MirrorManager

COLLABORATORS			
	TITLE : MirrorManager		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		November 2, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 MirrorManager **1**

1.1 MirrorManager.guide . . . 1
1.2 MirrorManager.guide/Introduction . . . 2
1.3 MirrorManager.guide/Installation . . . 3
1.4 MirrorManager.guide/Configuring . . . 4
1.5 MirrorManager.guide/MirrorManager Configuration Scripts . . . 4
1.6 MirrorManager.guide/Using Configure . . . 5
1.7 MirrorManager.guide/ToolTypes and Command Line Arguments . . . 6
1.8 MirrorManager.guide/Creating a Mirror . . . 8
1.9 MirrorManager.guide/Using the ListView . . . 8
1.10 MirrorManager.guide/Menu Items . . . 10
1.11 MirrorManager.guide/ARexx Scripts . . . 13
1.12 MirrorManager.guide/MirrorManager Scripts . . . 14
1.13 MirrorManager.guide/ARexx Commands . . . 17
1.14 MirrorManager.guide/MUI Built-In ARexx Commands . . . 17
1.15 MirrorManager.guide/MUI Error Codes . . . 19
1.16 MirrorManager.guide/MirrorManager ARexx Commands . . . 20
1.17 MirrorManager.guide/Calling Method . . . 26
1.18 MirrorManager.guide/Hints . . . 27
1.19 MirrorManager.guide/GUI-Dependent . . . 28
1.20 MirrorManager.guide/GUI-Independent . . . 30
1.21 MirrorManager.guide/Additional Tools . . . 32
1.22 MirrorManager.guide/AMM . . . 32
1.23 MirrorManager.guide/Installing AMM . . . 33
1.24 MirrorManager.guide/Options & Switches . . . 33
1.25 MirrorManager.guide/Configuring AMM . . . 37
1.26 MirrorManager.guide/Hints & Tips . . . 38
1.27 MirrorManager.guide/ALs . . . 39
1.28 MirrorManager.guide/Ex . . . 41
1.29 MirrorManager.guide/Politics . . . 41

1.30 MirrorManager.guide/Disclaimer 42
1.31 MirrorManager.guide/Copyright 42
1.32 MirrorManager.guide/Distribution 42
1.33 MirrorManager.guide/Usage Restrictions 43
1.34 MirrorManager.guide/MUI 43
1.35 MirrorManager.guide/MagicWB 44
1.36 MirrorManager.guide/Installer 45
1.37 MirrorManager.guide/ARexx Index 45
1.38 MirrorManager.guide/Master Index 48

Chapter 1

MirrorManager

1.1 MirrorManager.guide

```
                This document describes MirrorManager version 1.10, a Shareware ↩
                   Aminet
mirror management system for the Amiga.

Copyright (C) 1994 by Tobias Ferber and Harald Kunze

WARNING:  THIS MANUAL IS STILL UNDER DEVELOPMENT.  NOT ALL OF THE
AVAILABLE FEATURES OF MIRRORMANAGER ARE DESCRIBED HERE.  THERE MIGHT BE
ALSO OBSOLETE STUFF IN THIS MANUAL!

Since you received this file together with AMM you might be interested
especially in
                this
                section.  (^:



                Introduction
                              What (and why) is MirrorManager?

                Installation
                              How to install us

                Configuring
                               Customizing MirrorManager

Taking off with MirrorManager


                Creating a Mirror
                         Guidelines for the initial installment

                Using the ListView
                        Descrtiption of the listview items

                Menu Items
                                MirrorManager's menu structure
```

Controling MirrorManager

ARexx Scripts
Using (and writing new) ARexx script files

ARexx Commands
Description of GUI built-in ARexx commands

Additional Tools
External tools & utilities

Distribution Politics

Politics
Warranty, Copyright, License...

Appendix

ARexx Index
List of scriptfiles and commands

Master Index
Where can I find information about...?

1.2 MirrorManager.guide/Introduction

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 or RECENT
     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
                ARexx Commands
                .
See
                ARexx Scripts
                , for information about the supplied ARexx script
files.

You also need to set-up several paths and filenames in order to use
MirrorManager. See
                Installation
                ,
                Configuring
                 for details.

1.3 MirrorManager.guide/Installation

                Installing MirrorManager
****

MirrorManager is distributed together with the Installer program from
Commodore. See
                Installer
                , 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 assignments 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
                Configuring
                , for more information.

     Note: MirrorManager needs MUI.  See
                MUI
                , for information about how
     to obtain it.  You should also have the actual Aminet INDEX and
     TREE file to take off with MirrorManager.
```

1.4 MirrorManager.guide/Configuring

```
                Configuring MirrorManager
************************

MirrorManager comes with the MirrorManager MUI application and several
ARexx scripts configuring the GUI.  There are however also ARexx
scripts which can be used from within the Shell and which do not need
the MUI application.

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.



                 MirrorManager Configuration Scripts
                     What are they good for?

                 Using Configure
                                          The Installer script "Configure"

                 ToolTypes and Command Line Arguments
                    Controling MirrorManager on Startup
```

1.5 MirrorManager.guide/MirrorManager Configuration Scripts

```
                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 by
MirrorManager 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
                MirrorManager ARexx Commands
                , 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 most of the 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
                Using Configure
                , 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
                MirrorManager Scripts
                , MirrorManager.rexx for details.

1.6 MirrorManager.guide/Using Configure

                Using The Configure Program
==========================

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 set 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
                MirrorManager Scripts
                , for details.

     Caution: If any path or 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)

---------- Footnotes ----------

(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!

1.7 MirrorManager.guide/ToolTypes and Command Line Arguments

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 ToolType 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 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 ToolType CONFIGICON 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. (1)

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

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!

---------- Footnotes ----------

(1)  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.
```

1.8 MirrorManager.guide/Creating a Mirror

```
                Creating an Aminet Hierarchy
****************************

This chapter explains how to create your first local Aminet mirror.
Please make sure that MirrorManager is installed and configured before
you go on reading here.  See
                Installation
                ,
                Configuring
                 for details.

