

DOpusM2_ARexx

Dr Greg Perry, Jonathan Potter, Leo Davidson, and Andrew Dunbar Clarke

COLLABORATORS

	<i>TITLE :</i> DOpusM2_ARexx		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Dr Greg Perry, Jonathan Potter, Leo Davidson, and Andrew Dunbar Clarke	February 11, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	DOpusM2_ARexx	1
1.1	Directory Opus Magellan II ARexx Guide.	1
1.2	DOpus Magellan II ARexx Guide History	2
1.3	DOpus Magellan II ARexx Guide Credits	3
1.4	Purchasing Directory Opus Magellan	4
1.5	Official Notice From GPSSoftware: FFD	5
1.6	DOpus Magellan II ARexx: The Port.	6
1.7	DOpus Magellan II ARexx: Results from commands.	6
1.8	DOpus Magellan II ARexx: Introduction.	6
1.9	DOpus Magellan II ARexx: Commands.	7
1.10	DOpus Magellan II ARexx: 'dopus' commands.	7
1.11	DOpus Magellan II ARexx: 'lister' commands.	9
1.12	DOpus Magellan II ARexx: 'dopus addappicon'.	12
1.13	DOpus Magellan II ARexx: 'dopus addtrap'.	14
1.14	DOpus Magellan II ARexx: 'dopus back'.	15
1.15	DOpus Magellan II ARexx: 'dopus checkdesktop'.	15
1.16	DOpus Magellan II ARexx: 'dopus clear'.	15
1.17	DOpus Magellan II ARexx: 'dopus command'.	16
1.18	DOpus Magellan II ARexx: 'dopus desktoppopup'.	17
1.19	DOpus Magellan II ARexx: 'dopus error'.	18
1.20	DOpus Magellan II ARexx: 'dopus front'.	18
1.21	DOpus Magellan II ARexx: 'dopus getdesktop'.	18
1.22	DOpus Magellan II ARexx: 'dopus getfiletype'.	19
1.23	DOpus Magellan II ARexx: 'dopus getstring'.	19
1.24	DOpus Magellan II ARexx: 'dopus matchdesktop'.	21
1.25	DOpus Magellan II ARexx: 'dopus progress'.	21
1.26	DOpus Magellan II ARexx: 'dopus query'.	24
1.27	DOpus Magellan II ARexx: 'dopus query background'.	24
1.28	DOpus Magellan II ARexx: 'dopus query font'.	25
1.29	DOpus Magellan II ARexx: 'dopus query palette'.	25

1.30	DOpus Magellan II ARexx: 'dopus query pens'	26
1.31	DOpus Magellan II ARexx: 'dopus query sound'	27
1.32	DOpus Magellan II ARexx: 'dopus read'	27
1.33	DOpus Magellan II ARexx: 'dopus refresh'	28
1.34	DOpus Magellan II ARexx: 'dopus refresh all'	29
1.35	DOpus Magellan II ARexx: 'dopus refresh background'	29
1.36	DOpus Magellan II ARexx: 'dopus refresh icons'	30
1.37	DOpus Magellan II ARexx: 'dopus refresh lister'	30
1.38	DOpus Magellan II ARexx: 'dopus remappicon'	31
1.39	DOpus Magellan II ARexx: 'dopus remtrap'	31
1.40	DOpus Magellan II ARexx: 'dopus request'	32
1.41	DOpus Magellan II ARexx: 'dopus screen'	32
1.42	DOpus Magellan II ARexx: 'dopus script'	33
1.43	DOpus Magellan II ARexx: 'dopus send'	33
1.44	DOpus Magellan II ARexx: 'dopus set'	33
1.45	DOpus Magellan II ARexx: 'dopus set background'	34
1.46	DOpus Magellan II ARexx: 'dopus set font'	35
1.47	DOpus Magellan II ARexx: 'dopus set palette'	35
1.48	DOpus Magellan II ARexx: 'dopus set pens'	36
1.49	DOpus Magellan II ARexx: 'dopus set sound'	37
1.50	DOpus Magellan II ARexx: 'dopus setappicon'	37
1.51	DOpus Magellan II ARexx: 'dopus version'	38
1.52	DOpus Magellan II ARexx: 'lister add'	38
1.53	DOpus Magellan II ARexx: 'lister addstem'	39
1.54	DOpus Magellan II ARexx: 'lister clear'	41
1.55	DOpus Magellan II ARexx: 'lister clear value'	42
1.56	DOpus Magellan II ARexx: 'lister copy'	43
1.57	DOpus Magellan II ARexx: 'lister clearcaches'	43
1.58	DOpus Magellan II ARexx: 'lister close'	43
1.59	DOpus Magellan II ARexx: 'lister empty'	44
1.60	DOpus Magellan II ARexx: 'lister findcache'	44
1.61	DOpus Magellan II ARexx: 'lister getstring'	45
1.62	DOpus Magellan II ARexx: 'lister iconify'	46
1.63	DOpus Magellan II ARexx: 'lister new'	46
1.64	DOpus Magellan II ARexx: 'lister query'	48
1.65	DOpus Magellan II ARexx: 'lister query active'	50
1.66	DOpus Magellan II ARexx: 'lister query all'	50
1.67	DOpus Magellan II ARexx: 'lister query dest'	51
1.68	DOpus Magellan II ARexx: 'lister query source'	52

1.69	DOpus Magellan II ARexx: 'lister query abort'	54
1.70	DOpus Magellan II ARexx: 'lister query busy'	54
1.71	DOpus Magellan II ARexx: 'lister query case'	55
1.72	DOpus Magellan II ARexx: 'lister query commentlength'	55
1.73	DOpus Magellan II ARexx: 'lister query dirs'	55
1.74	DOpus Magellan II ARexx: 'lister query display'	57
1.75	DOpus Magellan II ARexx: 'lister query entries'	57
1.76	DOpus Magellan II ARexx: 'lister query entry'	59
1.77	DOpus Magellan II ARexx: 'lister query files'	61
1.78	DOpus Magellan II ARexx: 'lister query firstsel'	62
1.79	DOpus Magellan II ARexx: 'lister query flags'	63
1.80	DOpus Magellan II ARexx: 'lister query handler'	64
1.81	DOpus Magellan II ARexx: 'lister query header'	64
1.82	DOpus Magellan II ARexx: 'lister query hide'	65
1.83	DOpus Magellan II ARexx: 'lister query label'	66
1.84	DOpus Magellan II ARexx: 'lister query lock'	66
1.85	DOpus Magellan II ARexx: 'lister query mode'	68
1.86	DOpus Magellan II ARexx: 'lister query namelength'	68
1.87	DOpus Magellan II ARexx: 'lister query numdirs'	69
1.88	DOpus Magellan II ARexx: 'lister query numentries'	69
1.89	DOpus Magellan II ARexx: 'lister query numfiles'	70
1.90	DOpus Magellan II ARexx: 'lister query numsel dirs'	70
1.91	DOpus Magellan II ARexx: 'lister query numsel entries'	71
1.92	DOpus Magellan II ARexx: 'lister query numsel files'	72
1.93	DOpus Magellan II ARexx: 'lister query path'	72
1.94	DOpus Magellan II ARexx: 'lister query position'	73
1.95	DOpus Magellan II ARexx: 'lister query proc'	73
1.96	DOpus Magellan II ARexx: 'lister query seldirs'	74
1.97	DOpus Magellan II ARexx: 'lister query sel entries'	75
1.98	DOpus Magellan II ARexx: 'lister query selfiles'	76
1.99	DOpus Magellan II ARexx: 'lister query separate'	78
1.100	DOpus Magellan II ARexx: 'lister query show'	78
1.101	DOpus Magellan II ARexx: 'lister query sort'	79
1.102	DOpus Magellan II ARexx: 'lister query title'	80
1.103	DOpus Magellan II ARexx: 'lister query toolbar'	81
1.104	DOpus Magellan II ARexx: 'lister query value'	81
1.105	DOpus Magellan II ARexx: 'lister query visible'	82
1.106	DOpus Magellan II ARexx: 'lister query window'	82
1.107	DOpus Magellan II ARexx: 'lister read'	83

1.108DOpus Magellan II ARexx: 'lister refresh'	83
1.109DOpus Magellan II ARexx: 'lister reload'	84
1.110DOpus Magellan II ARexx: 'lister remove'	84
1.111DOpus Magellan II ARexx: 'lister request'	84
1.112DOpus Magellan II ARexx: 'lister set'	85
1.113DOpus Magellan II ARexx: 'lister set busy'	86
1.114DOpus Magellan II ARexx: 'lister set case'	87
1.115DOpus Magellan II ARexx: 'lister set commentlength'	87
1.116DOpus Magellan II ARexx: 'lister set dest'	88
1.117DOpus Magellan II ARexx: 'lister set display'	88
1.118DOpus Magellan II ARexx: 'lister set field'	90
1.119DOpus Magellan II ARexx: 'lister set flags'	91
1.120DOpus Magellan II ARexx: 'lister set header'	92
1.121DOpus Magellan II ARexx: 'lister set handler'	92
1.122DOpus Magellan II ARexx: 'lister set hide'	94
1.123DOpus Magellan II ARexx: 'lister set label'	95
1.124DOpus Magellan II ARexx: 'lister set lock'	95
1.125DOpus Magellan II ARexx: 'lister set mode'	97
1.126DOpus Magellan II ARexx: 'lister set namelength'	97
1.127DOpus Magellan II ARexx: 'lister set newprogress'	98
1.128DOpus Magellan II ARexx: 'lister set off'	100
1.129DOpus Magellan II ARexx: 'lister set path'	100
1.130DOpus Magellan II ARexx: 'lister set position'	101
1.131DOpus Magellan II ARexx: 'lister set progress'	101
1.132DOpus Magellan II ARexx: 'lister set separate'	102
1.133DOpus Magellan II ARexx: 'lister set show'	103
1.134DOpus Magellan II ARexx: 'lister set sort'	104
1.135DOpus Magellan II ARexx: 'lister set source'	105
1.136DOpus Magellan II ARexx: 'lister set title'	106
1.137DOpus Magellan II ARexx: 'lister set toolbar'	107
1.138DOpus Magellan II ARexx: 'lister set value'	107
1.139DOpus Magellan II ARexx: 'lister set visible'	107
1.140DOpus Magellan II ARexx: 'lister select'	108
1.141DOpus Magellan II ARexx: 'lister wait'	109
1.142DOpus Magellan II ARexx: 'command'	109
1.143DOpus Magellan II ARexx: Error Codes.	110
1.144DOpus Magellan II ARexx: Custom Handlers.	111
1.145DOpus Magellan II ARexx: Custom Handlers, The Basics.	112
1.146DOpus Magellan II ARexx: Custom Handlers for Listers.	113

1.147	DOpus Magellan II ARexx: Lister Handlers, Events.	113
1.148	DOpus Magellan II ARexx: Lister Handlers, 'active'	114
1.149	DOpus Magellan II ARexx: Lister Handlers, 'doubleclick'	114
1.150	DOpus Magellan II ARexx: Lister Handlers, 'drop'	115
1.151	DOpus Magellan II ARexx: Lister Handlers, 'dropfrom'	116
1.152	DOpus Magellan II ARexx: Lister Handlers, 'edit'	117
1.153	DOpus Magellan II ARexx: Lister Handlers, 'inactive'	118
1.154	DOpus Magellan II ARexx: Lister Handlers, 'parent'	118
1.155	DOpus Magellan II ARexx: Lister Handlers, 'path'	118
1.156	DOpus Magellan II ARexx: Lister Handlers, 'reread'	119
1.157	DOpus Magellan II ARexx: Lister Handlers, 'root'	119
1.158	DOpus Magellan II ARexx: Lister Handlers, 'snapshot'	119
1.159	DOpus Magellan II ARexx: Lister Handlers, 'unsnapshot'	120
1.160	DOpus Magellan II ARexx: Lister Handlers, Trapped Functions	120
1.161	DOpus Magellan II ARexx: Lister Handlers, AddStem Pop-Ups	121
1.162	DOpus Magellan II ARexx: Lister Handlers, Lister Pop-Ups	121
1.163	DOpus Magellan II ARexx: Custom Handlers for AppIcons.	121
1.164	DOpus Magellan II ARexx: ARexx Modules.	123
1.165	DOpus Magellan II ARexx: Guide Index.	124

Chapter 1

DOpusM2_ARexx

1.1 Directory Opus Magellan II ARexx Guide.

Directory Opus Magellan II ARexx Guide
©1998 GPSoftware, All Rights Reserved.

Introduction

Guide Credits

The ARexx Port

Guide History

Results From Commands

ARexx Commands

Purchasing Opus Magellan

-

dopus

-

lister

Index

-

command

Error Codes

Custom Handlers

-

The Basics

-

For Listers

(events list)

-

For AppIcons

ARexx Modules

This guide is best viewed with the OS3.0 (V39) or higher ←

AmigaGuide

datatype as it makes extensive use of coloured text. Things will be much less readable with earlier versions of AmigaGuide -- Upgrade!

This Amigaguide document was originally developed for Opus 5.5 by Leo Davidson and has been updated to support features in Opus Magellan 5.66 as of 29/Oct/1997 by Andrew Dunbar, GPSoftware Brisbane, Australia.

Updated again to support features of DOpus Magellan II as of 12.9.1998 by Dave Clarke, Melbourne Australia.

1.2 DOpus Magellan II ARexx Guide History

v50.0 -- 21/May/1995

Initial version based on the Directory Opus 5.0 Manual.
Never publicly released or properly polished off.

v55.0 -- 21/Sep/1996 (What is it about the 21st?)

Completely re-done. Based on the Directory Opus 5.5 Manual with numerous additions and several corrections.

v55.1 -- 22/Sep/1996

Lister events

snapshot
and
unsnapshot
were missing.

Fixed a small spelling error (I'm sure there are more :-)).

Rearranged the main

contents page
slightly.

v55.2 -- 2/Oct/1996

Second returned argument of
lister query position
now documented.

Improved

lister clear flags
section (description of flags).

Added a special note about

dropfrom's
to the main Opus window.

v56.1 -- 29/Oct/1997

Guide document updated to support features in Opus Magellan 5.66 as of 29/Oct/1997 by Andrew Dunbar, GPSoftware Brisbane, Australia.

v57.0 -- 29/Jul/1998

Guide document updated to support features in Opus Magellan II as of Dave Clarke, Beta-tester Melbourne, Australia.

v57.1 -- 1/Sep/1998

Found a few typo's, added the FTP ARexx commands, added 'lister2 popups' to Custom Handler section, found a few missing index links.

Dave Clarke, Beta-tester Melbourne, Australia.

v57.2 -- 1/Sep/1998

Removed the FTP ARexx commands, it doesn't have any.

Stupid me :-/

Dave Clarke, Beta-tester Melbourne, Australia.

v57.3 -- 12/Sep/1998

Found a few typos, I was missing from the credits for some reason :-/

Updated the distributors list.

Dave Clarke, Beta-tester Melbourne, Australia.

v57.4 -- 12/Sep/1998

Added 'lister query/set commentlength', fixed 'dopus/lister request'.

Dave Clarke, Beta-tester Melbourne, Australia.

v57.5 -- 16/Oct/1998

Typos in 'lister set/clear value' fixed. Typo in 'dopus set sound' fixed.

Dave Clarke, Beta-tester Melbourne, Australia.

1.3 DOpus Magellan II ARexx Guide Credits

Credits

Directory Opus and this guide are © Copyright Jonathan Potter and GPSSoftware, Brisbane, 1998. All Rights Reserved. The Directory Opus software was written by Jonathan Potter and designed by Jonathan Potter and Greg Perry. OpusFTP and AFC modules were programmed by Andrew Dunbar and Greg Perry. The manual which this guide was based on was written by Greg Perry and Jonathan Potter. AmigaGuide conversion, corrections and additional material by Leo Davidson. Lot's more additions and corrections done by Dave Clarke and his merry band of ELF's, (ELectronic Fingers).

As well as being much more user-friendly than the ARexx section of the printed manual, this guide contains everything in the manual and more.

Jonathan Potter:

email: jpotter@lss.com.au

wwwweb: <http://www.lss.com.au>

Dr Greg Perry / GPSSoftware:

email: greg@gpsoft.com.au

wwwweb: <http://www.gpsoft.com.au>

Leo 'Nudel' Davidson / Gods'Gift Utilities:

email: leo.davidson@keble.oxford.ac.uk

wwwweb: <http://users.ox.ac.uk/~kebl0364>

Dave Clarke:

email: 4wd@connexus.net.au
 wwweb: Waste of bandwidth, who needs it :)

There are also several mailing lists for the discussion of Directory Opus 5 related issues. Mail Greg Perry for more information (you will need to quote your serial number).

Software used in Guide Creation

----- - - - -
 Guide checked using CheckGuide v1.0 (C)1994 Eddy Carroll.
 Guide edited in CygnusEd Professional v3.5 (C)1987-1996 Cygnus Software.
 Guide edited in TurboText v2.0 (C)1990-1994 Martin Taillefer
 Guide index created by a CEed macro & Commodore's "sort" command. :-)
 v30 Multiview/AmigaGuide datatype used for viewing.
 Everything tied together and launched from Directory Opus 5, what else?

No "AmigaGuide editors" were used. IMO they're as much use as HTML editors.

And the idiot who invented TAB characters should be shot, hung, drawn and quartered...and then he should be really hurt.

Recent musical support (in alphabetical)

----- - - - -
 (GP Note: This was Leo's taste at the time this guide was originally created.
 No doubt they have changed by now;-)

Bjork	(Debut/Post),
BT	(Ima) This is amazing!,
Corrosion Of Conformity	(Wiseblood),
Flotsam & Jetsam	(Drift),
Fugees	(The Score) "I'll be right out...",
Jamiroquai	(Travelling Without Moving),
Metallica	(Load),
Nine Inch Nails	(Pretty Hate Machine/Quake),
Nirvana	(In Utero/Unplugged),
Rage Against The Machine	(Evil Empire),
The Prodigy	(Experience/Music For The Gilted Generation),
Tool	(Opiate/Undertow).

Additional musical support (required by Dave :)

Allanah Miles	(Allanah Miles),
NoiseWorks	(Touch),
Alanis Morissette	(Jagged Little Pill),
Pink Floyd	(Momentary Lapse of Reason)

1.4 Purchasing Directory Opus Magellan

Purchasing Directory Opus Magellan

----- - - - -
 Directory Opus Magellan can be ordered direct from GPSoftware, or from one of the distributors listed below.

If you already own a previous version of Directory Opus you may be able to upgrade to the current version for a reduced fee.

For further information about Directory Opus 5 Magellan and purchasing,
please mail

Greg Perry

The Amiga is in a perilous state right now, and, apart from the ageing OS, all that we have left are devoted developers and their innovative programs, and a whole lot of hopes and dreams. Developers like GPSoftware keep this platform alive by keeping it at the cutting edge with software like Directory Opus 5.5. If you are using a pirate version of Directory Opus, send a signal that you want there to be further versions by purchasing it. Don't let the bridge to our dreams die. The only way you can support the Amiga is by supporting Amiga Developers.

