MirrorManager

A local Aminet mirror management system for the Amiga
Edition 0.8beta, for Version 1.8
20 June 1994

by Harald Kunze and Tobias Ferber

This is the second edition of the MirrorManager documentation,
and is consistent with version 1.8 of 'MirrorManager'.

Published by Tobias Ferber and Harald Kunze
Goethestrasse 32,
76135 Karlsruhe
Printed copies are available for DM 10 each.

1 Introduction

MirrorManager is an ARexx based management system for directory trees. It is particularly useful for BBS or Mailbox sysops who use to download from an Aminet (ftp)site. It is however also useful for 'normal' users (like e.g. the authors) who simply hesitate to delete new software.

Before there was MirrorManager our usual way to archive downloaded files was very simple: We moved them from the '`incoming`' directory to a quickly blowing up archive directory which of course lead to confusion and chaos. But there is a much better way to organize a download archive: The local Aminet mirror ...

However, the Aminet hierarchy is not fix (in fact it changes quite often) and so keeping the proper Aminet hierarchy manually is obviously a work intensive task. But now MirrorManager is born and MirrorManager handles all this for you!

In detail, MirrorManager does the following:

- MirrorManager creates the complete Aminet tree in a given directory adding AmigaDOS filenotes to these directories according to those in the Aminet '`TREE`' file.
- MirrorManager adds filenotes to the files in your '`incoming`' directory according to those listed in the Aminet '`INDEX`' file.
- MirrorManager copies or moves the files in your '`incoming:`' path to the location in your local Aminet mirror which is listed in the Aminet '`INDEX`' file – or optionally to a directory which is mapped to this location.
- MirrorManager examines your local Aminet mirror and reports differences in name, location and comment there and in the Aminet '`INDEX`' file.
- MirrorManager generates an index file for your local Aminet mirror.

MirrorManager is fully ARexx based and everything it does can be controled via ARexx. ARexx commands are discussed in Chapter 8 [ARexx Commands], page 16. See Chapter 7 [ARexx Scripts], page 13, for information about the supplied ARexx script files.

You also need to set-up several paths and filenames in order to use MirrorManager. See Chapter 2 [Installation], page 2, Chapter 3 [Configuring], page 3 for details.

2 Installing MirrorManager

MirrorManager is distributed together with the Installer program from Commodore. See Chapter 12 [Installer], page 33, for more information.

Simply double-click the '`Install-MirrorManager`' icon to install MirrorManager to disk.

We spent much time and tried to be extremly careful when writing this Installer script. MirrorManager needs neither assigns nor environment variables so the Installer script will not modify any parts of your system. During the installation procedure a directory '`MirrorManager`' will be created and all files will be copied into this directory. No other parts of your disk will be touched unless you do not specify them explicitly. We also included many help texts. Please read them if you are not sure about what's going on!

After installing MirrorManager you need to set-up several paths and filenames. See Chapter 3 [Configuring], page 3, for more information.

> **Note:** MirrorManager needs MUI. See Chapter 11 [MUI], page 32, for information about how to obtain it. You should also have the actual Aminet '`INDEX`' and '`TREE`' file to take off with MirrorManager.

3 Configuring MirrorManager

MirrorManager comes with the '`MirrorManager`' MUI application and several ARexx scripts configuring the GUI.

This chapter explains how to configure the MirrorManager GUI application. If you are only interested in the supplied ARexx Shell scripts then you might want to skip this part.

There are however also ARexx scripts which can be used from within the Shell and which do not need the MUI application.

3.1 MirrorManager Configuration Scripts

Before MirrorManager can take off you need to create a configuration script containing the path and filenames of your personal installation. This is quite obvious, since MirrorManager needs to know where to find your local Aminet mirror.

A configuration script is an ARexx program which is executed when you select the `Project/Load...` menu item. Of course these configuration scripts make extensive use of the GUI built-in ARexx command `ADD`. See Section 8.3 [MirrorManager ARexx commands], page 18, for details.

One way to configure MirrorManager is using the GUI's `Edit` menu. This is however a dull job — especially if you are configuring MirrorManager from scratch. For this reason I included my personal configuration script '`MirrorManager.rexx`', which contains the most needed features. You will however need to change the path and filenames in '`MirrorManager.rexx`' because these are correct for my personal installation only.

The prefered way to set-up '`MirrorManager.rexx`' for your installation is using the supplied Installer script '`Configure`', which can be found in the '`MirrorManager/rexx`' drawer. '`Configure`' will ask you interactively for all needed paths and filenames and it offers a short help text on every step. See Section 3.2 [Using Configure], page 4, for more information.

Alternatively you can edit '`MirrorManager.rexx`' with a text editor like e.g. '`MEmacs`'. This can however only be recommended for advanced users because it requires a little bit more knowledge of the MirrorManager interna. See Section 7.1 [MirrorManager scripts], page 13, '`MirrorManager.rexx`' for details.

3.2 Using `Configure`

Coming with MirrorManager is my personal configuration script '`MirrorManager.rexx`'. Since this file contains absolute path and filenames which are valid in my environment only, you need to set-up '`MirrorManager.rexx`' for your installation before you can use it. The prefered way to to set-up these values is with the aid of the Installer script '`Configure`', which is located in the '`MirrorManager/rexx`' drawer.

Simply double-click the '`Configure`' icon. You will be asked for all needed path and filenames and on every step a short help text will be available.

After you satisfied '`Configure`' with all requested locations, the `CONFIGNAME` ToolType in the '`MirrorManager`' program icon will be set to '`rexx/MirrorManager.rexx`'. This forces MirrorManager to load this configuration script when starting up.

'`Configure`' will remember your settings. I.e. if you run '`Configure`' again, it will prompt with your last selection for each file and pathname.

Advanced users may prefer editing '`MirrorManager.rexx`' with a text editor like e.g. '`MEmacs`'. See Section 7.1 [MirrorManager scripts], page 13, for details.

> **Caution:** If any path filename in '`MirrorManager.rexx`' has been modified without using the '`Configure`' script, then '`Configure`' will *not* prompt with the current default. This is due to the Installer program, which does not support evaluation of code at run-time. Use the supplied Shell script '`SetConfigureDefaults`' to copy your assignments from '`MirrorManager.rexx`' to '`Configure`'[1].

3.3 ToolTypes and Command Line Arguments

Unless you did not change the default configuration, MirrorManager will come up with a tutorial demo. To change this you should modify the MirrorManager ToolType value `CONFIGNAME`. Here is a list of legal ToolTypes:

`CONFIGICON=...`

If this ToopType is given, MirrorManager will create a project icon when saving the configuration from within the GUI. For example,

```
CONFIGICON=xtras/def_mm.info
```

will force MirrorManager to use '`def_mm.info`' from the '`xtras/`' drawer as the default icon. As you can see in the above example, the path to the project icon can

[1] I'm terribly sorry for that unsuffisticating solution, but I have no idea how to solve this problem in a better way. Please contact me if you have a good idea!

be specified relative to the MirrorManager tool, i.e. relative to '`PROGDIR:`'. Absolute pathnames like e.g. '`env:sys/def_mima.info`' would be legal as well.

