Centre de Ressources Multimédia
CRP Henri Tudor
6, rue Coudenhove Kalergi
L-1359 Luxembourg-Kirchberg

Tel : (352) 42 30 13 - 1
Fax : (352) 42 59 91 - 275
Web Site : http://www.mediatel.lu

User Manual

Title :	PageSucker User Manual	Project :	No Project name
Subtitle :	Version 1.0.1	Author :	Joël François
Filename :	PageSucker User Manual	Date :	24-Sep-97
Classification :	☒ General ☐ Confidential ☐ Multimedia Only	Pages :	13

Author	Date	Modification
Joël François	27-Jun-97	Creation
Joël François	24-Sep-97	Update for version 1.0.1

S u m m a r y

The user manual for PageSucker 1.0.1.

Contact the author at “support.crmm@kagi.com” or “support.crmm@crpht.lu” for bug reports, suggestions and comments.

PageSucker 1.0.1 - User Manual

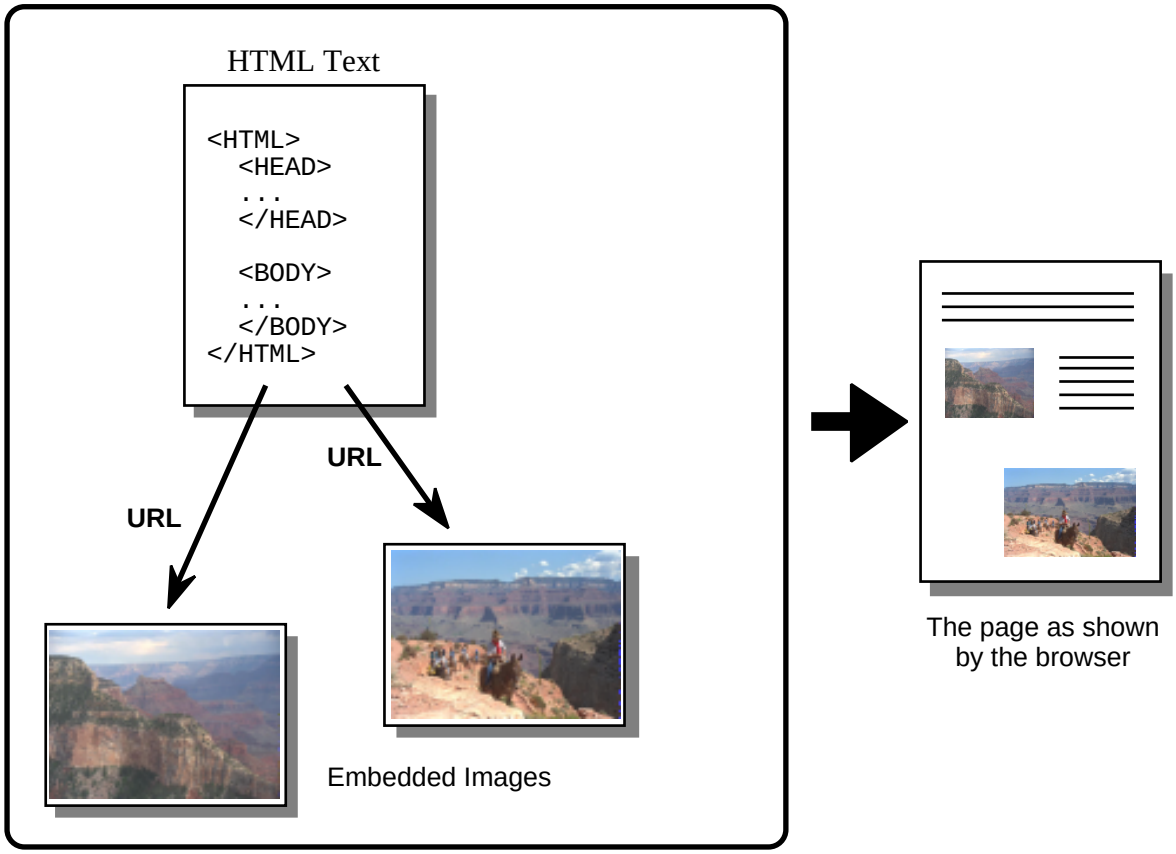
1.1 What Is It? - An Introduction And Some Background Information

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.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 end 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.