The first thing you need is an Aminet TREE file.  This file is
available (e.g. via ftp) from any Aminet site in info/adt/tree.  You
can of course also use the ALs tool to create such a file from an
existing Aminet mirror.  See
                ALs
                , for details.

To be written...
```

1.9 MirrorManager.guide/Using the ListView

```
                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
Configuring
, 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.
```

1.10 MirrorManager.guide/Menu Items

```
Menu Items and Menu Structure
*****************************

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.  I.e. these .mm scripts
     configure 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 MirrorManager.

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

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

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

     See also: ARexx command CLONE

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

     See also: ARexx command REMOVE

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

The Output Menu
===============

Open Window
     This will open the 'Working Window', the window into which
     MirrorManager's ARexx scripts 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
```

1.11 MirrorManager.guide/ARexx Scripts

```
                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 Space 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 by the MUI application.

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



                MirrorManager Scripts
                   Verbose documentation on all supplied script files

                Calling Method
                          How MirrorManager executes ARexx code

                Hints
                                  Hints for writing new scripts for  ↩
                                     MirrorManager
```

1.12 MirrorManager.guide/MirrorManager Scripts

```
                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

SplitIndex splits given FROM index file into several smaller files combining those which begin with the same character. With the TO parameter you can specify a 'stem' for the filenames; the first character which they have in comon will be appended to it as a suffix.

The CleanupIncoming script supports these kind of index files and the benefit in performance compared to non-splitted index files is great!

MakeIndex FROM/K/A,TO/A

MakeIndex 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

In general, ExamineIndex compares your two index files. (e.g. your LOCAL index and an original Aminet INDEX file.)

blabla...

> Hacker's Note: A great speed improvement can be made by using

```
          the agrep command instead of Commodore's Search command.
          Please take a closer look at the procedure searchcmd in
          ExamineIndex.mm and/or ExamineIndex.rexx respectively and
          replace the return 'Search ... line by the following:
               return 'agrep > "'tempinfo'"' '"'pattern'"' '"'file'"'

SortIndex FROM/K/A,FILE/S,DIR/S,QUICK/S,AUTO/S
     The SortIndex script allows you to sort a given FROM file by
     either filename or directory name.

CleanupIncoming FROM/K/A,TO/K,WITH=INDEX/K/A,MOVE/S,COPY/S, NOCOMMENT/S,MAKEPATH/S ↩
   ,REPLACE/S,FAST/S,REMAP/K
     CleanupIncoming 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 the MakeTree
     script.

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

          Hacker's Note: A great speed improvement can be made by using
          the agrep command instead of Commodore's Search command.
          Please take a closer look at the procedure searchcmd in
          CleanupIncoming.mm and/or CleanupIncoming.rexx respectively
          and replace the return 'Search ... line by the following:
               return 'agrep > "'tempinfo'"' '"'pattern'"' '"'file'"'

MirrorManager.rexx
     The file 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
                Hints
                , 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
                Using Configure
                , for
     further details.

Demo.rexx
```

```
     The Demo.rexx script 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.
```

1.13 MirrorManager.guide/ARexx Commands

```
                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
                MUI
                )



                 MUI Built-In ARexx Commands

                 MUI Error Codes

                 MirrorManager ARexx Commands
                If you want to write your own ARexx script files for MirrorManager ↩
                    you
should also read
                Hints
                .
```

1.14 MirrorManager.guide/MUI Built-In ARexx Commands

```
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
```

1.15 MirrorManager.guide/MUI Error Codes

```
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.
```

1.16 MirrorManager.guide/MirrorManager ARexx Commands

```
                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

Calling Method

, 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 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 it 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 using

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 ...

---------- Footnotes ----------

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

1.17 MirrorManager.guide/Calling Method

```
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.

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.

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 his own.
```

1.18 MirrorManager.guide/Hints

```
                Hints For Writing New ARexx Scripts 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.
```

GUI-Dependent

GUI-Independent

1.19 MirrorManager.guide/GUI-Dependent

```
                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
   MirrorManager Scripts
   , 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

```
              MUI Error Codes
              , 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!''"')
```

1.20 MirrorManager.guide/GUI-Independent

```
Writing GUI-Independent ARexx Scripts
-------------------------------------

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. Almost 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','"');
```

1.21 MirrorManager.guide/Additional Tools

```
                Additional Tools
****************
```

MirrorManager comes with a load of extras which obviously can be found in the xtras/ drawer. In this chapter we'll give you a brief overview of the tools which are available there and explain what they can do for you.

```
AMM
  AminetMirrorManager (adt file parser)

ALs
  Create (ADT) index files from your local mirror

Ex
   Executing shell scripts from a pipe
```

1.22 MirrorManager.guide/AMM

```
                AminetMirrorManager (AMM)
=========================
```

The AMM tool is a very flexible command which collects files and looks them up in index files of type adt-v1 or adt-v2. For each match (or mismatch), AMM can expand some macros in a given scriptfile, so that executing AMM output can perform various tasks.

```
Installing AMM
       How to install AMM and the provided scripts

Options & Switches
   How to tell AMM what to do

Configuring AMM
      How to write AMM scripts for your mirror
```

```
                Hints & Tips
                        Non-intuitive :) operations
```