Please note that France-Festival-Distribution (FFD) no longer distribute any GPSoftware products. Read this
press release
for further details.

Dr Greg Perry	Ph/fax:	+61 7 33661402
GPSoftware	WWW:	http://www.gpsoft.com.au/
PO Box 570		
Ashgrove		
Qld		
Australia 4060		

Wizard Developments	Phone:	+44 (0)181 303 1800
PO BOX 123	Fax:	+44 (0)181 303 1861
Sidcup	WWW:	http://wizard-d.demon.co.uk/
Kent DA15 9ZY		
ENGLAND		

Schatztruhe	Phone:	+49 201 788 778
Veronikastr 33	Fax:	+49 201 798 447
45131 Essen	WWW:	http://www.schatztruhe.de/
GERMANY		

Software Hut	Phone:	+1 610 586 5701
313 Henderson Drive	Fax:	+1 610 586 5706
Sharon Hill	WWW:	http://www.softhut.com/
PA 19079		
USA		

1.5 Official Notice From GPSoftware: FFD

GPSoftware French Distribution - France-Festival-Distribution

After recent actions by FFD, we have officially withdrawn all of our products and licences from FFD and refuse to let them ever be associated with any of our products again. I have advised all of my other distributors that they may not supply FFD with any of our products. Further, being the

copyright holder of trademarks, product names and other copyrights for our products, we officially withdraw permission for FFD or any associated company to use the name Directory Opus, GPFax, or GPSoftware in any advertising or related activities.

Because of unreconciled differences, FFD is no longer a distributor or reseller of GPSoftware products and GPSoftware will no longer honour warranty or support for any of our products purchased from FDD after 30th June 1996.

Dr Greg Perry, GPSoftware 11th September 1996

1.6 DOpus Magellan II ARexx: The Port.

The DOpus5 ARexx Port.

The Directory Opus 5 ARexx port name is DOPUS.x , where x is the invocation count of the program (the first and most often used one is DOPUS.1). Since ARexx scripts launched from Directory Opus do not automatically inherit the command address, you may want to use the {Qp} command sequence in Opus functions (this is described elsewhere in the printed manual).

1.7 DOpus Magellan II ARexx: Results from commands.

Results from commands.

If a command returns a value or information, the data will generally be returned in the RESULT variable. The only exceptions to this are the

```
dopus getdesktop
,
dopus getstring
and
lister getstring
commands
```

which return information in the special DOPUSRC variable.

Error
codes are returned in the RC variable.

You must include the line "OPTIONS RESULTS" near the top of your script to enable the RESULT variable. See an ARexx manual for more information.

1.8 DOpus Magellan II ARexx: Introduction.

Introduction to ARexx support in Directory Opus 5.

The ARexx command set is very comprehensive and flexible. Almost complete control of listers in name mode is offered, along with the ability to launch functions and commands. You can add your own commands to Opus via ARexx scripts which are loaded automatically - the commands appear identical to the built-in functions. You can even replace the default commands with your own.

There is also a powerful custom handler ability. This allows your ARexx program to receive messages from Opus for a variety of user actions, including lister and icon events. See the section on

Custom Handlers
for more information.

1.9 DOpus Magellan II ARexx: Commands.

ARexx Commands

For simplicity, the Directory Opus 5 command set is arranged in a hierarchical structure, with only three main (or base) commands:-

dopus

The first base command is `dopus`. This is a general purpose command, and allows you to perform functions not falling into the other categories.

lister

The next base command, `lister`, allows you to control listers and entries within listers.

command

The third base command is `command`. This allows you to call the internal commands of Directory Opus 5 from an ARexx script. The commands execute exactly as if they had been run from a custom button or menu.

1.10 DOpus Magellan II ARexx: 'dopus' commands.

'dopus' commands. (Alphabetical listing)

These are a general purpose commands, allowing you to perform functions not falling into the other categories.

dopus addappicon
dopus addtrap
dopus back
dopus checkdesktop
dopus clear
dopus command
dopus desktoppopup
dopus error
dopus front
dopus getdesktop
dopus getfiletype
dopus getstring
dopus matchdesktop
dopus progress
dopus query
 --->
background
font
palette
pens
sound
dopus read
dopus refresh
 --->
all
background
icons
lister
dopus remappicon
dopus remtrap
dopus request

```
dopus script
dopus screen
dopus send
dopus set
  --->
background
font
palette
pens
sound
dopus setappicon
dopus version
```

1.11 DOpus Magellan II ARexx: 'lister' commands.

'lister' commands. (Alphabetical listing)

----- -- -- --
These commands allow you to control listers and entries within listers.

```
lister add
lister addstem
lister copy
lister clear
value
lister clearcaches
lister close
lister empty
lister findcache
lister getstring
lister iconify
lister new
```


lister query

--->

active

all

dest

source

--->

abort

busy

case

commentlength

dirs

display

entries

entry

files

firstsel

flags

handler

header

hide

label

lock

mode

namelength

numdirs

numentries

numfiles

numseldirs

numselentries

numselfiles

path
proc
position
seldirs
selentries
selfiles
separate
show
sort
title
toolbar
value
visible
window
lister read
lister refresh
lister reload
lister remove
lister request
lister set
 --->
busy
case
commentlength
dest
display
field
flags
handler

header
hide
label
lock
mode
namelength
newprogress
off
path
position
progress
separate
show
sort
source
title
toolbar
value
visible
lister select
lister wait

1.12 DOpus Magellan II ARexx: 'dopus addappicon'.

```
dopus addappicon <port> <label> <id> [pos <x> <y>] [icon <filename <↵>
>]
[quotes] [info] [snap] [close] [local] [locked]
[menu <stem>] [base <base>]
```

This command allows you to add your own AppIcons to the Opus (and optionally the Workbench) screen from ARexx. You can specify an icon image to use, the label of the icon and the position on the screen, as well as several other parameters. You can also specify the items for the pop-up menu for the new icon. Opus sends messages to the message port you specify

- see the example scripts provided for more information on how to receive and process these messages.

The parameters for `addappicon` are:-

```

port    -    the name of the port messages are sent to
label   -    the icon label (text displayed under the icon)
id      -    your own ID for the icon; this is returned in
             messages
pos     -    position for the icon (xy coordinates)
icon    -    optional pathname of icon file to use
             (without the .info suffix)
quotes  -    specify this keyword if you want filenames
             quoted when sent to the message port
info    -    if you want "Information" to work on this icon
snap    -    if you want "Snapshot" to work on this icon
close   -    if you want "Close" instead of "Open" in
             the pop-up menu
local   -    if you want the icon to be local to Opus
             (and not appear on Workbench)
locked  -    the icon will start out locked in position,
             unable to be moved
menu    -    name of a stem variable containing
             your own menu items
base    -    allows you to specify the base ID of
             messages sent from the pop-up menu

```

The `menu` parameter allows you to specify your own items for the icon's pop-up menu. The stem variable must be in the following format:

```

stem.COUNT - number of items
stem.0     - item 1
stem.1     - item 2
etc.

```

If you specify "---" as an item, a separator bar will appear.

When you receive a message that the user has selected one of these menu items, the message will contain the ID of the item. This is the value corresponding to the item's position in the stem array (eg 0 for item 1, 1 for item 2, etc). If the base field is specified, the value given for the base will be added to this ID.

This command returns an `appicon` handle in
`RESULT`
if it is successful.

This same handle can be passed to
`dopus setappicon`
to modify the icon,
and must be passed to
`dopus remappicon`
to remove the appicon when you
are finished.

If you do not specify an icon file to use then the system's default "tool" icon will be used for the image.

Messages from the appicon are sent to the port you specified. The messages are structured in much the same way as messages sent from listers, so there's no reason you shouldn't be able to use the same code for both. For more information, see the [Custom Handlers](#) section.

See also:

`dopus setappicon`

`dopus remappicon`

[AppIcon Handlers](#)

1.13 DOpus Magellan II ARexx: 'dopus addtrap'.

```
dopus addtrap (abort|<command>) <handler> [port <portname>]
```

This command allows your script to trap the progress bar's abort button or any Opus internal command.

Specify the `abort` keyword to trap the abort message, or the name of the internal command you wish to trap.

`handler` is the name of your custom handler message port. If you pass the name of a message port with the optional `port` keyword, the message will be sent to that port instead of your usual handler port. This can be very useful when used with `abort` if your handler is busy doing something synchronously when the abort is pressed. If the abort port is on a separate process, it may be able to interrupt the main handler process using signals for instance.

You can also add a trap for all internal commands using an asterisk as a wildcard, for example: `dopus addtrap * myhandler`

The

```
dopus remtrap
command is used to remove trapped functions.
```

See the

[Custom Handlers](#) section for more information on the messages

sent.

See also:

`dopus remtrap`

[Trapped Commands](#)

1.14 DOpus Magellan II ARexx: 'dopus back'.

```
dopus back
```

This command moves the Directory Opus 5 window (and screen) to the rear of the display.

See also:

```
dopus front
```

1.15 DOpus Magellan II ARexx: 'dopus checkdesktop'.

```
dopus checkdesktop
```

This command is used when your script has just copied one or more files to a directory. You pass it the destination path that you copied the files to, and Opus compares this path with the location of the Desktop folder. If they are the same, Opus will then scan the Desktop folder and update the icons on-screen if necessary.

See also:

```
dopus getdesktop
```

```
dopus matchdesktop
```

1.16 DOpus Magellan II ARexx: 'dopus clear'.

```
dopus clear <item>
```

This command allows you to clear a setting, it the equivalent of doing a

```
dopus set  
with a null string.
```

For example,

```
+ dopus clear sound Startup  
+ dopus clear background desktop
```

See also:

```
dopus set
```

1.17 DOpus Magellan II ARexx: 'dopus command'.

```
dopus command <name> program <scriptname> [desc <description>]
[template <template>] [source] [dest] [private]
[help <help file>] [handler] [temp]
[ext <menu name> type <filetype>] [remove]
```

This command provides the ability to add new internal commands to Directory Opus, or to replace existing commands. It is generally called from within the init function of an Opus ARexx module and the program parameter will be the name of that module, without the ".dopus5" extension. The program field is mandatory, and Opus will run the script name you provide here whenever this function is invoked.

Use the name parameter to specify what the command should be called, and, optionally, use the desc parameter to define a description of the command which will appear in command lists.

Using the handler parameter will allow the item to be displayed only if a custom handler matching the program parameter is present for the lister.

The remove flag allows you to remove any command specified with name .

temp will allow you to add a 'temporary' command, which does not have an external command file. This command will then do nothing unless it is trapped with the

```
dopus addtrap
command. Use the remove flag
```

of the

```
dopus command
to remove it when you are done.
```

Using the private flag will stop the temporary command from being seen in the internal command list.

The template is not parsed for you in any way -- it is simply shown to the user when they request a template for your command. It is up to your script to handle the argument string sent to it.

If you give the source or dest keyword (or both), your command will not run unless there is a lister of the given type. When run, the lister will automatically go busy, and the "source" or "dest" argument given to your script will contain the lister's handle. (Each argument still exists when the relevant keyword is not given, but is always "0").

Note that the standard ARexx module example code parses the source lister handle into a variable called source and the destination handle into dest. Be careful that when you specify the source keyword for dopus command that you put it in quotes ("source"), otherwise the contents of the source variable may be used in place of the actual word "source". Take similar precautions with the "dest" word.

You can specify a help file for the new command with the `help` parameter.

PopUpExtensions

You can add a command to the pop-up menu of icons of a given type, or to the lister itself, by specifying a filetype name or ID for the `type` parameter and the string to appear as the menu item for the `ext` parameter.

To match more than one filetype you can specify the `type` parameter multiple times (but you can only have one `ext`). You can also specify one of the following keywords for the `type` parameter:

`all, disk, drawer, tool, project, trash, baddisk, leftout, lister, lister2.`

Specifying `lister` for the `type` parameter will add items to the lister pop-up menu rather than the icon pop-up menu.

Specifying `lister2` for the `type` parameter will add items to the pop-up menu available from the SRCE/DEST display in the lister status bar.

If you add menus to a filetype at priority `-124`, they will only be shown if no other filetype matched. This allows you to have a "default filetype" with menus that will only be shown if no other filetype menus are displayed.

Commands which are intended for pop-up menus are unlikely to be of general use. You can use the `private` keyword to hide the command (it will not appear in any list shown to the user). (This is not restricted to PopUpExtensions.)

See

ARexx modules
and the example scripts for more information.

1.18 DOpus Magellan II ARexx: 'dopus desktoppopup'.

`dopus desktoppopup [flags]`

This command triggers the Desktop popup menu at the current mouse position, and returns a value in

RC
indicating the user's selection.

`flags` specifies what options will be disabled, by default all options are available. Add the flag values as appropriate.

Add to disable	Item	Result Code
2	Create Left-Out	1
4	Copy to Desktop	2
8	Move to Desktop	3

If the user cancels the operation,

RC
will have the value 0.

For example,
+ dopus desktoppopup 6
> 2

See also:

dopus getdesktop

1.19 DOpus Magellan II ARexx: 'dopus error'.

dopus error <code>

This command is used to retrieve meaningful error messages when passed an Opus ARexx error code.

For example,
+ dopus error 1
> File rejected by filters

+ dopus error 10
> Invalid lister handle

See also:

Errors

1.20 DOpus Magellan II ARexx: 'dopus front'.

dopus front

This command moves the Directory Opus 5 window (and screen) to the front of the display.

See also:

dopus back

1.21 DOpus Magellan II ARexx: 'dopus getdesktop'.

dopus getdesktop

This command allows you to find the path of the desktop folder. It returns

the path in

```

        RESULT
    , and returns a value in
    DOPUSRC
    that indicates the

```

user's desktop settings:

```

0    desktop popup disabled
1    no default action
2    default : create leftout
3    default : move
4    default : copy

```

For example,

```

+ dopus getdesktop
> HD0:Opus5/Desktop/

```

See also:

```
dopus checkdesktop
```

```
dopus matchdesktop
```

1.22 DOpus Magellan II ARexx: 'dopus getfiletype'.

```
dopus getfiletype <filename> [id]
```

This command allows you to query a file to see if it is recognised by Directory Opus 5. `filename` is the name of the file, including the full path. By default, if the file is recognised the Filetype description string will be returned in

```

        RESULT
    . If you specify the ID keyword, the

```

Filetype ID will be returned instead.

For example,

```

+ dopus getfiletype ram:testfile.lha
> LHA Archive
+ dopus getfiletype ram:picture.jpg id
> JPEG

```

1.23 DOpus Magellan II ARexx: 'dopus getstring'.

```
dopus getstring <text> [secure] [<length>] [<default>] [<buttons>]
```

This command allows you to prompt the user to input a text string.

`text` is a string of text to be displayed in the requester, and should be surrounded by quotes if it contains spaces.

The `secure` keyword causes the string to be displayed as asterisks ("`*`"), which can be useful for passwords.

`length` is the maximum length of string to accept and defaults to 80 if not specified.

`default` is the default value of the string; that is, the text you wish to initially appear in the field.

`buttons` are the buttons you wish the requester to have; each button should be separated by a vertical bar character ("`|`"). If the `buttons` parameter is omitted the requester will have a single button marked "OK".

For example,

```
> dopus getstring "Please enter some text" 40 "" Okay|Cancel'
```

Note the quotes around the entire argument string.

You can have multiple lines in the requester text by putting the "return" character at the end of lines.

This would display a requester with the string "Please enter some text", a maximum input length of 40 characters, an empty default sting, and buttons labelled "Okay" and "Cancel".

The string (if any) is returned in

`RESULT`

. The cardinal number of the

selected button is returned in the special variable

`DOPUSRC`

, but the

last (rightmost) gadget is always number 0. In the above example, if the user clicked "Okay",

`DOPUSRC`

would contain 1, and if the user clicked

"Cancel" it would contain 0. If the RETURN key is used in the string gadget to accept the text, then the

`DOPUSRC`

variable will contain the value -1.

This command and the

`lister getstring`

command are the only ones that use

the

`DOPUSRC`

variable currently, but this may change in the future.

Please note that previous versions of Opus 5 did not clear

`RESULT`

if an

empty string was entered. Make sure that this change does not affect your scripts. Also note that the

`DOPUSRC`

variable did not exist in the

original version of Opus 5.

See also:

```
dopus request
lister getstring
lister request
```

1.24 DOpus Magellan II ARexx: 'dopus matchdesktop'.

```
dopus matchdesktop <path>
```

This command allows you to match a supplied path against the desktop folder.

```
RC
will be set according to whether a match was made.
```

For example,

```
+ dopus matchdesktop 'dopus5:desktop/'
> 1
```

See also:

```
dopus getdesktop
dopus checkdesktop
```

1.25 DOpus Magellan II ARexx: 'dopus progress'.

```
dopus progress [name] [file] [info] [info2] [info3] [bar] [abort]
```

This opens a progress indicator centred on the Opus screen. It is similar to the

```
lister set newprogress
command which you should use if you want
the indicator centred on a lister window instead.
```

```
name - allocates space for filename display
file - allocates space for file progress display
info - allocates space for information line 1
info2 - allocates space for information line 2
info3 - allocates space for information line 3
bar - allocates space for progress bar
abort - adds "Abort" gadget
```

Progress windows that show both the bar graph and the file progress will have the graph and file displays swapped around. This means that instead of the graph showing the percentage of files copied, and a 'xx%' display

showing the progress of that file, the graph shows the file progress and a 'xxx of yyy' display gives overall information.

A handle is returned in

RESULT

which you should use to control the

indicator with the commands below.

For example,

```
+ dopus progress name file info bar abort
> 137025148
```

The following commands are for use once your progress window is open.

```
dopus progress <handle> abort
```

handle is the handle of the progress indicator in question, obtained when you first open it.

This is similar to

```
lister query abort
```

```
command for lister-based progress
```

indicators. It returns 1 if the user has clicked the "Abort" gadget of the indicator, and 0 otherwise.

For example,

```
+ dopus progress 137025148 abort
> 0
```

```
dopus progress <handle> off
```

handle is the handle of the progress indicator in question, obtained when you first open it.

This is similar to

```
lister clear progress
```

```
for lister-based indicators. It
```

closes the progress indicator window.

For example,

```
+ dopus progress 137025148 off
```

```
dopus progress <handle> name <filename>
```

handle is the handle of the progress indicator in question, obtained when you first open it.

If the progress bar was opened with the name parameter, this will set the current filename to filename .

For example,

```
+ dopus progress 137025148 name 'myfile.txt'
```

```
dopus progress <handle> file <total> <count>
```

`handle` is the handle of the progress indicator in question, obtained when you first open it.

If the progress indicator was opened with the file but not the bar parameter, this will set the total number of files and the number of the current file. This is shown as 'xx%' in the top right of the requester.