1.2 Installation

As PageSucker is a Java Application, it needs a Java interpreter to run.

On Macintosh, Apple's MRJ (MacOS Runtime for Java) can be used for that purpose. It can be downloaded for free from "http://applejava.apple.com/". Simply double click MRJ's installer icon to install it. PageSucker can then be launched by double clicking its icon.

On Windows 95 and NT 4.0, you can use JRE (Java Runtime Environment) from SUN. It is available on the Web at "http://www.javasoft.com/". Follow these simple steps to get up and running:

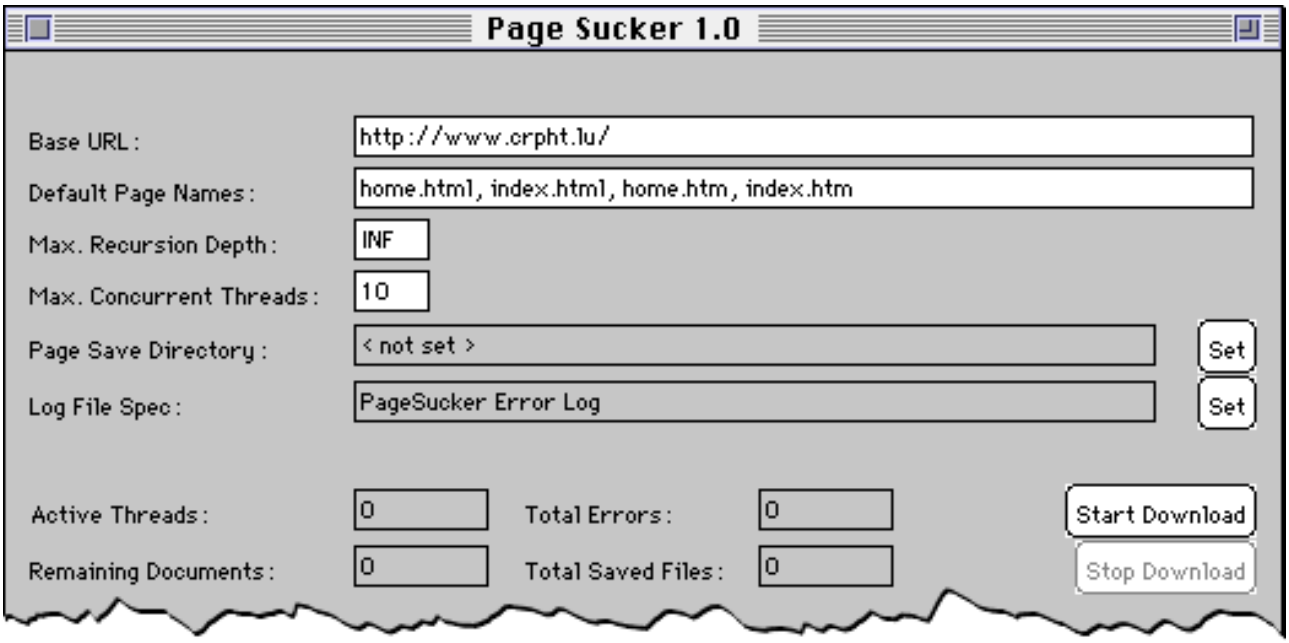
1) Execute the “JRE114.EXE” file to install JRE.
2) Copy the file “PageSuckerClasses.ZIP” to your harddisk. **Do not unzip it!**
3) Edit the provided “PageSucker.BAT” file to reflect the installation path of JRE and the “PageSuckerClasses.ZIP” file on your machine.
4) Execute “PageSucker.BAT” to run PageSucker.

PageSucker has not yet been tested on the various UNIX systems, but if you can find a Java interpreter for the UNIX flavor you’re using, there’s no reason why it shouldn’t run on your platform. (However, version 1.0.1 doesn’t yet recognize a UNIX filesystem as such and thus assumes very severe restrictions in file naming conventions. In fact, the current version attributes MSDOS style filenames to saved files on all unknown platforms.)

