# GrabURL

Serge Emond

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>GrabURL | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | Serge Emond | February 12, 2023 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# GrabURL

## 1.1 GrabURL.guide/Contents

---

GrabURL

Version 0.6



---



## 1.2 GrabURL.guide/About English

A Note About English

You will find in this document sentences incorrectly formed. Bad English. Very bad. My orthograph is bad, my verbs are bad, everything is bad.

I write the documentation of my programs in English because it's the only way to write ONE document that anyone can understand.

Sometimes when I browse new files on AmiNet, I see files that I must grab... I grab them. It always happens to the one that seems the most interesting: After minutes of downloading, no English documentation. Even the program is in a forein language. The first thing I do after such a discovery is to delete the lot. (Except if the forein language is French, but it never happened to me.)

It would be *LOT* faster for me to write this document in French but...

Anyway, if you didn't get the point, all the flafla above means:

There are many grammatical errors in this document and I don't care about them. What is important to me for now is to produce a readable documentation, nothing more. (And I'm sure it's readable...)

## 1.3 GrabURL.guide/Introduction

Introduction

You enter on Yahoo, do a search about "Brain" and find a marvelous site containing lots of data on the Brain. You want to grab everything but all the information is scattered across thousands of .html files. A real nightmare to grab everything with a web browser.

More, one of your hobbies is to view pictures. It relaxes yourself and you have enough space on your harddrive not to bother with selecting the images before you download them. Yesterday you found a superb site containing thousands of sunsets. You don't want to stay 20 hours to click each one and press enter in the requester. Worst some of them are not clickable and your browser don't have any option to save those to disk, you have to find the original filename in the html file and copy the file from the disk cache (if there is one!).

The solution: a robot that can download a file, parse it to find other files, grab them, and so on. More, it should grab some files and forget others. It should allow to download only during x hours, no more than y megs or z files.

GrabURL is such a thing. It uses GrabHTTP, UrlManager and ParseHTML to do his job quickly and efficiently. Please read the docs of those 3 programs before using GrabURL.

I know, there is other programs doing this: One that I can't remember its name (I really don't like it) and GetURL.rexx .

The main reason I created GrabHTTP, UrlManager, ParseHTML and GrabURL is that I wanted a FAST way to retreive HTTP urls the way I wanted.

## 1.4 GrabURL.guide/Features

Features

- Use 3 'C' programs, which means QUICK compared with ARexx-only scripts

- Is ARexx, which means flexible compared with 'C' only programs.

- Give a lot of control to the user

- Puts the full URL in file's comment

## 1.5 GrabURL.guide/Legal information

Legal information

License

No warranty

## 1.6 GrabURL.guide/License

License

GrabURL is released under the concept of freeware. This means you are allowed to use and copy this program freely, as long as the following requirements are fulfilled:

All files are copied without any alterations or modifications. If any
extra files are added, it must be obvious that they do not belong to
the original distribution, and that they do not need to be included in
any redistribution. Exception: So called "BBS ads" may not be added.
The copying is done on a non-commercial basis. A small fee to cover
media costs etc. may be charged.
The copier is not claiming the copyright of this program.
You are allowed to modify GrabURL.rexx and redistribute it but not to
add it to the original distribution.
Any exceptions from the above require a written permission from the author.

## 1.7 GrabURL.guide/Legal issues/No warranty

No warranty

There is no warranty for the programs, to the extent permitted by applicable
law. Except when otherwise stated in writing the copyright holder and/or
other parties provide the programs "as is" without warranty of any kind,
either expressed or implied, including, but not limited to, the implied
warranties of merchantability and fitness for a particular purpose. The
entire risk as to the quality and performance of the programs is with you.
Should the programs prove defective, you assume the cost of all necessary
servicing, repair or correction.
In no event unless required by applicable law or agreed to in writing will
any copyright holder, or any other party who may redistribute the programs as
permitted above, be liable to you for damages, including any general,
special, incidental or consequential damages arising out of the use or
inability to use the programs (including but not limited to loss of data or
data being rendered inaccurate or losses sustained by you or third parties or
a failure of the programs to operate with any other programs), even if such
holder or other party has been advised of the possibility of such
damages.

## 1.8 GrabURL.guide/Requirements

Requirements

GrabURL requires at least OS v39 (2.0 release) and AmiTCP (or Miami). It also requires RexxDosSupport.library (Aminet: util/rexx/rexxdossupport.lha) V2+.

GrabURL also requires three external programs I made.

GrabHTTP is an ARexx host that can grab files using the HTTP transfer protocol. It has a simple but nice window showing progress information.

