ReqChange

Magnus Holmgren

COLLABORATORS			
	TITLE : ReqChange		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Magnus Holmgren	October 27, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ReqChange

1.1 ReqChange.guide: User documentation for ReqChange

  This file describes ReqChange, version 3.5, a program that replaces the
most common requesters with the ones ReqTools (copyright © Nico François)
offers. ReqChange is similar to RTPatch, included in the ReqTools
distribution, but offers a lot more.

 Introduction

  Preface   Please read this first.
  Introduction   Why ReqChange?
  System requirements What you need to run ReqChange.

 Legal issues

  License   What you may and may not do with ReqChange.
  No warranty   Use ReqChange at your own risk.
  Disclaimer   Software disclaimer.

  Shareware   Shareware information.

 Documentation for ReqChange

  Installation   How to install ReqChange.

  Patched requesters   Which requesters ReqChange can patch.

  The AssignWedge   What is an AssignWedge?
  ARexx messages   How to send ARexx messages.
  Excepted programs   Except programs from patching.

  The prefs editor   How the prefs editor works.
  StartRC   The starter program.

  Acknowledgements   Who did what?
  The future   Possible new features in ReqChange.
  Program history   What have changed in ReqChange.

  Glossary   Explanation of some "strange" words.

1.2   ReqChange.guide/Preface

Preface

   ReqChange is offered to you under the concept of Shareware. You may use
ReqChange for an evaluation period of up to 30 days without paying any
charge. If you are going to use ReqChange after the evaluation period, you
have to register.

   See License and Shareware for more information.

1.3   ReqChange.guide/Introduction

Introduction

   One of the many new features with Amiga OS 2.0, is a new library called
Asl, which contains a standard file and a font requester (in OS 2.1 or
higher there is also a screenmode requester). Before OS 2.0, there were no
standard requesters of this kind, which lead several programmers to write
their own version(s) of a file/font requester. Some programmers put them in
a library – often with other requesters and/or functions – so that other
programs could use them as well. Of these external requesters (or rahter,
libraries), there were basically two that became really popular, Arp and
Req.

   Arp was the first one, and that library also contains a number of useful
function that the Amiga OS didn't have before version 2.0. Req on the other
hand have a couple of extra requesters and requester-related functions, and
the combined file and font requester is rather powerful (although maybe not
the most beautiful one).

   Anyway, the thought is that all programs should use the new standard Asl
requesters, to provide a nice looking and uniform environment for the user.
However, I think these new requesters have a couple of drawbacks. For one
thing, the file requester is slow (although this is improved in OS 2.1), and
doesn't look that good (IMHO). The information requester (which also is
improved in OS 2.0) is glued to the upper left corner of the screen, and can
only be confirmed with the mouse, or some cryptic key-combination (although
there is a thought behind that).

   Another problem is that there are a lot of older programs that still use
Arp and Req. These requesters simply doesn't fit that well into the new OS
2.0 environment. Something had to be done about this. But what?

   One day I got a copy of ReqTools, a library written by Nico François.
This library contains a set of requesters that looks good, and are
generally faster and better than the other possible alternatives (again,
IMHO). When I saw these requesters, I thought:

   "Why can't all programs use these requesters instead?"

So, I decided to do the best I could so that as many programs as possible
would use the ReqTools requesters instead, regardless if they previously had
used Asl, Arp or Req.

The result of this effort is this program, ReqChange, which replaces most
requesters in Intuition, Asl, Arp and Req with the equivalents found in
ReqTools. It also offers a couple of extra features.

Note: ReqChange only replaces the requester functions in a library, not
the entire library. You still need the library in question.

1.4 ReqChange.guide/System requirements

System requirements

ReqChange should work with any Amiga that has at least OS 2.04 and 512
Kbyte of RAM.

ReqTools is ofcourse required, but it is included in the ReqChange
distribution, so this shouldn't cause any problems.

When active, ReqChange uses about 20 Kbyte of memory, exluding the memory
used by ReqTools, and the diskbased libraries ReqChange may patch. Any so
called Rexx, deny or exception lists also takes some memory, ofcourse (see
ARexx messages, The AssignWedge and Excepted programs).

The libraries are only patched when the library in question is in use, a
feature that can save quite a bit of memory. This feature was first
implemented in RTPatch by Nico Francois.

A program called SetMan (Copyright (C) 1991,1992 by Nicola Salmoria) can
be nice to have installed, but it is by no means required. SetMan is a
program that extends the way patches are handled within the system, and can
prevent system crashes with improperly written "patchers" (ReqChange is not
"improperly written" :). SetMan also provides a list over all patches in the
system, and which program that installed them, and a possibility to
disable (and re-enable) a patch.

1.5 ReqChange.guide/License

License

· This license applies to the product called ReqChange, a set of programs
for the Amiga computer, published by Magnus Holmgren under the concepts of
Shareware, and the accompanying documentation, example files and anything
else that comes with the original distribution. The terms "Programs" and
"ReqChange" below, refer to this product. The licensee is addressed as
"you".

· You may copy and distribute verbatim copies of the programs' executable
code and documentation as you receive it, in any medium, provided that you

conspicuously and appropriately publish only the original, unmodified
programs, with all copyright notices and disclaimers of warranty intact and
including all the accompanying documentation, example files and anything
else that came with the original.

· You may not copy and/or distribute these programs without the
accompanying documentation and other additional files that came with the
original. You may not copy and/or distribute modified versions of these
programs.

· You may not copy, modify, sublicense, distribute or transfer the
programs except as expressly provided under this license. Any attempt
otherwise to copy, modify, sublicense, distribute or transfer the programs
is void, and will automatically terminate your rights to use the programs
under this license. However, parties who have received copies, or rights to
use copies, from you under this license will not have their licenses
terminated so long as such parties remain in full compliance.

· By copying, distributing and/or using the programs you indicate your
acceptance of this license to do so, and all its terms and conditions.

· Each time you redistribute the programs, the recipient automatically
receives a license from the original licensor to copy, distribute and/or
use the programs subject to these terms and conditions. You may not impose
any further restrictions on the recipients' exercise of the rights granted
herein.

· You may not disassemble, decompile, re-source or otherwise reverse
engineer the programs.

· You may use the programs for a period of up to 30 days for evaluation.
After that, you have to register.

· If you wish to incorporate parts of the programs into other programs,
write to the author to ask for permission.

· You agree to cease distributing the programs and data involved if
requested to do so by the author.

· You may charge a fee to recover distribution costs. The fee for
diskette distribution may not be more than the cost to obtain a public
domain diskette from Fred Fish.

