CFG

Color Fractal Generator

Scripting Guide

version 2.3

for Color Fractal Generator 2.3

by John A. Schlack

July 1995

Table of Contents

New in Version 2.3

The Apple Events have been updated to support the ten new fractal types. This impacts the following commands: *Get Fractal Data*, *Set Fractal Data*, *Display Fractal Dialog*, *Get Fractal Type*, and *Set Fractal Type*. Note that the fractal type numbers for the basic fractal have changed between version 2.2x and 2.3. This change simplifies the classification similar fractals and is necessary for new fractal types that will be added in the future.

Note that some constants have changed as a result of moving to PowerPC data alignment. Previous scripts will still work if these were written using the defined constants. Otherwise, one will need to update the scripts.

Overview

Color Fractal Generator (CFG) has been designed as a server application. That is, it accepts and implements commands from other applications. These commands are passed using Apple Events.

Scripting applications such as UserLand Frontier and Script Editor (for AppleScript) are designed to communicate with server applications (making these applications clients). CFG versions 2.2 and later are specifically designed to work with both Frontier and AppleScript.

The Apple Events that CFG supports are arranged into three groups (or suites): Required Suite, Frontier Applet Suite, and CFG Suite. This document describes all of the commands in these suites.

The current Apple Event implementation of CFG does not support Apple's Object Model or the Core Suite of Apple Events. However, many commands similar to those in the Core Suite are implemented in the Frontier Applet Suite or the CFG Suite.

Apple Event Definitions

CFG supports three suites: Required Suite, Frontier Applet Suite, and CFG Suite. Functions from the Required Suite should not be called by scripters. They are really meant for the Finder and have been included in this document only for completeness. The Frontier Applet Suite are functions defined by UserLand in their Applet code (CFG uses the verbs defined in Frontier SDK 2.1). Most functions from this suite are supported. Functions not supported include: Print Window, Put Text, Set Font, Set Font Size, and Get Page Rect. Note that some functions may differ in their implementation. Consult the DocServer information that came with this guide for Frontier function calling conventions. The CFG Suite is a newly defined suite which contains other functions which are necessary to exploit the features of Color Fractal Generator.

Most functions direct their operations toward windows. In order to avoid passing the target window as a parameter for all functions that require a target, one sets up a target window with functions such as *Set Target* and *Select Window*. All functions called after setting a target will be directed to the target window. If no target has been assigned but a function requires a target window, the front-most window will be assigned as the target. Note that the target window does not need to be the front-most window. Each function described below has a section entitled "Target" which specifies the type of windows that can be the target of the function. A target of *N/A* means that the function does not require a target.

One may notice that the method of calling a certain function in Frontier does not always conform to the parameter list. The reason is that the calling method has been simplifies in the "glue" table. The glue table is a list of Frontier functions and their equivalent Apple Event calls. Sometimes, extra processing is performed by the glue routines to simplify the interface. If you have any questions about how to call a function from Frontier, consult the DocServer documentation or the glue table (default location in Frontier database is *system.verbs.apps.CFG*).

Many default values for function parameters have already been defined. Functions that use such constants or enumerated types (referred to simply as constants from this point on) will either explicitly show the constants or will refer to another function that contains these definitions. One should always use the constants rather than the values assigned to those constants as the values may change in future versions of CFG. A complete list of Frontier constants is located in the Frontier database (default location: *system.verbs.apps.CFG.const*). The AppleScript constants for a particular verb are located with that verb in the AppleScript dictionary.

Each function has an example showing how a routine is called. Script code appears in plain style. Server responses will be shown after the "»" symbol. Finally, italicized text are simply informational messages to clarify the example. AppleScript examples are all located within a "tell application" block. This directs the commands to CFG. Also note that the function results are always copied into the variables on the line following the AppleScript command (using the variable *result*). This is meant to clearly display the function calling sequence. One will not need to extract the sequence out of a *copy* command.

Required Suite

Open Application - Launches Color Fractal Generator.

This event is normally sent only by the Finder.

Event Class:	kAERequiredSuite
Event ID:	kAEOpenApplication
Target:	N/A

Parameters:
« none »

Reply Parameters:
« none »

Examples:
Frontier

```
required.openApplication( CFG.id )
```

Open Document - Opens fractal files.

This is a low level function. To open fractals, one should use the *Open Window* command from the Frontier Applet Suite. If this function is used, the target will not be modified.

Event Class: kAERequiredSuite
Event ID: kAEOpenDocument
Target: N/A

Parameters:
keyDirectObject

Description:	list of file specification records
Descriptor Type:	'list'
Required/Optional:	required

Reply Parameters:
« none »

Examples:
Frontier

Frontier places a higher level interface on this routine, permitting the user to specify a full path name instead of a list of file specification records

```
required.OpenDocument( "HD:myFractal" )
```

Print Document - Not currently handled by CFG.

This function is handled by CFG by simply returning an error code stating that the Apple Event is not handled.

Event Class:	kAERequiredSuite
Event ID:	kAEPrintDocument
Target:	N/A

Parameters:

keyDirectObject

Description:	path to document to print
Descriptor Type:	'TEXT'
Required/Optional:	required

Reply Parameters:

« none »

<u>Quit Application</u> - Quits Color Fractal Generator.

This is a low level function that forces a quit. No items will be saved. To permit saving or prompts for saves, use *Quit* from the Frontier Applet Suite.

Event Class: kAERequiredSuite
Event ID: kAEQuitApplication
Target: N/A

Parameters:
« none »

Reply Parameters:
« none »

Examples:
AppleScript
```
    tell application "CFG"
        quit
    end tell
```
Frontier
```
    required.quitApplication( CFG.id )
```

Frontier Applet Suite

Alert Dialog - Displays an alert box with the specified message.

This verb displays an alert box with the "Caution" icon. The message appears next to the icon and a single "OK" button appears below the message. The alert waits until the user strikes the OK button.

If CFG dialog display is disabled, this function will not display anything. It simply returns True. One should only call this function if CFG is the front application (i.e. useful for shared menus).

Event Class:	'app1'
Event ID:	'alrt'
Target:	N/A

Parameters:

keyDirectObject

Description:	message - text to display to the user
Descriptor Type:	'TEXT'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	result - always True
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    alert dialog "Warning!  Low memory situation."
end tell
```

Frontier

```
CFG.alertDialog( "Warning!  Low memory situation." )
```

Ask Dialog - Displays an dialog box with a prompt and waits for the user to enter the answer string.

A dialog is displayed with a prompt, an editable text field with an initial value, and two buttons: OK and Cancel.

If CFG dialog display is disabled, this function will not display anything. It simply returns True. One should only call this function if CFG is the front application (i.e. useful for shared menus).

Event Class:	'app1'
Event ID:	'askd'
Target:	N/A

Parameters:

keyDirectObject

Description:	prompt - message to display to the user
Descriptor Type:	'TEXT'
Required/Optional:	required

'dflt'

Description:	answer - default response from user
Descriptor Type:	'TEXT'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	result - true if user chose OK and false if Cancel selected
Descriptor Type:	'bool'

'ansr'

Description:	response - User response, valid if "result" was true
Descriptor Type:	'TEXT'

Examples:

AppleScript

```
    tell application "CFG"
        ask dialog "Who is your favorite person?" default "Alicia"
            answer myAnswer
        prior command should exist on a single line
        copy result to OKhit
    end tell
```

Frontier

```
    local( OKhit, answer = "Alicia" )
    OKhit = CFG.askDialog( "Who is your favorite person?", @myAnswer )
    myAnswer stores the user's response ("Alicia" if the default answer was kept)
```

<u>Close Window</u> - Close the target window, optionally saving data.

After the window is closed, the target is cleared. If no new target is assigned with *Set Target*, the default target window becomes the front-most window.

The *New Parameters* window may not be closed. Attempting to close it always returns false. If CFG dialog display is disabled, this function will not display any prompts. Windows will simply not be saved.

Event Class:	'app1'
Event ID:	'cwin'
Target:	not *New Parameters* window

Parameters:

keyDirectObject

Description:	save action - either save, discard, or prompt		
	<u>AppleScript</u>		
		saving	1
		discarding	2
		prompting	3
	<u>Frontier</u>		
		CFG.const.saveDataYes	1
		CFG.const.saveDataNo	2
		CFG.const.saveDataPrompt	3
Descriptor Type:	'shor'		
Required/Optional:	optional		

Reply Parameters:

keyDirectObject

Description:	result - true if window closed or false on error
Descriptor Type:	'bool'

Examples:

AppleScript

```
	tell application "CFG"
		close window
		copy result to windClosed
	end tell
```

or

```
	tell application "CFG"
		close window prompting
		copy result to windClosed
	end tell
```

Frontier

```
	local( windClosed )
	windClosed = CFG.closeWindow()
```

or

```
	local( windClosed )
	windClosed = CFG.closeWindow( CFG.const.saveDataPrompt )
```

Confirm Dialog - Displays a dialog querying the user to continue or cancel the action.

This verb displays a dialog with the specified question. The user may hit OK to continue the action or Cancel to abort.

If CFG dialog display is disabled, this function will not display anything. It simply returns true. One should only call this function if CFG is the front application (i.e. useful for shared menus).

Event Class:	'app1'
Event ID:	'cnfm'
Target:	N/A

Parameters:

keyDirectObject

Description:	question - text to display to the user
Descriptor Type:	'TEXT'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	result - true if OK selected or false if Cancel selected
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    confirm dialog "Do you really want to erase your hard drive?"
    copy result to OKhit
end tell
```

Frontier

```
local( OKhit )
OKhit = CFG.confirmDialog( "Do you really want to erase your hard drive?" )
```

<u>Count Windows</u>- Count the number of application windows that are currently open.

