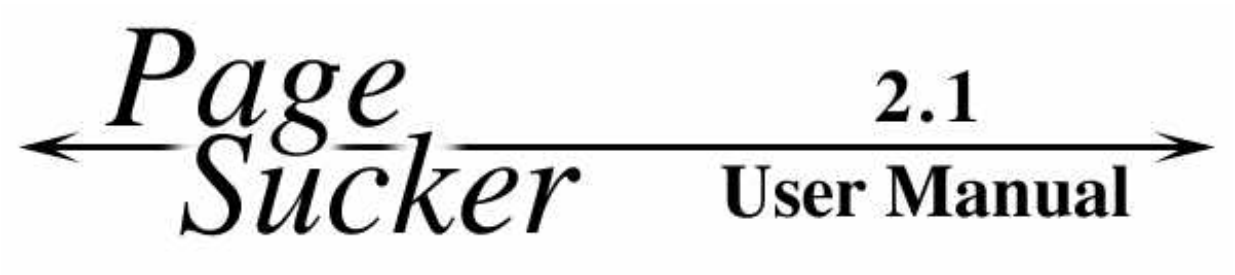


1. TABLE OF CONTENTS

1. Table Of Contents *2*
2. What Is It? - An Introduction And Some Background Information *3*
3. Installation *4*
4. Getting Started – A Basic Tutorial *5*
4.1 Example 1 – Downloading A Complete Web Site **5**
4.2 Example 2 – Downloading Selected Data File Types Only **6**
5. The Main Controls *9*
6. When To Stop? - The End Of A Download Process *12*
7. The Settings Menu *13*
7.1 The HTML Files Window **13**
7.2 The Data Files Window **17**
7.3 The RealAudio Files And MPEG Layer 3 Files Windows **19**
7.4 The JavaScript Window **22**
7.5 The Options Window **24**
7.6 The Expert Window **26**
7.7 The Proxy Window **31**
7.8 The Authentication Window **32**
7.9 Saving And Restoring Settings **33**
8. How PageSucker Applies Filters *34*
9. The File Menu *36*
9.1 Registering PageSucker **36**
9.2 Checking For A New Release **38**
10. Some Regular Expression Examples *40*
11. Known Bugs And Limitations *41*
12. Contact Information *43*
13. The Legal Stuff *44*

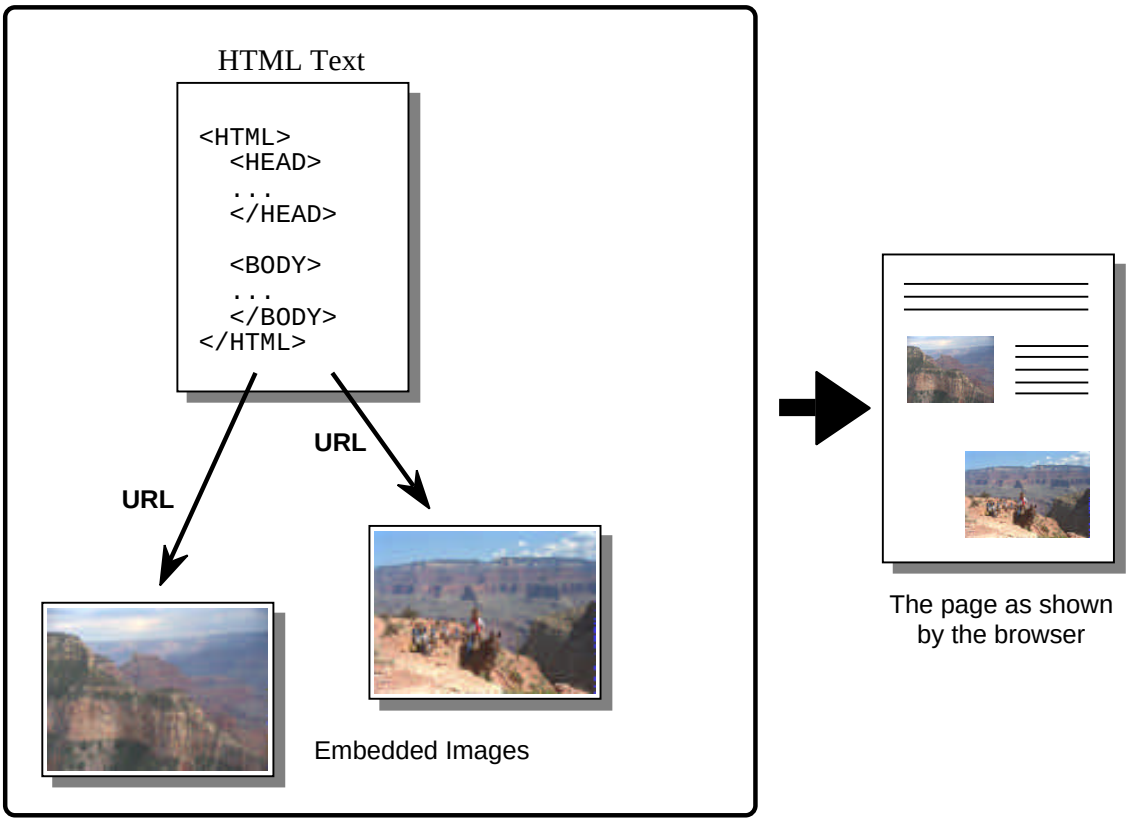
2. What Is It? - An Introduction And Some Background Information

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.mediatel.lu"), a file path in the server's filesystem, and a file name.

`http://`	`www.mediatel.lu`	`/crmm/art_mm/`	`h_art_mm.html`
protocol	*server address*	*file path*	*file name*

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 (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, all 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.

3. INSTALLATION

For installation instructions, please refer to the separate external "Read Me" document.

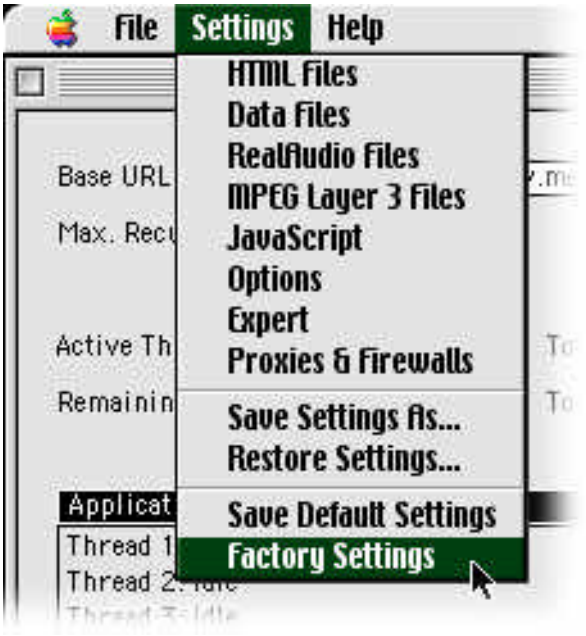
4. GETTING STARTED – A BASIC TUTORIAL

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 these 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 all functionality is identical.

4.1 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 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 you 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 sure: select "Factory Settings" from the "Settings" pull-down menu.

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

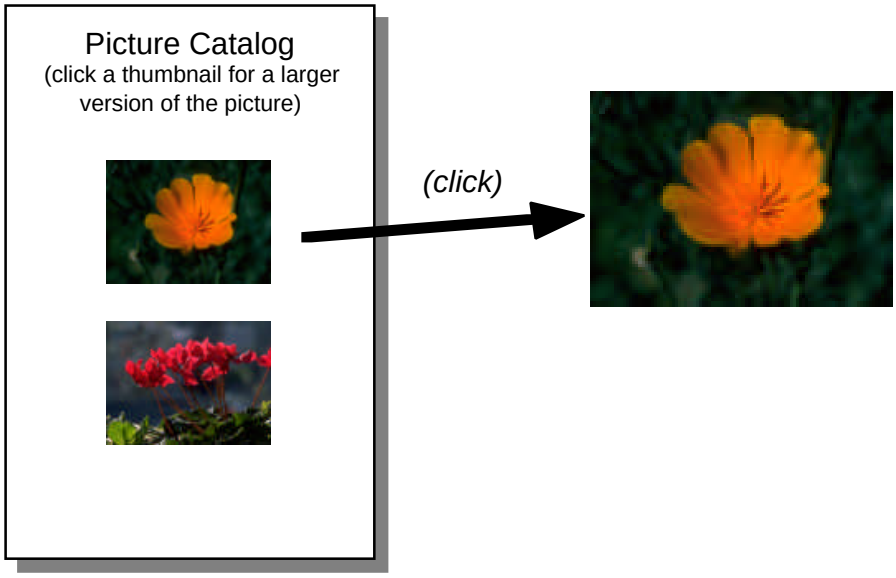
The URL displayed here is only an example which will not work for real. So choose a URL of a real server you are interested in. Then click the "Start Download" button in the main window.

PageSucker will ask for a download directory via a standard system file dialog. Navigate to an empty directory on your disk and open it. 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".

