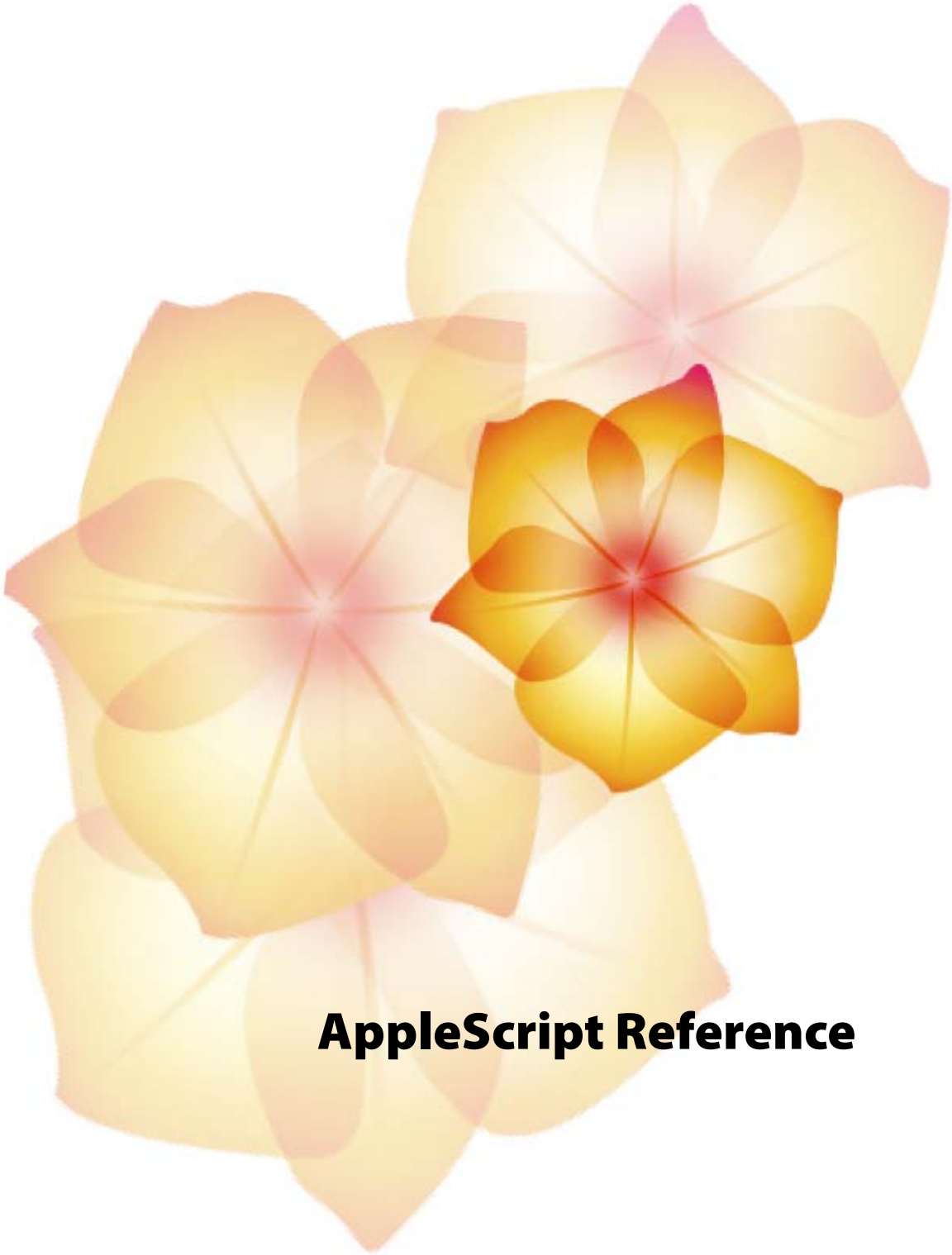


Adobe® Illustrator® CS2

AppleScript Reference

Adobe

Adobe® Illustrator® CS2 AppleScript Reference for Macintosh®.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Contents

1 Introduction

This manual provides an introduction to scripting Adobe® Illustrator® CS2 in Mac OS® X.

About this Manual

This document contains the following chapters:

- "Introduction" —An introduction to scripting.
- "Scripting Basics" — The basics of the AppleScript scripting language for Mac OS. If you are new to scripting, be sure to read this chapter.
- "Scripting Illustrator" — A brief introduction to the specifics of scripting Illustrator. Concepts and approaches specific to the application are covered here, such as measurement units, matrices, and color models.
- "AppleScript Objects" — Details and examples for every object in Illustrator's AppleScript dictionary.
- "AppleScript Commands" — Details and examples for every command in Illustrator's AppleScript dictionary.

For further information and developments on this and other Adobe products, see the Adobe Solutions Network website:

```
http://partners.adobe.com/asn
```

What is Scripting?

A script is a series of commands that tells Adobe Illustrator CS2 to perform a series of actions. These actions can be simple, and affect only a single, selected object in the current document; or complex, and affect all of the objects in all of your Illustrator documents. The actions might involve only Illustrator, or they might involve other applications, such as word processors, spreadsheets, and database management programs. Many of the tasks you can perform with Illustrator's tools, menus, palettes, and dialog boxes can be performed by a script (a notable exception is third-party plug-ins, which cannot be scripted at this time).

We naturally think of scripting as a way to automate repetitive tasks, but it can also be a creative tool. You can use scripts for creative tasks that would be too difficult or time consuming to do manually. For example, you could write a script to systematically create a series of objects, modifying the new objects' position, stroke, and fill properties along the way. You could also write a script that accesses Illustrator's built-in transformation matrix functions to stretch, scale and distort a series of objects. Without scripting, you'll likely miss out on the creative potential of such labor-intensive techniques.

Scripting isn't just for computer programmers—it's for everybody. You don't need a degree in computer science or mathematics to write scripts that can automate a wide variety of common tasks. If you can read this text, you can write scripts.

The language you use to write scripts depends on the operating system of the platform you're using: AppleScript for Mac OS; Visual Basic for Windows®; JavaScript for either platform. Each of these languages is described in a separate manual.

Why use scripting?

Graphic design is a field characterized by creativity, but aspects of the actual work of illustration and page layout are anything but creative. When you think about the work that you do, chances are good you'll find that you spend most of your time doing the same or similar production tasks, over and over again. In fact, you'll probably notice that the time you spend placing and replacing images, correcting errors in text, and preparing files for printing at an image setting service provider often reduce the time you have available for doing creative work.

Wouldn't it be great if you had an assistant—one that wouldn't mind doing some or all of the boring, repetitive tasks for you? With that kind of help, you'd have more time to concentrate on the creative aspects of your work.

With a small investment of time, Illustrator scripting can be the assistant you need. You can start with short, simple scripts that save you a few seconds every day, and move on to scripts that work all night while you're sleeping.

Think about your work—is there a repetitive task that's driving you crazy? If so, you've identified a candidate for a script. What are the steps involved in performing the task? What are the conditions in which you need to do the task? Once you understand the process you go through to perform the task, you'll be ready to turn it into a script.

What about actions?

Illustrator actions are different from scripts. An Illustrator action is a series of tasks you have recorded while using the application—menu choices, tool choices, object selection, and other commands. When you "play" an action, Illustrator performs all of the recorded commands.

You record, play, edit, and delete actions using Illustrator's built-in Actions palette. The "Automating Tasks" chapter in the Adobe Illustrator User Guide covers actions in detail.

With the introduction of scripting for Illustrator, it is important to avoid any confusion about the difference between actions and scripting. Actions and scripts are both ways of automating repetitive tasks, but they work very differently. The following points summarize the key differences.

- Actions use a program's user interface to do their work. As an action runs, menu choices are executed, objects are selected, and recorded paths are created. Scripts do not use a program's user interface to perform tasks, and can execute faster than actions.
- Actions have very limited facilities for getting and responding to information. You cannot add conditional logic to an action. Therefore, actions cannot make decisions based on the current situation. Scripts are capable of getting information and making decisions and calculations based on the information they receive from Illustrator.
- A script can execute an action, but actions cannot execute scripts.

Script Support in Adobe Illustrator CS2

The Scripts menu supports AppleScript and JavaScript scripts for Mac OS, and VBScript, JavaScript, and Visual Basic scripts for Windows.

For a file to be recognized by Adobe Illustrator CS2 as a valid script file it must have the correct file type (Mac OS) or name extension (Mac OS and Windows).

Script Type	File Type	Extension	Platform(s)
AppleScript	compiled script OSAS file	`.scpt` (none)	Mac OS
JavaScript ExtendScript	text	`.js` `.jsx`	Mac OS & Windows
VBScript	text	`.vbs`	Windows
Visual Basic	executable	`.exe`	Windows

For files in Mac OS (9.x and 10.x) it is possible for files to be identified by the classic file type and creator codes, file name extensions, or both. The `.scpt` extension is only required for AppleScript files which do not have file type information, such as those installed with Mac OS X. There is no harm in having a name extension when a file has file type codes.

Executing scripts

Adobe Illustrator CS2's interface includes a Scripts menu (**File > Scripts**) which provides quick and easy access to your scripts. Scripts can be listed directly as menu items, that run when you select them, or you can navigate to and run any script in your file system.

If Illustrator encounters an error during script execution, it displays the error message returned by the script in an error dialog.

Note: It is not possible to execute scripts that contain the `do script` command from the **Scripts** menu. Attempting to do so causes an error.

Installing scripts

To install a script in the Scripts menu, place it in the Scripts folder (**Illustrator CS2 > Presets > Scripts**). The names of the scripts in the Scripts folder, less any file name extension, will be displayed in the Scripts menu. Any number of scripts may be installed in the Scripts menu.

Scripts added to the Scripts folder while Illustrator is running will not appear in the Scripts menu until the next time you launch Illustrator.

If you have a large collection of scripts you wish to use, you may use sub-folders in the Scripts folder to help organize the scripts in the Scripts menu. Each subfolder will be displayed as a separate submenu containing the scripts in that subfolder.

Note: Because of a limitation in Mac OS, there is a limit of 4 levels of nested sub-folders inside the Scripts folder for the Mac OS version of Illustrator.

Executing other scripts

The **Other Scripts** item at the end of the **Scripts** menu (**File > Scripts > Other Scripts...**) allows you to execute scripts which are not installed in the Scripts folder. Selecting **Other Scripts** displays a file browser dialog which allows you to select a script file for execution. Only files which are of one of the supported file types are displayed in the browse dialog. When you select a script file, it is executed the same way as an installed script.

System requirements for Mac OS

Make sure the scripting plug-in is installed on your system before attempting to script Illustrator.

To write scripts in Mac OS, you must have Mac OS X 10.2. You will also need AppleScript and a script editor installed. AppleScript and the Script Editor application from Apple come installed on all supported versions of Mac OS. This manual uses the Script Editor in examples.

As your scripts become more complex, you may find the need for debugging and productivity features not found in Script Editor. For information on those topics, see the AppleScript page at http://www.apple.com/applescript/.

Changes to AppleScript support in Adobe Illustrator CS2

- Updates to `PDF save options`, `SVG export options`, and `Flash export options` to reflect new capabilities in the corresponding dialogs.
- In earlier version, a script used the `make` command for a `raster item` to place a raster file format into a document. In this version, use the `make` command for a `placed item` instead. To embed the art in the document, use the `embed` command on the `placed item`, which converts the art to art items and deletes the `placed item` object.
- New scripting support for underline and strikethrough font styles.
- New scripting capability for converting raster art into vector art, called *tracing*. The tracing operation reorders the raster art into the source art of a plugin group, and converts it into a group of filled and/or stroked paths that resemble the original image.
  - New `trace placed` and `trace raster` commands initiate tracing, creating a new `plugin item` for the new vector art.
  - The `plugin item` property `is tracing` is `true` for the new item, and `tracing` contains a reference to a new `tracing object` object.
  - The `tracing object` property `tracingOption` references a `tracing options` object that collects the parameters used for the tracing operation. You can save tracing options to a preset file, and load previously saved tracing presets, using the `load preset` and `store preset` commands.