If the progress indicator was opened with both the file and bar parameters, this will set the total number of bytes and the current byte count. This is shown in the bar graph part of the requester.

```
+ dopus progress 137025148 file 12 4
```

```
dopus progress <handle> <info|info2|info3> <text>
```

`handle` is the handle of the progress indicator in question, obtained when you first open it.

`text` is a text string to be displayed between the filename and the bar graph of the progress indicator. You can only use this if you allocated this space when you created the progress bar.

For example,

```
+ dopus progress 137025148 info "From 'T' to 'Ram:'"
```

```
dopus progress <handle> bar <total> <count>
```

`handle` is the handle of the progress indicator in question, obtained when you first open it.

If the progress indicator was opened with the bar but not the file parameter, this will set the total number of bytes and the current byte count. This is shown in the bar graph part of the requester.

If the progress indicator was opened with both the file and bar parameters, this will set the total number of files and the the current file count. This is shown as 'xxx of yyy' in the top right of the requester.

For example,

```
+ dopus progress 137025148 bar 1024 100
```

```
dopus progress <handle> title <text>
```

`handle` is the handle of the progress indicator in question, obtained when you first open it.

`text` is a text string to be displayed in the title bar of the progress indicator.

For example,

```
+ dopus progress 137025148 title 'Copying...'
```

See also:

```
lister set newprogress
```

```
lister set progress
```

1.26 DOpus Magellan II ARexx: 'dopus query'.

```
dopus query <item>
```

This command allows you to get information about the current Opus settings and complements the

```
dopus set  
command.
```

See also:

```
dopus query background
```

```
dopus query font
```

```
dopus query palette
```

```
dopus query pens
```

```
dopus query sound
```

```
dopus set
```

```
dopus set background
```

```
dopus set font
```

```
dopus set palette
```

```
dopus set pens
```

```
dopus set sound
```

1.27 DOpus Magellan II ARexx: 'dopus query background'.

```
dopus query background <desktop|lister|req>
```

This command allows you to query the current background picture for either the Desktop, Listers, or Requesters. The path and filename will

be returned in

```
RESULT
```

```
.
```

For example,

```
+ dopus query background desktop  
> DOpus5:Backgrounds/Stonewash.iff
```

See also:

```
dopus query
```

```
dopus refresh background
```

```
dopus set
```

```
dopus set background
```

1.28 DOpus Magellan II ARexx: 'dopus query font'.

```
dopus query font <type>
```

Returns information about the current Opus fonts in use.

type can be one of the following:

```
screen - The font Opus uses when on it's own screen.  
listers - Current font used for the listers file display.  
iconsd - Font used for the Desktop icon display.  
iconsw - Font used for icon display in windows.
```

For example,

```
+ dopus query font screen  
> "XHelvetica.font" 11
```

See also:

```
dopus query
```

```
dopus refresh
```

```
dopus set
```

```
dopus set font
```

1.29 DOpus Magellan II ARexx: 'dopus query palette'.

```
dopus query palette
```

Returns the current Opus palette information as a group of 16 hex numbers. The last 8 represent the Opus colours, the first 8 represent the system colours when Opus is on it's own screen.

For example,

```
+ dopus query palette
> 0x909090 0x000000 0xF0F0F0 0x606DA0 0x808080 0xA0A0A0 0xA09070 0xF0A090
   0xB4E494 0xC8FC00 0x004C94 0xFCFCD4 0xD8D8D8 0x183454 0xE8E8E8 0x505050
```

See also:

```
dopus query
```

```
dopus refresh
```

```
dopus set
```

```
dopus set palette
```

1.30 DOpus Magellan II ARexx: 'dopus query pens'.

```
dopus query pens <type>
```

Returns information about the current Opus pen colours used for the various display components.

type	Returned values
icons	<dfg> <dbg> <dstyle> <wfg> <wbg> <wstyle>
files	<fg> <bg>
dirs	<fg> <bg>
selfiles	<fg> <bg>
seldirs	<fg> <bg>
devices	<fg> <bg>
assigns	<fg> <bg>
source	<fg> <bg>
dest	<fg> <bg>
gauge	<normal> <full>
user	<count>

The pen number is mapped in the following way:

```
1-4    bottom four system colours
5-8    top four system colours
9-16   Opus user pens
```

For example,

```
+ dopus query pens icons
> 8 1 3 1 12 2
```

```
+ dopus query pens gauge
> 11 13
```

See also:

```
dopus query
dopus refresh
dopus set
dopus set pens
```

1.31 DOpus Magellan II ARexx: 'dopus query sound'.

```
dopus query sound <event>
```

This command allows you to query the current sound for the specified event . The path and filename will be returned in
RESULT

.

For example,

```
+ dopus query sound Startup
> DOpus5:Sounds/Danger.8svx
```

See also:

```
dopus query
dopus set
dopus set sound
```

1.32 DOpus Magellan II ARexx: 'dopus read'.

```
dopus read [delete] [pos <x/y/w/h>] [<handle>] [hex|ansi|smart] < ↔
filename>
```

```
dopus read <handle> quit
```

The read command is provided to allow you greater control over the Opus text viewer. It basically allows you to open a viewer and view multiple files in it one by one before closing it again.

The pos keyword will allow you to open the viewer at a specific position on the screen.

When you do not specify a handle a new viewer is opened and its viewer handle is returned in

RESULT

. You use the viewer handle in much the same way as you would a lister handle, you can either get it to load new files or quit .

Normally the viewer will default to ASCII mode to view files, you can specify another mode to use by using one of the keywords: hex , ansi or smart .

If you give the delete keyword the file will be deleted once the viewer has been closed. This is useful for displaying temporary files.

For example,

```
+ dopus read ram:file1.txt
> 121839492
+ "dopus read pos 10/10/400/200 ram:file2.txt"
> 121839492
+ dopus read 121839492 ram:file2.txt
> 121839492
+ dopus read hex C:Info
> 121839492
+ dopus read 121839492 quit
> 0
```

1.33 DOpus Magellan II ARexx: 'dopus refresh'.

```
dopus refresh <item> [options]
```

This command lets you refresh the relevant item in the Directory Opus display.

The various refresh commands are needed after modifying some settings with the

```
dopus set
command.
```

See also:

```
dopus refresh all
```

```
dopus refresh background
```

```
dopus refresh icons
```

```
dopus refresh lister
```

```
dopus set
```

1.34 DOpus Magellan II ARexx: 'dopus refresh all'.

```
dopus refresh all
```

Refreshes the entire Directory Opus display.

For example,

```
+ dopus refresh all
```

See also:

```
dopus refresh
```

```
dopus refresh background
```

```
dopus refresh icons
```

```
dopus refresh lister
```

```
dopus set
```

1.35 DOpus Magellan II ARexx: 'dopus refresh background'.

```
dopus refresh background [custom]
```

This command causes the Opus display backgrounds to be updated after the execution of a

```
dopus set background  
command.
```

If you used the `custom` keyword on the

```
dopus set background  
then you
```

must use it here, if you don't the pictures will revert back to the environment settings.

For example,

```
+ dopus refresh background  
+ dopus refresh background custom
```

See also:

```
dopus query background
```

```
dopus refresh
```

```
dopus refresh all
```

```
dopus refresh icons
```

```
dopus refresh lister
```

```
dopus set background
```

1.36 DOpus Magellan II ARexx: 'dopus refresh icons'.

```
dopus refresh icons
```

This command refreshes the icon display in all listers and the Opus Desktop.

For example,

```
+ dopus refresh icons
```

See also:

```
dopus refresh
```

```
dopus refresh all
```

```
dopus refresh background
```

```
dopus refresh lister
```

```
dopus set
```

1.37 DOpus Magellan II ARexx: 'dopus refresh lister'.

```
dopus refresh lister [full]
```

This command refreshes the lister displays.

The `full` flag causes everything in the listers to be refreshed, instead of just the lister itself.

For example,

```
+ dopus refresh lister  
+ dopus refresh lister full
```

See also:

```
dopus refresh
```

```
dopus refresh all
```

```
dopus refresh background
```

```
dopus refresh icons
```

```
dopus set
```

1.38 DOpus Magellan II ARexx: 'dopus remappicon'.

```
dopus remappicon <handle>
```

Removes an appicon that was added previously with the `dopus addappicon` command.

`handle` is the value returned by `dopus addappicon`.

See also:

```
dopus addappicon
```

```
dopus setappicon
```

AppIcon Handlers

1.39 DOpus Magellan II ARexx: 'dopus remtrap'.

```
dopus remtrap (abort|<command>) <handler>
```

Disables the trapping of the progress bar's abort button or the specified Opus internal command as initiated with the `dopus addtrap` command.

If you specify "*" as the command, all traps added for this handler will be removed.

`handler` is the name of the message port as specified in the

```
dopus addtrap  
command.
```

See also:

```
dopus addtrap
```

Trapped Commands

1.40 DOpus Magellan II ARexx: 'dopus request'.

```
dopus request <text> <buttons>
```

This command allows you to request a choice from the user.

`text` is a string of text to be displayed in the requester.

`buttons` are the buttons you wish the requester to have; each button should be separated by a vertical bar character ("|").

For example,

```
+ dopus request "Please choose an option" Option 1|Option 2|Option 3'
```

Note the quotes around the entire argument string.

You can have multiple lines in the requester text by putting the "return" character at the end of lines.

This would display a requester with the string "Please choose an option", and three buttons labelled "Option 1", "Option 2" and "Option 3".

The cardinal number of the selected button is returned in

```
RC
```

```
. The last
```

button supplied ("Option 3" in this case) is designated a Cancel button, and so returns the value 0. Therefore, the values returned by this example are 1, 2 and 0 respectively.

See also:

```
dopus getstring
```

```
lister request
```

```
lister getstring
```

1.41 DOpus Magellan II ARexx: 'dopus screen'.

```
dopus screen
```

This command returns information about the Opus screen in

```
RESULT
```

```
in the
```

following format:-

```
<name> <width> <height> <depth> <barheight> <lister width> <lister height>
```

If Opus is iconified it does not have a screen. In this case

```
RC
```

```
will be
```

set to 5. You can use this to find out whether or not Opus is currently

iconified.

```
For example,  
+ dopus screen  
> DOPUS.1 640 512 2 10 308 341
```

The `barheight` value is useful if you intend to open listers or other windows just below the screen title bar.

Any screen depth of more than 8 bits, (256 colours), will be reported as 8.

1.42 DOpus Magellan II ARexx: 'dopus script'.

```
dopus script <name> [data]
```

This command allows you to trigger both internal and custom scripts.

`name` is the name of the script, (case insensitive).

`data` is an optional string that is passed to the script in the `{Qa}` parameter.

```
For example,  
+ dopus script 'DOpus5:ARexx/ArcDir.dopus5' 'RAM:foo.lha'
```

1.43 DOpus Magellan II ARexx: 'dopus send'.

```
dopus send <port name> <string>
```

This command does nothing to Opus itself, but instead makes it easy for you to send a string (of any length) to another ARexx task (via a message). The string is supplied in `Arg0` of the message sent to the named port.

See the
Custom Handlers
section.

1.44 DOpus Magellan II ARexx: 'dopus set'.

```
dopus set <background|font|palette|pens|sound> <options>
```

This command allows you to change the various Directory Opus settings, it is used extensively by the Themes system.

See also:

```
dopus query
dopus query background
dopus query font
dopus query palette
dopus query pens
dopus query sound
dopus set background
dopus set font
dopus set palette
dopus set pens
dopus set sound
```

1.45 DOpus Magellan II ARexx: 'dopus set background'.

```
dopus set background <file> [desktop|lister|req] [centre|tile] [ ←
precision <precision>] [custom]
dopus set background on
```

This command allows you to set the background picture for either the Desktop, Listers, or Requesters.

The display will not be updated until a
refresh
is executed.

`file` specifies the path and filename of the background picture to use.

If you do not specify either `desktop`, `lister` or `req`, then `desktop` will be assumed.

`precision` can be one of `none`, `gui`, `icon`, `image`, or `exact`.

The `custom` keyword allows you to change the backgrounds without modifying the actual environment settings.

You must specify `custom on` both
dopus set background
and
dopus refresh background
for it to work.

`on` enables backgrounds for DOpus.

For example,

```
+ dopus set background on
+ dopus set background 'DOpus5:Backgrounds/StoneWash.iff' desktop center ↔
  precision exact
+ dopus set background 'DOpus5:Backgrounds/Marble.iff' desktop center ↔
  custom
```

See also:

```
dopus query
dopus query background
dopus refresh
dopus refresh background
dopus set
```

1.46 DOpus Magellan II ARexx: 'dopus set font'.

```
dopus set font <screen|listers|iconsd|iconsw> <name size>
```

This command allows you to set the font for Directory Opus' display.

```
screen    - specifies that this is the font to use when Opus is on it's
            own screen.
listers   - use this font for the listers' file display.
iconsd    - use this font for icons on the Desktop.
iconsw    - use this font for icons in windows.
```

name size is the name of the font, followed by it's size.

For example,

```
+ dopus set font screen courier.font 13
```

See also:

```
dopus query
dopus query font
dopus set
```

1.47 DOpus Magellan II ARexx: 'dopus set palette'.

```
dopus set palette <values>
```

This command allows you to configure the Opus palette. It accepts a string of up to 16 hex values; the last 8 represent the Opus colours, the first 8 represent the system colours when Opus is on it's own screen.

For example,

```
+ dopus set palette 0xffaabc 0x00aa55
```

See also:

```
dopus query
```

```
dopus query palette
```

```
dopus set
```

1.48 DOpus Magellan II ARexx: 'dopus set pens'.

```
dopus set pens <type>
```

This command allows you to configure the pen numbers used for various Opus display items.

Where `type` is a string signifying the pen set to alter, and is followed by a number of values which represent which of the pens to use.

`type` can be any one of the following:

```
icons    <dfg> <dbg> <dstyle> <wfg> <wbg> <wstyle>
files    <fg> <bg>
dirs     <fg> <bg>
selfiles <fg> <bg>
seldirs  <fg> <bg>
devices  <fg> <bg>
assigns  <fg> <bg>
source   <fg> <bg>
dest     <fg> <bg>
gauge    <normal> <full>
user     <count>
```

The pen number is mapped in the following way:

```
1-4      bottom four system colours
5-8      top four system colours
9-16     Opus user pens
```

For example,

```
+ dopus set pens icons 1 2 2 5 7 2
+ dopus set pens gauge 3 6
+ dopus set pens source 9 15
```

See also:

```
dopus query
dopus query pens
dopus set
```

1.49 DOpus Magellan II ARexx: 'dopus set sound'.

```
dopus set sound <event> <file> <volume>
```

This command allows you to set the sound for the specified event .

volume is the playback volume, allowable values are from 0 (minimum) to 64 (maximum).

For example,

```
+ dopus set sound Startup 'DOpus5:Sounds/Danger.8svx' 64
```

See also:

```
dopus query
dopus query sound
dopus set
```

1.50 DOpus Magellan II ARexx: 'dopus setappicon'.

```
dopus setappicon <handle> <item>
```

This allows you to do things to AppIcons added with the `dopus addappicon` command.

Valid item fields are:-

```
text <text> - Change the icon label.
busy (on|off) - Make icon busy or non-busy.
locked (on|off) - Make icon locked or unlocked.
```

If an icon is busy, it is unselectable by the user. It will not respond to double-clicks, pop-up menu events, drag'n'drops, etc. The icon image is ghosted when it is busy.

When an icon is locked, its position can not be changed and it can not be moved manually by the user, nor will a CleanUp will not affect it.

You can remove the icon's label by setting it to "" (the empty string).

See also:

```
dopus addappicon
dopus remappicon
AppIcon Handlers
```

1.51 DOpus Magellan II ARexx: 'dopus version'.

```
dopus version
```

The version command returns in

```
RESULT
a string in the format
```

```
<version> <revision>
```

and is useful in ARexx scripts for determining if certain features exist.

1.52 DOpus Magellan II ARexx: 'lister add'.

```
lister add <handle> <name> <size> <type> <seconds> <protect> < ←
comment>
```

This command adds an entry to the specified lister (handle).

```
name  is the full name of the entry (no path);
size  is the size of the entry (bytes);
type  is the type of the entry (less than 0 for a file, greater than 0
      for a directory);
seconds is the datestamp of the entry in seconds from 1/1/78;
protect is the protection bits of the file (in ASCII format, eg "rwed");
comment is the comment of the entry (if any).
```

Valid entry types are:-

```
0      device
1      plain directory
-1     plain file
2      directory in assign colour
-2     file in device colour
3      directory in bold (link)
-3     file in bold (link)
```

```

4      directory in assign colour and bold
-4     file in device colour and bold

```

After a `lister add` command, the display is not updated until you execute a

```

lister refresh
command.

```

For example,

```
+ lister add 121132636 "My file!" 12839 -1 540093905 prwed my comment
```

The ARexx calender/date functions provide routines perfect for converting various time formats to the datestamp entry and back again.

See also:

```

lister addstem

lister remove

```

1.53 DOpus Magellan II ARexx: 'lister addstem'.

```
lister addstem <handle> <stem>
```

This command adds files to a lister via a stem variable. It is more powerful than the

```

lister add
command and should be used in preference.

```

The fields of the stem variable are very similar to those returned by a

```

lister query <handle> entry stem
command (in fact, you could pass the

```

result the query directly to an `addstem` to add an identical entry to another lister.)

The fields that are used are :-

name	- name of entry (no path)
size	- file size (bytes)
type	- type of entry
protstring	- protection bits (ASCII string, eg "rwed")
protect	- protection value (number, used if protstring is not given)
comment	- file comment
datestring	- creation date and time (ASCII string)
date	- number of seconds since 1/1/78 (used if datestring is not given) Can be set to 0 for today's date.
filetype	- ascii string for file type display
selected	- 0 (not selected) or 1 (selected)
version	- version number
revision	- revision number

```

verdate      - version date string
userdata     - user data (value, not a string)
display      - custom display string
menu         - custom pop-up menu
base        - base ID for pop-up menu

```

Valid entry types are:-

```

0      device
1      plain directory
-1     plain file
2      directory in assign colour
-2     file in device colour
3      directory in bold (link)
-3     file in bold (link)
4      directory in assign colour and bold
-4     file in device colour and bold

```

Not all of these fields are required. As a bare minimum you should specify either the name or the display field.

```

For example (as a sequence of commands),
+ mynewentry.COMMENT = "This is my comment"
+ mynewentry.NAME = "anewfile.test"
+ mynewentry.TYPE = -1
+ lister addstem 121132636 mynewentry.

```

The `display` string allows you to specify a completely custom string to display for the entry. None of the other information will be displayed if this string is supplied. The maximum length is 256 characters.

The `userdata` field allows you to specify your own ID value (or any other value) to be associated with this entry. Its main usage is with the custom pop-up menu and

```

        custom handlers
.