1.6 ReqChange.guide/No warranty

No warranty

THERE IS NO WARRANTY FOR THE PROGRAMS, TO THE EXTENT PERMITTED BY
APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDER AND/OR OTHER PARTIES PROVIDE THE PROGRAMS "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAMS
IS WITH YOU. SHOULD THE PROGRAMS PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY REDISTRIBUTE THE
PROGRAMS AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAMS (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAMS TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

1.7 ReqChange.guide/Disclaimer

Disclaimer

   No warranty, either express or implied, is made with respect to the
fitness or merchantability of ReqChange.

   Magnus Holgren (referred to as "the author"), reserve the right to not
develop any future versions of ReqChange.

   The author will try to make a good faith attempt at correcting any
problems if any are discovered, but are in no way required, nor bound to
correct them.

   The author neither assume nor accept any responsibility for the use or
misuse of these programs. He also will not be held liable for damages or
any compensation due to loss of profit or any other damages arising out of
the use, or inability to use these programs.

   Magnus Holmgren will not be liable for any damage arising from the
failure of these programs to perform as described, or any destruction of
other programs or data residing on a system attempting to run the programs.
While I know of no damaging errors, the user of these programs uses it at
his or her own risk.

1.8 ReqChange.guide/Shareware

Shareware

   ReqChange is shareware. This means that if you use this program regularly,
you are required to send me, the author, a Shareware fee as payment for the
program. This fee is US$10 (70 SKR). Fill in the registration form (the
file Registration.form), and send it with the money to:

   Magnus Holmgren
   Kvarnbergsvägen 5
   S-444 47 Stenungsund
   SWEDEN

   Or send the money to the postal giro account 921 71 60-2 (in Sweden).
Please, only send SKR, as the exchange fee is rather high. A recommended

way to send the money is via the so called "International money order".
Your local post office should be able to help you with this.

   You can also e-mail me the registration form. In this case, send it to:

   "Magnus Holmgren", 2:204/404.6@fidonet.org

   or, if you prefer Internet:

   cmh@augs.se

   If you send me $15 (90 SKR) or include a disk + return postage (e.g. one
international reply cupon), I'll send you the next version when it is
finished. Additional updates costs $5 (20 SKR) each (you can also send me a
disk, SAE + return postage, and I'll do it for free! :). This is only if
you want me to send you the next update as soon as it is ready. Otherwise
you don't need to pay me anything to get an update, if you e.g. download it
from a local BBS.

   Since this program is released as "true" Shareware (i.e. the program
isn't crippled in any way) you might think you won't gain any by
registering this program. In a way that is true, since you won't get any
new features by registering (instead you can really test the program to see
if you think it is worth registering). Instead you encourage me to continue
to improve this program, and add new features you would like to see. Also,
if you have an e-mail address (fidonet or Internet) I'll mail you update
information shortly before an update is released. New features requests
from registered users will ofcourse have a much higher priority than those
from others! :)

   I've put many hours into this program to make it as good as I only could.
I would sure appreciate to get something for all this work. Please support
Shareware!

   Note: The alternative to "true" Shareware would be to add keyfile
protection, so that all features can't be used from the start. It would
also increase the Shareware fee, to cover the extra costs (to send out the
keyfiles). You don't want that, do you? If your anser is "no", then send me
the Shareware fee right away! (Assuming you use this program ofcourse :)

1.9 ReqChange.guide/Installation

Installation

   ReqChange is fairly easy to install. Simply use the supplied installer
script (which uses the Commodore Installer utility). If you don't have the
Installer (included in OS 2.1 or higher, and several commercial and non-
commercial programs. It is also available on most BBS:es), you need to do
it by hand. This isn't that hard to do, if you have some familiarity with
the Amiga.

   First of all, copy Libs/reqchange.library and Libs/reqtools.library to
Libs:. If you already have reqtools.library installed, it is a good idea to
check the version already installed, and only copy the supplied version if
it is newer (use Version to find out which version you have). The included

version is always the latest one at the time of the release.

   The prefs editor (Prefs/ReqChange) should go to your Sys:Prefs drawer, or
any other drawer you find convenient (preferably somewhere in your command
search path).

   If you have AmigaGuide® installed on your Amiga, you may want to be able
to use online-help in the prefs editor. Then you should copy ReqChange.guide
(either the english or the swedish version; which ever you prefer :) to
either the same directory as the prefs editor, or to any directory in the
AmigaGuide® search path (as set by the AmigaGuide/Path environment
variable).

   In order to start/stop ReqChange, the small program StartRC is needed.
Install it anywhere in your path where you find it convenient, or drop it in
your Sys:WBStartup drawer, for automatic start every boot.

   If you don't put StartRC in Sys:WBStartup, a line like this in your
S:User-Startup file will start ReqChange each boot:

   StartRC

   If any errors occurs, you will be informed about it. See StartRC for more
information.

   Lastly, if you have OS 2.1 or higher, and would like to operate ReqChange
in another language than english, you need to install at a couple of
catalog files. Copy the files in the drawer Catalogs/<language> (were
"<language>" is the language is the language as set in the Locale
preferences editor) to Locale:Catalogs/<language>. There are four files for
each language, two for ReqChange and two for ReqTools (currently, there
aren't files for all languages).

1.10 ReqChange.guide/Patched requesters

Patched requesters

   ReqChange patches a number of different requester functions, found in a
few libraries. All these patches are optional, and will only be installed
if you have activated them, and the library in question actually is in
memory. In order to accomplish this, one (or rather, two ;) function in
exec.library need to be patched. This patch have been written to be
completely transparant and shouldn't cause any problems.

   These are the different libraries ReqChange may patch:

   Intuition
   Asl
   Arp
   Req

1.11 ReqChange.guide/Patched requesters/Intuition

Intuition

   Intuition has two requesters, called AutoRequest() and EasyRequest().
These requesters are very similar to the user; they only differ in the way
the are called. These requesters are used for displaying information (e.g.
an about requester) or simple queries, where there are a few, predefined,
answers.

   The requesters ReqChange installs behaves almost like the old ones, but
with a little extra: It also contains an AssignWedge (see The AssignWedge).
This AssignWedge can be turned off if you want (see
The prefs editor/Main window 49).

   Some programs (like DiskSalv version 2) uses the AutoRequest()er in a way
that ReqChange can't emulate properly. When such requesters are used,
ReqChange will use the original one instead. ARexx messages will still be
sent, but with a minor catch: There will be linefeeds here and there in the
body that normally shouldn't be there.

   Note: AutoRequest() and EasyRequest() also exist in asyncronous versions
(which allows the program to continue processing while the requester is up).
These functions can't be patched, since that calling interface can't be
emulated with the current version of ReqTools.

1.12   ReqChange.guide/Patched requesters/Asl

Asl

   Asl currently contains three requesters. These are a file, a font and a
screen mode requester (only in OS 2.1 or higher). If the Asl patches are
installed, then the font and the screen mode requester patches are optional.

