
Working with AppleScript

This guide provides helpful information and examples for using AppleScript to automate the Macintosh version of KaleidaGraph. A variety of sample scripts, from basic to complex, demonstrate how to control KaleidaGraph from an Apple Event engine.

Before you begin, it is important to note that the following sections are not intended to teach you how to use AppleScript. If you are new to AppleScript, it is highly recommended that you read a book that thoroughly explains the use of AppleScript. One such book is *The Tao of AppleScript*, written by Derrick Schneider, Hayden Books.

You also need to understand that KaleidaGraph is capable of being driven by an Apple Event engine, but KaleidaGraph itself is not an Apple Event engine. The only way to control KaleidaGraph or any other application is through the use of an Apple Event engine, such as AppleScript.

1.1 AppleScript Basics

The following terms are used throughout the examples in this guide.

- **integer** - Any positive whole number, including zero (0, 1, 2, 3, ...).
- **string** - A maximum of 255 characters surrounded by quote marks ("").
- **Boolean** - yes/no/true/false
- **white space character** - space/tab

The following AppleScript commands are also used in the examples. If you are not familiar with any of these commands, it is recommended that you do so before continuing.

activate	contains	get	list folder	set
& (ampersand)	copy	if	repeat	tell
as	display dialog	info for	result	

Comments can be added to a script in one of two ways. The first method is to precede any comments with two hyphens (--). This is useful for single-line comments.

The second method is to place the “(“ and “)” characters at the beginning and end of the comment. This is useful for multi-line comments.

Anytime you want to use **quote marks** (") within a string in AppleScript’s Script Editor, a backslash (\) must precede each quote mark. The reason for this is quote marks indicate the beginning and end of text strings. For example, the text string "file=\"filename\"" contains backslashes prior to the quote marks within the text string.

1.2 KaleidaGraph Dictionary

1.2.1 Overview

AppleScript has a way to view what specific events an application supports for use in a script. This is accomplished by choosing **Open Dictionary** from the **File** menu in the Script Editor and selecting the application of interest. All of the events supported by the application are displayed in a dialog. You can select a specific event and obtain general information about it.

The events supported by KaleidaGraph fall under two categories. The first category is the required suite of events, which are common events that every application should support. The **Run**, **Open**, **Print**, and **Quit** events are all considered part of the required suite.

The remaining events fall into the second category, which is KaleidaGraph events. These are events which only pertain to the KaleidaGraph application. Examples of these events include **AppendColumn**, **RunTextScript**, and **Rebuild**.

The events in this section are listed in alphabetical order. Please note that this is not the same way that they are listed in the Dictionary itself. This was done to make it easier to locate a specific event.

The syntax used in the examples is:

event - description of the event, including any default settings

event parameters
 [optional parameters, where applicable]
 [Result of executing the event, where applicable]

examples using the event

Note: The optional parameters are enclosed in brackets to differentiate them from the required parameters. The result returned by the script is also enclosed in brackets to keep it separate from the event and its parameters.

1.2.2 List of Events

AppendColumn - This event appends one or more columns of data to the right of any existing data in the frontmost data window. The data should be tab-separated and stored as a string. The first row in the string may optionally contain titles. By default, the first row is assumed to contain data.

AppendColumn string
[Titles Boolean]

```

set item1 to "1.00      1.40      6.20
1.50      4.50      5.55"

set item2 to "Time      Test #1  Test #2
1.00      1.40      6.20
1.50      4.50      5.55"

tell application "KaleidaGraph"
  GetData "-1"
  set string1 to result
  AppendColumn string1
  SelectWindow "Data 2"
  AppendColumn item1
  SelectWindow position 3
  AppendColumn item2 with Titles
end tell

```

AppendRow - This event appends one or more rows of data below any existing data in the frontmost data window. The data should be tab-separated and stored as a string. The first row in the string may optionally contain titles. By default, the first row is assumed to contain data.