UrlManager is an ARexx host that maintains a list of urls with flags. It is used to quickly parse big lists of urls.

ParseHTML scans an HTML file to find all the files mentionned in it. It then writes an url-list that UrlManager can read. It is extremely quicker than if this operation was done with ARexx. If you want even quicker results, you can make it resident using the DOS command "resident ParseHTML".

## 1.9 GrabURL.guide/History

The Past

Version 0 Revision 1 - 19.8.1996 and before

Nothing recorded.

Version 0 Revision 2 - 20.8.1996

· Added MaxCount

· Added MaxTime

· Added MaxBytes

· Added NoSrc

· Added NoBG

· Added NoHRef

· Added filecomment support

· Added config opts to change the font used by GrabHTTP

· Other stuff not recorded.

· Added def.WriteBufSize.

· Added def.Translate.

· Added def.TranslateTo and changed the way the translation works.

## 1.10 GrabURL.guide/What's next

What's Next - The To Do List

· Replace special characters in filenames (ie "~" -> "_")

· Add FTP support via NCFTP or FTPMount (or GrabFTP? :)

· Strip '?' in filenames

## 1.11 GrabURL.guide/Bugs

Known Bugs

· Urls beginning with "ftp://" are always flagged as 'Q'ueued

## 1.12 GrabURL.guide/Credits

Credits

GrabURL was entirely written by Serge Emond altough GetURL.rexx inspired me for the list of options. GetURL was written by James Burton and can be found on AmiNet.

## 1.13 GrabURL.guide/Author

About the author

If you wish to send me comments, bug reports, gifts, money or whatever, then use this address:

Serge Emond

3392 des Anémones

Jonquière, Québec

Canada, G7X 7V3

I do not live there most of the time but whatever you send WILL reach me.

EMail: emonds@microtec.net

emonds@ere.umontreal.ca

(!) Currently no email (!)

## 1.14 GrabURL.guide/Flags

Flags

Each url maintained in memory have a "flag" representing its status. Here is a list of the flags used by GrabURL:

Q Queued. This file is to be grabbed.

R Received. This file has been grabbed.

F Failed. An error occured while grabbing.

U Unused. This cannot be processed. (Ie do not begins with "http://")

For more information about flags, see:

WorkFile

## 1.15 GrabHTTP.guide/This is BETA

This is a BETA release

This release is BETA. This means it has not been fully tested. I release it only because it seems stable and it works well on my computer. As stated in legal issues I may not be responsible for any dammage caused by the use of this program.

## 1.16 GrabURL.guide/Arguments

Arguments

GrabURL can only be used from a Shell.

You can break it at anytime by sending it a CTRL-C break.

Text arguments

URL Url to download.

PATTERN Pattern to select urls.

INFILE File containing an url list.

SAVEROOT Directory where to put urls.

WORKFILE File containing all urls with their flags.

Numeric arguments

DELAY Delay to wait between each grab.

DEPTH Recursion level.

MAXCOUNT Max. number of file to grab.

MAXTIME Max. time to grab.

MAXSIZE Max. size a file can have.

MINSPACELEFT Min. space to leave on disk.

Switches

HELP Display arguments.

HEADERSONLY Save the headers but not the files.

IFMODIFIED Grab the file if modified since last grab.

NOBG Don't grab background images.

NODIRS Don't create directories - save in current dir.

NOHREF Don't grab referenced files.

NOSRC Don't grab src urls.

NOTEXISTS Grab only if file does not exists on disk.

QUERY Allow '?' in urls.

RECURSIVE Allow recursion.

RETRY Retry urls that failed.

SAVEHEADERS Save transfer headers to disk.

## 1.17 GrabURL.guide/Arguments/URL

URL (URL/M)

This allows you to add url(s) directly from the command line. These URLs will be flagged as Queued . You can add multiple urls this way. Any url added on the command line will prevail over file-urls. (Ie an url flagged received in the workfile will be flagged not received).

An url must begins with "HTTP://" since this is the only protocol supported for now.

See also:

InFile

WorkFile

## 1.18 GrabURL.guide/Arguments/MaxCount

MaxCount (MAXCOUNT=MC/N) (Abbrev: MC)

This is the maximal number of file to download. You can use this switch with WorkFile to download only a defined number of file per day. Use 0 disable this option. Default = 0.

See also:

MaxTime

MaxBytes

## 1.19 GrabURL.guide/Arguments/MaxTime

MaxTime (MAXTIME=MT/N) (Abbrev: MT)

This is the maximal time in minutes to download. You can use this switch with WorkFile to download only a certain time / day. Use 0 to disable this option. Default = 0.