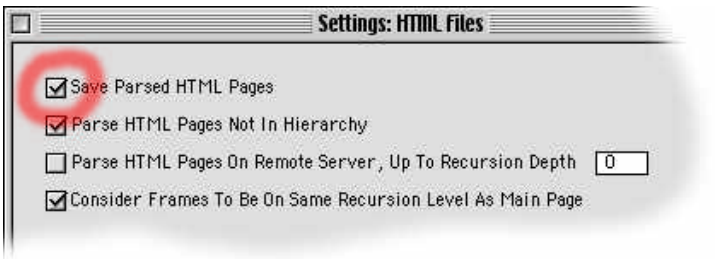
When the download is complete, PageSucker will emit a beep and display a dialog informing you of the end of the download process. You will then find a directory hierarchy on your local disk which reflects the directory structure of the remote server. You can now open the local HTML files from your favorite Web browser to view them.

4.2 Example 2 – Downloading Selected Data File Types Only

Let's suppose you have the URL of a 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.



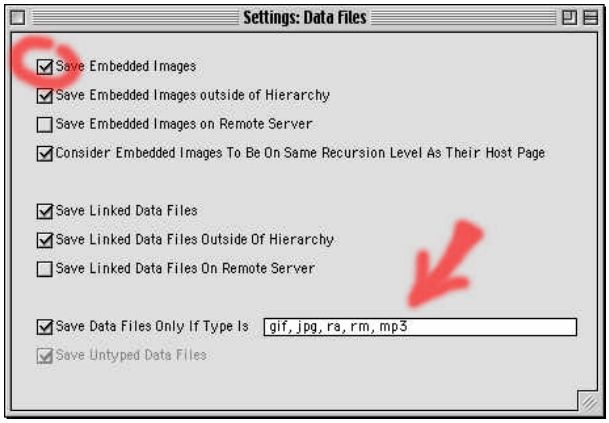
Let's suppose further that you'd like to download only the big pictures, but neither the thumbnails nor the HTML page itself. Moreover, let's assume that you don't want to download other pages that might be linked to this page. You can do all this with PageSucker; there are just a couple of settings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



Uncheck the option "Save Parsed HTML Pages" to tell PageSucker not to save the HTML code of the page.

Close the *HTML Files* window (though you can also leave it open) and select "Data Files" from the "Settings" menu. This will open the *Data Files* window:

Here, you need to uncheck the option called "Save Embedded Images". 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). Finally, edit the list of data file types to download (marked with an arrow in the screenshot). This list should contain the types "gif" and "jpg" only, so that no other file types are downloaded.



You may now close the *Data Files* window, or just switch the main window back to the front. Here you'll have to enter the URL of the picture catalog page:

Moreover, you'll have to enter a "1" in the "Max. Recursion Depth" field. This prevents PageSucker from jumping to other linked pages by specifying that no more than one link may be followed in a row. It is necessary to allow one link to be followed, as the pictures to be downloaded are also linked to the page.

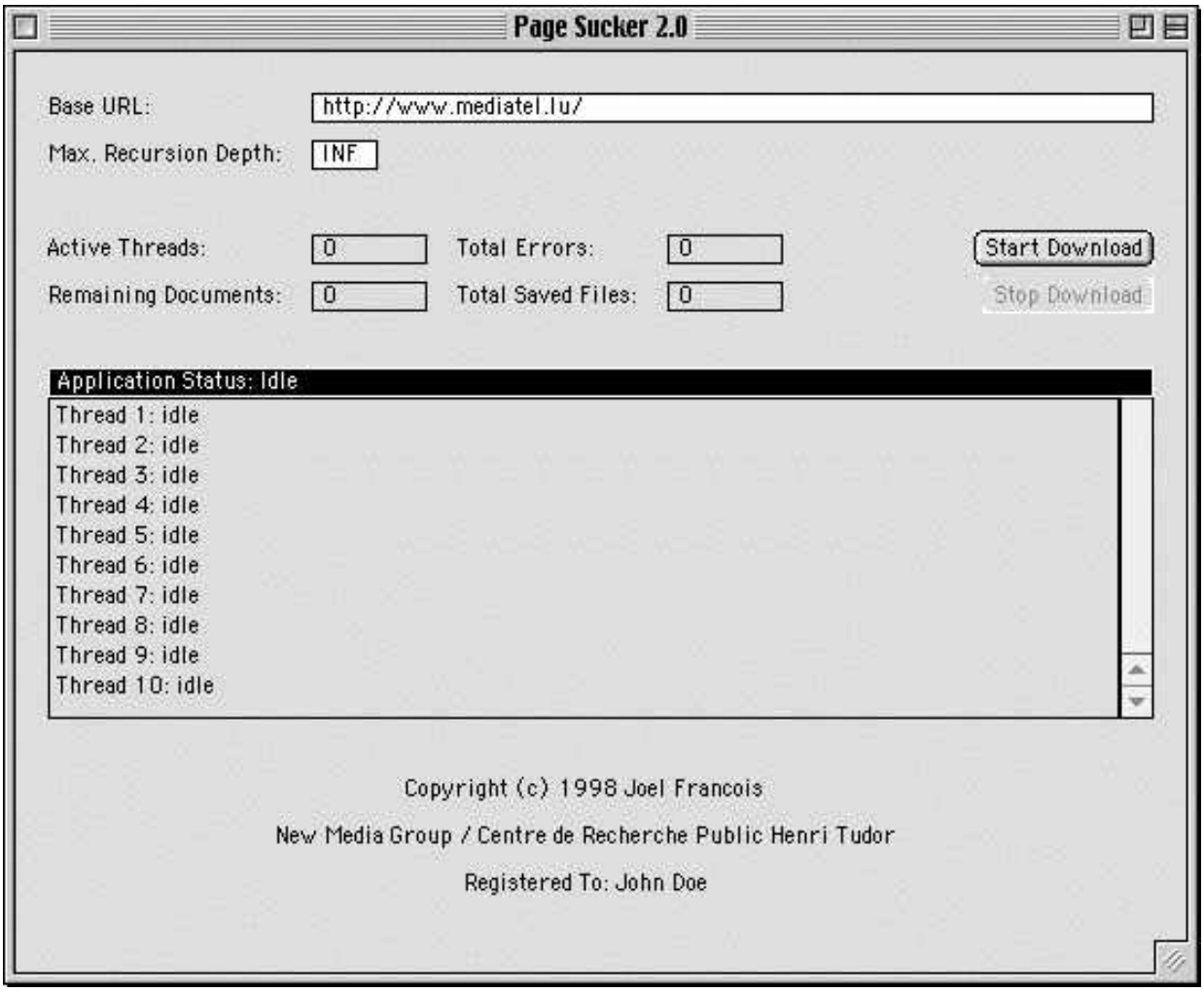
Once all these settings are set, you may press the "Start Download" button to start the download. When it is complete, you'll find a mirror of the server directory structure on your local disk, but only the big versions of all linked JPG and GIF pictures will be present. There will be no HTML files or 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 options set absolutely correctly.

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.

5. THE MAIN CONTROLS

When PageSucker is launched, the main window appears (the picture shows the window as it looks on a Macintosh running system 8.5 and MRJ 2.0; on a Windows machine 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 application and thread status views.

- **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 has been reached. This URL also fixes the hierarchy to be scanned (see section 6 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. Recursion Depth:**	Sets the maximum recursion depth. See section 6 below for more details on the recursion depth. The default value for this field is the special value INF, which specifies an infinite maximum recursion depth.
♦ **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.
♦ **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 for the application.
♦ **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 for the application. 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 console window[1] and is inscribed in the log file.
♦ **Total Saved Files:**	The number of files actually saved to the local disk during the last download process. Not all downloaded objects are saved; this can be controlled via various filters (see section 7 below for more details).
♦ **Start / Stop Download:**	The *Start Download* button starts a download process. Normally it stops automatically when complete. To abort the process before it is complete, the *Stop Download* button can be clicked. PageSucker will then try to cancel all running threads to stop the download. The *Start* button will become available again only after the download currently in progress has terminated or has been successfully canceled. 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

[1] The console window is a separate text window managed by the Java interpreter. Depending on the interpreter used, it is visible as soon as the application is started, or appears only after a message has been written into it. On Windows machines using JRE, a DOS (shell) window is used as console.

download process and must be quitted and restarted in order to be used again.

After clicking the *Start Download* 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.

Below these controls in the main window can be seen a status line showing the application's overall status and the list of threads (and what they're currently doing). Here's a summary of the various thread status displays:

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 currently 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.
Analyzing *URL* *(shown in blue)*	The thread is currently analyzing an HTML page to find embedded links, or it is just about to start the download of a data file.
Downloading *URL* *(shown in red)*	The thread is currently downloading a data file. A percentage enclosed in parentheses shows the progress of the download.

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

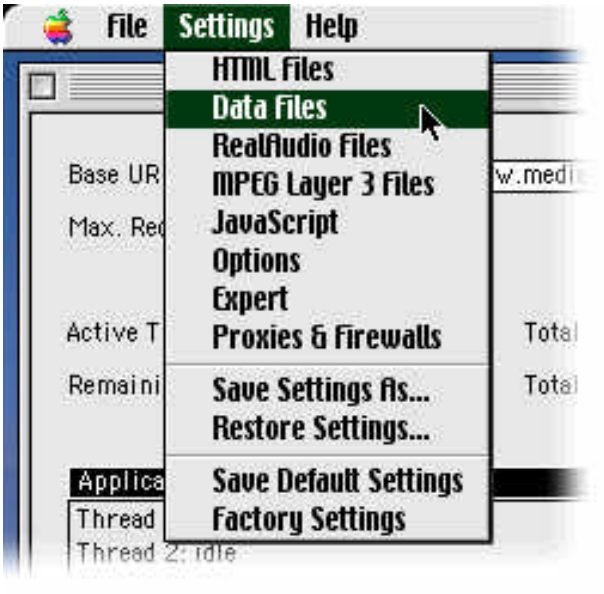
A download process stops **if no more files liable for download can be found**. Two global rules determine if a file should be downloaded:

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 recursion depth less than or equal to the maximum recursion depth will be analyzed. Example: When 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.

