AmiSlate

Jeremy Friesner

COLLABORATORS			
	TITLE : AmiSlate		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Jeremy Friesner	July 20, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmiSlate

1.1 AmiSlate ARexx Commands

AmiSlateRexx V1.2 by Jeremy Friesner

This document describes the ARexx commands that are supported by
AmiSlate and how to use them. Although I have tried to be as clear
and accurate as possible, it's likely that some descriptions will
contain errors or ambiguities. With that in mind, it's a good idea
to look at the example ARexx scripts included with the AmiSlate
distribution as well as this document.

Commands prefixed with a "*" are currently not implemented or have
no useful effect. They will be completed in a later release.

Conventions - Things assumed in this document

Commentary - General info on AmiSlate ARexx operation

Common Problems - Common problems and their solutions

BreakARexxScripts - Make AmiSlate try to break all ARexx scripts

Circle - Draw a circle or ellipse

Clear - Clear the Drawing area

Connect - Connect to remote AmiSlate client

Disconnect - Disconnect from remote AmiSlate client

DisplayBeep - DisplayBeep the local or remote screen

EasyRequest - Present user with an EasyRequester

FileRequest - Present user with a FileRequester

Flood - Flood fill an area

GetPixel - Return palette number of pixel

GetRemoteStateAttrs - Get data on remote client's status

GetStateAttrs - Get data on local client's status

GetVersion - Get AmiSlate version information

GetWindowAttrs - Get data on AmiSlate's window size, etc.

Line - Draw a line

LoadIFF - Load an IFF ILBM file onto the canvas

Lock - Keep user from drawing

LockPalette - Lock Palettes

Pen - Draw next line in a chain of lines

PenReset - Start a new line

PlayScript - Playback an AmiSlate script

Point - Plot a point

PutRasterPixels - Send a stream of pixels to raster window

Quit - Cause AmiSlate to quit

RecordScript - Cause AmiSlate to begin recording to a file

RemoteRexxCommand - Start Rexx script on remote client

RemoteStringRequest - Present remote user with StringRequester

SaveIFF - Save the current screen to an IFF ILBM file

SendMessage - Send message to remote client's ARexx program

* SetBColor - Set Background pen by RGB value

* SetBPen - Set Background pen by palette #

SetFColor - Set Foreground pen by RGB value

SetFPen - Set Foreground pen by palette #

SetPenColor - Set the RGB value of a palette position

SetRaster - Set Raster window parameters

SetRemoteWindowTitle - Set remote client's window title

* SetToolBehavior - Set a Tool's behavior

* SetUserBColor - Set the user's background pen by RGB value

* SetUserBPen - Set the user's background pen by palette #

SetUserFColor - Set the user's foreground pen by RGB value

SetUserFPen - Set the user's foreground pen by palette #

SetUserTool - Set the user's selected tool

SetWindowTitle - Set the Window Title

StringRequest - Present user with StringRequester

TransmitPointer - Set whether or not to transmit pointer position

TransmitGraphics - Set whether or not to transmit drawn graphics

TypeKeys - Enter keystrokes into Chat Line

Square - Draw a square or rectangle

WaitEvent - Wait for an event

1.2 conventions

In this document, the following rules and conventions hold true for
all commands:
1) All commands that return multiple results use the ARexxBox convention of
"var" and "stem" variables. That is, you can have your results returned
as a single string into one variable via the "var" keyword, or you
can have your results returned into a stem array via the "stem" keyword.
For example:
remotestringrequest stem answ. ARexx_Test Def_String Gad_Text
will set the stem array variable answ.message to (whatever the user
typed into the requester).
By contrast, the command:
remotestringrequest var answ ARexx_Test Def_String Gad_Text
will set the variable answ to the string the user typed in.
2) Argument meaning is implicit in the ordering of the arguments, and thus
keywords should not be explicitely stated in the command (the exception
being "var" and "stem", above). Arguments will be designated with
<anglebrackets>. Switches are stated explicitely, of course, and will
be designated with [brackets].
3) When an ARexx script is run by an AmiSlate client, it is provided with
two command line arguments: the first being the name of the AmiSlate
client's ARexx port, and the second being either "LOCAL" or "REMOTE"
depending on whether the execution was initiated by the local or the
remote user.
4) Commands that are not yet implemented will have * in front of them.