This is not exact. For example, if you specify MaxTime=10 and you download this:

Bozo.jpg 3 mins (Tot: 3 mins)

Clown.jpg 6 mins (Tot: 9 mins)

Nuage.png 2 mins (Tot: 11 mins)

<it stops here>

See also:

MaxBytes

MaxCount

## 1.20 GrabURL.guide/Arguments/MaxBytes

MaxBytes (MAXBYTES=MB/N) (Abbrev: MB)

This is the maximal number of bytes to download. You can use this switch with WorkFile to download only a certain quantity of bytes per day. Use 0 to disable this option. Default = 0.

This is not an exact number. For example, if you specify MaxBytes=4096 and you download this:

Bozo.jpg 1000 bytes (Tot: 1000 bytes)

Clown.jpg 3090 bytes (Tot: 4090 bytes)

Nuage.png 1500 bytes (Tot: 5590 bytes)

<it stops here>

See also:

MaxCount

MaxTime

## 1.21 GrabURL.guide/Arguments/NoBG

NoBG (NOBG/S)

If specified, all background images will not be grabbed. This is only useful in conjuntion with Recurvise .

See also:

Recursive

NoHRef

NoSrc

## 1.22 GrabURL.guide/Arguments/NoHRef

NoHRef (NOHREF/S)

If specified, all files referenced will not be grabbed. This is only useful in conjuntion with Recurvise .

An HREF file is a file on which you have to click to get when you are in a web browser. (Images, another web page, etc...)

See also:

Recursive

NoBG

NoSrc

## 1.23 GrabURL.guide/Arguments/NoSrc

NoSrc (NOSRC/S)

If specified, all urls referred with a SRC tag will not be grabbed. This is
only useful in conjuntion with Recursive . SRC tags are used
to specify an image to be shown with the text. It is also used with non-standard
kludges like the frames of NetScape. Thus using NOSRC will prevent grabbing HTML
referenced by frames AND images to be shown with the text. I'm sorry but there is
no way to distinguish them in this preliminary version.

See also:

Recursive

NoBG

NoHRef

## 1.24 GrabURL.guide/Arguments/Pattern

Pattern (PATTERN=P/K) (Abbrev: P)

This allows you to specify an AmigaDOS pattern. Each URL recursively found will only be added if it matchs this pattern.

## 1.25 GrabURL.guide/Arguments/SaveRoot

SaveRoot (SAVEROOT=SR/K) (Abbrev: SR)

This is the directory to use as a base for all files we receive.

Suppose we execute:

GrabURL http://www.da.com/flowers/images/coquelicot.png sr ram:

Then the file will be saved as

ram:www.da.com/flowers/images/coquelicot.png

The default root dir is set in GrabURL. (See Setting Defaults )

## 1.26 GrabURL.guide/Arguments/WorkFile

WorkFile (WORKFILE=WF/K) (Abbrev: WF)

This is a file where all the urls processed are saved. If the file exists at run-time, it will be loaded. This way you can break a transfer and continue it later.

Each line of that file follow this template:

<flag> <depth> <url>

See also:

Flags

Depth

InFile

URL

## 1.27 GrabURL.guide/Arguments/InFile

InFile (INFILE=IN/K) (Abbrev: IN)

This is a file containing one url/line. These urls are processed exactly as if they were given on the command line.

See also:

URL

## 1.28 GrabURL.guide/Arguments/Help

Help (HELP/S) (Abbrev: H)

This gives the supported command line options and a brief description for each one.

## 1.29 GrabURL.guide/Arguments/HeaderOnly

HeaderOnly (HEADERONLY=HO/S) (Abbrev: HO)

When present, this indicates GrabURL not to download the file but only the transfer header.

This is only useful in conjontion with SaveHeaders .

## 1.30 GrabURL.guide/Arguments/NoDirs

NoDirs (NODIRS=ND/S) (Abbrev: ND)

This tells GrabURL not to create directory but to put the file in the current directory. See SaveRoot for more explanations.

## 1.31 GrabURL.guide/Arguments/NotExists

NotExists (NOTEXTSTS=NE/S) (Abbrev: NE)

If present, GrabURL will only download a file if it is not already existing on disk.

## 1.32 GrabURL.guide/Arguments/Query

Query (QUERY=Q/S) (Abbrev: Q)

If present, GrabURL will download files containing '?'. Normally such a file is in reality a query or an executable with arguments. For example Yahoo will use it to do a search, and many counters ("You are the x to visit this page") use it.

For general use, you should never use this switch.

## 1.33 GrabURL.guide/Arguments/Recursive

Recursive (RECURSIVE=R/S) (Abbrev: R)

If present, GrabURL will scan each reeived HTML file to find new urls. These urls will then be added to the list and eventually be downloaded.

