

PowerCache

COLLABORATORS

	<i>TITLE :</i> PowerCache		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 27, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	PowerCache	1
1.1	PowerCache ARexx Documentation (18-Jul-1993)	1
1.2	Introduction	2
1.3	PowerCache and ARexx	3
1.4	ARexx Command: CREATEICONS	6
1.5	ARexx Command: VERSION	7
1.6	ARexx Command: ENABLECACHES	7
1.7	ARexx Command: DISABLECACHES	7
1.8	ARexx Command: LISTCURRENTLYCACHED	8
1.9	ARexx Command: GETCACHEINFO	9
1.10	ARexx Command: CACHEABLE	10
1.11	ARexx Command: RESCAN	11
1.12	ARexx Command: GETDEVICEINFO	12
1.13	ARexx Command: ADDCACHE	13
1.14	ARexx Command: REMOVECACHE	14
1.15	ARexx Command: EDITCACHE	14
1.16	ARexx Command: GETSTATS	15
1.17	ARexx Command: SETSHOWKEY	16
1.18	ARexx Command: SETENABLEKEY	16
1.19	ARexx Command: SETDISABLEKEY	16
1.20	ARexx Command: LOADPREFS	17
1.21	ARexx Command: SAVEPREFS	18
1.22	ARexx Command: HIDE	18
1.23	ARexx Command: SHOW	19
1.24	ARexx Command: DEFAULTPREFS	19
1.25	ARexx Command: LASTSAVED	19
1.26	ARexx Command: BEEP	20
1.27	ARexx Command: POPUP	21
1.28	ARexx Command: QUIT	21
1.29	ARexx Command: USEHELP	22

Chapter 1

PowerCache

1.1 PowerCache ARexx Documentation (18-Jul-1993)

Documentation for

PowerCache v37.58

A Flexible and Powerful Disk
Caching System for the Amiga

Written by Michael Berg
Copyright (C) 1993 by Michael Berg
All rights reserved.

Subsection: The ARexx Interface

This document describes the various ARexx commands currently incorporated into PowerCache. Please select any one of the following items:

Brief Introduction to ARexx	General Information
PowerCache and ARexx	Further details..

Available ARexx Commands

ADDCACHE	Add a cache to an AmigaDOS device
BEEP	Change BEEP mode
CACHEABLE	List cacheable AmigaDOS devices
CREATEICONS	Set create-icons mode
DEFAULTPREFS	Use default preferences
DISABLECACHES	Disable all caches
EDITCACHE	Change cache characteristics
ENABLECACHES	Enable all caches
GETCACHEINFO	Get cache info
GETDEVICEINFO	Get extensive device info
GETSTATS	Get statistics info on a cache
HIDE	Hide PowerCache's window
LASTSAVED	Load last saved preferences
LISTCURRENTLYCACHED	List cached devices

LOADPREFS	Load preferences
POPUP	Set POPUP mode
REMOVECACHE	Remove a cache
RESCAN	Rescan AmigaDOS device list
SAVEPREFS	Save preferences
SETENABLEKEY	Define enable-caches hotkey
SETDISABLEKEY	Define disable-caches hotkey
SETSHOWKEY	Define show-window hotkey
SHOW	Show PowerCache's window
USEHELP	Enable or disable online help feature
VERSION	Return a version string in RESULT
QUIT	Quit PowerCache
Run Test Program	Requires PowerCache to be running

1.2 Introduction

For those of you who haven't got a clue what ARexx is, let me just try to outline the general idea.

When someone writes a program, the programmer implements a number of features in it. These features are all available when using the program. For example, in a drawing program, you would have functions to plot a point when the user clicks somewhere on the screen, plus functions to draw lines, circles, squares, polygons and filled shapes. All these functions can be activated by the user, in an interactive session.

And interactive is indeed the keyword here. Suppose that the user wanted to draw a shape from a huge list of (x,y) coordinates (geographic data, for example). The only way to do this would be for the user to read each coordinate individually, and then use the mouse to accurately click in the point on the screen.

As you can well imagine, this would take almost forever. And even if our poor user ever finishes, the problem will still be there the next time such a situation occurs.

Luckily, there's ARexx. If the programmer is clever enough to implement an ARexx interface in his drawing program, the plotting could be done automatically, assuming the presence of a PLOT command.