AppendRow string
[Titles Boolean]

```

set item1 to "1.00      1.40      6.20
1.50      4.50      5.55"

set item2 to "Time      Test #1  Test #2
1.00      1.40      6.20
1.50      4.50      5.55"

tell application "KaleidaGraph"
  GetData "-1"
  set string1 to result
  AppendRow string1
  SelectWindow "Data 2"
  AppendRow item1
  SelectWindow position 3
  AppendRow item2 with Titles
end tell

```

Close - This event closes a single window. The frontmost window is closed by default, unless a window is specified by either its name in the title bar or by its position (from front to back, where the frontmost window is position 0). The default action is to ask if any changes should be saved; however, it is possible to automatically save the changes or to close the window without saving its contents using the saving option.

```
Close string
  [position integer]
  [saving yes/no/ask]

  tell application "KaleidaGraph"
    Close "Data 1"
    Close position 1
    Close without saving
    Close position 3 with saving
  end tell
```

CloseAllWindows - This event closes all visible windows, without saving their contents.

```
CloseAllWindows

  tell application "KaleidaGraph"
    CloseAllWindows
  end tell
```

CloseFrontWindow - This event closes the frontmost window, without saving its contents.

```
CloseFrontWindow

  tell application "KaleidaGraph"
    CloseFrontWindow
  end tell
```

ExportPlot - This event exports the frontmost plot to a file or to the Clipboard (provided KaleidaGraph is the frontmost application). By default, the file is stored in the current directory. You can specify a different location by either using the SetRefDirectory command or including the path in the file name. The attributes of the file can be set as part of a string and are listed under the #PICT/OPT Formula Script command, which is discussed in Section 5.10.2 of the KaleidaGraph manual.

```
ExportPlot string
  [Name alias]
  [Clipboard Boolean]
  [Result: picture]

  set item2 to "
  type = pict
  scale 50
  post_pict yes
  file = \"Sample PICT\"

  tell application "KaleidaGraph"
    activate
    ExportPlot item2
    ExportPlot with Clipboard
  end tell
```

GetData - This event makes a selection in the frontmost data window and stores the data within it as a string of tab-separated data. The selection range should be a string containing numbers separated by any white space characters, where the ordering is (start row, end row, start column, end column). As a special case, if start row has a value of -1, the entire window is selected. The selection addresses for both row and column positions begin at 0 and are counted from the upper-left corner of the data window (position 0,0).

```
GetData string
[Result: string]

tell application "KaleidaGraph"
  GetData "-1"
  GetData "0 100 1 3"
end tell
```

GetWindowName - This event returns the name of one or more windows. If a specific window position (from front to back, where the frontmost window is position 0) is given, the name of the window is returned as a string. Otherwise, the names of all windows are returned in a record.

```
GetWindowName
[position integer]
[Result: string or record]

tell application "KaleidaGraph"
  GetWindowName
  GetWindowName position 1
end tell
```

LoadPlotScript - This event imports the specified plot script file.

```
LoadPlotScript string

set item1 to "Macintosh HD:Examples folder:Scripts:Test script"

tell application "KaleidaGraph"
  LoadPlotScript "Macintosh HD:Examples:Sample script"
  RunPlotScript
  LoadPlotScript item1
  RunPlotScript
end tell
```

Open - This event opens the specified file. It can be used to open any type of file which KaleidaGraph recognizes.

```
Open alias string

set item1 to "Macintosh HD:Examples:sunspot data"

tell application "KaleidaGraph"
  Open alias "Macintosh HD:Examples:Housing Starts"
  Open alias item1
  copy {"Macintosh HD:sunspot data", "Macintosh HD:Sample Plot"} to List1
  repeat with i in List1
    Open alias i
  end repeat
end tell
```

OpenDatafile - This event is used to open binary and text data files. For all other types of files (plots, styles, macros), use the **Open** event. The attributes of the data file can be set as part of a string and are listed under the #DATAFILE Formula Script command, which is discussed in Section 5.10.2 of the KaleidaGraph manual.

OpenDatafile string

[Name alias]

[Result: string]

```
set item1 to "Macintosh HD:KaleidaGraph™ 3.0:Examples folder:Scripts:"
```

```
set item2 to "file = \"Macintosh HD:Test folder:example - text\"  
delimiter = tab  
skip = 3  
read_titles = yes"
```

```
tell application "KaleidaGraph"  
  SetRefDirectory item1  
  OpenDatafile "file = \"sunspot data\""  
  OpenDatafile item2  
end tell
```

PlotPrint - This event prints one or more plots. By default, the frontmost plot is printed directly from the plot window. It is possible to print multiple plots from the layout window using the PageLayout option.

PlotPrint

[PageLayout Boolean]

```
tell application "KaleidaGraph"  
  PlotPrint  
  SetSelection "10 20 0 3"  
  Rebuild  
  PlotPrint with PageLayout  
end tell
```

Print - This event prints the specified file. It can be used to print any data (binary or text) or plot file.