2 Scripting Basics

When you use Illustrator, you work with documents and their contents. You create documents, layers, colors, and design elements. You probably think of these things as objects, that you can look at and move around, and they are in fact represented by objects in the Illustrator object model. The Illustrator object model contains documents, layers, colors, and page items—objects that can appear in an Illustrator document.

Automating Illustrator with scripting uses the same object-oriented way of thinking. Each type of object has its own special properties, and the scripting language has ways to look at and change these properties.

This chapter provides a brief introduction to the basic concepts and syntax of AppleScript in Mac OS. A bibliography at the end contains references to more complete language guides.

- For more information on Illustrator's object model and specific Illustrator concepts, see "Scripting Illustrator' on page 47.
- For detailed information on the AppleScript Illustrator objects and commands, see the reference chapters later in this manual.

Object Model Concepts

In object-oriented programming, *objects* belong to *classes* and have *properties* that describe them. You manipulate the objects and their properties using *commands* in AppleScript. (In other languages, these can be called *methods* or *functions*.) What do these terms mean in this context?

Here's a way to think about objects and their properties. Imagine that you live in a house that responds to your commands (you can think of this house as technologically advanced, or magical, or both). The house is an object, and its properties might include the number of rooms, the color of the exterior paint, or the date of its construction.

Your house can also contain other objects. Similarly, the objects within the house can also contain smaller objects. Each room, for example, is an object in the house, while each window, door, or appliance is an object inside a room.

Each object can respond to various commands according to its capabilities. Windows and doors, for example, can open or close—but the floor and ceiling cannot. Using scripting, you can talk to each object directly, or you can talk to them as part of the container. You have to be very specific, though—you can't tell your house to open a window without telling it which window you want to open. So windows, like all other objects, need names or at least a numbering system so you can refer to them specifically. For example, you might say "Tell the house to open the north window of the living room."

Objects also have properties that describe specific details about them, like color and size. Imagine that the properties of objects in your house can be changed. You might say, "Door, paint yourself blue." Because your door can respond to the command "paint," you'll soon have a door of a different color.

Now let's apply this object model idea to Illustrator. The Illustrator application is the house, its documents are the rooms, and the objects in your documents are the windows and doors. You can tell Illustrator documents to add and remove objects. You can ask objects to get or change their properties.

Object classes and containers

Objects with the same properties and behaviors are grouped into classes. In the house example, windows and doors belong to their own classes, since they have unique properties, like number of panes for windows, or the door style for doors. In Illustrator, every type of graphic object—paths, text, meshes, and so on—belongs to its own class, each with its own set of properties and behaviors. Properties such as `visible` `bounds`, `width`, and `height`, for example, are common to all `page items`.

Some types of objects are containers: the house contains rooms. Similarly, a document object might contain text or image objects. There are hierarchies of containment; the house contains a room, which contains a door. In a document that contains text, the text object might further contain sentences, that contain words, that contain characters.

Object inheritance

Objects can share an overall set of properties, but specialize them in different ways. In our house example, houses have various types of openings, including doors and windows. They all share the property of being open or closed, for example, and of having hinges.

You can define a *parent* class for the most general set of shared properties in such a hierarchy. This is called a *superclass*. Each specialization can then be a child of that parent, or *subclass*. The child classes may *inherit*, or share, the properties of the parent. Windows and doors can be subclasses of an openings class, which contains properties that are common to all types of openings, such as open-state. In Illustrator, `path items`, for example, inherit geometric properties like `width` and `height` from the `page item` class.

Subclasses often have additional properties that are not shared with their superclass. In our house, both a window and door inherit an open-state property from the opening class, but a window has number-of-panes property which the opening class does not. In Illustrator, `path items`, for example, have the property `stroke color` which is not inherited from the `page item` class.

Object elements

Object *elements*, in AppleScript, are objects contained within other objects. For example, rooms are elements of our house, contained within the house object. In Illustrator, documents are elements of the application object, and page items are elements of a document object.

Object references

The objects in your documents are arranged in a hierarchy like the house object—page items are in layers, which are inside a document, which is inside Illustrator. When you send a command to an Illustrator object, you need to make sure you send the message to the right object. To do this, you identify objects by their position in the hierarchy. You might, for example, write the following statement.

```
page item 1 of layer 1 of document 1
```

When you identify an object in this fashion, you are creating an *object reference*. This gives the script a way of finding the object you want to work with.

Scripting Concepts

This section discusses how various programming concepts are dealt with in the AppleScript language.

Comments

Comments are a way to add descriptive text to a script. Comments come in handy when you want to document the operation of a script (for yourself or for someone else). The use of comments is the most important technique for good scripting. Comments are where you should leave important notes about the specific operation of a script that might provide valuable help when the script is modified at a later date. The time you save later trying to figure out what the script does may be your own. Comments are ignored by the scripting system as the script executes and cause no run-time speed penalty.

To enter a single-line comment in AppleScript, type "--" to the left of your description. For multiple line comments, start your comment with the characters "(*" and end it with "*)".

```
-- this is a single-line comment
(* this is a
multiple line comment *)
```

Long script lines

In some cases, individual script lines are too long to print on a single line in this guide.

AppleScript uses the special character (¬) to show that the line continues to the next line. This continuation character denotes a "soft return" in the script. You can enter this character in the script editor by pressing Option-Return at the end of the line you wish to continue.

Value types

Values are the data your scripts use to do their work. Most of the time, the values used in your scripts will be numbers or text. The following table shows AppleScript value types:

Value type	What it is	Example
`Boolean`	Logical true or false.	`true`
`Integer`	Whole numbers (no decimal points). Integers can be positive or negative.	`14`
`Real`	A number which may contain a decimal point.	`13.9972`
`String`	A series of text characters. Strings appear inside (straight) quotation marks.	`"I am a string"`
`List`	An ordered list of values. The values of a list may be any type.	`{10.0, 20.0, 30.0, 40.0}`
`Object reference`	A specific reference to an object.	`document 1`
`Record`	An unordered list of properties, Each property is identified by its label.	`{name: "you", index: 1}`

Variables

Variables are containers for data. A variable might contain a number, a string, a list (or array), or an object reference. Variables have names, and you refer to a variable by its name. To put data into a variable, we assign the data to the variable. The file name of the current Illustrator document or the current date are both examples of data that can be assigned to a variable.

Why not simply enter the value directly in the script rather than using a variable? When you use a value directly the flexibility of script is reduced. By using variables the scripts you write will be reusable in a wider variety of situations. As a script executes, it can assign data to the variables that reflect the state of the current document and selection, for example, and then make decisions based on the content of the variables.

In AppleScript, it is not necessary to declare your variables before assigning values to them. Assigning values to variables is fairly simple, as shown below.

```
set thisNumber to 10
set thisString to "Hello, World!"
```

Variables can also be used to store references to objects. In AppleScript, a reference is returned when you create a new object in an Illustrator document. This returned reference points to the newly created object. Storing references in variables is just the same as assigning any other value to the variable.

```
set thisLayer to make new layer at beginning of document 1
```

or you can fill the variable with a reference to an existing object:

```
set thisLayer to layer 1 of document 1
```

Variable naming

Try to use descriptive names for your variables—something like `firstPage` or `corporateLogo`, rather than `x` or `c`. While it will take a little more time to type the longer names, using them will make your scripts much easier to read. The length of a variable's name has no effect on the execution speed of your script, so use descriptive names.

You can also give your variable names a standard prefix so that they'll stand out from the objects, commands, and keywords of your scripting system.

Variable names must be a single word, but you can use internal capitalization (such as `myFirstPage`) or underscore characters (`my_first_page`) to create more readable names. Variable names cannot begin with a number, and they cannot contain punctuation or quotation marks.

Script properties

AppleScript allows you to define properties for your scripts. Script properties are much like variables, but with additional features and requirements specific to the language. The meaning and usage of script properties differs greatly between languages; consult the bibliography for appropriate language references.

Operators

Operators perform calculations (addition, subtraction, multiplication, and division) on variables or values and return a result. For example:

```
docWidth/2
```

This returns a value equal to half of the content of the variable `docWidth`. So if `docWidth` contained the number `20.5`, the value returned would be `10.25`.

You can also use operators to perform comparisons (equal to, not equal to, greater than, or less than). For example:

```
docWidth > docHeight
```

This returns the value `true` if `docWidth` is greater than `docHeight`, or `false`, if it is not.

AppleScript uses a slashed equal sign (≠) as a non-equality symbol. Use OPTION = to obtain this character.

Use the ampersand (`&`) as the concatenation operator to join two strings.

```
"Pride " & "and Prejudice."
```

This returns the string "Pride and Prejudice."

Commands

If objects are "nouns" and properties are "adjectives" in our scripting systems, then commands are the "verbs"—they are the parts of the script that make things happen. The type of the object you're working with determines which commands you can use to manipulate it.

In AppleScript, use the `make` command to create new objects, the `set` command to assign object references to variables and to change object properties, and the `get` command to retrieve objects and their properties.

Conditional statements

If you could speak to Illustrator in the course of a work session, you might say, "If the selected object is a path, then set its stroke width to 12 points." This is a conditional statement. Conditional statements make decisions—they give your scripts a way to evaluate something (the color of the selected object, or the number of color swatches in the document, or the date) and then act according to the result. Conditional statements generally start with the word `if`.

The following example checks the number of currently open documents. If no documents are open, the script displays a messages in a dialog box.

```
tell application "Adobe Illustrator"
  activate
     set documentCount to count every document
        if documentCount = 0 then
           display dialog "No Illustrator documents are open!"
        end if
end tell
```

Control structures

If you could talk to Illustrator, you might say, "Repeat the following procedure twenty times." In scripting terms, this sort of direction is called a "control structure." Control structures provide for repetitive processes, or *loops*. The idea of a loop is to repeat some action over and over again, with or without changes each time through the loop, until some condition is met.

AppleScript has a variety of different control structures to choose from. The simplest form of a loop is one that repeats some series of script operations a set number of times.

```
repeat with counter from 1 to 20
   display dialog counter
end repeat
```

A more complicated type of control structure includes conditional logic, so that it loops while or until some condition is true or false.

```
repeat while flag = false
   set flag to (button returned of display dialog "Quit?") = "Cancel"
end repeat

repeat until flag = true
   set flag to (button returned of display dialog "Quit?") = "OK"
end repeat
```

Handlers

In AppleScript, *handlers* are scripting modules you can refer to from within your script. These are sometimes called routines or subroutines. Handlers are ways to re-use parts of scripts. Typically, you send one or more values to a handler and it returns one or more values. A handler might, for example, perform conversions from one measurement system to another, or calculate the geometric center point of an object from its geometric bounds.

There's nothing special about the code used in handlers—they are simply conveniences that save you from having to type the same lines of code over and over again in your script. If you find yourself typing or pasting the same lines of code into several different places in a script, you've identified a good candidate for a handler.