```

The `menu` field allows you to specify a stem variable containing custom items for the pop-up menu that appears when the user presses the right button on this entry. Its format is the same as for AppIcon pop-up menus:

```

stem.COUNT   - number of entries
stem.BASE    - base ID
stem.0       - entry 1
stem.1       - entry 2
etc.

```

If `count` is set to 0, right-button pop-ups will be disabled for this file. If this field is not specified, the default pop-up menu will be displayed. If you specify "---" as an item, a separator bar will appear. When you receive a message that the user has selected one of these menu items, the message will contain the ID of the item. This is the value corresponding to the item's position in the stem array (eg 0 for item 1, 1 for item 2, etc). If the `base` field is specified, the value given for the base will be added to this ID.

```

For example (as a sequence of commands),
+ mymenu.0 = "Edit"

```

```

+ mymenu.1 = "---"
+ mymenu.2 = "View"
+ mymenu.3 = "Play"
+ mymenu.COUNT = 4
+ mynewentry.COMMENT = "This is my comment"
+ mynewentry.NAME = "anewfile.test"
+ mynewentry.TYPE = -1
+ mynewentry.MENU = mymenu.
+ lister addstem 121132636 mynewentry.

```

See the

Custom Handlers
section for more information on the messages

sent.

See also:

```

lister add

lister remove

```

1.54 DOpus Magellan II ARexx: 'lister clear'.

```
lister clear <handle>
```

This command clears the contents of the specified lister (handle). The display will not be updated until you execute a

```
lister refresh
```

command.

In previous versions of Opus 5, this command also cleared the

custom handler
name. This is no longer the case.

```
lister clear <handle> <item> <value>
```

This command clears a particular item of information in the specified lister.

handle is the handle of the lister in question;

item can be one of the following keywords:-
 abort

This clears the abort flag in the specified lister.
 For example, + lister clear 121132636 abort

```
flags <flags>
```

Clears sort/display flags for this lister. The display is not updated unless you execute a

```
lister refresh
```

command.

These flags are:-

```
reverse      - sort in reverse order
noicons      - filter icons
hidden      - filter hidden bit
```

For example, + lister clear 121132636 flags reverse
progress

This turns the progress indicator off in the specified
lister.

For example, + lister clear 121132636 progress

See also:

```
lister copy
lister empty
lister query abort
lister set newprogress
lister set progress
lister query flags
lister set flags
```

1.55 DOpus Magellan II ARexx: 'lister clear value'.

```
lister clear <handle> value <name>
```

This command lets you clear a name/value association that you previously made with

```
lister set value
.
```

handle is the handle of the lister to take the entries from while
name is required to clear the name/value pair.

For example,
+ lister clear 121132636 value MyName

See also:

```
lister query value
lister set value
```

1.56 DOpus Magellan II ARexx: 'lister copy'.

```
lister copy <handle> <destination>
```

This command copies the contents of one lister to another lister. Unlike most commands, the display of the destination lister is refreshed immediately.

`handle` is the handle of the lister to take the entries from while `destination` is the handle of the lister to copy the entries to.

For example,

```
+ lister copy 121132636 121963868
```

Note that what gets copied is only the entry data. No files are physically copied. The entries may not even represent files at all.

See also:

```
lister clear
```

```
lister empty
```

1.57 DOpus Magellan II ARexx: 'lister clearcaches'.

```
lister clearcaches <handle>
```

This command will flush any caches that were created by your lister, using your custom handler. No other caches will be affected.

`handle` is the handle of the lister in question.

See also:

```
lister findcache
```

1.58 DOpus Magellan II ARexx: 'lister close'.

```
lister close (all|<handle>)
```

This command closes the specified lister or all listers if the `all` keyword is supplied in place of a lister handle. Any function that is currently taking place will be aborted.

`handle` is the handle of the lister to close.

For example,

```
+ lister close 121132636
```

See also:

```
lister iconify
```

1.59 DOpus Magellan II ARexx: 'lister empty'.

```
lister empty <handle>
```

This command will display an empty cache for the specified lister handle (unlike

```
lister clear
which clears the contents of the current cache).
```

If no empty caches are available (and a new one can not be created), the existing cache will be cleared. If the lister has a

```
custom handler
attached, it will receive an inactive message.
```

Note that previous versions of Opus 5 did not send the 'inactive' message to the

```
custom handler
when this command was used.
```

See also:

```
lister copy
```

```
lister clear
```

1.60 DOpus Magellan II ARexx: 'lister findcache'.

```
lister findcache <handle> <path>
```

This command allows you to find a cached directory and display it in the lister. When it returns,

```
RESULT
is set to 0 if the path was not found,
```

or 1 if it was found. If the path is found, it will be automatically displayed in the lister and you don't need to do any more. If it is not found, you'll have to read the directory as normal.

handle is the handle of the lister in question.

path is the path of the cache to find.

For example,

```
+ lister findcache 137025148 Ram:testfile
> 1
```

See also:

```
lister clearcaches
```

1.61 DOpus Magellan II ARexx: 'lister getstring'.

```
lister getstring <handle> <text> [secure] [<length>] [<default>]
 [<buttons>]
```

This command allows you to prompt the user to input a text string. It is identical to the

```
dopus getstring
command
```

except that it takes a `lister handle` as an additional parameter and the requester will be centred over that `lister`.

`text` is a string of text to be displayed in the requester, and should be surrounded by quotes if it contains spaces.

The `secure` keyword causes the string to be displayed as asterisks ("`*`"), which can be useful for passwords.

`length` is the maximum length of string to accept and defaults to 80 if not specified.

`default` is the default value of the string; that is, the text you wish to initially appear in the field.

`buttons` are the buttons you wish the requester to have; each button should be separated by a vertical bar character ("`|`"). If the buttons parameter is omitted the requester will have a single button marked "OK".

For example,

```
+ lister getstring 121132636 "Please enter some text" 40 "" Okay|Cancel'
```

Note the quotes around the entire argument string.

You can have multiple lines in the requester text by putting the "return" character at the end of lines.

This would display a requester with the string "Please enter some text", a maximum input length of 40 characters, an empty default sting, and buttons labelled "Okay" and "Cancel".

The string (if any) is returned in

```
RESULT
. The cardinal number of the
```

selected button is returned in the special variable

```
DOPUSRC
, but the
```

last (rightmost) gadget is always number 0. In the above example, if the user clicked "Okay",

DOPUSRC
 would contain 1, and if the user clicked
 "Cancel" it would contain 0. If the RETURN key is used in the string gadget
 to accept the text, then the
 DOPUSRC
 variable will contain the value -1.

This command and the
 dopus getstring
 command are the only ones that use
 the
 DOPUSRC
 variable currently, but this may change in the future.

See also:

lister request

 dopus getstring

 dopus request

1.62 DOpus Magellan II ARexx: 'lister iconify'.

```
lister iconify (all|<handle>) [<state>]
```

This command causes either all listers or the specified lister to become iconified if state is 1, "on", or omitted altogether, and to deiconify if state is 0 or "off".

Specify all listers with the all parameter, or specify the handle of a lister with the handle parameter.

For example,
 + lister iconify 121132636
 + lister iconify all off

See also:

lister new iconify

 lister close

1.63 DOpus Magellan II ARexx: 'lister new'.

```
lister new [<x/y/w/h>] [toolbar <toolbar>] [inactive] [invisible]  

[iconify] [mode <name|icon|action|showall>] [fromicon <path>] [<path>]
```

This command creates a new lister.

You may optionally specify the position and size of the new lister. The default position of -1/-1 causes the lister to open under the mouse pointer with the default width and height as specified by the environment.

The position is specified by the [<x/y/w/h>] parameter:

```
x      - x-coordinate for top-left of lister window, or -1;
y      - y-coordinate for top-left of lister window, or -1;
w      - width of the lister window;
h      - height of the lister window.
```

A custom toolbar for the lister can be specified with the `toolbar` keyword; toolbar files are expected to be found in the `DOpus5:Buttons` directory if the full path is not given.

You may also specify a `path` to read when the lister opens.

Any `path` specified must occur at the end of the command line.

The initial state of a lister can be set by providing one or more of these additional keywords:

`inactive` prevents the newly opened lister from being the active window.
`invisible` causes the lister to be invisible initially.

See

```
    lister set visible
, and
    lister query visible
iconify causes the lister to start iconified.
```

See

```
    lister iconify
fromicon will open the new lister using the size and position ←
    information
from the specified directory's icon (only if a path is specified and an ←
    icon
exists). Eg, lister new fromicon sys:tools
```

`mode` lets you specify the initial mode of the new lister.
 ie. name, icon, icon action and showall

For example,

```
+ lister new
+ lister new 100/50/400/300
+ lister new ram:
+ lister new 80/30/200/200 dh0:work
+ lister new toolbar custom_toolbar work:
+ lister new fromicon SYS:Devs work:
+ lister new mode icon work:
> 121132636 (Typical return from any one of the above).
```

If the lister opens successfully, its handle is returned in the `RESULT`

```
variable. You must save the value of this handle if you wish to do ←
anything
```

further with this lister. In the above example, a handle of 121132636 was returned. This is used in many of the examples.

1.64 DOpus Magellan II ARexx: 'lister query'.

```
lister query <type>
```

This command returns the handles of all listers matching `type` in

```
RESULT
, unless an
error
occurs (for example there are no listers
```

of the given type).

`type` can be one of the following keywords:-

```
all
dest
source
```

```
You can also return the result in a variable of your choosing ↔
instead
```

of

```
RESULT
, or return the result in a stem variable. Select one of
the types above for more information.
```

```
lister query <handle> <item>
```

This command returns a particular item of information from the specified lister. `handle` is the handle of the lister in question. All information is returned in the

```
RESULT
variable, unless an
error
occurs.
```

`item` can be one of the following keywords:-

```
abort
active
busy
case
commentlength
dirs
```

entries
entry
files
firstsel
flags
handler
header
hide
label
lock
mode
namelength
numdirs
numentries
numfiles
numseldirs
numselentries
numselfiles
path
position
proc
seldirs
selentries
selfiles
separate
show
sort
title
toolbar

visible

window

Several of the keywords allow you to return the result in a ↔
variable of

your choosing instead of

RESULT

, or return the result in a stem

variable. Additionally, some of the keywords take further arguments.

Select one of the items above for more information.

See also:

lister set

1.65 DOpus Magellan II ARexx: 'lister query active'.

lister query active

This command returns the the handle of the current active lister, ie. the lister window that is active, the lister may or may not be in state source or destination.

The handle will be returned in

RESULT

, unless an

error

occurs (for example there are no listers).

For example,

```
+ lister query active
> 121132636
```

If no listers are available the result will be empty and

RC

will

contain the relevant

error code

.

1.66 DOpus Magellan II ARexx: 'lister query all'.

lister query all

This command returns the handles of all non-busy listers (that is, any

listers that are not performing a function at the time).

```
For example,
+ lister query all
> 121132636 121963868
```

If no matching listers are available the result will be empty and
 RC
 will
 contain the relevant
 error code
 .

You can use the Word() function in ARexx to traverse the list.

```
lister query all var <varname>
```

This is exactly the same as lister query all , except that the result
 is stored in the variable called varname instead of
 RESULT
 .

```
For example,
+ lister query all var all_handles
would return
all_handles = "121132636 121963868"
```

```
lister query all stem <stemname>
```

This is exactly the same as lister query all , except that the result
 is stored in a stem variable whose base-name is stemname .

```
For example,
+ lister query all stem all_handles
would return
all_handles.count = 2
all_handles.0 = 121132636
all_handles.1 = 121963868
```

Note that the stem count will correctly return 0 if there are no
 matching listers.

See also:

```
lister query dest
```

```
lister query source
```

1.67 DOpus Magellan II ARexx: 'lister query dest'.

```
lister query dest
```

This command returns the handles of all destination listers.

For example,

```
+ lister query dest
> 121963868
```

If no matching listers are available the result will be empty and

```
RC
will
contain the relevant
error code
.
```

You can use the Word() function in ARexx to traverse the list.

```
lister query dest var <varname>
```

This is exactly the same as lister query dest , except that the result is stored in the variable called varname instead of

```
RESULT
.
```

For example,

```
+ lister query dest var dest_handles
would return
dest_handles = 121963868
```

```
lister query dest stem <stemname>
```

This is exactly the same as lister query dest , except that the result is stored in a stem variable whose base-name is stemname .

For example,

```
+ lister query dest stem dest_handles
would return
dest_handles.count = 1
dest_handles.0 = 121963868
```

Note that the stem count will correctly return 0 if there are no matching listers.

See also:

```
lister query all
```

```
lister query source
```

1.68 DOpus Magellan II ARexx: 'lister query source'.

```
lister query source
```

This command returns the handles of all source listers.

```
For example,  
+ lister query source  
> 121132636 128765412
```

If no matching listers are available the result will be empty and

```
RC  
will  
contain the relevant  
error code  
.
```

You can use the Word() function in ARexx to traverse the list.

```
lister query source var <varname>
```

This is exactly the same as `lister query source` , except that the result is stored in the variable called `varname` instead of

```
RESULT  
.
```

```
For example,  
+ lister query source var source_handles  
would return  
source_handles = "121132636 128765412"
```

```
lister query source stem <stemname>
```

This is exactly the same as `lister query source` , except that the result is stored in a stem variable whose base-name is `stemname` .

```
For example,  
+ lister query source stem source_handles  
would return  
source_handles.count = 2  
source_handles.0 = 121132636  
source_handles.1 = 128765412
```

Note that the stem count will correctly return 0 if there are no matching listers.

See also:

```
lister query all
```

```
lister query dest
```

1.69 DOpus Magellan II ARexx: 'lister query abort'.

```
lister query <handle> abort
```

This returns a boolean (0 or 1) value indicating the status of the lister's abort flag. This query command is only valid if the lister has a progress indicator open (as this is the only way the user can abort a function anyway). This will return 1 if the user has clicked the abort gadget, 0 if she has not.

`handle` is the handle of the lister to query.

```
For example,  
+ lister query 121132636 abort  
> 0
```

Note that in Opus 4, querying the abort flag would also reset it. This is not the case in Opus 5; if you wish to reset the state of the abort flag you must use the

```
lister clear  
command.
```

See also:

```
lister clear abort  
  
lister set newprogress  
  
lister set progress
```

1.70 DOpus Magellan II ARexx: 'lister query busy'.

```
lister query <handle> busy
```

Returns a boolean value (0 or 1) indicating the lister busy status. That is, if the lister is currently busy, it will return 1, otherwise it will return 0.

`handle` is the handle of the lister to query.

```
For example,  
+ lister query 121132636 busy  
> 1
```

See also:

```
lister set busy  
  
lister wait
```

1.71 DOpus Magellan II ARexx: 'lister query case'.

```
lister query <handle> case
```

This command turns tells you whether the lister is case sensitive or not.

`handle` is the handle of the lister in question.

Returns 1 if the lister is case sensitive and 0 otherwise.

Since Amiga filenames are not case sensitive this setting defaults to off. It may be useful for some

custom handlers

to turn case sensitivity on,

however. OpusFTP is one such handler, and it needs case sensitivity when displaying UNIX directories because there could be entries called "RECENT", "recent", "Recent", and "ReCeNt" which are all different.

For example,

```
+ lister query 121132636 case
> 0
```

See also:

```
lister set case
```

1.72 DOpus Magellan II ARexx: 'lister query commentlength'.

```
lister query <handle> commentlength
```

Returns the maximum number of characters that can currently be displayed in the comment field of the lister.

`handle` is the handle of the lister in question.

For example,

```
+ lister query 121132636 commentlength
> 79
```

See also:

```
lister set commentlength
```

1.73 DOpus Magellan II ARexx: 'lister query dirs'.

```
lister query <handle> dirs [<separator>]
```

Returns the names of all directories in the lister (handle), surrounded by quotes. If you specify a separator character it will be placed between the quotes, otherwise space is used.

For example, (The quotes are included in the strings)

```
+ lister query 121132636 dirs
> "Clipboards" "ENV" "T"
+ lister query 121132636 dirs ,
> "Clipboards","ENV","T"
```

You should not rely on the ARexx Word() function to traverse the list because it does not support quotes and any directory with a space in its name will cause your script to malfunction. Use a stem variable instead (see below).

Note that the empty string will be returned if there are no directories in the lister. In previous versions of Opus 5, however, this was not the case.

```
lister query <handle> dirs var <varname>
```

This is exactly the same as lister query dirs , except that the result is stored in the variable called varname instead of

```
RESULT
```

```
.
```

For example,

```
+ lister query 121132636 dirs var dirs
would return
dirs = ' "Clipboards" "ENV" "T" '
```

```
lister query <handle> dirs stem <stemname>
```

This is exactly the same as lister query dirs , except that the result is stored in a stem variable whose base-name is stemname .

For example,

```
+ lister query 121132636 dirs stem dirs
would return
dirs.count      = 1
dirs.0          = "Clipboards"
dirs.1          = "ENV"
dirs.2          = "T"
```

Note that the stem count will correctly return 0 if there are no directories.

See also:

```
lister query entry
```

```
lister query entries
```

```
lister query files
```

```
lister query numdirs
lister query numseldirs
lister query seldirs
```

1.74 DOpus Magellan II ARexx: 'lister query display'.

```
lister query <handle> display
```

This returns a string indicating the current display items. The string will consist of the same keywords as for

```
lister query sort
, in the order
```

that they appear in the lister (if they appear at all).

Valid keywords are:-

name	- filename
size	- file size
protect	- protection bits
date	- datestamp
comment	- comment
filetype	- file type
version	- file version

For example,

```
+ lister query 121132636 display
> name size date protect comment
```

See also:

```
lister set display
lister query separate
lister set separate
lister query sort
lister set sort
lister query lock
lister set lock
```

1.75 DOpus Magellan II ARexx: 'lister query entries'.

```
lister query <handle> entries [<separator>]
```

Returns the names of all entries (that is, both files and directories) in the `lister (handle)`, surrounded by quotes. If you specify a separator character it will be placed between the quotes, otherwise space is used.

For example, (The quotes are included in the strings)

```
+ lister query 121132636 entries
> "Clipboards" "ENV" "T" "abc" "Disk.info"
+ lister query 121132636 entries ,
> "Clipboards","ENV","T","abc","Disk.info"
```

You should not rely on the ARexx `Word()` function to traverse the list because it does not support quotes and any entry with a space in its name will cause your script to malfunction. Use a stem variable instead (see below).

Note that the empty string will be returned if there are no entries in the `lister`. In previous versions of Opus 5, however, this was not the case.

```
lister query <handle> entries var <varname>
```

This is exactly the same as `lister query entries ,` except that the result is stored in the variable called `varname` instead of

```
RESULT
```

```
.
```

For example,

```
+ lister query 121132636 entries var ents
```

would return

```
ents = ' "Clipboards" "ENV" "T" "abc" "Disk.info"'
```

```
lister query <handle> entries stem <stemname>
```

This is exactly the same as `lister query entries ,` except that the result is stored in a stem variable whose base-name is `stemname`.