1.23 MirrorManager.guide/Installing AMM

```
                Installing AMM
--------------

To install AMM simply copy the executable amm.000 (or amm.030 if you
have an MC-68030 Amiga) somewhere into your path (e.g. to C:) and
rename it to amm.  For example,
     Copy CLONE amm.000 TO c:amm

Next you should get an Aminet index file of type adt-v1 or adt-v2.
These can be obtained (e.g. via ftp) from all Aminet sites in the
info/adt/ directory.  If you're using MUIAdt then you will
automatically get the RECENT.adt file each time MUIAdt connects to an
Aminet site.  See
                Hints & Tips
                , for further details.

Now you need some .amm scripts.  These are text files which are
expanded by AMM.  Here is a very simple example:
     "%p%n" lives in "%d"

This (or a similar) example script called print.amm is included in this
distribution, so AMM is now ready for a first example take-off.

Example: Suppose you've downloaded some files from the Aminet into a
directory incoming: and you've also downloaded the RECENT.adt file into
this directory.  Now try out:
     amm -f incoming:RECENT.adt -y print.amm incoming:

AMM will now collect all files in your incoming: directory and for each
file there, which also exists in the index RECENT.adt it will print a
line of the form:
     "incoming:foo.bar" lives in "which/witch"

Note that you don't really need to uncompress the .Z or .gz files to
disk since AMM can read the index file from the standard input stream.
For example,
     zcat ADT_AMINET.gz | amm -y print incoming:

does exactly the same as the example above but with a gzip'ed index
file of type adt-v2.
```

1.24 MirrorManager.guide/Options & Switches

Arguments, Options & Switches

Before AMM starts reading the index file, it collects all items in the command line. AMM distinguishes between, tree types of arguments:

filenames
: Ordinaly filenames are inserted into AMM's internal search tree for a later lookup in the index file.

directories
: AMM scans directories recursively, collects all files and inserts them into it's internal search tree.

options and switches
: These items always begin with one - or two dashes - and give you a finer control over AMM's behaviour.

The files collected by AMM -- no matter whether they appear in a file list or in the command line -- are always checked for existence, and non-existent files or directories are simply ignored. However, if the -x switched is given in the command line, AMM will also process non-existent files. Here is a list of all options and switches:

@ filename: Read a list of files
- or -stdin: Read a file list from stdin
: The @ option allows you to pass a list of files to AMM. This is important since on the one hand the length of a command line is limited on most systems and on the other hand AMM does not know anything about patterns or regular expressions. In general, a command like List or ls gives you much more power over the process of file gathering. For example,

```
List >ram:foobar FILES PAT ~(#?.info) LFORMAT "%p%n" DIR incoming:
```

will non-recursively collect all files which are not icons from your incoming: directory into a file ram:foobar.

A special version of the @ option is - or -stdin. This forces AMM to read the file list from the standard input stream which e.g. allows piping the file list to AMM like that:

```
List FILES LFORMAT "%p%n" | amm -f ADT_AMINET -y touch
```

The number of file lists passed to AMM is not limited but only depends on the available free store. However you can pass the -stdin option only once of course.

> Caution: If -stdin is used then the index file must be either specified via the -f option or the environment variable ENV:ADTFILE must be set to a valid index file. AMM will of course not be able to read the index file from the standard input stream as well in this case.

Note also that the name 'file list' might lead to the conclusion that only files can be part of AMM's file lists. This is however not correct. The file list must contain exactly one unquoted path or filename per line; empty lines are ignored. Directories are expanded recursively.

```
-y or -yes-file filename: Script for indexed files
-n or -no-file filename: Script for non-indexed files
     With these two options you can tell AMM what to do.  At least one
     of them must be present in the command line.

     If invoked with -y filename, AMM will read the file filename, and
     for each file in the input which also exists in the index file,
     AMM will expand the %-macros in filename and write the result
     to the standard output stream or the file specified with the -o
     option, respectively.  The same applies to -n filename, however
     this file will be expanded and written for those files which are
     not listed in the index file.

          Note: AMM will automatically append a suffix .amm to the
          given filename if the plain file filaname does not exist.  If
          still unsuccessful, AMM will look up the value of the
          environment variable ENV:AMMPATH and search the directory it
          is set to for both filename and filename.amm.  For example,
               SetEnv AMMPATH s:amm/

          can be used to foce AMM to look into the s:amm/ directory if
          the given script file does not exist in your current
          directory.  Use the following command to make it resident:
               Echo NOLINE "s:amm/" >ENVARC:AMMPATH

     The following %-macros can be used in the script files:
    %n
          The name of the file without a path.  This string will be
          taken from the index file if %n appears in a file loaded with
          the -y option to ensure correct capitalization.

    %p
          The pathname of the input file, i.e. the location where the
          local file resides on your disk.  (e.g. incoming:).  The
          replacement for %p will always have a trailing slash / or
          colon :.

    %d
          The directory of the file in the Aminet including a trailing
          slash /.  This is something like util/misc/ or biz/demo/.
          This macro is of course not expanded in files loaded via -n.

    %c
          The comment of the file in the Aminet with all double quotes
          " replaced with *" and single asterisks * replaced with
          **.  This allows something like
               FileNote QUIET FILE "%p%n" COMMENT "%c"

          This macro is not expanded in files loaded via -n.

    %D, %T
          Respectively date and time of the upload.  These strings are
          valid for the SetDate command and can be used for example
          like that:
               SetDate FILE "%p%n" DATE "%D" TIME "%T"
```

```
          These macros are not expanded in files loaded via -n.

     %%
          Always replaced with a single percent character %.

     Other macros (such as %x or %y) actually expand to themselves.  It
     is however dangerous to rely on that because they might do not in
     future versions of AMM.  If you really need something like %x then
     you should better write %%x instead.

-f  or -index-file filename: Specify an ADT Aminet index file
     With this option you can specify the name of the Aminet index
     file.  If omitted, AMM will read the index from the standard input
     stream unless and environment variable ADTINDEX is set to a valid
     filename.  For example,
          set ADTINDEX=incoming:ADT_RECENT

     will force AMM to read the index from incoming:ADT_RECENT instead
     of from stdin.  If this file is non-existent and -f is omitted then
     AMM will fail.  You can use the pseudo filename - to force AMM to
     read from stdin even if ADTINDEX is set.

-o  or -output-file filename: Name of the output file
     AMM will normally write all output to the standard output stream.
     With this option however you can force AMM to write into the file
     filname instead.  This can be useful if your environment does not
     support redirectioning of stdout.  For example,
          amm -y touch incoming: -o ram:doit

     and
          amm -y touch incoming: >ram:doit
     are absolutely equivalent.

-h or -help: Print short usage information and exit
     When invoked with this option, AMM displays a short usage
     information and exits.  You may also use -h1 to see only the
     "short" options (those with only one leading dash -) or -h2 to see
     only the "long" options (those with two leading dashes -).

-r or -no-readmes: Do not handle *.readme files
     By default, AMM will find the *.readme files in the index even if
     they are not explicitly mentioned there (as it is the default in
     adt-v2 type index files).  However, when invoked with this option,
     AMM will only find those files which really exist in the index
     file and ignore the .readmes completely.

     Let me explain what AMM does, if the -r option is omitted: The
     adt-v1 index file format also holds the names of the *.readme
     files, so AMM will use these to expand the %n of a .readme file if
     you feed such an index.  Otherwise, if the index file is of type
     adt-v2 then AMM will cut off the original suffix and append
     .readme instead.  All the %-macros of the original file are
     copied to those of the .readme file but only the %n macro has the
     new suffix .readme.

-v or -verbose: Print warnings and progress information
     This switch instructs AMM to print some information to the
```

```
     standard error stream.  By default, AMM will behave absolutely
     silent and only print error messages.  Note that -v should be the
     first option!

-x  or -ignore-exist: Lookup also non-existent files
     By default, AMM will only insert existent files into it's search
     tree.  Non-existent files passed to AMM are simply ignored.  With
     this option you can force AMM to process also those files which do
     not really exist on your disk.  This can be useful when working
     with file lists.
          Caution: If a non-existent pathname with a trailing slash /
          or colon : is passed to AMM and the -x switch is present,
          then AMM will insert an empty file "" with the given
          pathname.  In this case the result of a search in the index
          file is unpredictable!

     Note also that since AMM traverses the command line from left to
     right, the -x switch has infuence on only those files or pathnames
     which are mentioned right of it.  In general it is a good idea to
     use -x as the very first switch, if needed.
```