The window count should always be at least one since the New Parameters window is always open.

Event Class:	'app1'
Event ID:	'twin'
Target:	N/A

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	number of open windows
Descriptor Type:	'shor'

Examples:
AppleScript

```
tell application "CFG"
    count windows
    copy result to numWindows
end tell
```

Frontier

```
local( numWindows )
numWindows = CFG.countWindows()
```

Enable Dialogs - Enables or disables alerts and dialog boxes. This is useful if user interaction is not desired.

Use this command very carefully! CFG does not distinguish between commands directly from the user and commands from scripts. If user interaction is disabled, it will stay that way until re-enabled by this command. If the alerts were disabled, make sure they are re-enabled before exiting the script.

During the time that alerts and dialogs are disabled, the only alerts or dialogs displayed are ones due to errors. If a dialog or alert is suppressed, the program will proceed as if the user chose the default dialog response. Exception: if the default action may require user interaction, then the alternative action is chosen (example: the default for "Save Changes" is "OK", but since the user may need to enter the path, "Don't Save" would be chosen).

Event Class:	'app1'
Event ID:	'enbd'
Target:	N/A

Parameters:

keyDirectObject

Description:	enable - true to enable dialog use and false to suppress it
Descriptor Type:	'bool'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	previous - dialog enable status prior to call to enable dialogs
Descriptor Type:	'bool'

Examples:

AppleScript

```
	tell application "CFG"
		enable dialogs true
		copy result to prevEnable
	end tell
```

Frontier

```
	local( prevEnable )
	prevEnable = CFG.enableDialogs( true )
```

if you don't care about the previous enable value, don't save the reply parameter

Get Error String- Returns the last error or informational message.

This function returns a null string ("") if no errors or messages have occurred since program start or since the last time the error string was cleared.

Event Class:	'app1'
Event ID:	'gers'
Target:	N/A

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	last error string or informational message
Descriptor Type:	'TEXT'

Examples:
AppleScript
```
	tell application "CFG"
		get error string
		copy result to lastMessage
	end tell
```
Frontier
```
	local( lastMessage )
	lastMessage = CFG.errorString()
```

<u>Get File Path</u>- Retrieves the path to the target's save file.

The target window must be a fractal window. The full path is limited to 255 characters. If it is larger that 255 characters, an empty string is returned.

Event Class:	'app1'
Event ID:	'gpth'
Target:	fractal window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	A string with the path to the file or an empty string if no path exists or if the full path cannot fit in 255 characters.
Descriptor Type:	'TEXT'

Examples:
AppleScript
```
    tell application "CFG"
        get file path
        copy result to filePath
    end tell
```
Frontier
```
    local( filePath )
    filePath = CFG.getFilePath()
```

Get Fractal Data - Obtains a copy of the target's internal (fractal) data.

The target window must be a fractal window.

The fractal data is returned in a record. The fields present and their meaning depend on the type of target fractal. The tables below contain a list of field tags, the key word of those tags, data types, and descriptions. The field tag lists the AppleScript enumerated type followed on the next line by the Frontier constant. For additional information, one should consult the *CFG User's Manual* in the *Detailed Fractal Description* section.

Fields for Mandelbrot, Julia, and Dragon

Field Tag	Key Word	Data Type	Data
accuracy CFG.const.fdataAccuracy	'accr'	'long'	iteration limit
Xcenter CFG.const.fdataCenterX	'cenx'	'exte'	coordinate plane horizontal center (constant for Julia)
Ycenter CFG.const.fdataCenterY	'ceny'	'exte'	coordinate plane vertical center (constant for Julia)
Xfixed CFG.const.fdataFixedX	'fixx'	'exte'	real constant used in fractal calculation (variable for Julia)
Yfixed CFG.const.fdataFixedY	'fixy'	'exte'	imaginary constant used in fractal calculation (variable for Julia)
freedom CFG.const.fdataFreedom	'free'	'long'	iteration escape value
random CFG.const.fdataRandom	'rand'	'long'	bit field of values to generate randomly
Xscale CFG.const.fdataScaleX	'sclx'	'exte'	horizontal viewing distance
Yscale CFG.const.fdataScaleY	'scly'	'exte'	vertical viewing distance

The *Random* field is a bit field which use the constants defined below. If a bit corresponding to one of the constants below is set, then that field will be generated randomly. Any value presently in that field is ignored.

CFG.const.randCenterX	0x80000000
CFG.const.randCenterY	0x40000000
CFG.const.randScaleX	0x20000000
CFG.const.randScaleY	0x10000000
CFG.const.randFixedX	0x08000000
CFG.const.randFixedY	0x04000000

Fields for Polynomial and Transcendental Fractals

Field Tag	Key Word	Data Type	Data
accuracy CFG.const.fdataAccuracy	'accr'	'long'	iteration limit
Xcenter CFG.const.fdataCenterX	'cenx'	'exte'	coordinate plane horizontal center
Ycenter CFG.const.fdataCenterY	'ceny'	'exte'	coordinate plane vertical center
Xfixed CFG.const.fdataFixedX	'fixx'	'exte'	real constant used in fractal calculation
Yfixed CFG.const.fdataFixedY	'fixy'	'exte'	imaginary constant used in fractal calculation
freedom CFG.const.fdataFreedom	'free'	'exte'	iteration escape value
random CFG.const.fdataRandom	'rand'	'long'	bit field of values to generate randomly
XMultiply CFG.const.fdataMultiplyX	'mulx'	'exte'	real multiplier (only used for transcendental functions)
YMultiply CFG.const.fdataMultiplyY	'muly'	'exte'	imaginary multiplier (only used for transcendental functions)
Xscale CFG.const.fdataScaleX	'sclx'	'exte'	horizontal viewing distance
Yscale CFG.const.fdataScaleY	'scly'	'exte'	vertical viewing distance

Note that the freedom value for polynomial and transcendental functions is an extended type, not a long integer.

The *Random* field is a bit field which use the constants defined below. If a bit corresponding to one of the constants below is set, then that field will be generated randomly. Any value presently in that field is ignored. The constant *CFG.const.randFreedom* is only defined for transcendental fractals.

CFG.const.randCenterX	0x80000000
CFG.const.randCenterY	0x40000000
CFG.const.randScaleX	0x20000000
CFG.const.randScaleY	0x10000000
CFG.const.randFixedX	0x08000000
CFG.const.randFixedY	0x04000000
CFG.const.randMultiplyX	0x02000000
CFG.const.randMultiplyY	0x01000000
CFG.const.randFreedom	0x00800000

<u>Fields for Random Walk</u>

Field Tag	Key Word	Data Type	Data
color method CFG.const.fdataColorMethod	'chow'	'long'	bit field of fractal coloring methods
iterations CFG.const.fdataIterations	'iter'	'long'	maximum number of steps before discarding point
max points CFG.const.fdataMaxPoints	'maxp'	'long'	number of points in fractal

The *Color Method* field is a bit field which use the constants defined below. If a bit corresponding to one of the constants below is set, then that method is used as the coloring method. If more than one bit is set, then the desired colors are calculated for each method and the final color is an average of all of the results. The explanation for these coloring methods is located in the *CFG User's Manual* in the *Fractal Parameters* section.

CFG.const.rwColorLastDir	0x80000000
CFG.const.rwColorTouchDir	0x40000000
CFG.const.rwColorAngle	0x20000000
CFG.const.rwColorTime	0x10000000
CFG.const.rwColorCenterDist	0x08000000
CFG.const.rwColorLastDist	0x04000000

Event Class: 'app1'

Event ID: 'gbin'

Target: fractal window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	fractal data - A record containing the target's fractal data. The exact field and data types of those fields depend upon the fractal type of the target. See above for a complete description of the fields.
Descriptor Type:	'reco'

Examples:

AppleScript

```
tell application "CFG"
    get fractal data
    copy result to fracData
end tell
» {«property accr»:100, «property free»:2, «property cenx»:0.0, ... }
```

Frontier

```
local( fracData )
fracData = CFG.getFractalData()
» {'accr':100, 'free':2, 'cenx':-0.24136686, ... }
```

<u>Get PICT</u>- Get the PICT data for the current selection.

The target window must be a fractal window with a selection. The unregistered version of CFG will not supply any data.

Event Class: 'app1'
Event ID: 'gpic'
Target: fractal window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	The PICT data for the target's selection.
Descriptor Type:	'PICT'

Examples:
AppleScript
```
tell application "CFG"
    get pict
    copy result to pictData
end tell
```
Frontier
```
local( pictData )
pictData = CFG.getPICT()
```

Get Target- Returns the name of the current target window.

If no target window was ever assigned, the front-most window becomes the target and its name is returned.

Event Class:	'app1'
Event ID:	'gtrg'
Target:	see above

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	Name of the current target window.
Descriptor Type:	'TEXT'

Examples:
AppleScript

```
tell application "CFG"
    get target
    copy result to targName
end tell
```

Frontier

```
local( targName )
targName = CFG.getTarget()
```

<u>Get Text</u>- Get the text of the current selection.

The target window must be the Help Window with a region of selected text.

Event Class:	'app1'
Event ID:	'gtex'
Target:	*Help* window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	A copy of the currently selected text.
Descriptor Type:	'TEXT'

Examples:

AppleScript

```
tell application "CFG"
    get text
    copy result to theText
end tell
```

Frontier

```
local( theText )
theText = CFG.getText()
```

<u>Get Window Position</u>- Get the target window's position.

The window position is returned as a rectangle in global coordinates representing the content region of the window. The rectangle includes spaces used by scroll bars (if present).