Here is an example of an ARexx script that could pull this off:

```
01 /* PlotTest.rexx - plot lots of coordinates stored in a file */
02 /* Author: John Doe */
03 /* Date: 4 feb 1993 */
04
05 /* Talk to the drawing program */
06 address 'SuperDuperDraw'
07
08 /* Open up the coordinates file */
09 if open('inputfile','dhl:coords.dat','r') then do
10     do while ~eof('inputfile')
11         xcoord = readln('inputfile')
12         ycoord = readln('inputfile')
```

```
14
15      /* Now, tell SuperDuperDraw to plot this pair of coordinates */
16      plot xcoord ycoord
17  end
18 end
19 else say 'Cannot open input file!'
```

The numbers to the left (in the beginning of each line) denote line numbers and are for your viewing pleasure only. They would not be entered into the actual ARexx script.

The command in line 6 tells the ARexx server that we want to talk to the program that has an ARexx port named SuperDuperDraw. In a real ARexx script, you would have to check if SuperDuperDraw was actually running or not, but this part is omitted for the sake of clarity and simplicity. In any event, if the ARexx server cannot find the program, the script will abort with an error message.

Next, in line 9, the coordinate file is opened, and read until the end of file marker is reached. The coordinate file is supposed to have one number per line, in pairs of alternating X and Y coordinates.

The command in line 16 does all the magic. There is no "real" ARexx command called PLOT, so the server forwards this command to whoever we are addressing, asking it to process that command. Command names are case insensitive, so it makes no difference if we say "plot" or "PLOT" or even "pLoT".

As it happens, our SuperDuperProgram does have an ARexx command called PLOT, so it responds to the request by plotting the coordinate pair given as parameters to the PLOT command.

By now you must have realized that using the ARexx interface would save our user a lot of work, and also speed up the process tremendously. This is the real advantage of ARexx: It automates sequences of actions that would otherwise have to be carried out interactively. This is also the reason why ARexx scripts are sometimes called macros.

Enough about drawing programs. Let's get back on track and talk about PowerCache's ARexx interface. Please go on and read the section on "PowerCache and ARexx". Take me there!

1.3 PowerCache and ARexx

PowerCache has a complete ARexx interface, and there is virtually nothing you can't make it do with ARexx commands. Be it adding or removing caches, inquiring about AmigaDOS devices, or obtaining statistical information from running caches, it's all there.

PowerCache's ARexx port name is POWERCACHE (note the all capital letters).

Most of the ARexx commands return values. Error messages are also returned in case something goes wrong. PowerCache uses the RC ARexx variable to return success or failure. A value of 0 means the command executed as expected and a value of 10 indicates an error of some kind.

In accordance with the suggestions made by C= in the Style Guide, PowerCache returns secondary error codes in the ARexx variable RC2. This variable may contain either a number or a string. In case of a number, the DOS command Fault can be used to discover the meaning of it. For example, if RC2 is 103, this means that not enough memory is available to execute the command (103 = No free store). Please refer to your AmigaDOS documentation for more information on how to use the Fault command (it's pretty simple: 1> Fault 103).

If there isn't any numeric code that matches a particular error condition, then PowerCache will set RC2 to a string describing the nature of the error. If you are using locale, then this message will even appear in the appropriate language (as will all output from PowerCache).

You can control which variable receives the return value of an ARexx command. If you do not do this, then PowerCache will simply place the result in the RESULT variable.

To put a return value in the variable PREVIOUSKEY, use this:

```
SETSHOWKEY "alt s" VAR PREVIOUSKEY
```

After this call returns successfully, PREVIOUSKEY (and not RESULT) will contain the previous hotkey.

Some commands return record-like variables. Unless otherwise specified, the return values will be placed in the RESULT variable, which isn't always a Good Thing. For example, the GETCACHEINFO command (which is used to return characteristics about installed caches), returns device, sets, lines, prefetch, type, algorithm, purgetime, free after purge (yes/no), cachebuffer, buffersize and overhead all at once. The call

```
GETCACHEINFO 'DF0:'
```

will return ALL these variables in the RESULT variable. RESULT may then contain something like this:

```
RESULT = 'df0: 8 32 4 R/O LRU 2 NOFREEAFTERPURGE MEMORY 655210 7006'
```

It can be quite hard to separate and extract individual values when you get it all in a single string. So here's what you do:

```
GETCACHEINFO 'DF0:' STEM RESULT.
```

^ The period is important

Now, in stead of returning everything in RESULT, you get individual values as "sub-variables" of RESULT. The above string will now be returned as:

```
RESULT.DEVICE    = DF0:
RESULT.SETS      = 8
RESULT.LINES     = 32
RESULT.PREFETCH  = 4
RESULT.TYPE      = 'R/O'
...and so on.
```

You will agree with me that this method is a bit more convenient to use than simply having everything in one variable.