1.25 MirrorManager.guide/Configuring AMM

```
                Configuring AMM For Your Mirror
-------------------------------

As explained in the description of the options -y and -n, AMM expands
macros for indexed and/or non-indexed files respectively.  See

                Options & Switches
                , for more information.  We shall now explain some
example scripts to help you getting some experience in writing such
scripts for your system.

One of the most important script is touch.amm.  It sets the file date
and comment according to the values in the index file.  This is very
useful, especially if you're using a tool like DOpus to examine the
downloaded files.  With touch.amm you can get the filenote in the DOpus
listview as you are used to when downloading via adt.
     ; $VER: $ld: touch.amm,v 1.3 1995/07/26 03:08:45 tf Exp $
     ; (c)Copyright 1995 by Tobias Ferber,  All Rights Reserved

     If EXISTS "%p%n"
       Echo "%d%n (%c)"
       SetDate FILE "%p%n" DATE "%D" TIME "%T"
       FileNote QUIET FILE "%p%n" COMMENT "%c"
     EndIf

We need the If EXISTS ... EndIf to avoid errors because -- especially
when invoked with the -x switch -- AMM will also process files which
don't really exist on your disk.

Another very useful script is cleanup.amm which is quite similar to
touch.amm, but additionally moves the files to your local Aminet
```

```
mirror.  The following example assumes your mirror in downloads:aminet/.

     ; $VER: $ld: cleanup.amm 1.4 1995/07/26 03:08:41 tf Exp $
     ; (c)Copyright 1995 by Tobias Ferber,  All Rights Reserved

     If EXISTS "%p%n"
       SetDate FILE "%p%n" DATE "%D" TIME "%T"
       FileNote QUIET FILE "%p%n" COMMENT "%c"
       Copy QUIET CLONE FROM "%p%n" TO "downloads:aminet/%d%n"
       Delete QUIET FILE "%p%n"
     EndIf

Warning: The Delete is of course dangerous and might kill your original
file if you run amm -y cleanup directly on your mirror directory
downloads:aminet/!

One possible enhancement would be using Kai Iske's Move command instead
of Copy and Delete.  For example,
     Move QUIET CLONE FROM "%p%n" TO "downloads:aminet/%d%n"
```

1.26 MirrorManager.guide/Hints & Tips

```
                Hints & Tips For Efficient AMM Utilization
-----------------------------------------

This section is supposed to give you some deeper insight into AMM's
work and to show you some special solutions for problems which you
wouldn't have if you wouldn't use AMM. :)

Catching the index file from MUIAdt
...................................

I most often use MUIAdt to download files from the Aminet.  MUIAdt
automatically downloads the info/adt/RECENT.adt.Z or
info/adt/SHORT.adt.Z file respectively, however it deletes it
immediately when exiting so you most likely forget copying the file to
a safe location.  One way to solve this problem is via WaitNotify; for
example,
     Copy `WaitNotify FMT=%s RAM:RECENT.adt RAM:SHORT.adt` incoming:

This will save a copy of the first index file which appears in your Ram
Disk: into your incoming: directory.  It's a good idea to put the
following lines into your AmiTCP/bin/startnet script:
     Echo >T:grab-adt "Copy *`WaitNotify FMT=%s RAM:RECENT.adt*` incoming:"
     Run <NIL: >NIL: Execute T:grab-adt

Removing From Your Mirror What Is Available On CD
.................................................

Many of you who use AMM might also have the Aminet5: CD.  Now why
should you keep all those files on your harddisk if you've got them on
a CD?  So I thought and home I went writing the following script which
(re)moves the those files from my downloads:aminet/ drawer.  Actually
```

```
I'm not that keen on using the Delete command immediately, and so I
decided to make a new drawer downloads:aminet5/ and move the mentioned
files out of my mirror into it using the following cd5.amm script:
     If EXISTS "%p%n"
       Echo "%d%n (%c)"
       Move QUIET CLONE FROM "%p%n" TO "downloads:aminet5/%n"
     EndIf

Again I used Kai Iske's excellent Move command instead of Copy and
Delete.  To execute this script you simply have to type the following:
     amm -y cd5 downloads:aminet/ -f AMINET5:Aminet/info/adt/ADT_AMINET |ex

The ex command is explained in one of the following sections.  See
                Ex
                ,
for details.
```