Note: MirrorManager tries to be very careful with your project icon.

- MirrorManager will adjust the stack size in your project icon to the maximum of the values in the MirrorManager tool icon your default project icon. Note that MirrorManager needs at least 10240 bytes of stack!
- MirrorManager will set the icon position of the `CONFIGICON` to `NO_ICON_POSITION`.

By default, i.e. if no `CONFIGICON` ToolType is given, no icon will be saved with the configuration.

`APPSTART=...`

This ToolType defines the default application startup method. Actually it it used to force a certain default tool entry in the project icon saved with the configuration. If no project icon has been specified via the `CONFIGICON` ToolType then `APPSTART` does nothing.[2]

For example,

```
APPSTART=/MirrorManager
```

sets the default tool for the configuration icon specified via `CONFIGICON` to be '`MirrorManager`' in the parent directory.

By default, MirrorManager will choose the default tool for the saved project icon in the following way:

- If the configuration is saved into a drawer inside the MirrorManager directory then MirrorManager will use a relative pathname, i.e. a pathname relative to MirrorManager's '`PROGDIR:`'. This has the advantage that you can move your MirrorManager directory to any location on your hard drive without changing the default tool for all existing project icons.
- On the other hand, if you select a save path for your configuration which is located outside the MirrorManager directory then MirrorManager will use an absolute pathname for the project icon's default tool.

`CONSOLE=...`

This ToolType sets the default console window for ARexx. You normally do not need this console window and so the default is

```
CONSOLE=NIL:
```

It is however useul to set a console window like e.g.

```
CONSOLE=CON:30/20/600/80/MirrorManager/CLOSE/AUTO
```

[2] This is a lie: In fact, if you force icon creation by selecting the '`Create Icons`' item in the `Settings` menu without specifying a project icon via the `CONFIGICON` ToolType then MirrorManager uses the system default project icon (usually '`env:sys/def_project.info`') and adjusts the default tool for this icon according to the ToolType value of `APPSTART`.

for debugging purposes.

`CONFIGNAME=...`

This ToolType specifies the startup configuration scriptfile. By default a tutorial demo configuration script will be loaded on startup.

`CONFIGNAME` can also be used as a Shell argument keyword. If you execute MirrorManager from within a Shell then you can specify the configuration scriptfile either relative to your current directory or with an absolute pathname.

The ToolType value can be given as an absolute pathname or relative to the GUI application's path: '`PROGDIR:`'.

`MAXPATHLEN=...`

Not implemented yet!

`MAXITEMLEN=...`

Not implemented yet!

`MAXCOMMANDLEN=...`

Not implemented yet!

`MAXARGLEN=...`

Not implemented yet!

4 Creating an Aminet Hierarchy

This chapter explains the initial creation of the local Aminet mirror. Please make sure that MirrorManager is installed and configured before you go on reading here. See Chapter 2 [Installation], page 2, Chapter 3 [Configuring], page 3 for details.

5 Description of the ListView Items

When starting MirrorManager with the configuration script '`MirrorManager.rexx`' last will add some items to the ListView. This chapter will explain the use of these items.

You must have configured the file '`MirrorManager.rexx`' to follow the steps below. See Chapter 3 [Configuring], page 3, for more information. You might want to skip this chapter if you don't want to use the MUI application.

`Create/Update Aminet TREE`
: This item executes '`MakeTree`' in order to create the complete Aminet directory hierarchy from your Aminet '`TREE`' file. A filenote will be added to all directories listed in the '`TREE`' file.

`Delete Empty Directories`
: This item executes '`CutTree.mm`' on your Aminet directory which will delete all empty directories there.

`Update Aminet DirNotes`
: For each existing directory in your Aminet Hierarchy, a filenote will be added. The filenote will be read from the Aminet '`TREE`' file.

`Add FileNotes to INCOMING`
: For all files in your incoming directory a filenote will be added. Last will be looked up in the Aminet '`INDEX`' file.

`Add FileNotes from RECENT`
: Works like `Add FileNotes to INCOMING` except that the filenotes will be looked up in the '`RECENT`' file.

`Add FileNotes from WANTED`
: Works like `Add FileNotes to INCOMING` except that the filenotes will be looked up in the '`WANTED`' file.

`Cleanup INCOMING`
: Works like `Add FileNotes to INCOMING`. Additionally the files will be moved into the correct directory in your Aminet hierarchy. Non-existant directories will be created.

`Cleanup with new RECENT`
: Works like `Cleanup INCOMING` except that the filenotes and the destination directory will be looked up in the '`RECENT`' file.

`Create FAST Index`
: Splits the Aminet '`INDEX`' file and creates several small '`FAST`' index files. These files will be written to the fast index path which has been asked for in the '`Configure`' script.

`Create/Update LOCAL Index`
: A local index file will be created for your local Aminet mirror.

`Edit LOCAL Index File`

Edits your local Aminet mirror's index file.

`Reorganize Local Mirror`

Compares the contents of your local index file with the Aminet '`INDEX`' file. Files located in a different directory than listed in the Aminet '`INDEX`' file will me moved to the correct location. Non-existant directories will be created automatically.

Files which do not exist in the Aminet '`INDEX`' will be moved to the `KickedPath`, a directory which has been asked for in the '`Configure`' script.

`Re-Insert Kicked Out Files`

Files which have been moved to the `KickedPath` (e.g. by `Reorganize Local Mirror`) will be moved back to their former location.

> **Caution:** This operation needs your former local index file to look up the kicked out files' locations. I.e. you should not update your local index file after calling `Reorganize Local Mirror` if you want to move back some of the kicked out files into their old directories.

`Sort Aminet INDEX File`

Some people prefer an Aminet '`INDEX`' file which is sorted by filenames, not by directories. This item asks you for the first sorting creterium and then sorts the Aminet '`INDEX`' file accordingly.

`Edit Aminet INDEX File`

This item runs an Editor and loads the Aminet '`INDEX`' file.

`Edit WANTED Index`

This item runs an Editor and loads your '`WANTED`' file.

`Edit Configuration Script`

This item runs an Editor and loads this configuration sctiptfile.

6 Menu Items and Menu Structure

6.1 The `Project` Menu

`Load...` This will bring up a filerequester asking for a MirrorManager configuration.

Because MirrorManager configurations are based on ARexx it will be loaded by executing it as an ARexx script, configuring MirrorManager via its ARexx port.

See also: ARexx command LOAD

`Save...` This will write your current configuration to disk without asking for a filename. It will be written over the last loaded configuration which will be deleted by this action. So be careful with this command!

See also: ARexx command SAVE

`Save As...`

This will bring up a filerequester asking for a name under which you would like to save your current configuration.

See also: ARexx command SAVE

`Locked` This switch is set by MirrorManager whenever it executes an ARexx script or command and is unset as soon as MirrorManager has finished processing and is idle again.

