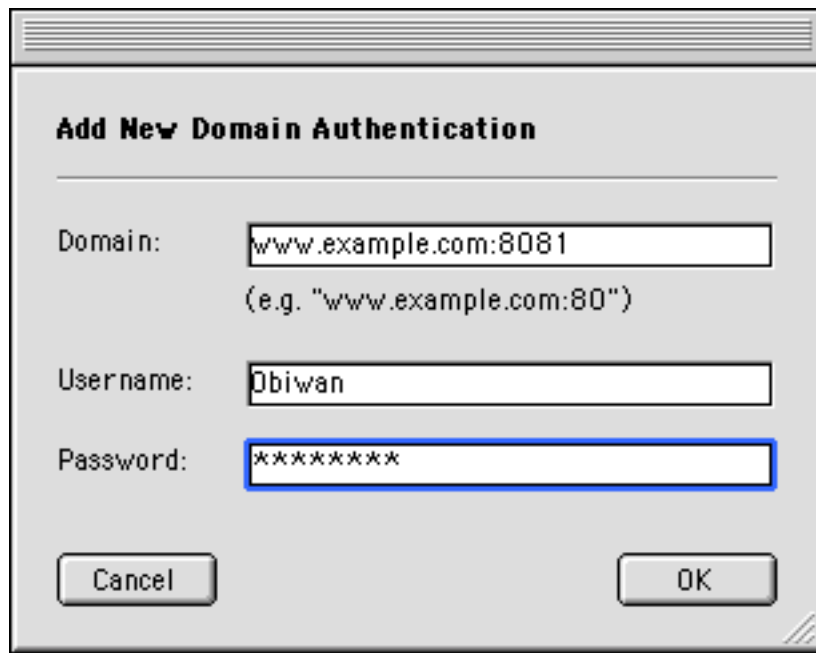
The *Authentication Settings* window is accessed by choosing “Authentication” from the “Settings” menu. Here you can define which domains require a login, i.e. a username and a password to be accessed. This enables PageSucker to access password protected Web sites.



To the left, you can see the list of domains for which a login has already been defined. The list can be sorted by domain or by user name, simply by clicking the appropriate table header. You can add a new domain by clicking the “**Add**” button. By selecting a domain in the list, then clicking the “**Edit**” button, you can modify this domain’s login. The same effect is achieved by double-clicking a domain name in the list. To remove a domain from the list, select it and click the “**Remove**” button.



When a domain is to be added or edited, the following window appears:



**Add New Domain Authentication**

Domain:   
(e.g. "www.example.com:80")

Username:

Password:

The **Domain** field should contain the domain name (without the “http://” protocol annotation). If the login applies to only a certain port in the domain, append the port number at the end of the domain name, separated from the name by a colon (:) - for example: “www.protected.com:9999”, where “www.protected.com” is the domain name per se, and “9999” is the port number.

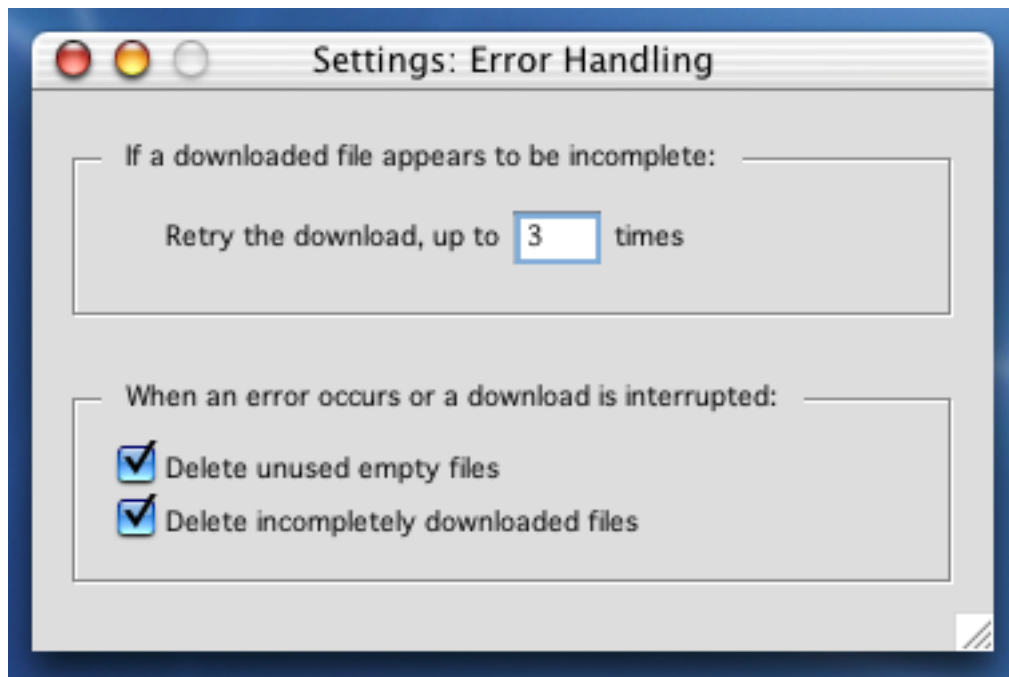
Enter the user name to be used in the **Username** field, then enter the corresponding password in the **Password** field. For security reasons, the password is displayed as all stars (\*\*\*\*) when you type it. It is however accepted correctly by the program.



**CAUTION:** Enter authentication information only for those domains that actually require a login! If you specify a login for a domain which doesn’t require one (or if you specify an incorrect password and/or username), PageSucker will not be able to download files from that domain, even if it really is unprotected.

## 5.10 The Error Handling Window

Select “Error Handling” from the “Settings” menu to display the following window:



◆ **Retry the download, up to xxx times**

Sometimes it can happen that the connection to a server gets lost during a download. This would normally result in half downloaded and thus invalid files. To make sure that each file was downloaded completely, PageSucker therefore double checks if a downloaded file appears to be incomplete: if the local and the remote files' sizes do not match, the server can then be contacted anew with the request to send the missing part of the file.