1.27 MirrorManager.guide/ALs

```
ALs - Create Aminet Index Files
===============================

The ALs command creates index files in various formats from your local
mirror directory.  To be more precise: ALs collects all files in a
given path (or in the current directory, if omitted) and writes those
which are nested two subdirectories deep in form of an index file to
the standard output stream.(1)

ALs will preferably take the desctiption text from the .readme file's
Short: line however, if there's no .readme file available it will use
the filenote instead.

Installation
------------

Installing ALs is very easy: Simply copy als.000 (or als.030 if you
have an MC-68030 Amiga) somewhere into your path (e.g. to C:) and
rename it to als.  For example,
     Copy CLONE als.000 TO c:als

Invoking ALs
------------

By default, ALs creates has an 80 columns wide SHORT index file as
output and reads from the current directory.  You can however control
ALs via the following options.  (Note that options must be specified
before the path name)

-f1, -f2: Create machine readable ADT index files
     With these options, ALs writes index files of type adt-v1 or
     adt-v2, respectively.  Actually, the name of the *.readme
     files in the adt-v1 index is created for all files by changing the
     original suffix to .readme.  The size of them is always set to 0,
```

```
     because actually all *.readme files are simply ignored.  This
     might change in future versions.

-fs, -fl: Create human readable index files
     These options set ALs's output format to those known from the
     SHORT and LONG index files, respectively.  By default, ALs
     uses -fs. The main difference is that the file comment in the
     SHORT index file is truncated to 42 characters, whereas it is
     up to 80 characters wide in the LONG index file, which
     additionally provides an Age column with the age of each file in
     days.  However, 20 characters are reserved for the filename in
     SHORT index files, whereas there are only 18 characters reserved
     for the filename in the LONG index file format.

-ft: Create an Aminet tree file
     With this option, ALs prints a tree file for the given directory.
     The desctiption text will be preferably taken from the first line
     of a file .message however if it doesn't exist, ALs will use the
     directory's filenote.

-fg, -fh: Create AmigaGuide/HTML index files (experimental!)
     These Options are still under development.  They work, but the
     result is not very nice yet.

-d directory: Make given path a certain directory
     By default, ALs only prints files which are nested two directories
     deeper than the given path.  With this option you can tell ALs
     that when scanning the given path, it has already entered the
     given directory.  For example:
          als -d util downloads:aminet/util

     will print the index for the util/ directory only.  Of course you
     can also print the contents of your incoming: directory via als -d
     foo/bar incoming:.

- dayrange
     This option allows you to restrict the files included into the
     output to be at most dayrange days old.  For example
          als -14 >RECENT

     creates a file RECENT which contains all files of the last two
     weeks.

Example: Suppose you have an Aminet mirror in your downloads:aminet/
directory and now you want to create a file RECENT.adt.Z to allow other
people to connect to your Amiga via MUIAdt.  Here is what you have to
type:
     als -f1 -14 downloads:aminet | compress >downloads:aminet/info/adt/RECENT.adt ↩
         .Z

---------- Footnotes ----------

(1)  Note that ALs additionally hides both: files and directories
beginning with a period . and *.readme files
```

1.28 MirrorManager.guide/Ex

```
Ex - Execute Shell Scripts From A Pipe
======================================

The Ex command is an enhancement of C:Execute.  The advantages of Ex
over C:Execute are:
   -     Ex allows multiple file names of shell scripts to execute in
     the command line instead of only one

   -     Ex has a QUIET/S switch to suppress console output of any of
     the commands in the given script files.

   -     Ex can execute a shell script from the pipe (!)

   -     Ex can execute shell scripts asynchronously (i.e. Run them).

Installation
------------

Installing Ex is very easy: Simply copy ex.000 (or ex.030 if you have an
MC-68030 Amiga) somewhere into your path (e.g. to C:) and rename it to
ex.  For example,
     Copy CLONE ex.000 TO c:ex

That's all! :)
```

1.29 MirrorManager.guide/Politics

```
                Distribution Politics
**********************


License Agreement - Terms and Conditions


                 Disclaimer
                                  Warranty?  No warranty.

                 Copyright
                                   Who did it?

                 Distribution

                 Usage Restrictions
                External packages which are either supplied, needed or supported  ↩
                   by MirrorManager


                 MUI
                                             Stuntzi's MUI
```

```
MagicWB
                        How can I get more of those beautiful icons?

Installer
                      Commodore's installer program
```

1.30 MirrorManager.guide/Disclaimer

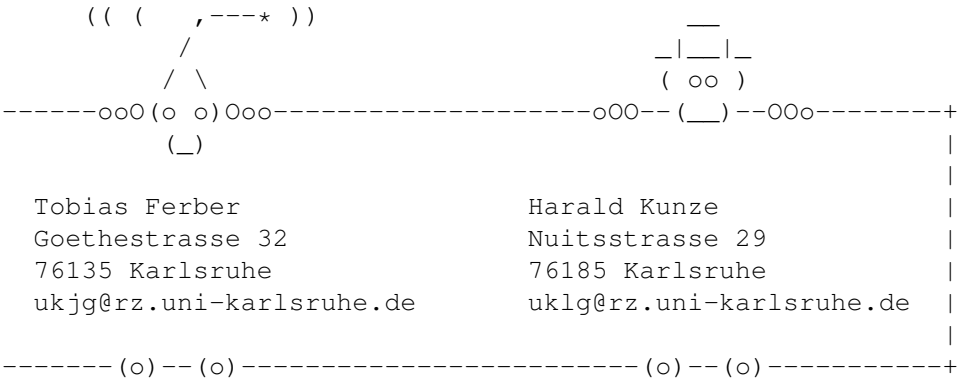
```
Disclaimer: Warranty? No warranty.
==================================
```

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.