   Some of the features in the Asl font and screen mode requesters doesn't
exist in the ReqTools requesters ^{1}. When any of these features are used
the patch will automatically use the original requester instead. Thus, you
can have the "Asl font requester" and "Asl screen mode requester" patch
flags on without loosing any functionality (see
The prefs editor/Main window).

   Currently it isn't possible to select "no font" (i.e. clear the font name
gadget and select ok) in the font requester, as one can do in the Asl font
requester. Because of this, a special font called " no font" is included
(the leading space is there to try to get the font at the top of the list).
It is a very small font, and if that one is selected, ReqChange will
interpret it as if no font at all was selected. Maybe not the most elegant
solution, but it works! :) If a new version of ReqTools is released that
supports the "no font" selection somehow, ReqChange will be updated to use
this feature instead.

   -- Footnotes --

   ^{1} The ReqTools font requester doesn't allow you to select color(s) and
draw mode for the text, and the screen mode requester can't show a custom

```
screen mode list
```

1.13   ReqChange.guide/Patched requesters/Arp

```
Arp
```

  There is only one requester in Arp, and that is a file requester. Not all
features of this requester can be emulated, but since these features are
very seldom used (in my experience at least. I've heard of one program that
use any of these features), this patch will not fall back to the original
when needed. If this is a problem for you, you can always send me a note, or
use the exception feature.

  Note: If you have a 68030 CPU (or better), there is a function in Arp that
doesn't work that well, and that is ArpExit(). It seems like earlier
versions of ReqChange somehow managed to make ArpExit() to work (at least on
my 68040 Amiga), if a patched requester was called just before ArpExit().
Since this isn't the case any more (and I haven't the foggiest idea why! :)
ReqChange will replace the ArpExit() function with one that should work
better ^{1}. This should also make programs like e.g. Move (one of the Arp
commands) to work better.

  -- Footnotes --

 ^{1} Technical note: it is basically the original function with a call to
CacheClearU() added at a suitable place.

1.14   ReqChange.guide/Patched requesters/Req

```
Req
```

  This is the big one. Req contains most of the requester ReqTools does.
Thus, rather many functions are patched. Here is a list of all requesters
patched:

```
  TextRequest()
```

  This is a text requester, similar to AutoRequest()/EasyRequest() in
Intuition (see Patched requesters/Intuition). All features are supported,
including the timeout and the waitmask.

```
  ReqFileRequest()
```

  This is a combined file and font requester. There is only one thing one
need to think about when using this one: The original requester have a
(configurable) limit on the number of files one can multiselect (if this is
enabled). ReqTools doesn't have any such limitation, so you may select as
many as you can/like. ReqChange will then only use as many as the limitation
says, and will ignore the rest of the files. This isn't really a problem,
but it can useful to know.

GetString()

A simple string requester.

GetLong()

An integer requester.

ColorRequester()

A palette requester.

1.15 ReqChange.guide/The AssignWedge

The AssignWedge

Some of you may have seen the program AssignWedge (or maybe AssignX, the
original program). This program improves the (in)famous "Please insert
volume" requester. What AssignWedge does is to add a few gadgets to the
normal "Retry" and "Cancel" gadgets, so that you can do something about it,
without having to use a Shell or something like that. ReqChange have an
AssignWedge like this builtin (but a somewhat better version).

The following gadgets are added to the "Please insert volume" requester:

Mount

This gadget will try to mount the volume in question. This requires
C:Mount to be available. If there is no suitable entry in your
Devs:MountList file or in the drawer Sys:Storage/DosDrivers ^{1},
this will ofcourse fail. Useful for seldom used units like Rad: or Pc0:.
If the mount fails, the requester will re-appear.

Assign...

This gadget shows a file requester, where you can select a drawer or a
file that the volume name should be assigned to. If the assign fails (maybe
you entered a file/drawer that doesn't exist), the requester will re-appear.

Note: In the requester there is a filename field. This is needed, since an
assign can refer to a file, not only a drawer.

Deny

This will make ReqChange remember the volume name on a list (the so called
deny list) and the next time a requester for this volume is about to appear,
ReqChange will automatically cancel the requester before it is shown.

This deny list can be edited and saved, using the Edit window and the
Deny window in The prefs editor.

-- Footnotes --

^{1} This only applies if you have OS 2.1 or better.

1.16 ReqChange.guide/ARexx messages

ARexx messages

  ReqChange have one powerful feature, and it is the ability to send an
ARexx message when a certain requester appears. This allows you to e.g. add
sound effects to your requesters, like ARQ does. But you can add a lot more
if you like. One thing could be to start e.g DiskSalv each time a requester
indicating a disk is corrupt is shown. Or it could be to try to reboot the
computer if a "Task held" requester appears. Anything that can be done via
ARexx.

  Earlier versions of ReqChange could only send ARexx messages like this
for the text requesters, much like ARQ does. But as of ReqChange 3.0, you
can send a message when any of the patched requesters appears.

  To be able to send different messages from different requesters, you can
specify the following requester attributes:

  Requester type

  The requester must be of this type for a match. You may also specify
'Any', which will match on all requesters.

  Title

  The title of the requester (i.e. the window title) must contain a certain
string. Case is ignored when checking. If not specified, the requester title
is ignored.

  Body

  This only applies to the text requester (or if you have set the requester
type to 'Any'). The body text must contain a certain string. Case is
ignored when checking. If not specified, the body text is ignored. If the
requester that is about to open doesn't have any body text, this string is
ignored.

  Program

  The calling program must have a certain name. Case is ignored when
checking. If not specified, the program name is ignored.

  These attributes, together with a name ^{1} and a message is stored in a
Rexxnode. All nodes are placed in the Rexx list, and you may edit the list
and the nodes, using the Edit window and the Rexx window in
The prefs editor.

  The message is a so called string file macro. This means that the command
is executed like it was executed using a command line looking like:

  Rx "say 'This is a string file macro'"

I.e. the string after the Rx command is the string file macro.

When a requester is about to appear, ReqChange will search through the
Rexx list (from top to bottom), and as soon a node that matches the current
requester is found, the corresponding ARexx message is sent. Thus,
specific nodes should be placed near the top of the list, and more general
ones near the bottom.

When the message returns, the return code will be checked (unless there
was any error), and an automatic gadget selection may occur for text
requesters. The return value specifies which gadget to select: '-1' equals
the rightmost gadget (usually cancel), '1' equals the leftmost gadget, '2'
equals the next one, and so on... If the return doesn't match any gadget,
the requester will appear.

A word of warning: This auto gadget selection can cause a
deadlock situation (eating up a lot of CPU time) if you aren't careful. An
example: A "No disk present" requester is set to return 1. The OS tries to
open a requester telling you to insert a disk, while the message tells the
OS to retry, instead of opening the requester. The only way to get out of
this loop is to insert a disk.