1.3 Mode Index Numbers

When communicating information about which tool (a.k.a mode) the local
or remote client is in, AmiSlate makes use of the following constants.
It is a good idea to copy the following declarations into the beginning
of your ARexx program in order to later easy refer to different modes.
/* Mode Constants for use with AmiSlate's ARexx interface */
AMode.DOT = 0
AMode.PEN = 1
AMode.LINE = 2
AMode.CIRCLE = 3
AMode.SQUARE = 4

AMode.POLY = 5
AMode.FLOOD = 6
AMode.DTEXT = 7
AMode.FILLCIRCLE = 13
AMode.FILLSQUARE = 14
When replying to the WaitEvent function, AmiSlate uses a set of message
codes to indicate which type of event occured. Note that despite the
binarily-disjoint nature of the following constants, only one type
of event currently is replied per WaitEvent call.
/* Message reply codes for use with AmiSlate's ARexx interface */
AMessage.TIMEOUT = 1 /* No events occurred in specified time period */
AMessage.MESSAGE = 2 /* Message recieved from remote Amiga */
AMessage.MOUSEDOWN = 4 /* Left mouse button press in drawing area */
AMessage.MOUSEUP = 8 /* Left mouse button release in drawing area */
AMessage.RESIZE = 16 /* Window was resized--time to redraw screen? */
AMessage.QUIT = 32 /* AmiSlate is shutting down */
AMessage.CONNECT = 64 /* Connection established */
AMessage.DISCONNECT = 128 /* Connection broken */
AMessage.TOOLSELECT = 256 /* Tool Selected */
AMessage.COLORSELECT = 512 /* Palette Color selected */
AMessage.KEYPRESS = 1024 /* Key pressed */
AMessage.MOUSEMOVE = 2048 /* Key pressed */
Furthermore, when MESSAGE events are replied, the following subtypes
(returned in the code1 result variable) reflect the origin of the
received method:
AMessageOrg.STRINGUSER = 1 /* String was sent by remote ARexx script */
AMessageOrg.STRINGEASYREQ = 2 /* String was sent by remote EasyRequester */
AMessageOrg.STRINGSTRINGREQ = 3 /* String was sent by remote StringRequester */

1.4 commentary

AmiSlate's ARexx port is designed to be able to run in parallel with the
local and remote user's input. Thus, the ARexx port has its own internal
state (current foreground color, etc.) and most ARexx commands will not
interfere with or change the state of the user's settings. (Notable
exceptions to this are the SetUser(*) commands) If you do not want the
user to be drawing while the ARexx script is running, execute a 'lock on'
command.
AmiSlate is based on the idea of keeping the local and remote user's
drawing area as identical as possible. Thus there are no commands
available that draw without sending the drawing info to the remote
user. EasyRequests, StringRequests, and WindowTitles can be sent
either locally or remotely only, however.

1.5 problems

Common Problems and their solutions:

1) I can't get proper results from any of the commands I send to AmiSlate!

- Make sure you set "options results" at the beginning of your script.

2) ARexx scripts don't work properly if I execute them from the second

(or later) concurrent instance of AmiSlate that is running on my system.

- Probably this is because your script isn't parsing the Port Name

correctly from the command line arguments AmiSlate starts it with.

Make sure the following lines are at the beginning of your script:

parse arg CommandPort ActiveString

address (CommandPort)

3) I need to see the output of my Rexx script in order to debug it!

- Use the REXXOUTPUT=<outputfile> startup option.

4) My request for information from the remote user terminated immediately,

before the remote user even responded to the requester!

- Commands that ask the remote user for information, including

RemoteRexxCommand, RemoteStringRequest, and RemoteEasyRequest,

all return immediately. In order to find out the user's response,

you need to follow the Remote command with a "WaitEvent MESSAGE".

When the remote user responds to the requester, his/her response

will be returned as a message string.

5) Anytime I try to put string literals with spaces in them into my

arguments to AmiSlate, I get a warning/error message, and my command

doesn't work.

- This is apparently due to ARexx being lame about argument passing--

when AmiSlate receives the messages, the quote marks have been

stripped and AmiSlate thinks each word is a separate argument.

If anyone knows a way to get around this, please tell me! For now,

here is the workaround:

You have to specify a set of quotation marks separately and concat-

enate them in explicitely. So, instead of:

SetWindowTitle "Hi There, Welcome to AmiSlate!"

you should use:

SetWindowTitle '"' || "Hi There, Welcome to AmiSlate!" || '"'

1.6 displaybeep

DISPLAYBEEP

Arguments

[LOCAL] [REMOTE]

Effect/Description

Causes a DisplayBeep() on the local or remote screen, or both.