This example calculates the geometric center of a selected art item. It assumes you have a single art item selected.

```
tell application "Adobe Illustrator"
-- Get the selection from the current document
        set selectedItems to selection
-- Make sure there is a selected item, and that selection is not text
        if selectedItems ≠ {} and class of selectedItems ≠ text then
-- Get the first item from the list and get it bounds
           set firstItem to item 1 of selectedItems
           set itemBounds to geometric bounds of firstItem
        end if
end tell

set itemCenter to GetItemCenter(itemBounds)
display dialog "Center x:" & item 1 of itemCenter & ", y:" & item 2 of
itemCenter
```

The following lines define the handler:

```
-- This handler finds the center of an item given its bounds
on GetItemCenter(itemBounds)
   -- Assign coordinates from the bounds to individual variables
   set {itemLeft, itemTop, itemRight, itemBottom} to itemBounds

   -- Calculate the center position
   set xCenter to (itemLeft + itemRight) / 2
   set yCenter to (itemTop + itemBottom) / 2
   return {xCenter, yCenter}
end GetItemCenter
```

Testing and Troubleshooting

The scripting environment provides tools for monitoring the progress of your script while it is running—which make it easier for you to track down any problems your script might be encountering or causing.

While the basic syntax of your script will be checked when compiled, it is possible to create and compile scripts in AppleScript that will not run properly. The Script Editor does not have extensive debugging tools, but it does have the an Event Log window.

To watch the commands your script sends and the results it receives, choose **Controls** > **Open Event Log**. The Script Editor displays the Event Log window. Check the Show Events and Show Events Results options at the top of the Event Log window and then run your script. As the script executes, you'll see the commands sent to Illustrator, and Illustrator's responses.

In addition, the Result window (choose **Controls > Show Result**) will display the value from the last script statement evaluated. Third-party editors offer additional debugging features.

About error handling

Imagine that you've written a script that formats the current text selection. What should the script do if the current selection turns out not to be text at all, but a path item? You can add *error-handling* code to your script to respond to conditions other than those you expect it to encounter.

If you have complete control over the situations in which your script will run, there's no need for you to worry about error handling. If not, however, you'll have to add some error handling capabilities to your script. The following examples show how to how you can stop a script from executing when a specific path item cannot be found.

```
--Store a reference to the fifth path item of the document in a variable
--If the object does not exist in the current document, display a message
tell application "Adobe Illustrator"
   activate
   try
         set itemCount to count of path items in current document
         set fifthItem to path item 5 of current document

   on error
         display dialog "Couldn't locate 5th path object - Only " ¬
               & itemCount & " objects."
   end try
end tell
```

AppleScript Bibliography

For further information and instruction in using the AppleScript scripting language, see these documents and resources:

- "Adobe Illustrator Scripting; with Visual Basic and AppleScript," Ethan Wilde, Peachpit Press, 2003. ISBN 0-321-11251-2. (For Illustrator 10).
- "AppleScript for the Internet: Visual QuickStart Guide," 1st ed., Ethan Wilde, Peachpit Press, 1998. ISBN 0-201-35359-8.
- "AppleScript Language Guide: English Dialect," 1st ed., Apple Computer, Inc., Addison-Wesley Publishing Co., 1993. ISBN 0-201-40735-3.
- "Danny Goodman's AppleScript Handbook," 2nd ed., Danny Goodman, iUniverse, 1998. ISBN 0-966-55141-9.
- Apple Computer, Inc. AppleScript website:

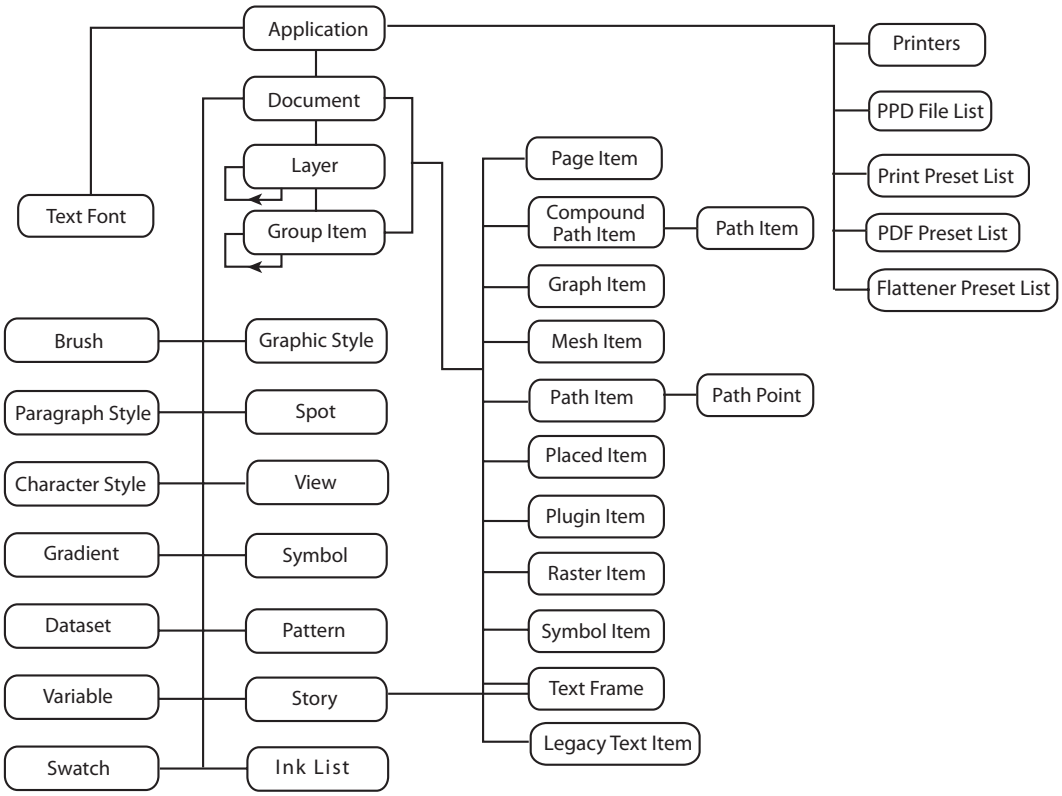
  www.apple.com/applescript

3 Scripting Illustrator

At this point, you should have a good idea of what scripting is and how it works. We are ready to begin looking at scripting Adobe Illustrator.

The Illustrator Object Model

A good understanding of the Illustrator object model will improve your scripting abilities. The following figure shows the containment hierarchy of the object model, starting with the application object.



Adobe Illustrator Scripting Object Model

Note that the `layer`, `group item`, and all text classes can contain additional objects of the same class which can in turn contain additional nested objects. For the text area of the object model, see "Working with text art" on page 53.

Looking at Illustrator objects and commands

While the objects and commands available in Illustrator are all documented in this guide, you can also view them from inside your scripting system.

➤ **To view the Illustrator AppleScript dictionary:**

1. Start Illustrator and then your script editor. Apple's Script Editor comes with all Macintosh systems. If you cannot find the Script Editor application, you must reinstall it from your Mac OS System CD.
2. In Script Editor, choose **File > Open Dictionary**. Script Editor displays an Open File dialog.
3. Find and then select the Illustrator application and click the OK button. Script Editor displays a list of the Illustrator objects and commands. You'll also be able to see the properties and elements associated with each object, as well as the parameters for each command.

Your first Illustrator script

The traditional first project in any programming language is to display the message "Hello World!" In this example, we'll create a new Illustrator document, then add a text frame containing this message.

➤ **To create an AppleScript script follow these steps:**

1. Locate and open Script Editor.
2. Enter the following script.

```
--Send the following commands to Illustrator
tell application "Adobe Illustrator"
--Create a new document with the string "Hello World"
set docRef to make new document
    set textRef to make new text frame in docRef ¬
        with properties {contents: "Hello World!", position:{200, 200}}
end tell
```

   In this script:

   - The lines preceded by "`--`" are comments, and will be ignored by the scripting system. They're included to document the operation of the script.
   - The AppleScript command `tell` indicates the object that will receive the next message we send. Every Illustrator script starts with this line.
   - Notice how the script creates, then addresses, each object in turn.

3. Run the script.

   Illustrator will create a new document, add a text frame at position (200, 200) and set the text to "Hello World!".

Adding features to "Hello World"

Next, let's create a new script that makes changes to the Illustrator document you created with your first script. Don't worry if you've closed the Illustrator document without saving it—just run your script to create a new one.

Our second script will demonstrate how to:

- Get the active document.
- Get the width of the active document.
- Resize the text frame to match the document's width.

➤ **To create the enhanced script follow these steps:**

1. Choose **File** > **New in Script Editor** to create a new script.
2. Enter the following code:

```
tell application "Adobe Illustrator"
-- current document is always the active document
   set docRef to the current document
   set docWidth to the width of docRef
-- resize the text frame to match the page width
   set width of text frame 1 of docRef to docWidth
-- alternatively, one can reference the item directly, as follows:
   set width of text frame 1 of current document to docWidth
end tell
```

3. Save the script.
4. Make sure you have the document created by the original "Hello World" script open, then run the script.

Object References

In AppleScript, Illustrator returns object references by index position or name. For example, a reference to the first path in layer 2 would be: `path item 1 of layer 2 of document` An object's index position may change when other objects are created or deleted. For example, when a new path item is created on layer 2, it will become `path item 1 of layer 2 of document 1`. This new object displaces our original path item, forcing it to index position 2. Therefore, any references made to `path item 1 of layer 2 of document 1` will refer to the new object. Consider the following sample script.

```
-- Make 2 new objects and try to select both
tell application "Adobe Illustrator"
   set newDocument to make new document
   set rectPath to make new rectangle in newDocument
   set starPath to make new star in newDocument
   set selection of newDocument to {rectPath, starPath}
end tell
```

This script does not select both the rectangle and the star, as intended; instead, it selects only the star. Try running the script with the Event Log window open to observe the references returned from Illustrator for each of the consecutive `make` commands. Notice that both commands return the same object reference: `path item 1 of layer 1 of document 1`. Therefore, the last line resolves to:

```
set selection of document 1 to {path item 1 of layer 1 of document 1, ¬
   path item 1 of layer 1 of document 1}
```

A better approach is to reference the objects by name:

```
tell application "Adobe Illustrator"
   set newDocument to make new document
   make new rectangle in newDocument with properties {name:"rectangle"}
   make new star in newDocument with properties {name:"star"}
   set selection of newDocument to ¬
      {path item "rectangle" of newDocument, ¬
       path item "star" of newDocument}
end tell
```

This example illustrates the need to uniquely identify objects. It is recommended that you assign names to objects you need to access at a later time, as there is no guarantee you are accessing the objects you expect when accessing them by index.

Some objects can be renamed; that is, the name property is writable. Objects are sorted alphabetically by name, so if you rename an object, its index changes, as do the index values of other object in the list. Again, this can cause errors if you refer to objects by index, so it is better to refer to them by name. Be careful, however, not to refer to an object by its old name after changing the name, as an object with the old name no longer exists. Object types that can renamed include:

```
brush
gradient
graphic style
pattern
swatch
symbol
variable
```

Objects that cannot be created by a script

These objects cannot be created from a script:

- Graphic styles
- Brushes
- Graphs
- Mesh art
- Plugin art
- Spirals

Object containment: document vs. layer

In Illustrator, all artwork objects are contained in layers, groups or compound paths that are themselves contained in a document. The index of an object in a layer or group indicates the object's position in the stacking order of the layer or group. This means that `page item 1 of layer 1` is the frontmost object in a document, while `page item 2 of layer 1` lies directly behind in the stacking order.

Note that if you delete all the layers in a document, the document is left with the default empty layer called `layer 1`.