If you’re running PageSucker on any platform other than Macintosh get the Windows version, as it contains the “PageSuckerClasses.zip” file, which contains all the compiled program code. This file can then be fed to the Java runtime engine (it need not be unzipped). The Macintosh archive doesn’t contain the “PageSuckerClasses.zip” file, as the program code has been merged into a double-clickable Macintosh application.

1.3 Getting Started - The Main Controls

When PageSucker is launched, the *Control* user interface panel appears, the top section of which is shown here:



- **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 1.4 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”).

♦ **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 by the server and not visible from the client's point of view. When downloading a page specified by such an incomplete URL, PageSucker attempts 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 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. 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.
♦ **Max. Recursion Depth:**	Sets the maximum recursion depth. See section 1.4 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.
♦ **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, some of the open connections may fail or block, and the overall performance will be diminished. 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.
♦ **Page Save 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 page save directory. The directory path may not be entered directly in the *Page Save Directory* field. To set the directory, press the *Set* button next to the field. If no save directory has been set when the download is started, PageSucker will ask for it before analyzing the initial page.
♦ **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 "PageSucker Log" 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.

♦ **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 this 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 1.5 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 download process, and must be quitted and restarted in order to be used again.

Below these controls in the main window can be seen a status line showing the application's overall status and a list of all pages having been downloaded or currently being processed. Each line of this list shows a URL preceded by one of the following words:

Analyzing	the page is currently being analyzed
PARSED	the page has been analyzed, but did not qualify to be saved; other pages referenced from this page may still have been saved
SAVED	the page has been downloaded and saved to the local disk
ABORTED	the page was in the process of being downloaded when a user abort occurred, or the entire download process was aborted due to a fatal error
ERROR	an error occurred which prevented the successful downloading of the page

At the bottom of the window can be seen three buttons that switch between the three user interface panels: *Control*, *Filters* and *Preferences*. The *Control* panel has been described above, *Filters* and *Preferences* are described in the following chapters.

[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.

1.4 When To Stop? - The End Of A Download Process

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

1. PageSucker ignores all files not contained inside the hierarchy defined by the base URL (entered in the *Control* user interface panel – see section 1.3). 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: 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/"`

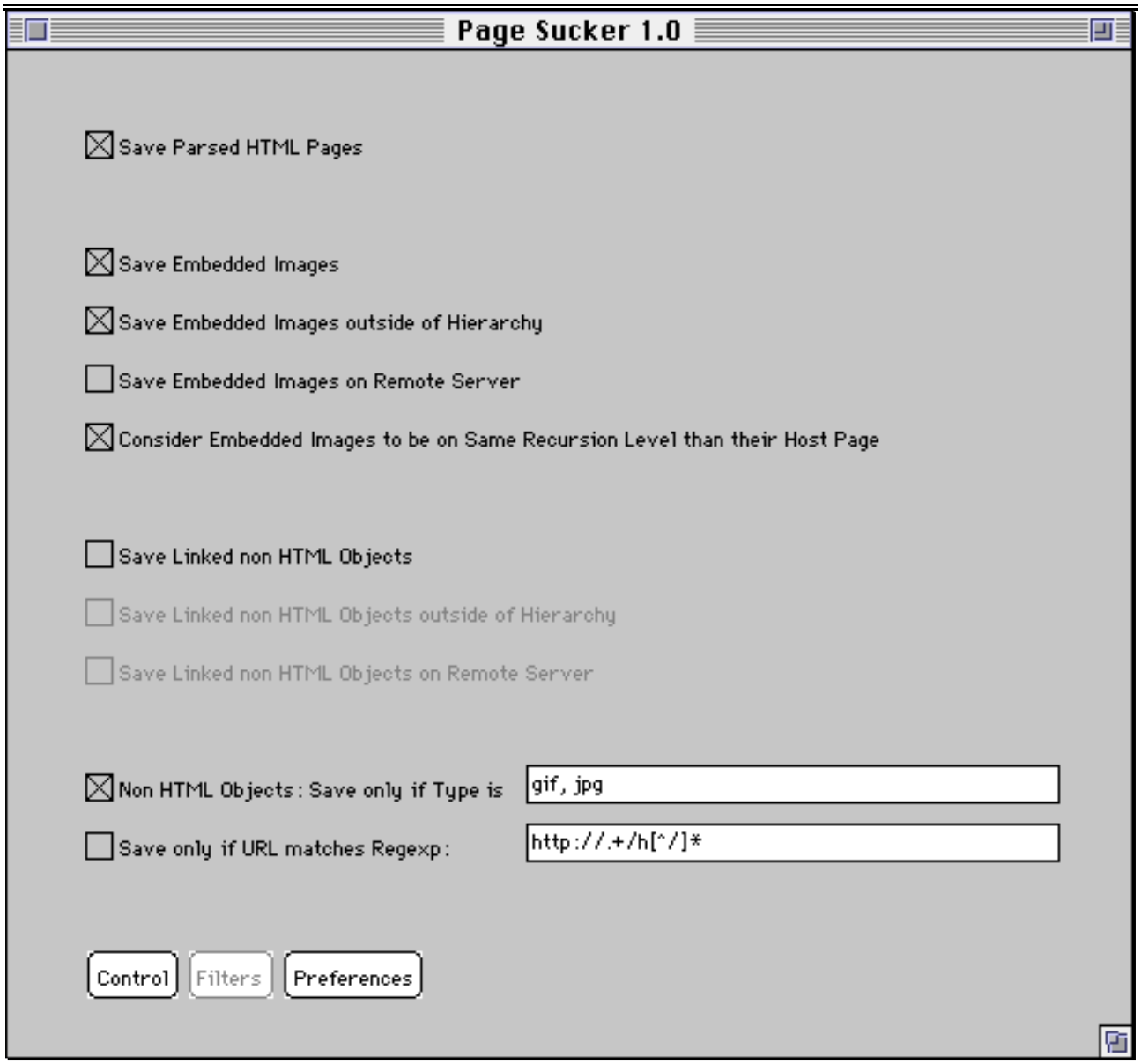
   For embedded images and for linked non HTML objects, this can be overridden with certain filter options (see section 1.5). The rule is however always respected for HTML text files.