Return Values

----

rc is always 1.

Example

Beep LOCAL REMOTE

1.7 breakarexxscripts

BREAKAREXXSCRIPTS

Arguments

None.

Effect/Description

Causes AmiSlate to try to remove all ARexx scripts, by sending
a QUIT command to any waiting script and closing its port.
Warning: Unless you are very careful, the script YOU are running
will be one of the scripts that are "broken"!
What AmiSlate does to break the ARexx scripts is this:
If any script is waiting on a WaitEvent command, it will
be replied with a QUIT message (yes, even if it didn't
specify QUIT as something to wait on!). Then AmiSlate
will close its ARexx port for about three quarters of
a second. Any script that tries to send AmiSlate a
command during that time will abort with "host environment
not found" or somesuch.
If you want to keep your script running after executing this
command, have your script do nothing for about two seconds
after sending the BreakARexxScripts.

Return Values

----

rc is always 1.

1.8 circle

CIRCLE

Arguments

<x> <y> <rx> <ry> [FILL] [XOR]

Effect/Description

Draws a circle with center (x,y) and radii (rx, ry) in the current

Foreground color. If the FILL switch is specified, the circle will

be filled. If XOR is specified, it will draw the circle in

XOR mode instead of using the current foreground color.

Return Values

On success, rc = 1. Otherwise, rc = 0.

Example

Circle 50 60 20 25 FILL

1.9 clear

CLEAR

Arguments

----

none

Effect/Description

Clear the drawing window and Chat Lines.

Return Values

----

rc is set to 1.

Example

Clear

1.10 connect

CONNECT

Arguments

<hostname>

Effect/Description

Attempts to connect to the given host computer. Fails if there is already

a connection running, or if the connect attempt failed.

Return Values

----

rc is 1 on success, 0 on failure.

Example

Connect jfriesne.extern.ucsd.edu

1.11 disconnect

DISCONNECT

Arguments

----

none

Effect/Description

Disconnects from a remote client. Fails if no connection is active

or AmiTCP is not running.

Return Values

----

rc is 1 on success, 0 on failure.

Example

Disconnect

1.12 easyrequest

EASYREQUEST

Arguments

<title> <message> <gadgets>

Effect/Description

Presents the local user with an EasyRequester from which he/she
may make a choice.
<gadgets> should be a string of the form "name1|name2|...|nameN"
where each |-separated substring is the name of one gadget the user
can select.

Return Values

----

rc is set to the option selected by the user, in EasyRequester
style--i.e. The rightmost button in the list returns 0 when
selected, and the rest are numbered 1..(N-1), where N is the
total number of gadgets.

Example

EasyRequest "A_Nice_Title" "Do_you_like_llamas?" "Yes|No|Maybe"
if (rc == 0) then say "You picked Maybe"
if (rc == 1) then say "You picked Yes"
if (rc == 2) then say "You picked No"

1.13 filerequest

FILEREQUEST

Arguments

<title> <dir> <file> [SAVE]

Effect/Description

Brings up an ASL FileRequester from which the user may choose a file.
The default directory and file of the requester may be specified, and
if the SAVE switch is specified, the ASL Requester will operate slightly
differently, disallowing file selection by double-clicking on the file.

Return Values

<file> - the dir/file path chosen by the user.

Example

FileRequest stem filer. Select_a_File "ram:" DefaultFile.txt
say "You picked: " filer.file

1.14 flood

FLOOD

Arguments

<x> <y> [UNSAFE]

Effect/Description

Affects a flood fill in the current foreground color at the given location.

Specifying UNSAFE will cause the flood fill to be made in Not-So-Safe mode.

(see AmiSlate.guide for more info on this)

Return Values

----

rc is 1 if flood succeeded, 0 on failure.

Example

Flood 25 50

1.15 getpixel

GETPIXEL

Arguments

<x> <y>

Effect/Description

----

returns the palette number and red, green, and blue values of the pixel

at the given co-ordinates.

Return Values

----

rc is 1 if the coordinate was on the canvas, otherwise 0.

rc2 is the pen number of the pixel at the given co-ordinate, or -1 if

the coordinate was off the canvas.

Example

GetPixel 50 30

say "The pixel at 50 30 is of color " rc2

1.16 getremotestateattrs

GETREMOTESTATEATTRS

Arguments

----

none

Effect/Description

Returns data about the state of the remote peer on a connection.