Print alias string

```
set item1 to "Macintosh HD:Examples:Test plot"
```

```
tell application "KaleidaGraph"  
  Print alias "Macintosh HD:Examples:Sample plot"  
  Print alias item1  
  copy {"Macintosh HD:sunspot data", "Macintosh HD:Sample Plot"} to List1  
  repeat with i in List1  
    Print alias i  
  end repeat  
end tell
```

Quit - This event quits the KaleidaGraph application. The default action is to ask if any changes should be saved; however, by specifying an option it is possible to automatically save the changes or to quit without saving.

Quit yes/no/ask

```
tell application "KaleidaGraph"  
  Run  
  activate  
  Quit no  
end tell
```

Ready - This event checks to see if KaleidaGraph is up and running.

Ready
[Result: Boolean]

```
tell application "KaleidaGraph"  
  repeat while (Ready) is false  
  end repeat  
end tell
```

Rebuild - This event rebuilds the plot based on the current selection in the data window.

Rebuild

```
tell application "KaleidaGraph"  
  SetSelection "100 200 0 4"  
  Rebuild  
end tell
```

Run - This event launches the KaleidaGraph application.

Run

```
tell application "KaleidaGraph"  
  Run  
end tell
```

RunPlotScript - This event executes the current plot script.

RunPlotScript

```
tell application "KaleidaGraph"  
  LoadPlotScript "Macintosh HD:Examples:Sample script"  
  RunPlotScript  
end tell
```

RunTextScript - This event executes the text script stored as a string. This event can execute any of the Formula Script commands listed in Section 5.10.2 of the KaleidaGraph manual. If you are using the #FORMULA command, it is not necessary to include the #FORMULA and #END statements for this command. It is the default action for the RunTextScript event.

RunTextScript string
[Result: picture]

```
set item1 to "  
#PASTE  
title = true 0 10 1 2  
#END"  
  
set test1 to "macro(\"Integrate - Area\")"  
  
set test2 to "  
#SCRIPT  
plot_type=scatter  
begin_group  
x 0  
y 1  
end_group  
#END"  
  
tell application "KaleidaGraph"  
  RunTextScript "  
c4=mean([0:0, 1:3]);  
c5=stderr([0:0, 1:3]);"  
  RunTextScript item1  
  RunTextScript test1  
  RunTextScript test2  
end tell
```

SelectWindow - This event selects a window specified either by name (in the title bar) or by position (from front to back, where the frontmost window is position 0).

SelectWindow string
[position integer]

```
set item3 to "Data 3"  
  
tell application "KaleidaGraph"  
  SelectWindow position 1  
  SelectWindow "Data 1"  
  SelectWindow item3  
end tell
```

SendData - This event sends data to KaleidaGraph to be displayed in a new data window. The data should be tab-separated and stored as a string. By default, the first row is assumed to contain titles.

SendData string
[Titles Boolean]

```
set item1 to "1      1      1
2      2      2
3      3      3"

tell application "KaleidaGraph"
    if Ready is true
        SendData item1 without titles
    end if
end tell
```

SetRefDirectory - This event sets the Reference Directory. The string is a fully or partially qualified path to the desired directory. The path should end in a colon (:) if the last name is not a file. To move up one directory, the path should be represented by two colons (::). To set the default directory to be the application directory, the path should be left blank("").

SetRefDirectory string

```
set item1 to "Macintosh HD:KaleidaGraph™ 3.0:Examples folder:Sample Data:"

tell application "KaleidaGraph"
    SetRefDirectory item1
    OpenDatafile "file = \"sunspot data\""
    SetRefDirectory "Macintosh HD:Test folder:"
    OpenDatafile "file = \"example #1\""
end tell
```

SetSelection - This event makes a selection in the frontmost data window. The selection range should be a string containing numbers separated by any white space characters, where the ordering is (start row, end row, start column, end column). As a special case, if start row has a value of -1, the entire window is selected. The selection addresses for both row and column positions begin at 0 and are counted from the upper-left corner of the data window (position 0,0).

SetSelection string

```
tell application "KaleidaGraph"
    GetData "0 15 0 1"
    set string1 to result
    SelectWindow position 1
    SetSelection "0 15 1 2"
    AppendColumn string1
end tell
```

1.3 AppleScript Examples

All of the examples in this section follow the same basic format. An overview describes what the script is going to do. Then the example script is listed, just as it would appear if it were being entered directly into AppleScript's Script Editor. Following the script is an explanation which describes each step of the script.

