XAD Plugin 1.0

The Multi-Format Unarchiver For Hollywood

Andreas Falkenhahn

Table of Contents

1 General information

1.1 Introduction

This plugin allows Hollywood scripts to open archives using the XAD system. The XAD system is a toolkit designed for handling various file and disk archiver formats. It consists of a master library and a number of XAD clients for handling different file and disk archiver formats. By default, the XAD system already supports many common archiver formats like ZIP, LhA, LZX, CAB, bzip2, TAR, and RAR, which makes it a powerful tool for people that have to deal with compressed data a lot.

The XAD system has its origins on the Amiga platform for which it was originally developed by Dirk Stoecker in the 1990s. That's why all versions of xad.hwp for AmigaOS and compatibles still use the `xadmaster.library` provided by the host system. Both AmigaOS 4 and MorphOS come with `xadmaster.library` installed. For AmigaOS 3 and AROS `xadmaster.library` is available as third-party software. On all other platforms, xad.hwp uses an open-source version of the XAD system.

The XAD plugin uses Hollywood 6.0's new file and directory adapter plugin interfaces which allow you to iterate through file archives supported by the XAD system as if they were normal directories. Files within archives supported by the XAD system can also be accessed as if the archive was a normal directory. It is not necessary to unpack a file stored in an archive to a temporary file before it can be opened. Hollywood 6.0's file adapter plugin interface allows direct streaming from the archive into the respective file handler.

Additionally, xad.hwp offers a range of functions to deal with archives supported by the XAD system for people needing fine-tuned control over the XAD system. These functions allow you to conveniently iterate over all files and directories in an archive, read file attributes like date, protection bits, compression details, etc. and extract files to an external or virtual file.

1.2 Terms and conditions

the implied warranty of merchantability or fitness for a particular purpose. See Section A.2 [LGPL license], page 17, for details.

This software uses xadmaster.library by Dirk Stoecker. See Section A.1 [xadmaster.library license], page 17, for details.

xad.hwp for non-Amiga platforms uses the XAD RAR client which was written by Chris Young <chris@unsatisfactorysoftware.co.uk> and Stephan Matzke <stephan.matzke.adev@gmx.de>. This client is built on a modified version of RAR_Extractor by Shay Green which is based on the UnRAR source code by Alexander L. Roshal.

Amiga is a registered trademark of Amiga, Inc.

All other trademarks belong to their respective owners.

DISCLAIMER: THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDER AND/OR OTHER PARTIES PROVIDE THE PROGRAM "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 PROGRAM IS WITH YOU. SHOULD THE PROGRAM 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 PROGRAM 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 PROGRAM (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 PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

1.3 Requirements

- minimum: Hollywood 6.0 or better
- recommended: Hollywood 7.0 or better is recommended for Unicode support

1.4 Installation

Installing xad.hwp is straightforward and simple: Just copy the file `xad.hwp` for the platform of your choice to Hollywood's plugins directory. On all systems except on AmigaOS and compatibles, plugins must be stored in a directory named `Plugins` that is in the same directory as the main Hollywood program. On AmigaOS and compatible systems, plugins must be installed to `LIBS:Hollywood` instead. On Mac OS X, the `Plugins` directory must be inside the `Resources` directory of the application bundle, i.e. inside the `HollywoodInterpreter.app/Contents/Resources` directory. Note that

`HollywoodInterpreter.app` is stored inside the `Hollywood.app` application bundle itself, namely in `Hollywood.app/Contents/Resources`.

On Windows you should also copy the file `xad.chm` to the `Docs` directory of your Hollywood installation. Then you will be able to get online help by pressing F1 when the cursor is over a xad.hwp function in the Hollywood IDE.

On Linux and Mac OS copy the `xap` directory that is inside the `Docs` directory of xad.hwp's distribution archive to the `Docs` directory of your Hollywood installation. Note that on Mac OS the `Docs` directory is within the `Hollywood.app` application bundle, i.e. in `Hollywood.app/Contents/Resources/Docs`.