The “Retry the download, up to xxx times” setting controls how often a download is retried before PageSucker finally gives up. The special value “INF” can be used to specify endless retries. That value should be used carefully though, as the connections to some servers might fail continuously, and the application would then end up spending its time pointlessly retrying connections which are bound to fail over and over again. Usually the default value of 3 works well for most servers. If you have a particularly bad Internet connection though, you might want to increase that value a bit.

◆ **Delete unused empty files**

While downloading a site, PageSucker needs to create lots of files, which will be empty at first, and which will be filled up with data later on. If a download is interrupted though, or a particular file cannot be downloaded to to some problem (for example because the remote file cannot be found), the empty local file will remain on your disk, unless the “Delete unused empty files” option has been checked. This option, when checked, will make sure that such unused files will be deleted.

◆ **Delete incompletely downloaded files**

This option, when checked, will make sure that a half downloaded file will be deleted, if the download of that particular file cannot be completed due to an error, or because the entire download was stopped upon user request. If incomplete files can still be of some use though, you should uncheck this option.

## 5.11 Saving And Restoring Settings

When you start PageSucker for the first time, it uses certain default settings (the *factory settings*), which are useful for downloading most simple sites. You can modify various settings in the main window as well as in the dedicated settings windows. These modified settings are normally lost upon quitting the application (PageSucker does not automatically save modified settings).

To prevent this, PageSucker allows you to control which settings it uses the next time you start it up. Just select “Save Default Settings” from the “Settings” menu to remember the current settings as defaults. The next time PageSucker is launched, these settings will be used automatically. To get back to the original default settings, choose “Factory Settings” from the “Settings” menu.

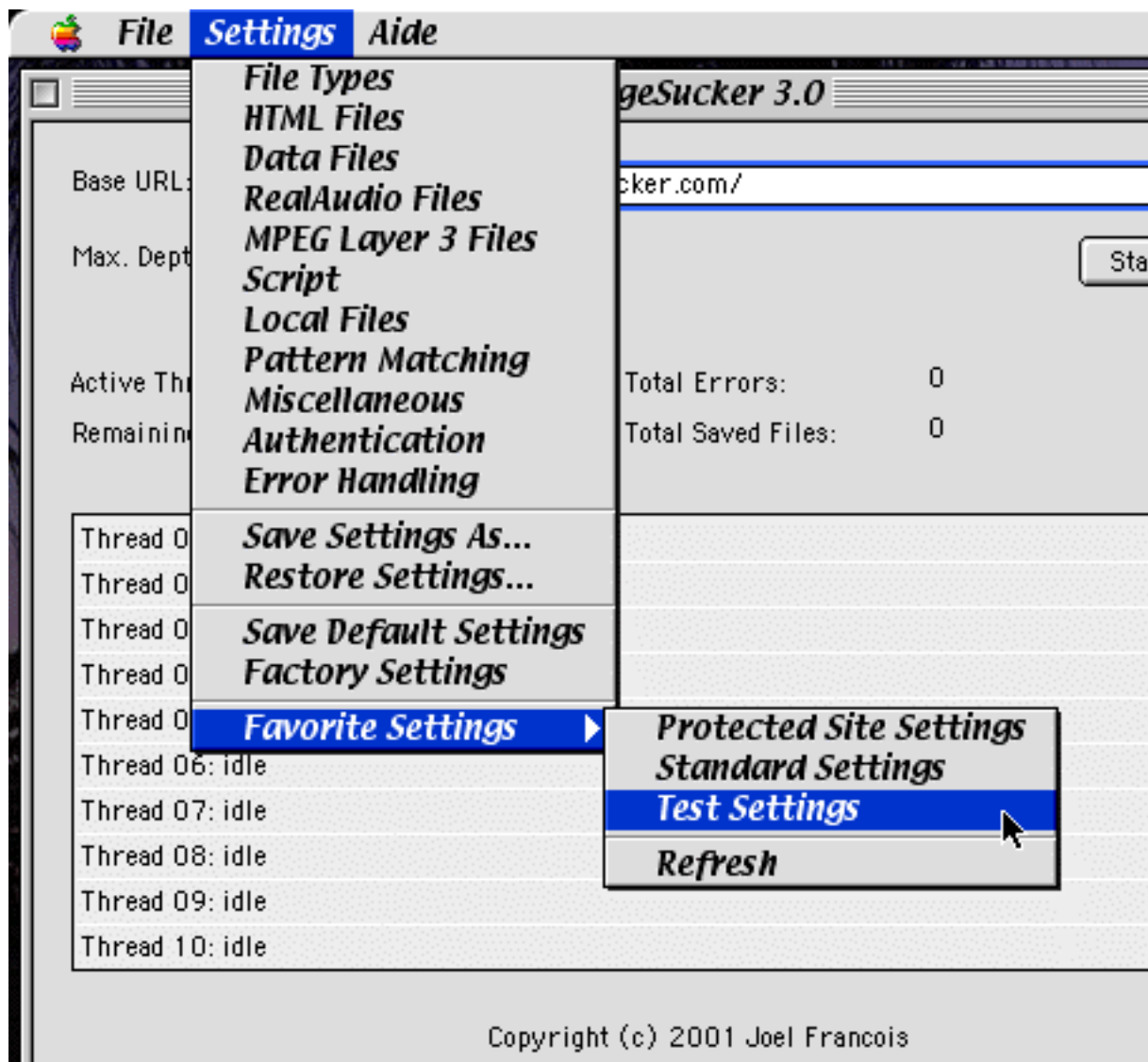
It often turns out that certain settings are good for certain downloads only. For example, you might want to have one set of settings for downloading pictures only, another one for downloading complete sites. To make this easier to handle, you can save as many settings sets<sup>5</sup> as you wish by using the “Save Settings As...” menu item in the “Settings” menu. You will be asked to choose a file name and location for your new settings file. Later on, these settings can be restored again by selecting “Restore Settings...” from the “Settings” menu, then choosing your desired settings file. When running on a Macintosh, you can also double-click a settings file in the Finder to cause PageSucker to load it<sup>6</sup>.