For example,

```
+ lister query 121132636 entries stem ents
```

would return

```
ents.count      = 5
ents.0          = Clipboards
ents.1          = ENV
ents.2          = T
ents.3          = abc
ents.4          = Disk.info
```

Note that the stem count will correctly return 0 if there are no entries.

See also:

```
lister query entry
```

```
lister query dirs
```

```

lister query files

lister query numentries

lister query numselentries

lister query selentries

```

1.76 DOpus Magellan II ARexx: 'lister query entry'.

```
lister query <handle> entry <name>
```

Returns information about the specified entry.

`name` is the actual name of the entry to return information about. You can supply `#xxx` for the name (where `xxx` is a number), to specify the cardinal number of the desired entry.

Note that you do not include the path in the name.

The information returned is in the format

```
<name> <size> <type> <selection> <seconds> <protect> <comment>
```

where `name` is the full name of the entry (no path),
`size` is the size of the entry (bytes),
`type` is the type of the entry (<0 means a file,
>0 means a directory),
`selection` indicates the selection status of the entry
(1 if the entry is selected, 0 if it is not selected),
`seconds` is the datestamp of the entry in seconds from 1/1/78,
`protect` is the protection bits of the file
(in ASCII format, eg "h---rwed"),
`comment` is the comment of the entry (if any).

The ARexx calender/date functions provide routines perfect for converting various time formats to the datestamp entry and back again.

For example,

```
+ lister query 121132636 entry ENV
> ENV -1 2 0 543401724 ----rwed
```

```
lister query <handle> entry <name> var <varname>
```

This is exactly the same as `lister query entry`, except that the result is stored in the variable called `varname` instead of `RESULT`.

For example,

```
+ lister query 121132636 entry "disk.info" var myvar
would return
myvar = "Disk.info 828 -1 0 590488349 ----rw-d"
```

```
lister query <handle> entry <name> stem <stemname>
```

This is exactly the same as `lister query entry`, except that the result is stored in a stem variable whose base-name is `stemname`.

The specified stem variable will have several fields, each containing information about the entry in question. These fields are as follows:-

```

name           - entry name (no path)
size           - file size (bytes)
type           - type (<0 = file, >0 = dir)
selected       - 0 (selected) or 1 (not selected)
date           - seconds since 1/1/78
protect        - protection bits (long value)
datestring     - datestamp in ASCII form
protstring     - protection bits in ASCII form, eg "----rwed"
comment        - file comment (if any)
filetype       - file type (if any)
version        - version number
revision       - revision number
verdate        - version date in numerical dd.mm.yy format
datenum        - file date in numerical dd.mm.yy format
time           - file time in hh:mm:ss 24 hour format
userdata       - userdata value (see
                lister addstem
                )
display        - display line (if any) (see
                lister addstem
                )

```

For example,

```
> lister query 121132636 entry "Disk.info" stem fileinfo.
```

would return,

```

fileinfo.name      = "Disk.info"
fileinfo.size      = 828
fileinfo.type      = -1
fileinfo.selected  = 0
fileinfo.date      = 590488349
fileinfo.protect   = 2
fileinfo.datestring = "17-Sep-96 08:32:29"
fileinfo.protstring = "----rw-d"
fileinfo.comment   = ""      (There comment is empty)
fileinfo.filetype  = ""      (Filetypes not shown in the lister)
fileinfo.version   = ""      (Versions not shown in the lister,
                              and the file has no version string
                              inside it anyway).
fileinfo.verdate   = "17-09-96"
fileinfo.datenum   = "08:32:29"
fileinfo.time      = 0       (This is the default userdata value)
fileinfo.userdata  = 0
fileinfo.display   = ""

```

See also:

```
lister add
```

```
lister addstem
lister query dirs
lister query entries
lister query files
lister query firstsel
lister query numdirs
lister query numentries
lister query numfiles
lister query numsel dirs
lister query numsel entries
lister query numsel files
lister query sel dirs
lister query sel entries
lister query sel files
lister select
```

1.77 DOpus Magellan II ARexx: 'lister query files'.

```
lister query <handle> files [<separator>]
```

Returns the names of all files in the `lister (handle)`, surrounded by quotes. If you specify a `separator` character it will be placed between the quotes, otherwise space is used.

For example, (The quotes are included in the strings)

```
+ lister query 121132636 files
> "abc" "Disk.info"
+ lister query 121132636 files ,
> "abc", "Disk.info"
```

You should not rely on the ARexx `Word()` function to traverse the list because it does not support quotes and any file with a space in its name will cause your script to malfunction. Use a stem variable instead (see below).

Note that the empty string will be returned if there are no files in the `lister`. In previous versions of Opus 5, however, this was not the case.

```
lister query <handle> files var <varname>
```

This is exactly the same as `lister query files` , except that the result is stored in the variable called `varname` instead of

```
RESULT
```

.

For example,

```
+ lister query 121132636 files var fillies
```

would return

```
fillies          = '"abc" "Disk.info"'
```

```
lister query <handle> files stem <stemname>
```

This is exactly the same as `lister query files` , except that the result is stored in a stem variable whose base-name is `stemname` .

For example,

```
+ lister query 121132636 files stem fyels
```

would return

```
fyels.count      = 2
fyels.0          = abc
fyels.1          = Disk.info
```

Note that the stem count will correctly return 0 if there are no entries.

See also:

```
lister query entry
```

```
lister query dirs
```

```
lister query entries
```

```
lister query firstsel
```

```
lister query numfiles
```

```
lister query numselfiles
```

```
lister query selfiles
```

1.78 DOpus Magellan II ARexx: 'lister query firstsel'.

```
lister query <handle> firstsel
```

Returns the name (without path) of the first selected entry in the lister.

`handle` is the handle of the lister in question.

The entry is not deselected, so if you don't deselect it yourself this

command will only ever return the one name.

For example,

```
+ lister query 121132636 firstsel  
> "ENV"
```

See also:

```
lister query entry  
lister query dirs  
lister query entries  
lister query files  
lister query numseldirs  
lister query numselentries  
lister query numselfiles  
lister query seldirs  
lister query selentries  
lister query selfiles  
lister select
```

1.79 DOpus Magellan II ARexx: 'lister query flags'.

```
lister query <handle> flags
```

This returns a string indicating any sort or display flags that are active for the lister.

These flags are:-

```
reverse      - sort in reverse order  
noicons     - filter icons  
hidden      - filter hidden bit
```

handle is the handle of the lister in question.

For example,

```
+ lister query 121132636 flags  
> noicons
```

See also:

```
lister set flags
```

```
lister clear flags
lister query show
lister set show
lister query hide
lister set hide
lister query display
lister set display
lister query separate
lister set separate
lister query sort
lister set sort
lister query lock
lister set lock
```

1.80 DOpus Magellan II ARexx: 'lister query handler'.

```
lister query <handle> handler
```

Returns the name of the current custom handler port for a lister

handle is the handle of the lister in question.

For example,

```
+ lister query 121132636 handler
> ArcDir121132636
```

See also:

```
lister set handler
```

Custom Handlers

1.81 DOpus Magellan II ARexx: 'lister query header'.

```
lister query <handle> header
```

This command returns the current header string of a lister.

handle is the handle of the lister in question.

The header will be returned in
RESULT
, unless an
error
occurs (for example there are no listers matching the handle given ↵
).

If the header is the default, then
RESULT
will be empty.

For example,
+ lister query 121132636 header
> Delete files?

If no matching listers are available the result will be empty and
RC
will
contain the relevant
error code
.

See also:

lister query label
lister query title
lister set header
lister set label
lister set title

1.82 DOpus Magellan II ARexx: 'lister query hide'.

```
lister query <handle> hide
```

This returns the current hide filter for a lister.
Usually this will be empty to indicate that nothing is being hidden.

handle is the handle of the lister in question.

For example,
+ lister query 121132636 hide
> #?.o

See also:

```
lister set hide
lister query show
lister set show
lister query flags
lister set flags
lister clear flags
lister query display
lister set display
lister query separate
lister set separate
lister query sort
lister set sort
lister query lock
lister set lock
```

1.83 DOpus Magellan II ARexx: 'lister query label'.

```
lister query <handle> label
```

This command returns the label that appears beneath this lister when it is iconified. By default, the label will be the name of the current directory. This label can, however, be changed by calling the

```
lister set label
```

command.

handle is the handle of the lister in question.

For example,

```
+ lister query 121132636 label
> Ram Disk
```

```
lister set label
```

1.84 DOpus Magellan II ARexx: 'lister query lock'.

```
lister query <handle> lock [state|format]
```

This command returns the current lock status of either the lister's state or format.

`handle` is the handle of the lister in question.

Returns 0 to indicate "unlocked", and 1 to indicate "locked".

For example,

```
+ lister query 121132636 lock state
> 0
```

A lister state and/or format can be locked and unlocked by the

```
lister set lock
```

command. While a lister's state (source/dest/off/etc.) is locked the user cannot change it. Similarly, while a lister's format is locked the user cannot change it (currently this just disables the Format Edit Requester and it is still possible to change the format with the field gadgets, if enabled -- hopefully this will be fixed in the future).

Note that "locked" in this sense is nothing to do with the "locked source" and "locked destination" states which you can put listers in to have multiple source or destination listers, nor is it anything to do with when a lister is "locked" in position.

See also:

```
lister set lock
```

```
lister query display
```

```
lister set display
```

```
lister set separate
```

```
lister query sort
```

```
lister set sort
```

```
lister query separate
```

```
lister query show
```

```
lister set show
```

```
lister query hide
```

```
lister set hide
```

```
lister query flags
```

```
lister set flags
```

```
lister clear flags
```

1.85 DOpus Magellan II ARexx: 'lister query mode'.

```
lister query <handle> mode
```

This returns the current mode of the lister and also the word showall if the lister is in an icon mode and displaying files without icons.

handle is the handle of the lister in question.

The lister modes are:-

```
name          - name mode
icon          - workbench style icon mode
icon action   - icon action mode
```

For example,

```
+ lister query 121132636 mode
> icon action showall
```

See also:

```
lister set mode
```

```
lister set source
```

```
lister set dest
```

```
lister set off
```

```
lister query lock
```

```
lister set lock
```

1.86 DOpus Magellan II ARexx: 'lister query namelength'.

```
lister query <handle> namelength
```

This command gets the maximum length allowed for filenames in the lister.

handle is the handle of the lister in question.

The absolute minimum length is 30 characters which is also the default length (and the current filename length limit of AmigaDOS).

Changing the length (with

```
lister set namelength
) will only be useful to
```

writers of
 custom handlers
 . Note that the internal Opus commands, for
the most part, do not currently support filenames longer than 30
characters.

For example,
 + lister query 121132636 namelength
 > 30

See also:

```
lister set namelength
```

1.87 DOpus Magellan II ARexx: 'lister query numdirs'.

```
lister query <handle> numdirs
```

Returns the number of directories in the lister.

handle is the handle of the lister in question.

For example,
 + lister query 121132636 numdirs
 > 3

See also:

```
lister query dirs  
lister query numentries  
lister query numfiles  
lister query numseldirs  
lister query numselentries  
lister query numselfiles  
lister query seldirs
```

1.88 DOpus Magellan II ARexx: 'lister query numentries'.

```
lister query <handle> numentries
```

Returns the total number of entries in the lister (files + dirs).

handle is the handle of the lister in question.

For example,

```
+ lister query 121132636 numentries  
> 7
```

See also:

lister query entry

lister query entries

lister query numdirs

lister query numfiles

lister query numselentries

lister query selentries

1.89 DOpus Magellan II ARexx: 'lister query numfiles'.

```
lister query <handle> numfiles
```

Returns the number of files in the lister.

handle is the handle of the lister in question.

For example,

```
+ lister query 121132636 numfiles  
> 4
```

See also:

lister query files

lister query numdirs

lister query numentries

lister query numselfiles

lister query selfiles

1.90 DOpus Magellan II ARexx: 'lister query numseldirs'.

```
lister query <handle> numseldirs
```

Returns the number of selected directories in the lister.

`handle` is the handle of the lister in question.

For example,

```
+ lister query 121132636 numsel dirs
> 1
```

See also:

```
lister query dirs
lister query firstsel
lister query numdirs
lister query numselentries
lister query numselfiles
lister query seldirs
lister select
```

1.91 DOpus Magellan II ARexx: 'lister query numselentries'.

```
lister query <handle> numselentries
```

Returns the number of selected entries (files + dirs) in the lister.

`handle` is the handle of the lister in question.

For example,

```
+ lister query 121132636 numselentries
> 2
```

See also:

```
lister query entry
lister query entries
lister query firstsel
lister query numentries
lister query numsel dirs
lister query numselfiles
lister query selentries
```

```
lister select
```

1.92 DOpus Magellan II ARexx: 'lister query numselfiles'.

```
lister query <handle> numselfiles
```

Returns the number of selected files in the lister.

`handle` is the handle of the lister in question.

For example,

```
+ lister query 121132636 numselfiles  
> 1
```

See also:

```
lister query files
```

```
lister query firstsel
```

```
lister query numfiles
```

```
lister query numseldirs
```

```
lister query numselentries
```

```
lister query selfiles
```

```
lister select
```

1.93 DOpus Magellan II ARexx: 'lister query path'.

```
lister query <handle> path
```

Returns a string indicating the current path visible in the lister.

`handle` is the handle of the lister in question.

For example,

```
+ lister query 121132636 path  
> ram:
```

See also:

```
lister set path
```

```
lister read
```

1.94 DOpus Magellan II ARexx: 'lister query position'.

```
lister query <handle> position
```

Returns the current position and size of the lister. The word locked will also be returned if the lister is locked in position.

handle is the handle of the lister in question.

The position returned is in the following format:

```
<x/y/w/h> [locked] <L/C/c/l>
```

```
x      - x-coordinate for top-left of lister window;
y      - y-coordinate for top-left of lister window;
w      - width of the lister window;
h      - height of the lister window;
L      - total number of lines in the lister;
C      - total number of columns in the lister;
c      - number of visible columns in the lister;
l      - number of visible lines in the lister.
```

Note that the lines/columns data was not returned before Opus 5.5.

For example,

```
+ lister query 121132636 position
> 360/47/360/151 27/82/42/14

+ lister query 121132636 position
> 360/47/360/151 locked 27/82/42/14
```

See also:

```
lister set position
```

1.95 DOpus Magellan II ARexx: 'lister query proc'.

```
lister query <handle> proc
```

This command returns the process address of the lister.

handle is the handle of the lister in question.

The process address will be returned in

```
RESULT
, unless an
```

error occurs, then

```
RC
will contain the relevant
error code
```

.

For example,
 + lister query 121132636 proc
 > 122132648

1.96 DOpus Magellan II ARexx: 'lister query seldirs'.

```
lister query <handle> seldirs [<separator>]
```

Returns the names of all selected directories in the lister (handle), surrounded by quotes. If you specify a separator character it will be placed between the quotes, otherwise space is used.

For example, (The quotes are included in the strings)
 + lister query 121132636 seldirs
 > "ENV" "T"
 + lister query 121132636 seldirs ,
 > "ENV", "T"

You should not rely on the ARexx Word() function to traverse the list because it does not support quotes and any directory with a space in its name will cause your script to malfunction. Use a stem variable instead (see below).

Note that the empty string will be returned if there are no selected directories in the lister. In previous versions of Opus 5, however, this was not the case.

```
lister query <handle> seldirs var <varname>
```

This is exactly the same as lister query seldirs , except that the result is stored in the variable called varname instead of
 RESULT
 .

For example,
 + lister query 121132636 seldirs var sds
 would return
 sds = ' "ENV" "T" '

```
lister query <handle> seldirs stem <stemname>
```

This is exactly the same as lister query seldirs , except that the result is stored in a stem variable whose base-name is stemname .

For example,
 + lister query 121132636 seldirs stem sdirs
 would return
 sdirs.count = 2
 sdirs.0 = ENV
 sdirs.1 = T

Note that the stem count will correctly return 0 if there are no

selected directories.

See also:

```

lister query dirs

lister query firstsel

lister query numdirs

lister query numseldirs

lister query selentries

lister query selfiles

lister select

```

1.97 DOpus Magellan II ARexx: 'lister query selentries'.

```
lister query <handle> selentries [<separator>]
```

Returns the names of all selected entires (files + dirs) in the lister (handle), surrounded by quotes. If you specify a separator character it will be placed between the quotes, otherwise space is used.

For example, (The quotes are included in the strings)

```

+ lister query 121132636 selentries
> "ENV" "T" "abc" "Disk.info"
+ lister query 121132636 selentries ,
> "ENV","T","abc","Disk.info"

```

You should not rely on the ARexx Word() function to traverse the list because it does not support quotes and any entry with a space in its name will cause your script to malfunction. Use a stem variable instead (see below).

Note that the empty string will be returned if there are no selected entries in the lister. In previous versions of Opus 5, however, this was not the case.

```
lister query <handle> selentries var <varname>
```

This is exactly the same as lister query selentries , except that the result is stored in the variable called varname instead of

```

RESULT
.

```

For example,

```
+ lister query 121132636 selentries var sentryz
```

would return

```
stentryz          = ' "ENV" "T" "abc" "Disk.info"'
```

```
lister query <handle> selentries stem <stemname>
```

This is exactly the same as `lister query selentries` , except that the result is stored in a stem variable whose base-name is `stemname` .

For example,

```
+ lister query 121132636 selentries stem ststentryz
```

would return

```
ststentryz.count      = 4
ststentryz.0          = ENV
ststentryz.1          = T
ststentryz.2          = abc
ststentryz.3          = Disk.info
```

Note that the stem count will correctly return 0 if there are no selected entries.

See also:

```
lister query entry
lister query entries
lister query firstsel
lister query numentries
lister query numselentries
lister query seldirs
lister query selfiles
lister select
```

1.98 DOpus Magellan II ARexx: 'lister query selfiles'.

```
lister query <handle> selfiles [<separator>]
```

Returns the names of all selected files in the `lister (handle)`, surrounded by quotes. If you specify a `separator` character it will be placed between the quotes, otherwise space is used.

For example, (The quotes are included in the strings)

```
+ lister query 121132636 selfiles
> "abc" "Disk.info"
+ lister query 121132636 selfiles ,
> "abc", "Disk.info"
```

You should not rely on the ARexx Word() function to traverse the list because it does not support quotes and any file with a space in its name will cause your script to malfunction. Use a stem variable instead (see below).

Note that the empty string will be returned if there are no selected files in the lister. In previous versions of Opus 5, however, this was not the case.

```
lister query <handle> selfiles var <varname>
```

This is exactly the same as `lister query selfiles`, except that the result is stored in the variable called `varname` instead of

```
RESULT
```

```
.
```

For example,

```
+ lister query 121132636 selfiles var sellyfilles
```

would return

```
sellyfilles = '"abc" "Disk.info"'
```

```
lister query <handle> selfiles stem <stemname>
```

This is exactly the same as `lister query selfiles`, except that the result is stored in a stem variable whose base-name is `stemname`.