The fact that ReqChange waits until the message returns means you may
need to e.g. Run a program you start, and that you should avoid doing
things that may take some time to complete. To make an ARexx macro run
asyncronously, you'll need to write something like this in the command:

ADDRESS COMMAND "Run Rx <Macro filename>"

-- Footnotes --

^{1} This name field is ignored by ReqChange. It is included so that you can
enter something descriptive, to make it easy to see what a Rexxnode does.
This field is what you see in the list.

1.17   ReqChange.guide/Excepted programs

Excepted programs

In some cases there might be a problem with the patches ReqChange
installs. If so, that program can be excepted from patching. In
The Prefs editor, such programs can be entered via the Except window
(opened from the Edit window). ReqChange will try to get a name for a given
process, so that it will be the same, regardless if it was stared from the
Workbench or the Shell.

Hopefully, this feature won't be needed much. But if you do need it,
please tell me if you use it because of something you consider to be a
problem with ReqChange, so that I can fix it.

1.18   ReqChange.guide/The prefs editor

The prefs editor

```
  Arguments           Command line arguments and tooltypes.
  Main window         The gadgets in the main window.
  Main window menus   The menus in the main window.
  Edit window         The gadgets in the edit window.
  Edit window menus   The menus in the edit window.
  Deny window         The gadgets in the deny window.
  Rexx window         The gadgets in the Rexx window.
  Except window       The gadgets in the except window.
  Program window      The gadgets in the program window.
```

   The prefs editor allows you to customize the behaviour of the patches,
which libraries to patch, etc. You can also edit the lists that holds
information about what message to send from certain requesters (see
ARexx messages) and which volumes that always should be denied (see
The AssignWedge), and if some programs shouldn't use the patches (see
Excepted programs). You may also install, update or remove the patches (see
Main window menus/Patches).

   All changes you do have immidate effect, so you don't need to exit the
editor to try them.

   Pressing the HELP key will open an AmigaGuide® window with help (for the
currently active window), if you have AmigaGuide® installed, and placed the
file ReqChange.guide somewhere in the AmigaGuide® search path (this is
either the same directory as you have the prefs editor, or any of the
drawers specified in the AmigaGuide/Path environment variable). You can also
press HELP when you're in a menu, to get help about the current menu.

   The F1 key will zip/unzip the window, and cursor left and cursor right
will activate the "previous" and "next" window, respectively.

   As ReqChange is implemented as a library and doesn't have any IPrefs-
like program (that would wait for any changes to the preferences files), a
simply copy of a preferenes file to Env:ReqChange/ will not be noticed. Use
the prefs editor argument USE in combination with the FROM argument do
something like this (see Arguments).

   Any version of the prefs editor should work with any version of
reqchange.library (and vice versa ofcourse. However, sometimes a new
version of the preference editor might require a new function in the
library). You maybe won't be able to edit all things, or all gadgets maybe
doesn't have any effect, but these should be the only problems. The
interface between the prefs editor and the library have been designed to
eliminate compatibility problems.

1.19   ReqChange.guide/The prefs editor/Arguments

Arguments

   The preferences editor have the following template. The arguments can be

specified on the command line, or as tooltypes in the icon:

    FROM/M,EDIT/S,USE/S,SAVE/S,PUBSCREEN/K

    FROM

    Here you can specify a preferences file to load into the preferences
editor. Unlike the normal preferences editors, you can enter several files
here (space separated). You can't use patterns though. ReqChange will check
what type of file it is ^{1}, and if it indeed is a valid ReqChange preferences
file, it will be loaded before any window is opened. If you enter several
files of the same kind, only the last one will have any effect.

    EDIT

    Normally, when either USE or SAVE (see below) have been used, the editor
will not open its window. This switch will make it open the window anyway.

    SAVE

    If this option is specified, the editor will save the current
configuration to EnvArc:ReqChange, just as if you had selected the "Save"
gadget in the Main window. The main window will not be opened. If any FROM
files are specified, they will be read first.

    USE

    If this option is specified, the editor will save the current
configuration to Env:ReqChange, just as if you had selected the "Use"
gadget in the Main window. The main window will not be opened. If any FROM
files are specified, they will be read first

    PUBSCREEN

    This argument lets you specify which public screen the windows should open
on. If the screen couldn't be found, they will open on the default public
screen (which also happens if you don't specify any name). Please note that
this name is case sensitive.

    -- Footnotes --

    ^{1} There are four different preferences files: The main configuration file
(ReqChange.prefs), the denylist file (Deny.prefs), the Rexxlist file
(Rexx.prefs), and the exception file (Except.prefs).

1.20   ReqChange.guide/The prefs editor/Main window

Main window

    This window contains general flags for the patches, and the settings for
the different requester types. The settings of the gadgets and menu items
are saved in a separate file (called the main preferences file). The
following gadgets exists:

Asl font requester

If this gadget is checked, it means that the Asl font requester will be
patched. You don't need to clear the checkmark in this gadget in order to be
able to select e.g. different colors in the Font preferences program, since
ReqChange automatically will use the original font requester when needed.

Asl screen mode requester

If this gadget is checked, it means that the Asl screen mode requester
will be patched. If an unsupported feature is used (currently the "custom
screen mode list"), then ReqChange will automatically use the original
screenmode requester.

AssignWedge

If this gadget is checked, it means that The AssignWedge is activated.
This means that when a "Volume requester" is displayed, there will be three
extra gadgets, allowing you to mount the volume, enter an assign, or deny
this volume name.

ARQ mode

If this gadget is checked, it means that all text requester patches (and
the requesters the prefs editor opens) will use the text requester function
in Intuition. This way, another text requester patcher may be used instead
(e.g. ARQ), while still having the AssignWedge feature available. If you
use this feature, remember that the other patcher should be started before
ReqChange is started.

You can use the ARexx messages for these requesters, even if this
gadget is checked.

Note: The Intuition patches must be enabled for this to work.

Keys

If this gadget is checked, you can use the keyboard to select the gadgets
in the text requesters. Please refer to the ReqTools documentation for more
information about these keys.

Center text

If this gadget is checked, the body text in the text requesters will be
centered. This often makes the requester look better. Sometimes this
centering will be disabled, if a requester have centering "by hand". The
current algorithm for this is fairly simple: If a line starts with a space,
then the centering is disabled! ;) Please remember that this doesn't fix all
cases. Proportional fonts in the requester will also cause "troubles". Use a
fixed-width font, and the requester should look good.

Buffer

If this gadget is checked, ReqChange will try to buffer the filelist in
the different file requester patches. In some cases this isn't possible (the
Arp requester can't be buffered, while some of the Req and Asl file-
requesters' can be. It depends on the calling program), but when possible,

it means that the requester remembers all shown files from the previous
call. This uses a little memory, but removes the directory-reading time.
This is especially useful when you are using floppies, and you don't use the
new DirCache filesystem that exist in OS 3.0 and higher.