Note: The following examples do not use the **Run** event. This is because the **tell** command launches KaleidaGraph if it is not already running.

1.3.1 Basic Examples

Basic Example 1

This example script performs the following functions:

- Makes a data selection in the frontmost data window and stores it as a string.
- Selects the second window from the front.
- Appends the string to the frontmost data window.
- Brings KaleidaGraph to the foreground.

The example script is listed below, followed by a description of how it operates.

```
tell application "KaleidaGraph"  
  GetData "-1"  
  set string1 to result  
  SelectWindow position 1  
  AppendColumn string1  
  activate  
end tell
```

The **GetData** command selects all of the cells in the frontmost data window and stores it as a text string of tab-separated data. This text string is then stored in a variable called `string1` using the **set** command. The **SelectWindow** command activates the second window from the front (position 1). The **AppendColumn** command appends the data from `string1` into the frontmost data window. The **activate** command makes KaleidaGraph the active application.

Basic Example 2

This example script performs the following functions:

- Opens a data file.
- Performs calculations on the data set.
- Closes the data window, after saving changes.

The example script is listed below, followed by a description of how it operates.

```

tell application "KaleidaGraph"
    Open alias "Macintosh HD:KaleidaGraph™ 3.0:Examples folder:Sample Data:Housing Starts"
    RunTextScript "
c4=mean([0:0, 1:3]);
c5=stderr([0:0, 1:3]);"
    Close with saving
end tell

```

The **Open** command opens the specified data file. The **RunTextScript** command executes two Formula Entry commands which calculate the mean and standard error on a portion of the data file. The **Close** command saves the data file and closes the data window.

Basic Example 3

This example script performs the following functions:

- Makes a selection in the frontmost data window.
- Executes two Formula Entry commands.
- Creates a Scatter plot of the raw data.
- Brings KaleidaGraph to the foreground.

The example script is listed below, followed by a description of how it operates.

```

set test1 to "
c0 = index()+1;
c1 = log(c0);"

set test2 to "
#SCRIPT
x axis limits 0 100
plot_type=scatter
begin_group
x 0
y 1
end_group
#END"

tell application "KaleidaGraph"
    SetSelection "0 99 0 0"
    RunTextScript test1
    RunTextScript test2
    activate
end tell

```

The two **set** commands at the beginning of the script store text strings in the test1 and test2 variables. The test1 string contains two Formula Entry commands. The test2 string contains information for running a plot script.

The **SetSelection** command selects rows 0 through 99 in column 0 of the frontmost data window. The first **RunTextScript** command creates two columns of data in the data window by executing two commands from Formula Entry. The second **RunTextScript** command generates a Scatter plot of this data. The **activate** command makes KaleidaGraph the active application.

1.3.2 Intermediate Examples

Intermediate Example 1

This example script performs the following functions:

- Checks to see if KaleidaGraph is running.
- Selects two different windows and brings them to the front.
- Appends data to each of these data windows.

The example script is listed below, followed by a description of how it operates.

```
set item1 to "1.00      1.40      6.20
1.50      4.50      5.55
2.00      4.30      5.00
2.50      5.35      4.50
3.00      4.75      4.10"

set item2 to "Time      Test #1   Test #2
1.00      1.40      6.20
1.50      4.50      5.55
2.00      4.30      5.00
2.50      5.35      4.50
3.00      4.75      4.10"

tell application "KaleidaGraph"
    repeat while (Ready) is false
    end repeat
    SelectWindow "Data 1"
    AppendColumn item1
    SelectWindow position 1
    AppendColumn item2 with Titles
    activate
end tell
```

The two **set** commands at the beginning of the script store text strings in the item1 and item2 variables. The item1 string contains five rows of data without any titles. The item2 string contains six rows of data, with column titles in the first row.

The **repeat** loop uses the **Ready** command to see if KaleidaGraph is running. If not, the script waits until KaleidaGraph has launched. The **SelectWindow** command brings the window titled **Data 1** to the front. The **AppendColumn** command appends the data in the item1 string to this data window. The second **SelectWindow** command activates the second window from the front. The second **AppendColumn** command operates in the same manner as the first, except that this time the first row contains titles. The **activate** command makes KaleidaGraph the active application.