When saving your settings, you can place them into the special folder called “Favorite Settings”, inside the PageSucker folder. In that case, an entry will be added to the “Favorite Settings” submenu:

---

<sup>5</sup> I apologize for such unwieldy word constructs (and for using the word “settings” 200 times in this section), but I couldn’t come up with a better term :-)

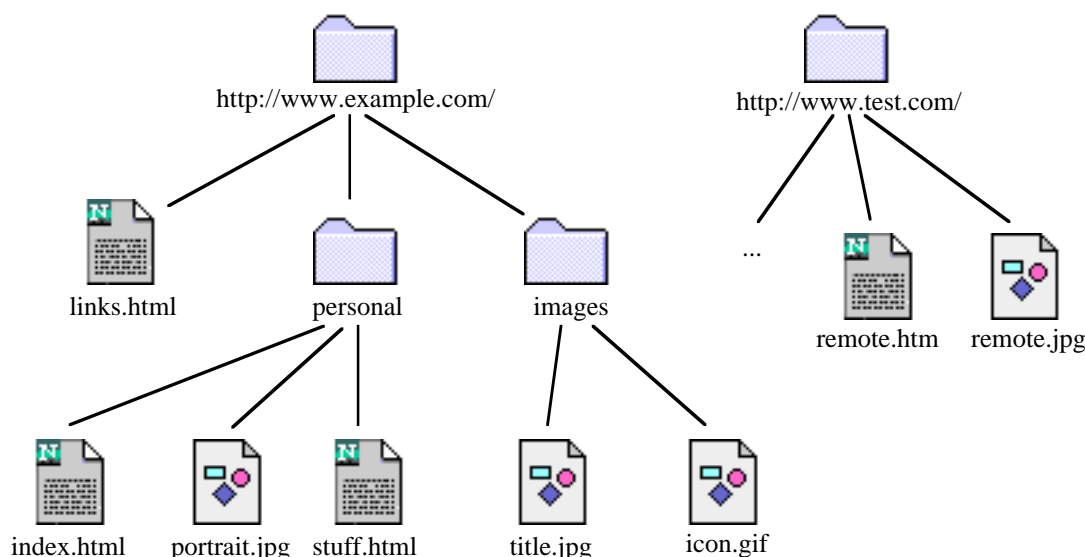
<sup>6</sup> Note that on MacOS X this currently only works if PageSucker has already been launched. Also note that settings files saved with an older version of PageSucker (3.0.1 or lower) will cannot be loaded with a double-click, as they don’t have the correct filetype and creator. You can change the type and creator of these files manually though (using one of the type utilities available on the Net). In that case, use “Sett” as filetype and “PSuk” as creator.



To restore one of the settings sets referred to in the “Favorite Settings” submenu, simply select the appropriate entry. If the contents of the “Favorite Settings” folder were modified outside of PageSucker, the submenu list may be out of date. In that case, use the “Refresh” item to rebuild the submenu.

## 6. HOW PAGESUCKER APPLIES FILTERS

Please note that all of PageSucker's filters are always applied collectively, i.e. for a file to be downloaded, all filters applicable to that file must accept the file. Let's consider an example to illustrate the filtering principles: the server with the (fictional) address "www.example.com" has the following structure:



There's a second server, "www.test.com", the detailed file structure of which is not shown. Furthermore, let's assume you use PageSucker's factory settings except for the following values which you entered in the main window:

<b>Base URL</b>	<code>http://www.example.com/personal/</code>
<b>Max. Depth To Dig</b>	1

The base URL doesn't have a filename, so the server chooses which file to send back to PageSucker, based on its settings. In this case, let's suppose it's the "index.html" file in the directory "personal". PageSucker doesn't know the name of that file, but as "index.html" is the most common default name for these cases, it saves the received data locally under that same name.

The "index.html" file is downloaded and analyzed because the default file type settings specify that HTML files should be parsed (there's a checkmark in the "Parse" column in *File Types* window). However, the file will be saved to the local disk only if there is also a checkmark in the "Save" column next to the entry for HTML files in the *File Types* window.

Let's assume "index.html" contains URLs pointing to the following files, either as embedded images, or as linked objects:

File	Kind	Discussion
stuff.html	linked HTML file	Downloaded and analyzed. Saved only if HTML files are set to be saved in the <i>File Types</i> window. Any links contained in this file will be ignored, as the maximum recursion depth of 1 is reached (we had to follow one link to reach this file already). Embedded images, if any, will be downloaded though, unless you uncheck the “ <b>Consider embedded files to be part of their host page</b> ” option in the <i>Data Files</i> window (it is checked by default).
links.html	linked HTML file	Not downloaded unless “ <b>Parse HTML pages not in hierarchy</b> ” is checked in the <i>HTML Files</i> window, as this file is not in the specified hierarchy (“links.html” is not a direct or indirect member of the directory “personal”).
portrait.jpg	embedded image	Downloaded and saved if “ <b>Save embedded files</b> ” is checked in the <i>Data Files</i> window, and the settings in the <i>File Types</i> window specify that “.jpg” files should be saved.
title.jpg	linked data file	Downloaded and saved only if both the “ <b>Save linked data files</b> ” and “ <b>also data files outside of hierarchy</b> ” options are checked in the <i>Data Files</i> window, as the file is not contained in the hierarchy starting at the directory “personal”. Of course the <i>File Types</i> window will be consulted here too, to find out if “.jpg” files should be saved in general.
icon.gif	embedded image	Downloaded and saved only if both the “ <b>Save embedded files</b> ” and “ <b>also embedded files outside of hierarchy</b> ” options are checked in the <i>Data Files</i> window, and “.gif” files are set to be saved in the <i>File Types</i> list.
remote.jpg	embedded image	On remote server, thus not saved by default as both the “ <b>Save embedded files</b> ” and “ <b>also embedded files on remote server</b> ” options would need to be checked in the <i>Data Files</i> window.
remote.html	linked HTML file	On remote server, thus not downloaded unless the <b>Parse HTML pages on remote server</b> option is checked in the <i>HTML Files</i> window.