Event Class:	'app1'
Event ID:	'gwps'
Target:	any window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	Window's position in global coordinates.
Descriptor Type:	'qdrt'

Examples:
AppleScript
```
    tell application "CFG"
        get window position
        copy result to pos
    end tell
```
Frontier
```
    local( pos )
    pos = CFG.getWindowPos()
```

Get Window Rectangle- Returns the target's view rectangle relative to the entire data region.

For non-fractal windows, this command simply returns the window boundaries in global coordinates (same as *Get Window Position* function).

For fractals, the position of the view rectangle (window's content region seen on screen) relative to the entire data region (fractal size) is returned. Thus, the content region (in global coordinates) minus the amount that the window has been scrolled is returned.

Event Class:	'app1'
Event ID:	'gwrc'
Target:	any window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	Window's position in global coordinates shifted by the scroll amount
Descriptor Type:	'qdrt'

Examples:

AppleScript

```
tell application "CFG"
    get window rectangle
    copy result to pos
end tell
```

Frontier

```
local( pos )
pos = CFG.getWindowRect()
```

Has Selection- Determines whether the target has a selected area.

The target window cannot be the *New Parameters* window since this window type will never have a selection.

Event Class: 'app1'
Event ID: 'hsel'
Target: not *New Parameters* window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	True if the target has a selection, false if not.
Descriptor Type:	'bool'

Examples:
AppleScript

```
    tell application "CFG"
        has selection
        copy result to hasSelect
    end tell
```

Frontier

```
    local( hasSelect )
    hasSelect = CFG.haveSelection()
```

<u>Made Changes</u>- Determines whether any changes have been made to the target since the last time it was saved.

The target window must be a fractal window. If the target was never saved (such as a fractal just created with *New* from the *File* menu), true will always be returned.

Event Class:	'app1'
Event ID:	'chgs'
Target:	fractal window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	True if changes were made, or false if not "dirty".
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    made changes
    copy result to changed
end tell
```

Frontier

```
local( changed )
changed = CFG.madeChanges()
```

Move Window - Moves and resizes a target window.

The target can be any window type. Fractal windows are moved and resize, while other windows types are simply moved (bottom and right rectangle fields are ignored).

If the desired location forces the window off the screen, the operation will fail and the window remains at its present location.

Event Class: 'app1'
Event ID: 'mwin'
Target: any window

Parameters:

keyDirectObject

Description:	Rectangle whose top and left fields represent the new window location while the bottom-top and right-left define the new window size (for fractal windows). Thus, the rectangle passed to this function represents the new content region of the window in global coordinates.
Descriptor Type:	'qdrt'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True if window moved (and resized), false on operation failure (window position and size unchanged).
Descriptor Type:	'bool'

Examples:

AppleScript

```
	tell application "CFG"
		copy {50, 50, 250, 400} to newWindRect
		move window newWindRect
		copy result to moved
	end tell
```

Frontier

```
	local( newWindRect, moved )
	newWindRect = rectangle.set( 50, 50, 250, 400 )
	moved = CFG.moveWindow( newWindRect )
	» true
```

The window moved to top-left of 50,50 and it is 350 pixels across and 200 pixels high.

New Window - Creates a new fractal window using the current parameters in the *New Parameters* window.

A window name may be supplied to CFG. If no name is specified or an empty string is specified, CFG generates its own name for the window. The fractal information for the new window is taken from the parameters specified in the *New Parameters* window.

The new window is **NOT** made the target window. It is, however, made the front-most window.

Event Class: 'app1'
Event ID: 'nwin'
Target: N/A

Parameters:
keyDirectObject

Description:	name for the new window
Descriptor Type:	'TEXT'
Required/Optional:	optional

Reply Parameters:
keyDirectObject

Description:	The name of the window. An empty string is returned if the new fractal could not be created.
Descriptor Type:	'TEXT'

Examples:

AppleScript

```
	tell application "CFG"
		new window
		copy result to windName
	end tell
```

or

```
	tell application "CFG"
		new window "myWindowName"
		copy result to windName
	end tell
```

Frontier

```
	local( windName )
	windName = CFG.newWindow()
```

or

```
	local( windName )
	windName = CFG.newWindow( "myWindowName" )
```

Nth Window - Selects a Nth window from the top.

The specified window is made the front-most window and also the target.

The top window is considered window 1, the window below that is 2, etc. If the specified window does not exist (windowN is greater than the total number of windows), an empty string is returned.

Event Class:	'app1'
Event ID:	'nthw'
Target:	see above

Parameters:

keyDirectObject

Description:	Nth window
Descriptor Type:	'shor'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	The name of selected window. An empty string is returned if the specified window does not exist.
Descriptor Type:	'TEXT'

Examples:

AppleScript

```
tell application "CFG"
    nth window
    copy result to windName
end tell
```

Frontier

```
local( windName )
windName = CFG.nthWindow( 2 )
```

Open Window - Opens a fractal file using the specified path name.

The newly opened window is **NOT** made the target window. It is, however, made the front-most window.

One can obtain the window name by getting the file name from the full path name.

Event Class:	'app1'
Event ID:	'owin'
Target:	N/A

Parameters:

keyDirectObject

Description:	path - Full path name to the fractal to open.
Descriptor Type:	'TEXT'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True if the fractal opened successfully, false on error.
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
        open window "HD:mySavedFractal"
        copy result to openResult
end tell
```

Frontier

```
local( openResult )
openResult = CFG.openWindow( "HD:mySavedFractal" )
```

Performance Test - This verb subtracts one rectangle from another and returns the result.

This verb supports the performance benchmark test written in Frontier.

Event Class:	'app1'
Event ID:	'perf'
Target:	N/A

Parameters:

'prm1'

Description:	first rectangle
Descriptor Type:	'qdrt'
Required/Optional:	required

'prm2'

Description:	second rectangle
Descriptor Type:	'qdrt'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	Result of the subtraction of the second rectangle from the first.
Descriptor Type:	'qdrt'

Examples:

AppleScript

```
tell application "CFG"
    copy { 50, 100, 200, 400 } to r1
    copy {20, 40, 400, 300 } to r2
    performance test first rectangle r1 second rectangle r2
    copy result to rResult
end tell
```

Frontier

```
local( r1, r2, rResult )
r1 = rectangle.set( 50, 100, 200, 400 )
r2 = rectangle.set( 20, 40, 400, 300 )
rResult = CFG.performanceTest( r1, r2 )
```

<u>Put PICT</u> - Inserts a picture into the target's selection.

The graphic passed to this function must be of type 'PICT' and the target window must be a fractal.

The picture will be "pasted" inside the selection and stretched to fit the selection. If there is no selection, the picture is scaled to fit the window.

This command cannot be used when the target window is building a fractal.

Event Class:	'app1'
Event ID:	'ppic'
Target:	fractal window

Parameters:

keyDirectObject

Description:	PICT data to insert into the fractal image.
Descriptor Type:	'PICT'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True if the operation was successful, or false on error.
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    perform some operation that places PICT info into thePICT
    put PICT thePICT
end tell
```

Frontier

```
local( thePICT )
perform some operation that places PICT info into thePICT
CFG.putPICT( thePICT )
```

Quit - Quit CFG, optionally saving data.

If CFG dialog display is disabled, this function will not display any prompts. Fractal windows will simply not be saved.

Event Class: 'app1'
Event ID: 'quit'
Target: N/A

Parameters:

keyDirectObject

Description:	save action - either save, discard, or prompt (see the *Close Window* command for a list of the constants)
Descriptor Type:	'shor'
Required/Optional:	optional

Reply Parameters:

keyDirectObject

Description:	true
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
	quit CFG
	the CFG is part of the AppleScript name to distinguish it from the required "quit" command
end tell
or
tell application "CFG"
	quit CFG prompting
end tell
```

Frontier

```
CFG.quit()
or
CFG.quit( CFG.const.saveDataPrompt )
```

Revert Fractal - Resets the target its state when it was last saved.

The target window must be a fractal window. This command cannot be used when the target window is building a fractal.

If the target was never saved (and no file is associated with the target), the target cannot be reverted.

Event Class:	'app1'
Event ID:	'rwin'
Target:	fractal window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	True if the target was reverted or false on error.
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    revert fractal
    copy result to reverted
end tell
```

Frontier

```
local( reverted )
reverted = CFG.revertFractal()
```

Save Fractal - Saves the contents of the target window to disk.

The target window must be a fractal window.

If the path is specified, *Save Fractal* functions as the *Save As* command from the *File* menu. If no path is specified, this verb functions as the *Save* command from the *File* menu. If no path is specified and no file is associated with the fractal, false is returned (and an error is flagged).

Event Class:	'app1'
Event ID:	'swin'
Target:	fractal window

Parameters:

keyDirectObject

Description:	path - Full path name with the folder location to save the file and the file's name.
Descriptor Type:	'TEXT'
Required/Optional:	optional

Reply Parameters:

keyDirectObject

Description:	True if the fractal data was saved or false if it could not be saved.
Descriptor Type:	'bool'

Examples:

AppleScript

```
    tell application "CFG"
        save fractal
        copy result to saved
    end tell
```

or

```
    tell application "CFG"
        save fractal "HD:myFractal"
        copy result to saved
    end tell
```

Frontier

```
    local( saved )
    saved = CFG.saveAs()
```

or

```
    local( saved )
    saved = CFG.saveAs( "HD:myFractal" )
```

Scroll Window - Scrolls the target window in the direction and amount specified.

The *New Parameters* window cannot be scrolled at all and the *Help* window may only be scrolled vertically. If attempts are made to scroll in illegal directions, false will be returned.

If false is returned, it means that the scrolling has reached its limit in the specified direction.

Event Class: 'app1'
Event ID: 'lwin'
Target: not *New Parameters* window

Parameters:

'dire'

Description: Direction of scrolling. Frontier users can also use *up*, *down*, *left*, *right*, *pageUp*, *pageDown*, *pageLeft*, and *pageRight*.

AppleScript

up	1
down	2
left	3
right	4
page up	9
page down	10
page left	11
page right	12

Frontier

CFG.const.dirUp	1
CFG.const.dirDown	2
CFG.const.dirLeft	3
CFG.const.dirRight	4
CFG.const.dirPageUp	9
CFG.const.dirPageDown	10
CFG.const.dirPageLeft	11
CFG.const.dirPageRight	12

Descriptor Type: 'shor'

Required/Optional: required

'coun'

Description:	count - number of times to apply the scrolling (number of "clicks" in the page or arrow regions)
Descriptor Type:	'shor'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True on a successful scroll and false if scroll could not be accomplished.
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    scroll window direction page up count 1
    copy result to didScroll
end tell
```

Frontier

```
local( didScroll )
didScroll = CFG.scrollWindow( CFG.const.dirPageUp, 1 )
```

or

```
local( didScroll )
didScroll = CFG.scrollWindow( pageUp, 1 )
```

Select All - Selects the entire contents of the target.