7. THE SETTINGS MENU

PageSucker allows very detailed control on which files should be downloaded and which 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[2]:



Here’s a quick overview:

The top items from 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”, “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.

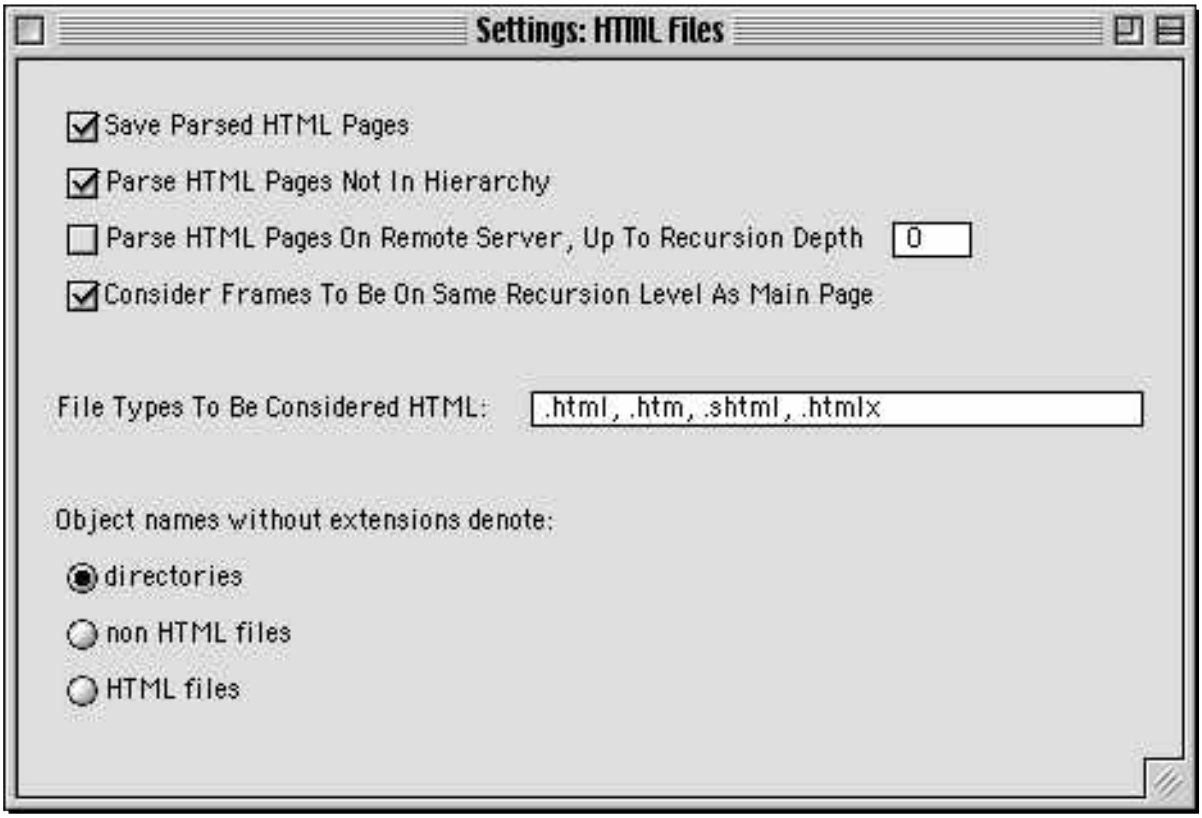
Finally, “Factory Settings” restores the default settings for the entire application.

Please note that you can have multiple settings windows open at any one time. Changed settings will be effective immediately, except if a download is currently in progress. In that case, the new settings will be used for the following download only.

7.1 The HTML Files Window

HTML file filters handle all 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:

[2] 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 8.5.



- **Save Parsed HTML Pages**

  Detected HTML pages are always downloaded and analyzed. However, they are only saved to the local disk if the *Save Parsed HTML Pages* option is checked.

- **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 above). 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. Consider the base URL

  `"http://www.mediatel.lu/crmm/art_mm/h_art_mm.html"`

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

  `"http://www.mediatel.lu/crmm/art_mm/"`

  If the *Parse HTML Pages Not In Hierarchy* option is enabled however, PageSucker isn't concerned about the hierarchy: all HTML pages 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. Example: let's assume the base URL is

```
"http://www.mediatel.lu/crmm/art_mm/h_art_mm.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.mediatel.lu
```

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: as most pages on the Internet are linked to several remote pages, PageSucker would jump to these 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 on remote servers to a fixed recursion depth[3]. It's the number in the text field you can see in the window 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 these 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 harddisk, would you?

- **Consider Frames To Be On Same Recursion Level As Main Page**

Wow! That's one complicated name for an option. To understand what it means, you must know that HTML frames (independantly 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 6 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 techno-babble; just leave this option checked (it is checked by default).

[3] For an explanation of the term "recursion depth", see section 6.

♦ **File Types To Be Considered HTML**

PageSucker tries to identify the nature of a file by looking at its filename extension. This way, for example, it knows that the URL "http://www.example.com/file.gif" refers to a GIF picture. In the same way, PageSucker tries to identify which files contain HTML code (i.e. which files may contain links and must thus be analyzed in a more detailed way). This is what this filter does: all files ending in one of the specified extensions are considered to contain HTML code. You can enter one or more filename extensions, separated by commas. The leading dot (".") is optional.

By default, this field already contains the most common HTML filename extensions. You'll need to add an extension only if you encounter a server using uncommon filenames.

♦ **Object Names Without Extensions Denote**

When reading the discussion about file types and filename extensions 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 in a directory with the URL's filename. The name of the file is defined on the server side. 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".

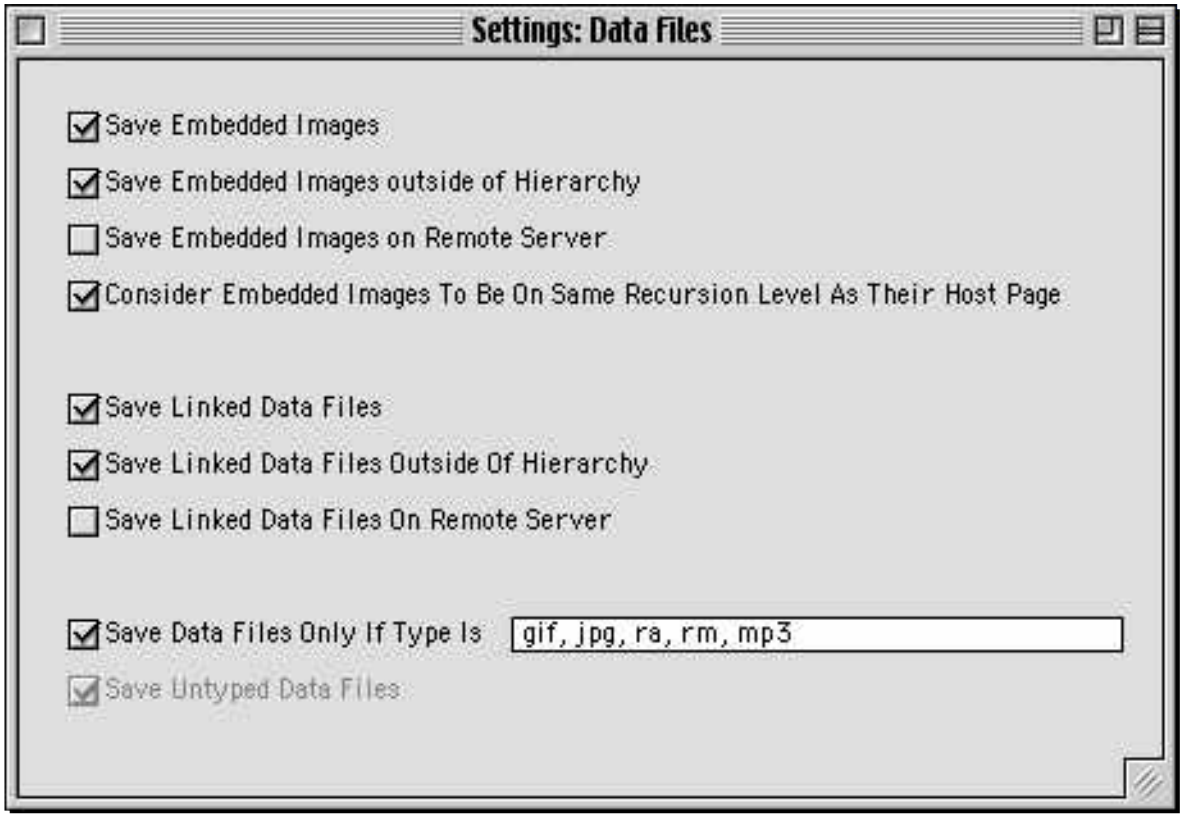
Sometimes however, this is not true. Some servers use filenames with no extension at all. Thus, the above example could also refer to the file "text" on the server "www.example.com". As PageSucker is currently not able to find out if a name without an extension denotes a file or a directory, you must 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 "**HTML Files**" if you know for sure that the server you are downloading uses files without filename extensions that contain HTML code.
- Choose "**Non HTML 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.