If there is no active connection, then all values will be set

to -1.

Return Values

----

mode - The mode index number of the remote amiga's current mode.

fpen - The remote amiga's current foreground pen's palette position

bpen - The remote amiga's current background pen's palette position

fred - The remote amiga's current foreground pen's red value (0..7)

fgreen - The remote amiga's current foreground pen's green value (0..7)

fblue - The remote amiga's current foreground pen's blue value (0..7)

bred - The remote amiga's current background pen's red value (0..7)

bgreen - The remote amiga's current background pen's green value (0..7)

bblue - The remote amiga's current background pen's blue value (0..7)

pointerx - The x-coordinate of the remote pointer, if displayed (else -1)

pointery - The y-coordinate of the remote pointer, if displayed (else -1)

Example

GetRemoteStatAttrs stem rsa.

say "Remote mode = " rsa.mode

say "Remote fpen = " rsa.fpen

...

1.17 getstateattrs

GETSTATEATTRS

Arguments

----

none

Effect/Description

Gets data about the local client's current state.

Return Values

----

mode - The mode index number of the local amiga's current mode.

fpen - The local amiga's current foreground pen's palette position

bpen - The local amiga's current background pen's palette position

fred - The local amiga's current foreground pen's red value (0..7)

fgreen - The local amiga's current foreground pen's green value (0..7)

fblue - The local amiga's current foreground pen's blue value (0..7)

bred - The local amiga's current background pen's red value (0..7)

bgreen - The local amiga's current background pen's green value (0..7)

bblue - The local amiga's current background pen's blue value (0..7)

pendown - 1 if user is currently drawing, else 0

locked - 1 if user is currently locked out of drawing on the screen,

else 0.

Example

GetStateAttrs stem sta.

say "Local mode = " sta.mode

say "Local fpen = " sta.fpen

...

1.18 getversion

GETVERSION

Arguments

----

none

Effect/Description

Returns a string containing AmiSlate's current version number.

Return Values

----

version - a version string.

Example

GetVersion var ThisVersion

say "Current AmiSlate version is: " ThisVersion

1.19 getwindowattrs

GETWINDOWATTRS

Arguments

----

none

Effect/Description

Returns data about the current state of the AmiSlate window.

Note that maxwidth and maxheight are not always the same as the

screen width and screen height. They may be less if the remote

client is using a smaller screen than the local client.

Return Values

----

top - distance of window from top of screen, in pixels

left - distance of window from left of screen, in pixels

width - width of window, in pixels

height - height of window, in pixels

depth - Screen depth/Number of bitplanes used by screen window is on

maxwidth - Maximum possible width of window, in pixels

maxheight - Maximum possible height of window, in pixels

Example

GetWindowAttrs stem wat.

say "Window top = " wat.top

say "Window left = " wat.left

...

1.20 line

LINE

Arguments

<x1> <y1> <x2> <y2> [XOR]

Effect/Description

Draws a line from (x1, y1) to (x2, y2) in the current foreground color.

If the XOR switch is specified, draws the line in XOR mode instead.

Return Values

----

rc is 1 if the line draw was wholly successful, else 0.

A line draw is wholly successful if both endpoints were in the window.

Example

Line 5 5 25 35

1.21 loadiff

LOADIFF

Arguments

<file> [EXPAND] [LOADPALETTE] [PROTECTGUI] [LEFT #] [TOP #]

Effect/Description

Causes AmiSlate to attempt to load the specified file in as an IFF

ILBM picture file. If successful, the file will be displayed on screen,

and transmitted to the remote client.

If the EXPAND switch is specified, AmiSlate will attempt to resize

its window larger to fit the ILBM, if need be.

If LOADPALETTE is specified, AmiSlate will load in the new color map

of the IFF.

If PROTECTGUI is specified, AmiSlate will make sure not to alter

pens 0 through 3, thus keeping the screen looking mostly sane.

If TOP is specified, AmiSlate will load in the picture starting

at the given vertical offset.

If LEFT is specified, AmiSlate will load in the picture starting

at the given horizontal offset.

Return Values

----

rc is 1 on success, 0 on failure.

Example

LoadIFF "work:graphics/lo-res/alien.IFF" EXPAND LOADPALETTE PROTECTGUI TOP 40 LEFT 100

1.22 lock

LOCK

Arguments

[ON] [OFF]

Effect/Description

A Lock ON command will disallow the user from drawing on the drawing
area. A Lock OFF command will re-allow the user to draw.