This is but a simplified example, as the more advanced filters (such as the pattern matching filters) have not been taken into account here. In practice, don’t forget that all the filters must accept a file for it to be parsed or saved. If only one single filter rejects a URL, it will not be considered for download. Should PageSucker unexpectedly reject a file you are trying to download, you can use the *Diagnostic Logging* feature to track down which filter is causing the problem. See section 5.8 for more info on that option.

## 7. THE FILE MENU

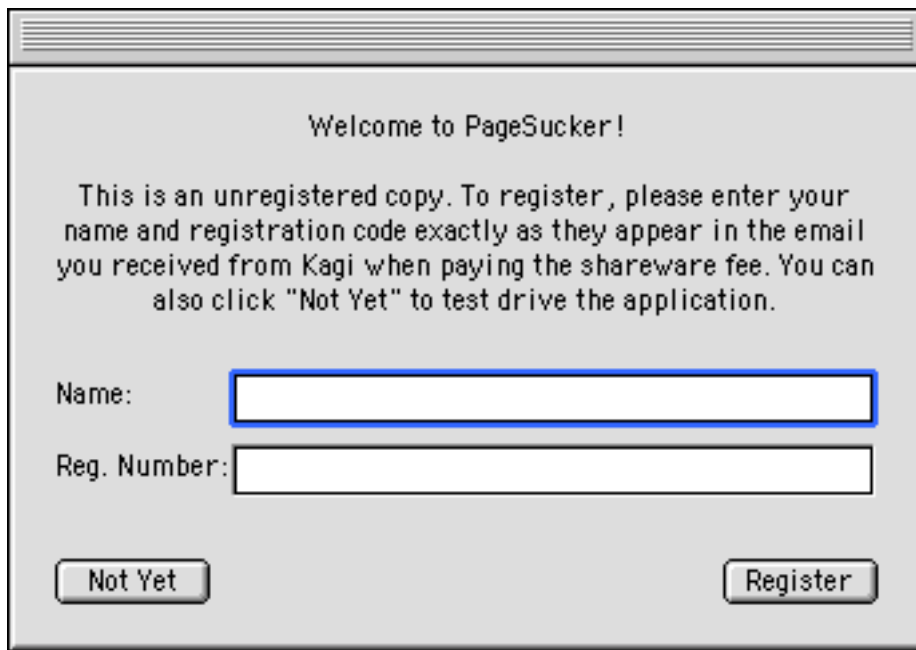
The File menu contains just a few items:

- The “Close” command (present in the Mac version only) can be used to close the main window or the log window. This is the same as clicking a window’s close box. Note that closing the main window will actually quit the application.
- The “Quit” command quits the application. If a download is currently running, PageSucker will show a warning that quitting will interrupt the download. Note that on MacOS X, the “Quit” command can be found in the application menu, which is the menu called “PageSucker” to the left of the File menu.
- The other items allow you to register and unlock the application after paying the shareware fee, check if a new version has been released, and adjust the application preferences. These items will be detailed in the following sections.



### 7.1 Registering PageSucker

When you first start PageSucker, it will ask for a name and a registration code:



Click the “Not Yet” button if you’d like to test drive the application. You’ll find that certain options are disabled when in demo mode. When trying to activate such an option, PageSucker will inform you that this only works with the registered version. For example, in demo mode you are limited to three threads (three simultaneous connections to a server), while there is no such limitation once PageSucker has been unlocked.

Ok, so where do you get the registration code needed to unleash the application’s full power? Well, PageSucker is distributed under the concept of shareware. That means that it can be distributed for free and you can test and use it for as long as you like in demo mode without paying anything. However, if you find it useful and if you’d like to use the more advanced features, you should pay the modest fee of **USD 10.-** (*single user license*).

If you’d like to license PageSucker for a company, you can also purchase a *site license* for the price of **USD 300.-** It covers all locations for your organization. One big advantage of a site license is that you do not need to keep track of how many people at your site are using the software.

### 7.1.1 Registering online

---

Paying for PageSucker is fairly simple. You can buy a license online with your credit card in just a couple of minutes. In order to do so, simply double-click the icon labeled "Register Online" in the PageSucker folder. Alternately, you can use your favorite browser to surf over to the homepage at

<http://www.pagesucker.com>

and click the "Register" button. You will be taken to the secure PageSucker Online Store (powered by eSellerate). Please note that this Web site uses strongly encrypted connections, to ensure the safety of your credit card data while in transit.



### 7.1.2 How to use your registration code

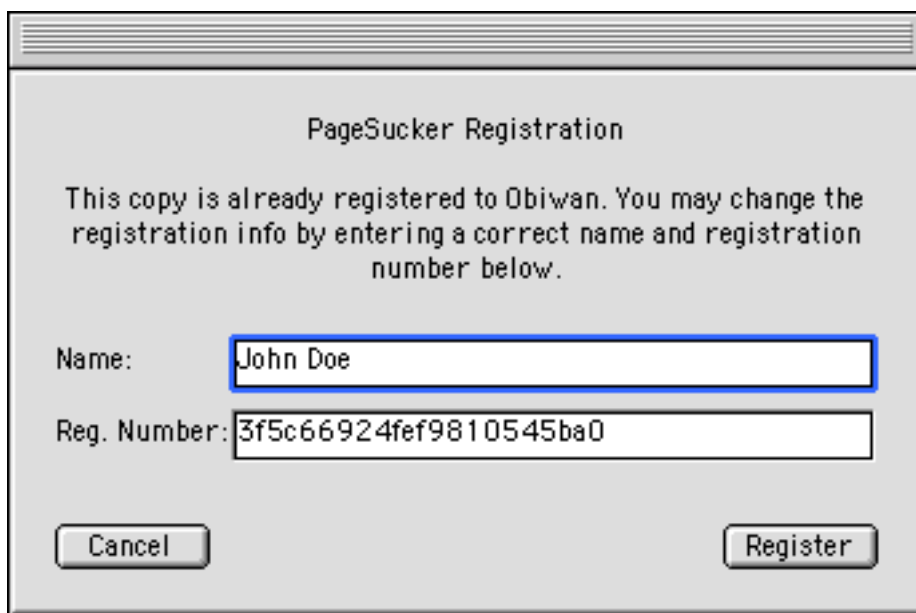
The good part is that once you have purchased a license, you'll get a registration code, either directly online (if you registered with a credit card via eSellerate), or from Kagi via email (if you registered with a check or cash). You will have to enter the registration name and number into PageSucker's registration dialog (shown above), then click the "Register" button. If your code was entered correctly, PageSucker will unlock itself. Please note that your registration information is personal and may only be used with the copy/copies of PageSucker it was purchased for. So, please don't give away your code, OK?