The target window cannot be the *New Parameters* window as nothing can be selected in this type of window. If the target is a fractal window, the entire fractal picture is selected. For *Help* windows, all text for the current help section is selected.

This command cannot be used when the target window is building a fractal.

Event Class:	'app1'
Event ID:	'sela'
Target:	not *New Parameters* window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	True if successful, false if not selection made.
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
        select all
end tell
```

Frontier

```
CFG.selectAll()
```

Select Window - Makes the specified window the front-most window.

The selected window becomes the target window.

Event Class:	'app1'
Event ID:	'xwin'
Target:	see above

Parameters:

keyDirectObject

Description:	name of window to select
Descriptor Type:	'TEXT'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True if the named window was found or false if not located.
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
        select window "My Pretty Fractal"
end tell
```

Frontier

```
CFG.selectWindow( "My Pretty Fractal" )
```

Set Fractal Data - Inserts the fractal data into the target.

The target window must be a fractal window.

The field definitions are located with the *Get Fractal Data* command. Fields that the user does not wish to alter can be omitted. Data verification will be performed before the data is applied to the target. To view the results of a data change, one must *Redraw* the fractal.

This command cannot be used when the target window is building a fractal.

Event Class:	'app1'
Event ID:	'pbin'
Target:	fractal window

Parameters:

keyDirectObject

Description:	Record with new fractal data. For safety, one should only supply fields that one wishes to modify. See the *Get Fractal Data* command for a description of this record format.
Descriptor Type:	'reco'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True if data applied and false if it was rejected.
Descriptor Type:	'bool'

Examples:

AppleScript

```
	tell application "CFG"
		copy { accuracy:50, Xscale:0.01 } to fractalData
		set fractal data fractalData
		copy result to fracModified
	end tell
```

Frontier

```
	local( fractalData, fracModified )
	fractalData = { CFG.const.fdataAccuracy:50, CFG.const.fdataScaleX:0.01 }
	fracModified = CFG.setFractalData( fractalData )
```

Set Target - Makes the specified window the target window.

If an empty string is passed as the desired target's title, the front-most window is set as the target and its name is returned. If the specified target was not found, an empty string is returned. In this case, the target of subsequent operations will default to the front-most window.

Note that the specified window is not selected (i.e. it is not moved to the front). To perform this action, call *Select Window*.

Event Class:	'app1'
Event ID:	'strg'
Target:	see above

Parameters:

keyDirectObject

Description:	This is the name of the window to make the target. Supply an empty string to select the front-most window.
Descriptor Type:	'TEXT'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	Name of the new target window. An empty string is returned if the desired window was not located.
Descriptor Type:	'TEXT'

Examples:

AppleScript

```
	tell application "CFG"
		set target "myFractalWindow"
		copy result to newTarg
	end tell
```

or

```
	tell application "CFG"
		set target
		copy result to newTarg
	end tell
```

Frontier

```
	local( newTarg )
	newTarg  = CFG.setTarget( "myFractalWindow" )
```

or

```
	local( newTarg )
	newTarg  = CFG.setTarget("")
```

Zoom Window - Enlarges / Reduces the size of the target window.

The target window cannot be the *New Parameters* window. This window has a fixed size.

Event Class: 'app1'
Event ID: 'zwin'
Target: not *New Parameters* window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	True if successful, false if zoom failed
Descriptor Type:	'bool'

Examples:
AppleScript
```
tell application "CFG"
    zoom window
end tell
```
Frontier
```
CFG.zoomWindow()
```

CFG Suite

Cancel Build - Stops the fractal generation process for the target.

The target window must be a fractal window. This function will work properly whether or not the fractal is being generated (no error returned if fractal is not generating).

Event Class:	'fraG'
Event ID:	'abrt'
Target:	fractal window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	result - true on success, false on error
Descriptor Type:	'bool'

Examples:
AppleScript

```
tell application "CFG"
    cancel build
    copy result to cancelRslt
end tell
```

Frontier

```
local( cancelRslt )
cancelRslt = CFG.cancelBuild()
```

CFG Version - Determines the version number of Color Fractal Generator.

The first character returned indicates whether the program is registered ('R') or not registered ('U'). The last three characters indicate the major version number, revision number, and maintenance release number, respectively. A returned value of 'R230' indicates the CFG program is version 2.30 and is registered.

Event Class: 'fraG'
Event ID: 'vers'
Target: N/A

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	version - CFG version and registration information (see above)
Descriptor Type:	'type'

'flag'

Description:	additional flags - Currently, returns 1 if PowerMac version or 0 if 68K.
Descriptor Type:	'LONG'

Examples:
AppleScript
```
tell application "CFG 2.3"
    CFG version storing flags in xtraFlags
    copy result to version
end tell
```
Frontier
```
local( version, xtraFlags )
version = CFG.cfgVersion( @xtraFlags )
```

Clear Selection - Clears a fractal window's selection.

The target window must be a fractal window and at least some portion of the image must be selected. This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'cler'
Target:	fractal window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	result - true if anything cleared, false otherwise
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    clear selection
end tell
```

Frontier

```
CFG.clear()
```

<u>Clear Error String</u> - Clears the error string.

Use this function to clear old errors and messages so that one may see if a set of instructions generates any new errors.

Event Class:	'fraG'
Event ID:	'clrE'
Target:	N/A

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	result - always true
Descriptor Type:	'bool'

Examples:
AppleScript

```
tell application "CFG"
    clear error string
end tell
```

Frontier

```
CFG.clearErrorString()
```

Continue Build - Build the target fractal from where it left off.

The target window must be a fractal window. The fractal must be idle (it cannot be generating).

A continue may not be possible if fractal parameters were changed since the last generation process was stopped. In that case, false is returned as the result and the fractal will not be generated.

Event Class:	'fraG'
Event ID:	'cont'
Target:	fractal window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	result - true if fractal starts generating, false on error
Descriptor Type:	'bool'

Examples:

AppleScript

```
    tell application "CFG"
        continue build
        copy result to didContinue
    end tell
```

Frontier

```
    local( didContinue )
    didContinue = CFG.continueBuild()
```

<u>Copy Selection</u> - Copies current selection to the Clipboard.

The information copied can be either text from the *Help* window or fractal image data from a fractal window.

Fractal images cannot be copied for non-registered versions of Color Fractal Generator. This command cannot be used when the target window is building a fractal. Data is only placed in the Clipboard if CFG is the active application.

Event Class: 'fraG'
Event ID: 'copy'
Target: not *New Parameters* window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	result - true if data copied to Clipboard, false on a failed copy
Descriptor Type:	'bool'

Examples:
AppleScript

```
tell application "CFG"
    copy selection
end tell
```

Frontier

```
CFG.copy()
```

<u>Count Fractal Windows</u> - Counts the number of open fractal windows.

The window count includes all open windows except the *New Parameters* window or the *Help* window.

Event Class: 'fraG'
Event ID: 'cFrc'
Target: N/A

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	number of open fractal windows
Descriptor Type:	'shor'

Examples:
AppleScript

```
tell application "CFG"
    count fractal windows
    copy result to fracWindows
end tell
```

Frontier

```
local( fracWindows )
fracWindows = CFG.countFracWindows()
```

Cut Selection - Copies current selection to the Clipboard and then clears the selection.

The target window must be a fractal window. For non-registered versions of CFG, *Cut* operates exactly as *Copy*.

Data is only placed in the Clipboard if CFG is the active application.

Event Class:	'fraG'
Event ID:	'cut '
Target:	fractal window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	result - true if image cut, false if operation fails or on error
Descriptor Type:	'bool'

Examples:
AppleScript
```
tell application "CFG"
    cut selection
end tell
```
Frontier
```
CFG.cut()
```

Cycle Windows - This function moves the top window behind all other windows. The second window is now the top and it is made the target window.

The new top window is now the target window.

Event Class: 'fraG'
Event ID: 'cycl'
Target: see above

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	target - name of the new target window
Descriptor Type:	'TEXT'

Examples:
AppleScript

```
tell application "CFG"
    cycle windows
    copy result to newTarget
end tell
```

Frontier

```
local( newTarget )
newTarget = CFG.cycleWindows()
```

Deselect - Remove any selection in the target.

This verb should not be applied to the *New Parameters* window as it cannot have any selections. If applied to the correct window type, true is returned even if no selection exists. This command cannot be used when the target window is building a fractal.

Event Class: 'fraG'
Event ID: 'unsl'
Target: not *New Parameters* window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	true if no error, false on error
Descriptor Type:	'bool'

Examples:
AppleScript