Intermediate Example 2

This example script performs the following functions:

- Opens a saved data file.
- Creates a Line plot.
- Saves the plot as a PICT file.

The example script is listed below, followed by a description of how it operates.

```

set path1 to "Macintosh HD:KaleidaGraph™ 3.0:Examples folder:Sample Data:"

set item1 to "
#SCRIPT
y axis title \"Range\"
plot_type = Line
begin_group
x 0
y 1
y 2
y 3
end_group
#END"

set item2 to "
type = pict
scale 50
post_pict yes
file = \"Sample PICT\""

tell application "KaleidaGraph"
    SetRefDirectory path1
    OpenDatafile "file = \"sunspot data\""
    RunTextScript item1
    ExportPlot item2
end tell

```

The three **set** commands at the beginning of the script store text strings in the path1, item1, and item2 variables. The path1 string contains the path to the **Sample Data** folder. The item1 string contains information for running a plot script. The item2 string contains the parameters for exporting a PICT file.

The **SetRefDirectory** command sets the current directory to the string contained in path1. The file named **sunspot data** in the current directory is opened using the **OpenDatafile** command. The **RunTextScript** command executes the Plot Script defined in the item1 string. The **ExportPlot** command saves the newly created plot as a PICT file. The file is saved in the same directory as the original data file.

Intermediate Example 3

This example script performs the following functions:

- Checks to see if KaleidaGraph is running.
- Generates data using two commands from Formula Entry.
- Loads and executes a saved plot script. The plot script is assumed to use a template which has a Polynomial curve fit and error bars applied to it.
- Stores the curve fit values in the data window and calculates the residuals.
- Updates the plot to include the new data and prints it.

The example script is listed below, followed by a description of how it operates.

```
set data1 to "  
macro("\pi Series\  
macro("\sinc(5x)\");"  
  
set data2 to "  
c3=poly(c0,c1);  
name("\Residuals\  
c2=c3-c1;"  
  
tell application "KaleidaGraph"  
  repeat until Ready  
  end repeat  
  RunTextScript data1  
  LoadPlotScript "Macintosh HD:KG Script"  
  RunPlotScript  
  RunTextScript data2  
  Rebuild  
  PlotPrint  
end tell
```

The two **set** commands at the beginning of the script store text strings in the data1 and data2 variables. The data1 string contains two formulas which execute two macros from the **Macros** menu. The data2 string contains formulas for calculating the values and residuals of a Polynomial curve fit.

The **repeat** loop uses the **Ready** command to see if KaleidaGraph is running. If not, the script waits until KaleidaGraph has launched. The first **RunTextScript** command creates two columns of data. The **LoadPlotScript** command imports a saved plot script, which is executed by the **RunPlotScript** command. The second **RunTextScript** command calculates the values and residuals of the Polynomial curve fit. The **Rebuild** command updates the plot to reflect the changes made to the data window by the previous formulas. The resulting plot is printed using the **PlotPrint** command.

1.3.3 Advanced Examples

Advanced Example 1

This example script performs the following functions:

- Prompts the user to enter the full path to a data file.
- Opens the data file and a saved Style file.
- Creates a Column plot of the data and prints it.
- Makes a selection in the data window.
- Rebuilds the plot and prints it.

The example script is listed below, followed by a description of how it operates.

```

set test1 to "
#FORMULA
c4=mean([0:0, 1:3]);
name(\"Average\", c4);
#END

#SCRIPT
y axis title \"Range\"
plot_type column
begin_group
x 0
y 1
y 2
y 4
end_group
#END"

tell application "KaleidaGraph"
  display dialog "Please enter the path to the data file." default answer "" buttons {"OK", "Cancel"}
  default button "OK"
  set rec1 to result
  get text returned of rec1
  set path1 to result
  Open alias path1
  Open alias "Macintosh HD:Modified Style"
  RunTextScript test1
  PlotPrint
  SelectWindow position 1
  SetSelection "0 5 0 4"
  Rebuild
  PlotPrint
end tell

```

The **set** command at the beginning of the script stores two formulas and the information for running a plot script into the test1 variable.

The **display dialog** command prompts the user to enter the full path to a data file, which is opened later by the script. The two **set** commands and the **get** command that follow are used to store the full path in the path1 variable. The first **Open** command opens the specified data file. The second **Open** command opens a saved Style file. The **RunTextScript** command executes the commands in the test1 string which calculate the mean and generate a plot. The plot is printed using the first **PlotPrint** command.