Please note that there is currently no way to solve the problem that arises when a given server contains both files that have filenames without extensions and URLs that denote directories. You will be able to partially download such servers, but you will most probably get incorrect results for certain files or directories. This issue will be addressed in a future release of PageSucker.

7.2 The Data Files Window

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



Whereas HTML file filters 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 Images**	Pictures embedded in Web pages will be ignored unless this option is enabled. Technically speaking, this includes all images referenced via the <IMG> HTML tag. Images 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 Images* option is checked.
♦ **Save Embedded Images Outside Of Hierarchy**	Embedded images 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. You should generally leave this option checked (it is checked by default), as it is common practice to store pictures in a hierarchy separate from the HTML files.
♦ **Save Embedded Images On Remote Server**	Embedded images that are located on a different host server than the one specified by the base URL will be ignored unless this

option is checked.

- **Consider Embedded Images To Be On Same Recursion Level As Their Host Page**

  As explained in section 6 above, the recursion depth is incremented each time a URL found inside a page is analyzed. Given that an embedded image is a separate file and is also referenced via its 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 not incremented for embedded images.

  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 *Consider Embedded Images To Be On Same Recursion Level Than Their Host Page* option is unchecked, none of the images contained in the base page will be downloaded. With the option checked, the images will be saved, although all other links are ignored.

  Normally you should leave this option checked.

- **Save Linked Data Files**

  This filter concerns all files linked to a page (via an <A> HTML tag) that are not HTML files. If this option is unchecked, all of these files will be ignored. Embedded images are not affected by this filter.

  Moreover, the following two options are available only if this option is on.

- **Save Linked Data Files Outside Of Hierarchy**

  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.

  HTML files or embedded images are not affected by this filter.

- **Save Linked Data Files On Remote Server**

  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.

  HTML files or embedded images are not affected by this filter.

- **Save Data Files Only If Type Is**

  If this option is checked and one or more file name extensions are entered in the text field, only files of these types will be saved. Several extensions can be specified, provided they are separated by commas. The leading dot (".") is optional.

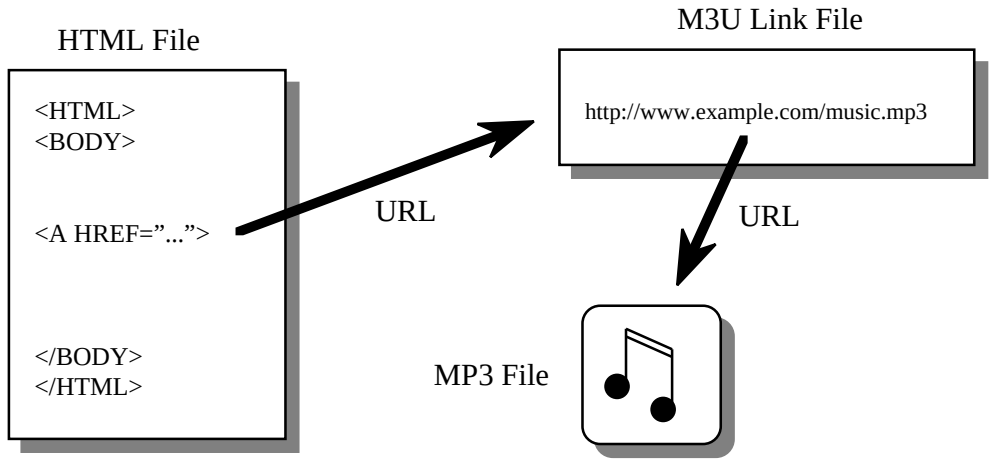
  This option only concerns linked data files and embedded images, i.e. it has no effect on HTML files, which are controlled by the filters in the *HTML Files* window (see section 7.1 above).

  For example, if you want to download data files only if they are JPEG pictures, enter only the extension “jpg” in the text field.

♦ **Save Untyped Data Files**	This option is enabled if and only if filenames without extensions are set to denote non HTML files via the controls in the HTML Files window (see section 7.1 above). In that case, the *Save Untyped Data Files* filter specifies if such files are to be saved to the local disk or not.

7.3 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 files (".RA" or ".RM" files) and MPEG Layer 3 files (".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".

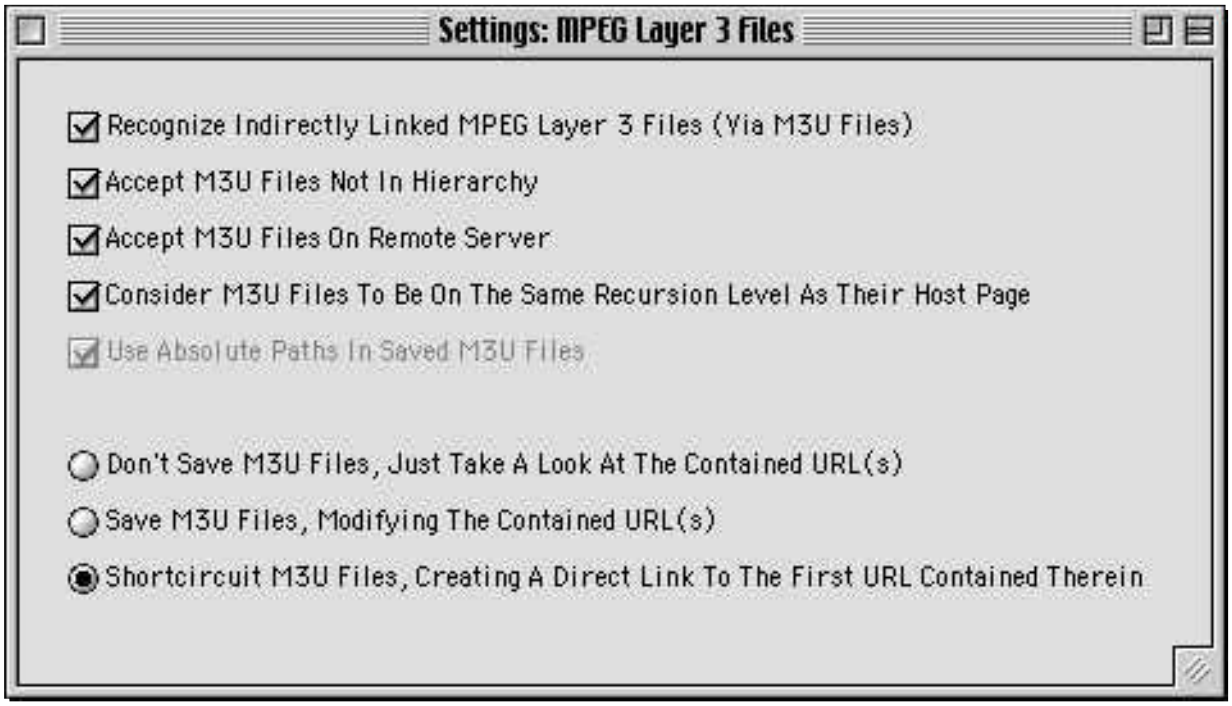


In both cases the reason for using an intermediate file is that this way a specialized application (not the Web browser) can be used to download the data files and play them at the same time (the playback is said to be "streaming"). PageSucker will of course need to handle this kind of link files in 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 the special "pnm://" protocol 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://". 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 Data Files window (see section 7.2 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 these 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 Files window discussion applies to it as well.



♦ **Recognize Indirectly Linked MPEG Layer 3 Files (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 Data Files window description in section 7.2 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 On The Same Recursion Level As Their Host Page**	As explained in section 6 above, the recursion depth is incremented each time a URL found inside a page is analyzed. Given that an M3U file is a separate file referenced via its 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 not incremented for M3U files.

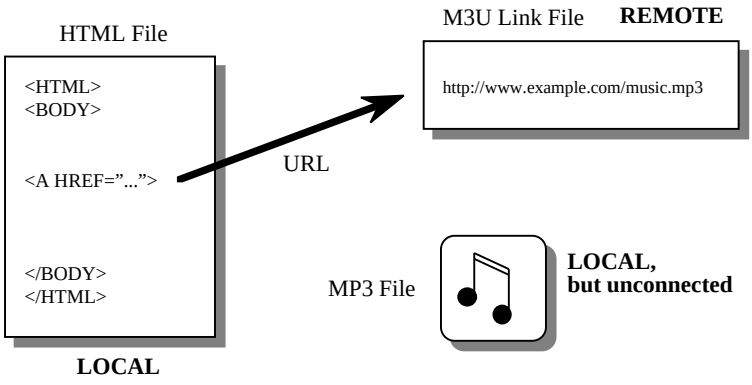
This is the same treatment that embedded images get when the *Consider Embedded Images To Be On Same Recursion Level Than Their Host Page* option in the Data Files window is used.