Here's a sample<sup>7</sup> registration code, such as the one you'll get after paying:

Name: John Doe

Reg. Number: 3f5c66924fef9810545ba0

Please note that the name and the registration number are linked together: both elements must be entered correctly (including upper and lowercase letters) for the code to be accepted by PageSucker. The best way to get it right is to copy/paste the name and the number into PageSucker's dialog:

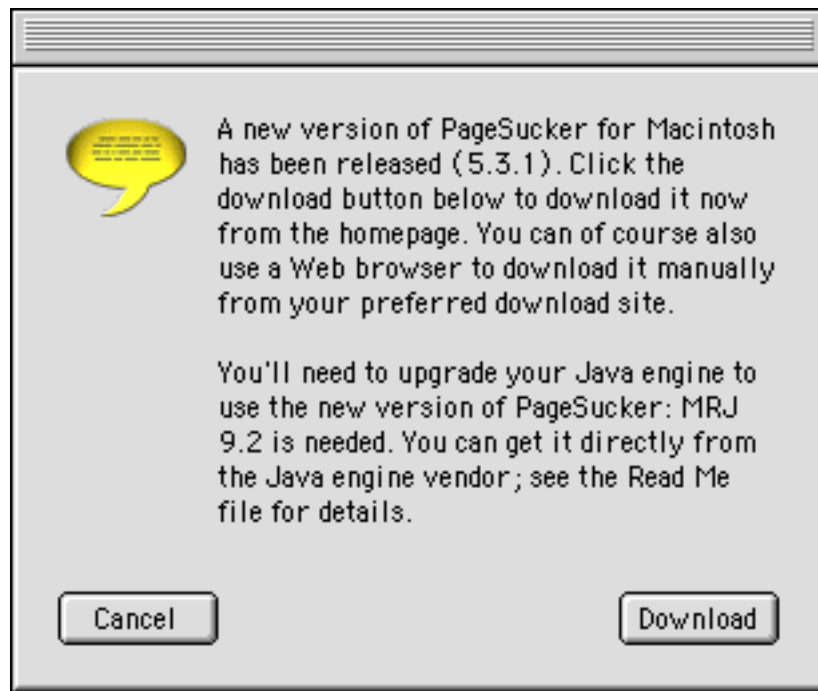


Should you ever want to change your registration information once you have entered it, you can do so by selecting "Register..." from the "File" menu. This command also allows you to unlock a demo version without the need to restart the application.

## 7.2 Checking For A New Release

If you're using PageSucker on Macintosh or Windows, the "Check For New Release..." item in the "File" menu makes it real easy to find out if you're running the most recent release of PageSucker. This item is not available when running on another platform. Before selecting this command, make sure that you are connected to the Internet. PageSucker will then check via the Web if a new version has been released. It will even inform you if you'll need to upgrade your Java engine in order to run the new version. Below is an example dialog you might see.

<sup>7</sup> This really is only a sample, so don't even think of trying the code shown here – it won't work ;-)



This informs you that the newest release for your platform (Macintosh in this case) is PageSucker version 5.3.1. It also tells you that this new release will not run reliably with your current Java engine and that you should upgrade to MRJ version 9.2 before using it<sup>8</sup>. At this point, you can just click “Cancel” to close the window and do nothing further. However, you can also click “Download” to automatically get the newest release from the home server. Please note that this will not download a newer Java engine though (even if one is needed). You will have to do that manually using a standard Web browser. Also, depending on where you are located, it may be more efficient for you to use a standard Web browser to download PageSucker from some Web site or FTP server other than the home page, because of a potentially faster connection.

How does the “Check For New Release” function work? Well, it’s real easy, actually: PageSucker uses the Internet to get a small text file from the home server. This text file contains information on the newest release available, in a machine readable form. PageSucker then compares that information to its own version number to decide if it is up to date.

For the technically minded among you, you can download the information text file with a standard Web browser, just for fun. The URLs are (for MacOS Classic, MacOS X and Windows versions respectively):

```
http://www.pagesucker.com/ReleaseInfo/Macintosh.dat
http://www.pagesucker.com/ReleaseInfo/MacOSX.dat
http://www.pagesucker.com/ReleaseInfo/Windows.dat
```

---

<sup>8</sup> These version numbers are of course only bogus numbers for the example. As of this writing there is no version 9.2 of MRJ, just as PageSucker version 5.1.3 does not exist (yet)!