2. The Download stops when the maximum recursion depth is reached. 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) 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, PageSucker stops the download process when the recursion depth reaches 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, only the initial page will be downloaded, whereas a value of one will download the initial page plus all pages linked from that page.

3. 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 section 1.5.

1.5 Some Words On Filters

When the *Filters* button at the bottom of the main window is clicked, the user interface switches to the "Filters" panel. (You can switch to one of the other user interface panels using the *Control* or *Preferences* button at the bottom of the window.)



PageSucker's filters are a mechanism to control which of the detected files are actually saved to the local disk. You could for example tell PageSucker to only download HTML text files, thereby omitting all embedded images. The available options are detailed below:

- **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.

- **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.

- **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 than their Host Page:**

  As explained in section 1.4 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 non HTML objects:**

  This filter concerns all objects linked to a page (via an <A> HTML tag) that are not HTML text 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 non HTML objects outside of Hierarchy:**

  Linked non HTML objects 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.

- **Save Linked non HTML objects on Remote Server:**

  Linked non HTML objects that are located on a different host server than the one specified by the base URL will be ignored, unless this option is checked.

- **Non HTML objects: Save 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.

  This option only concerns non HTML objects and embedded images, i.e. it is not necessary to include the “html” and “htm” file extensions, as HTML files are controlled by the *Save Parsed HTML pages* option explained above.