If you unset this switch you are able to execute further ARexx commands but this might be dangerous:

- Most ARexx scripts in the MirrorManager package work directly on your files, moving them from one directory to another, adding filenotes etc.
- Every MirrorManager ARexx script writes information into the MirrorManager Working Window.

Because of these two facts the scripts might 'disturb' each other. So the default behaviour of the scripts is to lock the MirrorManager GUI.

See also: ARexx command LOCK

`About...` This will bring up an About requester with some information about MirrorManager and its authors.

`Quit` This will terminate MirrorManager. If your configuration has changed since you last saved it you will have to confirm a requester to actually quit MM.

Note: MirrorManager will not quit under the following circumstances:

- there are some open filerequester
- the application is locked
- there are still some ARexx commands running

6.2 The `Edit` Menu

`Add` This will add an entry at the bottom of the listview. The initial name of the new entry is set to '`-- UNNAMED --`', the command and argument fields are emtpy.

See also: ARexx command ADD

`Insert` This will insert an entry into the listview at cursorposition. The initial name of the new entry is set to "– UNNAMED –", the command and argument fields are emtpy.

See also: ARexx command INSERT

`Remove` This will delete the currently active entry in the listview and activate its successor.

See also: ARexx command REMOVE

`Clone` This will make an identical copy of the currently active entry at cursorposition.

See also: ARexx command CLONE

`Edit...` This will bring up a new window where you are able to edit a listview entry. As long as the 'Edit Window' is open the 'Main Window' is busy.

An entry is split into three parts:

- The name of the entry appearing in the listview.
- The command which has to be an ARexx script or inline code.
- The arguments for the command. If you specify inline code in the command field the argument field is ignored.

`Clear` This will remove all entries from the listview and unset the internal `CONFIGNAME` variable.

See also: ARexx command CLEAR

`Sort` This will perform a case insensitive sort on the entries in the listview.

See also: ARexx command SORT

`Top` This will move the currently active entry to the beginning of the listview.

See also: ARexx command TOP

`Up` This will move the currently active entry in the listview one line up.

See also: ARexx command UP

`Down` This will move the currently active entry in the listview one line down.

See also: ARexx command DOWN

`Bottom` This will move the currently active entry to the end of the listview.

See also: ARexx command BOTTOM

6.3 The `Output` Menu

`Open Window`

This will open the 'Working Window' where the MirrorManager ARexx scripts will print their information.

`Clear Window`

This will clear the contents of the 'Working Window'

See also: ARexx command MESSAGE

`Save Log...`

This will bring up a filerequester asking you for a filename under which you would like to save the contents of the 'Working Window'.

See also: ARexx command SAVELOG

7 ARexx Scripts

All ARexx scripts coming with MirrorManager expect their arguments introduced by a keyword which must be seperated from the argument value by one or more `SPC` characters.

For example,

```
rx MakeTree.rexx FROM "treefile" TO "pathname"
```

is the correct way to run `MakeTree.rexx` which has the template `FROM/K/A,TO/A`

> **Caution:** Legal keywords for each command are listed here in `ReadArgs()` template style. There is however a difference between the way `ReadArgs()` parses the arguments and the way we do.
>
> For example,
>
> ```
> rx MakeTree.rexx FROM="treefile" TO="pathname"
> ```
>
> would be legal for `ReadArgs()` but it is *not* legal for the ARexx scripts listed here. The '`=`' is not a legal keyword value delimiter for ARexx.

There are '`.rexx`' and '`.mm`' script files coming with the MirrorManager distibution

'`.rexx`' script files are absolutely *independent* and do never address to the MirrorManager's GUI application. This means that you can execute them from a CLI/Shell, configure them for a use with Stefan Becker's ToolManager, or install them on any directory management application.

'`.mm`' script files need the MirrorManager MUI application. They depend on the way the GUI passes arguments to it's ARexx scripts and they heavily use the ARexx commands offered my the MUI application.

In addition to the options listed below, '`.mm`' scripts usually know a switch `AUTO/S`. Last can be used to force closing of the message window after the script has completed.

7.1 MirrorManager ARexx scripts

`MakeTree FROM/K/A,TO/A,NOCREATE/S`

`MakeTree.rexx` creates all directories listed in the given `FROM` file relative to a given `TO` directory. The directory structure and a comment for each directory will be taken from that `FROM` file, which is usually called '`TREE`' in case of the Aminet.

The '`TREE`' file will be scanned top-down and line by line, empty lines are ignored as well as lines beginning with a '`#`' character. The first word (i.e. everything upto the first white space) in each line will be taken as the direcory name, the rest of the line will be added as a filenote to the directory.

For example, executing `rx MakeTree.rexx FROM tree TO a:b` with the file '`TREE`' looking somewhat like this

```
# All directories on Aminet

new           Upload area
recent        Files uploaded the last seven days
biz           Business software
biz/cad       Computer aided design
...
```

creates the directories '`a:b/new`', '`a:b/recent`', ... adding the filenotes (comments) '`Upload area`', ... respectively.

> **Caution:** `MakeTree.rexx` is smart enough to create a path '`a:b/c`' by creating the directory '`a:b`' before creating '`a:b/c`'. However, '`a:`' must be legal of course.

Using the option keyword `NOCREATE` will add filenotes for existing directories only. You might prefer this behaviour if you don't use to keep a complete Aminet hierarchy and `CleanupIncoming` or `ExamineIndex` created new directories.

`MakeTree.rexx` needs the AmigaDOS commands `MakeDir` and `FileNote` (and optionally `RequestFile` and `Delete`) available in your command searchpath.

`CutTree PATH/A`

`CutTree` deletes empty directories in the given path. I usually don't keep a complete Aminet directory hierarchy because many of them stay empty anyway and scanning them for new files

`SplitIndex FROM/K/A,TO/K/A`

bzzzzzz.

`MakeIndex FROM/K/A,TO/A`

`MakeIndex.rexx` is ment to create a local Aminet mirror index. In general it creates a list of all files available in a given path including file size and file comment.

> **Caution:** Due to a problem with `pragma('D')` you *must* execute this script using `RX` explicitly – even in the WShell.

This ARexx script needs the AmigaDOS commands `List` and `Sort` available in your path.

`ExamineIndex FILE/K/A,WITH/K/A,PATH/K,MOVE/S,COMMENT/S,MAKEPATH/S,FAST/S`

hmmmmmm.

`SortIndex FROM/K/A,FILE/S,DIR/S,QUICK/S,AUTO/S`

`CleanupIncoming FROM/K/A,TO/K,WITH=INDEX/K/A,MOVE/S,COPY/S,NOCOMMENT/S,MAKEPATH/S,REPLACE/S,`

`CleanupIncoming.rexx` examines (non-recursively) the files in your '`incoming:`' directory and looks them up in an Aminet mirror `INDEX` file. For each file in your '`incoming:`' directory which is listed exactly *once* in the Aminet index file the following actions can be performed:

- A filenote (comment) can be added according to the one listed in your Aminet index file and
- The file can be copied (or moved) to the location listed in the Aminet index.