```
tell application "CFG"
    deselect
end tell
```

Frontier

```
CFG.deselect()
```

Display Fractal Dialog - Displays the dialog box for the target's fractal or specified fractal.

If the target is a fractal, the optional parameter is ignored. The data used in the dialog box is from the target's fractal type and data.

For non-fractal targets, if a fractal type parameter is specified, the global data for that fractal type is edited. Otherwise, the data for the fractal type specified in the New Parameters Window is displayed.

If CFG dialog display is disabled, this function will not display anything. It simply returns true. This command cannot be used when the target window is building a fractal.

Event Class: 'fraG'
Event ID: 'fDlg'
Target: any window

Parameters:
keyDirectObject

Description: Fractal - either fractal name or type to display. Note that for AppleScript, only the types have been defined.

AppleScript

Mandelbrot	101
Julia	201
Dragon	301
Cosine	102
Sine	103
Exponential	104
Hyperbolic Cosine	105
Hyperbolic Sine	106
Tchebychev T5	1015
Tchebychev C6	1026
Legendre 4	1054
Laguerre 4	1064
Hermite 4	1074
Random Walk	901

Frontier

CFG.const.ftypeMandelbrot	101
CFG.const.ftypeJulia	201
CFG.const.ftypeDragon	301
CFG.const.ftypeCos	102
CFG.const.ftypeSin	103
CFG.const.ftypeExp	104
CFG.const.ftypeCosh	105
CFG.const.ftypeSinh	106
CFG.const.ftypeTchebyT5	1015
CFG.const.ftypeTchebyC6	1026
CFG.const.ftypeLege4	1054
CFG.const.ftypeLagu4	1064
CFG.const.ftypeHerm4	1074
CFG.const.ftypeRandomWalk	901
CFG.const.fnameMandelbrot	"Mandelbrot"
CFG.const.fnameJulia	"Julia"
CFG.const.fnameDragon	"Dragon"
CFG.const.fnameCos	"Cosine"
CFG.const.fnameSin	"Sine"
CFG.const.fnameExp	"Exponential"
CFG.const.fnameCosh	"Hyperbolic Cosine"
CFG.const.fnameSinh	"Hyperbolic Sine"
CFG.const.fnameTchebyT5	"Tchebychev T"
CFG.const.fnameTchebyC6	"Tchebychev C"
CFG.const.fnameLege4	"Legendre"
CFG.const.fnameLagu4	"Laguerre"
CFG.const.fnameHerm4	"Hermite"
CFG.const.fnameRandomWalk	"Random Walk"

Descriptor Type:	'TEXT' or 'shor'
Required/Optional:	optional

Reply Parameters:

keyDirectObject

Description:	result - true if OK selected, false on Cancel or error
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    display fractal dialog
    copy result to OKhit
end tell
```

or

```
tell application "CFG"
    display fractal dialog Mandelbrot
    copy result to OKhit
end tell
```

Frontier

```
local( OKhit )
OKhit = CFG.displayFracDialog()
```

or

```
local( OKhit )
OKhit = CFG.displayFracDialog( CFG.const.fnameMandelbrot )
```

<u>Display Prefs Dialog</u> - Displays the preferences dialog box with the target's preferences.

If the target is not a fractal window, the global preferences will be displayed in the dialog box. These are used for creating new fractals.

If CFG dialog display is disabled, this function will not display anything. It simply returns true. This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'pDlg'
Target:	any window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	true if user chooses OK and false on Cancel
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    display prefs dialog
    copy result to OKhit
end tell
```

Frontier

```
local( OKhit )
OKhit = CFG.displayPrefsDialog()
```

<u>Exp</u> - Calculate exp(x) (or e^x)

This event calculates the exponential of a number.

Event Class:	'fraG'
Event ID:	'exp '
Target:	N/A

Parameters:

keyDirectObject

Description:	'x' paramter of expression exp(x)
Descriptor Type:	'exte'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	Result of the exponential.
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    exp 2.5
    copy result to theExp
end tell
```

Generating - Determine whether the target fractal is being generated.

The target window must be a fractal window.

Event Class:	'fraG'
Event ID:	'bld?'
Target:	fractal window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	true if fractal is being built, false otherwise
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    generating
    copy result to building
end tell
```

Frontier

```
local( building )
building = CFG.generating()
```

<u>Get Depth</u> - Get the target's color depth (bits per pixel).

If the target is not a fractal, the default depth for a new fractal is returned. This information is displayed in the *New Parameters* window.

A response of -1 means to use the color depth of the deepest monitor attached to the system. Thus, if a computer has two monitors, one set to 4 bits / pixel and the other to 16, a *Get Depth* of -1 means to use the 16 bit depth.

The depth is actually a field of the preferences. However, since it is unwieldy to handle the preferences structure just to modify the fractal color depth, this function and *Set Depth* were created.

Event Class:	'fraG'
Event ID:	'gDep'
Target:	any window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	depth - either the number of bits per pixel or use default value. See *Set Depth* command for a list of the constants.
Descriptor Type:	'shor'

Examples:
AppleScript

```
tell application "CFG"
    get depth
    copy result to depth
end tell
```

Frontier

```
local( depth )
depth = CFG.getDepth()
```

<u>Get Fractal Size</u> - Obtains the image size of the target's offscreen fractal data.

If the target is not a fractal, the *New Parameters* window's fractal size is returned. This represents the fractal size for newly created fractals.

Event Class:	'fraG'
Event ID:	'gSiz'
Target:	any window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	size - a point representing the horizontal and vertical image size in pixels
Descriptor Type:	'QDpt'

Examples:

AppleScript

```
tell application "CFG"
    get fractal size
    copy result to fSize
end tell
```

Frontier

```
local( fSize )
fSize = CFG.getFractalSize()
```

<u>Get Fractal Type</u> - Get the fractal type of the target and the fractal name.

The target can be any type of window. For non-fractal windows, the fractal type represents the type displayed in the *New Parameters* window.

Event Class: 'fraG'
Event ID: 'gFrc'
Target: any window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	fractal type - an integer which specifies the fractal type of the target. See *Display Fractal Dialog* for a list of valid responses (Frontier constants start with "CFG.const.ftype").
Descriptor Type:	'shor'

'name'

Description:	fractal name - a string containing the name of the target's fractal. A list of valid names is located in the *Display Fractal Dialog* command.
Descriptor Type:	'TEXT'

Examples:
AppleScript

```
tell application "CFG"
    get fractal type
    copy result to fractalType
end tell
```

or

```
tell application "CFG"
    get fractal type storing name in theName
    copy result to fractalType
end tell
```

Frontier

```
	local( fractalType )
	fractalType = CFG.getFractalType()
	or
	local( fractalType, theName)
	fractalType  = CFG.getFractalType( @theName)
```

<u>Get Limit Extreme</u> - Get current setting for limiting outlying values of randomly generated fractal variables.

The target may be any type of window. If the target is the *Help* window or *New Parameters* window, the setting will be the global setting which will be applied to new fractals.

The limit extreme value only applies to the current target. Other windows may be set differently.

Event Class: 'fraG'
Event ID: 'gLim'
Target: any window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	true if limiting is enabled, false if disabled
Descriptor Type:	'bool'

Examples:
AppleScript

```
tell application "CFG"
	get limit extreme
	copy result to limitExt
end tell
```

Frontier

```
local( limitExt )
limitExt = CFG.getLimitExtreme()
```

<u>Get Preferences</u> - Get a copy of the target's preferences.

Preferences are either targeted at a specific window or at the entire program. For window specific preferneces, if the target is not a fractal window, the returned data is from the global preferences, which are applied to new fractals. Only the depth menu and window flags are window specific. All other preferences apply to the entire program (i.e. all windows). The table below contains a list of field tags, the key word of those tags, data types, and descriptions. For additional information, one should consult the *CFG User's Manual* in the *Preferences* section.

Field Tag	Key Word	Data Type	Data
depth menu CFG.const.prefsDepthMenu	'dmnu'	'long'	depth menu setting (see below)
window flags CFG.const.prefsFlags	'flag'	'long'	window flags (see below)
precision CFG.const.prefsPrecision	'prec'	'long'	floating point display precision (number of significant digits)
program flags CFG.const.prefsPgmFlags	'pflg'	'long'	program flags (see below)
image creator CFG.const.prefsCreatImage	'CrtI'	'type'	application signature for default graphics file creator
text creator CFG.const.prefsCreatText	'CrtT'	'type'	application signature for default text file creator

The *Window Flags* field is a bit field which use the constants defined below. If a bit corresponding to one of the constants below is set, then that preferences option is selected. An explanation for these options is located in the *CFG User's Manual* in the *Preferences* section. The following constants are not defined for AppleScript.

CFG.const.pwflagSaveBitmap	0x00008000
CFG.const.pwflagSaveSelection	0x00004000
CFG.const.pwflagSavePalette	0x00002000
CFG.const.pwflagFasterColor	0x00001000

The *Depth Menu* field is a short integer with a value representing the currently selected item in that menu. Below are a list of valid values. Note that these values correspond to menu selections. They are not the actual bits per pixel as is used by the *Get*

Depth and *Set Depth* commands. The following constants are not defined for AppleScript.

CFG.const.depthMenuDeep	1
CFG.const.depthMenu4Bit	3
CFG.const.depthMenu8Bit	4
CFG.const.depthMenu16Bit	5
CFG.const.depthMenu32Bit	6

The valid values for the floating point precision range from 3 to 18 digits.

The *Program Flags* field is a bit field which use the constants defined below. If a bit corresponding to one of the constants below is set, then that preferences option is selected. An explanation for these options is located in the *CFG User's Manual* in the *Preferences* section. The following constants are not defined for AppleScript.

CFG.const.ppflagDrawNow	0x00008000
CFG.const.ppflagDispScale	0x00004000
CFG.const.ppflagLineByLine	0x00002000
CFG.const.ppflagDispWalk	0x00001000
CFG.const.ppflagLimitWindow	0x00000800

Event Class: 'fraG'
Event ID: 'gPrf'
Target: any window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	preferences data - A record containing the target's preferences data. See above for a complete description of the fields.
Descriptor Type:	'reco'

Examples:

AppleScript