The **SelectWindow** command brings the data window to the front. The **SetSelection** command causes rows 0 to 5 in columns 0 through 4 to be highlighted. The **Rebuild** command causes the plot to be updated using the data which is currently selected in the data window. The second **PlotPrint** command prints the updated plot.

Advanced Example 2

This example script performs the following functions:

- Determines the file types of all files in a specified folder.
- Opens any KaleidaGraph binary data files that are found.
- Opens any text files using the specified file format.
- Displays a dialog showing the total number of files that were opened.

The example script is listed below, followed by a description of how it operates.

```
set path1 to "Macintosh HD:Examples:"

set item1 to "
delimiter = tab
skip = 5
read_titles = yes
del_number = 1"

set x to 0

tell application "KaleidaGraph"
    list folder path1
    set List1 to result
    repeat with i in List1
        set item2 to path1 & i
        info for file item2
        set rec1 to result
        if rec1 contains {file type:"QDAT"} or rec1 contains {file type:"TEXT"} then
            if rec1 contains {file type:"QDAT"} then
                Open alias item2
                copy x + 1 to x
            else
                set item3 to ("file =" & "\"" & item2 & "\"") & item1
                OpenDatafile item3
                copy x + 1 to x
            end if
        end if
    end repeat
    display dialog (result as string) & " data windows were opened by the script."
end tell
```

The three **set** commands at the beginning of the script store strings in the path1 and item1 variables, in addition to storing a value of 0 in the x variable. The path1 string contains the path to the **Examples folder**. The item1 string contains parameters for importing a text file into KaleidaGraph.

The **list folder** command compiles a list of all the items in the folder specified by path1. The **set** command stores this list in the List1 variable. The **repeat** command sets up a loop so that the commands in the loop operate on each file in the folder. The i variable stores the name of the file currently being evaluated.

The first **set** command within the loop concatenates the path1 and i strings into a single string. This string is stored in the item2 variable. The **info for** command obtains a list of information about the specific file. Using the second **set** command in the loop, this information is stored as a record in the rec1 variable.

The **if** command tests if the file type of each file is either QDAT or TEXT (QDAT is the file type for KaleidaGraph binary data files and TEXT is the file type for generic text files). If the file type does not match either one of these, the file is skipped and the script operates on the next file in the folder. If the file type is QDAT, the file is opened using an **Open** command. If the file type is TEXT, the **set** command concatenates five text strings and stores the resulting string in the item3 variable. The file is then opened using the **OpenDatafile** command.

The **copy** command increases the value of the x variable each time a data file is opened. After all of the files have been processed, the **display dialog** command shows how many files were opened by the script.

Advanced Example 3

This example script performs the following functions:

- Displays four dialogs so the user can enter the path to the folder containing the data files, the base name of the data files, the number of files to be opened, and the number of columns in each file. The data files are assumed to have the same base name followed by a numeric suffix (for example, data_1, data_2, data_3, where **data_** is the base name).
- Merges the specified number of data files into a single file.

The example script is listed below, followed by a description of how it operates.

```
set part1 to "  
#MERGEFILE  
file = \"  
  
set part2 to \"\  
delimiter = tab  
skip = 3  
read_titles = yes  
del_number = 1  
position = 0 "  
  
set part3 to "  
#END"  
  
set x to 1  
  
tell application "KaleidaGraph"  
    display dialog "Please enter the path to the data files." default answer "" buttons {"OK", "Cancel"}  
    default button "OK"  
    set rec1 to result  
    get text returned of rec1  
    set path1 to result  
    display dialog "Please enter the base name of the data files." default answer "" buttons {"OK",  
"Cancel"} default button "OK"  
    set rec2 to result  
    get text returned of rec2  
    set base to result  
    display dialog "Please enter the number of files to be opened." default answer "" buttons {"OK",  
"Cancel"} default button "OK"  
    set rec3 to result  
    get text returned of rec3  
    set num to result as number  
    display dialog "Please enter the number of columns in the files." default answer "" buttons {"OK",  
"Cancel"} default button "OK"  
    set rec4 to result  
    get text returned of rec4  
    set col to result as number  
  
    repeat until x > num  
        copy col * (x - 1) to colnum  
        set name1 to path1 & base & x as string  
        set item1 to (part1 & name1 & part2 & colnum as string) & part3  
        RunTextScript item1  
        copy x + 1 to x  
    end repeat  
end tell
```

The first three **set** commands define portions of a **#MERGEFILE** command so that the script can substitute the file name and the starting location where the file should be merged. The fourth **set** command stores a value of 1 in the *x* variable.