For example,

```
+ lister query 121132636 selfiles stem sellyfilles
```

would return

```
sellyfilles.count = 2
sellyfilles.0 = abc
sellyfilles.1 = Disk.info
```

Note that the stem count will correctly return 0 if there are no selected files.

See also:

```
lister query files
```

```
lister query firstsel
```

```
lister query numfiles
```

```
lister query numselfiles
```

```
lister query seldirs
```

```
lister query selentries
```

```
lister select
```

1.99 DOpus Magellan II ARexx: 'lister query separate'.

```
lister query <handle> separate
```

This returns a keyword indicating the current file separation method in the lister.

handle is the handle of the lister in question.

Valid methods are:-

```
mix          - mix files and directories
dirsfirst    - directories first
filesfirst   - files first
```

For example,

```
+ lister query 121132636 separate
> dirsfirst
```

See also:

```
lister set separate
```

```
lister query sort
```

```
lister set sort
```

```
lister query lock
```

```
lister set lock
```

```
lister query flags
```

```
lister set flags
```

```
lister clear flags
```

```
lister query display
```

```
lister set display
```

1.100 DOpus Magellan II ARexx: 'lister query show'.

```
lister query <handle> show
```

This returns the current show filter for the lister.

Usually this will be empty, indicating that everything not matched by the hide filter is shown.

handle is the handle of the lister in question.

For example,

```
+ lister query 121132635 show
```

> mod.#?

See also:

```
lister set show
lister query hide
lister set hide
lister query flags
lister set flags
lister clear flags
lister query display
lister set display
lister query separate
lister set separate
lister query sort
lister set sort
lister query lock
lister set lock
```

1.101 DOpus Magellan II ARexx: 'lister query sort'.

```
lister query <handle> sort
```

This returns a keyword indicating the current sort method in the lister.

handle is the handle of the lister in question.

Valid sort methods are:-

```
name          - filename
size          - file size
protect       - protection bits
date          - datestamp
comment       - comment
filetype      - file type
version       - file version
```

For example,

```
+ lister query 121132636 sort
> name
```

Note that reverse sorting may or may not be in effect in the lister. Use

```
lister query flags
to check.
```

See also:

```
lister set sort
lister query flags
lister set flags
lister clear flags
lister query display
lister set display
lister query separate
lister set separate
lister query lock
lister set lock
```

1.102 DOpus Magellan II ARexx: 'lister query title'.

```
lister query <handle> title
```

This command returns the current title string of a lister.

handle is the handle of the lister in question.

The title will be returned in

```
RESULT
, unless an
error
occurs (for example there are no listers matching the handle given ←
).
```

If the header is the default, then

```
RESULT
will be empty.
```

For example,

```
+ lister query 121132636 title
> Current Directory RAM:
```

If no matching listers are available the result will be empty and

```
RC
will
```

contain the relevant
error code
.

See also:

```
lister query label  
lister query header  
lister set header  
lister set label  
lister set title
```

1.103 DOpus Magellan II ARexx: 'lister query toolbar'.

```
lister query <handle> toolbar
```

This returns the toolbar currently being used by the lister.

handle is the handle of the lister in question.

For example,

```
+ lister query 121132636 toolbar  
> DOpus5:Buttons/toolbar
```

See also:

```
lister set toolbar
```

1.104 DOpus Magellan II ARexx: 'lister query value'.

```
lister set value <handle> <name>
```

This command lets you query a value you have previously set with
lister set value
.

handle is the handle of the lister in question.

name is the name for which a value is queried.

For example,

```
+ lister query 121132636 value MyName  
> Jon Citizen
```

See also:

```
lister query value
```

```
lister set value
```

1.105 DOpus Magellan II ARexx: 'lister query visible'.

```
lister query <handle> visible
```

Returns a boolean (0 = no, 1 = yes) value indicating if the lister is currently visible.

`handle` is the handle of the lister in question.

A lister's visibility can be changed by the
`lister set visible`
command.

When a lister is invisible its window is completely hidden. A lister is "visible" when its window is open on a screen, even if it is behind another window so that you cannot see it.

For example,
+ lister query 121132636 visible
> 1

See also:

```
lister set visible
```

```
lister new invisible
```

1.106 DOpus Magellan II ARexx: 'lister query window'.

```
lister query <handle> window
```

Returns a pointer to the window structure of the lister, or 0 if the lister has no window.

`handle` is the handle of the lister in question.

For example,
+ lister query 121132636 window
> 0

1.107 DOpus Magellan II ARexx: 'lister read'.

```
lister read <handle> <path> [force]
```

This command will read the given path into the specified lister (handle).

By default a new cache is used to read the directory; if the force keyword is specified, the current cache will be cleared and the directory will be read into that.

The old path is returned in
RESULT

.

For example,

```
+ lister read 121132636 'dh0:test'  
> RamDisk:
```

See also:

```
lister query path
```

```
lister set path
```

1.108 DOpus Magellan II ARexx: 'lister refresh'.

```
lister refresh (all|<handle>) [full] [date]
```

This command refreshes the display of the specified lister (handle) or all listers if the keyword all is given in place of a lister handle. Unlike Opus 4, none of the lister modifying commands will actually refresh or update the lister display; hence, you must use this command after making any changes (changing sort method, adding files, etc) to have the changes display.

The optional full keyword causes the lister title and status displayed to be refreshed as well.

For example,

```
+ lister refresh 121132636 full
```

If the date keyword is specified, the lister will update its directory timestamp, which will stop it re-reading itself the next time it is activated. If this keyword is specified, the lister display itself is not refreshed.

For example,

```
+ lister refresh 12113236 date
```

1.109 DOpus Magellan II ARexx: 'lister reload'.

```
lister reload <handle> <name> [update]
```

This command reloads (or loads for the first time) a file in a lister.

`name` is either the name of the entry, or #xxx (where xxx is a number) to specify the cardinal number of the entry.

The `update` keyword causes the lister datestamp to be updated (to save unnecessary reloading).

For example,

```
+ lister reload 12381928 'filename.lha' update
```

If the file previously existed in the lister, the user data, user menus and version information of the file is preserved.

1.110 DOpus Magellan II ARexx: 'lister remove'.

```
lister remove <handle> <name>
```

This command removes an entry from the specified lister.

`name` is either the name of the entry, or #xxx (where xxx is a number) to specify the cardinal number of the entry.

The display is not updated until you execute a
lister refresh
command.

For example,

```
+ lister remove 121132636 #5
```

See also:

```
lister add
```

```
lister addstem
```

1.111 DOpus Magellan II ARexx: 'lister request'.

```
lister request <handle> <text> <buttons>
```

This command is identical to the

```
dopus request  
command except that it
```

takes an additional `handle` parameter and the requester will be centred over that lister.

`text` is a string of text to be displayed in the requester.

`buttons` are the buttons you wish the requester to have; each button should be separated by a vertical bar character.

For example,

```
+ lister request 121132636 ' "Choose one" Option 1|Option 2|Option 3'
```

Note the quotes around the entire argument string.

You can have multiple lines in the requester text by putting the "return" character at the end of lines.

This would display a requester with the string "Choose one", and three buttons labelled Option 1 to Option 3.

The cardinal number of the selected button is returned in

`RC`

. The last

button supplied (Option 3 in this case) is designated a "Cancel" button, and so returns the value 0. Therefore, the values returned by this example are 1, 2, and 0, respectively.

See also:

`lister getstring`

`dopus request`

`dopus getstring`

1.112 DOpus Magellan II ARexx: 'lister set'.

```
lister set <handle> <item> [<arguments>]
```

This command sets a particular item of information for the specified lister.

`handle` is the handle of the lister in question.

`item` can be one of the following keywords:-

`busy`

`case`

`commentlength`

`dest`

`display`

`field`

flags
header
handler
hide
label
lock
mode
namelength
newprogress
off
path
position
progress
separate
show
sort
source
title
toolbar
visible
arguments , if any, are specific to each command.

See also:

lister query

1.113 DOpus Magellan II ARexx: 'lister set busy'.

```
lister set <handle> busy (0|1|off|on) [wait]
```

Sets the busy status for the lister.

`handle` is the handle of the lister in question.

Specify 1 or "on" to make the lister go busy, or specify 0 or "off" to un-busy the lister.

When turning busy status on you can also provide the `wait` keyword which will cause the command to be synchronous (it won't return until the lister has finished going busy). You should use this when you want to do something else to the lister right after making it go busy, otherwise it could ignore you.

For example,

```
+ lister set 121132636 busy on wait
+ lister set 121132636 busy 0
```

See also:

```
lister query busy
```

```
lister wait
```

1.114 DOpus Magellan II ARexx: 'lister set case'.

```
lister set <handle> case (0|1|off|on)
```

This command turns on (1 or "on") or off (0 or "off") case sensitivity for the lister.

`handle` is the handle of the lister in question.

Since Amiga filenames are not case sensitive this setting defaults to off. It may be useful for some

```
custom handlers
to turn case sensitivity on,
```

however. OpusFTP is one such handler, and it needs case sensitivity when displaying UNIX directories because there could be entries called "RECENT", "recent", "Recent", and "ReCeNt" which are all different.

For example,

```
+ lister set 121132636 case on
+ lister set 121132636 case off
```

See also:

```
lister query case
```

1.115 DOpus Magellan II ARexx: 'lister set commentlength'.

```
lister set <handle> commentlength <length>
```

Allows you to set the maximum comment length allowed in the lister.

NOTE: This only affects the lister display, the filesystem still restricts the maximum file comment that can be saved to disk as 79 characters.

handle is the handle of the lister in question.

length is the amount of characters to display in the comment field.

For example,

```
+ lister set 121132636 commentlength 100
```

See also:

```
lister query commentlength
```

1.116 DOpus Magellan II ARexx: 'lister set dest'.

```
lister set <handle> dest [lock]
```

Makes the lister the destination.

handle is the handle of the lister in question.

If you specify the lock keyword, it will be locked as a destination.

For example,

```
+ lister set 121132636 dest
```

See also:

```
lister set source
```

```
lister set off
```

```
lister query lock
```

```
lister set lock
```

```
lister query mode
```

```
lister set mode
```

1.117 DOpus Magellan II ARexx: 'lister set display'.

```
lister set <handle> display <items>
```

Sets the display items for this lister.

`handle` is the handle of the lister in question.

The display will not be updated until you execute a
lister refresh
command.

The items are the same as those returned by the
lister query display
and
lister query sort
commands.

Valid items are:-

name	- filename
size	- file size
protect	- protection bits
date	- datestamp
comment	- comment
filetype	- file type
version	- file version

For example,

```
+ lister set 121132636 display name date size protect
```

See also:

```
lister query display  
lister query separate  
lister set separate  
lister query sort  
lister set sort  
lister query lock  
lister set lock  
lister query show  
lister set show  
lister query hide  
lister set hide  
lister query flags  
lister set flags
```

```
lister clear flags
```

1.118 DOpus Magellan II ARexx: 'lister set field'.

```
lister set <handle> field [<number> <string>]
```

This allows you to set your own strings to be used in the lister field titles. You can not change the nature of the columns in the lister - this just allows you to change the heading.

The `number` (0-9) specifies which string to replace from the table:-

<code>name</code>	<code>0,</code>
<code>size</code>	<code>1,</code>
<code>access</code>	<code>2,</code>
<code>date</code>	<code>3,</code>
<code>comment</code>	<code>4,</code>
<code>type</code>	<code>5,</code>
<code>owner</code>	<code>6,</code>
<code>group</code>	<code>7,</code>
<code>net</code>	<code>8,</code>
<code>version</code>	<code>9.</code>

Set to an empty string to restore the default.

You will need to do a

```
lister refresh <handle> full
```


to update the display
once you have changed the titles.

`handle` is the handle of the lister in question.

For example,

```
+ lister set 121132636 field 0 "FileName" 4 "Notes"
```

```
lister set <handle> field (on|off)
```

This allows you to remove field titles altogether ("off") and replace them when you are finished ("on"). (You cannot use 1 and 0 instead of "on" and "off", for obvious reasons.)

`handle` is the handle of the lister in question.

Note that if field titles have not been enabled in the configuration, an ARexx script is unable to turn them on.

See also:

```
lister query label
```

```
lister set label  
  
lister set title  
  
lister set header
```

1.119 DOpus Magellan II ARexx: 'lister set flags'.

```
lister set <handle> flags <flags>
```

Sets sort/display flags for the lister.

`handle` is the handle of the lister in question. You may use `+flag` to turn a flag on, `-flag` to turn a flag off, or `/flag` to toggle a flag. By default, flags will be turned on.

The display is not updated unless you execute a
lister refresh
command.

Valid flags are:-

```
reverse          - sort in reverse order  
noicons          - filter icons  
hidden - filter hidden bit
```

For example,

```
+ lister set 121132636 flags +reverse -noicons
```

This will turn the reverse flag on and the noicons flag off.

See also:

```
lister query flags  
  
lister clear flags  
  
lister query show  
  
lister set show  
  
lister query hide  
  
lister set hide  
  
lister query display  
  
lister set display  
  
lister query separate  
  
lister set separate  
  
lister query sort
```

```
lister set sort  
  
lister query lock  
  
lister set lock
```

1.120 DOpus Magellan II ARexx: 'lister set header'.

```
lister set <handle> header <string>
```

This works just like

```
lister set title  
except it changes the text in the
```

'Files x/y Dirs x/y' bar.

handle is the handle of the lister in question.

Specify the new header text for the string parameter.

The old header string is returned in

```
RESULT
```

.

Set this to an empty string to restore the default.

If you wish to actually display an empty header, set it to a "-" (hyphen character).

You will need to do a

```
lister refresh <handle> full  
to update the display
```

once you have changed the header.

For example,

```
+ lister set 121132636 header Super-Custom Header  
+ lister set 121132636 header (To reset it)
```

See also:

```
lister query label  
  
lister set label  
  
lister set field  
  
lister set title
```

1.121 DOpus Magellan II ARexx: 'lister set handler'.

```

    lister set <handle> handler <port name> [quotes] [fullpath] [ ↔
        editing]
    [nopopups] [guage] [leaveguage] [synctraps] [subdrop]

```

Sets the

```

    custom handler
    port name for this lister.

```

`handle` is the handle of the lister in question.

`port name` is the name of the message port to which messages from Opus will be sent.

Note that message port names are case sensitive. Make sure that you use the correct case and that you quote the name properly so that ARexx doesn't uppercase it.

If you specify the `quotes` flag, any filenames sent in messages to the port will be enclosed in quotes (this is a good idea as it allows you to support filenames containing spaces).

If you specify the `fullpath` flag, messages will always contain the full path name of a file, irrespective of whether it came from an Opus lister or not. (Usually, if the file comes from a lister you will only get the filename itself, plus the lister handle with which to find out the path).

If you specify the `editing` flag, inline lister editing will be enabled for this lister.

If you specify the `nopopups` flag, all file popups will be disabled for this lister.

If you specify the `guage` flag, you can enable the 'Free Space Gauge' to show the proportion of space free on the disk. (note that the user may still have turned them off in the environment)

Specifying the `leaveguage` flag, will cause an existing fuelguage to remain there, or leave it absent if it wasn't. This is in contrast to the above `guage` flag, which will always add a fuelguage.

Not providing the `guage` or the `leaveguage` will always cause a lister to have no fuelguage.

The `synctraps` flag causes messages trapped by an ARexx lister handler to be handled synchronously. By default, (and for compatibility), all handler messages remain asynchronous.

The `subdrop` flag allows handlers to use the Drag into subdirectories feature of Opus. See Environment/Lister Options in the Opus manual for more information on this option.

For example,

```

+ lister set 121132636 handler 'lhadir_handler' quotes
+ lister set 121132636 handler 'lhadir_handler' quotes guage nopopups
+ lister set 121132636 handler 'lhadir_handler' quotes leaveguage

```

See

Custom Handlers
section for more details.

See also:

lister query handler

Custom Handlers

1.122 DOpus Magellan II ARexx: 'lister set hide'.

```
lister set <handle> hide <pattern>
```

Sets the hide pattern for the lister to the string given for the pattern parameter.

handle is the handle of the lister in question.

The pattern is applied immediately but the display is not updated until you execute a

```
lister refresh  
command.
```

For example,

```
+ lister set 121132636 hide '#?.info'
```

See also:

lister query hide

lister query show

lister set show

lister query flags

lister set flags

lister clear flags

lister query display

lister set display

lister query separate

lister set separate

lister query sort

```
lister set sort  
lister query lock  
lister set lock
```

1.123 DOpus Magellan II ARexx: 'lister set label'.

```
lister set <handle> label [<label>]
```

This command can be used to set the label that will be displayed beneath the iconified lister.

`handle` is the handle of the lister in question.

`label` is the icon-label string you want.

To remove a custom label, simply use this command with no label specified.

For example,

```
+ lister set 121132636 label Custom Lister  
+ lister set 121132636 label
```

See also:

```
lister query label  
lister set field  
lister set title  
lister set header
```

1.124 DOpus Magellan II ARexx: 'lister set lock'.

```
lister set <handle> lock <type> (0|1|off|on)
```

`handle` is the handle of the lister in question.

The `type` parameter may currently be state or format:-

state

The state parameter allows you to lock the lister to its current state (source/dest/off/etc) so the user will be unable to change it until you unlock it.

format

The format parameter allows you to lock the lister to its current display format. Currently this just prevents the user bringing up the Format Edit Requester and it is still possible to change the format with the field gadgets, if enabled -- hopefully this will be fixed in the future.

After the type parameter, add 1 or "on" to turn it on, and 0 or "off" to turn it off.

Note that "locked" in this sense is nothing to do with the "locked source" and "locked destination" states which you can put listers in to have multiple source or destination listers, nor is it anything to do with when a lister is "locked" in position.

You can string these commands on the one line.

For example,

```
+ lister set 121132636 lock state on format on
```

See also:

```
lister query lock
lister set source
lister set dest
lister set off
lister query show
lister set show
lister query hide
lister set hide
lister query flags
lister set flags
lister clear flags
lister query display
lister set display
lister query sort
lister set sort
lister query separate
lister set separate
```

1.125 DOpus Magellan II ARexx: 'lister set mode'.

```
lister set <handle> mode <mode>
```

This command sets the mode of the lister.

handle is the handle of the lister in question.

Valid lister modes are:-

```
name          - name mode
icon          - workbench style icon mode
icon action   - icon action mode
```

You can also add showall to either of the icon modes to tell them to display files without icons as well as those with by giving them default icons depending on their filetype (see the printed manual for more information).