Normally you should leave this option checked.

♦ **Use Absolute Paths In Saved M3U Files**

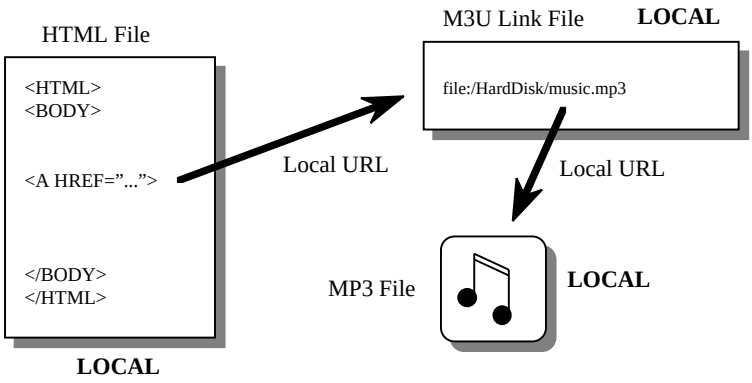
To explain this option, we must first take a look at the three possible ways to handle an M3U file. Which treatment is used is specified with the radio buttons at the bottom of the window:

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.

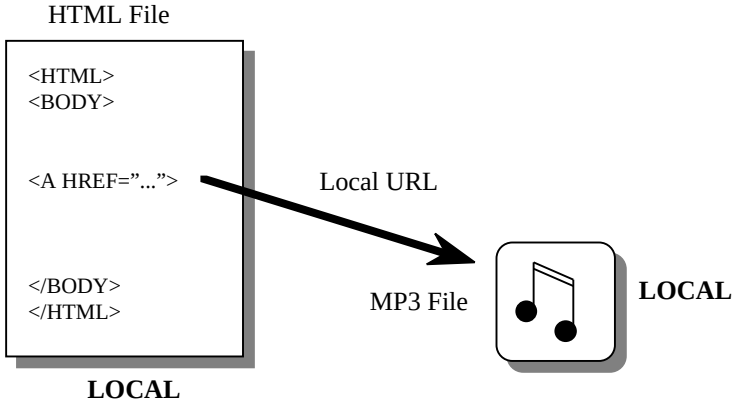


2. ***Save M3U Files, Modifying The Contained URL(s)***. This saves the M3U locally after extracting the contained URLs. The M3U file is then updated to reflect the new local path of the downloaded MP3 files. This is where the *Use Absolute Paths In Saved M3U Files* option is used. If it is checked, 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. ***Shortcircuit M3U Files, Creating A Direct Link To The First URL Contained Therein.*** 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 possibly 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.



7.4 The JavaScript Window

This window, accessible via the "JavaScript" item in the "Settings" menu, contains some options that enable PageSucker to detect certain types of links in HTML documents that would otherwise not be found. This includes URLs contained inside JavaScript routines and single JavaScript statements embedded in the HTML code, and 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 artficial 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 a URL, then it tries to open a connection to that URL. If the connection fails, PageSucker concludes that this was not really a URL, so it forgets about it. If the connection is successful however, it adds the URL to its internal list of URLs 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";
}
```

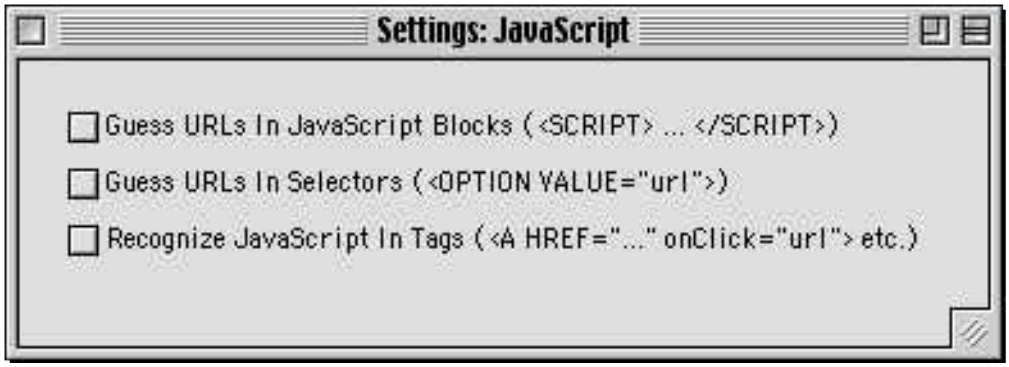
Both routines produce exactly the same URLs. Two different URLs are generated according to the "variant" parameter:

```
      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 thus consider it (incorrectly again) for downloading. This will not only cause unneeded downloads, but might 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. Below, you'll find a screenshot of the JavaScript 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 JavaScript 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**