The method for supplying parameters for the individual ARexx commands mimics that which is used by most normal DOS commands. Each command has a template that describes allowable parameters, and each individual parameter may or may not have options attached. These options consist of a / and a letter. For example, the /A option means that a parameter is required, and that the command will fail if you do not supply it. The command GetCacheInfo has a template like this:

```
GETCACHEINFO DRIVE/A
```

This means that you MUST supply a specification of a drive (like DH1:), or an error will be returned.

Here is a brief description of what some of the template options mean:

```
/A - Required  
/N - Numeric input expected  
/S - Parameter is a switch
```

There are a couple of other parameter options that are used extensively by many AmigaDOS commands. You'll find a detailed description of the template format in your AmigaDOS manuals.

If no options are appended to a parameter, it means that string input is optionally expected (that is, you may leave it out). Some parameters may have more than one option (like /N/A which means that this numeric parameter is required).

When you have a command that has several, optional parameters, it may be necessary to tell the template parser exactly what's what. Consider the following command:

```
ConvertFile SOURCENAME TARGETNAME TEMPORARYFILE
```

This imaginary command would read the file specified as SOURCENAME, do something with it using TEMPORARYFILE as a temporary file, and finally write the result to TARGETNAME. Since all parameters are optional, the following command will be legal, but very confusing to the parser:

```
ConvertFile dh1:data.txt ram:bank.cvg
```

Is RAM:bank.cvg the name of the destination file, or the temporary file? The grammar is ambiguous. What you have to do is this:

```
ConvertFile dh1:data.txt TEMPORARYFILE=ram:bank.cvg
```

The template parser understands this format, which clears up the confusion about what is what. The "=" may be left out, and you are free to shuffle parameters around on the command line, as long as you remember to denote each one with the correct tag. For example, the following commands are identical to the parser:

```
ConvertFile TEMPORARYFILE=dh1:data.txt TARGETNAME=ram:bank.cvg
ConvertFile TARGETNAME=ram:bank.cvg TEMPORARYFILE=dh1:data.txt
ConvertFile TARGETNAME=ram:bank.cvg TEMPORARYFILE dh1:data.txt
```

As previously mentioned, this `ConvertFile` is an imaginative command. PowerCache doesn't have a `ConvertFile` command. It serves merely as an illustration of parameter passing, and nothing else.

If you need to supply a parameter that contains blanks, you must do it like this:

```
ConvertFile 'TEMPORARYFILE="dh1:my file"' TARGETNAME=dh1:newfile.dat
```

Did you notice the double quotes? This is necessary because of the way the command line is first interpreted by `ARexx`, and then PowerCache. It looks strange, but it works. The importance of this becomes obvious in the context of the `SETxxxKEY` commands, which are used to redefine PowerCache's hotkeys. Since a hotkey is often something like "shift alt i" (which contains blanks), you have to use double-quoting.

Here's how to change the popup-window hotkey:

```
SETSHOWKEY '"shift alt i"'
```

I apologise for this. PowerCache's `ARexx` interface code is largely created by `ARexxBox`, which has some inherent problems with strings containing blanks (other than that, `ARexxBox` is a fantastic program!)

1.4 ARexx Command: CREATEICONS

NAME

`CREATEICONS` - Set "Create Icons" mode

SYNOPSIS

`CREATEICONS ON/S OFF/S`

FUNCTION

This function is used to control if PowerCache should create icons when preferences are saved. Specify `ON` to enable icon creation, and `OFF` otherwise.

INPUTS

`ON/S` - Specify this to activate icon creation
`OFF/S` - Specify this to deactivate icon creation

OUTPUTS

`RESULT` will contain the previous state (either '`ON`' or '`OFF`')

NOTES

Although it doesn't make much sense, it is allowable to specify both `ON` and `OFF`, in which case `OFF` will override `ON`. Also, you can toggle the current state by omitting both `ON` and `OFF`.