2 About xad.hwp

2.1 Credits

xad.hwp was written by Andreas Falkenhahn. Work on this project was started in early 2014 as a proof-of-concept demonstration of Hollywood 6.0's powerful new file and directory adapter plugin interfaces which allow plugins to hook into Hollywood's file and directory handlers. xad.hwp makes use of this feature by making Hollywood think that archives supported by the XAD system are just directories so that it is possible to iterate through them using Hollywood's normal directory functions or open files within archives without extracting them first.

If you want to contact me, you can either send an e-mail to `andreas@airsoftsoftwair.de` or use the contact form on `http://www.hollywood-mal.com`.

2.2 Frequently asked questions

This section covers some frequently asked questions. Please read them first before asking on the mailing list or forum because your problem might have been covered here.

Q: Where can I ask for help?

A: There's a lively forum at `http://forums.hollywood-mal.com` and we also have a mailing list which you can access at `airsoft_hollywood@yahoogroups.com`. Visit `http://www.hollywood-mal.com` for information on how to join the mailing list.

Q: I have found a bug.

A: Please post about it in the dedicated sections of the forum or the mailing list.

2.3 Known issues

Here is a list of things that xad.hwp doesn't support yet or that may be confusing in some way:

- tbd

2.4 Future

Here are some things that are on my to do list:

- tbd

Don't hesitate to contact me if xad.hwp lacks a certain feature that is important for your project.

2.5 History

Please see the file `history.txt` for a complete change log of xad.hwp.

3 Usage

3.1 Activating the plugin

There are two ways of using this plugin: You can either activate the plugin globally by setting the `InstallAdapter` tag to `True` when `@REQUIRE-ing` it. To do this, simply put the following preprocessor command at the top of your script:

```
@REQUIRE "xad", {InstallAdapter = True}
```

If you activate the plugin in this way, it will become globally available and all ensuing commands that deal with files will support the opening of files from XAD archive sources. For example, you could do something like this then:

```
LoadBrush(1, "test.rar/testpicture.jpg")
```

If you only need to open very few files from XAD archive sources, you can also choose to not activate the plugin globally by omitting the `InstallAdapter` tag on `@REQUIRE` and simply use the `Adapter` tag offered by most Hollywood commands to tell the respective Hollywood command to open the file using the xad.hwp plugin. Here is an example:

```
LoadBrush(1, "test.rar/testpicture.jpg", {Adapter = "xad"})
```

By using the `Adapter` tag, `LoadBrush()` is told to open the specified file using the specified adapter, which is "xad" in our case. Thus, the `Adapter` tag allows you to use this plugin even without having installed a global file adapter for it first.

The same is true for Hollywood functions dealing with directories. Once the XAD plugin has been activated, it is possible to do things like the following:

```
OpenDirectory(1, "test.rar")
```

You could then iterate over all files and directories in `test.rar`. If you haven't activated a global adapter for xad.hwp, then just use the `Adapter` like above, e.g.

```
OpenDirectory(1, "test.rar", {Adapter = "xad"})
```

See the next chapter for more details on treating archives supported by the XAD system as directories.

3.2 Archives as directories

When setting the `InstallAdapter` tag to `True`, the XAD plugin hooks into Hollywood's directory handler to make Hollywood believe that archives supported by the XAD system are normal directories. This allows you to iterate over all files and directories inside an archive using normal functions from Hollywood's DOS library.