For example,

```
+ lister set 121132636 mode name
+ lister set 121132636 mode icon action showall
```

See also:

```
lister query mode
```

```
lister set source
```

```
lister set dest
```

```
lister set off
```

```
lister query lock
```

```
lister set lock
```

1.126 DOpus Magellan II ARexx: 'lister set namelength'.

```
lister set <handle> namelength <length>
```

This command sets the maximum length allowed for filenames in the lister.

handle is the handle of the lister in question.

length is the new length to set, in characters.

The minimum length is 30 characters which is also the default length (and the current filename length limit of AmigaDOS).

This command will only be useful for writers of
custom handlers

.

Note that the internal Opus commands, for the most part, do not currently support filenames longer than 30 characters.

For example,

```
+ lister set 121132636 namelength 256
```

See also:

```
lister query namelength
```

1.127 DOpus Magellan II ARexx: 'lister set newprogress'.

```
lister set <handle> newprogress [name] [file] [info] [info2] [ ←
info3] [bar] [abort]
```

This turns the progress indicator on for the specified lister.

This is similar to the old

```
lister set progress
```

```
command, but allows
```

greater control over the information displayed.

handle is the handle of the lister in question.

(If you don't want to open the progress window over a lister, use the

```
dopus progress
command instead.)
```

```
name - allocates space for filename display
file - allocates space for file progress display
info - allocates space for information line 1
info2 - allocates space for information line 2
info3 - allocates space for information line 3
bar - allocates space for progress bar
abort - adds "Abort" gadget
```

Progress windows that show both the bar graph and the file progress will have the graph and file displays swapped around. This means that instead of the graph showing the percentage of files copied, and a 'xx%' display showing the progress of that file, the graph shows the file progress and a 'xxx of yyy' display gives overall information.

For example,

```
+ lister set 121132636 newprogress name file info bar abort
```

The following commands are for use once your progress window is open.

```
lister set <handle> newprogress name <filename>
```

If the progress bar was opened with the name parameter, this will set the current filename to filename .

For example,

```
+ lister set 121132636 newprogress name 'myfile.txt'
```

```
lister set <handle> newprogress file <total> <count>
```

If the progress indicator was opened with the file but not the bar parameter, this will set the total number of files and the number of the current file. This is shown as 'xx%' in the top right of the requester.

If the progress indicator was opened with both the file and bar parameters, this will set the total number of bytes and the current byte count. This is shown in the bar graph part of the requester.

```
+ lister set 121132636 newprogress file 12 4
```

```
lister set <handle> newprogress <info|info2|info3> <text>
```

text is a text string to be displayed between the filename and the bar graph of the progress indicator. You can only use this if you allocated this space when you created the progress bar.

For example,

```
+ lister set 121132636 newprogress info "From 'T' to 'Ram:'"
```

```
lister set <handle> newprogress bar <total> <count>
```

If the progress indicator was opened with the bar but not the file parameter, this will set the total number of bytes and the current byte count. This is shown in the bar graph part of the requester.

If the progress indicator was opened with both the file and bar parameters, this will set the total number of files and the the current file count. This is shown as 'xxx of yyy' in the top right of the requester.

For example,

```
+ lister set 121132636 newprogress bar 1024 100
```

```
lister set <handle> newprogress title <text>
```

text is a text string to be displayed in the title bar of the progress indicator.

For example,

```
+ lister set 121132636 newprogress title 'Copying...'
```

You can use the old

```
lister set progress  
commands on a "newprogress"
```

indicator, but obviously they can only change the filename and bar count. Use

```
lister clear progress  
to remove either the "old" or "new" progress
```

indicators.

See also:

```
lister set progress
lister query abort
lister clear progress
lister clear abort
dopus progress
```

1.128 DOpus Magellan II ARexx: 'lister set off'.

```
lister set <handle> off
```

Turns the lister off (ie neither source nor destination).

`handle` is the handle of the lister in question.

For example,

```
+ lister set 121132636 off
```

See also:

```
lister set source
lister set dest
lister query lock
lister set lock
lister query mode
lister set mode
```

1.129 DOpus Magellan II ARexx: 'lister set path'.

```
lister set <handle> path <path string>
```

Sets the current path string in the lister.

`handle` is the handle of the lister in question.

path string is the new string you want.

Note that this does not cause the directory to be read, it merely changes the displayed string. To read a new directory, use the

```
lister read
command.
```

For example,

```
+ lister set 121132636 path 'dh0:work'
```

See also:

```
lister query path
```

```
lister read
```

1.130 DOpus Magellan II ARexx: 'lister set position'.

```
lister set <handle> position <x/y/w/h>
```

This sets the current position and size of the lister if it has not been locked.

handle is the handle of the lister in question.

The position is specified by the <x/y/w/h> parameter:

```
x      - x-coordinate for top-left of lister window;
y      - y-coordinate for top-left of lister window;
w      - width of the lister window;
h      - height of the lister window.
```

If the lister is visible the window will be moved immediately.

For example,

```
+ lister set 121132636 position 20/20/400/300
```

See also:

```
lister query position
```

1.131 DOpus Magellan II ARexx: 'lister set progress'.

Please note that the "lister set progress" commands have now been superseded by the

```
lister set newprogress
commands. Please use those
```

commands in any new scripts.

```
lister set <handle> progress <total> <text>
```

This turns the progress indicator on in the specified lister.

`handle` is the handle of the lister in question.

`total` specifies the total amount to be processed, and controls the bar graph display.

`text` is a text string to be displayed in the title bar of the progress indicator.

For example,

```
+ lister set 121132636 progress 38 'Archiving files...'
```

```
lister set <handle> progress count <count>
```

This updates the bar graph display in the progress indicator (which must have already been turned on).

`count` is the current progress count to be indicated by the bar graph.

For example,

```
+ lister set 121132636 progress count 4
```

```
lister set <handle> progress name <name>
```

This updates the filename display in the progress indicator. The filename is displayed above the bar graph.

For example,

```
+ lister set 121132636 progress name 'myfile.txt'
```

See also:

```
lister set newprogress
```

```
lister query abort
```

```
lister clear progress
```

```
lister clear abort
```

```
dopus progress
```

1.132 DOpus Magellan II ARexx: 'lister set separate'.

```
lister set <handle> separate <method>
```

Sets the separation method for the lister.

`handle` is the handle of the lister in question.

`method` can be any one of the following:-

```
mix           - mix files and directories
dirsfirst    - directories first
filesfirst   - files first
```

The list is rearranged immediately, but the display will not be updated until you execute a

```
lister refresh
command.
```

For example,

```
+ lister set 121132636 separate mix
```

See also:

```
lister query separate
```

```
lister query show
```

```
lister set show
```

```
lister query hide
```

```
lister set hide
```

```
lister query flags
```

```
lister set flags
```

```
lister clear flags
```

```
lister query display
```

```
lister set display
```

```
lister query sort
```

```
lister set sort
```

```
lister query lock
```

```
lister set lock
```

1.133 DOpus Magellan II ARexx: 'lister set show'.

```
lister set <handle> show <pattern>
```

Sets the show pattern for the lister.

`handle` is the handle of the lister in question.

`pattern` is any valid AmigaDOS file pattern.

The pattern is applied immediately but the display is not updated until you execute a

```
lister refresh
command.
```

For example,

```
+ lister set 121132636 show '#?.c'
```

See also:

```
lister query show
```

```
lister query hide
```

```
lister set hide
```

```
lister query flags
```

```
lister set flags
```

```
lister clear flags
```

```
lister query display
```

```
lister set display
```

```
lister query separate
```

```
lister set separate
```

```
lister query sort
```

```
lister set sort
```

```
lister query lock
```

```
lister set lock
```

1.134 DOpus Magellan II ARexx: 'lister set sort'.

```
lister set <handle> sort <method>
```

Sets the sort method for this lister.

`handle` is the handle of the lister in question.

`method` can be any one of the following:-

name	- filename
size	- file size
protect	- protection bits
date	- datestamp
comment	- comment
filetype	- file type
version	- file version

The list is resorted immediately, but the display will not be updated until you execute a

```
lister refresh
command.
```

For example,

```
+ lister set 121132636 sort date
+ lister set 121132636 sort filetype
```

See also:

```
lister query sort
lister query show
lister set show
lister query hide
lister set hide
lister query flags
lister set flags
lister clear flags
lister query display
lister set display
lister query separate
lister set separate
lister query lock
lister set lock
```

1.135 DOpus Magellan II ARexx: 'lister set source'.

```
lister set <handle> source [lock]
```

Makes the lister the source.

handle is the handle of the lister in question.

If you specify the lock keyword, it will be locked as a source.

For example,

```
+ lister set 121132636 source lock
```

See also:

```
lister set dest
```

```
lister set off
```

```
lister query lock
```

```
lister set lock
```

```
lister query mode
```

```
lister set mode
```

1.136 DOpus Magellan II ARexx: 'lister set title'.

```
lister set <handle> title <string>
```

Sets the title for the lister (the title displayed in the lister title bar).

handle is the handle of the lister in question.

string is the text you want to replace the title with.

The old title is returned in

```
RESULT
```

```
.
```

The title bar display will not be updated until you execute a

```
lister refresh <handle> full  
command.
```

For example,

```
+ lister set 121132636 title 'hello'  
> RESULT  
+ lister set 121132636 title  
> hello
```

See also:

```
lister query label
```

```
lister set label
```

```
lister set field  
lister set header
```

1.137 DOpus Magellan II ARexx: 'lister set toolbar'.

```
lister set <handle> toolbar <toolbarname>
```

This command changes the toolbar that is used in the lister.

`handle` is the handle of the lister in question.

`toolbarname` is the name of the toolbar to use, either its full path or just its filename if it's in the DOpus5:Buttons directory.

For example,

```
+ lister set 121132636 toolbar Ram:custom_toolbar
```

See also:

```
lister query toolbar
```

1.138 DOpus Magellan II ARexx: 'lister set value'.

```
lister set <handle> value <name> <value>
```

This command lets you associate your own data with a lister in the form of name/value pairs. The lister will maintain the values until it is closed.

For example,

```
+ lister set 121132636 value MyName "Jon Citizen"
```

See also:

```
lister query value
```

```
lister clear value
```

1.139 DOpus Magellan II ARexx: 'lister set visible'.

```
lister set <handle> visible <state>
```

Sets the visible status for this lister.

`handle` is the handle of the lister in question.

By default, listers are visible when they are created.

If you set this `state` to 0 or "off", the lister will disappear from the display until you make it visible again (with 1 or "on").

For example,

```
+ lister set 121132636 visible off
+ lister set 121132636 visible 1
```

See also:

```
lister query visible
```

```
lister new invisible
```

1.140 DOpus Magellan II ARexx: 'lister select'.

```
lister select <handle> <name> <state>
```

This command changes the selection status of an entry in the lister.

`handle` is the handle of the lister in question.

`name` is either the name of the entry, or #xxx (where xxx is a number) to specify the cardinal number of the entry.

`state` is the desired selection status (0 or "off" for off, 1 or "on" for on). If `state` is not given then the state of the entry is toggled.

The display is not refreshed until you execute a

```
lister refresh
command.
```

The previous selection state of the entry is returned in

```
RESULT
```

```
.
```

For example,

```
+ lister select 121132636 ENV on
> off
```

See also:

```
lister query firstsel
```

```
lister query numseldirs
```

```
lister query numselentries
```

```
lister query numselfiles
lister query seldirs
lister query selentries
lister query selfiles
```

1.141 DOpus Magellan II ARexx: 'lister wait'.

```
lister wait <handle> [quick]
```

This command causes the rexx script to wait for the specified lister to finish whatever it is doing.

Because Opus 5 multitasks, all rexx commands (like

```
lister read
, or
```

```
lister new
```

) will return immediately, even if the lister has not completed its task. This command will force the script to wait until the lister goes non-busy.

If the lister is not in a busy state when this command is called, the program will wait for up to two seconds for it to go busy, otherwise the call is aborted.

handle is the handle of the lister in question.

If the quick keyword is specified, the command will return immediately if the lister is not busy, instead of waiting for two seconds.

It would be silly to do

```
lister set busy 1
and then lister wait.
```

For example,

```
+ lister read 121132636 'c:'
+ lister wait 121132636
```

See also:

```
lister query busy
```

```
lister set busy
```

1.142 DOpus Magellan II ARexx: 'command'.

```

        command [wait] [source <handle>] [dest <handle>] [original]
command [<arguments>]

```

The `command` command allows you to call the internal commands of Directory Opus 5 from an ARexx script. The commands execute exactly as if they had been run from a custom button or menu.

If the `wait` flag is specified, the command will be run synchronously, otherwise it will return immediately.

```

        RC
will contain a
result code indicating whether the command was successful or not.

```

Ordinarily, commands operate on the current source and destination listers - the `source` and `dest` parameters allow you to specify alternative listers to use by their handles.

The `original` flag allows you to run an original Opus internal function if the command has been replaced in the command list by an

```

        external module
(
        external modules
        which add commands to Opus override the internal list).

```

This means you could have a module that replaced some Opus commands, did something special in some circumstances, and in others just called the original Opus function.

The `command` parameter is the name of the command, and `arguments` are any optional arguments for the command, as normal.

Some examples,

```

+ command all
+ command wait copy
+ command read s:startup-sequence
+ command source 121132636 mkdir name=MyDir noicon
+ command original wait delete ram:#?

```

1.143 DOpus Magellan II ARexx: Error Codes.

```

        ARexx Error Codes
-----

```

Lister handles are the actual address in memory of the lister structure. Opus 5 will reject any non-valid handles with an

```

        RC
of 10.

```

All commands that return data return it in

```

        RESULT
(with the exception of

```

```

        dopus getstring
and
        lister getstring

```

) or a specified (stem) variable;
 if an error occurs, the error code is returned in
 RC
 .

An

RC
 of 0 generally indicates that everything is ok.

Error codes are:-

- 1 RXERR_FILE_REJECTED
 The file you tried to add was rejected by the current lister
 filters.
 Note that this is not an error, just a warning. The file is still
 added, it will just not be visible until the filters are changed.
- 5 RXERR_INVALID_QUERY
 RXERR_INVALID_SET
 The query/set item you specified was invalid.
- 6 RXERR_INVALID_NAME
 RXERR_INVALID_KEYWORD
 The filename or keyword you specified was invalid.
- 8 RXERR_INVALID_TRAP
 The trap you tried to remove didn't exist.
- 10 RXERR_INVALID_HANDLE
 The lister handle you gave was invalid.
- 12 RXERR_NO_TOOLBAR
 The lister has no valid toolbar.
- 15 RXERR_NO_MEMORY
 There wasn't enough memory to do what you wanted.
- 20 RXERR_NO_LISTER
 A lister failed to open (usually because of low-memory).

You can convert the error codes returned in
 RC
 into meaningful error
 messages (for error reports and so on) with the
 dopus error
 command.

1.144 DOpus Magellan II ARexx: Custom Handlers.

Custom Handlers

The Basics

```

Custom Handlers for Listers
-
List of events

Custom Handlers for AppIcons

```

1.145 DOpus Magellan II ARexx: Custom Handlers, The Basics.

Custom Handlers, The Basics.

----- -- - -

The custom handler system allows you to specify the name of an external public message port. This port will be sent messages whenever certain things happen that you are interested in. Messages that are sent are properly formatted ARexx messages.

An example code fragment to receive a message is:

```

call waitpkt(myportname)      /* wait for messages to arrive */

packet=getpkt(myportname)     /* get waiting message */
arg0=getarg(packet,0)        /* get Argument 0 */
arg1=getarg(packet,1)        /* get Argument 1 */
arg2=getarg(packet,2)        /* get Argument 2, etc... */

call reply(packet,0)         /* reply to the message */

```

Because of the multi-tasking nature of Opus 5, information custom handlers receive can not be 100% relied on. For example, you may receive an "active" message, but the cache that caused it may have immediately gone "inactive" again. You should therefore check your port is clear of all messages before processing any that have come in, and you should also use the

```

lister query
command to make sure that things are how you expect them.

```

Also note that listers (unless you have turned

```

busy
on) can be closed by

```

the user at any time. An "inactive" message is sent when the lister is closed. To check that a lister is still open, use the

```

lister query path
command (or any other
query
command). If the lister no longer exists,

```

```

RC
will contain the error code
XERR_INVALID_HANDLE
(10). Be aware,

```

though, that while these possibilities exist, generally they will not cause a problem. For the most part it will only be if the user is "playing around" that weird situations will occur.

1.146 DOpus Magellan II ARexx: Custom Handlers for Listers.

Custom Handlers for Listers

----- -- - -
 A custom handler is "attached" to a lister by calling

```

    lister set <handle> handler
    for that lister, giving the name of your
  message port. Whenever something interesting happens to your lister, the
  handler will be sent an ARexx message.
  
```

The handler can be implemented either as a rexx program or as a C program (in which case it must interpret the rexx message itself).

Unlike Opus 4, messages sent to handlers do not cause Directory Opus 5 to "hang" until they are replied (although you should try to reply to any messages as soon as possible).

Note that custom handlers for listers are specific only to the cache that is visible in the lister at the time the handler name is set. The same handler port may be used set for multiple caches, and indeed for multiple listers.

Note also that message port names are case-sensitive. Be careful that ARexx doesn't uppercase them.

The rexx message identifies the type of event, the lister the event happened to, and other pertinent data.

List of events

1.147 DOpus Magellan II ARexx: Lister Handlers, Events.

Custom Handlers for Listers: The Events

----- -- - -
 The rexx message identifies the type of event, the lister the event happened to, and other pertinent data.

The events that you will be notified of are:-

active

doubleclick

drop

dropfrom

edit

inactive


```
parent
path
reread
root
snapshot
unsnapshot
Trapped Functions
AddStem Pop-Ups
Lister2 Pop-Ups
```

1.148 DOpus Magellan II ARexx: Lister Handlers, 'active'

Custom Handlers for Listers: active

This event indicates that a cache with a custom handler attached has just become visible.

The message arguments are:-

```
Arg0 - "active"           (a string indicating the event type)
Arg1 - <handle>          (lister handle)
Arg2 - <title>           (cache title)
Arg3 - undefined
Arg4 - <path>            (path of the lister)
```

Arg2 will contain the custom title of the cache that became active, if it has been set with

```
lister set title
```

. If no custom title has been defined,

the path string of the cache is returned instead (ie in this case Arg2 will be the same as Arg4).

1.149 DOpus Magellan II ARexx: Lister Handlers, 'doubleclick'

Custom Handlers for Listers: doubleclick

This is a double-click event, and indicates that an item in the lister has been double-clicked on by the user.

The message arguments are:-

```

Arg0 - "doubleclick"      (a string indicating the event type)
Arg1 - <handle>           (lister handle)
Arg2 - <name>             (entry name)
Arg3 - undefined
Arg4 - undefined
Arg5 - <userdata>        (if userdata was specified with the
                           lister addstem
                           command)
Arg6 - <qualifiers>       (string indicating qualifiers pressed
                           - shift, alt, and control keys)

```

1.150 DOpus Magellan II ARexx: Lister Handlers, 'drop'

Custom Handlers for Listers: drop

This is a drag'n'drop event, and indicates that one or more entries have been dropped into a lister.