The Aminet directory hierarchy can be created using `MakeTree.rexx`

`CleanupIncoming.rexx` needs the AmigaDOS commands `List`, `Sort`, `Search`, `Filenote`, `Copy` and `Delete` available in your path.

`MirrorManager.rexx`

`MirrorManager.rexx` is the configuration file for the `MirrorManager` MUI application. `MirrorManager.rexx` will be executed by it's host on startup when the ToolType or CLI option `CONFIGNAME=rexx/MirrorManager.rexx` was set. On the other hand if `MirrorManager.rexx` was executed via `RX` or by double-clicking on it's icon and no empty application is running then it tries to run it's own MirrorManager executable. If you want to write your own configuration scripts then you should see Section 8.5 [Hints], page 25, for details.

To set-up path and filenames in `MirrorManager.rexx` you should use the configure script, which can be found in the '`MirrorManager/rexx`' directory. See Section 3.2 [Using Configure], page 4, for further details.

`Demo`

`Demo.rexx` demonstrates the use of the GUI's ARexx interface. You should execute this demo either via double-clicking on its icon or by loading it as a *configuration* via `Project/Load`. The actions performed by '`Demo.rexx`' are fully self-explaining and need no further documentation here.

`whoami.rexx`

This script resolves the own path and filename. We decided to include this scriptfile as a tutorial example because it is quite a difficult task to do this correctly.

Currently the `PARSE SOURCE` instruction is limited to a length of 64 characters. This is an enormous problem since this makes it impossible for an ARexx script to resolve its pathname if it is located deeper in the directory hierarchy. We have reported this bug, but we did not receive a reply yet.

8 ARexx Commands

MirrorManager is fully ARexx based and everything it does can be controled via ARexx.

Since MirrorManager is a MUI application, it offers the default commands which are understood by every MUI program. Every MUI application is able to receive commands via the built-in ARexx port. (For more information about MUI see Chapter 11 [MUI], page 32)

If you want to write your own ARexx script files for MirrorManager you should also read Section 8.5 [Hints], page 25.

8.1 MUI built-in ARexx commands

`QUIT` Ends the application.

The behaviour of this command is very much like clicking the closing Gadget of the application or pressing `RCOMMAND-Q` or `ESC`.

`HIDE` Hides (iconifies) the MirrorManager application. This will normally bring up the application icon '`MirrorManager.info`' or – if this is non-existant – '`def_MUI.info`' which will be loaded from '`env:sys/`'. This can however be customized with the MUI Preferences program.

`SHOW` Shows (pops up) the iconified MirrorManager application.

`INFO ITEM/A`

According to the given `ITEM` parameter the result string is filled with the following contents:

`TITLE` Title of the application (I.e. '`MirrorManager`')

`AUTHOR` Author of the application

`COPYRIGHT`
Copyright message

`DESCRIPTION`
Short description

`VERSION` Version string

`BASE` Name of the ARexx port

`SCREEN` Name of the public screen

For example,

```
OPTIONS RESULTS
ADDRESS 'MIRRORMANAGER.1'
```

```
'INFO Title';       SAY "Application title .......:" result
'INFO Author';      SAY "Author of the application:" result
'INFO Copyright';   SAY "Copyright message .......:" result
'INFO Description'; SAY "Short description ........:" result
'INFO Version';     SAY "Version string ..........:" result
'INFO Base';        SAY "Name of the ARexx port ..:" result
'INFO Screen';      SAY "Name of the pub screen ..:" result
```

`HELP FILE/A`

A list of all ARexx commands available for the application is written into the given file. In addition to the default commands MirrorManager supports many application specific commands. The help list will contain these commands as well.

For example,

```
ADDRESS 'MIRRORMANAGER.1'; HELP "ram:help.out"
```

creates the file '`ram:help.out`' with the following contents:

```
Standard-Commands:

Command        Template
-------        --------
quit
hide
show
info           ITEM/A
help           FILE/A


Commands for application "MirrorManager":

Command        Template
-------        --------
sort
clear
numentries
add            NAME/A,COMMAND/A,ARGS
clone          NAME
remove         NAME
rename         NAME/A
execute        NAME
activate       NAME
up             NAME
down           NAME
top            NAME
bottom         NAME
load           FILE
save           FILE
configname     FILE
appstart       COMMAND
message        CLEAR/S,OPEN/S,CLOSE/S,STRING
complete       PERCENTAGE/N
working        STRING
requestchoice  TITLE/K,GADGETS/A,BODY/A
requestfile    DRAWER,FILE/K,TITLE/K,SAVEMODE/S,DRAWERSONLY/S,NOICONS/S
```

```
savelog         FILE
lock            ON/S,OFF/S
```

8.2 MUI Error codes

In case of an error, MUI returns the following values to the rexx script:

`-1` Wrong command definition in host program. (Should never happen.)

`-2` Out of memory.

`-3` Unknown ARexx command.

`-4` Syntax error.

If any of MirrorManager's own ARexx commands fail then either one of the above or `1` will be returned. In general: A value `rc ~= 0` *always* indicates an error.

8.3 `MirrorManager` ARexx commands

`SORT` The `SORT` command performs a case insensitive sort on the MirrorManager's ListView. If `SORT` is called on an already sorted ListView then identically named entries will be exchanged.

Calling this function via ARexx has the same result as pressing `RCOMMAND-S`.

`CLEAR` The `CLEAR` command removes all items from the MirrorManager's ListView and it will unset the internal `CONFIGNAME` variable. As a consequence `NUMENTRIES` will return `result = 0` after this action has been performed. This can also be achieved by pressing `RCOMMAND-/`.

`NUMENTRIES`

Returns the number of items in the MirrorManager's ListView.

For example,

```
ADDRESS 'MIRRORMANAGER.1'
OPTIONS RESULTS

NUMENTRIES
SAY 'There are currently' result 'items'
```

`ADD NAME/A,COMMAND/A,ARGS`

Creates a new entry and adds it to the *bottom* of the ListView. The contents of the `NAME` field will appear in the MirrorManager's ListView, `COMMAND` must contain the pathname of an ARexx program and `ARGS` are the arguments for this ARexx program.

For example,

```
ADDRESS 'MIRRORMANAGER.1'
ADD '"Add filenotes to INCOMING"',
    '"rexx/CleanupIncoming.mm"',
    '"FROM incoming: WITH aminet:INDEX"'
```

will add an item `Add filenotes to INCOMING` which calls '`CleanupIncoming.mm`' with arguments `FROM incoming: WITH aminet:INDEX`.

Note that all the commata ',' in the above example are ARexx line continuation tokens. They must be removed if the `ADD` command is used in one single line.

When pressing `RCOMMAND-A` a new item will be added to the ListView. The new item will be named `-- UNNAMED --` by default and can be edited pressing `RCOMMAND-E`.