For example, to iterate over all files and directory inside a file named `test.rar` you could use the following code:

```
OpenDirectory(1, "test.rar")
Local e = NextDirectoryEntry(1)
While e <> Nil
    DebugPrint(e.name)
    e = NextDirectoryEntry(1)
Wend
CloseDirectory(1)
```

If you don't want to start from the root directory inside `test.rar`, you can also conveniently start from a subdirectory by just pretend that `test.rar` is a directory, e.g. to access a subdirectory named `files` inside `test.rar` just do the following:

```
OpenDirectory(1, "test.rar/files")
```

Finally, it is also possible to recursively iterate through all files and directories inside an archive. Here is a function which does that:

```
Function p_DumpArchive(d$, idt)
    Local id = OpenDirectory(Nil, d$)
    Local e = NextDirectoryEntry(id)
    While e <> Nil
        If e.Type = #DOSTYPE_DIRECTORY
            DebugPrint(RepeatStr(" ", idt) .. "+", e.name)
            p_DumpArchive(FullPath(d$, e.name), idt + 4)
        Else
            DebugPrint(RepeatStr(" ", idt) .. "",e.name,e.size,e.time)
        EndIf
        e = NextDirectoryEntry(id)
    Wend
    CloseDirectory(id)
EndFunction
```

To dump the contents of an archive, just call this function like that:

```
p_DumpArchive("test.rar", 0)
```

It will then print a nice tree of the archive's contents.

3.3 Extracting files

Since xad.hwp can hook into Hollywood's file handler, extracting files is just a matter of using Hollywood's `CopyFile()` function on the file you wish to extract. For example, to extract a file named `testpicture.jpg` from `test.rar`, just use the following line:

```
CopyFile("test.rar/testpicture.jpg", "outputdir")
```

Since `CopyFile()` can also copy whole directories including all subdirectories and because xad.hwp hooks into Hollywood's directory handler as well, it is even possible to extract a whole archive using `CopyFile()`, like this:

```
CopyFile("test.rar", "outputdir")
```

This, however, will be pretty slow because xad.hwp will open and close the RAR archive for every single file that needs to be extracted which is of course a performance killer. That's why xad.hwp also offers a dedicated function to extract files for fine-tuned control over archives. See Section 4.2 [xad.ExtractFile], page 11, for details.

3.4 Archive basics

Archives supported by the XAD system are just a collection of files that are stored at indices ranging from 0 to the number of entries in the archive minus 1. It is not necessary to store directories as individual entries. Instead, they can also be stored as part of a filename, e.g. if a file is stored as `a/b/c/test.txt` in the archive, then the directories a, b, and c are

implicitly declared as existing even though they don't have their individual entries in the archive but just exist as part of a file. You have to be prepared to deal with that if you manually parse archives using the functions provided by this plugin.

3.5 Linking files

Keep in mind that all files declared in the preprocessor commands are linked automatically into your applet or executable when Hollywood is in compile mode. Thus, if you do something like the following, not only `testpicture.jpg` but the whole RAR archive `test.rar` will be linked to your applet or executable:

```
@BRUSH 1, "test.rar/testpicture.jpg"
```

If you don't want that, you can set the optional `Link` to `False`. If `Link` is set to `False`, Hollywood won't link the specified file to your applet or executable. This means, however, that you have to distribute `test.rar` with your applet or executable so that the data can be loaded from it. Here's how to disable linking:

```
@BRUSH 1, "test.rar/testpicture.jpg", {Link = False}
```

When done like this, Hollywood will never link the file into your applet or executable. Instead, it will always be loaded from the specified file.

4 Function reference

4.1 xad.CloseArchive

NAME
xad.CloseArchive – close xad archive

SYNOPSIS
`xad.CloseArchive(id)`

FUNCTION
This function closes the specified xad archive.

INPUTS

`id`	identifier of the xad archive to be closed

4.2 xad.ExtractFile

NAME
xad.ExtractFile – extract file from xad archive

SYNOPSIS
`xad.ExtractFile(id, idx, dst$[, table])`

FUNCTION
This function can be used to extract the file at the index `idx` inside the xad archive specified by `id` to the external file specified by `dst$`. If `dst$` already exists, it will be overwritten. An optional table argument allows you to specify further options for the operation.

The following tags are currently recognized in the optional table argument:

`Password:`
If the file you wish to extract is protected by a password, you have to specify this password here. If you don't specify this tag, the default password set using `xad.SetDefaultPassword()` is used.