When you refer to an object in your document, you can reference it directly as part of the document or by its complete containment hierarchy, including layers and any group or compound path if valid. When you refer to objects contained by the document directly, you can access the entire flattened contents of the document, without regard to the containment of objects within layers, groups, or compound paths. All objects, whether or not they are contained in groups or compound paths, are returned as individual objects contained by the document.

The following script demonstrates how to reference an object as part of a document.

```
-- Get reference for first page item of document 1
tell application "Adobe Illustrator"
   set pageItemRef to page item 1 of document 1
end tell
```

In the script below, the variable `pageItemRef` will not necessarily refer to the same object as the above script since this script includes a reference to a layer:

```
-- Get reference for first page item of layer 1 of document 1
tell application "Adobe Illustrator"
   set pageItemRef to page item 1 of layer 1 of document 1
end tell
```

Working with Document Contents

The following sections provide details of how to work with various kinds of document contents:

- "Working with selections"
- "Working with paths"
- "Working with color"
- "Working with symbols and symbol items"
- "Working with text art"

Working with selections

There are instances where you will want to write scripts that act upon the currently selected object(s). For example, you might want to have a script that applies formatting to selected text, or changes a selected path's shape. To do this, you need to know the number of selected objects and the type of each object.

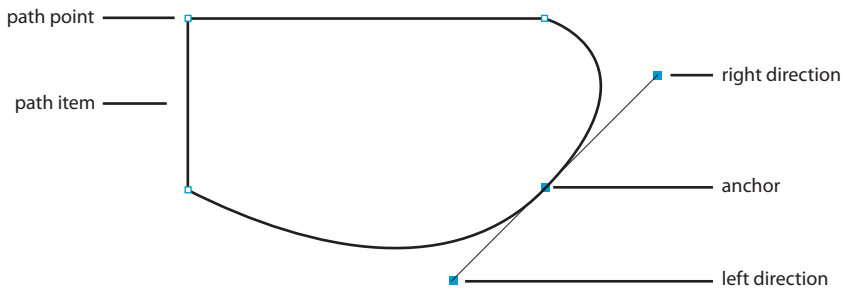
The following script demonstrates how to get and work with the current selection.

```
--selection sorter
tell application "Adobe Illustrator"
   set selectedObjects to selection
   try
      if selectedObjects is {} then
         display dialog "No objects are selected"
      else
         if class of selectedObjects = list and ¬
            (count of items in selectedObjects > 1) then
            --selection contains more than one object.
         else
            --a single object is selected. What is it?
            set selectedObjectClass to class of selectedObjects
            if selectedObjectClass = list then ¬
               set selectedObjectClass to class of item 1 of
                     selectedObjects
            if selectedObjectClass = text then
               -- text is selected
            else
               -- determine what type of object is selected.
               if selectedObjectClass = path item then
                  -- object is a path item
               else if selectedObjectClass = compound path item then
                  -- object is a compound path
               else if selectedObjectClass = raster item then
                  -- object is a raster image
               else if selectedObjectClass = placed item then
```

```
                -- object is a placed image
            else if selectedObjectClass = mesh item then
                -- object is a mesh
            else if selectedObjectClass = text frame then
                -- object is a text frame
            else if selectedObjectClass = plugin item then
                -- object is a plugin art item
            else if selectedObjectClass = path point then
                -- object is a point of a path
            else if selectedObjectClass = group item then
                -- object is a group
            end if
        end if
    end if
  end if
on error errString
  display dialog errString
end try
end tell
```

Working with paths

Path items include all artwork that contain paths, including rectangles, ellipses, polygons, as well as freeform paths. In Illustrator, every path consists of a series of points.



Path items, as well as path points, can be created and manipulated from a script. Every aspect of a path point can be accessed from scripting, including the `anchor point` and both control points, known as the `left direction` and `right direction` properties.

For more information on working with paths, Bézier curves, and path points, refer to the *Illustrator Plug-in Software Development Kit Function Reference*. This document is available as part of the Illustrator Software Development Kit (SDK), which can be downloaded from the Adobe Solutions Network (ASN) web site:

`http://partners.adobe.com/asn/developer/sdks.html`

Working with color

Swatches can be created and manipulated from your scripts. You can also create new patterns, gradients and spot colors from scripts. Just as in the user interface, percentages (0.0 through 100.0) are used to specify grayscale, individual CMYK values and spot tints. The range 0.0 to 255.0 is used for the individual RGB color values. Special attention should be paid to working with CMYK and RGB color values. Illustrator CS2 supports only a single color model within each document, either CMYK or RGB. When you specify a CMYK color value in a document that uses the RGB color model, Illustrator will convert the values

to RGB and return an RGB color, and vice-versa when specifying RGB colors in a CMYK document. However, there is some data loss during this conversion. Refer to the "Applying Color" chapter in the Adobe Illustrator User Guide for more information on working with color.

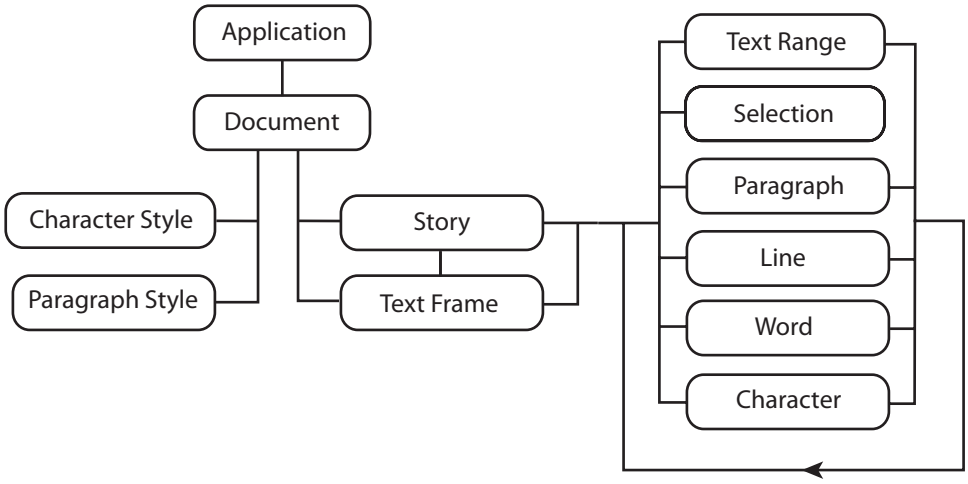
Working with symbols and symbol items

Symbols are art items that are stored in the Symbols palette and applied to documents. You can create, delete and duplicate symbols. When you create symbols, Illustrator adds them to the Symbols Palette for the target document. When you save the document, Illustrator also saves the symbols you created and used in the document.

Symbol items refer to instances of symbols in a document. You can create, delete, and duplicate symbol items. They are "linked" to the symbol definition such that changing the definition of a symbol causes all of the instances of the symbol to change as well. Symbol items are Illustrator art items and therefore can be treated as other art items or page items. In other words, you can rotate, resize, select, lock, hide and perform other operations on them.

Working with text art

The following figure shows the object model for text.

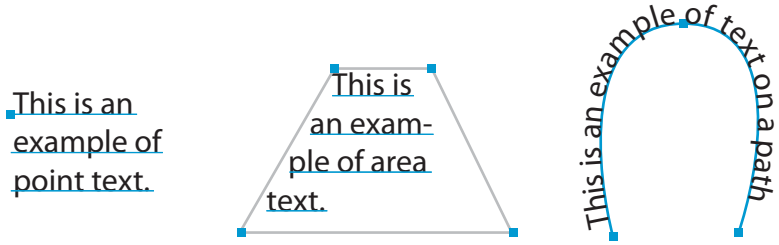


Adobe Illustrator Scripting — Object Model for Text

This text model was introduced in Illustrator CS (version 11). A `legacy text item` object allows you to import and update text from earlier versions.

With this text model, the text in an Illustrator document is contained inside a *story*. A document can have multiple stories, and each story has a *text range*. A story can contain one or more *text frames*. In this case, multiple text frames are linked together to form a single story. There are special sets of text ranges within a text range that have semantic meanings such as paragraphs, lines, words, characters.

There are three types of text frames in Adobe Illustrator: point text, path text, and area text. The `kind` property of a text frame is used to determine the type of the text frame. While all three kinds of text art have some common characteristics, such as an orientation, each kind of text art also has unique characteristics.



All three kinds of text frames have least one text path associated with them. A `text path` is not the same as a path art item, but defines the text frame's position on the artboard and its orientation (horizontal or vertical). Point text is defined completely by the properties of its text frame and associated text path.

For path and area text, text paths are associated with normal path art items. These path art items can be accessed and manipulated to modify the appearance of the associated text frame. If the text frame is path text, it will have a `text path offset` property, which indicates where on the path object the text begins.

All text art items also have at least one line of text depending on the object's geometry. A line of text is all of the characters that fit on a single line in the text frame. Text art will have multiple text lines if it contains hard line breaks or its characters flow to a new line because they do not fit in the width of the text art. Unlike characters, paragraphs and words, lines can only be created by the Illustrator application.

Refer to the "Using Type" chapter in the Adobe Illustrator User Guide for more information on working with text art.

Content of a text range

You can set the content of a text range by passing in a Unicode string known to that particular scripting language. If you know the Unicode value of a character, you can enter it using a C language style backslash escape sequence. If the string starts with a single or double quote, treat the quote as a delimiter.

Character and paragraph styles

You can change the display properties of a text range by applying an appropriate character or paragraph style or providing local overrides of character attributes at the text or paragraph levels.

The character and paragraph styles are derived from root character and paragraph styles, with all character attributes defined and set to default values. This means that there is always a valid value for any attribute for a character or paragraph in any text range.

Measurement Units

Illustrator uses points as the unit of measurement for almost all distances, where one inch is equal to 72 points. The one exception is that for values for properties such as kerning, tracking, and the *aki* properties (used for Japanese text composition), em units are used. For an explanation of em units, see “Em space units” on page 55.

Even if you change the current document ruler's units of measurement, Illustrator will still use points when communicating with your scripts. Your scripts will need to perform any unit conversions needed to

represent your measurements as points. For example, to move the current selection to a position 2 inches to the right of, and 6 inches above, its current position, you'd use the following script in AppleScript:

```
tell application "Adobe Illustrator"
(* first, manually select the text frame from the previous exercise or use
AppleScript to make the selection *)
    set selection to text frame 1 of current document
(* There are 72 points per inch. To translate an item by 2 inches to the
right and 3 inches to the left, multiply by 72 *)
      translate selection delta x (2 * 72) delta y (3 * 72)
end tell
```

If your script depends on adding, subtracting, multiplying, or dividing specific measurement values for units other than points, the script will need to convert between the units numerically. For example, to use English measurements such as inch values for coordinates or measurement units, your script will need to multiply all inch values by 72 to convert to points, since there are 72 points in an inch. To use metric measurements such as centimeters, you will need to multiply all centimeter values by 28.346, since there are 28.346 points in a centimeter.

Unit conversion to points

This table shows the conversion formulae for various units of measurement:

Unit	Conversion formula
centimeters	28.346 points = 1 centimeter
inches	72 points = 1 inch
millimeters	2.834645 points = 1 millimeter
picas	12 points = 1 pica
Qs	0.709 point = 1 Q (1 Q equals 0.23 millimeter)

Em space units

One exception to the rule of points being used for all measurements is the use of *em* units (a traditional typesetting measure) for a few properties such as for kerning and tracking. Values for these properties are measured in thousandths of an em space.