Return Values

----

rc is always 1.

Example

Lock ON

1.23 lockpalette

LOCKPALETTE

Arguments

[ON] [OFF]

Effect/Description

Lock ON causes AmiSlate to Lock Palettes with the remote client. This
means that it will copy the remote client's palette to the local client's
palette, and thereafter any palette changes made on one client will
be echoed to the other.
Lock OFF releases the two palettes so that they may be modified separately.
A Lock will only succeed if there is a TCP connection active.

Return Values

----

rc is 1 on success, 0 on failure.

Example

LockPalette ON

1.24 pen

PEN

Arguments

<x> <y> [XOR]

Effect/Description

If this is the first Pen command issued since a PenReset command, Pen
will set the beginning co-ordinates of a line. Otherwise, Pen will
draw a line from the beginning co-ordinates given previously and then
draw a line to the co-ordinates (x,y). After the draw, (x,y) becomes
the beginning co-ordinates for the next line draw.
If XOR is specified, it will draw the line segment in XOR mode instead
of the current foreground color.

Return Values

----

rc is 1 if the specified co-ordinates were within the drawing area,
else 0.

Example

Pen 15 25

1.25 penreset

PENRESET

Arguments

----

none

Effect/Description

Causes the Pen tool to reset to its initial state. After executing
a PenReset, the next Pen command will be the start of a new line,
rather than the continuation of an existing line.

Return Values

----

rc is always 1.

Example

PenReset

1.26 playscript

PLAYSCRIPT

Arguments

<file>

Effect/Description

Plays the specified AmiSlate script file back to the user.

Return Values

----

rc is 1 on success, 0 on failure.

Example

PlayScript "dh0:AmiSlate/Scripts/FunnyFace.script"

1.27 point

POINT

Arguments

<x> <y> [XOR]

Effect/Description

Plots a single pixel at the the co-ordinate (x,y) in the current

foreground color. If XOR is specified, it will draw the pixel in

XOR mode instead.

Return Values

----

rc is 1 if the point was within the drawing area, else 0.

Example

Point 20 30

1.28 putrasterpixels

PUTRASTERPIXELS

Arguments

<length> <red> <green> <blue>

<length> PEN <pennumber>

Effect/Description

Paints <length> pixels into the Raster drawing area specified

by a previous call to SetRaster. The pixels will be drawn in

the closest color matching the given R, G, B, values, or if the

PEN keyword is specified, they will be drawn with the given

pen.

Return Values

If no Raster drawing area has been set, rc is 0. Otherwise 1.

Example

PutRasterPixels 15 12 3 0

PutRasterPixels 15 PEN 3

1.29 quit

QUIT

Arguments

[FORCE]

Effect/Description

Causes AmiSlate to quit. Specifying the FORCE switch causes the

user not to be asked for confirmation first. (Currently this has

no effect, as the user is not asked for confirmation anyway)

Note that sending any commands to AmiSlate after this command will

cause your ARexx script to error out.

Return Values

----

rc is always 1.

Example

Quit

1.30 recordscript

RECORDSCRIPT

Arguments

<file> [FORCE] [START] [STOP]

Effect/Description

Causes AmiSlate to begin or end recording to the given file. The

START and STOP keywords control whether recording is to begin or

end, and RecordScript will fail if it is asked to START when a

script is already being recorded or asked to STOP when no script

is being recorded. The FORCE keyword currently has no effect.

Return Values

----

rc is 1 on success, 0 on failure.

Example

RecordScript "ram:TempScript.script" START

1.31 remoterexxcommand

REMOTEREXXCOMMAND

Arguments

<file>

Effect/Description

This command will attempt to execute an ARexx script on the remote client's

Amiga, first getting permission from the user thereof with an EasyRequest.

When the user responds to the requester asking him whether or not he wants

to run the given script, the remote AmiSlate client will send back a one

character message giving a result code. Use "WaitMessage MESSAGE" directly

after executing a RemoteRexxCommand to catch the response.

Possible response values are:

0) Remote user refused to run the script, or another failure occured.

1) Remote user allowed the script to be run.

2) Remote user chose another script to run instead.

Return Values

----

1 on success, 0 on failure.