   Requester type

   This gadget specifies which requester type you currently are editing the
settings for. All gadgets within the "double" border applies to the
requester specific settings.

   Center mode

   This gadget specifies the center mode to use for the current requester
type. There are several possible center modes, but most of them should be
rather obvious. A few comments are in place though:

   'Normal' makes the requester center as they normally would have done if
the patch wasn't installed.

   'Default' causes ReqChange to use the current default as set in the
ReqTools preferences editor.

   The other ones does what the name implies. However, window-relative
options may not always behave like expected. In some cases, ReqTools can't
find any window to be relative to, and if so, the requester will be centered
relative to the screen instead. Also, the window may not always be the one
you expect. Thus the centering can be a bit odd sometimes. Mostly it should
work fine though.

   Offset

   For the 'Top left ...' centering modes you can also specify an offset,
using the Offset gadgets. The leftmost gadget holds the 'Left' offset,
while the rightmost holds the 'Top' offset.

   These gadgets can be disabled, depending on which center mode you are
using. Also, if the Default gadget is checked, these gadgets will be
disabled.

   Default

   If this gadget is checked, the offset set in the ReqTools preferences will
be used. Ohterwise the offset in the Offset gadgets will be used.

   This gadget can be disabled, depending on which center mode you are using.

   Default height

   If this gadget is checked, the requesters that can be resized will have
the default height, as set in the ReqTools preferences. Otherwise they will
have the height the calling program requested.

   Pop screen

   If this gadget is checked, then ReqTools will pop the screen a requester
appears on. This is similar to the ReqTools option, but here you can specify

it on a requester type basis, instead of a single global option.

   Backfill

   This gadget only applies to the string and integer requesters. If checked,
the requester will have a background pattern, like the one used in the text
requester. Otherwise the background will be blank (using the current
background color).

   ARexx messages

   If this gadget is checked, the ARexx messages for this requester type
will be enabled. Otherwise no messages will be sent when a requester of
this type appears.

   Font

   This gadget opens up a font requester, that allows you to choose a font
that will be used in the requesters. If no font is specified, the current
screen font will be used.

   Note: ReqChange will not issue any warning if a font can't be opened
during library init (i.e. when the library is loaded into memory, either
by StartRC or The prefs editor).

   Save

   This gadget saves the main window settings for permanent use. The lists
(in the Edit window) will be restored.

   Use

   This gadget saves the main window settings. Any changes will be lost when
you reboot or turn of your computer. The lists will be restored.

   Cancel

   This gadget undos any changes you have done to the main window settings,
and exits. Any changes done to the lists are not affected. The lists will
be restored.

1.21   ReqChange.guide/The prefs editor/Main window menus

Main window menus

   The main window have the following menus:

   Project   File operations, etc.
   Edit   Different edit functions.
   Patches   Control the patches.
   Options   Options.

1.22   ReqChange.guide/The prefs editor/Main window menus/Project

```
Project
```

   Open...

   This menu item allows you to select a previously saved main preferences
file to use. You can only load a valid preferences file.

   Save as...

   This menu item allows you to save the current main preferences to a file
of your choise.

   About...

   This menu item shows a requester with some information about the program
and the patches. You can see if the patches are installed, how many of the
patches that were active when you selected this menu item, and the number
of allocated requesters.

   Quit

   This menu item quits the preferences editor. Works like the "Cancel"
gadget in the Main window.

1.23   ReqChange.guide/The prefs editor/Main window menus/Edit

```
Edit
```

   Lists...

   This menu item opens the list editor window, where you can edit the deny,
Rexx and exception list. If the window already is open, it will be unzoomed
if needed, and then brought to the front.

   Defaults

   This menu item sets the main settings to the builtin defaults (note:
these defaults are stored in the library, and not in the editor).

   Last saved

   This menu item reloads the last saved main preferences file (saved with
the "Save" gadget in the main window).

   Restore

   This menu item restores the main settings to the state they were in when
the editor was started.

1.24   ReqChange.guide/The prefs editor/Main window menus/Patches

Patches

   Intuition

   If this menu item is checked, it means that the Intuition patches are
enabled. If you change the state of it, ReqChange will try to install/remove
the patches as needed. If it isn't possible to remove the patches, they will
remain in a disabled state, so that you won't notice them. See
Patched requesters/Intuition.

   Asl

   If this menu item is checked, it means that the Asl patches are enabled.
If you change the state of it, ReqChange will try to install/remove the
patches as needed. If it isn't possible to remove the patches, they will
remain in a disabled state, so that you won't notice them. See
Patched requesters/Asl.

   Note however that if you install the patches, the currently running
programs may or may not use the new ReqTools requesters. This depends on the
program (technical note: if they keep a requester structure allocated or
not). If you want them to use ReqTools directly, quit and restart the
programs, and they should now use ReqTools.

   So some of you says: "But can't the same thing happen when I try to remove
the patches?" No, it can't, since ReqChange have the ability to fall back to
the original Asl function when needed.

   Arp

   If this menu item is checked, it means that the Arp patches are enabled.
If you change the state of it, ReqChange will try to install/remove the
patches as needed. If it isn't possible to remove the patches, they will
remain in a disabled state, so that you won't notice them. See
Patched requesters/Arp.

   Req

   If this menu item is checked, it means that the Req patches are enabled.
If you change the state of it, ReqChange will try to install/remove the
patches as needed. If it isn't possible to remove the patches, they will
remain in a disabled state, so that you won't notice them. See
Patched requesters/Req.

   Update

   This menu item will try to update the patches so that they reflect the
current settings in this menu. What this means is that ReqChange will try
to install/remove any patches that need to be.

   Each time you alter the state of any of the patch checkmarks in this
menu, such an update will be performed, so you'll probably not need this
one that often.

Install...

  This menu item will install the patches. If the patches already are
installed, you will be informed about it.

  Remove...

  This menu item will try to remove all patches. If this isn't possible,
you'll be asked if all patches should be disabled or not. Disabling all
patches is equivalent to clearing all checkmarks in this menu. See StartRC
for more information about the problems in removing patches.

1.25   ReqChange.guide/The prefs editor/Main window menus/Options

Options

  Create icons

  If this menu item is checked, ReqChange will write an icon together with
all main preferences files that are saved using Project/Save as... (see
Project).

1.26   ReqChange.guide/The prefs editor/Edit window

Edit window

  In this window you can edit the three different lists, the deny list, the
Rexx list and the except list. Each list is saved in a separate file. These
lists are not affected by anything you do in the main window (except for
the gadgets that can exit the editor, which will close this window, and
restore the lists from Env:ReqChange), and the main window is not affected
by anything you do here.

  The window contains the following gadgets:

  List type

  This gadget lets you choose which list to edit. There are currently three
lists you can edit, the deny list, the Rexx list and the exception list.