Em units are proportional to the current font size. For example, in a 6-point font, 1 em equals 6 points; in a 10-point font, 1 em equals 10 points. Similarly, a kerning value of 20 em units for a 10-point font would be equivalent to:

```
(20 units x  10 points) / 1000 units/em  =  0.2 points
```

Coordinates

Illustrator uses simple two-dimensional geometry to record the position of objects in a document. The coordinates used in Illustrator are the same as the "traditional" geometric coordinate system you learned about in school. The horizontal component of a coordinate pair (or "point") is referred to as "x" and the vertical position is denoted by "y". You can see these coordinates in the Info palette when you select or create an object in Illustrator.

Illustrator scripting uses a special class called `fixed point` to receive and return coordinate data. The fixed point is represented as a list of two items in AppleScript. The first item is the horizontal or "x" coordinate, while the second item is the vertical or "y" coordinate. The `position` of objects on a document are described with a fixed point.

Fixed points

In AppleScript, a fixed point with an x coordinate of 5.0 and a y coordinate of 10.2 is represented as a list that looks like this:

```
{5.0, 10.2}
```

Zero point

The zero point (0, 0) for coordinate numbering in Illustrator is in the lower left corner of the document. On the horizontal axis, coordinates to the right of the ruler's zero point are positive numbers, and on the vertical axis, coordinates above the zero point are positive. The `page origin` of a document defines the lower left corner of the printable region of the document as a fixed point.

Fixed rectangle

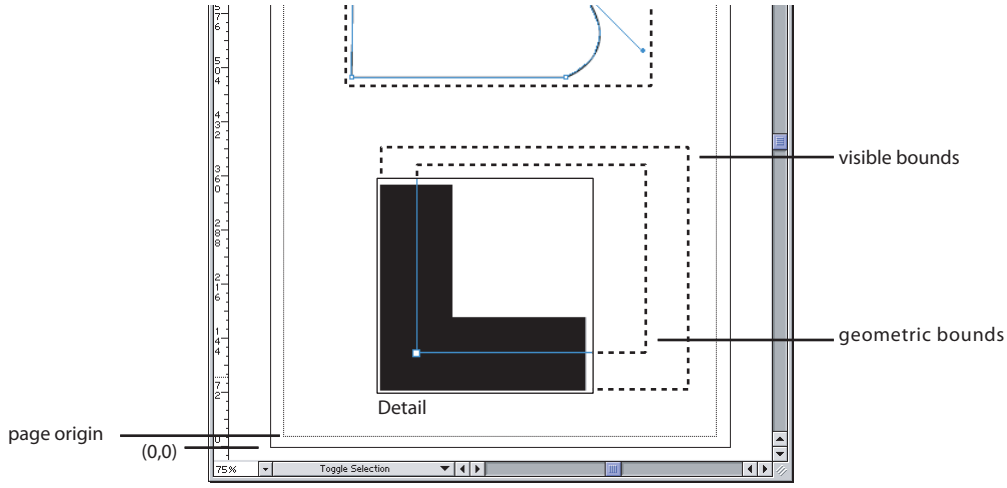
To work with rectangular coordinates where there are a pair of x and y values, Illustrator uses the special class called a `fixed rectangle`. This class consists of a list with four items in AppleScript. The coordinates of a fixed rectangle in order are: left, top, right, bottom.

In AppleScript, a fixed rectangle with a left-top corner of (5.0, 200.0) and a right-bottom corner of (100.0, 20.0) is represented by a list that looks like this:

```
{5.0, 200.0, 100.0, 20.0}
```

Page item positioning and dimensions

Every object, or page item, in a document has a position described by a fixed point and a width and height. The maximum value allowed for the width or height of a page item is 16348 points.



Every page item also has three properties that describe the object's overall extent using fixed rectangles. The `geometric bounds` of a page item are the rectangular dimensions of the object excluding stroke width. The `visible bounds` of a page item are the dimensions of the object including any stroke widths. Finally, the `control bounds` define the rectangular dimensions of the object including in- and out- control points.

Printing Illustrator Documents

You can use the full range of Illustrator CS2 print capabilities from a script. In many instances, the print features available via scripting are greater than those available through the print user interface.

Using the print scripting feature, scripters can capture and automate parts of their print workflow, which allows them to focus on other more creative work. Scripting exposes the full capabilities of Illustrator printing, some of which may not be accessible through the normal print user interface.

Illustrator supports at most one print session at any give time because of limitations in the current printing architecture.

When printing, you may provide an options parameter to more fully control the printing process. Illustrator CS2 supports an extensive list of printing options, all of which have default values. As a scripter, you can override any one of these printing options. If you override those printing options with illegal values (such as specifying a printer or paper that does not exist), an error is returned.

The following lists the categories of printing options that you can specify. Each one of these categories is optional. Within each category, default values have been provided for all properties.

Print Option	Description
Printer name	Name of the selected printer
PPD file name	File name of the selected PPD file

Print Option	Description
Print style name	Specifies the printing style
Paper options	Specifies the paper name and custom paper sizes
Print job options	Options which control things such as number of print copies
Color separation options	Specifies the separation mode, ink list, etc.
Page marks options	Controls the printing of page marks
Coordinate options	Specifies the positioning and scaling of artwork on the media
Font options	Controls the fonts used for printing
PostScript® options	Controls parameters such as the PostScript Language Level
Color management	Sets color profiles
Flattener options	Controls the transparency flattening

The print settings are determined through the following precedence order:

- The print style settings, if any are specified, override the default print settings.
- The specific printing options, if any are specified, override the print style settings.

Transformation Matrices

Thanks to the `matrix` class and the many commands that support matrices, you have access to the power of geometric transformation matrices. Transformation matrices are mathematical concepts originating in the field of linear algebra. Geometric manipulations like scaling, rotating and moving can all be described using transformation matrices.

Matrices are the basis of how Illustrator internally performs a user's request to scale, rotate or move an object. Using the command set available to create, concatenate, and apply matrices, you can transform objects in documents with programmatic precision and control. By concatenating a series of rotation, translation and scaling matrices together and applying the resulting matrix, you can perform a large series of geometric transformations in record speed. The following examples demonstrate how to combine multiple modifications in a single matrix and then apply the matrix to every object in a document.

Refer to the Illustrator Plug-in Software Development Kit Function Reference for more information on working with transformation matrices.

This script gets the identity matrix, combines it with a rotation matrix and a scale matrix and then applies the resulting matrix to all page items.

```
tell application "Adobe Illustrator"
   set transformationmatrix to get identity matrix
   set transformationmatrix to concatenate rotation matrix ¬
      transformationmatrix angle 45.0
   set transformationmatrix to concatenate scale matrix ¬
      transformationmatrix horizontal scale 60
   transform every page item of document 1 using transformationmatrix
end tell
```

A matrix object in Illustrator consists of six properties. In AppleScript, these properties are:

```
mvalue_a
mvalue_b
mvalue_c
mvalue_d
mvalue_tx
mvalue_ty
```

By experimenting with the matrix concatenation commands, you can discover how to construct matrices that can be applied to perform movement (also called translation), rotation, scaling, skewing and other transformations. See the script examples for the matrix commands for working samples.

Working with Variables and Datasets

By creating dynamic objects, you can create data-driven graphics. You can define dynamic objects by using variables. In scripting, the `variable` class corresponds to these variables. Variables are document-level objects; therefore, you create them in the document object. You can add and delete variables to and from a script by using the `make` and `delete` commands.

Datasets are closely related to variables in that a dataset collects variables and their associated dynamic data into a single object. The `dataset` class is the object that corresponds to an `AI DataSet`. The `dataset` collection in the `document` class provides methods so you can create, update and delete datasets.

Launching and Quitting Illustrator from a Script

Your scripts can control the activation and quitting of the Illustrator application.

Use the `activate` and `quit` commands to control application's run state. The `activate` command brings the Illustrator application to the front if it is not already the frontmost application. Note that if the clipboard contains data at the time of quitting, Illustrator may show a dialog asking if the data on the clipboard should be saved for other applications. You can avoid this dialog by first clearing the clipboard with the statement:

```
set the clipboard to {}
```

User Interaction Levels

An application usually presents a dialog when it needs to provide feedback or request information. This is called user interaction, and is useful and expected when you are directly interacting with the application. On the other hand, when a script is interacting with an application, an unexpected dialog will bring the execution of the script to a halt until the dialog is dismissed. This can be a serious problem in an automation environment where there is typically no one present to deal with dialogs.

The Illustrator CS2 application class contains a user interaction level property. By setting this property a script can control the level of interaction allowed during script execution. All interaction is normally suppressed in an automation environment, and some interaction might be useful where scripts are being used in a more interactive fashion.

There are four possible values for the user interaction level property in AppleScript:

Property Value	Result
`never interact`	No interaction is allowed
`interact with self`	Interact only with scripts executed from the scripts menu
`interact with local`	Interact with script executed on the local machine (including self)
`interact with all`	Interact with all scripts

Using AppleScript, it is possible to send commands from one machine to another. The four possible values allow you to control interaction based on the source of the script commands. For example, if the application is acting as a server for remote users, it would be difficult for a remote user to dismiss a dialog, but it would be no problem for someone sitting in front of the machine. In this case, an interaction level of *'interact with local'* would prevent dialogs from halting remote scripts but would allow dialogs to be presented for local scripts.

4 AppleScript Objects

This chapter provides a complete reference for the objects and commands in the Illustrator AppleScript dictionary. The objects are presented alphabetically. For each object, the following information is provided:

- Elements that can be contained within the object.
- Properties of the object, with read-only status, value type, and a description.
- Valid commands, with links to "AppleScript Commands" on page 245, which describes all of the commands in the Illustrator dictionary.
- Notes to explain special issues.
- Script examples. Note that these example are intended to illustrate concepts, and do not necessarily represent the best or most efficient way to construct an AppleScript script. They contain little error checking, and assume that the proper context exists for the scripts to execute in (for instance, that there is a document open or items selected).

For an overview of the Illustrator object model, see "The Illustrator Object Model" on page 47.

application

The Adobe Illustrator application object, which contains all other Illustrator objects.

application elements

Elements	Refer to by
`document`	name, numeric index, range of elements, before/after another element, satisfying a test
`text font`	numeric index, range of elements, before/after another element, satisfying a test

application object properties

Property	Value type	What it is
`best type`	`type class`	Read-only. The best type for the application object's value; always returns `reference`.
`browser available`	`boolean`	Read-only. If `true`, a web browser is available.
`class`	`type class`	Read-only. The object's class, which is `application`.
`current document`	`document`	The active (frontmost) document in Illustrator.
`default type`	`type class`	Read-only. The default type for the application object's value; always returns `reference`.
`flattener presets`	`list of Unicode text`	Read-only. The list of flattener style names currently available for use.
`free memory`	`integer`	Read-only. The amount of unused memory (in bytes) within the Adobe Illustrator partition.
`frontmost`	`boolean`	Read-only. If `true`, this is the frontmost (active) application.
`name`	`Unicode text`	Read-only. The application's name (not related to the filename of the application file); always returns "`Adobe Illustrator CS2`".
`PDF presets`	`list of Unicode text`	Read-only. The list of preset PDF-options names available for use.
`PPDs`	`list of PPD files`	Read-only. The list of PPD files currently available for use. (A document must be open or an error is returned).
`print presets`	`list of Unicode text`	Read-only. The list of preset printing-options names available for use.
`printers`	`list of printers`	Read-only. The list of installed printers currently available for use. (A document must be open or an error is returned).