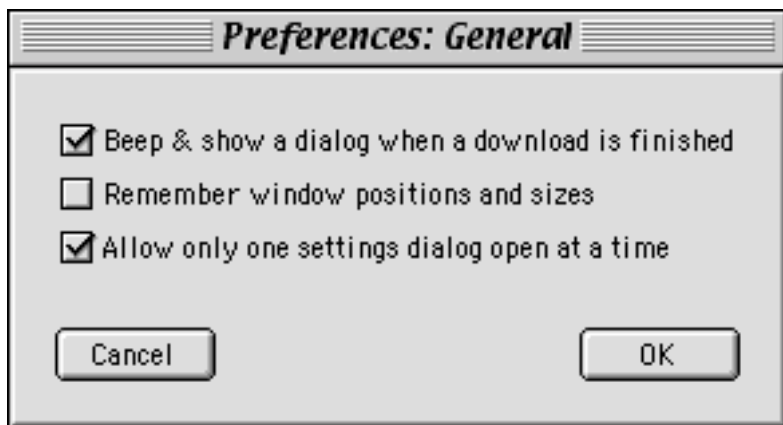
## 7.3 Adjusting The Application's Preferences

The “File” menu contains a submenu entitled “Preferences”, which has two more items, discussed below. These items let you adjust the overall application preferences. The terms *preferences* and *settings* should not be confused, as there are a number of differences between both concepts:

- Settings are adjusted via the various settings windows (detailed in section 5), whereas preferences are set via the “Preferences” submenu in the “File” menu.
- Settings are relative to one particular download: they include the base URL to download and all filters necessary to achieve the desired result when running that download. Preferences on the other hand are valid application-wide; they control those adjustments which are not related to one particular download operation.
- Settings are never saved automatically; if you made some changes to your settings, you have to use one of the “Settings” menu’s commands to save them permanently (see section 5.11 above). Preferences however are always automatically saved when the application quits, so you won’t have to adjust them again next to you start PageSucker.

### 7.3.1 General Preferences

Select “General...” from the “Preferences” submenu to open the following window:



Below is a detailed description of what each of the window’s options does.

After selecting the desired options, click “OK” to close the window and retain your changes. Click “Cancel” if you changed your mind and would prefer not to modify your preferences after all.

◆ **Beep & show dialog when a download is finished**

If this option is checked, PageSucker will emit a sound and show a dialog whenever a download process finishes.

◆ **Remember window positions and sizes**

This option causes PageSucker to keep track of the size and position of all PageSucker windows on your screen<sup>9</sup>. With this option checked, you can close a window and when it is reopened later on, it will appear at exactly the same spot as it was when you closed it. What’s more, this even works across sessions: if one or more windows are open when you quit PageSucker, these windows will be automatically reopened at the next launch. To avoid this from happening, uncheck the *Remember window*

<sup>9</sup> Currently only the main screen is supported. If your computer has more than one monitor connected, windows placed on a secondary monitor will be restored to the main screen when they are reopened.

◆ **Allow only one settings dialog open at a time**

*positions* option.

Check this option if you don't want to have several settings windows open at the same time. In that case, each settings window will have a "Cancel" and an "OK" button near the bottom of the window. Both of these buttons can be used to close the window, but any modifications done while the window was open will only be retained if the "OK" button is clicked.

If the *Allow only one settings dialog open* option is unchecked, you can open as many settings windows as you wish without first closing the other open windows<sup>10</sup>. In that case, the windows will have a close box, but no "OK" or "Cancel" buttons. All settings modifications done in the windows will become effective immediately. There is no undo or cancel option in that case! However, if a download is currently running, the modified settings will not affect the running operation, but the next download only. This mode can be useful if you have a large monitor and want to keep the most frequently used controls close at hand.

### 7.3.2 Proxy And Firewall Settings

If your computer is located behind a firewall or is using a proxy server (such as is common nowadays in major companies), you will need to adjust your proxy preferences before you can use PageSucker. You should ask your network administrator or your Internet provider what are the names and ports of the proxy servers you are using. Select "Proxies & Firewalls..." from the "Preferences" submenu to open the following window:

**Preferences: Proxies & Firewalls**

HTTP Proxy Host:  Port:

FTP Proxy Host:  Port:

Proxy Authentication

Username:

Password:

<sup>10</sup> This is the way previous (pre 3.0) versions of PageSucker handled settings windows.

◆ **HTTP Proxy Host**

Enter the name of your HTTP proxy host here, e.g. “myproxy.domain.com”. You don’t need to prefix the proxy host name with “http://”. Leave the field blank if you don’t want to use a proxy server. The HTTP proxy is the one used to access resources over the World Wide Web. The proxy’s port number should go into the field labeled *Port*, directly behind the host name field.

◆ **FTP Proxy Host**

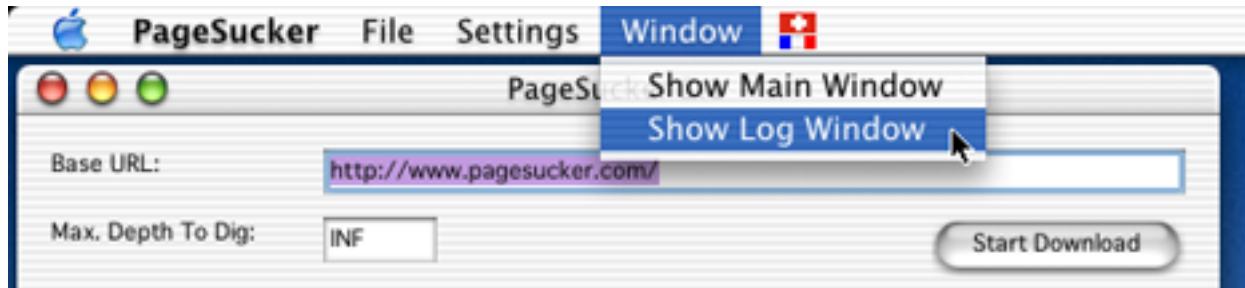
Enter the name of your FTP proxy host here, e.g. “myproxy.domain.com”. You don’t need to prefix the proxy host name with “ftp://”. Leave the field blank if you don’t want to use a proxy server. This proxy is used when accessing resources via the File Transfer Protocol. The proxy’s port number should go into the field labeled *Port*, directly behind the host name field.

◆ **Proxy Authentication**

If your proxy server requires a login, enter your username and your password in the respective fields. CAUTION: Only enter login information if you are sure that it is needed for your proxy server, otherwise PageSucker will not be able to connect to servers on the Internet. If in doubt, ask your network administrator.