`Callback:`
This tag allows you to pass a function that should be called from time to time. This can be useful if you'd like to show a status bar or something while the xad file is being extracted. The function will receive a table as its sole argument. The table will have the following fields initialized:

`Action:` Initialized to `"ExtractFile"`.

`ID:` Contains the identifier of the xad archive.

`Progress:`
Contains a value between 0 and 100 indicating how much work has already been done.

`UserData:`
Contains the value you passed in the `UserData` argument (see below).

You can also pass user data that should be forwarded to your callback using the tag below.

`UserData:`
This tag can be set to arbitrary data that should be passed to the callback you passed in the `Callback` tag. If you specify this tag without the `Callback` tag, it is simply ignored.

INPUTS

`id`	identifier of the xad archive to use
`idx`	index of file to extract
`dst$`	desired destination file
`table`	optional: table containing further parameters

4.3 xad.GetFileAttributes

NAME
xad.GetFileAttributes – get file attributes

SYNOPSIS
```
t = xad.GetFileAttributes(id, idx)
```

FUNCTION
This function returns attributes of the file at the index specified by `idx` inside the xad archive specified by `id`. `xad.GetFileAttributes()` returns a table with the following information about the file:

`Type:` This will be `#DOSTYPE_FILE` if the entry is a file or `#DOSTYPE_DIRECTORY` if the entry is a directory.

`Name:` This field will contain a string with the full path to this file or directory.

`Size:` The size of the file in bytes or 0 for directories.

`CompressedSize:`
The compressed size of the file in bytes or 0 for directories.

`Encryption:`
This will be set to `True` if the file is stored with encryption, `False` otherwise.

`Flags:` This field will receive a combination of protection flags of the file or directory. See the documentation of the `GetFileAttributes()` function in your Hollywood manual for details on possible values of this field.

`Time:` The datestamp for the file. This will be in the standard Hollywood date format of `dd-mmm-yyyy hh:mm:ss`.

INPUTS

`id` identifier of the xad archive to use

`idx` index of the file to query

RESULTS

`t` table containing file attributes

4.4 xad.GetFileAtIndex

NAME

xad.GetFileAtIndex – get name of file by index

SYNOPSIS

```
name$ = xad.GetFileAtIndex(id, idx)
```

FUNCTION

This function returns the name of the file at index `idx` in the xad archive specified by `id`.

To find out the number of files in a xad archive, you can query `#XADATTRNUMENTRIES` with Hollywood's `GetAttribute()` function. See Section 4.5 [xad.GetObjectType], page 13, for details.

INPUTS

`id` identifier of the xad archive to use

`idx` index to query (in the range of 0 to number of entries minus 1)

RESULTS

`name$` name of entry at index

4.5 xad.GetObjectType

NAME

xad.GetObjectType – get xad archive object type

SYNOPSIS

```
type = xad.GetObjectType()
```

FUNCTION

This function returns the object type used by xad archives opened using the `xad.OpenArchive()` function. You can then use this object type with functions from Hollywood's object library such as `GetAttribute()`, `SetObjectData()`, `GetObjectData()`, etc.

In particular, Hollywood's `GetAttribute()` function may be used to query certain properties of xad archives opened using `xad.OpenArchive()`. The following attributes are currently supported by `GetAttribute()` for xad archives:

`#XADATTRNUMENTRIES:`
: Returns the number of entries in the xad archive.

`#XADATTRTYPE:`
Returns a string describing the archiver name used by the current xad archive, e.g. "RAR" or "LhA".

INPUTS
none

RESULTS

`type` internal xad archive type for use with Hollywood's object library

EXAMPLE

```
xad.OpenArchive(1, "test.rar")
XAD_ARCHIVE = xad.GetObjectType()
numentries = GetAttribute(XAD_ARCHIVE, 1, #XADATTRNUMENTRIES)
type$ = GetAttribute(XAD_ARCHIVE, 1, #XADATTRTYPE)
```

The code above opens `test.rar` and queries the number of entries in the archive and its type via `GetAttribute()`.