Property	Value type	What it is
properties	record	All of the application's properties returned in a single record. Properties that are individually read-only remain so in this record.
scripting version	Unicode text	Read-only. The version of the Scripting plug-in.
selection	anything	All of the currently selected objects in the active (frontmost) document. Illustrator does not support the `select` command to change the application's current selection. Use `set the selection to` in place of `select`. See the examples below. The application's `selection` can be accessed and modified. When there are no selected objects, `selection` contains an empty list, `{}`. To deselect all objects in the current document, set `selection` to an empty list. When there is an active insertion point in the contents of a text frame, `selection` returns a reference to the insertion point. When characters are selected in the contents of a text frame, `selection` returns a reference to the range of text.
settings	Illustrator preferences	Read-only. Preferences for the Illustrator application.
tracing presets	list of Unicode text	Read-only. The list of preset tracing-options names available for use.

user interaction level	Valid values:
	`interact with all`
	`interact with local`
	`interact with self`
	`never interact`
		The level of interaction with the user that is allowed when handling script commands. Default: `interact with all`

version	Unicode text	Read-only. The version of the Adobe Illustrator application.

application commands

➤ Select an object

```
-- Select the first object in the document
tell application "Adobe Illustrator"
```

```
-- Make sure there is a page item to select
   if (document 1 exists) and (page item 1 of document 1 exists) then
      set the selection to page item 1 of document 1
   end if
end tell
```

➤ Copy and paste a selection

You do not need to make objects part of the selection to act on them. Selection is most useful for moving objects to and from the clipboard using the `cut`, `copy` and `paste` commands, which act on the current selection. The clipboard can be used effectively for moving data between applications that do not share common object classes.

Note that Illustrator must be the frontmost application when executing any command that deals with the clipboard. This example brings Illustrator to the front using AppleScript's `activate` command.

```
-- Copy current selection to clipboard then paste into a new doc
tell application "Adobe Illustrator"

-- If Illustrator is not the frontmost application, activate it.
  if not frontmost then activate
-- Make sure there is a document to copy from
  if (count documents) > 0 then
    set selectedItems to selection of current document
    if selectedItems is not {} then
      copy
      set colorSpace to color space of current document
      make new document with properties {color space:colorSpace}
      paste
    end if
  end if
end tell
```

brush, brushes

A brush or list of brushes. Brushes are contained in `document` objects. Scripts cannot create new brushes.

brush object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the `brush` object's value; always returns `reference`.
class	type class	Read-only. The object's class, which is `brush`.
container	object reference	Read-only. A reference to the document that contains this `brush`.
default type	type class	Read-only. The default type for the `brush` object, which is `reference`.
index	integer	Read-only. The index of this `brush`.
name	Unicode text	The name of this `brush`.
properties	record	All of the properties of this object returned as a record.

brush object commands

apply
count
exists

➤ Applying brushes

This example demonstrates how to apply a series of brushes to objects.

```
-- Duplicate the current selection (if it is a single item)
-- and apply each available brush to the new object
tell application "Adobe Illustrator" to
   set selectedItems to selection

-- Check for selection of single non-text object
if class of selectedItems is text or (count items of selectedItems)¬
   is not 1 then
      display dialog "Select a single path item before running this script"
else
   tell application "Adobe Illustrator"

      set pathItem to item 1 of selectedItems

      -- Get the item's position and use it to tile the new items below
      set {itemX, itemY} to position of pathItem

      -- Get a list of all brushes and apply each brush to the selected item
      set brushList to every brush of current document

      -- Get coordinates of upper-left of document
      set docLeft to 0
```

```
        set docTop to height of current document

        set brushCount to count items of brushList
        repeat with i from 1 to brushCount

          set aBrush to item i of brushList
          set itemOffset to i * 20 -- use to tile the duplicated items

          -- Duplicate the selected path item, tiling them from the
          -- upper-left of the document
          set pathRef to duplicate pathItem to beginning of current document¬
            with properties {position:{docLeft+itemOffset, docTop-itemOffset}}

          -- Must clear the document's selection before applying a brush
          -- since the duplicate above seems to add to it each time through
          set selection of current document to {}

          apply aBrush to pathRef

        end repeat
      end tell
    end if
```

character

Specifies the properties of a character. The text contained within text frames in Illustrator can be accessed using the `character`, `insertion point`, `word`, `line`, `paragraph`, and `text` classes. The properties and valid commands for all of these classes are similar, but not identical. For example, while `character` has a `kerning` property, the other text classes do not.

character object elements

Elements	Refer to by
`character style`	name, numeric index, range of elements, before/after another element, satisfying a test
`character`	numeric index, range of elements, before/after another element, satisfying a test
`insertion point`	numeric index, range of elements, before/after another element, satisfying a test
`line`	numeric index, range of elements, before/after another element, satisfying a test
`paragraph style`	name, numeric index, range of elements, before/after another element, satisfying a test
`paragraph`	numeric index, range of elements, before/after another element, satisfying a test
`text`	numeric index, range of elements, before/after another element, satisfying a test
`word`	numeric index, range of elements, before/after another element, satisfying a test

character object properties

Property	Value Type	What it is
`aki left`	`real`	The amount of inter-glyph space added to the left side of the glyph in Japanese text (in thousandths of an em).
`aki right`	`real`	The amount of inter-glyph spacing added to the right side of the glyph in Japanese text (in thousandths of an em).

`alignment`	Valid values:
	`bottom`
	`center`
	`icf bottom`
	`icf top`
	`roman baseline`
	`top`
		The character alignment type.

Property	Value Type	What it is

alternate glyphs	Valid values:
	`default`
	`expert`
	`full width`
	`half width`
	`jis78`
	`jis83`
	`proportional width`
	`quarter width`
	`third width`
	`traditional`
		Specifies which kind of alternate glyphs to use.

auto leading	`boolean`	If `true`, use automatic leading.

baseline direction	Valid values:
	`standard`
	`Tate Chu Yoko`
	`vertical rotated`
		The Japanese text baseline direction.

baseline position	Valid values:
	`normal`
	`subscript`
	`superscript`
		The baseline position of text.

baseline shift	`real`	The amount of shift (in points) of the text baseline.
best type	`type class`	Read-only. The best type for the object's value.

capitalization	Valid values:
	`all caps`
	`all small caps`
	`normal`
	`small caps`
		Specifies whether the text is normal, all uppercase, all small caps, or a mix of small caps and lowercase.

character offset	`integer`	Offset of the first character.
class	`type class`	Read-only. The object's class.
connection forms	`boolean`	If `true`, use the OpenType® connection forms.
container	`reference`	Read-only. The object' s container.
contents	`Unicode text`	The text content of the text range.
contextual ligature	`boolean`	If `true`, use the contextual ligature.
default type	`type class`	Read-only. The default type for the object's value.
discretionary ligature	`boolean`	If `true`, use the discretionary ligature.

Property	Value Type	What it is

figure style	Valid values:
	`default`
	`proportional`
	`proportional oldstyle`
	`tabular`
	`tabular oldstyle`
		Specifies the figure style to use in an OpenType font.

fill color	`color info`	The color of the text fill.
fractions	`boolean`	If `true`, use OpenType fractions.
horizontal scale	`real`	The horizontal scaling factor for the character.
index	`integer`	Read-only. The index of this instance of the object.
italics	`boolean`	If `true`, the Japanese OpenType supports italics.
kerning	`integer`	Controls the spacing between two characters, in thousandths of an em space.

kerning method	Valid values:
	`auto`
	`none`
	`optical`
		The type of automatic kerning method to use.

Property	Value Type	What it is
language	Valid values: Bokmal Norwegian Brazillian Portuguese Bulgarian Canadian French Catalan Chinese Czech Danish Dutch English Finnish Greek Hungarian Icelandic Italian Japanese Nynorsk Norwegian old German Polish Romanian Russian Spanish Serbian standard French standard German standard Portuguese Swedish Swiss German Turkish UK English Ukranian	The language.
leading	real	The amount of space between two lines of text, in points.
length	integer	Length of text range in characters. Minimum: 0
ligature	boolean	If true, use the ligature.
no break	boolean	If true, no break is allowed.
ordinals	boolean	If true, use the OpenType ordinals.
ornaments	boolean	If true, use the OpenType ornaments.
overprint fill	boolean	If true, overprint the fill of the text.
overprint stroke	boolean	If true, overprinting of the stroke of the text is allowed.
properties	record	All of the properties of this object returned as a record.
proportional metrics	boolean	If true, Japanese OpenType supports proportional fonts.

Property	Value Type	What it is
rotation	real	The character rotation angle in degrees.
selection	text *or* list of texts	Read-only. The selected text ranges within the text range.
size	real	The font size in points.
story	story	Read-only. The story of the text range.
strike through	boolean	If true, characters use strike-through style.
stroke color	color info	The color of the text stroke.
stroke weight	real	Line width of stroke.
stylistic alternates	boolean	If true, use OpenType stylistic alternates.
swash	boolean	If true, use the OpenType swash character.
TCY horizontal	integer	The Tate-Chu-Yoko horizontal adjustment in points.
TCY vertical	integer	The Tate-Chu-Yoko vertical adjustment in points.
text font	text font	The text font.
titling	boolean	If true, use the OpenType titling alternates.
tracking	integer	The tracking or range kerning amount in thousandths of an em.
Tsume	real	The percentage of space reduction around a Japanese character.
underline	boolean	If true, characters use underline style.
vertical scale	real	Character vertical scaling factor, expressed as a percentage (100 is 100%).
warichu characters after break	integer	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.
warichu characters before break	integer	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.
warichu enabled	boolean	If true, Wari-Chu is enabled.

Property	Value Type	What it is
warichu gap	integer	The Wari-Chu line gap in points.
warichu justification	Valid values: auto justify center full justify last line center full justify full justify last line left full justify last line right left right	The Wari-Chu justification.
warichu lines	integer	The number of Wari-Chu (multiple text lines fit into a space meant for one) lines.
warichu scale	real	The Wari-Chu scale.

character object commands

➤ Make selected text superscript

In this example, the `selection` property has all of the properties that `character` or any other text class would have.

```
-- Make the currently selected text superscript
tell application "Adobe Illustrator"
   -- Make sure one or more characters of text are selected
   set selectedText to selection of current document
   if class of selectedText is text or ¬
      class of selectedText is character then
      -- Adjust the properties of the selected text to superscript it
      set fontSize to size of selectedText
      set fontBaseline to baseline shift of selectedText
      set properties of selectedText to ¬
         {size:fontSize / 2, baseline shift:fontBaseline + (fontSize / 2)}
   end if
end tell
```

➤ Stretch characters

This example demonstrates how to use character properties to create unique effects from a script.

```
(* Distort every character in the first text frame of a document
by incrementally modifying the horizontal scaling of each character
```

```
to give the effect of stretching words out *)

-- A smaller value here means more difference between largest and
-- smallest horizontal scaling of the characters
property pVariability : 1.0

tell application "Adobe Illustrator"
   -- Is there is a document and a text frame to work with
   if (exists text frame 1 of current document) then
      -- Make sure the text frame contains some text
      set textFrame to first text frame of current document
      if textFrame is not "" then -- contains some text
         -- Gather info needed to calculate the scale factor
         set characterCount to count characters in textFrame
         set factor to (characterCount + 1) / 2
         -- Iterate over each character, changing its horizontal scale
         repeat with i from 1 to characterCount
            set hScaling to (factor - i) / factor
            if hScaling < 0 then set hScaling to -hScaling
            set widthScale to 100 + 100 * hScaling
            set horizontal scale of character i of text frame 1 ¬
               of document 1 to widthScale
         end repeat
      end if
   end if
end tell
```

character style, character styles

A named style that specifies character attributes.