The script uses four **display dialog** commands to enter information about the files which will be merged. The first dialog prompts for the path to the data files, which is stored in the *path1* variable. The second dialog prompts for the base name of the files, which is stored in the *base* variable. The third dialog asks for the number of files to be merged, which is stored in the *num* variable. The last dialog asks for the number of columns in the files, which is stored in the *col* variable.

The **repeat** command sets up a loop that runs until the specified number of files have been merged. The **copy** command stores the current column number in the *colnum* variable. The first **set** command in the loop combines the *path1*, *base*, and *x* strings into a single string, which is stored in the *name1* variable. The second **set** command combines the various portions of the **#MERGEFILE** command, the *name1* string, and the starting column number into a single string, which is stored in the *item1* variable. The **RunTextScript** command executes the string stored in *item1* to merge each file into the frontmost data window. The **copy** command increases the *x* variable by 1.

1.4 Tips

You may want to place a **repeat** loop at the beginning of your scripts so that the script waits until KaleidaGraph has finished launching before it continues. This is particularly important when the script requires a data window, such as when using Formula Entry commands. If KaleidaGraph is not running, the script may attempt to execute the formulas before a data window has been created, resulting in an error. The first and last example scripts in Section 1.3.2 make use of the repeat loop in conjunction with the **Ready** command.

KaleidaGraph and AppleScript do not have any special commands for selecting commands or buttons in dialogs. You need to have a supplemental program like QuicKeys (a commercial package from CE Software) to perform these types of functions.

The only way to apply curve fits or error bars to a plot, without using a supplemental application, is to open a saved plot script. The saved plot script needs to be pointed at a template which has the curve fit or error bars already applied. Any plots created by the plot script automatically have any curve fits or error bars applied.

If you need to generate the trademark symbol (TM), it can be created by typing **Option+2**. It is not possible to create this symbol with the 2 key located on the numeric keypad.

1.5 Troubleshooting

If you run into problems when attempting to execute your script, AppleScript has two commands in the **Controls** menu which can help you see what is happening. The **Show Result** command displays the result returned by the last script. The **Open Event Log** command displays all of the events sent by a script as it executes.

Beware of System Extensions that modify the Open and Save dialogs. You may have problems saving files into a different directory unless these kinds of extensions are disabled.

Errors can result if any of the following occur:

- Specifying a position number for which no window exists.
- Specifying a window by an incorrect name.
- Trying to use the **GetData** or **SetSelection** events when no data windows are open.
- Trying to use the **ExportPlot**, **PlotPrint**, or **Rebuild** events when no plot windows are open.

When using the Formula Script commands within the **RunTextScript** command, errors result if the # statements are not entered in all capital letters or if any lines within a Formula Script command are indented. The Formula Script commands should line up with the **tell** command in the script.

To successfully use the #COPY command as part of a RunTextScript event, KaleidaGraph must be made the active application prior to copying data to the Clipboard. The easiest method to do this is to place an **activate** command prior to the RunTextScript event. Otherwise, the information on the Clipboard is not altered when the #COPY is executed. This is because it is only possible to replace the contents of the Clipboard from the frontmost application.

Under System 7.5 or later, the **General Controls** control panel has a setting which can interfere with the **SetRefDirectory** command. The **Documents** portion of the General Controls dialog must have the **Folder which contains the application** option selected for this command to work properly.

1.6 Apple Events

KaleidaGraph supports the following set of Apple Events. A description for each of the events is located after the table.