4.6 xad.LocateFile

NAME
xad.LocateFile – find file in xad archive

SYNOPSIS

```
idx = xad.LocateFile(id, name$[, table])
```

FUNCTION
This function searches for the file specified by `name$` inside the xad archive specified by `id` and returns its index if it is found, otherwise -1 is returned.

The optional table argument can be used to specify further options. The following table tags are currently recognized:

`NoCase:` If this tag is set to `True`, `xad.LocateFile()` won't distinguish between upper and lower case characters. This makes the search slower. Defaults to `False`.

`NoDir:` If this tag is set to `True`, `xad.LocateFile()` will just match the file name so it will also trigger if the file is in a subdirectory in the archive. Defaults to `False`.

INPUTS

`id` identifier of the xad archive to use

`name$` name of the file to locate

`table` optional: table argument containing further options (see above)

RESULTS

`idx` index of file inside xad archive or -1 if it couldn't be found

4.7 xad.OpenArchive

NAME

xad.OpenArchive – open a xad archive

SYNOPSIS

```
[id] = xad.OpenArchive(id, filename$)
```

FUNCTION

This function attempts to open the xad archive specified by `filename$` and assigns `id` to it. If you pass `Nil` in `id`, `xad.OpenArchive()` will automatically choose a vacant identifier and return it. If the file does not exist, this function will fail.

Although xad.hwp will automatically close all open xad archives when it quits, it is strongly advised that you close an open xad archive when you are done with it using the `xad.CloseArchive()` function because otherwise you are wasting resources.

Note that `xad.OpenArchive()` will create a standard Hollywood object which can also be used with functions from Hollywood's object library such as `GetAttribute()`, `SetObjectData()`, `GetObjectData()`, etc. See Section 4.5 [xad.GetObjectType], page 13, for details.

INPUTS

`id` identifier of the file or `Nil` for auto id selection

`filename$`
name of the file to open

RESULTS

`id` optional: identifier of the file; will only be returned when you pass `Nil` as argument 1 (see above)

4.8 xad.SetDefaultPassword

NAME

xad.SetDefaultPassword – set default password

SYNOPSIS

```
xad.SetDefaultPassword(pwd$)
```

FUNCTION

This function can be used to set a default password that is used to decrypt files if no other password is provided. If you pass an empty string in `pwd$`, the default password is unset.

The default password is used by `xad.ExtractFile()` if no other password is explicitly specified. It is also used by normal Hollywood functions if the global adapter has been enabled for xad.hwp by setting the `InstallAdapter` tag to `True`.

INPUTS

`pwd$` new default password or empty string to unset the default password

Appendix A Licenses

A.1 xadmaster.library license

The xadmaster.library is Shareware with very special conditions. There are no usage restrictions (means this is no Crippleware!), and xadmaster.library use can be implemented into programs without any restrictions. Only the end users have to pay shareware fee, if using xadmaster.library. There are 3 forms of payment to get a registered user:

- Send 20 EUR or US $20 to my snail mail address or use any other more secure transfer system.
- Make detailed bug reports about 3 currently unknown bugs. I decide whether a bug report is counted or not. Mainly this depends on two facts: was the bug already reported / does the description help me to find the bug
- You make an own external client and send me the source and the client for inclusion in next release. Some more are good as well!

For programmers using xadmaster.library in own applications (f.e. in virus scanners):

- It would be fine, if you would tell me suggestions or bugs, as implementing xadmaster.library use may show you some problems.
- You must state above conditions in your distribution, at a place, where users can read it.
- And please do not forget: Most time you are users as well, so think about above conditions.

NOTE: This conditions only cover the xadmaster.library! External clients may have different legal state. For example LZX and Zoom both are Freeware. So expect clients to have any legal state. You need to check this for every client. Normally the version string in the client should tell you what type it is.

A.2 LGPL license

GNU LESSER GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the Combined Work with a copy of the GNU GPL and this license document.

c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.

d) Do one of the following:

0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

Index