1.31 MirrorManager.guide/Copyright

```
Copyright
=========
```

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



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

Please Support ShareWare! A registration form can be found in your MirrorManager/docs drawer.

1.32 MirrorManager.guide/Distribution

Distribution Conditions
=======================

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.

> Developing MirrorManager is and was hard, unpayed work! Nevertheless we decided to distribute MirrorManager uncrippled under the concept of Shareware. To register, please fill out the file OrderForm which can be found in your MirrorManager/docs/ drawer. Registered users will be shipped a disk with the newest release of MirrorManager, along with a personalized GUI program and some additional scriptfiles.
>
> Thank you for supporting Shareware!

1.33 MirrorManager.guide/Usage Restrictions

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.

1.34 MirrorManager.guide/MUI

MUI - MagicUserInterface
========================

MirrorManager uses

MUI - MagicUserInterface

(C) Copyright 1993/94 by Stefan Stuntz

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 30.- or US$
20.- to

Stefan Stuntz

Eduard-Spranger-Straße 7

80935 München

GERMANY

1.35 MirrorManager.guide/MagicWB

MagicWB
=======

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

MagicWB has been designed and published by Martin Huttenloher under the
concepts of SHAREWARE and is Copyright (C) 1993 by Martin Huttenloher,
All rights reserved.

Martin Huttenloher

Am Hochstraess 4

D-89081 Ulm

Germany/Europe

1.36 MirrorManager.guide/Installer

```
                Installer
=========

Along with MirrorManager comes the Installer from Commodore:

     Installer and Installer project icon
     (c) Copyright 1991-93 Commodore-Amiga, Inc.  All Rights Reserved.
     Reproduced and distributed under license from Commodore.

     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
                MagicWB
                ,
for details.
```

1.37 MirrorManager.guide/ARexx Index

```
                ARexx Index
***********

ARexx Commands
==============



                 ACTIVATE
                                          MirrorManager ARexx Commands

                 ADD
                                               MirrorManager ARexx Commands

                 APPSTART
                                          MirrorManager ARexx Commands

                 BOTTOM
                                            MirrorManager ARexx Commands

                 CLEAR
                                             MirrorManager ARexx Commands

                 CLONE
                                             MirrorManager ARexx Commands

                 COMPLETE
                                          MirrorManager ARexx Commands
```

CONFIGNAME
MirrorManager ARexx Commands

DOWN
MirrorManager ARexx Commands

EXECUTE
MirrorManager ARexx Commands

HELP
MUI Built-In ARexx Commands

HIDE
MUI Built-In ARexx Commands

INFO
MUI Built-In ARexx Commands

INFO AUTHOR
MUI Built-In ARexx Commands

INFO BASE
MUI Built-In ARexx Commands

INFO COPYRIGHT
MUI Built-In ARexx Commands

INFO DESCRIPTION
MUI Built-In ARexx Commands

INFO SCREEN
MUI Built-In ARexx Commands

INFO TITLE
MUI Built-In ARexx Commands

INFO VERSION
MUI Built-In ARexx Commands

LOAD
MirrorManager ARexx Commands

LOCK
MirrorManager ARexx Commands

MESSAGE
MirrorManager ARexx Commands

MESSAGE CLEAR
MirrorManager ARexx Commands

MESSAGE CLOSE
MirrorManager ARexx Commands

MESSAGE OPEN
MirrorManager ARexx Commands

NUMENTRIES
MirrorManager ARexx Commands

QUIT
MUI Built-In ARexx Commands

REMOVE
MirrorManager ARexx Commands

RENAME
MirrorManager ARexx Commands

REQUESTCHOICE
MirrorManager ARexx Commands

REQUESTCHOICE BODY
MirrorManager ARexx Commands

REQUESTCHOICE GADGETS
MirrorManager ARexx Commands

REQUESTCHOICE TITLE
MirrorManager ARexx Commands

REQUESTFILE
MirrorManager ARexx Commands

SAVE
MirrorManager ARexx Commands

SAVELOG
MirrorManager ARexx Commands

SHOW
MUI Built-In ARexx Commands

SORT
MirrorManager ARexx Commands

TOP
MirrorManager ARexx Commands

UP
MirrorManager ARexx Commands

WORKING
MirrorManager ARexx Commands

ARexx Scripts

CleanupIncoming.mm
MirrorManager Scripts

```
CleanupIncoming.rexx
                MirrorManager Scripts

CutTree.mm
                           MirrorManager Scripts

CutTree.rexx
                         MirrorManager Scripts

Demo.rexx
                            MirrorManager Scripts

ExamineIndex.mm
                       MirrorManager Scripts

ExamineIndex.rexx
                     MirrorManager Scripts

MakeIndex.mm
                         MirrorManager Scripts

MakeIndex.rexx
                       MirrorManager Scripts

MakeTree.mm
                           MirrorManager Scripts

MakeTree.rexx
                         MirrorManager Scripts

MirrorManager.rexx
                    MirrorManager Scripts

SortIndex.mm
                         MirrorManager Scripts

SplitIndex.mm
                         MirrorManager Scripts

SplitIndex.rexx
                       MirrorManager Scripts

whoami.rexx
                           MirrorManager Scripts
```

1.38 MirrorManager.guide/Master Index

```
                Master Index
*************

                _
                                                Options & Switches
```