New items will always become the active item of the ListView.

You may also write ARexx inline code into the `COMMAND` field. The `ARGS` field should however be empty then.

For example,

```
ADDRESS 'MIRRORMANAGER.1'
ADD '"Edit current config"',
    '"*"OPTIONS RESULTS; CONFIGNAME; ADDRESS COMMAND ''Ed'' result*""'
```

will start an Editor '`Ed`' with the name of the current configuration script as argument.

See Section 8.4.2 [Calling Method], page 23, for more information about how MirrorManager executes ARexx code.

`CLONE NAME`

The `CLONE` command makes an identical copy of an item in the ListView. If no name is given, then the active item will be cloned. The new item will be inserted directly before the original item in the list and it will become active[1].

`CLONE` can be called via `RCOMMAND-C`.

`REMOVE NAME`

The `REMOVE` command deletes an item from the ListView. If no name is given then the active item will be removed. `REMOVE` activates the next item if there is a next one, otherwise, if the removed item was the last in the list then its previous item (which now is the last item) will become active.

For example,

```
ADDRESS 'MIRRORMANAGER.1'

REMOVE '"Edit current config"'
```

removes the item in the above example. This operation is identical to pressing `RCOMMAND-X`.

`RENAME NAME/A`

The `RENAME` command changes the name of the active item to the given one. This operation will only redraw the active line in the ListView. Renaming the current

[1] This is a lie. To be honest: The new item will be added directly below the original entry. The active item remains unchanged.

item can also be done by editing this item either by pressing RCOMMAND-E or selecting Edit/Edit... from the menu.

EXECUTE NAME

Executes a command by the name of its entry in the ListView. If no name is given then the active item's command will be executed.

EXECUTE can also be performed by double-clicking or pressing the RETURN key on the desired item.

ACTIVATE NAME

ACTIVATE highlights a ListView item and makes it become the active item. If no name is given then ACTIVATE returns the name of the active item. Otherwise the command behind the given item will be returned.

For example,

```
OPTIONS RESULTS
ADDRESS 'MIRRORMANAGER.1'

ACTIVATE; name = result

IF rc ~= 0 THEN SAY 'There is currently no active item.'
           ELSE DO
             ACTIVATE '"' || name || '"'; cmd = result
             SAY name 'performs the following command:' cmd
             END
```

UP NAME Moves a ListView item one line up. If no name is given then the active item will be moved. This operation is identical to pressing RCOMMAND-U.

For example,

```
OPTIONS RESULTS
ADDRESS 'MIRRORMANAGER.1'

UP; pos= result

IF rc = 0 THEN DO WHILE pos > 0
  'UP'; pos= result
  END
ELSE SAY 'There is currently no item active.'
```

would be a straight forward implementation of the TOP command.

DOWN NAME Moves a ListView item one line down. If no name is given then the active item will be moved. This operation is identical to pressing RCOMMAND-D.

TOP NAME Moves an item to the beginning of the ListView. If no name is given then the active item will be moved. This operation is identical to pressing RCOMMAND-T.

BOTTOM NAME

Moves an item to the end of the ListView. If no name is given then the active item will be moved. This operation is identical to pressing RCOMMAND-B.

LOAD FILE The LOAD command executes a configuration file. If called without arguments the last configuration will be re-loaded.

`SAVE FILE` `SAVE` writes an ARexx executable to disk which, when executed restores the current MirrorManager configuration. If no filename is passed then the '`.rexx`' file will be written with the current default name.

The `SAVE` command is identical to selecing `Project/Save` from the menu. You cannot write back the current configuration via a single keypress. Pressing `RCOMMAND-W` will pop-up a filerequester asking you for the filename to write to.

`CONFIGNAME FILE`

The `CONFIGNAME` command sets the current filename for the configuration script. If called without parameters `CONFIGNAME` will return the current filename.

The config filename will be set always after loading a configuration via `Project/Load` or after saving via `Project/Save as...`. It will also be set if these actions are performed via the ARexx commands `LOAD` or `SAVE` respectively.

If MirrorManager has been run by executing a configuration scriptfile then this scriptfile will set `CONFIGNAME` to its own name. Otherwise, if MirrorManager has been started by double-clicking its icon then `CONFIGNAME` will be taken from the ToolTypes. When started from the CLI/Shell then the startup `CONFIGNAME` can be given via

```
1> MirrorManager CONFIGNAME=rexx/myconfig.mm
```

Caution: If MirrorManager is executed without specifying `CONFIGNAME` then `CONFIGNAME` will be *unset*! This is so because the configuration scripts run MirrorManager without any arguments and of course they don't want MirrorManager to execute another configuration script. In this case `Project/Save` will behave like `Project/Save as...` and open a file requester.

`APPSTART COMMAND`

The `APPSTART` command sets the default command sequence for running MirrorManager. If no arguments are given then the current default will be returned.

`APPSTART` can be set via a ToolType called `APPSTART=<command sequence>`. The default is '`Run MirrorManager`'.

`MESSAGE CLEAR/S,OPEN/S,CLOSE/S,STRING`

The `MESSAGE` command prints the given string in the MirrorManager working window appending a new line to the current window contents. Calling `MESSAGE` without parameters is a no-op.

`MESSAGE CLEAR`
: wipes out the contents of the working window

`MESSAGE OPEN`
: opens the working window

`MESSAGE CLOSE`
: closes the working window

Given string will be appended to the window contents even if the working window is closed at the time MirrorManager recieves the `MESSAGE` command.

`COMPLETE PERCENTAGE/N`

The `COMPLETE` command sets the gauge in the MirrorManager working window to the given value which has to be between 0 and 100.

If COMPLETE is called without any parameters it will set the result string to the current gauge level.

Even if the working window is closed at the time MirrorManager recieves the COMPLETE command, the gauge will be set to the given value.

`WORKING STRING`

The `WORKING` command sets the title of the MirrorManager working window to the given string. If it is called without any argument WORKING returns the current window title in the result string.

`REQUESTCHOICE TITLE/K,GADGETS/A,BODY/A`

The `REQUESTCHOICE` command is similar to the Commodore '`RequestChoice`' program. This gives the possibility to interact with the user. `REQUESTCHOICE` opens a MUI-Requester which can be configured by

`TITLE` Title for the requester window. Defaults to "MirrorManager"

`GADGETS` Pointer to a string containing the possible answers. The format looks like "_Save|_Use|_Test|_Cancel". If you precede an entry with a '*', this answer will become the active object. Pressing `RETURN` will terminate the requester with this response. A underscore '_' character indicates the keyboard shortcut for this response.

`BODY` The requester's body text

The gadget and body string is parsed by MUI's text engine so it may contain the special characters

''`\n`'' Start a new line. With this character you can e.g. create multi line buttons.

'`ESC -`' Disable text engine, following chars will be printed without further parsing.

'`ESC u`' Set the soft style to underline.

'`ESC b`' Set the soft style to bold.