Example

RemoteRexxCommand "SlateRexx:Chess.Rexx"

WaitEvent MESSAGE stem waitev.

if (waitev.message > 0) then say "Remote script running"

1.32 remotestringrequest

REMOTESTRINGREQUEST

Arguments

<title> <defaultstring> <message>

Effect/Description

This command presents a StringRequester to the remote user. This

command does NOT wait for an answer, however. Use WaitEvent immediately

after this command to be sure to catch the MESSAGE event that will

come back when the remote user responds.

RemoteStringRequest will fail if a connection is not active.

If you wish to have one or more of the parameters be an empty string,

see this note.

Return Values

----

rc is 1 on success, 0 on failure.

Example

RemoteStringRequest RemoteStringTitle Jeremy WhatIsYourName

WaitEvent MESSAGE stem wat.

say "Reply was: " wat.message

1.33 saveiff

SAVEIFF

Arguments

<file>

Effect/Description

Saves the current canvas as an IFF ILBM file to <filename>.

Return Values

----

rc is 1 on success, 0 on failure.

Example

SaveIFF "work:graphics/newfile.IFF"

1.34 sendmessage

SENDMESSAGE

Arguments

<message>

Effect/Description

Sends the given string to the remote client, where if an ARexx script is
waiting for it with a WaitEvent MESSAGE, it will be received by the
remote script.
SendMessage will fail if a connection is not active.

Return Values

----

rc is 1 on success, 0 on failure.

Example

SendMessage HiRemoteClient

1.35 setbcolor

SETBCOLOR

Arguments

<red> <green> <blue> [NOTBACKGROUND]

Effect/Description

Will set the background color to the closest matching palette position
to the given (R,G,B) triplet. The NOTBACKGROUND flag, if specified,
will ensure that the new color is not the previous background color.

This command currently has no useful effect.

Return Values

----

rc is always 1.

rc2 is the pen number of the color chosen.

Example

SetBColor 15 1 1 /* A bright red */

1.36 setbpen

SETBPEN

Arguments

<pen>

Effect/Description

Sets the current background color to the given palette position.

This command currently has no useful effect.

Return Values

----

rc is 1 unless <pen> is an invalid palette position.

Example

SetBPen 1

1.37 setfcolor

SETFCOLOR

Arguments

<red> <green> <blue> [NOTBACKGROUND]

Effect/Description

Will set the foreground color to the closest matching palette position

to the given (R,G,B) triplet. The NOTBACKGROUND flag, if specified,

will ensure that the new color is not the background color.

Return Values

----

rc is always 1.

rc2 is the palette position of the number chosen.

Example

SetFColor 0 9 0 /* Medium green */

1.38 setfpen

SETFPEN

Arguments

<pen>

Effect/Description

Sets the current foreground color to the given palette position.

Return Values

----

rc is 1 as long as <pen> is a valid palette position.

Example

SetFPen 4

1.39 setraster

SETRASTER

Arguments

<top> <left> <width> <height> <pixeloffset>

Effect/Description

Defines a drawing area for raster pixels. After executing this call,
the raster drawing area can be filled from top to bottom, left to right,
with pixels of various colors, by using the PutRasterPixels command.
pixeloffset is the number of pixels into the raster window that you
wish to start drawing pixels at. This number can range from zero to
((width * height)-1), with zero being the top left, and the latter
expression being the lower right of the window.

Return Values

If the raster is entirely on the drawing canvas, rc is 1.
Otherwise it will be zero (although the raster will still work--it
will be clipped at the edges of the drawing area)
Example

SetRaster 10 10 50 55 0

1.40 setremotewindowtitle

SETREMOTEWINDOWTITLE
Arguments

<message>
Effect/Description

Sets the remote client's WindowTitle to "R:<message>". The R: lets
the remote user know that the message came from a remote location.
This command will fail if there is no active connection.
Return Values
----
rc is 1 on success, 0 on failure.
Example

SetRemoteWindowTitle HelloFromTheOtherGuy

1.41 settoolbehavior

SETTOOLBEHAVIOR
Arguments

<not specified>
Effect/Description

This command is not yet implemented. It will eventually be used
to control the behavior and possibly the appearance of the various
buttons in the ToolBar.
Return Values

<not specified>
Example

<haven't got a clue>

1.42 setuserbcolor

SETUSERBCOLOR

Arguments

<red> <green> <blue> [NOTBACKGROUND]

Effect/Description

Will set the user's background color to the closest matching palette position

to the given (R,G,B) triplet. This command will cause a visible change to

the Palette, and will set the color that is drawn with the mouse.