SEE ALSO

1.5 ARexx Command: VERSION

NAME

VERSION - Give out version information

SYNOPSIS

VERSION

FUNCTION

This command gives out the same kind of version information that you would get if you used the AmigaDOS "Version" command.

INPUTS

None.

OUTPUTS

Returns version string in the RESULT variable.

NOTES

Remember to put "options results" in your script, or you won't get anything back.

SEE ALSO

1.6 ARexx Command: ENABLECACHES

NAME

ENABLECACHES - Enable all installed caches

SYNOPSIS

ENABLECACHES

FUNCTION

Enable all installed caches. If these are already enabled, this function is a no-op.

INPUTS

None.

OUTPUTS

None.

NOTES

A future version of PowerCache may support enabling and disabling of individual caches.

SEE ALSO

DISABLECACHES

1.7 ARexx Command: DISABLECACHES

NAME

DISABLECACHES - Temporarily disable all installed caches

SYNOPSIS

DISABLECACHES

FUNCTION

Disable all caches that are currently installed. If the caches have already been disabled, this command is a no-op.

INPUTS

None.

OUTPUTS

None yet.

NOTES

Disabling a read/write cache may involve having to flush it out to disk. This may take some time. Expect this.

You should disable all caches before commencing such relatively dangerous operations as disk reorganization or repair. PowerCache will handle such situations, but should be disabled -- just to be on the safe side. Better safe than sorry :-)

A future version of PowerCache may support enabling and disabling of individual caches.

SEE ALSO

ENABLECACHES

1.8 ARexx Command: LISTCURRENTLYCACHED

NAME

LISTCURRENTLYCACHED - List all caches

SYNOPSIS

LISTCURRENTLYCACHED

FUNCTION

Return information on every installed cache.

INPUTS

None.

OUTPUTS

'LISTCURRENTLYCACHED STEM RESULT.' will return the following:

RESULT.DEVICE.COUNT Number of installed caches

RESULT.DEVICE.0 AmigaDos device name ('DF0:' for example)

RESULT.SETS.0 Number of sets

```

RESULT.LINES.0      Number of lines

RESULT.PREFETCH.0   Prefetch value

RESULT.TYPE.0       'R/O' for read-only caches
                   'R/W' for read/write caches

RESULT.ALGORITHM.0  'LRU' for LRU cache algorithm
                   'SPC' for SPC cache algorithm

RESULT.PURGETIME.0   Purge time in seconds

RESULT.FREEAFTERPURGE.0 'FREEAFTERPURGE' or 'NOFREEAFTERPURGE'

RESULT.CACHEBUFFER.0 'MEMORY' for a general memory cache
                   'CHIPMEM' for a cache using chip-memory
                   'FASTMEM' for a cache using fast-memory
                   'FILE filename' for a file cache

RESULT.BUFFERSIZE.0  Size of main cache buffer (in bytes)

RESULT.OVERHEAD.0    Administrative overhead (in bytes)

```

Information about the next installed cache can be found by replacing the ".0" with ".1". Typically you would use a DO loop to process each cache information entry - like this:

```

OPTIONS RESULTS

LISTCURRENTLYCACHED STEM RESULT.

DO i = 0 TO RESULT.DEVICE.COUNT-1
    SAY i': ' RESULT.DEVICE.i RESULT.PREFETCH.i
END

```

This tiny ARexx program would list the prefetch value for every installed cache to the console.

NOTES

Remember to enclose the device parameter in quotes, so the ARexx command parser won't be confused about the colon after the device name.

Purge delays of zero or lower will be truncated to 1.

SEE ALSO

GETCACHEINFO

1.9 ARexx Command: GETCACHEINFO

NAME

GETCACHEINFO - List details about a cache

SYNOPSIS

GETCACHEINFO DRIVE/A

FUNCTION

This function resembles the LISTCURRENTLYCACHED command very closely. The only difference between the two is that this one only returns information about a single installed cache.