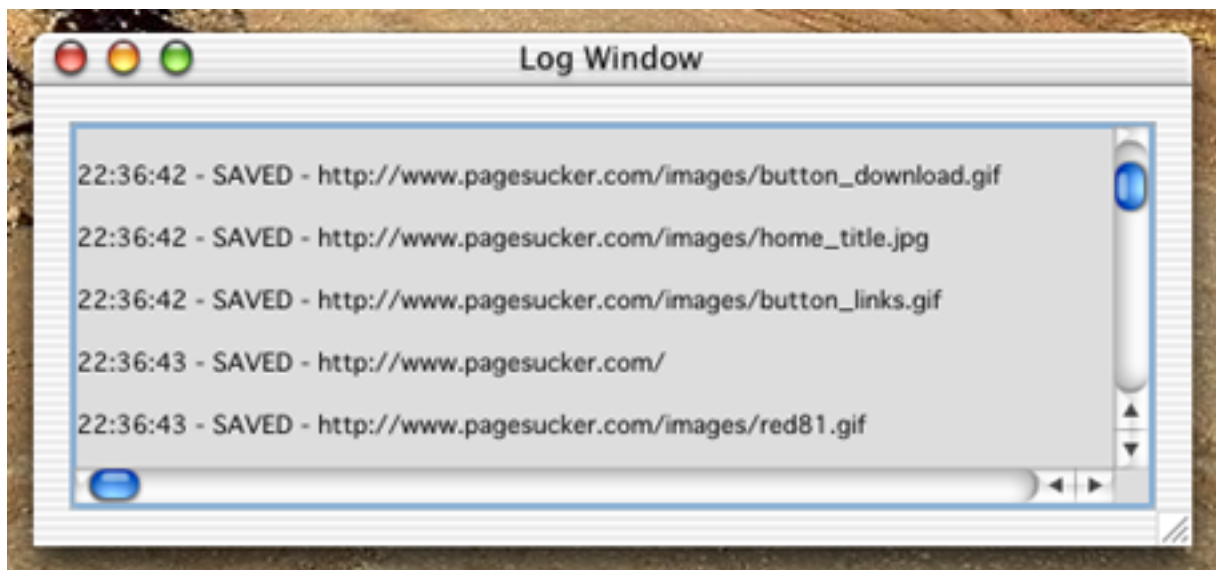
## 8. THE WINDOW MENU & THE LOG WINDOW

To the right of the *Settings* menu is the *Window* menu:



The first item, “Show Main Window”, can be used to, well, show the main window (in case it has been covered with the log window). This command is available in the Macintosh version only, as on Windows, the menu bar is attached to the main window, so it would hardly make sense to include this command.

The second item, “Show Log Window,” will open and show the log window. The log window is a text window which contains all kinds of output from the application, such as error and informational messages:



When the application starts up, the log window is not visible. It will automatically be shown when a log message needs to be displayed for the first time. The window can be closed by clicking its close button; in that case it will be hidden until the “Show Log Window” menu command is selected. Most messages displayed in the log window will also be recorded in a log file, by the way (see section 5.8 for more info on this).

## 9. KNOWN BUGS AND LIMITATIONS

- Downloaded HTML pages will be modified by PageSucker to point to the local copies of other files downloaded in the same process. If however a page contains URLs of files that didn't qualify for download or for being saved locally, these URLs will continue to point to the original remote server. When the downloaded pages are then viewed later on with a browser when the machine is off-line, these unmodified URLs may cause the browser to show "undefined picture" icons or even to attempt to connect to the Internet. This is perfectly normal and not to be considered a bug in PageSucker.
- If PageSucker is used to download HTML pages that contain errors (other than the ones described in section 5.8) the download may not work correctly. Support for the recovery of other types of HTML errors might be added in a future release.
- Currently, there are still a number of problems with the available Java interpreters. As PageSucker is directly dependent on the underlying interpreter, you might note a number of problems that are due to interpreter bugs. Generally, you should use the interpreter version recommended in the Read Me file to avoid most problems. Often a newer interpreter will also work well, but older versions should be avoided.
- There may be problems when trying to download files from a server the operating system of which allows longer filenames than the local machine's file system. Although PageSucker shortens filenames and removes illegal characters as necessary, the locally created directory structure might not exactly correspond to the server's directory structure due to name clashes.
- These same name clashes may cause certain files not to be downloaded unless one of the "Modify new file's name by prefixing/suffixing a number" options is selected in the *Local Files* window. To avoid this kind of problem, make sure to get unique file names by using one of these two options.
- Java applets and certain kinds of embedded data usually interpreted by browser plugins (including Macromedia Flash files) are currently not recognized by PageSucker. Support for such files will be added in a future release.
- Connections to a server that block for one reason or another cannot always be interrupted, nor can a time-out be directly defined by a Java application. Depending on the Java interpreter, these blocked connections might generate a time-out after a certain time or just remain blocked forever. Hopefully more control over this kind of problem will be added in a future Java release. For now, the only solution is to quit and restart PageSucker if certain connections seem to be stuck for an elongated period of time.
- Test results showed that the application's execution speed is reduced when a very large number of files is being downloaded. This is probably due to internal data structures becoming too large. Try to avoid downloading more than about 2000 files in one pass if you are experiencing this problem.
- If the target Web site features dynamic Web pages, i.e. pages generated at runtime by a CGI program residing on the server, single pages may be downloaded, but it is technically

impossible to download the CGI programs themselves. As a result, the downloaded pages will be static snapshots of the server's state at the time of the download.