-help
Options & Switches

-ignore-exist
Options & Switches

-index-file
Options & Switches

-no-file
Options & Switches

-no-readmes
Options & Switches

-output-file
Options & Switches

-stdin
Options & Switches

-verbose
Options & Switches

-yes-file
Options & Switches

-<number>
ALs

-d
ALs

-f
Options & Switches

-f1
ALs

-f2
ALs

-fg
ALs

-fh
ALs

-fl
ALs

-fs
ALs

-ft
ALs

-h
Options & Switches

-n
Options & Switches

-o
Options & Switches

-r
Options & Switches

-v
Options & Switches

-x
Options & Switches

-y
Options & Switches

.amm
Options & Switches

.mm
ARexx Scripts

.rexx
ARexx Scripts

@
Options & Switches

- UNNAMED -, Listview item
Menu Items

.rexx
ARexx Scripts

About..., Project Menu
Menu Items

Add FileNotes from RECENT
Using the ListView

Add FileNotes from WANTED
Using the ListView

Add FileNotes to INCOMING
Using the ListView

Add, Edit Menu
Menu Items

adt-v1,-v2, Index format
AMM

APPSTART, ToolType
ToolTypes and Command Line Arguments

Bottom, Edit Menu
Menu Items

called
GUI-Dependent

Cleanup INCOMING
Using the ListView

Cleanup with new RECENT
Using the ListView

Clear Window, Output Menu
Menu Items

Clear, Edit Menu
Menu Items

Clone, Edit Menu
Menu Items

CONFIGICON, ToolType
ToolTypes and Command Line Arguments

CONFIGNAME
GUI-Dependent

CONFIGNAME, ToolType
ToolTypes and Command Line Arguments

CONSOLE, ToolType
ToolTypes and Command Line Arguments

Create FAST Index
Using the ListView

Create/Update Aminet TREE
Using the ListView

Create/Update LOCAL Index
Using the ListView

Delete Empty Directories
Using the ListView

Down, Edit Menu
Menu Items

Edit Aminet INDEX File
Using the ListView

Edit Configuration Script
Using the ListView

Edit LOCAL Index File
Using the ListView

Edit WANTED Index
Using the ListView

Edit..., Edit Menu
Menu Items

Edit Menu
Menu Items

Esc-Sequence, in MUI texts
MirrorManager ARexx Commands

hostname
GUI-Dependent

INDEX
MirrorManager Scripts

Insert, Edit Menu
Menu Items

Installer, Configure script
Using Configure

Load..., Project Menu
Menu Items

Locked, Project Menu
Menu Items

MAXARGLEN, ToolType
ToolTypes and Command Line Arguments

MAXCOMMANDLEN, ToolType
ToolTypes and Command Line Arguments

MAXITEMLEN, ToolType
ToolTypes and Command Line Arguments

MAXPATHLEN, ToolType
ToolTypes and Command Line Arguments

NO_ICON_POSITION
ToolTypes and Command Line Arguments

Open Window, Output Menu
Menu Items

Output Menu
Menu Items

PARSE SOURCE
MirrorManager Scripts

Procject Menu
Menu Items

Quit, Project Menu
Menu Items

Re-Insert Kicked Out Files
Using the ListView

RECENT
MirrorManager Scripts

Remove, Edit Menu
Menu Items

Reorganize Local Mirror
Using the ListView

resolved
GUI-Dependent

Save As..., Project Menu
Menu Items

Save Log..., Output Menu
Menu Items

Save, Project Menu
Menu Items

Sort Aminet INDEX File
Using the ListView

Sort, Edit Menu
Menu Items

Top, Edit Menu
Menu Items

TREE
MirrorManager Scripts

Update Aminet DirNotes
Using the ListView

Up, Edit Menu
Menu Items

Abstract
Introduction

adt
Configuring AMM

ADTFILE
Options & Switches

ADT_AMINET
Installing AMM

ADT_RECENT
Installing AMM

agrep
MirrorManager Scripts

agrep
MirrorManager Scripts

ALs
ALs

Aminet
Introduction

Aminet directory hierarchy
MirrorManager Scripts

Aminet Index, Create
ALs

Aminet5 CD
Hints & Tips

AminetMirrorManager
AMM

AmiTCP:bin/startnet
Hints & Tips

AMM
AMM

AMMPATH
Options & Switches

Application is Locked
Menu Items

APPSTART
MirrorManager ARexx Commands

ARexx
ARexx Scripts

ARexx
ARexx Commands

ARexx code calling method
Calling Method

ARexx Commands
MirrorManager ARexx Commands

ARexx Commands
ARexx Commands

ARexx Scripts
ARexx Scripts

Argument parsing
GUI-Independent

Arguments
Options & Switches

Arguments
ToolTypes and Command Line Arguments

Arguments (to the .rexx scripts)
ARexx Scripts

Asynchronous Execution
Ex

Becker, Stefan
ARexx Scripts

C:Execute
Ex

Calling Method
Calling Method

Check the local mirror
MirrorManager Scripts

cleanup.amm
Configuring AMM

ClickMeForColors
MagicWB

Coding hints
Hints

Command line option parsing
GUI-Independent

Conditions, For distribution
Distribution

CONFIGNAME
MirrorManager ARexx Commands

Configuration
ToolTypes and Command Line Arguments

Configuration script
MirrorManager Scripts

Configuration scripts
MirrorManager Configuration Scripts

Configuration scripts
GUI-Dependent

Configuration, Loading a
Menu Items

Configuration, Save
Menu Items

Configuration, Save
Menu Items

Configure
Using Configure

Configuring
Configuring

Copyright
Copyright

Default configuration
MirrorManager Scripts

Default listview
Using the ListView

Default tool
ToolTypes and Command Line Arguments

Delete empty Directories
MirrorManager Scripts

Demo script
MirrorManager Scripts

Directory hierarchy
MirrorManager Scripts

DirectoryOpus
Configuring AMM

Disclaimer
Disclaimer

Distribution
Distribution

Distribution