INPUTS

DRIVE/A - A cached AmigaDOS device (example: 'DF0:')

OUTPUTS

"GETCACHEINFO 'DF0:' STEM RESULT." will return the following:

RESULT.DEVICE AmigaDos device name ('DF0:' for example)

RESULT.SETS Number of sets

RESULT.LINES Number of lines

RESULT.PREFETCH Prefetch value

RESULT.TYPE 'R/O' for read-only caches
 'R/W' for read/write caches

RESULT.ALGORITHM 'LRU' for LRU cache algorithm
 'SPC' for SPC cache algorithm

RESULT.PURGETIME Purge time in seconds

RESULT.FREEAFTERPURGE 'FREEAFTERPURGE' or 'NOFREEAFTERPURGE'

RESULT.CACHEBUFFER 'MEMORY' for a general memory cache
 'CHIPMEM' for a cache using chip-memory
 'FASTMEM' for a cache using fast-memory
 'FILE filename' for a file cache

RESULT.BUFFERSIZE Size of main cache buffer (in bytes)

RESULT.OVERHEAD Administrative overhead (in bytes)

NOTES

Remember to enclose the device parameter in quotes, so the ARexx command parser won't be confused about the colon after the device name.

Purge delays of zero or lower will be truncated to 1.

SEE ALSO

LISTCURRENTLYCACHED

1.10 ARexx Command: CACHEABLE

NAME

CACHEABLE - List all cacheable devices in the system

SYNOPSIS

CACHEABLE

FUNCTION

This function will scan the AmigaDOS device list for devices that appear to be cacheable. Actually, it more or less returns the contents of the "Cacheable Devices" listview gadget, and as such, a RESCAN operation may be required.

INPUTS

None.

OUTPUTS

'CACHEABLE STEM RESULT.' will return the following:

RESULT.DEVICE.COUNT Number of cacheable devices found
RESULT.DEVICE.0 AmigaDOS device name (example: 'DF0:')

The name of the next cacheable device can be found by replacing the ".0" with ".1". Typically you would use a DO loop to process each device entry - like this:

OPTIONS RESULTS

CACHEABLE STEM RESULT.

```
DO i = 0 TO RESULT.DEVICE.COUNT-1
  SAY i': ' RESULT.DEVICE.i
END
```

This tiny ARexx program would list all cacheable devices to the console.

NOTES

SEE ALSO

1.11 ARexx Command: RESCAN

NAME

RESCAN - Rescan the AmigaDOS device list for cacheable devices

SYNOPSIS

RESCAN

FUNCTION

This function will do a physical re-scan of all installed AmigaDOS devices. If PowerCache's window is open, the "Cacheable Devices" listview will also be updated by this operation.

Rescanning may be necessary if you mount additional devices AFTER you start PowerCache.

INPUTS

None.

OUTPUTS

None.

NOTES

Functionally equivalent to pressing the Rescan gadget in PowerCache's main window.

SEE ALSO

1.12 ARexx Command: GETDEVICEINFO

NAME

GETDEVICEINFO - List detailed information about an AmigaDOS device

SYNOPSIS

GETDEVICEINFO DRIVE/A

FUNCTION

This function will return a heap of information about an AmigaDOS device.

INPUTS

DRIVE/A - Any cacheable AmigaDOS device (example: 'DF0:')

OUTPUTS

"GETDEVICEINFO 'DF0:' STEM RESULT." will return the following:

```
RESULT.LOGICALDRIVE AmigaDOS logical device ('DF0:')
RESULT.DEVICENAME Device driver name ('trackdisk.device')
RESULT.UNIT Unit number (expect a zero here)
RESULT.TASKTCB Address of controlling task (null=no task)
RESULT.SECTORSIZE Sector size (almost always 512 bytes)
RESULT.SURFACES Device surfaces (floppy = 2, hd > 2)
RESULT.BLOCKSPERTRACK Blocks per track (floppy = 11)
RESULT.CAPACITY Total unit capacity (in bytes)
RESULT.RESERVEDSTART Blocks reserved by AmigaDOS (start)
RESULT.RESERVEDEND Blocks reserved by AmigaDOS (end)
RESULT.INTERLEAVE Interleave value
RESULT.LOWCYL Cylinder at which this unit begins
RESULT.HIGHCYL Cylinder at which this unit ends
RESULT.INITIALDOSBUFFERS Initial AmigaDOS buffers (typically 30)
RESULT.BUFMEMTYPE AmigaDOS buffer memory type
RESULT.MAXTRANSFER Max number of bytes to transfer per request
RESULT.MASK Address mask
RESULT.BOOTPRI Device and unit boot priority
RESULT.FILESYSTEM File system type
RESULT.BOOTBLOCKS Number of boot blocks
```

NOTES

Address mask is returned as an integer. You can translate it to hexadecimal with ARexx and compare it with the comment made by PowerCache in the Info window.