'`ESC i`' Set the soft style to italic.

'`ESC n`' Set the soft style back to normal.

'`ESC <n>`' Use pen number n (2..9) as front pen. n must be a valid DrawInfo pen as specified in '`intuition/screens.h`'.

'`ESC c`' Center current (and following) line(s). This sequence is only valid at the beginning of a string or after a newline character.

'`ESC r`' Right justify current (and following) line(s). This sequence is only valid at the beginning of a string or after a newline character.

‘ESC l’ Left justify current (and following) line(s). This sequence is only valid at the beginning of a string or after a newline character.

‘ESC I[s]’ Draw MUI image with specification <s>. See autodocs of image class for image spec definition.

a quick example:

```
OPTIONS RESULTS
ADDRESS 'MIRRORMANAGER.1'

REQUESTCHOICE TITLE "Dolle Sache" "_Yep!" "*Ec*EbMUI*En*Nis magic"
```

REQUESTFILE DRAWER,FILE/K,TITLE/K,SAVEMODE/S,DRAWERSONLY/S,NOICONS/S

The REQUESTFILE command ...

SAVELOG FILE

The SAVELOG command ...

LOCK ON/S,OFF/S

The LOCK command blafasel ...

8.4 How MirrorManager executes ARexx Code

If you double-click one of MirrorManager's ListView items, MirrorManager will examine the Command and Arguments field associated with the selected item.

8.4.1 ARexx inline code

If the contents of the Command field begins with a double quote ‘"’ then MirrorManager expects this to be inline code and executes it like the ARexx starter command ‘RX’ would. The only difference to ‘RX’ is that the default host is set to the MirrorManager's ARexx port and the default extension for external ARexx scripts is ‘.mm’.

8.4.2 ARexx script files

On the other hand, if the contents of the Command field does not begin with a double quote then the contents is expected to be the (path and) filename of an ARexx script. The pathname should be given relative to the path of the MirrorManager MUI application. However, also absolute pathnames are possible.

> **Caution:** You must not quote the contents of the Command field unless it is ment as ARexx inline code! MirrorManager will be able to execute your script file even if the path and/or the filename contains spaces.

Before executing an ARexx script MirrorManager will generate inline code on its own.

Consider the following example:

Name `Add filenotes to INCOMING`

Command `Ram Disk:MirrorManager/rexx/CleanupIncoming.mm`

Arguments

`FROM "Ram Disk:incoming" WITH "aminet:INDEX"`

After setting the default host for ARexx to the name of MirrorManager's ARexx port, MirrorManager will execute the following inline code in order to invoke '`CleanupIncoming.mm`':

```
"CALL 'Ram Disk:MirrorManager/rexx/CleanupIncoming.mm',
      'FROM','Ram Disk:incoming','WITH','aminet:INDEX'"
```

Double quotes inside the `Arguments` field are treated as you would expect that for ordinary Shell commands. I.e. A double quote inside a double quoted argument needs a leading asterisk '`*`'.

For example,

```
"a *quote* *" inside"
```

An asterisk however does not need to be prefixed by another asterisk.

Here is an overview how MirrorManager converts the contents of the `Arguments` field which is shown on the left hand side of the following table:

`a b c`	`'a','b','c'`
`a "b c"`	`'a','b c'`
`a"b`	`'a""b'`
`a'b`	`'a''b'`
`"a*"b"`	`'a""b'`
`"a'b"`	`'a''b'`
`"a""b"`	`'a','b'`

Consider the source code '`rxcallstr.c`' for verbose documentation and further details.

As you can see in the above example, MirrorManager will pass the arguments to the ARexx script in tokenized form. From within the ARexx script one can reach the `i`-th argument with `arg(i)`. The ARexx variable `arg()` contains the number of supplied aruments. This is identical to an invocation method of ARexx scripts with `RXFB_TOKEN` set. And it is much more convenient for the ARexx programmer because he doesn't need to parse an argument string on its own.

8.5 Hints for writing new ARexx script files for MirrorManager

Before writing ARexx scripts for MirrorManager you've got to make up your mind if you want to make it GUI-dependent or if your script file should better be executable from within any environment.

8.5.1 Writing GUI-dependent ARexx scripts

Programming the GUI's ARexx interface will most often be done in order to configure MirrorManager to your own digests. If your work should however be useful for a greater variety of MirrorManager users then you should keep in mind the following:

1. MirrorManager has *no* assignments and needs *no* environment variables. You should therefore never write MirrorManager configuration scripts depending on your environment and your assignments. Instead you should take advantage of the fact that MirrorManager guarantees the current directory for all '`.mm`' scripts to be the directory MirrorManager resides in. And usually the sub-directory '`rexx/`' contains these '`.mm`' scripts.
2. An important part of your configuration script makes up finding a legal host. The following example demonstrates how to do this poperly:

```
/*
**  Finding a MirrorManager host
*/

portbase = 'MIRRORMANAGER.'
portlist = SHOW('P',,'0A'X)

OPTIONS RESULTS
DO WHILE WORDS(portlist) > 0
  PARSE VAR portlist portname '0A'X portlist
  IF COMPARE(portname,portbase) = LENGTH(portbase)+1 THEN DO
    ADDRESS(portname); 'NUMENTRIES'
    IF result = 0 THEN portlist= ""
                  ELSE ADDRESS
    END
  END

IF ADDRESS() ~= portname THEN DO
  SAY 'MirrorManager is not running... exiting...'
  EXIT
  END

/* ... */

EXIT
```

3. ARexx filenames containing spaces are currently somewhat difficult to execute. They are no problem for MirrorManager but they seem to be for several other hosts. Even `RX` needs a double quoted ToolType value for `CMD`, for example

   ```
   CMD="my script.rexx"
   ```

   or a kludge for executing '`my script.rexx`' from within the Shell:

   ```
   RX "call 'my example.rexx'"
   ```

4. If your configuration script should be executable from outside the GUI then you *must* set your own name to the application via the `CONFIGNAME` command. Otherwise `CONFIGNAME` will either be *unset* and calling `SAVE` without a parameter would fail, or if `CONFIGNAME` was set then `SAVE` would overwrite the old configuration.

   In fact it is not easy to determine the own full pathname from within an ARexx script, so here is an example

   ```
   PARSE SOURCE . . s

   t= LEFT(s,LASTPOS(':',s))
   called= STRIP( LEFT(t,LASTPOS(' ',t)) )

   CALL PRAGMA('W','N')
   DO WHILE ~EXISTS(called) & LASTPOS(' ',called) > 0
     called= LEFT(called,LASTPOS(' ',called)-1)
     END

   IF LEFT(cr,1) = 'R' THEN DO
     host= ADDRESS()
     PARSE VAR s (called) s (host) .

     resolved= STRIP(s)

     CALL PRAGMA('W','N')
     DO WHILE ~EXISTS(resolved)
       resolved= LEFT(resolved,LASTPOS(' ',resolved)-1)
       END

     END

   /* ... */
   ```

   At the bottom of this example, i.e. after the `/* ... */` the variable `called` contains the filename used for calling this script while `resolved` contains the expanded path including the root's volume name.