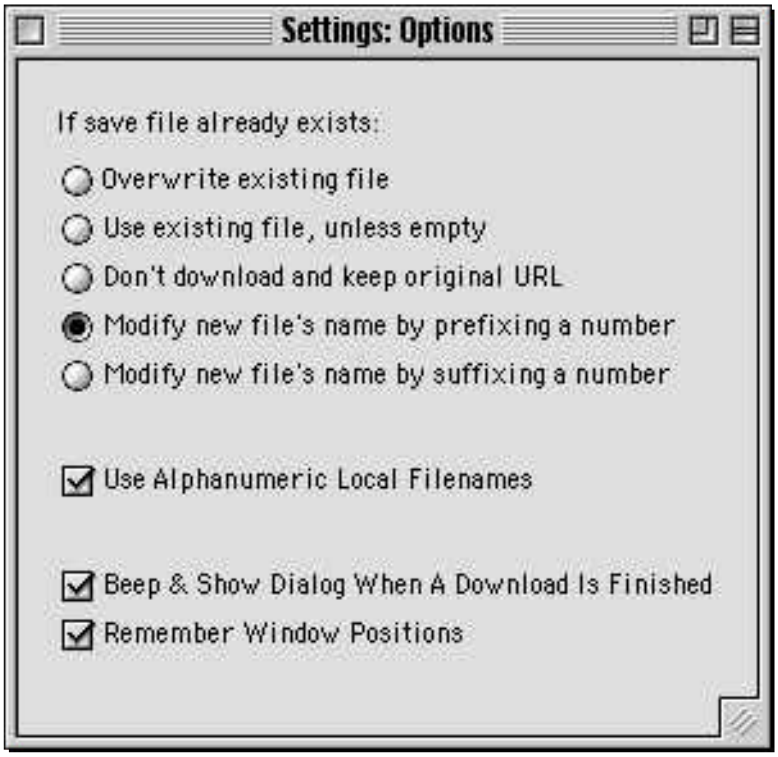
  If you'd like PageSucker to recognize single lines of JavaScript embedded in various HTML tags, you'll need to check this option. For example the HTML code

  ```
  <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.

7.5 The Options Window

Select "Options" from the "Settings" menu to show the options window:



♦ **If Save File 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.

- ***Don't download and keep original URL*** – the remote file is not downloaded and the local copy of the referring HTML page is not updated; it will contain the original remote URL.

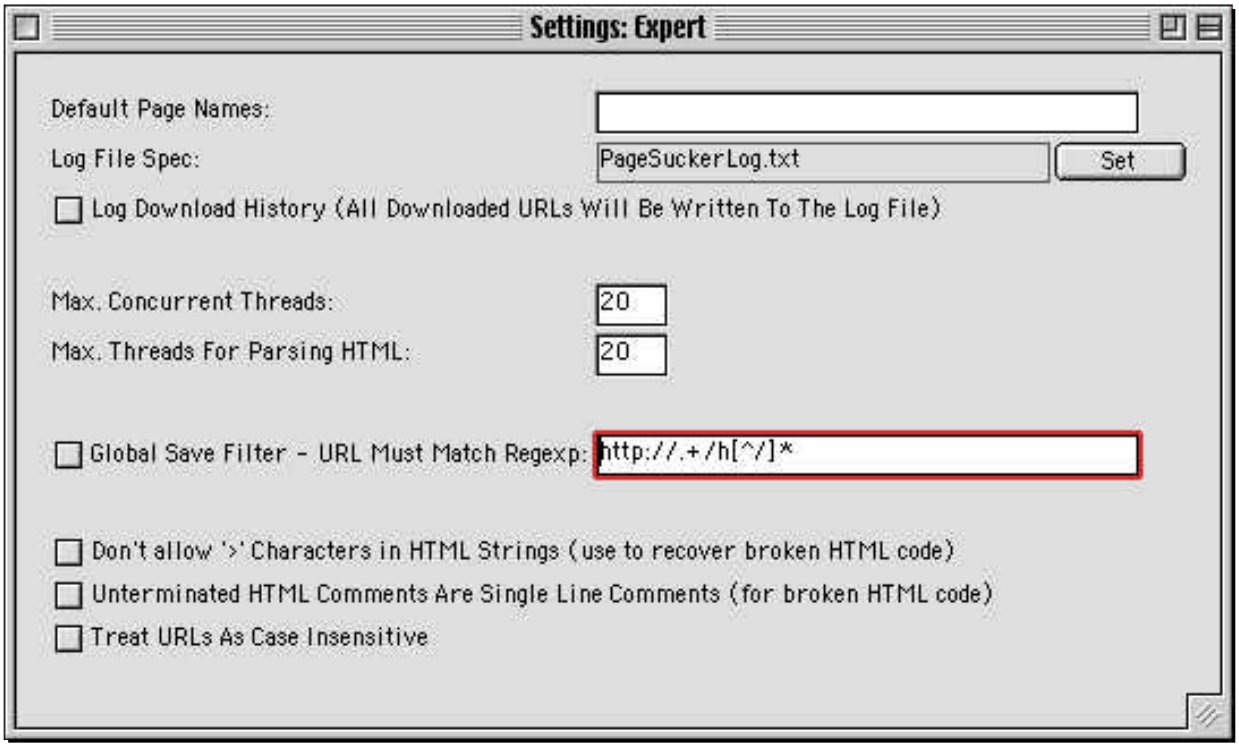
- ***Modify new file's name by prefixing a number*** – the remote file is downloaded under a different name, which is constructed by adding a number and the '~' character to the beginning of the original file name (e.g. "picture.gif" becomes "0~picture.gif").

- ***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*.
♦ **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, these are 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 checked. Normally you should leave it that way unless you have a particular reason to uncheck it.
♦ **Remember Window Positions**	This option is checked by default. It causes PageSucker to keep track of the size and position of all PageSucker windows on your screen. 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 settings 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 Positions" option. Don't forget to save your default settings in that case, so they are restored when PageSucker is relaunched!
♦ **Beep And 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.

7.6 The Expert Window

This window contains a collection of options that are normally only useful in certain specialized cases. Select "Expert" from the "Settings" menu to open the expert settings window:


Settings: Expert
Default Page Names:
Log File Spec:
PageSuckerLog.txt
Set
Log Download History (All Downloaded URLs Will Be Written To The Log File)
Max. Concurrent Threads:
20
Max. Threads For Parsing HTML:
20
Global Save Filter - URL Must Match Regexp:
http://.+/h[^/]*
Don't allow '>' Characters in HTML Strings (use to recover broken HTML code)
Unterminated HTML Comments Are Single Line Comments (for broken HTML code)
Treat URLs As Case Insensitive

♦ **Default Page Names**

Usually, a URL's file name and/or file path can be omitted (e.g. "http://www.crpht.lu/"), in which case a default file will be automatically used by the server. This file's name is known only to the server and not visible from the client's point of view.

When downloading a page specified by such an incomplete URL, PageSucker can attempt to guess the file name by trying out each of the names given in the *Default Page Names* field. Several names can be entered, provided they are separated by commas. All default file names must have an extension that is recognized by PageSucker as identifying an HTML file. The list of filename extensions that designate an HTML file can be defined by the user in the *HTML Files* window (see section 7.1 above). For example, if you have defined in the *HTML Files* window that only files the name of which ends with ".html" or ".htm" are to be considered HTML files, your default page names must end with ".html" or ".htm".

If none of the given default names matches the real file name, the page can still be downloaded, but will be saved locally under a different file name. This may in certain situations cause the page to be downloaded twice, but isn't harmful otherwise.

Please note also that with the current implementation certain servers may block when PageSucker tries to determine their default file name. A direct work-around for this problem is to simply leave the *Default Page Names* field blank. PageSucker may also execute a little faster if no default names are given.

Normally it is not crucial to enter any default file names.

♦ **Log File Spec**

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 console 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 12 below) because of a problem related to downloading a certain site, it would be helpful if you sent us the log file related to the problem download. This avoids that you need to tell us exactly what settings you were using.

♦ **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 downloaded 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 looked at but PageSucker has decided that it didn't need to save the file locally
- ***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.

♦ **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.

♦ **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 and greater than zero.

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

- **Global Save Filter: URL Must Match RegExp**

If this option is checked and a regular expression string is entered in the text field, all detected URLs are first matched against the given regular expression. The file specified by the URL will only be downloaded and saved if this match succeeds. This concerns HTML files as well as linked data files and embedded images.

More information on regular expressions can be found in section 10 below. If you don't know what a regular expression is, you probably shouldn't touch this option.

- **Don’t allow ‘>‘ Characters in HTML Strings**

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 *Don’t allow ‘>‘ Characters in HTML 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. You should thus only activate that option when the page(s) to be downloaded are known to contain this kind of error.

- **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**

In theory URLs are considered to be case sensitive, i.e. the following two URLs don't denote the same file:

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

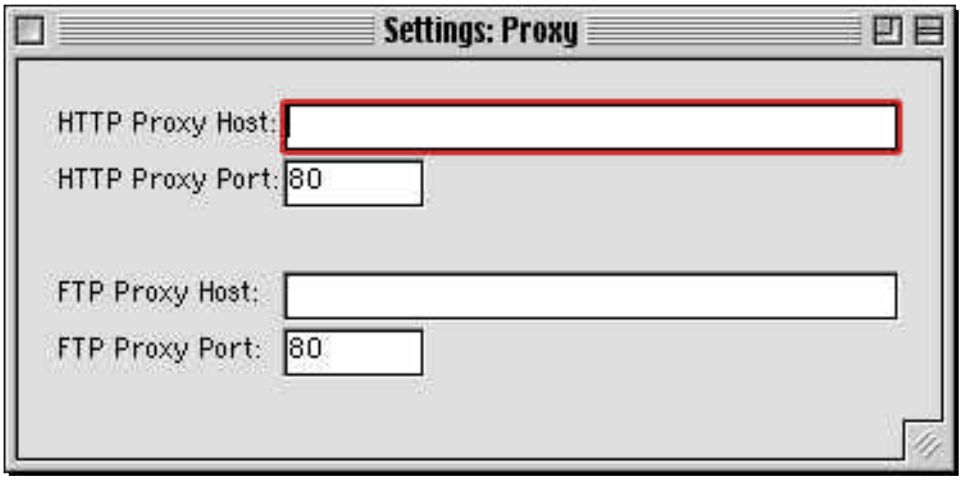
In practice however, both URLs point to the same file if the server for the "www.example.com" site is running an operating system which has case insensitive filenames, like Windows NT or the MacOS. If the server is running UNIX however, the two URLs are not equivalent.

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.

7.7 The Proxy Window

You can access the *Proxy Settings* window by choosing "Proxies & Firewalls" from the "Settings" menu. It lets you enter settings to enable PageSucker to use a proxy server. These are the same settings that you have to enter in your standard Web browser's preferences dialog. You won't need these settings unless you're using a proxy server to access the Internet.

If you don't know what a proxy is, chances are that you're not using one. If in doubt, ask your network administrator or Internet service provider, or just try out PageSucker without entering a proxy host name. If it works OK, you don't need to specify a proxy host.



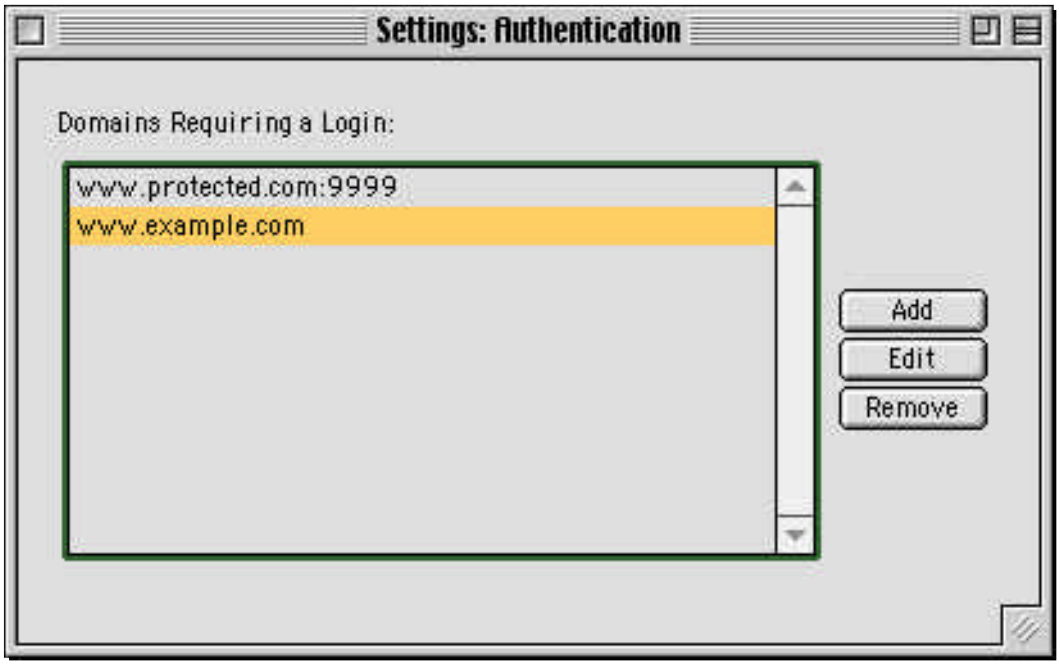
♦ **HTTP Proxy Host**

Enter the name of your HTTP proxy host here, e.g. "proxy.example.com". Don't 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.

♦ **HTTP Proxy Port**	Enter the port number of your HTTP proxy server here. This field is ignored if you didn't enter a proxy server name.
♦ **FTP Proxy Host**	Enter the name of your FTP proxy host here, e.g. "proxy.example.com". Don't 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 **F**ile **T**ransfer **P**rotocol.
♦ **FTP Proxy Port**	Enter the port number of your FTP proxy server here. This field is ignored if you didn't enter a proxy server name.

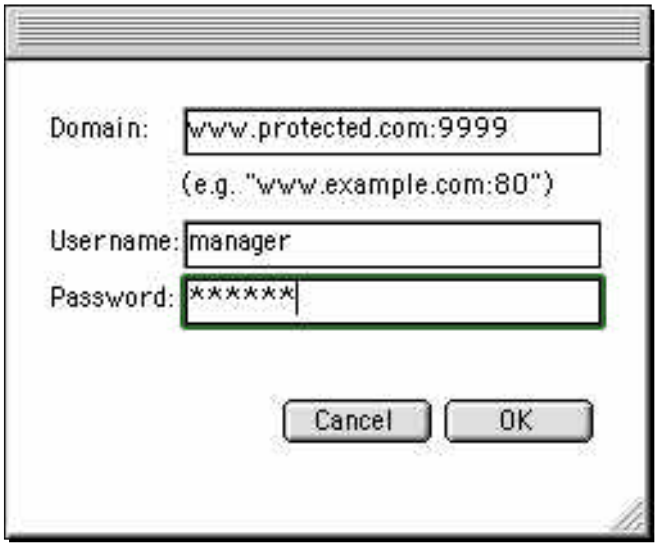
7.8 The Authentication Window

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. 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:



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 username 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 an authentication! 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.

7.9 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).

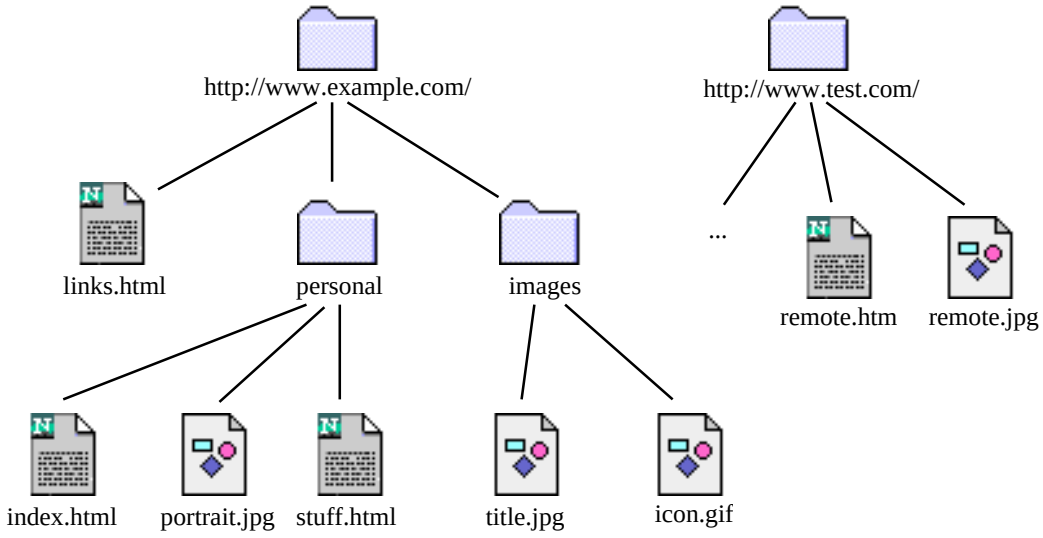
To prevent this, PageSucker allows you to control which settings it uses the next time you start it up. Just select "Save Default Settings" from the "Settings" menu to remember the current settings as defaults. The next time PageSucker is launched, these settings will be used automatically. To get back 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[4] 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.

[4] 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 **: -)**

8. 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 enter the following values as PageSucker's settings:

Base URL ***(in the main window)***	http://www.example.com/personal/
Max. Recursion Depth ***(in the main window)***	1
Default Page Names ***(in the Expert Settings window)***	home.html, index.html, home.htm, index.htm

As the given base URL doesn't have a file name, PageSucker checks all default page names to see if it can find an existing file. It first attempts to open the URL

`"http://www.example.com/personal/home.html"`

which isn't valid (as no such file exists). PageSucker then tries out the URL

`"http://www.example.com/personal/index.html"`

which can indeed be opened. That file is thus downloaded and analyzed. The file "index.html" being an HTML file, it is saved to the local disk only if the *Save Parsed HTML Pages* option is enabled in the *HTML Files* window.

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

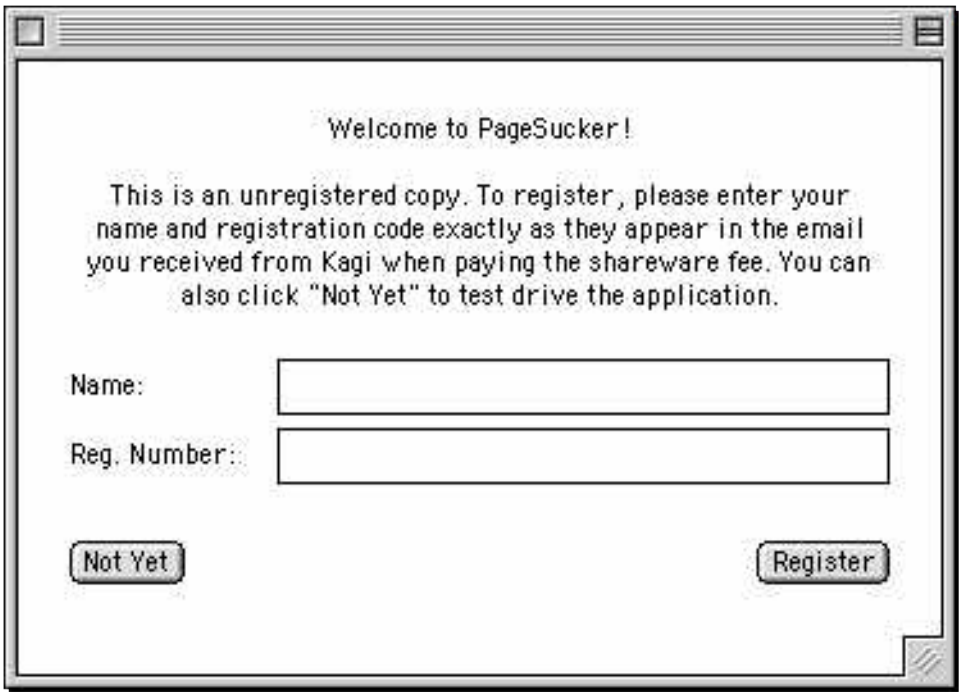
File	Kind	Discussion
stuff.html	linked HTML file	Downloaded and analyzed. Saved only if "**Save Parsed HTML Pages**" is enabled in the *HTML Files* window. Any links contained in this file will be ignored, as the maximum recursion depth of 1 is reached. Embedded images, if any, will not be downloaded unless "**Consider Embedded Images To Be On Same Recursion Level As Their Host Page**" is checked in the *Data Files* window.
links.html	linked HTML file	Not downloaded unless "**Parse HTML Pages Not In Hierarchy**" is checked in the *HTML Files* 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 "**Save Embedded Images**" is checked in the *Data Files* window.
title.jpg	linked data file	Downloaded and saved if both the "**Save Linked Data Files**" and "**Save Linked Data Files Outside Of Hierarchy**" options are checked in the *Data Files* window, as the file is not contained in the hierarchy starting at the directory "personal".
icon.gif	embedded image	Downloaded and saved if both the "**Save Embedded Images**" and "**Save Embedded Images Outside Of Hierarchy**" options are checked in the *Data Files* window.
remote.jpg	embedded image	On remote server, thus only saved if both the "**Save Embedded Images**" and "**Save Embedded Images On Remote Server**" options are checked in the *Data Files* window.
remote.html	linked HTML file	On remote server, thus not downloaded unless the **Parse HTML Pages On Remote Server** option is checked in the *HTML Files* window.

9. THE FILE MENU

The File pull down menu contains just three items[5]. They allow you to register and unlock the application after paying the shareware fee, check if a new version has been released, and quit. While there's no big discussion needed to discuss the functionality of the "Quit" command, the other two items are detailed in the following sections.

9.1 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 PageSucker'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 one month in demo mode without paying anything. However, if you find it useful and if you'd like to keep using it after that period, you should pay the modest fee of **USD 10.-** (*single user license*).

[5] In the Mac version, there are now actually four items: the new "Close" command can be used to close the currently active window.

You can also purchase a *site license* for **USD 300.-** (equal to 30 users). It covers all locations for your organization within a 160 kilometer radius of your site (100 miles). 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. A *world-wide license* costs **USD 1500.-** and it covers all locations for your organization on the planet earth.

Such as to make paying as easy as possible, we are using the Kagi payment service. This means that you can pay cash (in a variety of currencies), with a check or a credit card. On Macintosh or PC you can use the included “Register” application. See the separate “Read Me” file for detailed instructions on how to use it. You can also register online by using the Web page at