The message arguments are:-

```

Arg0 - "drop"             (a string indicating the event type)
Arg1 - <handle>           (destination lister handle)
Arg2 - <names>            (filenames)
Arg3 - <handle>           (source lister handle)
Arg4 - undefined
Arg5 - <destination>     (full destination path)
Arg6 - <qualifiers>       (string indicating qualifiers pressed
                           - shift, alt, control keys, or whether drop
                           was over a sub-directory)

```

The filenames are separated by spaces (if there is more than one), and will be within quotes if the quotes keyword was specified for the

```

lister set handler
command.

```

Arg5 contains the full destination path. You can compare this against the path of the destination lister (handle in Arg1) to see whether the drop was into a sub-directory or into the lister itself.

You should not rely on the ARexx Word() function to traverse the list of filenames because it does not support quotes and any name with a space in it will cause your script to malfunction.

If the files originated from another Opus 5 lister, Arg3 gives the handle of that lister. If this is the case, and the fullpath option was not specified for

```

lister set handler
, only the filenames (and not their
paths) are supplied in Arg2 (you can get the source path using

```

```

lister query path
). If Arg3 is null then the drop most likely originated

```

from Workbench, and the names in Arg2 include the full paths.

If the files are dropped over a sub-directory in the lister then Arg6 will contain the qualifier 'subdrop', providing that the handler was invoked using

```
lister set handler
with the subdrop keyword.
```

1.151 DOpus Magellan II ARexx: Lister Handlers, 'dropfrom'

Custom Handlers for Listers: dropfrom

This is basically the same as the drop event, except that it indicates a drop from a lister rather than a drop to one.

The message arguments are:-

Arg0 - "dropfrom"	(a string indicating the event type)
Arg1 - <handle>	(source lister handle)
Arg2 - <names>	(filenames)
Arg3 - <handle>	(destination lister handle)
Arg4 - undefined	
Arg5 - <destination>	(path the file was dropped into)
Arg6 - <qualifiers>	(string indicating qualifiers pressed - shift, alt, control keys, or whether drop was over a sub-directory)

The filenames are separated by spaces (if there is more than one), and will be within quotes if the quotes keyword was specified for the

```
lister set handler
command.
```

Arg5 of the 'dropfrom' custom handler message contains the destination path; that is, the path that the file was dropped into. If the file was dropped onto an icon, the destination path will be that of the object it was dropped onto. For example, if a file from your custom lister is dropped onto the Prefs icon in the Workbench lister, Arg5 will contain "DH0:Prefs/", while Arg3 will contain the handle of the destination lister.

If Arg3 is 0 the drop was probably to the desktop. If the file was dropped onto an icon on the desktop, Arg5 will contain the path. If the file was dropped onto empty space on the desktop, Arg5 will contain the string 'desktop'. If you get the string 'desktop' you can find the proper path to copy the file to (or whatever) using the getdesktop command.

If the files are dropped over a sub-directory in the lister then Arg6 will contain the qualifier 'subdrop', providing that the handler was invoked using

```
lister set handler
```

with the subdrop keyword.

Note that if the user drops a file onto an icon that is not a disk or a drawer, you will still get the full pathname of the icon in Arg5. It is up to you to determine whether the given destination path is a drawer or a file and act appropriately.

Also note: the old Arg3 was a little bit confusing; it was the destination lister handle if there was one, otherwise it was the source lister handle. This has now been changed; Arg3 is ALWAYS the destination lister handle, and 0 if there is none. Arg1 is still the source lister handle.

You should not rely on the ARexx Word() function to traverse the list of filenames because it does not support quotes and any name with a space in it will cause your script to malfunction.

Note that

```

    appicons
    can also receive dropfrom events but they have
slightly different arguments. They can be distinguished by the word
"icon" always present in Arg4. See the
    appicons
    section.
```

1.152 DOpus Magellan II ARexx: Lister Handlers, 'edit'

Custom Handlers for Listers: edit

This event indicates that the cache this custom handler is attached to is no longer active (visible in the lister).

This event indicates that an entry has been changed via inline editing.

The message arguments are:-

Arg0 - "edit"	(a string indicating the event type)
Arg1 - <handle>	(lister handle)
Arg2 - <name>	(entry name)
Arg3 - <field>	("name", "protect", "date", or "comment")
Arg4 - <new value>	(value of string after editing)
Arg5 - <userdata>	(userdata specified via the

```

    lister addstem
    command)
```

This message will only be sent if inline editing was enabled by specifying the editing option of the

```

    lister set handler
    command.
```

1.153 DOpus Magellan II ARexx: Lister Handlers, 'inactive'

Custom Handlers for Listers: inactive

This event indicates that the cache this custom handler is attached to is no longer active (visible in the lister).

The message arguments are:-

Arg0 - "inactive"	(a string indicating the event type)
Arg1 - <handle>	(lister handle)
Arg2 - <title>	(cache title)
Arg3 - undefined	
Arg4 - <path>	(path of the lister)

This message is caused by the cache in the lister being changed (either by the user or under rexx control), or even by the lister being closed. Note that you may receive an active message for another cache with a custom handler, or even for the same cache, immediately after receiving an inactive message.

1.154 DOpus Magellan II ARexx: Lister Handlers, 'parent'

Custom Handlers for Listers: parent

This event will be received when the Parent Directory item is chosen from the lister cache pop-up menu, or whenever the user clicks on the border parent gadget or uses the parent hot key, "/".

The message arguments are:-

Arg0 - "parent"	(a string indicating the event type)
Arg1 - <handle>	(source lister handle)
Arg2 - <path>	(path of lister)
Arg3 - undefined	
Arg4 - undefined	
Arg5 - undefined	
Arg6 - <qualifiers>	(string indicating qualifiers pressed - only shift keys)

1.155 DOpus Magellan II ARexx: Lister Handlers, 'path'

Custom Handlers for Listers: path

When the user enters a new path in the path gadget of a lister you will receive this message.

The arguments are:-

Arg0 - "path"	(a string indicating the event type)
Arg1 - <handle>	(lister handle)
Arg2 - <path>	(path of lister)

1.156 DOpus Magellan II ARexx: Lister Handlers, 'reread'

Custom Handlers for Listers: reread

The reread event will be sent to your handler when the Re-read Directory item is chosen from the lister cache pop-up menu.

Its arguments are:-

Arg0 - "reread"	(a string indicating the event type)
Arg1 - <handle>	(lister handle)
Arg2 - <path>	(new path for lister)

1.157 DOpus Magellan II ARexx: Lister Handlers, 'root'

Custom Handlers for Listers: root

A root event will be received when the Root Directory item is chosen from the lister cache pop-up menu, or whenever the user uses the root hot key, ":".

The message arguments are:-

Arg0 - "root"	(a string indicating the event type)
Arg1 - <handle>	(source lister handle)
Arg2 - <path>	(path of lister)
Arg3 - undefined	
Arg4 - undefined	
Arg5 - undefined	
Arg6 - <qualifiers>	(string indicating qualifiers pressed - only shift keys)

1.158 DOpus Magellan II ARexx: Lister Handlers, 'snapshot'

Custom Handlers for Listers: 'snapshot'

The snapshot event will be sent to your handler when the user snapshots a lister under your control.

Its arguments are:-

Arg0 - "snapshot"	(a string indicating the event type)
Arg1 - <handle>	(lister handle)

Unlike

snapshot for AppIcons
, the lister's position is not sent with the
event and it is up to you to get it using
lister query position
.

It is also up to you to save the information for later use, if you wish to

support snapshots on your handler.

1.159 DOpus Magellan II ARexx: Lister Handlers, 'unsnapshot'

Custom Handlers for Listers: 'unsnapshot'

The unsnapshot event will be sent to your handler when the user un-snapshots a lister under your control.

Its arguments are:-

Arg0 - "unsnapshot"	(a string indicating the event type)
Arg1 - <handle>	(lister handle)

It is up to you to remember that the lister has been un-snapshotted if you wish to support snapshots on your handler.

1.160 DOpus Magellan II ARexx: Lister Handlers, Trapped Functions

Custom Handlers for Listers: Trapped Functions

Messages for

trapped commands
are sent to the lister much like the other

messages.

The message arguments are:-

Arg0 - <command> 'abort')	(name of the command, or)
Arg1 - <handle>	(source lister handle, if any)
Arg2 - <files>	(selected files, if any)
Arg3 - <handle>	(destination lister handle, if any)
Arg4 - <path>	(source path; useful if there's no lister associated with it)
Arg5 - <args>	(user-supplied arguments to the function)
Arg7 - <path>	(destination path; allows you to support the Select Destination requester)
Arg8 - <functionhandle>	(If <code>synctraps</code> flag is set, contains address in decimal of structure for this function)

The filenames are separated by spaces (if there is more than one), and will be within quotes if the quotes keyword was specified for the

lister set handler
command.

You should not rely on the ARexx Word() function to traverse the list of filenames because it does not support quotes and any name with a space

in it will cause your script to malfunction.

1.161 DOpus Magellan II ARexx: Lister Handlers, AddStem Pop-Ups

Custom Handlers for Listers: AddStem Pop-Ups

If you have added files to a lister with
`lister addstem`
 and specified
 your own pop-up menus for the files, you will receive messages when these
 menus are chosen by the user.

The message arguments are:-

Arg0 - "menu"	(string identifies this as a menu event)
Arg1 - <handle>	(lister handle)
Arg2 - <name>	(entry name)
Arg3 - <id>	(ID of the menu item + base ID if specified)
Arg4 - "file"	(string identifying this as a "file" menu event)
Arg5 - <userdata>	(userdata specified via the <code>lister addstem</code> <code>command</code>)

1.162 DOpus Magellan II ARexx: Lister Handlers, Lister Pop-Ups

Custom Handlers for Listers: Lister Pop-Ups

If you have added menu items to the lister pop-up menu, accessed by using the
 LMB over the SRCE/DEST status display, then messages will be received when
 these menu items are chosen.

The message arguments are:-

Arg0 - "menu text"	(the menu item text you added with 'dopus command' ext parameter)
Arg1 - <handle>	(lister handle)

1.163 DOpus Magellan II ARexx: Custom Handlers for AppIcons.

Custom Handlers for AppIcons

 Like listers with handles, AppIcons added with the
`addappicon`
`command`
 will also cause messages to be sent. See the
`lister handlers`

section

for more details.

All AppIcon messages have the same arguments:-

Arg0 - <event>	(string identifying the event)
Arg1 - <id>	(ID specified in the addappicon command)
Arg2 - <data>	(filenames/menu ID/other information)
Arg3 - <handle>	(source lister handle - if applicable)
Arg4 - "icon"	(string identifying this as an "icon" event)

These are the events that apply to AppIcons:-

doubleclick

This event indicates that an icon has been double-clicked, or has had "Open" selected from its menu. ("Open" is replaced with "Close" if you give the close option to

```
dopus addappicon
, and you'll get close events
```

instead.)

dropfrom

This is a drag'n'drop event, and indicates that one or more entries have been dropped on this appicon from a lister or elsewhere. The names of the entries are available in Arg2. The names will be surrounded by quotes if you gave the quotes option to

```
dopus addappicon
.
```

You should not rely on the ARexx Word() function to traverse the list of filenames because it does not support quotes and any name with a space in it will cause your script to malfunction.

snapshot

This event occurs when the Snapshot menu item is selected. The current position of the icon is available in Arg2 (as an x,y string). You are responsible for storing this position. (You will only get snapshot events if you give the snap option to

```
dopus addappicon
.)
```

unsnapshot

This event occurs when the Un-Snapshot menu item is selected. (You will only get unsnapshot events if you give the snap option to

```
dopus addappicon
.)
```

removed

This event warns that Opus has quit and the handler code should now clean up and exit.

info

This event occurs when the Information menu item is selected. (You will only get info events when you give the info option to

```
dopus addappicon
.)
```

close

This will occur when Close is selected from the pop-up menu. (You will only get close events when you give the close option to

```
dopus addappicon
.)
```

menu

This event indicates that one of the user-supplied menu items has been selected in the pop-up menu. The number of the menu item will be returned in Arg2, and your base value will have been added to it if you specified one when you called

```
dopus addappicon
.
```

menuhelp

This event indicates that the help key was pressed while the mouse pointer was over one of the user-supplied menu items. The number of the menu item will be returned in Arg2, and your base value will have been added to it if you specified one when you called

```
dopus addappicon
.
```

1.164 DOpus Magellan II ARexx: ARexx Modules.

ARexx Modules

ARexx Modules are ARexx scripts which are installed in the DOpus5:Modules directory. They must have the suffix .dopus5 to work correctly.

Each ARexx Module can add new internal commands to Opus.

Once both Opus 5 and ARexx have been started on the computer, the DOpus5:Modules directory will be scanned for ARexx Modules and each will have its init function called. Every ARexx Module must have an init function or else it will not work. This is the function that adds the extra commands to Opus.

The scripts are called with 4 or more parameters. The first 4 are always provided - the portname of Directory Opus 5 and the function name, followed by the source and destination lister handles. Any user-supplied arguments to the function will follow.

Note that although the source and destination lister handle arguments are always given when your script is called, they will always be zero unless you specified that you wanted source and/or destination handles when you added the command.

You can add as many commands as you like. To add commands, use the

```
dopus command
command.
```

Here is a complete example of an ARexx Module:-

```
/* Example Directory Opus 5 ARexx Module */

parse arg portname function source dest arguments
address value portname
options results

/* Initialise */

if function='init' then do
    dopus command "Test1" program "test-command" desc "'Test command 1'" ←
        template "TEST/S"
    dopus command "Test2" program "test-command" desc "'Test command 2'" ←
        source"
    exit
end

/* Test function 1 */

if function='Test1' then do
    dopus request "'Test command 1 received!'" "Ok"
    exit
end

/* Test function 2 */

if function='Test2' then do
    str="'Test command 2 received - source handle " || source
    dopus request str "Ok"
    exit
end
```

1.165 DOpus Magellan II ARexx: Guide Index.

Index

abort
active
active
add
addappicon
addstem
AddStem Pop-Ups

addtrap
all
all
AppIcons
ARexx Modules.
back
background
background
background
busy
busy
case
case
centre
clear
clear
clear value
close
command
command
Commands
commands
commands
commentlength
commentlength
copy
Credits
custom

custom

Custom Handlers

Custom Handlers for AppIcons

Custom Handlers for Listers

Custom Handlers, The Basics

desktoppopup

dest

dest

dirs

display

display

dopus addappicon

dopus addtrap

dopus back

dopus clear

dopus command

dopus commands

dopus desktoppopup

dopus error

dopus front

dopus getfiletype

dopus getstring

dopus matchdesktop

dopus progress

dopus query

dopus query background

dopus query font

dopus query palette

dopus query pens

dopus query sound
dopus read
dopus refresh
dopus refresh all
dopus refresh background
dopus refresh icons
dopus refresh lister
dopus remappicon
dopus remtrap
dopus request
dopus screen
dopus send
dopus set
dopus set background
dopus set font
dopus set palette
dopus set pens
dopus set sound
dopus setappicon
dopus version
doubleclick
drop
dropfrom
empty
entries
entry
error
Error Codes

Events

field

files

firstsel

flags

flags

flags

font

font

France-Festival-Distribution (FFD)

front

getfiletype

getstring

getstring

Guide Credits

Guide History

handler

handler

header

header

hide

hide

History

iconify

icons

inactive

Introduction

label

label

- lister
- lister add
- lister addstem
- lister clear
- lister clear value
- lister close
- lister commands
- lister copy
- lister empty
- lister getstring
- Lister Handlers, active
- Lister Handlers, AddStem Pop-Ups
- Lister Handlers, doubleclick
- Lister Handlers, drop
- Lister Handlers, dropfrom
- Lister Handlers, edit
- Lister Handlers, Events
- Lister Handlers, inactive
- Lister Handlers, lister2 pop-ups
- Lister Handlers, parent
- Lister Handlers, path
- Lister Handlers, reread
- Lister Handlers, root
- Lister Handlers, snapshot
- Lister Handlers, Trapped Functions
- Lister Handlers, unsnapshot
- lister iconify
- lister new

lister query
lister query abort
lister query active
lister query all
lister query busy
lister query case
lister query commentlength
lister query dest
lister query dirs
lister query display
lister query entries
lister query entry
lister query files
lister query firstsel
lister query flags
lister query handler
lister query header
lister query hide
lister query label
lister query lock
lister query mode
lister query namelength
lister query numdirs
lister query numentries
lister query numfiles
lister query numseldirs
lister query numselentries
lister query numselfiles
lister query path

lister query position
lister query proc
lister query seldirs
lister query selentries
lister query selfiles
lister query separate
lister query show
lister query sort
lister query source
lister query title
lister query toolbar
lister query value
lister query visible
lister read
lister refresh
lister remove
lister request
lister select
lister set
lister set busy
lister set case
lister set commentlength
lister set dest
lister set display
lister set field
lister set flags
lister set handler
lister set header

lister set hide
lister set label
lister set lock
lister set mode
lister set namelength
lister set newprogress
lister set off
lister set path
lister set position
lister set progress
lister set separate
lister set show
lister set sort
lister set source
lister set title
lister set toolbar
lister set value
lister set visible
lister wait
lock
lock
matchdesktop
mode
mode
namelength
namelength
new
newprogress
numdirs

numentries

numfiles

numseldirs

numselentries

numselfiles

off

Official Notice From GPSoftware: FFD

palette

palette

parent

path

path

path

pens

pens

precision

position

position

proc

progress

progress

Purchasing Directory Opus 5.5

query

query

query abort

query active

query all

query background

query busy
query case
query commentlength
query dest
query dirs
query display
query entries
query entry
query files
query firstsel
query flags
query font
query handler
query header
query hide
query label
query lock
query mode
query namelength
query numdirs
query numentries
query numfiles
query numseldirs
query numselentries
query numselfiles
query palette
query path
query pens
query position

query proc
query seldirs
query selentries
query selfiles
query separate
query show
query sort
query sound
query source
query title
query toolbar
query value
query visible
read
read
refresh
refresh
refresh all
refresh background
refresh icons
refresh lister
remappicon
remove
remove
remtrap
request
request
reread

Results from commands

root

screen

seldirs

select

selentries

selfiles

send

separate

separate

set

set

set background

set busy

set case

set commentlength

set dest

set display

set field

set flags

set font

set handler

set header

set hide

set label

set lock

set mode

set namelength

set newprogress

set off

set palette

set path

set pens

set position

set progress

set separate

set show

set sort

set sound

set source

set title

set toolbar

set value

set visible

setappicon

show

show

snapshot

sort

sort

source

source

temp

The Basics

The Port

tile

title

title
toolbar
toolbar
Trapped Functions
unsnapshot
value
value
value
version
visible
visible
volume
wait
wait