```
    tell application "CFG"
        get preferences
        copy result to prefData
        » {«property flag»:0, «property prec»:10, «property dmnu»:1,...}
    end tell
```

Frontier

```
    local( prefData )
    prefData = CFG.getPreferences()
    » {'flag':0, 'prec':10, 'dmnu':1,'pflg':43008,...}
```

<u>Get Scroll Position</u>- Gets the current and maximum scrolling parameters of the target.

The target cannot be the *New Parameters* window as it does not have a scroll region.

The data returned is a rectangle. The top and left portions of the rectangle correspond to the current vertical and horizontal scroll positions, respectively. The bottom and right represent the maximum vertical and horizontal scroll values, respectively.

A value of -1 indicates no scroll information is available for that parameter. For example: the help window would return a -1 for r.left and r.right since it has no horizontal scroll bar. The minimum scroll setting is 1.

Event Class:	'fraG'
Event ID:	'gScr'
Target:	not *New Parameters* window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	Rectangle whose top and left fields represent the current scroll values and bottom right fields represent the maximum scroll values.
Descriptor Type:	'qdrt'

Examples:

AppleScript

```
tell application "CFG"
        get scroll position
        copy result to scrollPos
end tell
```

Frontier

```
local( scrollPos )
scrollPos = CFG.getScrollPosition()
» { 2, 3, 10, 20 }
```

In this example, the current scroll positions (h,v) are (2,3) while the maximum are (10,20).

<u>Get Selection</u> - Retrieves the target's current selection.

This verb should not be applied to the *New Parameters* window as it cannot have any selections. For fractal windows, the rectangle is the selection in local coordinates. It is absolute with respect to the fractal's offscreen data (i.e. current scroll position does not affect the selection values). Only the left and right parameters of the rectangle are valid for the *Help* window. These specify the beginning and ending character positions of the selection. If nothing is selected, an empty rectangle is returned.

Event Class: 'fraG'
Event ID: 'gSel'
Target: not *New Parameters* window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	Rectangle with current selection. See above for a detailed description.
Descriptor Type:	'qdrt'

Examples:
AppleScript
```
    tell application "CFG"
        get selection
        copy result to selRect
    end tell
```
Frontier
```
    local( selRect )
    selRect = CFG.getSelection()
    » { 0, 0, 640, 480 }
```

<u>Get Speed</u>- Obtain the fractal generation speed.

Event Class: 'fraG'
Event ID: 'gSpd'
Target: N/A

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	A value indicating the fractal generation speed. See the *Set Speed* command for a list of speed constants.
Descriptor Type:	'shor'

Examples:
AppleScript

```
tell application "CFG"
    get speed
    copy result to genSpeed
end tell
```

Frontier

```
local( genSpeed )
genSpeed = CFG.getSpeed()
```

Gray Region Rectangle- Returns the total desktop area minus menu bar region and a 4 pixel frame surrounding the desktop.

This function is useful when moving and resizing windows. The title region of a window must remain in this rectangle.

Event Class: 'fraG'
Event ID: 'gRgR'
Target: N/A

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	Rectangle with coordinates of the total desktop area (minus a small border).
Descriptor Type:	'qdrt'

Examples:
AppleScript

```
tell application "CFG"
    gray region
    copy result to grayRect
end tell
```

Frontier

```
local( grayRect )
grayRect = CFG.grayRgnRect()
```

Invert Color- Inverts the selected region's colors.

The target must be a fractal window with a selection. Only those pixels in the selection are inverted. This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'ivrt'
Target:	fractal window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	True if selection is inverted, false on error.
Descriptor Type:	'bool'

Examples:
AppleScript

```
tell application "CFG"
    invert color
    copy result to inverted
end tell
```

Frontier

```
local( inverted )
inverted = CFG.invertColor()
```

List Fractal Types - Returns a list of fractals which are supported by CFG.

Event Class:	'fraG'
Event ID:	'lFrc'
Target:	N/A

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	fractal list - A list of fractal names. Each item in the list has a data type of 'TEXT'.
Descriptor Type:	'list'

Examples:

AppleScript

```
    tell application "CFG"
        list fractal types
        copy result to fList
        » {"Mandelbrot", "Julia", "Dragon", "Random Walk"}
    end tell
```

Frontier

```
    local( fList )
    fList = CFG.listFractalTypes()
    » {"Mandelbrot", "Julia", "Dragon",  ... }
```

<u>**List Help Items**</u> - Returns a list of topics available from the *Help* window.

Event Class:	'fraG'
Event ID:	'lHlp'
Target:	N/A

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	help topic list - A list of topics from the *Help* window. Each item in the list has a data type of 'TEXT'.
Descriptor Type:	'list'

Examples:

AppleScript

```
    tell application "CFG"
        list help items
        copy result to helpTopics
        » {"Help System", "Creating Fractals", "Applying Color", ... }
    end tell
```

Frontier

```
    local( helpTopics )
    helpTopics = CFG.listHelpItems()
    » {"Help System", "Creating Fractals", "Applying Color", ... }
```

List Palettes - Returns the *Palette* menu's list of palettes for the target.

If the target window is not a fractal, the returned list contains the global palettes available to a new fractal. The list will always contain the three standard palettes. If a custom palette is loaded, its name will also be in the list.

Event Class:	'fraG'
Event ID:	'gpLs'
Target:	any window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	palette list - A list of palette names. Each item in the list has a data type of 'TEXT'. Constant definitions for the three standard palettes are located with the *Set Palette* command. These constants are only valid for UserLand Frontier scripts.
Descriptor Type:	'list'

Examples:

AppleScript

```
	tell application "CFG"
		list palettes
		copy result to palList
		» {"Grayscale Palette", "Rainbow Palette", "System Palette", "myCustomPalette"}
	end tell
```

Frontier

```
	local( palList )
	palList = CFG.listPalettes()
	» {"Grayscale Palette", "Rainbow Palette", "System Palette", "myCustomPalette"}
```

Load Palette - Loads a specified palette file as the custom palette and makes that palette the current palette.

The loaded file will become the target's custom palette file. This change will be reflected in the Palette menu when the target becomes the top-most window.

Although the palette is loaded, the colors of the fractal will not be altered for 16 and 32 bit environments until it is regenerated. Also, for 4 and 8 bit depths, if the palette size differs from the target's previous palette size, some or all of the colors may not change until the fractal is rebuilt.

If the target is not a fractal, the loaded palette becomes the default palette for new fractals. This information is shown in the *New Parameters* window.

Load Palette only loads CFG created palettes. It cannot handle 'clut' and 'pltt' resources. This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'ldPl'
Target:	any window

Parameters:

keyDirectObject

Description:	path - Full path name to the palette file to load.
Descriptor Type:	'TEXT'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True if the palette loaded successfully, false on error.
Descriptor Type:	'bool'

Examples:

AppleScript

```
	tell application "CFG"
		load palette "HD:myFractal"
		copy result to loadResult
	end tell
```

Frontier

```
	local( loadResult )
	loadResult = CFG.loadPalette( "HD:myFractal" )
```

<u>Log</u> - Calculate ln(x)

This event calculates the natural logarithm of a number. The number must be greater than 0. Passing a non-positive value as a parameter results in a 0 as the result.

Event Class:	'fraG'
Event ID:	'log '
Target:	N/A

Parameters:

keyDirectObject

Description:	'x' paramter of expression ln(x)
Descriptor Type:	'exte'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	Result of the natural logarithm.
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    log 2.5
    copy result to theLog
end tell
```

<u>Main Screen Size</u>- Returns the size of the main screen in pixels.

This function is useful when adjusting fractal sizes.

Event Class: 'fraG'
Event ID: 'msSz'
Target: N/A

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	Point whose horizontal and vertical fields contain the number of horizontal and vertical pixels on the main screen, respectively.
Descriptor Type:	'QDpt'

Examples:
AppleScript

```
tell application "CFG"
    main screen size
    copy result to size
end tell
```

Frontier

```
local( size )
size = CFG.mainScreenSize()
```

<u>Open Help Window</u>- Opens the *Help* window and brings it to the front.

The *Help* window is made the target window. If the *Help* window is already open, it is simply selected and made the target.

Event Class: 'fraG'
Event ID: 'oHlp'
Target: see above

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	name of the help window
Descriptor Type:	'TEXT'

Examples:
AppleScript

```
tell application "CFG"
        open help window
        copy result to helpName
end tell
```

Frontier

```
local( helpName )
helpName = CFG.openHelpWindow()
```

<u>Paste</u> - Pastes a PICT image from the Clipboard into the fractal window.

The target window must be a fractal window and the image on the Clipboard must be in PICT format.

If the window has a selection, the image is pasted into the selection (scaled to fit the selected rectangle). Otherwise it is scaled to fit the window.

This command cannot be used when the target window is building a fractal. Data can only be copied from the Clipboard if CFG is the active application.