   An example procedure `whoami` can also be found in the '`rexx/`' drawer coming with MirrorManager. See Section 7.1 [MirrorManager scripts], page 13, '`whoami.rexx`' for more information.

5. You should always test the result of a call. All of the GUI's ARexx commands return 0 in the `rc` variable if they have been successful. A value `~= 0` *always* indicates an error. See Section 8.2 [MUI Error codes], page 18, for details on error codes.

6. Strings passed to the GUI always have to be quoted twice if they contain any white space. Additionally, double quotes '`"`' and asterisks '`*`' in a string passed to the GUI have to be

escaped with an asterisk '*' to '*"' and '**' respectively. For this reason it is often useful to implement a function that quotes a string for you.

For example,

```
/* translate '"' into '*"' and '*' into '**' */

transquote: PROCEDURE
  PARSE ARG s
  t= s
  q= MAX( LASTPOS('*',s), LASTPOS('"',s) )

  DO WHILE q > 0
    t= INSERT('*',t,q-1,1)
    s= LEFT(s,q-1)
    q= MAX( LASTPOS('*',s), LASTPOS('"',s) )
    END

  RETURN '"' || t || '"'
```

double quotes the given argument string and translates each occurence of a double quote '"' into '*"' and each occurence of '*' into '**'. Here is an example for the usage of `transquote` with the `ADD` command:

```
ADDRESS 'MIRRORMANAGER.1'

ADD transquote('Say hello') transquote('"SAY ''Hello world!''"')
```

8.5.2 Writing GUI Independent ARexx Scripts for MirrorManager

The '`.rexx`' script files supplied with MirrorManager are all GUI independent, i.e. they never address to the MirrorManager's GUI host. This guarantees functionality from within any environment.

The following is recommended to this kind of MirrorManager scripts:

1. The template for command line arguments sould be available in a `ReadArgs()` style. At most any user will expect this kind of argument passing meanwhile. The following example demonstrates a convenient argument parsing for a template `REQUIRED/K/A,OPTIONAL/K,SWITCH/S`

```
/*
**  parse command line arguments
*/

required = ""
optional = ""
switches = ""

IF ( ARG() < 1 ) | ( (ARG() = 1) & ARG(1)= '?' ) then do
  OPTIONS PROMPT "REQUIRED/A,OPTIONAL,BOOL=SWITCH/S: "
  PARSE PULL ARGS
  END
```

```
ELSE PARSE ARG args

DO WHILE WORDS(args) > 0
  av= next_arg()
  SELECT
    WHEN UPPER(av) = "REQUIRED" then do
      IF WORDS(args) > 0 THEN required= next_arg()
      ELSE DO
        SAY "missing argument value after REQUIRED keyword"
        EXIT usage()+5
        END
      END

    WHEN UPPER(av) = "OPTIONAL" THEN DO
      IF WORDS(args) > 0 THEN optional= next_arg()
      ELSE DO
        SAY "missing arument value after OPTIONAL keyword"
        EXIT usage()+5
        END
      END

    WHEN (UPPER(av) = "BOOL") | (upper(av) = "SWITCH") then do
      switches = switches || 's'
      END

    OTHERWISE DO
      IF av ~= '?' THEN SAY "Unknown keyword" av
      EXIT usage()+5
      END

  END /* select */

END /* do */
IF WORDS(required) < 1 then do
  SAY "required argument missing"
  EXIT usage()+5
  END

/* ... */

IF LASTPOS('s',switches) > 0 THEN DO
   /* what this switch stands for */
   END

/* ... */

next_arg: PROCEDURE EXPOSE args
  args= STRIP(args)
  IF LEFT(args,1) = '"' THEN PARSE VAR args '"' a '"' args
                        ELSE PARSE VAR args     a     args
  RETURN STRIP(a,'b','"');
```

9 Disclaimer

9.1 Warranty? No warranty.

There is no warranty for this software package. Although the authors have tried to prevent errors, they cannot guarantee that the software package described in this document is 100% reliable. You are therefore using this material at your own risk. The authors cannot be made responsible for any damage which is caused by using this software package.

10 License

10.1 Copyright

MirrorManager is (c)Copyright 1994 by Tobias Ferber and Harald Kunze


```
Tobias Ferber                   Harald Kunze
Goethestrasse 32                Nuitsstrasse 29
76135 Karlsruhe                 76185 Karlsruhe
ukjg@rz.uni-karlsruhe.de        uklg@rz.uni-karlsruhe.de
```


```
                  @ @
              ---oOO-(_)-OOo---
```

MirrorManager is ShareWare.

THIS IS A BETA VERSION !!!

Thanks to all beta testers, especially to Rene Petri, Mark Rose and Tobias Walter.

10.1.1 Support ShareWare!

Please feel free to fill out the registration form! It can be found in your '`MirrorManager/docs`' drawer.

10.2 Distribution

This software package is freely distributable under the concept of ShareWare. It may be put on any media which is used for the distribution of free software, like Public Domain disk collections, CDROMs, FTP servers or bulletin board systems.

In order to ensure the integrity of this software package, distributors should use the original archive file. The authors cannot be made responsible if this software package has become unusable due to modifications of the archive contents or of the archive file itself.

There is no limit on the costs of the distribution, e.g. for the media, like floppy disks, streamer tapes or compact disks, or the process of duplicating. Such limits have been proven to be harmful to the idea of freely distributable software, e.g. instead of reducing the price of the floppy disk below the limit, the software was simply removed from the master disk.

10.3 Usage Restrictions

No program, document, data file or source code from this software package, neither in whole nor in part, may be used on any machine which is used

- for the research, development, construction, testing or production of weapons or other military applications. This also includes any machine which is used in the education for any of the above mentioned purposes.
- by people who accept, support or use violence against other people, e.g. citizens from foreign countries.

11 MUI - MagicUserInterface

`MirrorManager` uses

MUI - MagicUserInterface

MUI is a system to generate and maintain graphical user interfaces. With the aid of a preferences program, the user of an application has the ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing lots of examples and more information about registration please look for a file called '`muiXXusr.lha`' ('`XX`' means the latest version number) on your local bulletin boards or on public domain disks.

If you want to register directly, feel free to send DM 20.– or US$ 15.– to

Stefan Stuntz
Eduard-Spranger-Straße 7
80935 München
GERMANY

12 Installer

Along with MirrorManager comes the 'Installer' from Commodore:

> Installer and Installer project icon
>
>
> INSTALLER SOFTWARE IS PROVIDED "AS-IS" AND SUBJECT TO CHANGE;
> NO WARRANTIES ARE MADE. ALL USE IS AT YOUR OWN RISK. NO LIABILITY
> OR RESPONSIBILITY IS ASSUMED.

Installer project icon redesigned by Martin Huttenloher. See Chapter 13 [MagicWB], page 34, for details.