File system type is returned as an integer. You can translate it to hexadecimal with ARexx and compare it with the comment made by

PowerCache in the Info window.

The 'boot blocks' value is only valid for bootable devices, and may contain garbage even so.

GETDEVICEINFO is functionally equivalent to pressing the 'Info' gadget below the "Cacheable Devices" listview gadget in PowerCache's main window.

SEE ALSO

CACHEABLE, RESCAN

1.13 ARexx Command: ADDCACHE

NAME

ADDCACHE - Add a cache to an AmigaDOS device

SYNOPSIS

ADDCACHE DRIVE/A SETS/N/A LINES/N/A PREFETCH/N/A MODE/A
ALGORITHM/A PURGETIME/N/A FREEAFTERPURGE/S CACHEBUFFER/A

FUNCTION

This is the big one. You can use this function to install a cache to a cacheable AmigaDOS device.

INPUTS

DRIVE/A - The AmigaDOS device to add the cache to ('DF0:')
SETS/N/A - Cache sets
LINES/N/A - Cache lines
PREFETCH/N/A - Prefetch value
MODE/A - 'R/O' for read-only mode, and 'R/W' for read/write
ALGORITHM/A - 'LRU' for LRU algorithm, 'SPC' for SPC algorithm
PURGETIME/N/A - Purge time delay in seconds
FREEAFTERPURGE/S - Omit if you don't want buffers freed after a purge

CACHEBUFFER/A - 'MEMORY' for a general memory cache,
'FASTMEM' for a fast-memory cache
'CHIPMEM'~for a chip-memory cache
'FILE filename' for a file-cache

OUTPUTS

None.

EXAMPLE

The following calls all add various types of 512k cache buffers to df0:

```
ADDCACHE 'DF0:' 8 32 4 'R/W' LRU 2 'FASTMEMORY'  
ADDCACHE 'DF0:' 8 32 4 'R/O' LRU 10 'MEMORY'  
ADDCACHE 'DF0:' 8 32 4 'R/W' SPC 1 FREEAFTERPURGE 'MEMORY'  
ADDCACHE 'DF0:' 4 64 4 'R/W' LRU 3 'FILE dh1:t/df0_cache'
```

NOTES

This call will fail if you attempt to add a cache to a device which already has one.

Purge delays of zero or lower will be truncated to 1.

SEE ALSO

CACHEABLE, EDITCACHE, REMOVECACHE

1.14 ARexx Command: REMOVECACHE

NAME

REMOVECACHE - Remove a cache

SYNOPSIS

REMOVECACHE DRIVE/A

FUNCTION

Turn a cached device back to normal operation.

INPUTS

DRIVE/A - The AmigaDOS device to be restored (example: 'DF0:')

OUTPUTS

None

NOTES

Obviously, you cannot remove a cache from a device which doesn't have one.

SEE ALSO

ADDCACHE, EDITCACHE

1.15 ARexx Command: EDITCACHE

NAME

EDITCACHE - Modify the characteristics of a cache

SYNOPSIS

EDITCACHE DRIVE/A SETS/N LINES/N PREFETCH/N MODE ALGORITHM
PURGETIME/N FREEAFTERPURGE/S CACHEBUFFER

FUNCTION

This function allows you to change the characteristics of any installed cache. It does so by first removing the cache, then adding a new one with the changes you request.

INPUTS

DRIVE/A - Cached AmigaDOS drive name (example: 'DF0:')
SETS/N - New number of sets
LINES/N - New number of lines
PREFETCH/N - New prefetch value
MODE - New mode ('R/O' or 'R/W')
ALGORITHM - New algorithm ('LRU' or 'SPC')
PURGETIME/N - New purge time delay (specified in seconds)
FREEAFTERPURGE/S - Specify (or omit) this flag to change purge mode

CACHEBUFFER - New buffering mode
 'MEMORY' for a general memory cache
 'FASTMEM' for a fast-memory cache
 'CHIPMEM' for a chip-memory cache
 'FILE filename' for a file-cache

OUTPUTS

None.

NOTES

Note that only DRIVE is required. To change the number of sets for the cache installed on DF0:, use either of the following:

```
EDITCACHE 'DF0:' SETS=6  
EDITCACHE 'DF0:' SETS 6
```

To change both the number of sets, the number of prefetch and also change the purge mode to 'free after purge', use

```
EDITCACHE 'DF0:' SETS=6 PREFETCH=8 FREEAFTERPURGE
```

Purge delays of zero or lower will be truncated to 1.