Note: Character attributes do not have default values, and are undefined until explicitly set.

character style object properties

Property	Value Type	What it is
aki left	real	The left aki (in thousandths of an em).
aki right	real	The right aki (in thousandths of an em).
alignment	Valid values: bottom center icf bottom icf top roman baseline top	The character alignment type.
alternate glyphs	Valid values: default expert full width half width jis78 jis83 proportional width quarter width third width traditional	The alternate glyphs form.
alternate ligature	boolean	If true, use the alternate ligature.
auto leading	boolean	If true, use automatic leading.
baseline direction	Valid values: standard Tate Chu Yoko vertical rotated	The Japanese text baseline direction.
baseline position	Valid values: normal subscript superscript	The baseline position of text.
baseline shift	real	The amount of shift (in points) of the text baseline.
best type	type class	Read-only. The best type for the object's value.

Property	Value Type	What it is

capitalization	Valid values:
	`all caps`
	`all small caps`
	`normal`
	`small caps`
		The case of the text.

class	`type class`	Read-only. The object's class.
connection forms	`boolean`	If `true`, use the OpenType connection forms.
contextual ligature	`boolean`	If `true`, use the contextual ligature.
container	`reference`	Read-only. The object's container.
default type	`type class`	Read-only. The default type for the object's value.
discretionary ligature	`boolean`	If `true`, use the discretionary ligature.

figure style	Valid values:
	`default`
	`proportional`
	`proportional oldstyle`
	`tabular`
	`tabular oldstyle`
		Specifies which figure style to use in the OpenType font.

fill color	`color info`	The color of the text fill.
font	`font`	The text font.
fractions	`boolean`	If `true`, use the OpenType fractions.
horizontal scale	`real`	Character horizontal scaling factor expressed as a percentage (100 = 100%).
index	`integer`	Read-only. The index of this instance of the object.
italics	`boolean`	If `true`, the Japanese OpenType supports italics.

kerning method	Valid values:
	`auto`
	`none`
	`optical`
		The automatic kerning method to use.

Property	Value Type	What it is
language	Valid values: Bokmal Norwegian Brazillian Portuguese Bulgarian Canadian French Catalan Chinese Czech Danish Dutch English Finnish Greek Hungarian Icelandic Italian Japanese Nynorsk Norwegian old German Polish Romanian Russian Spanish Serbian standard French standard German standard Portuguese Swedish Swiss German Turkish UK English Ukranian	The language.
leading	real	The amount of space between two lines of text, in points.
ligature	boolean	If true, use the ligature.
name	Unicode text	The character style's name.
OpenType position	Valid values: default denominator numerator subscript superscript	The OpenType font baseline position.
ordinals	boolean	If true, use the OpenType ordinals.
ornaments	boolean	If true, use the OpenType ornaments.
overprint fill	boolean	If true, the fill of the text should be overprinted.
overprint stroke	boolean	If true, the stroke of the text should be overprinted.
properties	record	All of the properties of this object returned as a record.

Property	Value Type	What it is
proportional metrics	boolean	If true, the Japanese OpenType font supports proportional glyphs.
rotation	real	The character rotation angle in degrees.
size	real	The font size in points.
strike through	boolean	If true, characters use strike-through style.
stroke color	color info	The color of the text stroke.
stroke weight	real	The line width of the stroke.
stylistic alternates	boolean	If true, use the OpenType stylistic alternates.
swash	boolean	If true, use the OpenType swash glyph.
TCY horizontal	integer	The Tate-Chu-Yoko horizontal adjustment in points.
TCY vertical	integer	The Tate-Chu-Yoko vertical adjustment in points.
titling	boolean	If true, use the OpenType titling alternates.
tracking	integer	The tracking or range kerning amount in thousands of an em.
Tsume	real	The percentage of space reduction around a Japanese character (100 = 100%).
underline	boolean	If true, characters use underline style.
vertical scale	real	The character vertical scaling factor expressed as a percentage (100 = 100%).
warichu characters after break	integer	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.
warichu characters before break	integer	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.
warichu enabled	boolean	If true, Wari-Chu is enabled.
warichu gap	integer	The Wari-Chu line gap.

Property	Value Type	What it is
warichu justification	Valid values: auto justify center full justify last line center full justify full justify last line left full justify last line right left right	The Wari-Chu justification.
warichu lines	integer	The number of Wari-Chu (multiple text lines fit into a space meant for one) lines.
warichu scale	real	The Wari-Chu scale.

➤ Character styles

This example makes a new document and three text frames with different names, contents, and positions. It creates a character style with color, tracking, size, and capitalization information. Then, for each character in each text frame, it applies the character style.

```
-- Character Styles

tell application "Adobe Illustrator"
   activate
   make new document
   make new text frame in document 1 with properties ¬
      {name:"text 1", contents:"Scripting is fun!", position:{50, 100}}
   make new text frame in document 1 with properties ¬
      {name:"text 2", contents:"Scripting is easy!", position:{100, 200}}
   make new text frame in document 1 with properties ¬
      {name:"text 3", contents:"Everyone can script!", position:{150, 300}}
   make new character style in document 1 with properties {name:"Big Red"}
   set the size of character style "Big Red" of document 1 to 40
   set the tracking of character style "Big Red" of document 1 to -50
   set the capitalization of character style "Big Red" of document 1 to ¬
      all caps
   set the fill color of character style "Big Red" of document 1 to ¬
      {class:RGB color info, red:255, green:0, blue:0}
   -- 'apply character style' is the event.
   -- 'style character style "Big Red" of document 1' is the style applied.
   -- note that character styles must be applied to text ranges.
   apply character style character style "Big Red" of document 1 to ¬
      the text range of text frame "text 1" of document 1
   apply character style character style "Big Red" of document 1 to ¬
      the text range of text frame "text 2" of document 1
   apply character style character style "Big Red" of document 1 to ¬
      the text range of text frame "text 3" of document 1

end tell
```

CMYK color info

A CMYK color specification, used to specify a CMYK color where a `color info` object is required. This class contains the color component values of a CMYK color. Use it to specify and get color information from an Illustrator document or page items.

If the `color space` of a document is RGB and you specify the color value for a page item in that document using `CMYK color info`, Illustrator translates the CMYK color specification into an RGB color specification. The same thing happens if the document's color space is CMYK and you specify colors using `RGB color info`. Since this translation can cause information loss you should specify colors using the color info class that matches the document's color space.

CMYK color info object properties

Note: This class inherits all properties from the `color info` class.

Property	Value type	What it is
cyan	real	The cyan color value. Range: 0.0 to 100.0 Default: 0.0
magenta	real	The magenta color value. Range: 0.0 to 100.0 Default: 0.0
yellow	real	The yellow color value. Range: 0.0 to 100.0 Default: 0.0
black	real	The black color value. Range: 0.0 to 100.0 Default: 0.0

➤ Assign color info

This example demonstrates how to create a new swatch in a document and assign a CMYK color to the swatch.

```
-- Make a new CYMK color swatch in the current document
tell application "Adobe Illustrator"
  if not (exists swatch "Our CMYK Swatch" in current document) then

     set swatchColor to {cyan:50.0, magenta:20.0, yellow:20.0, black:0.0}
     make new swatch at end of current document with properties ¬
        {name:"Our CMYK Swatch", color:swatchColor}

  end if
end tell
```

color info

An abstract parent class for all color classes used in Illustrator. Subclasses are

- CMYK color info
- gradient color info
- gray color info
- Lab color info
- no color info
- pattern color info
- RGB color info
- spot color info

color management options

Specifies the color management options when printing a document with the print command.

color management options object properties

Property	Value Type	What it is

intent	Valid values:
	`absolute colorimetric`
	`perceptual`
	`relative colorimetric`
	`saturation`
		The color management intent type. Default: `relative colorimetric`

name	`Unicode text`	The color management profile name.

profile kind	Valid values:
	`custom profile`
	`oldstyle profile`
	`printer profile`
	`source profile`
		The color management profile mode. Default: `source profile`

color separation options

Print color separation options when printing a document with the print command.

color separation options object properties

Property	Value Type	What it is
convert spot colors	boolean	If true, all spot colors are converted to process colors. Default: false
inks	list of ink	The list of inks for color separation.
over print black	boolean	If true, black is overprinted. Default: false
separation mode	Valid values: composite InRIP separation host based separation	The color separation type. Default: composite

compound path item, compound path items

A compound path or list of compound paths. Compound paths are objects that contain two or more paths that are painted so that holes appear where paths overlap.

All paths in a compound path share property values. Therefore, if you set the value of a property of any one of the paths in the compound path, all other path's matching property will be updated to the new value.

Paths contained within a compound path or group in a document are returned as individual paths when a script asks for the paths contained in the document. However, paths contained in a compound path or group are not returned when a script asks for the paths in a layer which contains the compound path or group.

compound path item object elements

Element	Refer to by
`path item`	name, numeric index, range of elements, before/after another element, satisfying a test

compound path item object properties

Note: This object class inherits all properties from the `page item` class.

Property	Value type	What it is
`properties`	`record`	All of the properties of this object returned as a record.

compound path item object commands

`count`
`delete`
`duplicate`
`exists`
`make`
`move`
`rotate`
`scale`
`transform`
`translate`

➤ Get paths

This handler demonstrates how to get a list containing all of the paths in a document that are not part of a compound path or a group by iterating through each layer in the document.

```
-- Get paths in a doc that are not part of a compound path or group

to GetPathItemsOfDocument(docRef)
   tell application "Adobe Illustrator"

      set pathItemList to {}
      set layerCount to count layers of docRef
```

```
      repeat with i from 1 to layerCount
         set pathItemList to pathItemList & every path item of ¬
            layer i of docRef
      end repeat
   end tell
   return pathItemList
end GetPathItemsOfDocument

-- Call handler
tell application "Adobe Illustrator" to set docRef to current document
set allPathItems to GetPathItemsOfDocument(docRef)
```

➤ Duplicate and group paths from a compound path

Compound paths contain path items that can be accessed from a script. This example shows how to duplicate the paths in a compound path and then group them in a new group item.

```
-- Create a group containing a set of paths duplicated from the
-- first compound path item of the document
tell application "Adobe Illustrator"
   set pathItemList to every path item of compound path item 1 ¬
      of current document
   set groupRef to make new group item at beginning of layer 1 ¬
      of document 1
   duplicate pathItemList to beginning of groupRef
end tell
```

coordinate options

The print coordinate options when printing a document with the print command.

coordinate options object properties

Property	Value Type	What it is
emulsion	boolean	If true, flip the artwork horizontally. Default: false
fit to page	boolean	If true, proportionally scale the artwork to fit on media. Default: false
horizontal scale	real	The horizontal scaling factor. 100.0 = 100% Range: 1.0 to 10000.0; Default: 100.0
orientation	Valid values: landscape portrait reverse landscape reverse portrait	The artwork orientation. Default: portrait
position	Valid values: bottom bottom left bottom right center left right top top left top right	The artwork position on media. Default: center
tiling	Valid values: full pages imageable areas single full page	The page tiling mode. Default: single full page
vertical scale	real	The vertical scaling factor. 100.0 = 100% Range: 1.0 to 10000.0; Default: 100.0

dataset, datasets

An object, or list of objects, that contains variables and their dynamic data.

dataset object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the dataset's value; always returns reference.
class	type class	Read-only. The object's class, which is dataset.
container	object reference	Read-only. A reference to the art item that contains this data set.
default type	type class	Read-only. The default type for the data set; always returns reference.
index	integer	Read-only. The index of this data set in the art item.
name	Unicode text	The name of the dataset.
properties	record	All of the properties of this object returned as a record.