See also:

Depth

## 1.34 GrabURL.guide/Arguments/Retry

Retry (RETRY/S)

This tells GrabURL to retry files that failed for one reason or another.

Each file will only be retried once and after all the other files have been tried at least one time.

## 1.35 GrabURL.guide/Arguments/SaveHeaders

SaveHeaders (SAVEHEADERS=SH/S) (Abbrev: SH)

If present GrabURL will save the transfer headers along with the files. The filename will be the same with '.HDR' added.

See also:

HeaderOnly

## 1.36 GrabURL.guide/Arguments/Delay

Delay (DELAY/N)

This is the number of seconds to wait between each file. You should not use a delay of 0 seconds if you plan to download a LOT of files - using this switch helps to diminish a little bit the load on the net.

The factory setting is 0.5 second and can be changed. (See Defaults )

## 1.37 GrabURL.guide/Arguments/Depth

Depth (DEPTH=D/K) (Abbrev: D)

This is the level of recursion to use. (ie How deep to look in the files)

Level ’0’ means the "current" file. All files contained in the current files have a level of "1" and so on.

Specifying 5 (along with the Recursive switch) would grab

the files on the command line and specified with InFile ,

giving them a "depth" of 5. The files contained in them would have a depth of 4.

Then 3, 2, 1 and finally 0. The files contained in the depth-0 file won’t be grabbed.

The default is "-1" which means "grab the world".

## 1.38 GrabURL.guide/Arguments/MaxSize

MaxSize (MAXSIZE=MS/N) (Abbrev: MS)

With this you can specify the maximal size a file can have in order to download it.

Ie "MaxSize 4096" will refuse a file having 4097 bytes but accepts 4096.

0 is the default which disable this option.

## 1.39 GrabURL.guide/Arguments/MinSpaceLeft

MinSpaceLeft (MINSPACELEFT=MSL/N) (Abbrev: MSL)

This is the minimal space to leave free on the device when downloading a file. The default is 0 (disabled).

## 1.40 GrabURL.guide/Arguments/IfModified

IfModified (IFMODIFIED=IM/S) (Abbrev: IM)

If present, each file that already exists on the disk will only be grabbed if the file on the server is newer than the one on the disk.

See also:

Defaults - Setting the timezone

## 1.41 GrabURL.guide/Defaults

Changing the defaults

The defaults are set directly in GrabURL.rexx using an ASCII editor.

When in the editor (ie GoldEd, CignusEd, ...), load "GrabURL.rexx" and go down some lines. You should see a line beginning with:

DoDefaults:

You are now where the defaults are all set.

Here's a list of the defaults used for the command line arguments :

opts.Url.Count (Factory: 0)

Must always be 0.

opts.Depth (Factory: -1)

This is the default depth to use. When doing recursion, this tells GrabURL not to care about depth.

opts.InFile (Factory: '')

This is the default file to load when starting GrabURL. This file contains one url/line. See InFile .

opts.MaxSize (Factory: 0)

This is the maximal size a file can have in order to be grabbed. See MaxSize .

opts.MinSpaceLeft (Factory: 0)

This is the minimal space to leave on disk when grabbing. See MinSpaceLeft .

opts.Pattern (Factory: '')

This is the default pattern to use when doing recursion. See Pattern .

opts.SaveRoot (Factory: 'Storage:Urls/')

This is the default path where to save urls. See SaveRoot .

opts.WorkFile (Factory: '')

This is the default work file to use.

opts.Delay (Factory: 0.5)

This is how much seconds to wait between each grabbed file. See Delay .

opts.MaxTime (Factory: 0)

This is the maximal time in minutes to download. See MaxTime .

opts.MaxCount (Factory: 0)

This is the maximal number of file to download. See MaxCount .

opts.MaxBytes (Factory: 0)

This is the maximal number of bytes to download. See MaxBytes .

Here's a list of the defaults used that cannot be modified on the command line:

def.KeepReceived (Factory: 1)

1 = Change the flag of received urls

0 = Remove them from the list

def.KeepFailed (Factory: 1)

1 = Change the flag of failed urls

0 = Remove urls that failed from the list.

def.KeepUnused (Factory: 1)

1 = Change the flag of urls that could not be processed.

0 = Remove them from the list.

def.Secure (Factory: 1)

1 = Save the workfile each time an url is modified (ie received, failed, removed, ...)