13 MagicWB

All icons coming with MirrorManager are designed, redesigned or inspired by Martin Huttenloher's MagicWB.

Martin Huttenloher
Am Hochstraess 4
D-89081 Ulm
Germany/Europe

ARexx Commands Index

A

ACTIVATE 20
ADD 18
APPSTART 21

B

BOTTOM 20

C

CLEAR 18
CLONE 19
COMPLETE 21, 22
CONFIGNAME 21

D

DOWN 20

E

EXECUTE 20

H

HELP 17
HIDE 16

I

INFO 16

L

LOAD 20
LOCK 23

M

MESSAGE 21

N

NUMENTRIES 18

Q

QUIT 16

R

REMOVE 19
RENAME 19
REQUESTCHOICE 22
REQUESTFILE 23

S

SAVE 20
SAVELOG 23
SHOW 16
SORT 18

T

TOP 20

U

UP 20

W

WORKING 22

ARexx Scripts Index

C

CutTree.mm 14
CutTree.rexx 14

D

Demo.rexx 15

E

ExamineIndex.mm 14
ExamineIndex.rexx 14

M

MakeIndex.mm 14
MakeIndex.rexx 14
MakeTree.mm 13, 14
MakeTree.rexx 13, 14
MirrorManager.rexx 15

S

SortIndex.mm 14
SplitIndex.mm 14
SplitIndex.rexx 14

W

whoami.rexx 15

Master Index

.

.rexx ... 13

A

About ... 10
Add ... 11
Add FileNotes from RECENT ... 8
Add FileNotes from WANTED ... 8
Add FileNotes to INCOMING ... 8
Aminet ... 1
Aminet directory hierarchy ... 13
APPSTART ... 5, 21
ARexx ... 13, 16
ARexx code calling method ... 23
ARexx Commands ... 16
ARexx Scripts ... 13
Argument parsing ... 27
Arguments ... 4
Arguments (to the '.rexx' scripts) ... 13
AUTHOR ... 16

B

BASE ... 16
BODY ... 22
Bottom ... 11

C

called ... 25
Calling Method ... 23
Check the local mirror ... 14
Cleanup INCOMING ... 8
Cleanup with new RECENT ... 8
Clear ... 11
Clear Window ... 12
ClickMeForColors ... 34
Clone ... 11
Coding hints ... 25
Command line option parsing ... 27
CONFIGICON ... 4
CONFIGNAME ... 6, 21, 25
Configuration script ... 15
Configuration scripts ... 3, 25
Configure ... 4
Configuring ... 3
CONSOLE ... 5
COPYRIGHT ... 16
Create FAST Index ... 8
Create/Update Aminet TREE ... 8
Create/Update LOCAL Index ... 8

D

Default tool ... 5
Delete empty Directories ... 14
Delete Empty Directories ... 8
Demo.rexx ... 15
DESCRIPTION ... 16
Directory hierarchy ... 13
Disclaimer ... 29
Distribution ... 30
Double quoting ... 25
Down ... 11

E

Edit Aminet INDEX File ... 9
Edit Configuration Script ... 9
Edit LOCAL Index File ... 8
Edit Menu ... 11
Edit WANTED Index ... 9
Edit ... 11
Enter ... 20
Error codes ... 18
Examine the local index file ... 14
Example code ... 25

F

Fast index file ... 14
Finding a MirrorManager host ... 25

G

GADGETS ... 22
GUI dependent scripts ... 25

H

Harald Kunze ... 30
Hello world! ... 25
Hints ... 25

hostname 25
Hostname 16

I

Icons 34
Inataller 2
Independent ARexx scripts 27
Initial ListView contents 8
`Insert` 11
Installation 2
Installer 33
Installer program 33
Introduction 1

L

License 30
`Load` 10
`Locked` 10

M

MagicUserInterface 32
MagicWB 34
Martin Huttenloher 34
MAXARGLEN 6
MAXCOMMANDLEN 6
MAXITEMLEN 6
MAXPATHLEN 6
Menu items 10
Menu structure 10
MUI 32
MUI Error codes 18

N

No Warranty 29
`NO_ICON_POSITION` 4

O

`Open Window` 12
Options 4
Order form 30
`Output` Menu 12

P

`PARSE SOURCE` 15
Portname 16
`Procject` Menu 10
Project icon 4, 5

Q

`Quit` 10
Quoting 25

R

RCommand-/ 18
RCommand-A 18
RCommand-B 20
RCommand-C 19
RCommand-D 20
RCommand-E 18, 19
RCommand-O 20
RCommand-Q 16
RCommand-S 18
RCommand-T 20
RCommand-U 20
RCommand-W 20
RCommand-X 19
`Re-Insert Kicked Out Files` 9
ReadArgs() 13, 27
Registring MirrorManager 30
`Remove` 11
`Reorganize Local Mirror` 9
resolved 25
Return 20
RX 13

S

`Save As` 10
`Save Log` 12
`Save` 10
SCREEN 16
SetConfigureDefaults 4
ShareWare 30
`Sort` 11
`Sort Aminet INDEX File` 9
Sorting an index file 14
Split index file 14
Stack size 4
Stefan Stuntz 32

T

Templates 13, 27
TITLE 16
`TITLE` 22
Tobias Ferber 30
ToolTypes 4

`Top` 11
transquote 25
`TREE` 13
TREE 14

U

`Up` 11
`Update Aminet DirNotes` 8
Using the ListView 8

V

VERSION 16

W

Warranty 29
whoami 25

Table of Contents

1 Introduction 1

2 Installing MirrorManager 2

3 Configuring MirrorManager 3

3.1 MirrorManager Configuration Scripts 3
3.2 Using `Configure` 4
3.3 ToolTypes and Command Line Arguments 4

4 Creating an Aminet Hierarchy 7

5 Description of the ListView Items 8

6 Menu Items and Menu Structure 10

6.1 The `Project` Menu 10
6.2 The `Edit` Menu 11
6.3 The `Output` Menu 12

7 ARexx Scripts 13

7.1 MirrorManager ARexx scripts 13

8 ARexx Commands 16

8.1 MUI built-in ARexx commands 16
8.2 MUI Error codes 18
8.3 `MirrorManager` ARexx commands 18
8.4 How MirrorManager executes ARexx Code 23
8.4.1 ARexx inline code 23
8.4.2 ARexx script files 23
8.5 Hints for writing new ARexx script files for MirrorManager 25
8.5.1 Writing GUI-dependent ARexx scripts 25
8.5.2 Writing GUI Independent ARexx Scripts for MirrorManager 27

9 Disclaimer 29

9.1 Warranty? No warranty. 29

10 License 30

10.1 Copyright 30
10.1.1 Support ShareWare! 30

10.2 Distribution 30
10.3 Usage Restrictions 31

11 MUI - MagicUserInterface 32

12 Installer 33

13 MagicWB 34

ARexx Commands Index 35

ARexx Scripts Index 36

Master Index 37