  List contents

  This gadget shows the contents of the current list. Use the arrow gadgets
or the drag-bar to move around in the list. Select an entry (a node) by
clicking on it. A double-click will open a window that allows you to edit
that node. See Deny window, Rexx window and
Except window.

  Add...

  This gadget opens an empty node-edit window so that you can enter a new
one.

```
  Delete...

  This gadget deletes the currently selected node.

  Edit...

  This gadget opens a node-edit window allowing you to edit the currently
selected node.

  Sort

  This gadget sorts the nodes in the current list in alfabetical order.

  Up

  This gadget moves the current node up one step in the list.

  Down

  This gadget moves the current node down one step in the list.

  Save

  This gadget saves the lists for permanent use, and closes the edit
window.

  Use

  This gadget saves the lists, and closes the edit window. Any changes will
be lost when you reboot or turn off your computer.

  Cancel

  This gadget closes the window. ReqChange will try to restore the lists
(by reading from Env:ReqChange/) to the way they were when you opened the
edit window.
```

1.27   ReqChange.guide/The prefs editor/Edit window menus

```
Edit window menus

  The edit window have the following menus:

  Project    File operations, etc.
  Edit       Different edit functions.
  Options    Options.
```

1.28   ReqChange.guide/The prefs editor/Edit window menus/Project

```
Project
```

Open...

This menu item will allow you to select a list to load. The current list will be deleted, but only if the file is of suitable type. You must select a file of the same type as specified in list type gadget.

Append...

This menu item allows you to select a list to load. The current list will not be deleted. You must select a file of the same type as specified in the list type gadget. The new nodes are added to the end of the list, and no check for duplicates is done.

Save as...

This menu item allows you to save the current list to a file of your choise. If you specify a name that already exists, no warning is issued, and the file will be deleted if there is an error during the saving.

About...

This menu item opens the About requester, as in the main window menus. See Main window menus/Project.

Quit...

This menu item closes the edit window, like the "Cancel" gadget in the Edit window.

1.29   ReqChange.guide/The prefs editor/Edit window menus/Edit

Edit

Clear list...

This menu item will delete the current list.

Last saved

This menu item allows you to load the last saved list of the current type, saved using the "Save" gadget (see Edit window). The old list will be replaced. Use "Append..." to append the last saved list (see Project).

Restore

This menu item will try to restore the currently selected list (by reading from Env:ReqChange/) to the way they were when the editor window was opened.

1.30   ReqChange.guide/The prefs editor/Edit window menus/Options

```
Options

   Create icons
```

If this menu item is checked, ReqChange will write an icon together with
all list files that are saved using Project/Save as... (see Project).
This menu item does not affect the "Create icons" option in the
Main window menus.

1.31   ReqChange.guide/The prefs editor/Deny window

Deny window

   This window allows you to enter/edit a deny node. A deny node is a very
simple one. It simply contains the name of the volume to deny. This is also
the name shown in the listview gadget.

   The following gadgets exist in the window:

   Deny name

   In this gadget you specify the name of the volume that should be denied.
Do not enter any colon (':'). E.g., to deny the volume 'DF9:' (which
normally doesn't exist! :) you should thus enter 'DF9'. You must enter
something here. An empty string, or a string that only consists of spaces
is not allowed.

   Ok

   This gadget accepts the string as it is, and uses that name in the
new/changed node.

   Cancel

   This gadget closes the window, and the list/node is left unchanged.

1.32   ReqChange.guide/The prefs editor/Rexx window

Rexx window

   This window allows you to enter/edit a Rexxnode. A Rexxnode contains
several fields so that you can specify exactly which requesters that should
have a certain ARexx message associated with them.

   The Rexx window contains the following gadgets:

   Name

   This gadget allows you to enter a name for the node. Since there isn't
really any field that is suitable to use for identifying a node (and to
place in the listview gadget), you have this one, where you can enter a

name of your own choise for the node. You must enter something here. An
empty string, or a string that only consists of spaces is not accepted.
Since this field is for you only, why not enter something that makes it
easier for you to see what kind of requester the node matches?

   The name can be up to 500 chars long.

   Requester type

   This gadget allows you to select a requester type to match on. Only if
the requester is of the specified type, a message can be sent for the
requester. 'Any' matches any requester.

   Title

   This gadget allows you to enter a string that the title must contain. If
empty, the title of the requester is ignored.

   The string can be up to 500 chars long.

   Body

   This gadget allows you to enter a string that the body must contain. If
empty, the body of the requester is ignored. You can only enter a string if
the "Requester type" gadget is set to 'Any' or 'Text', as these are the
only cases when there is a body to compare with.

   The string can be up to 500 chars long.

   Program

   This gadget allows you to enter the name of the program that called for
the requester. If the name doesn't match, no message will be sent. If this
gadget is empty, the no name check is done. If you press Shift-down, then
the Program window will appear.

   The name can be up to 500 chars long.

   Popup gadget

   This gadget (which contains a small arrow-down image) opens the Program
window. It containins a list of all the programs currently running, with
the name as ReqChange sees them. The program you select will be entered
into the Program gadget above.

   Command

   This gadget allows you to enter the command that should be sent to ARexx
if a requester matches this node. You must enter something. An empty string,
or a string that only contains spaces isn't accepted. See ARexx messages for
more information.

   The command can be up to 500 chars long.

   Ok

   This gadget accepts the Rexxnode as it is, and places puts the

new/changed node in the list.

   Cancel

   This gadget cancels the window, and the node/list is left unchanged.

1.33   ReqChange.guide/The prefs editor/Except window

Except window

   This window allows you to enter/edit an exceptnode. An exceptnode only
contains the name of the program that should be excepted from patching.

   The except window contains the following gadgets:

   Program

   This gadget allows you to enter the name of the program to except. If you
press Shift-down, the Program window will appear.

   The program name can be up to 500 chars long.

   Popup gadget

   This gadget (which contains a small arrow-down image) opens the Program
window. It containins a list of all the programs currently running, with
the name as ReqChange sees them. The program you select will be entered
into the Program gadget above.

   Ok

   This gadget accepts the exceptnode as it is, and places puts the
new/changed node in the list.

   Cancel

   This gadget cancels the window, and the list/node is left unchanged.

1.34   ReqChange.guide/The prefs editor/Program window

Program window

   This window allows you to select a program, which name will be entered in
the Program gadget in the Except window or the Program gadget in the
Rexx window.

   The program window contains the following gadgets:

   List contents

   This shows the names of all programs that were running when the window
opened. Use the arrow gadgets or the drag-bar to move around in the list.

Select a program by clicking on it. A double-click will select the program,
and close the Program window.