SEE ALSO

ADDCACHE, REMOVECACHE, GETCACHEINFO

1.16 ARexx Command: GETSTATS

NAME

GETSTATS - List current cache statistics

SYNOPSIS

GETSTATS DRIVE/A

FUNCTION

Return the current statistics for an installed cache.

INPUTS

DRIVE/A - AmigaDOS device name of cache to examine (ex: 'DF0:')

OUTPUTS

"GETSTATS 'DF0:' STEM RESULT." will return the following:

```
RESULT.EFFICIENCY = Current cache efficiency (in %)  
RESULT.USED       = Current cache buffer usage (in %)
```

```
RESULT.UPDATEFREQ = Current update frequency for statistics  
                  window for this cache (in 1/10 sec)
```

NOTES

This function will fail if a statistics window is not showing for the cache you wish to examine. Since there is no ARexx command to open such a statistics window for a cache, this may well turn out to be a problem. A future version of PowerCache will contain ARexx commands to control statistics windows.

SEE ALSO

1.17 ARexx Command: SETSHOWKEY

NAME

SETSHOWKEY - Define the hotkey to show PowerCache's window

SYNOPSIS

SETSHOWKEY HOTKEY/A

FUNCTION

Change the hotkey used for popping up PowerCache's main window.

INPUTS

HOTKEY/A - A valid hotkey specification

OUTPUTS

The old hotkey is returned in RESULT.

NOTES

The new hotkey will be active as soon as this command returns.

SEE ALSO

SETENABLEKEY, SETDISABLEKEY

1.18 ARexx Command: SETENABLEKEY

NAME

SETENABLEKEY - Define the hotkey to enable all caches

SYNOPSIS

SETENABLEKEY HOTKEY/A

FUNCTION

Change the hotkey which is used to enable all caches.

INPUTS

HOTKEY/A - A valid hotkey specification

OUTPUTS

The old hotkey is returned in RESULT.

NOTES

The new hotkey will be active as soon as this command returns.

SEE ALSO

SETSHOWKEY, SETDISABLEKEY

1.19 ARexx Command: SETDISABLEKEY

NAME

SETDISABLEKEY - Define the hotkey to disable all caches

SYNOPSIS

SETDISABLEKEY

FUNCTION

Change the hotkey which is used to disable all caches.

INPUTS

HOTKEY/A - A valid hotkey specification

OUTPUTS

The old hotkey is returned in RESULT.

NOTES

The new hotkey will be active as soon as this command returns.

SEE ALSO

SETSHOWKEY, SETENABLEKEY

1.20 ARexx Command: LOADPREFS

NAME

LOADPREFS - Load a set of preferences

SYNOPSIS

LOADPREFS PREFSFILE REQUEST/S

FUNCTION

Surprisingly enough, this command loads a set of preferences from disk. The filename is optional, and if you leave it out, the default preferences filename will be used.

INPUTS

PREFSFILE - The filename of the file you wish to load. Omit this if you wish to load the default preferences file.

REQUEST/S - Specify REQUEST to verify the operation by popping up a file requester allowing the user to change the name. This will also work if you leave out the PREFSFILE (filename).

OUTPUTS

None.

NOTES

If you leave out both PREFSFILE (the filename) and REQUEST, then this function is 100% identical in operation to the LASTSAVED command.

SEE ALSO

SAVEPREFS, DEFAULTPREFS, LASTSAVED

1.21 ARexx Command: SAVEPREFS

NAME

SAVEPREFS - Save current preferences

SYNOPSIS

SAVEPREFS PREFSFILE REQUEST/S

FUNCTION

Save the current set of preferences in a file. The filename is optional, and if you leave it out, the default preferences filename will be used.

INPUTS

PREFSFILE - The filename of the file you wish to save. Omit this if you wish to save the current preferences as the default preferences file.

REQUEST/S - Specify REQUEST to verify the operation by popping up a file requester allowing the user to change the name. This will also work if you leave out the PREFSFILE (filename).

OUTPUTS

None.

NOTES

If you leave out both PREFSFILE (the filename) and REQUEST, then this function is 100% identical in operation to clicking the Save gadget in PowerCache's main window.