Disclaimer

DOpus
Configuring AMM

Double quoting
GUI-Dependent

Enter
MirrorManager ARexx Commands

ENV:ADTFILE
Options & Switches

ENV:AMMPATH
Options & Switches

Error codes, From MUI
MUI Error Codes

ex
Hints & Tips

Ex
Ex

Examine the local index file
MirrorManager Scripts

Example code
Hints

Execute
Ex

Executing ARexx code
Calling Method

Fast index file
MirrorManager Scripts

Ferber, Tobias
Copyright

FileNote
Options & Switches

Finding a MirrorManager host
GUI-Dependent

GUI dependent scripts
GUI-Dependent

Hacker’s Note
MirrorManager Scripts

Hacker’s Note
MirrorManager Scripts

Hello world!
GUI-Dependent

Hierarchy, Aminet
Introduction

Hints, for AMM
Hints & Tips

Hints, for coding
Hints

Hostname
ARexx Commands

Huttenloher, Martin
MagicWB

Icon, Project
ToolTypes and Command Line Arguments

Icon, Tool
ToolTypes and Command Line Arguments

Icons
MagicWB

Inataller
Installation

Independent ARexx scripts
GUI-Independent

Index format adt-v1,-v2
AMM

Index, Create
ALs

Initial listview contents
Using the ListView

Installation
Installation

Installer
Installer

Installer program
Installer

Introduction
Introduction

Iske, Kai
Hints & Tips

Iske, Kai
Configuring AMM

Items, In Menus
Menu Items

Kunze, Harald
Copyright

License
Copyright

List
Options & Switches

Listview items, Add
Menu Items

Listview items, Clone
Menu Items

Listview items, default
Using the ListView

Listview items, Edit
Menu Items

Listview items, Insert
Menu Items

Listview items, Moving
Menu Items

Listview items, Moving
Menu Items

Listview items, Moving
Menu Items

Listview items, Moving
Menu Items

Listview items, Remove
Menu Items

Listview, Clear
Menu Items

Listview, Sort
Menu Items

Load configuration
Menu Items

ls
Options & Switches

MagicUserInterface
MUI

MagicWB
MagicWB

MEmacs
MirrorManager Configuration Scripts

Menu items
Menu Items

Move
Configuring AMM

Move
Hints & Tips

Moving listview items
Menu Items

Moving listview items
Menu Items

Moving listview items
Menu Items

Moving listview items
Menu Items

MUI
MUI

MUI Error codes
MUI Error Codes

MUI Text, Esc-Sequences
MirrorManager ARexx Commands

MUIAdt
Hints & Tips

MUIAdt
Installing AMM

No Warranty
Disclaimer

Non-GUI Scripts
GUI-Independent

Options
ToolTypes and Command Line Arguments

Options
Options & Switches

Order form
Copyright

Petri, Rene
Copyright

Pipe, Executing scripts from a
Ex

Portname
ARexx Commands

print.amm
Installing AMM

Project icon
ToolTypes and Command Line Arguments

Project icon
ToolTypes and Command Line Arguments

Quoting
GUI-Dependent

RCommand-/
Menu Items

RCommand-/
MirrorManager ARexx Commands

RCommand-?
Menu Items

RCommand-A
MirrorManager ARexx Commands

RCommand-A
Menu Items

RCommand-B
MirrorManager ARexx Commands

RCommand-B
Menu Items

RCommand-C
MirrorManager ARexx Commands

RCommand-C
Menu Items

RCommand-D
Menu Items

RCommand-D
MirrorManager ARexx Commands

RCommand-E
MirrorManager ARexx Commands

RCommand-E
MirrorManager ARexx Commands

RCommand-E
Menu Items

RCommand-I
Menu Items

RCommand-K
Menu Items

RCommand-L
Menu Items

RCommand-M
Menu Items

RCommand-O
MirrorManager ARexx Commands

RCommand-O
Menu Items

RCommand-Q
Menu Items

RCommand-Q
MUI Built-In ARexx Commands

RCommand-S
MirrorManager ARexx Commands

RCommand-S
Menu Items

RCommand-T
Menu Items

RCommand-T
MirrorManager ARexx Commands

RCommand-U
MirrorManager ARexx Commands

RCommand-U
Menu Items

RCommand-W
MirrorManager ARexx Commands

RCommand-W
Menu Items

RCommand-X
MirrorManager ARexx Commands

RCommand-X
Menu Items

ReadArgs()
GUI-Independent

ReadArgs()
ARexx Scripts

Registring MirrorManager
Copyright

Resolving own filename
MirrorManager Scripts

Restrictions, In usage
Usage Restrictions

Return
MirrorManager ARexx Commands

Rose, Mark
Copyright

Run Execute
Ex

RX
ARexx Scripts

Save configuration
Menu Items

Save configuration
Menu Items

Scripts, for AMM
Configuring AMM

Scripts, To configure
MirrorManager Configuration Scripts

SetConfigureDefaults
Using Configure

SetDate
Options & Switches

ShareWare
Copyright

Shell Scripts
Ex

Sorting an index file
MirrorManager Scripts

Split index file
MirrorManager Scripts

Stack size
ToolTypes and Command Line Arguments

startnet
Hints & Tips

Structure, of the menu
Menu Items

Stuntz, Stefan
MUI

Switches
ToolTypes and Command Line Arguments

Switches
Options & Switches

Templates
ARexx Scripts

Templates
GUI-Independent

Tips, for AMM
Hints & Tips

ToolManager
ARexx Scripts

Tools
Additional Tools

ToolTypes
ToolTypes and Command Line Arguments

touch.amm
Configuring AMM

transquote
GUI-Dependent

Transquoted FileNote
Options & Switches

uncompress
Installing AMM

Using the listview
Using the ListView

WaitNotify
Hints & Tips

Walter, Tobias
Copyright
Warranty
Disclaimer
whoami
GUI-Dependent
Working window
Menu Items
Working window
Menu Items
Working window
Menu Items
zcat
Installing AMM