   Technical note: ReqChange tries to get the "real" name of the program,
which not always match the name e.g. Exec thinks it has. This is needed
since the Exec name usually depends on how the program was started, and this
shouldn't make any difference in this list. And mostly it doesn't. I don't
know if the method used here is perfect, but it seems to be rather good at
least.. :)

   Note: This list is normally sorted, but it might not be, if the sort
function failed for some reason (this should normally not happen).

   Ok

   This gadget accepts the current program, closes the window, and enters the
program name into the proper gadget (overwriting any old program name).

   Cancel

   This gadget closes the window, but it leaves the gadget unchanged.

1.35   ReqChange.guide/StartRC

StartRC

   StartRC is a small program used to start or stop ReqChange. It simply
loads reqchange.library (if needed) and tells it to install or remove the
patches (see The prefs editor/Main window menus/Patches).

   If ReqChange already is installed when StartRC is started (:), it will
ask you if the patches should be removed.

   The patches can't always be removed. If some other program have "patched
over" ReqChange's patches (and a program like SetMan isn't installed), there
is no way for ReqChange to exit safely, and thus, it won't be done. Also, if
there are some allocated requesters around, ReqChange can't be removed
either, since these requesters must be freed first, and the original
functions can't handle that. And ofcourse, ReqChange can't exit if some
patch is active (e.g. a requester window is open).

   When the patches are removed, the library will be removed from memory
whenever the system needs the memory for other things (assuming no one else
is using reqchange.library, like The prefs editor).

   Note: Removing patches is never a completely safe operation. One can only
try to make it as safe as possible (well, there is a safe way, and that is
to never unload the patches from memory. But that isn't a good solution
either), and hope for the best. :) Normally, there shouldn't be any problems
though.

   StartRC can be started from both the Workbench and the Shell.

   Note for OS 2.1+ users: Since both The prefs editor and StartRC use the
same catalog file, you will either need to have two copies of

reqchange.catalog, or place it in Locale:Catalogs/<language> (where
"<language>" is the language you have selected in the Locale preferences
editor).

1.36 ReqChange.guide/Acknowledgements

Acknowledgements

  Thanks go to the following persons, for helping me with ReqChange in one
way or another:

  Nico Francois – For his great reqtools.library, the programming help, and
the ideas "borrowed" from RTPatch.

  Fredrik Orinius – For being the first one to send me a donation for
ReqChange! And this was before ReqChange became Shareware.

  Nicola Salmiora – For his SetPatch() function and the help with the
"expunge problem".

  Stefan Boberg – For the debugging help in some of the earlier versions.

  Michael Berg – For the BOOPSI "getalt" gadget (i.e. the arrow down image).

  Martin Huttenloher – For the nice MagicWB-icons in this archive.

  The beta-testers (in random order): Roger Andersson, Johan Billing,
Michael Berg, Stefan Johannesdal, Stefan Zeiger, Per-Anders Josefsson,
Markus Aalto, Lieven Lema, Stu Churchill, Roger Nordin and Dennis Björklund.

  The translators (also in random order): Michael Berg (Dansk); Stefan
Zeiger (German); Markus Aalto (Soumi); Andrea Suatoni (Italiano); Mike Rooze
and Marco Niese (Nederlands); Georges Goncalves (Français).

  All of you who have sent me some sort of feedback on ReqChange.

  Legal information largely stolen from the documentation to TrapDoor
(the swedish translation was largely stolen from the swedish translation of
the documentation to Spot).

  Icon inspiration from the icon to the ReqTools preferences program.

  Spelling errors etc. in this manual by myself (hey, english is only my
second language! I'm doing as good as I can! :)

  Persons mentioned here by name doesn't need to send any shareware fee
for using this program. They have done enough already! :)

1.37 ReqChange.guide/The future

The future

Although I've implemented most things I found important in ReqChange, there are still things that might be implemented at some future date. And maybe you have some idea on how to improve this program? If so, please send me a letter, and I'll consider it. To give you some inspiration, here are some of the ideas that popped up during the development of this program:

· A "requested width" for the req.library/GetString() patch (user request actually). But were should I place this in the prefs editor? :)

· A ReqTools-patch (!). :) This patch would add the ARexx-messages to all ReqTools requesters, and probably add some of the ReqChange prefs settings too.

· Formatting codes in the ARexx commands, so that e.g. the current title, requester type and body, can be inserted (useful for the automatic gadget selection).

· A RCPrefs program (similar to IPrefs) that would automatically re-load the config files if they are changed "externally" (i.e. not by the prefs editor). I certainly don't need such a program, but some persons maybe does?

· A possbility to use special prefs (some or all) for selected requesters (similar to the ARexx-message selection).

· Function keys to select one of the gadgets in the text requesters (ala Arq). Should be in ReqTools.. :)

1.38 ReqChange.guide/Program history

Program history

```
  Version 1.00-2.03
  Version 3.0
  Version 3.1
  Version 3.2
  Version 3.3
  Version 3.4
```

1.39 ReqChange.guide/Program history/Version 1.00-2.03

Version 1.00-2.03

  Ancient history.

1.40 ReqChange.guide/Program history/Version 3.0

Version 3.0

  This version is a complete rewrite from scratch!

· All resident code is written in efficient assembler, and stored in a
library.

· Patch installation/removal code a lot smaller and a bit smarter. SetMan
is no longer treated as a special case. Thanks to Nicola Salmiora for his
SetPatch() function, which was used as a base when I rewrote the patch
related code.

· Stack swapper redesigned. Now uses exec.library/SwapStack().

· ARexx messages can now be sent from any requester, based on requester
type and title (and/or body, for the text requester). These messages can be
disabled for certain requester types, for maximum efficiency.

· Centering modes, offsets, fonts etc. can be selected for each requester
type.

· Two new centering modes: 'Default' and 'Normal'. 'Default' will make
ReqChange use whatever set in the ReqTools preferences editor, while
'Normal' will make the requesters appear where they normally would have if
ReqChange wasn't installed (although the Asl patches won't remember their
position between calls).

· Prefs editor that allows you to modify the behaviour of ReqChange with a
nice user interface, complete with keyboard shortcuts and online help. You
can also edit the deny and Rexx list!

· Complete support for Asl V38 (as complete as it can be, I hope). This
includes the accept/reject patterns, the "dirpattern", all hooks & tags etc.

· The Asl patches can now fall back to the original requester if the
patches have been disabled or an unsupported feature is used. The code for
this is fairly small, thanks to a new approach of handling tags.

· Improved localization. Will now automatically adjust to any new language
selected.

· The AutoRequest()/EasyRequest() patch can now (optionally ofcource) call
the original function after the AssignWedge. I suspect some ARQ users will
like this. ;)

· Documentation completely rewritten in AmigaGuide® format.

· Requires OS 2.0, and is OS 3.0 aware.

· No longer a Commodity.

· Patterns will only be converted if necessary (i.e. the WILDSTAR bit in
dos.library is cleared).