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

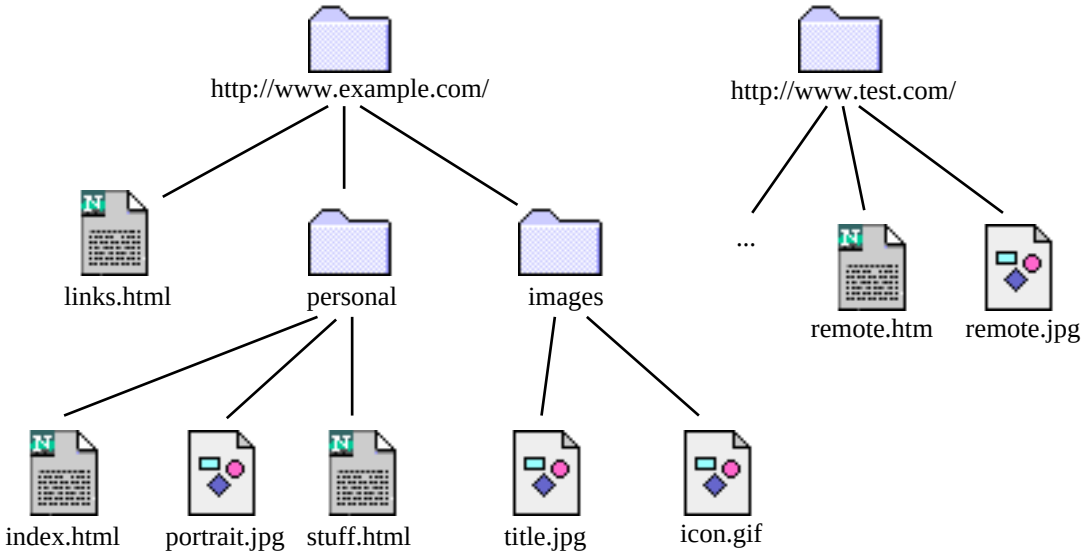
- **Save only if URL matches Regexp:**

  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 non HTML files and embedded images.

  More information on regular expressions can be found in section

1.6.

Please note that all of these 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 in PageSucker's control fields:

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. 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, is analyzed independently from the filter settings, but it is saved to the local disk only if the *Save Parsed HTML Pages* option is enabled in the filters panel.

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 on. 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 than their Host Page**" is checked.
links.html	linked HTML file	Not downloaded, as not in the specified hierarchy ("links.html" is not a direct or indirect member of the directory "personal"). HTML files not inside the hierarchy are always ignored.
portrait.jpg	embedded image	Downloaded and saved if "**Save Embedded Images**" is checked.
title.jpg	linked non HTML object	Downloaded and saved if both the "**Save Linked non HTML objects**" and "**Save Linked non HTML objects outside of Hierarchy**" options are checked, 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.
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.
remote.html	linked HTML file	Not downloaded, as on remote server. HTML files on remote servers are always ignored.

1.6 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 ...).

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 Perl5 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"`

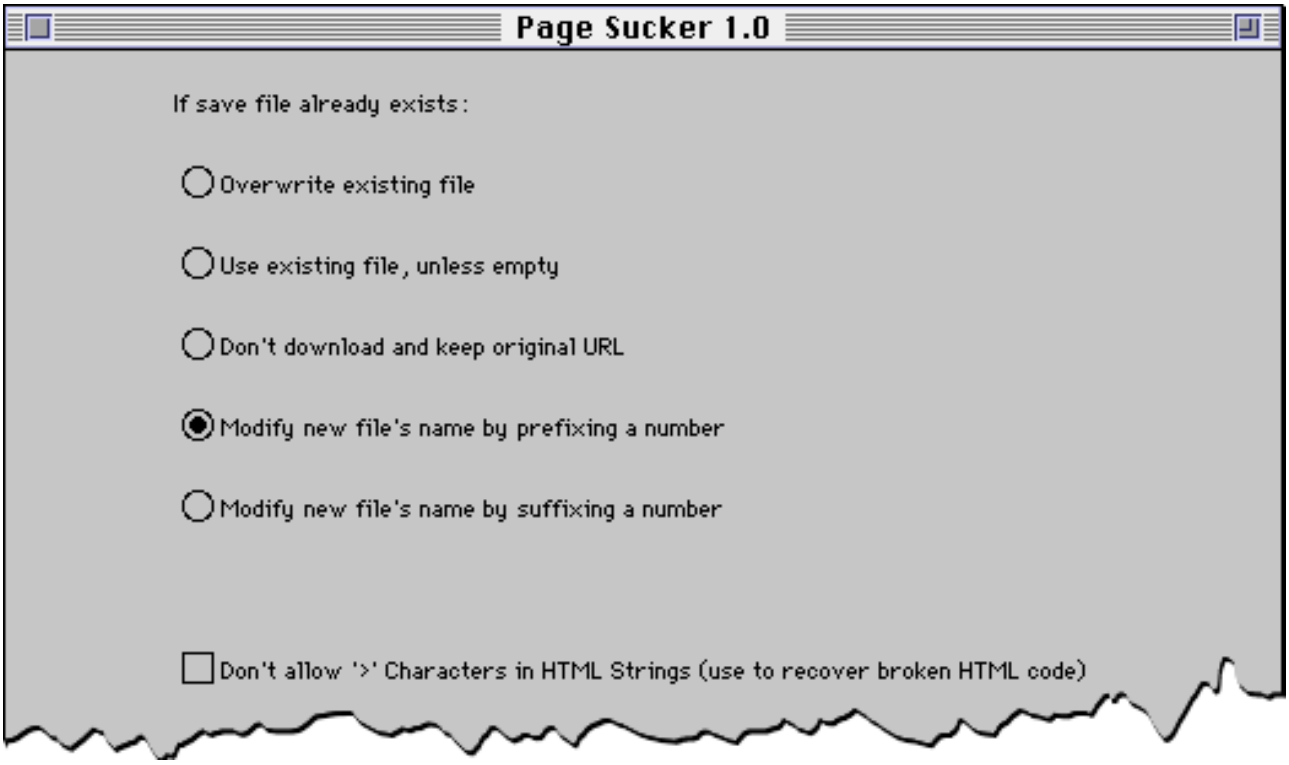
Finally, here are some examples of regular expressions that are of direct use with PageSucker's URL matching filter:

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.

1.7 Other Settings - The Preferences Panel

The Preferences user interface panel can be accessed with the *Preferences* button at the bottom of the main window. (You can switch to one of the other user interface panels using the *Control* or *Filters* button at the bottom of the window.)



The Preferences panel lets 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 page that contains the URL 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 page that contains the URL 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.
- **Don’t download and keep original URL** – the remote file is not downloaded and the local copy of the referring 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”).

There’s another checkbox in the Preferences panel that 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 to download picture </A>
```

which is accepted by older browsers (e.g. Netscape 1.0) but not understood by the more current browser versions (Netscape 3.0 etc.) because of the missing double quote after “picture.jpg”. 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.

1.8 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 1.7) 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. Common problems with version 1.0.2 of Apple’s MRJ are for example slight screen update errors.

- Usually the clicking of the main window’s close box should cause PageSucker to quit. Depending on the interpreter used, this might not work, and a security error might be reported in the console window. Try quitting the interpreter in that case to get rid of the running PageSucker instance.

- 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 *Preferences* panel. 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 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 recognizes the type of a file to download by the extension of its name. So, all files ending in “.html” or “.htm” are assumed to be HTML files, names that don’t have any extension (like, for example, the name “pictures”) are assumed to be directory names, and all other extensions are considered to belong to non HTML files. This involves that files having wrong extensions (e.g. a picture file with the name “pic.htm”) or data files having no file name extensions cannot be downloaded correctly. The only solution to this problem would be for PageSucker to download each file, try to interpret its contents, then decide if it is needed or not. This method is not only difficult to implement, but would also greatly reduce PageSucker’s execution speed. Moreover, it is impossible 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”. There is no way to determine from the client’s point of view which one is the right interpretation.

- 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 *Control* panel.

- The current PageSucker version only recognizes Macintosh and Windows 95/NT platforms correctly. 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.

- URLs containing a parameter part (a substring following a semicolon attached to the filename) are not yet handled correctly. This shouldn't be a big problem as such URLs tend to be extremely rare. Support for parameter strings will be added in a future release of PageSucker.

- In the Macintosh version of PageSucker, there's an "About" menu item in the Apple menu which doesn't do anything. This menu item is automatically added by Apple's MRJ engine. Specific functionality could have been added in the Mac version, but for cross platform compatibility reasons this has not been done.

1.9 Shareware Notice And Contact Information

PageSucker is shareware. That means that it can be distributed for free and you can test and use it for a certain time (let's say one month) 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).

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.

Please note that PageSucker has not been crippled in any way, i.e. it's not a demo version. You get the full functionality right away without the need of entering annyoing registration codes. We trust in your honesty that you will pay for the product if you like it.

Such as to make paying as easy as possible, we are using the Kagi payment service. 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. On other platforms, please use the Web page at

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

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:

Centre de Ressources Multimedia
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.

1.10 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.