```
http://order.kagi.com/?2YQ
```

The good part is that once you have purchased a license, you'll get a registration code from Kagi via email. You will have to enter the name and code from that email 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?

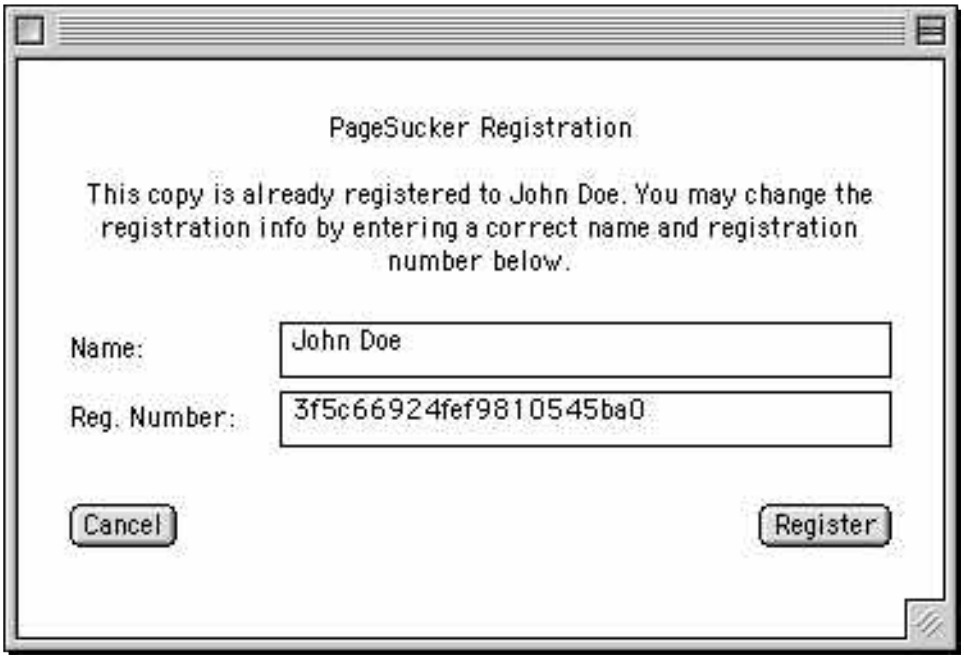
Note: Already registered customers who have bought an earlier release of PageSucker (which didn't need a registration code) should already have gotten their registration code via email. If you are such a customer and you haven't received a code yet, please contact us. You'll find contact information in section 12 below.

Here's an excerpt from a sample[6] email such as the one you'll get from Kagi after paying:

```
Name: John Doe
Reg. Number: 3f5c66924fef9810545ba0
```

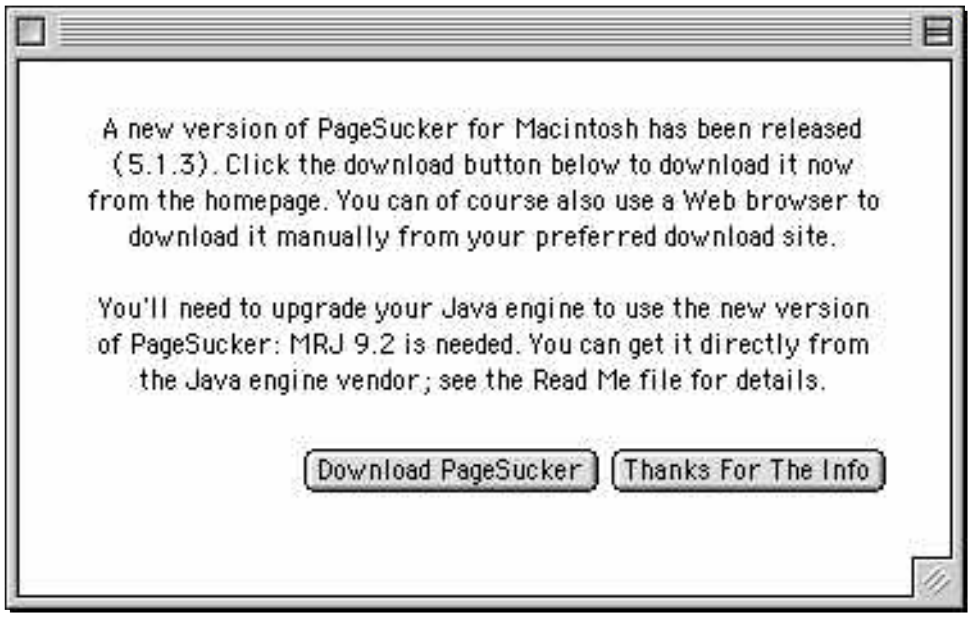
Should you ever want to review or change your registration information once you have entered it, you can do so by selecting "Register..." from the "File" menu. This command also allows you to unlock a demo version without the need to restart the application. You'll see the following dialog, very similar to the one shown above:

[6] This really is only a sample, so don't even think of trying the code shown here – it won't work **;-)**



9.2 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 on its home page 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. Here's an example dialog you might see:



This informs you that the newest release for your platform (Macintosh in this case) is PageSucker version 5.1.3. 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[7]. At this point, you can just

[7] 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)!

close the window, or click "Thanks For The Info" to do nothing further. However, you can also click "Download PageSucker" to automatically get the newest release from the home server. Please note that this will not download a newer Java engine (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 download PageSucker from some other Web site or FTP server 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 our 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 Macintosh and Windows versions respectively):

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

10. SOME REGULAR EXPRESSION EXAMPLES

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 ...). PageSucker uses them with the global matching filter found in the *Expert* window.

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. at:

`“http://www.perl.org/CPAN/doc/manual/html/pod/perlre.html”`

Finally, here are some examples of regular expressions that are of direct use with PageSucker’s URL matching filter:

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

When composing regular expressions for PageSucker, just keep in mind that the entire URL will be matched against your expression, not only the file name or path. Thus you need to include matching strings for the protocol (e.g. “http://”) and server name. Also, remember that all other enabled filters will be applied in addition to the regular expression filter.

11. KNOWN BUGS AND LIMITATIONS

- Downloaded HTML pages will be modified by PageSucker to point to the local copies of other objects downloaded in the same process. If however a page contains URLs of objects 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 stall or show "undefined picture" icons. 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 one described in section 7.6) 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 ReadMe file to avoid most problems.

- 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 they might contain 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 *Options* window. To avoid this kind of problem, make sure to get unique file names by using one of these two options.

- Java applets and embedded data usually interpreted by browser plugins are currently not recognized by PageSucker. Support for the <EMBED> tag will be added in a future release.

- As mentioned above, certain servers may block the connection when PageSucker tries to determine the server's default page name. It is currently not clear if this is a Java interpreter bug or if the problem lies within the standard Java libraries. The work-around for the moment is to leave PageSucker's default page names field blank.

- Connections to a server that block for one reason or another can currently not be interrupted, nor can a time-out be directly defined by the Java application. Depending on the 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.

- 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. Hopefully PageSucker will be optimized for that kind of problem in a future release. For now, try to avoid downloading more than about 2000 files in one pass.

- PageSucker currently recognizes the type of a file to download by the extension of its name. This involves that files having wrong extensions (e.g. a picture file with the name "pic.htm")

cannot be downloaded correctly. Moreover, the filename extension method doesn't allow to distinguish a directory from a data file with no file name extension. A URL such as:

```
“http://www.example.com/picture”
```

might denote a file named “picture” at the server’s root directory level, or the file “index.html” in the directory “picture”. A better algorithm allowing to solve these problems will be implemented in a future release of PageSucker.

- 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 the file hierarchy cannot be reconstructed locally, so the downloaded pages can’t be viewed correctly with a browser. There doesn’t seem to be a solution to this problem, as the CGI programs themselves cannot be downloaded.

- Certain Java interpreters seem to have stability problems when memory becomes tight and might crash the system in low memory situations. There is nothing the Java program can do about this. As a work-around, avoid low memory situations by either increasing the Java interpreter’s memory partition (if possible) or specifying a lower number of threads in PageSucker’s *Expert Settings* window.

- The current PageSucker version only recognizes Macintosh and Windows 95/98/NT platforms correctly. There is also a limited support for the OS/2 platform. All other systems (e.g. UNIX platforms) are assumed to have very rigid conventions when it comes to naming files. To be on the safe side, PageSucker thus attributes MSDOS style filenames to all unknown filesystems. Support for more systems will probably be added in some future release.

- On the Macintosh, PageSucker currently doesn't attribute the correct file types and creators to the files it downloads. This has the effect that those files don't get the expected icon and won't launch correctly when double-clicked. They can still be opened from within the appropriate applications, or by drag-and-dropping their icon onto the application's icon in the Finder. Support for Macintosh file types/creators will be added to PageSucker in a future release.

- When trying to interpret JavaScript programs, PageSucker currently doesn't correctly handle escaped single and double quotes (\' and \"). This may cause certain problems with pages containing escaped quotes in their JavaScript code. This is a known bug that will be corrected in an upcoming release.

12. CONTACT INFORMATION

Here's how to contact us for comments, suggestions and bug reports:

Email: **support.crmm@crpht.lu**
support.crmm@kagi.com

Fax: **+352 42 59 91-275**

Snail Mail: **New Media Group**
Centre de Recherche Public Henri Tudor
(att: Joël François)
6, rue Coudenhove-Kalergi
L - 1359 Luxembourg-Kirchberg
Luxembourg / Europe

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

13. THE LEGAL STUFF

This product is distributed “as is”. The author and the publisher make 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 CDROM.

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