· Tested with Enforcer & MungWall (earlier versions were only MungWall
tested).

· Uses memory pools whereever possible. This reduces memory fragmentation.

· Should handle text requesters with "manual" centering better.

· Asl font requester patch is now optional.

· Asl screenmode requester patch added (optional).

1.41   ReqChange.guide/Program history/Version 3.1

Version 3.1

· Minor bugfix: Prefs files would have the wrong protection bits when
saved. This update only reflects reqchange.library.

· Made a few corrections to the manual (there were some references to non-
existant sections).

1.42   ReqChange.guide/Program history/Version 3.2

Version 3.2

· Fixed incorrect fido-address in this manual.

· The ARexx-messages should now work without enforcer hits etc.

· The ARexx-messages were not always sent when they should have been. Now
this should work! :)

· ARexx-messages may now be sent when an Asl patch calls the original
requester.

· Translations for French, Italian and Dutch added.

· In the AssignWedge, the gadget "Assign..." will now default to the same
drawer as the program resides in, if this is defined. Otherwise Ram: will
be used as before.

· Leftoffset and topoffset should be properly saved now. Also improved
some other "related" code, to ensure no problems with newer OS versions (if
there will be any :( ).

· Fixed a bug in the library startup code. Also changed a message to be
nonlocalized to make things a little simpler. If this message is to be
shown, it isn't likely the catalog will load anyway! :)

· Included a small JPeg (8 Kb), so you can see what I look like! :) Use
e.g. PPShow (V4.0+), FastJPEG or ViewTek to view it.

· Bug in the Asl screenmode requester fallback fixed (usually led to
crashes. But not always).

· Version 2.2b of ReqTools included.

1.43   ReqChange.guide/Program history/Version 3.3

Version 3.3

  Bug fixes:

  · The initial value of the height gadget in the Asl screenmode requester
was not correct.

  · The prefs editor will now close the workbench.library before exiting.

  · The prefs editor will now unlock its lock on the Workbench screen
properly (it makes the lock to see if the prefs editor is on the Workbench
screen, and if so, activates the AppWindow support).

  · Various bugs/(layout) quirks in the prefs editor GUI code fixed, code
optimized a bit, etc.

  · Enforcer hit in the prefs editor during Workbench start and the icon
didn't have any tooltypes removed.

  · In the "Assign..." requester, any assigns relative to a programs home
directory (i.e. when the requester opens in the home directory) was
incorrectly made.

  · The Req/GetLong() patch didn't fall back properly.

  · Programs requesting pen selection in the Asl font requester would never
get a requester on OS 3.1+. A few default values weren't properly set,
causing the requester to fail.

  · A couple of other defaults weren't properly set, causing e.g. WSpeed to
crash, when showing an Asl screenmode requester. Was this caused by a
ReqTools bug as well? :)

  Improvements:

  · Exception list added. A program in this list will always use the
original functions.

  · Program name field added to the Rexxnode. Don't worry, old prefs files
can still be read.

  · Automatic gadget selection added (for text requesters only). Done via
the return code from ARexx messages.

  · In string gadgets in the prefs editor, shortcuts can now be selected by
pressing right amiga together the key in question. Because of that (and some
persons – including me – have StringClip installed), the shortcut for cancel
was changed from c to a (in the english translation ofcourse).

  · A few minor improvements/bug fixes in the library.

  · The prefs editor now handles larger than normal window gadgets (if
SysIHack is installed, this may be the case. If you are interested in how

this is done, check the file called Source/BottomBorder.c in the
distribution archive).

· The install script now automatically installs MagicWB-icons if MagicWB
version 2.0 or higher is installed.

· The prefs editor now checks for the icon def_PREF.info (instead of
def_PREFS.info) in the drawers Env:ReqChange and Env:Sys when saving
preferences files. This way, e.g. the MagicWB preset icon will be used, if
present.

1.44 ReqChange.guide/Program history/Version 3.4

Version 3.4

· The "Only AssignWedge" option didn't work too well. Incorrect ARexx-
messages was sent (possibly causing Enforcer hits), and the AssignWedge
wasn't activated at all.

· Changed behaviour of StartRC a little. If started when ReqChange is
installed, and all patches are inactive, then the patches will be activated
agian (instead of a requester popping up asking if you would like to remove
the patches).

· Did some minor changes to the prefs load/save routines, to prevent any
problems with virtual memory systems (such as VMM).

· The AssignWedge code trashed the current process' window pointer
(pr_WindowPtr). I discovered the problem while looking at the code, so I
never experienced any problems.. Strange.. :)

· The flag "Only AssignWedge" have been renamed into "ARQ mode". If
enabled, all text requesters opened by ReqChange will use the requester in
Intuition instead, if possible (so that e.g. ARQ can be used instead).
ARexx-messages will work for these requesters as well (Note to all
translators: I have changed the string for MSG_ONLYASSIGN_GAD to "ARQ".
Please send me any modification for this, if needed).

· Did some other minor changes in the prefs editor.

· Programs in the program list that have "high ASCII" in their names
could get the wrong name.

1.45 ReqChange.guide/Program history/Version 3.5

Version 3.5 (Library version: 4.2)

· reqchange.library crashed when trying to read a main preferences that
didn't exist (didn't try the new "VM-compatible" code that much.. :)

· If you managed to doubleclick an item twice in the listwindow, the edit
window in question would be opened twice.

· A few Enforcer hits (that didn't always occur) when the prefs editor
was started have been removed.

· Sending an ARexx-message for a text requester with only one gadget caused
the requester to be cancelled (by the auto gadget selection).

· Pressing Left Amiga + <key>, where <key> doesn't correspond to a
shortcut will not inactivate the string gadget any more.

· The windows will now backfill with the current background color, rather
than color 0 (although these usually are the same :).

· Removed a couple of Enforcer hits in the program window.

· Forgot to mention one thing about version 3.4: ReqTools 2.2c included.

· Corrected various incorrections in this file.

· Made a Swedish translation of this doc file.

· The "task name copier" didn't quite work when MungWall was installed
(and possibly at some other times as well). This meant that ARexx-messages
weren't sent when they should be.

1.46 ReqChange.guide/Glossary

Glossary

Catalog: A file that contains all text a program uses that needs to be
translated. By loading this file (and with the help of a few OS routines) a
program can operate in several different languages with relatively small
amount of work by the programmer.

IMHO: Acronym for in my humble opinion.

Library: A module containing a set of functions which other programs may
use. These modules can be stored on disk and loaded when needed.

String file macro: A special type of an ARexx macro. Instead of having the
macro in a file (which is the usual form), it is simply a string that
contains all the ARexx commands (separated with semi-colons) that should be
executed.

1.47 ReqChange.guide/Index

Index

Acknowledgements
ARexx messages
Arguments
Arp