The NOTBACKGROUND flag, if specified, will ensure that the new

color is not the previous background color.

This command is not yet implemented.

Return Values

----

rc is always 1.

Example

SetUserBColor 0 1 2 /* Almost black */

1.43 setuserbpen

SETUSERBPEN

Arguments

<pen>

Effect/Description

Will set the user's background to the palette position <pen>. This command

will cause a visible change to the Palette, and will set the color that is

drawn with the mouse. The NOTBACKGROUND flag, if specified, will ensure

that the new color is not the previous background color.

This command is not currently implemented.

Return Values

----

rc is 1 as long as <pen> is a valid palette position.

Example

SetUserBPen 5

1.44 setuserfcolor

SETUSERFCOLOR

Arguments

<red> <green> <blue> [NOTBACKGROUND]

Effect/Description

Will set the user’s foreground color to the closest matching palette position
to the given (R,G,B) triplet. This command will cause a visible change to
the Palette, and will set the color that is drawn with the mouse.
The NOTBACKGROUND flag, if specified, will ensure that the new
color is not the background color.

Return Values

----

rc is always 1.
rc2 is the pen number of the color chosen.

Example

SetUserFColor 9 9 9 /* Gray */

1.45 setuserfpen

SETUSERFPEN

Arguments

<pen>

Effect/Description

Will set the user’s foreground to the palette position <pen>. This command
will cause a visible change to the Palette, and will set the color that is
drawn with the mouse. The NOTBACKGROUND flag, if specified, will ensure
that the new color is not the background color. <pen> ranges from zero to
(number of colors in palette) - 1.

Return Values

----

rc is 1 as long as <pen> is a valid palette position.

Example

SetUserFPen 0 /* Background color? */

1.46 setusertool

SETUSERTOOL

Arguments

<mode>

Effect/Description

This command will cause the user's current Tool Setting to change to

<mode>. This command causes a visible change in the ToolBar.

Return Values

----

rc is 1 as long as <mode> is a valid mode index number

Example

SetUserTool 0 /* Selects "dot" mode */

1.47 setwindowtitle

SETWINDOWTITLE

Arguments

<message>

Effect/Description

Changes the local window title to <message>

Return Values

----

rc is always 1.

Example

SetWindowTitle HiThereUser

1.48 stringrequest

STRINGREQUEST

Arguments

<title> <defaultstring> <message>

Effect/Description

Presents the user with a StringRequester.

Return Values

----

message - What the user entered, or "(User aborted)" if the user clicked

the close box instead of pressing return.

If you wish to have one or more of the parameters be an empty string,

see this note.

Example

StringRequest ThisIsTheTitle SquidLips WhatsYourFavoriteFood

1.49 stringrequestnote

ARexx (or at least ARexxBox) seems to have a bit of trouble passing

empty strings ("") as parameters. As a work around, StringRequest is

programmed to treat strings starting with the ASCII value 1 as blank

strings.

For example, to display a StringRequest with no default text, you

could do:

StringRequest ThisIsTheTitle d2c(1) WhatsYourFavoriteFood

1.50 typekeys

TYPEKEYS

Arguments

<message>

Effect/Description

Enters <message> into the Chat Line as if the user had typed it in.

Return Values

----

rc is always 1.

Example

TypeKeys TheUserOnlyWishesHeCouldTypeThisFast

1.51 transmitpointer

TRANSMITPOINTER

Arguments

[ON] [OFF]

Effect/Description

Sets or unsets the "Transmit Pointer" option in the Options Menu.

Return Values

----

rc is always 1.

Example

TransmitPointer ON

1.52 transmitgraphics

TRANSMITPOINTER

Arguments

[ON] [OFF]

Effect/Description

If OFF is specified, this command will cause AmiSlate to stop
transmitting any data about graphics drawn on the local screen
to the remote AmiSlate peer. This could be used, for instance,
to draw something on the local screen that you don't want the
remote user to also see. ON will re-enable the transmission.
This command affects the Dot, Pen, Line, Circle, Square, Poly,
and Flood tools. Turning off graphics transmission will also
disallow the remote user's "Resynch Screen" option. (to ensure
the local screen's privacy)

Return Values

----

rc is always 1.

Example

TransmitGraphics OFF

1.53 setpencolor

SETPENCOLOR

Arguments

<pen> <red> <green> <blue>

Effect/Description

Changes the palette entry indicated by <pen> to the RGB color
combination given by the red, green, and blue args. (Each of
the color args should be expressed as a number between 0 and
15).