Event Class:	'fraG'
Event ID:	'past'
Target:	fractal window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	result - true if data pasted into fractal window, false on failure
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    paste
end tell
```

Frontier

```
CFG.paste()
```

Redraw - Reset the fractal creation process and start building the target fractal.

The target window must be a fractal window. This command cannot be used when the target window is building a fractal.

Event Class: 'fraG'
Event ID: 'rdrw'
Target: fractal window

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	True if fractal creation has begun, false on error.
Descriptor Type:	'bool'

Examples:
AppleScript

```
tell application "CFG"
    redraw
end tell
```

Frontier

```
CFG.redraw()
```

<u>Save Palette</u> - Saves the target's current palette to a file.

Note that the current palette may not be the target's custom palette.

Event Class: 'fraG'
Event ID: 'svPl'
Target: any window

Parameters:

keyDirectObject

Description:	path - Full path name with the folder location to save the palette and the new file's name.
Descriptor Type:	'TEXT'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True if the palette was saved or false on error.
Descriptor Type:	'bool'

Examples:

AppleScript

```
    tell application "CFG"
        save palette "HD:myPaletteFile"
        copy result to saved
    end tell
```

Frontier

```
    local( saved )
    saved = CFG.savePalette( "HD:myPaletteFile" )
```

Save Picture - Saves the selected region of the fractal to a graphics file.

The target window must be a fractal. A selected area is saved only if part of the image is selected and the "save selection" preferences flag is set.

Currently, this function only supports the PICT format. Future versions of CFG may support more graphics formats.

This function only works on registered versions of CFG.

Event Class: 'fraG'
Event ID: 'sPic'
Target: fractal window

Parameters:

'path'

Description:	path - Full path name with the folder location to save the file and the file's name.
Descriptor Type:	'TEXT'
Required/Optional:	required

'type'

Description:	graphics format	
	AppleScript	
	PICT	'PICT'
	Frontier	
	CFG.const.imagePICT	'PICT'
Descriptor Type:	'shor'	
Required/Optional:	required	

Reply Parameters:

keyDirectObject

Description:	True on success or false on failure.
Descriptor Type:	'bool'

Examples:

AppleScript

```
	tell application "CFG"
		save picture as "HD:myPicture" with format PICT
		copy result to saved
	end tell
```

Frontier

```
	local( saved )
	saved = CFG.savePicture( "HD:myPicture", CFG.const.imagePICT )
```

<u>Select Help Item</u> - Display help information for a given item.

The target must be the *Help* window. A list of valid help items is returned by *List Help Items*.

Event Class: 'fraG'
Event ID: 'sHlp'
Target: *Help* window

Parameters:

keyDirectObject

Description:	topic for which to display help
Descriptor Type:	'TEXT'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True if help topic displayed, false on error.
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    select help item "Registration"
end tell
```

Frontier

```
CFG.selectHelpItem( "Registration" )
```

<u>**Select New Parameters Window**</u> - Bring the *New Parameters* window to the front.

The *New Parameters* window is made the target. As the *New Parameters* window cannot be disposed, it should always be present. It may not always be named "New Parameters", however.

Event Class:	'fraG'
Event ID:	'sPrm'
Target:	see above

Parameters:
« none »

Reply Parameters:
keyDirectObject

Description:	The name of the New Parameters Window or an empty string on an error.
Descriptor Type:	'TEXT'

Examples:
AppleScript

```
    tell application "CFG"
        select parameters window
    end tell
```

Frontier

```
    CFG.selectParamsWindow()
```

<u>Select Rectangle, Absolute</u> - Select a region of the target using coordinates that references the top,left of the target's data.

This verb should not be applied to the *New Parameters* window as it cannot have any selections.

For fractal windows, the selected rectangle will appear at the specified location with respect to the fractal's offscreen data. Thus if the rectangle was { 10, 10, 30, 30 }, the selection would appear 10 pixels down and to the right of the offscreen's top,left corner. The location on screen depends on the current scroll coordinates.

For *Help* windows, only the left and right fields of the rectangle are valid. These specify the character positions to start and end the selection respectively. Relative and absolute have no meaning for the *Help* window. Therefore, this function is the same as *Select Rectangle Relative* for the *Help* window.

This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'srAb'
Target:	not *New Parameters* window

Parameters:

keyDirectObject

Description:	rectangle with selection in absolute coordinates (referencing the top,left corner of the fractal image as 0,0).
Descriptor Type:	'qdrt'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True if rectangle selected, false on error.
Descriptor Type:	'bool'

Examples:

AppleScript

```
    tell application "CFG"
        select absolute rect { 10, 10, 50, 100 }
    end tell
```

Frontier

```
    local( absRect )
    absRect = rectangle.set( 10, 10, 50, 100 )
    CFG.selectRecAbs( absRect )
```

Select Rectangle, Relative - Select a region of the target relative to the current view region.

This verb should not be applied to the *New Parameter* window as it cannot have any selections.

For fractal windows, the selected rectangle will appear relative to the top-left corner of the viewing region. Thus if the rectangle was { 10, 10, 30, 30 }, the selection would appear 10 pixels down and to the right of the window's top left corner, no matter where scroll coordinates are located.

For *Help* windows, only the left and right fields of the rectangle are valid. These specify the character positions to start and end the selection respectively. Relative and absolute have no meaning for the *Help* window. Therefore, this function is the same as *Select Rectangle Absolute* for the *Help* window.

This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'srRl'
Target:	not *New Parameters* window

Parameters:

keyDirectObject

Description:	rectangle with selection in coordinates relative to the top,left corner of the window
Descriptor Type:	'qdrt'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True if rectangle selected, false on error.
Descriptor Type:	'bool'

Examples:

AppleScript

```
	tell application "CFG"
		select relative rect { 10, 10, 50, 100 }
	end tell
```

Frontier

```
	local( absRect )
	absRect = rectangle.set( 10, 10, 50, 100 )
	CFG.selectRecRel( absRect )
```

<u>Set Depth</u> - Sets the target's color depth.

If the target is not a fractal window, the new depth will become the default depth for new fractals. This information will be reflected in the *New Parameters* window.

Always reset the target window after false is returned. It is possible that a memory allocation error will occur when attempting to change the depth of the offscreen buffer. In this case, the window may have been discarded.

The depth is actually a field of the preferences. However, since it is unwieldy to handle the preferences structure just to modify the fractal color depth, this function and the *Get Depth* function were created.

This command cannot be used when the target window is building a fractal.

Event Class: 'fraG'
Event ID: 'sDep'
Target: any window

Parameters:

keyDirectObject

Description: color depth

<u>AppleScript</u>

deepest	-1
4 bit	4
8 bit	8
16 bit	16
32 bit	32

<u>Frontier</u>

CFG.const.depthDeep	-1
CFG.const.depth4Bit	4
CFG.const.depth8Bit	8
CFG.const.depth16Bit	16
CFG.const.depth32Bit	32

Descriptor Type: 'shor'

Required/Optional: required

Reply Parameters:

keyDirectObject

Description:	True if depth altered, false on error (must reset target window on error).
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
	set depth deepest
	copy result to depthChanged
end tell
```

Frontier

```
local( depthChanged )
depthChanged  = CFG.setDepth( CFG.const.depthDeep )
```

Set Fractal Type - Sets the fractal type to use for new fractals.

The target cannot be a fractal window, as once a fractal is assigned to a window, it cannot be changed.

A fractal name takes precedence over the fractal type. Thus, if one supplies a name, the direct parameter (fractal type) is ignored.

Event Class: 'fraG'
Event ID: 'sFrc'
Target: not fractal window

Parameters:

keyDirectObject

Description:	fractal type; See *Display Fractal Dialog* for a list of the constants (Frontier constants start with "CFG.const.ftype").
Descriptor Type:	'shor'
Required/Optional:	required

'name'

Description:	This is the name of the fractal to select. See *Display Fractal Dialog* for a list of Frontier constants that start with "CFG.const.fname". Note that AppleScript does not have a list of constant names.
Descriptor Type:	'TEXT'
Required/Optional:	optional

Reply Parameters:

keyDirectObject

Description:	True if the desired fractal set, false on an error.
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    set fractal type Dragon
end tell
```

or

```
tell application "CFG"
    set fractal type named "Mandelbrot"
end tell
```

Frontier

```
CFG.setFractalType( CFG.const.ftypeDragon )
```

or

```
CFG.setFractalType( CFG.const.fnameMandelbrot )
```

<u>Set Fractal Size, Custom</u> - Sets the fractal size according to the user specified horizontal and vertical coordinates.

If the target window is not a fractal, the global size values are modified, which specify the size of newly created fractals. This information will be reflected in the *New Parameters* window.