0 = Save the workfile only when GrabURL is completely done.

def.Accept (Factory: '*/*')

This is the MIME type(s) the remote host is allowed to send.

def.EMail (Factory: '')

This is the UserName/EMail address to send to each remote host we encounter. Using '' forces GrabURL to send NO user information across the net. See Browsing anonymously . This information can looks like:

Serge Emond <emonds@ere.umontreal.ca>

def.TimeZone (Factory: 300)

This is the number of minutes to add to the local time to obtain GMT time. It is only used when using the IfModified argument. The default is 5 hours (300 minutes) which is Québec/Ontario and east part of USA.

def.WriteBufSize (Factory: 16*1024 (16k))

This is the size of the buffer GrabHTTP will use for the file itself. The received data will be written only every def.WriteBufSize bytes.

def.Translate (Factory: '~:[]()*'

If it is not '' then the characters given will be translated to chars in def.TranslateTo when saving the file to disk. For example, if def.Translate="*!" and def.TranslateTo="_+" then all "*" in the filename will be changed to "_" and all "!" to "+".

def.TranslateTo (Factory: copies("_", length(def.Translate)))

String to translate to. See def.Translate. This string MUST be at least the length of def.Translate.

def.TempFile (Factory: 't:GrabURL.'UID)

This is a temporary file used when scanning received HTML files to do recursion. UID is a variable containing a unique ID and should be used in the name (as in the factory setting) to prevent collision if you run multiple GrabURL at the same time.

def.ParsePattern (Factory: '(FTP|HTTP)://#?')

Only urls matching this pattern are added to the url list. If you want to see if there was ftp urls, allow them with this setting and use a workfile. You can then see them by searching for "ftp:" in the workfile.

def.Path (Factory: '')

This is the path where to find ParseHTML, GrabHTTP & UrlManager executables.

def.ParseHTML (Factory: def.Path'ParseHTML')

This is the path & name of ParseHTML. See Requirements .

def.GrabHTTP (Factory: 'run <>nil: 'def.Path'GrabHTTP')

This is the name of GrabHTTP. It must be started with "run". See Requirements .

def.UrlManager (Factory: 'run <>nil: ''def.PathUrlManager')

This is the name of UrlManager. It must be started with "run". See Requirements .

def.Output (Factory: 'CONSOLE:')

This is the filename where GrabURL will output all its.. output. "CONSOLE:" will output to the console from which GrabURL was started.

def.Font (Factory: 'topaz.font')

This is the font used for GrabHTTP's progress window. It must be fixed-width.

def.FSize (Factory: 8)

This is the size of the font.

um (Factory: 'UM'UID)

This is the name of the port to use to communicate with UrlManager. You should use the variable UID as in the factory setting to prevent collision.

gh (Factory: 'GH'UID)

This is the name of the port to use to communicate with GrabHTTP. You should use the variable UID as in the factory setting to prevent collision.

## 1.42 GrabURL.guide/MIME Types

MIME

What follows come directly from AWeb's HTML documentation:

MIME (Multipurpose Internet Mail Extensions) is a mechanism for specifying and describing the format of Internet message bodies. It was primarily designed for e-mail, but MIME types are used also to identify the type of data in the HTTP protocol, the most widely used protocol on the World Wide Web.

A MIME type consists of a type and a subtype. The type describes the major class of data, like text or image. The subtype is used for a subdivision of the major type into different formats, like GIF or JPEG images.

According to RFC 1521, the following official MIME types are defined:

TEXT/HTML

This is a document in the HTML hypertext format. Virtually all pages on the Web are in this format.

TEXT/PLAIN

This type is used for plain text documents (normally in ASCII).

APPLICATION/OCTET-STREAM

This describes a binary file. The file could be processed by some application. An example of this would be an LHA archive.

APPLICATION/POSTSCRIPT

The document is in PostScript format.

IMAGE/GIF

IMAGE/JPEG

These are images, in GIF and JPEG format.

AUDIO/BASIC

This type is used for audio data encoded using 8-bit ISDN mu-law [PCM].

VIDEO/MPEG

This is an animation in MPEG format.

In addition to these official types and subtypes, it is allowed to define extension MIME types and subtypes. These should start with X- to avoid collisions with future official MIME types.

## 1.43 GrabURL.guide/Browsing Anonymously

Browsing Anonymously

An HTTP request can have a field containing email and name information about the user. This way, some people send their email address on every server they connect to. Most of the time it won't do anything but... there are twisted webmasters who use this information to send anoying emails ie to sell products.

It you want to see if your Web Browser sends such information without letting you knowing about it, have a look at this page: HTTP://www.uiuc.edu/~ejk/WWW-privacy.html.

By default GrabURL won't send any information about the user. In fact, there is absolutely no function allowing to grab any information from your system so you can be sure it won't send anything. :)