Return Values

----

rc is 1 if the palette entry number was valid, else 0.

Example

SetPenColor 2 15 15 15

Sets Palette Pen #2 to pure white.

1.54 square

SQUARE

Arguments

<x1> <y1> <x2> <y2> [FILL] [XOR]

Effect/Description

Draws a rectangle with opposing corners (x1,y1) and (x2,y2) in the
current foreground color. If the FILL switch is specified, fills the
rectangle in. If the XOR switch is specified, draws the square in
XOR mode instead of using the current foreground color.
Fails if either set of co-ordinates is outside the drawing window.

Return Values

----

rc is 1 on success, 0 on failure.

Example

Square 50 50 120 100 FILL

1.55 waitevent

WAITEVENT

Arguments

<timeout> [MESSAGE] [MOUSEDOWN] [MOUSEUP] [RESIZE] [QUIT] [CONNECT]

[DISCONNECT] [TOOLSELECT] [COLORSELECT] [KEYPRESS] [MOUSEMOVE]

[REMMOUSEMOVE]

Effect/Description

This is an important function if you wish to make highly interactive

ARexx scripts. This function will wait until one of the specified

events happens, or <timeout> tenths of seconds have elapsed, and then

Return control to the ARexx script with the Message type Code and

other pertinent data fields set appropriately.

Any combination of the events/switches may be specified at once, the

event/switch types are described as follows:

[MESSAGE] - A Message was received from the remote client's ARexx port.

[MOUSEDOWN] - The user depressed the left mouse button in the drawing area.

[MOUSEUP] - The user let go of the left mouse button in the drawing area.

[RESIZE] - The window was resized.

[QUIT] - The user quit AmiSlate.

[CONNECT] - A connection to a remote AmiSlate client was made.

[DISCONNECT] - The connection to the remote AmiSlate client was broken.

[TOOLSELECT] - The user selected a new tool.

[COLORSELECT] - The user selected a new color

[KEYPRESS] - The user pressed a key.

[MOUSEMOVE] - The user moved the mouse.

[REMMOUSEMOVE] - The remote user broadcast a new mouse position.

Return Values

----

type - the unique Message type Code for the event.

x - In TIMEOUT, MOUSEUP and MOUSEDOWN events, the X co-ordinate of

the mouse when the event occured.

- In RESIZE events, the new width of the window.

y - In TIMEOUT, MOUSEUP and MOUSEDOWN events, the Y co-ordinate of

the mouse when the event occured.

- In RESIZE events, the new height of the window.

message - In MESSAGE events, the message string received.

- In CONNECT events, the name of the host connected to.

code1 - In KEYPRESS events, the ASCII code of the pressed key.

If the key pressed was a "Raw key" event, such as an arrow key

or function key, 300 will be added to this value.

- In TOOLSELECT events, the Mode Index Code of the selected tool.

- In COLORSELECT events, the palette entry number of the selected color.

- In MESSAGE events, the source ID of the Message.

- In RESIZE events, 0 if the RESIZE was caused locally, else 1.

code2 - unused.

mousex - Always contains the x coordinate of the mouse, with 0 being the

left edge of the drawing canvas.

mousey - Always contains the y coordinate of the mouse, with 0 being the

top of the drawing canvas.

remx - Always contains the last known x coordinate of the remote mouse,

or -1 if the remote mouse's position is undefined.

remy - Always contains the last known y coordinate of the remote mouse,

or -1 if the remote mouse's position is undefined.

button - This field will be nonzero if the left mouse button is currently

pressed, and zero if the left mouse button is not.

lastkey - Always contains the ASCII code of the last key pressed, or 0 if

no key has been pressed since the last ARexx message was returned.

If the key pressed was a "Raw key" event, such as an arrow key

or function key, 300 will be added to this value.

Example

```
/* Waits 5 seconds, or until one of the three given events happens, before returning */
WaitEvent 50 stem waitev. MESSAGE RESIZE QUIT
if (waitev.type == AMessage.Timeout) then say "WaitEvent timed out"
if (waitev.type == AMessage.Message) then say "Message received: " waitev.message
if (waitev.type == Amessage.Resize) then say "Resize to: " waitev.x waitev.y
if (waitev.type == AMessage.Quit) then say "Bye Bye!"
say "When I received this event, the mouse was at " || waitev.mousex || waitev.mousey
```