Events (Class: QKPT)				
ID	Direct	Optional	Return	Meaning
call	none	none	none	Close all visible windows without saving.
cfns	none	none	none	Close the frontmost window without saving.
clos	TEXT	none	none	Close the specified window, optionally saving its contents.
epic	TEXT	FILE	PICT (opt)	Export the frontmost plot to a file or the Clipboard as a PICT.
gsel	TEXT	none	TEXT	Get the specified data selection from the frontmost data window.
kdoc	TEXT	FILE	none	Load the specified data file.
kgqt	none	none	none	Quit KaleidaGraph as soon as possible.
ldac	TEXT	none	none	Append one or more columns of data to the frontmost data window.
ldap	TEXT	none	none	Append one or more rows of data to the frontmost data window.
ldat	TEXT	none	none	Load tab-separated data, which has titles in the first row.
ldnt	TEXT	none	none	Load tab-separated data, which has data in the first row.
lscp	TEXT	FILE	none	Load the specified plot script.
oapp	none	none	none	Run the application.
odoc	TEXT	FILE	none	Open the specified documents.
pdoc	TEXT	FILE	none	Print the specified documents.
prpl	TEXT	none	none	Print one or more plots.
quit	TEXT	none	none	Quit the application.
redy	none	none	none	Check to see if KaleidaGraph is running.
refd	TEXT or typeFSS	none	none	Set the reference directory.

Events (Class: QKPT)				
ID	Direct	Optional	Return	Meaning
rplt	none	none	none	Replot the active plot using the current selection in the data window.
rscp	none	none	none	Execute the current plot script.
selw	TEXT	none	none	Select a window by either name or position.
ssel	TEXT	none	none	Set the selection in the frontmost data window.
tscp	TEXT	none	PICT (opt)	Execute the text script.
wlis	TEXT	none	TEXT	Get the name of one or more windows in the list. If a position is not specified, the entire window list will be returned.

Note: The Application Signature is **QKPT**.

call

This event closes all of the windows, without saving their contents.

cfns

This event closes the frontmost window, without saving its contents.

clos

This event closes the specified window, optionally saving its contents.

epic

This event has several forms and both of the parameters are optional. The syntax for this event is the same as the #PICT/OPT Formula Script command. This command is discussed in Section 5.10.2 of the KaleidaGraph manual.

If no parameters are supplied, the frontmost plot is returned as a PICT. The direct object is text and sets the scale factor, Postscript PICT, and file parameters. The optional FILE parameter is of typeFSS or typeAlias and allows the PICT to be saved to a file.

Note: The PICT is returned only if a file is not specified.

gsel

Sets the selection in the frontmost data window and returns the data contained in it. The selection range is contained in the direct object as white space separated numbers. The syntax for this event is the same as the #COPY Formula Script command. This command is discussed in Section 5.10.2 of the KaleidaGraph manual.

kdoc

Loads the data file described in either the direct object or the FILE object. If the data file is text, it uses the information in the direct object (or the current text file definition) to import the file. The optional FILE object is of typeFSS or typeAlias. The syntax for this event is the same as the #DATAFILE Formula Script command. This command is discussed in Section 5.10.2 of the KaleidaGraph manual.

kgqt

This event quits KaleidaGraph as soon as it is possible.

ldac

Appends one or more columns of data contained in the direct object to the frontmost data window. The data is assumed to be tab-separated, with data in the first row.

ldap

Appends one or more rows of data contained in the direct object to the frontmost data window. The data is assumed to be tab-separated, with data in the first row.

ldat

Loads the data contained in the direct object into a new data window. The data is assumed to be tab-separated, with titles in the first row.

ldnt

Loads the data contained in the direct object into a new data window. The data is assumed to be tab-separated, with data in the first row.

lscp

Loads the plot script specified in either the direct object or the FILE object. This file should be in the same folder as the last plot script that was opened or specified by a fully qualified path. The optional FILE object is of typeFSS or typeAlias.

redy

Checks to see if KaleidaGraph is running.

refd

Sets the reference directory to be the directory specified in the direct object. The direct object may be of type TEXT, typeFSS, or typeAlias. If it is of type TEXT, the object should contain a simple text string describing either a full or partial path to the new directory.

The reference directory is used as the base directory for all simple file names and partial paths, except for those describing script files. Script files have their own base directory. This event has no optional parameter.

rplt

Replots the active plot, using the current selection in the data window.

rscp

Executes the current plot script.

selw

Select a window by either name or position.

sse1

Sets the selection in the frontmost data window. The selection range is contained in the direct object as white space separated numbers. The syntax for this event is the same as the #SELECTION Formula Script command. This command is discussed in Section 5.10.2 of the KaleidaGraph manual.

tscp

Executes the text script contained in the direct object. The commands supported by this event are listed in Section 5.10.2 of the KaleidaGraph manual. All of the commands can be used to write a formula script in the Posted Note of the Formula Entry window.

wlis

Gets the name of one or more windows in the list. If a position is not specified, the entire window list is returned.