SEE ALSO

LOADPREFS, DEFAULTPREFS, LASTSAVED

1.22 ARexx Command: HIDE

NAME

HIDE - Hide PowerCache's main window

SYNOPSIS

HIDE

FUNCTION

If PowerCache's main window is showing, then hide it. Otherwise, this function is a no-op.

INPUTS

None.

OUTPUTS

None.

NOTES

This function will fail if the cache editing window is showing.

SEE ALSO

SHOW

1.23 ARexx Command: SHOW

NAME

SHOW - Show PowerCache's main window

SYNOPSIS

SHOW

FUNCTION

If PowerCache's window is currently hidden, then show it. Otherwise, this function is a no-op.

INPUTS

None.

OUTPUTS

None.

NOTES

SEE ALSO

HIDE

1.24 ARexx Command: DEFAULTPREFS

NAME

DEFAULTPREFS - Reset preferences to defaults

SYNOPSIS

DEFAULTPREFS

FUNCTION

This function will reset the entire set of preferences to some neutral startup values. This involves resetting all hotkeys and global flags, as well as removing all installed caches.

INPUTS

None.

OUTPUTS

None.

NOTES

Since this function will remove all installed caches, there may be some disk activity. This is because read/write caches need flushing before they can be successfully removed.

SEE ALSO

LOADPREFS, SAVEPREFS, LASTSAVED

1.25 ARexx Command: LASTSAVED

NAME

LASTSAVED - Load the last saved set of preferences

SYNOPSIS

LASTSAVED

FUNCTION

This function will load the set of preferences that were last saved using the Save gadget in PowerCache's main window.

INPUTS

None.

OUTPUTS

None.

NOTES

This operation involves removing existing caches, and installing new ones, so some amount of disk activity can be expected. This is caused when dirty read/write caches are flushed back onto disk.

SEE ALSO

SAVEPREFS, LOADPREFS, DEFAULTPREFS

1.26 ARexx Command: BEEP

NAME

BEEP - Change beep mode

SYNOPSIS

BEEP ON/S OFF/S

FUNCTION

Set the BEEP mode. If BEEP is set to ON, then PowerCache will flash all screens and make a beep when caches are disabled or enabled with hotkeys.

INPUTS

ON/S - Specify this to activate BEEP mode
OFF/S - Specify this to deactivate BEEP mode

OUTPUTS

RESULT will contain the previous state (either 'ON' or 'OFF')

NOTES

Although it doesn't make much sense, it is allowable to specify both ON and OFF, in which case OFF will override ON. Also, you can toggle the current state by omitting both ON and OFF.

SEE ALSO

1.27 ARexx Command: POPUP

NAME

POPUP - Change popup mode

SYNOPSIS

POPUP ON/S OFF/S

FUNCTION

Set the popup mode. When popup mode is active, PowerCache will show its main window everytime it is started.

INPUTS

ON/S - Specify this to activate POPUP mode
OFF/S - Specify this to deactivate POPUP mode

OUTPUTS

RESULT will contain the previous state (either 'ON' or 'OFF')

NOTES

Although it doesn't make much sense, it is allowable to specify both ON and OFF, in which case OFF will override ON. Also, you can toggle the current state by omitting both ON and OFF.

SEE ALSO

1.28 ARexx Command: QUIT

NAME

QUIT - Quit PowerCache

SYNOPSIS

QUIT

FUNCTION

Make PowerCache clean up and exit.

INPUTS

None.

OUTPUTS

None.

NOTES

This operation involves removing existing caches, so some amount of disk activity can be expected. This is caused when dirty read/write caches are flushed back onto disk.

When PowerCache exits, a copy of the current set of preferences is always saved to 'ENV:PowerCache.prefs'. There is no need to use SAVEPREFS before exiting, if you only want PowerCache to go away temporarily.

SEE ALSO

1.29 ARexx Command: USEHELP

NAME

USEHELP - enable or disable online help

SYNOPSIS

USEHELP ON/S OFF/S

FUNCTION

This command can be used to either enable or disable the online help facility, and is functionally equivalent to the 'Use Help?' checkbox gadget in PowerCache's main window.

INPUTS

ON/S - Specify this to activate online help
OFF/S - Specify this to deactivate online help

OUTPUTS

RESULT will contain the previous state (either 'ON' or 'OFF')

NOTES

Although it doesn't make much sense, it is allowable to specify both ON and OFF, in which case OFF will override ON. Also, you can toggle the current state by omitting both ON and OFF.

SEE ALSO