- Certain Java interpreters seem to have stability problems when memory becomes tight and might crash the system in low memory situations. There is nothing a Java program can do about this. As a work-around, avoid low memory situations by either increasing the available free memory or specifying a lower number of threads in PageSucker's *Miscellaneous Settings* window.
- The current PageSucker version only recognizes Macintosh (Classic and OS X) and Windows 95/98/NT/2000/XP platforms correctly. There is also a limited support for the OS/2 platform and for some UNIX flavors. All other systems are assumed to have very rigid conventions when it comes to naming files. To be on the safe side, PageSucker attributes MS-DOS style filenames to all unknown filesystems. Support for more systems will probably be added in some future release.
- As noted in section 5.7, URLs containing illegal characters (such as spaces for example) can be matched with a pattern only if the pattern contains the encoded form of those invalid characters. An encoded character takes the form %xx, where "xx" is a hexadecimal number. Hopefully a better solution for this inconvenience will be implemented in a future release of PageSucker.
- Web pages asking for an authentication (username and/or password) via a Web form cannot currently be downloaded, as Web forms are not supported in the current release. Web forms are standard Web pages containing elements like text fields, checkboxes etc. On the other hand, pages asking for an authentication via a browser generated pop up dialog when viewed with a standard Internet browser can typically be handled by PageSucker. See section 5.9 for more information.



## 10. CONTACT INFORMATION

To contact the author for comments, suggestions and bug reports, send an email to the following address:

**`support@pagesucker.com`**

When submitting bug reports, please include the log file generated by PageSucker during the session when the problem occurred. This log might help me reproduce the problem. If I cannot reproduce it, chances are that I'll not be able to find and correct the bug.

## 11. THE LEGAL STUFF

This product is distributed “as is”. The author makes no warranty whatsoever on its functionality and cannot be held responsible for direct or indirect damage that it might cause to your software, hardware, health or other things. The author retains the full copyright © on all aspects of the product. Modification or reverse engineering of the compiled program code is not allowed. Under no circumstances may a modified version of this product or the accompanying documentation be distributed. Distribution of this package is allowed, even under decompressed or recompressed form, provided that it is complete and unmodified, that all original documentation is included and that no money is charged for the product. This product may be included in software collections on-line or on CD-ROM.

Macintosh is a trademark of Apple Computer, Inc. Windows is a trademark of Microsoft Corporation, Inc. Java is a trademark of Sun Microsystems, Inc. Flash is a trademark of Macromedia Inc. Parts of the product code are copyright © by Original Reusable Objects, Inc.

## 12. INDEX

This index contains the names of controls, menu items and windows you might see when using PageSucker. It can be used to quickly find information about how these controls should be used.

Accept M3U files not in hierarchy .....	32	Modify new file's name by suffixing a number .....	38
Accept M3U files on remote server .....	32	MPEG Layer 3 Files Window .....	31
Active Threads .....	13	Only save files marked in the list above .....	23
Allocated Memory .....	14	Overwrite existing file .....	38
Allow only one settings dialog open at a time .....	60	Parse .....	21
also data files on remote server .....	30	Parse HTML pages not in hierarchy .....	26
also data files outside of hierarchy .....	30	Parse HTML pages on remote server .....	26
also embedded files on remote server .....	30	Parse stylesheet code to find embedded URLs .....	28
also embedded files outside of hierarchy .....	29	Parse URLs only if they match the following pattern .....	42
Also save files the type of which could not be determined .....	23	Pattern Matching Window .....	41
Authentication Window .....	48	Preferences .....	59
Base URL .....	12	Proxies & Firewalls... ..	60
Beep & show dialog when a download is finished .....	59	Proxy Authentication .....	61
Check For New Release... ..	57	Quit .....	55
Close .....	55	RealAudio Files Window .....	31
Complement existing file, if incomplete .....	38	Recognize JavaScript in tags .....	37
Consider embedded files to be part of their host page .....	30	Recognize MP3 files linked via "M3U" files .....	32
Consider frames to be on same recursion level as main page .....	27	Recognize unterminated quoted strings .....	47
Consider linked data files to be part of their host page .....	31	Recursion depth .....	17
Consider M3U files to be part of their host page .....	33	Register... ..	57
Create a direct link to the M3U file's first URL .....	33	Regular Expressions .....	43
Data Files Window .....	29	Remaining Documents .....	13
Delete incompletely downloaded files .....	51	Remember window positions and sizes .....	59
Delete unused empty files .....	50	Restore Settings... ..	51
Description .....	21	Retry the download, up to xxx times .....	50
Don't save M3U files, just take a look at the contained URL(s) .....	33	Save .....	21
DOS Patterns .....	42	Save Default Settings .....	51
Enable Diagnostic Logging .....	45	Save embedded files .....	29
Error Handling Window .....	50	Save external stylesheet files .....	28
Extensions .....	21	Save linked data files .....	30
Factory Settings .....	51	Save M3U files, modifying the contained URL(s) .....	33
Favorite Settings .....	51	Save Settings As... ..	51
File Menu .....	55	Save URLs only if they match the following pattern .....	42
File Types Window .....	19	Script Window .....	35
FTP Proxy Host .....	61	Settings Menu .....	19
Guess URLs in JavaScript blocks .....	36	Show Descriptions .....	21, 23
Guess URLs in selectors .....	37	Show Log Window .....	62
Honor MIME Types .....	22, 23	Show Main Window .....	62
HTML Files Window .....	25	Start Download .....	14
HTTP Proxy Host .....	61	Stop Download .....	See Start Download
If a file to be saved already exists .....	38	Thread Status .....	
Ignore URLs if they match the following pattern .....	42	Downloading .....	15
Local Ext .....	22	Gathering info .....	16
Local Files Window .....	37	Idle .....	15
Log Download History .....	45	Opening/closing connection .....	15
Log File .....	44	Parsing .....	15
Mac Creator .....	23	Total Errors .....	13
Mac Type .....	23	Total Saved Files .....	14
Max Concurrent Threads .....	46	Treat URLs as case insensitive .....	47
Max Threads For Parsing HTML .....	46	Unterminated HTML comments are single line comments .....	47
Max. Depth To Dig .....	13	Untyped URLs denote .....	27
Maximum recursion depth .....	17	Use absolute paths in saved M3U files .....	33
MIME Type .....	22	Use alphanumeric local filenames .....	40
Miscellaneous Window .....	44	Use existing file, unless empty .....	38
Modify filename extensions .....	22, 23	Window menu .....	62
Modify new file's name by prefixing a number .....	38		