This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'sSzC'
Target:	any window

Parameters:

keyDirectObject

Description:	Point that specifies the size of the fractal image in pixels.
Descriptor Type:	'QDpt'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	True if the size was valid or false on error.
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    set size custom { 160, 120 }
end tell
```

Frontier

```
local( siz )
```

a little trick to use integers (instead of a string) to calculate a point in Frontier:

*point(h + v * 0x10000)*

```
siz = point(160 + 120*0x10000)
CFG.setFSizeCustom( siz )
```

<u>Set Fractal Size, Menu</u> - Sets the fractal size to standard sizes displayed in the *Size* menu.

If the target window is not a fractal, the global size values are modified, which specify the size of newly created fractals. This information will be reflected in the *New Parameters* window.

Use the *Set Fractal Size, Custom* command for non-standard image sizes.

This command cannot be used when the target window is building a fractal.

Event Class: 'fraG'
Event ID: 'sSiz'
Target: any window

Parameters:

keyDirectObject

Description: Value corresponding to a desired image size.

<u>AppleScript</u>

main screen	1	
9 inch	101	« 512x342
12 inch	102	« 512x384
13 inch	103	« 640x480
19 inch	104	« 1024x768
QuickTime small	201	« 160x120
QuickTime medium	202	« 320x240
QuickTime large	203	« 640x480

<u>Frontier</u>

CFG.const.fsizeMainScreen	1	
CFG.const.fsize9Inch	101	« 512x342
CFG.const.fsize12Inch	102	« 512x384
CFG.const.fsize13Inch	103	« 640x480
CFG.const.fsize19Inch	104	« 1024x768
CFG.const.fsizeQTSmall	201	« 160x120
CFG.const.fsizeQTMedium	202	« 320x240
CFG.const.fsizeQTLarge	203	« 640x480

Descriptor Type: 'shor'

Required/Optional: required

Reply Parameters:

keyDirectObject

Description:	True if the size was valid or false on error.
Descriptor Type:	'bool'

Examples:

AppleScript

```
	tell application "CFG"
		set size menu QuickTime medium
	end tell
```

Frontier

```
	CFG.setFSizeMenu( CFG.const.fsizeQTMedium )
```

<u>Set Limit Extreme</u> - Alter the target's limit extreme value (limits range of randomly calculated fractal data fields).

The target may be any type of window. If the target is the *Help* window or *New Parameters* window, the limit extreme value is applied to the global setting, which affects newly created fractals.

Event Class:	'fraG'
Event ID:	'sLim'
Target:	any window

Parameters:

keyDirectObject

Description:	true to limit outlying random values or false to permit them
Descriptor Type:	'bool'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	true
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
	set limit extreme true
end tell
```

Frontier

```
CFG.setLimitExtreme( true )
```

Set Palette - Makes the specified palette the target's current palette.

If the target is a fractal window, the palette is made the current palette. Colors are altered immediately for 4 and 8 bit systems. If the previous palette and current palette were different sizes or if the target is 16 or 32 bit color depth, the fractals will require a *Redraw*.

If the target is not a fractal window, the new palette will be the default palette for new fractals. This information is displayed in the *New Parameters* window.

The palette menu has a set of standard palette defined and one custom palette per fractal. If you wish to assign the custom palette, be sure to obtain its name by using the *List Palettes* command.

Note that the custom palette displayed in the Palette menu is for the top fractal window only. If the target is not the top-most window, the custom palette in the Palette menu may not be the target's custom palette.

This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'spLs'
Target:	any window

Parameters:

keyDirectObject

Description:	name of the palette; The three standard palettes are defined below for Frontier. Custom palette names must be obtained using the *List Palettes* command.

Frontier

CFG.const.palNameGrayscale	"Grayscale Palette"
CFG.const.palNameRainbow	"Rainbow Palette"
CFG.const.palNameSystem	"System Palette"

Descriptor Type:	'TEXT'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	true if palette loaded, false on error
Descriptor Type:	'bool'

Examples:

AppleScript

```
	tell application "CFG"
		set palette "Rainbow Palette"
		copy result to loaded
	end tell
```

Frontier

```
	local( loaded)
	loaded = CFG.setPalFromList( CFG.const.palNameRainbow )
```

<u>Set Preferences</u> - Copies the data into the target's preferences.

If the target is not a fractal window, the data is copied to the global preferences, which are applied to new fractals. Only those fields that will be altered need to be present in the record. Data validation is performed before the preferences are inserted.

This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'sPrf'
Target:	any window

Parameters:

keyDirectObject

Description:	Record with new preferences data. For safety, one should only supply fields that one wishes to modify. See the *Get Preferences* command for a description of this record format.
Descriptor Type:	'reco'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	true if data applied and false if rejected
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    copy { flags:32768, precision:9 } to prefData
    set preferences prefData
    copy result to applied
end tell
```

Frontier

```
local( prefData, applied )
prefData = { CFG.const.prefsFlags:32768, CFG.const.prefsPrecision:9 }
applied = CFG.setPreferences( prefData )
```

<u>Set Speed</u> - Change the fractal generation speed.

Setting to foreground mode will prevent CFG from processing further Apple Events while a fractal is generating.

Event Class: 'fraG'
Event ID: 'sSpd'
Target: N/A

Parameters:

keyDirectObject

Description:	Speed to set fractal generation.	
	<u>AppleScript</u>	
	foreground	1
	slow	2
	medium	3
	fast	4
	<u>Frontier</u>	
	CFG.const.speedForeground	1
	CFG.const.speedBackSlow	2
	CFG.const.speedBackMedium	3
	CFG.const.speedBackFast	4
Descriptor Type:	'shor'	
Required/Optional:	required	

Reply Parameters:

keyDirectObject

Description:	true on successful speed change, false on error
Descriptor Type:	'bool'

Examples:

AppleScript

```
	tell application "CFG"
		set speed medium
	end tell
```

Frontier

```
	CFG.setSpeed( CFG.const.speedBackMedium )
```

<u>Step Multiple</u> - Calculate scale multiplier for a smooth zoom.

This event calculates the value of the multiplier to zoom from one level to another. It is used to zoom in on a fractal in a particular number of steps (useful for a QuickTime movie). One provides the zoom scale (> 1.0) and the number of steps (> 2.0). The exact algorithm used is: `exp( log( scale ) / (steps - 1.0) )`.

Event Class: 'fraG'
Event ID: 'stpM'
Target: N/A

Parameters:

'scal'

Description:	total zoom amount (2.0 = 2x zoom) (> 1.0)
Descriptor Type:	'exte'
Required/Optional:	required

'dzmy'

Description:	number of steps for the zoom (> 2)
Descriptor Type:	'exte'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	A multiplier. When looping through fractal generation sequences, divide the scale by this value to zoom in or multiply to zoom out.
Descriptor Type:	'exte'

Examples:

AppleScript

```
tell application "CFG"
	step multiple with scale of 10.0 with steps of 101
	copy result to theLog
end tell
```

<u>Window Type</u> - Determine the window type of the target.

The *Custom Fractal* window and *Fractal Movie* window are not supported in the current version of CFG, but may be in future versions (see "Reply Parameters" below).

Event Class:	'fraG'
Event ID:	'gWTp'
Target:	any window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	window type		
	<u>AppleScript</u>		
		fractal	1
		custom fractal	2
		movie fractal	4
		new parameters	64
		help	128
	<u>Frontier</u>		
		CFG.const.wtypeFractal	1
		CFG.const.wtypeCustFrac	2
		CFG.const.wtypeFracMovie	4
		CFG.const.wtypeNewParam	64
		CFG.const.wtypeHelp	128
Descriptor Type:	'shor'		

Examples:

AppleScript

```
	tell application "CFG"
		window type
		copy result to wtype
	end tell
```

Frontier

```
	local( wtype )
	wtype = CFG.windowType()
```

<u>Zoom In or Out</u> - Zooms the target in or out by a factor of two.

The target must be a fractal, but it cannot be a Random Walk fractal. The target may automatically start generating if the appropriate preferences flag bit is set (see the *Get Preferences* command and the *Preferences* section of the *CFG User's Manual*).

This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'zmIO'
Target:	fractal window

Parameters:

keyDirectObject

Description:	True to zoom in or false to zoom out. The zoom in/out constants are not defined for AppleScript.
	<u>Frontier</u>
	CFG.const.zoomIn true
	CFG.const.zoomOut false
Descriptor Type:	'bool'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	true on successful zoom, false on failure
Descriptor Type:	'bool'

Examples:

AppleScript

```
    tell application "CFG"
        zoom in-out true
        copy result to zoomed
    end tell
```

Frontier

```
    local( zoomed )
    zoomed = CFG.zoomInOut( CFG.const.zoomIn )
```

<u>Zoom Selection</u> - Zooms in on the fractal, fitting the selected area to the fractal size.

The target must be a fractal, but it cannot be a Random Walk fractal. The fractal must have a selection. The selection becomes the fractal's bounding rectangle. Thus, the top of the selection will become the new top of the fractal, the selection's left side will become the fractal's left edge, etc.

The target may automatically start generating if the appropriate preferences flag bit is set (see the *Get Preferences* command and the *Preferences* section of the *CFG User's Manual*).

This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'zmSl'
Target:	fractal window

Parameters:

« none »

Reply Parameters:

keyDirectObject

Description:	true on successful zoom, false on failure
Descriptor Type:	'bool'

Examples:

AppleScript

```
tell application "CFG"
    zoom selection
    copy result to zoomed
end tell
```

Frontier

```
local( zoomed )
zoomed = CFG.zoomSelection()
```

<u>Zoom Value</u> - Zoom in (or out) on the fractal by the specified amount.

The target must be a fractal, but it cannot be a Random Walk fractal. The target may automatically start generating if the appropriate preferences flag bit is set (see the *Get Preferences* command and the *Preferences* section of the *CFG User's Manual*).

One zooms in by specifying a value greater than one, while one zooms out with values between zero and one. That is, to zoom out by a factor X, one specifies the zoom amount to be (1 / X). Values less or equal to 0 are invalid.

This command cannot be used when the target window is building a fractal.

Event Class:	'fraG'
Event ID:	'zmVl'
Target:	fractal window

Parameters:

'dzmx'

Description:	amount to zoom horizontally
Descriptor Type:	'exte'
Required/Optional:	required

'dzmy'

Description:	amount to zoom vertically
Descriptor Type:	'exte'
Required/Optional:	required

Reply Parameters:

keyDirectObject

Description:	true on successful zoom, false on failure
Descriptor Type:	'bool'

Examples:

AppleScript

```
	tell application "CFG"
		zoom value horizontal 2.0 vertical 3.0
		copy result to zoomed
	end tell
```

Frontier

```
	local( zoomed )
	zoomed = CFG.zoomValue( 2.0, 3.0 )
```