dataset object commands

count
delete
display
exists
make
update

➤ Datasets and variables

```
-- Datasets and Variables
--
-- Activate Illustrator
-- Make a new document
-- Make two variables, one of kind visibility and the other textual
-- Make a rectangle and a text frame, and attach the respective variables
-- Set the color of the rectangle and the contents of the text frame
-- Make the first dataset
-- Change the contents of the text and the visibility of the rectangle
-- Make the second dataset
-- display the two datasets
--

tell application "Adobe Illustrator"
   activate
   make new document
   make new variable in document 1 with properties ¬
      {name:"RecVariable", kind:visibility}
   make new variable in document 1 with properties ¬
      {name:"TextVariable", kind:textual}
```

```
    make new rectangle in document 1 with properties ¬
       {name:"Rec1", position:{100, 500}, ¬
        visibility variable:variable "RecVariable" of document 1}
    make new text frame in document 1 with properties ¬
       {name:"Text1", position:{100, 550}, ¬
        content variable:variable "TextVariable" of document 1}
    set the fill color of page item "Rec1" of document 1 to ¬
       {class:RGB color info, red:150, green:255, blue:255}
    set the contents of text frame "Text1" of document 1 ¬
       to "Now you see me..."
    make new dataset in document 1 with properties ¬
       {name:"My First Dataset"}
    set hidden of page item "Rec1" of document 1 to true
    set the contents of text frame "Text1" of document 1 ¬
       to "Now you don't!"
    make new dataset in document 1 with properties ¬
       {name:"My Second Dataset"}
    repeat 3 times
       delay 1
       display dataset "My First Dataset" of document 1
       delay 1
       display dataset "My Second Dataset" of document 1
    end repeat
end tell
```

document, documents

An Illustrator document or a list of documents. Documents are contained in the `application` object.

The default document settings—those properties starting with the word "default"—are global settings that affect the current document. Be sure to modify these default properties only when a document is open. Note that if you set default properties to desired values before creating new objects, you can streamline your scripts, eliminating the need to specify properties such as `fill color` and `stroked` that have analogous default properties.

A document's `color space`, `height`, and `width` can only be set when the document is created. Once a document is created, these properties cannot be changed.

The frontmost document can be referred to as either `current document` or `document 1`.

document object elements

Element	Refer to by
`brush`	name, index, before/after, range, test
`character style`	name, index, before/after, range, test
`compound path item`	name, index, before/after, range, test
`dataset`	name, index, before/after, range, test
`gradient`	name, index, before/after, range, test
`graph item`	name, index, before/after, range, test
`graphic style`	name, index, before/after, range, test
`group item`	name, index, before/after, range, test
`layer`	name, index, before/after, range, test
`legacy text item`	name, index, before/after, range, test
`mesh item`	name, index, before/after, range, test
`page item`	name, index, before/after, range, test
`paragraph style`	name, index, before/after, range, test
`path item`	name, index, before/after, range, test
`pattern`	name, index, before/after, range, test
`placed item`	name, index, before/after, range, test
`plugin item`	name, index, before/after, range, test
`raster item`	name, index, before/after, range, test
`spot`	name, index, before/after, range, test
`story`	index, before/after, range, test
`swatch`	name, index, before/after, range, test

Element	Refer to by
symbol	name, index, before/after, range, test
symbol item	name, index, before/after, range, test
tag	name, index, before/after, range, test
text frame	name, index, before/after, range, test
variable	name, index, before/after, range, test
view	index, before/after, range, test

document object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the document object's value; always returns reference.
class	type class	Read-only. The object's class, which is document.
color space	Valid values: RGB CMYK	Read-only. The color specification system to use for this document's color space.
crop marks	rectangle	The boundary of the document's cropping box for output.
crop style	Valid values: standard Japanese style	The style of the document's cropping box.
current dataset	dataset	The currently active dataset.
current layer	layer	The active layer in the document.
current view	view	Read-only. The document's current view.
default fill color	color info	The color to fill new paths if default filled is true.
default fill overprint	boolean	If true, the art beneath a filled object should be overprinted by default.
default filled	boolean	If true, a new path should be filled.
default stroke cap	Valid values: butted rounded projecting	Default type of line capping for paths created.
default stroke color	color info	The stroke color for new paths if default stroked is true.
default stroke dash offset	real	The default distance into the dash pattern at which the pattern should be started for new paths.

Property	Value type	What it is
default stroke dashes	list of real numbers	Default lengths for dashes and gaps in dashed lines, starting with the first dash length, followed by the first gap length, and so on. Set to an empty list, {}, for a solid line.
default stroke join	Valid values: mitered rounded beveled	Default type of joints in new paths.
default stroke miter limit	real	When a default stroke join is set to mitered, this property specifies when the join will be converted to beveled (squared-off) by default. The default miter limit of 4 means that when the length of the point reaches four times the stroke weight, the join switches from a miter join to a bevel join. Values: 1 to 500. 1 specifies a bevel join.
default stroke overprint	boolean	If true, the art beneath a stroked object should be overprinted by default.
default stroke width	real	Default width of stroke for new paths.
default stroked	boolean	If true, new paths should be stroked.
default type	type class	Read-only. The default type for the document object's value; always returns reference.
file path	file specification	Read-only. The file associated with the document, which includes the complete path to the file.
geometric bounds	rectangle	Read-only. The object's bounds excluding the stroke width.
height	real	Read-only. The height of the document, calculated from the geometric bounds.
index	integer	Read-only. The position of this document in the stacking order of all open documents. The current (frontmost) document is always document 1.
inks	list of inks	Read-only. The list of inks in this document.
Kinsoku set	list of Unicode text	Read-only. The Kinsoku set of characters that cannot begin or end a line of Japanese text.
modified	boolean	If true, the document has been modified since the last save.
Mojikumi set	list of Unicode text	Read-only. A list of names of predefined Mojikumi sets which specify the spacing for the layout and composition of Japanese text.
name	Unicode text	Read-only. The document's name (not the complete file path to the document).

Property	Value type	What it is
output resolution	real	Read-only. The current output resolution for the document in dots per inch (dpi).
page origin	list	The zero-point of the page in the document without margins, relative to the overall height and width.
print tiles	boolean	Read-only. If true, this document should print as tiled output.
properties	record	All of the document's properties returned in a single record. Properties that are individually read-only remain so in this record.
ruler origin	list	The zero-point of the rulers in the document relative to the bottom left of the document.
ruler units	Valid values: unknown inches centimeters points picas millimeters qs	Read-only. The default units for the rulers in the document.
selection	list of object references	The list of references to the objects in this document's current selection.
show placed images	boolean	Read-only. If true, the placed images should be displayed in the document.
split long paths	boolean	Read-only. If true, long paths should be split when printing.
stationery	boolean	Read-only. If true, the document should be saved as a stationery file.
tile full pages	boolean	Read-only. If true, full pages should be tiled when printing this document.
use default screen	boolean	Read-only. If true, use the printer's default screen when printing this document.
variables locked	boolean	If true, the variables are locked.
visible bounds	rectangle	Read-only. The object's visible bounds, including stroke width of any objects in the illustration.
width	real	Read-only. The width of this document, calculated from the geometric bounds.

document object commands

close
count
delete

duplicate
exists
export
export PDF preset
export variables
get
import character styles
import PDF preset
import print preset
import variables
make
open
print
save

➤ Making sure a document is open

The following example shows how to make sure a document is open before setting any of the application's default properties.

```
-- Check to make sure a document is open in Illustrator
-- before setting the application's default stroke width to 8 points
tell application "Adobe Illustrator"
   if not (document 1 exists) then
      make new document with properties ¬
         {color space: CMYK, width: 100.0, height: 50.0}
   end if
   set the default stroke width of document 1 to 8.0
end tell
```

➤ Making a new document

This example shows how to make new documents with custom defaults.

```
-- Present a dialog to the user to choose a new document type
-- from, then create a new document with its properties set accordingly
-- Note: You can only change writable document defaults when document is
-- open

-- Prompt user for new document properties from list of choices
set listChoice to (choose from list
   {"CMYK, filled, 2 pt stroke with dashes", ¬
      "RGB, filled, no stroke", ¬
      "RGB, no fill, 1 pt stroke"} ¬
      with prompt "What kind of new document to create?")

if listChoice is not false then

   -- Gather the values needed to set the document's properties
   set documentType to item 1 of listChoice

   set fillPaths to (documentType contains "filled")
   set strokePaths to (documentType contains "pt stroke")

   set strokeWidth to 0.0
   if documentType contains "1 pt" then
      set strokeWidth to 1.0
```

```
   else if documentType contains "2 pt" then
      set strokeWidth to 2.0
   end if

   if documentType contains "with dashes" then
      set strokeDashes to {2.5, 1, 2.5, 1, 2.5, 1}
   else
      set strokeDashes to {}
   end if

   tell application "Adobe Illustrator"

      -- Create a document with the requested color space
      if documentType starts with "CMYK" then
         set docRef to make new document with properties {color space:CMYK}
      else
         set docRef to make new document with properties {color space:RGB}
      end if

      -- Set the document's properties with one command
      set properties of docRef to
         {default filled:fillPaths
            , default stroked:strokePaths
            , default stroke width:strokeWidth
            , default stroke dashes:strokeDashes}
   end tell
end if
```

➤ Getting the file path of a document

This example demonstrates how to use document properties in other applications. In this case, the script uses the `file path` property of the active document to open the folder containing the Illustrator document in the Finder.

```
-- Reveal and select a documents file icon in the Finder
tell application "Adobe Illustrator"
   set filePath to file path of current document
end tell

tell application "Finder"
   activate
   reveal filePath
end tell
```

ellipse

Used to create an elliptical path in an Illustrator document. This object is available only in the context of a `make` command, which creates an instance of the `path item` class. This special class allows you to quickly create complex path items. Properties associated with `path items`, such as `fill color` and `note`, can also be specified at the time of creation.

ellipse object properties

Property	Value type	What it is
bounds	`list of points`	Write-once. The bounds of the ellipse.
inscribed	`boolean`	Write-once. If `true`, the ellipse path should be inscribed (drawn inside the rectangle described by the bounds).
reversed	`boolean`	Write-once. If `true`, the ellipse path reversed.

ellipse object commands

`make`

➤ Creating ellipses

This example demonstrates how to create a series of ellipses based on the geometry of a single selected object.

```
-- Embellish a single selected path item by adding a bright red
-- ellipse to each point on the path

property pEllipseScale : 0.1

tell application "Adobe Illustrator"
   activate
   set selectedItems to selection

   -- A bit of sanity checking
   if (count selectedItems) is not 1 ¬
      or class of selectedItems is text ¬
      or class of item 1 of selectedItems is not path item then

      display dialog "Select a single path item before running this script"
   else
      set pathItem to item 1 of selectedItems

      -- Set ellipse color based on document color space
      set docColorSpace to color space of current document
      if docColorSpace is RGB then
         set ellipseColor to {red:255.0, green:0.0, blue:0.0}
      else
         set ellipseColor to ¬
            {cyan:0.0, magenta:100.0, yellow:100.0, black:0.0}
      end if
```

```
        -- Gather needed info about the path item to be embellished
        set itemWidth to width of pathItem
        set itemHeight to height of pathItem
        set pathPointList to anchor of every path point of pathItem

        -- Calculate the position and bounds for each ellipse
        repeat with aPoint in pathPointList
           set {x, y} to aPoint

           set rectLeft to x - (itemWidth * pEllipseScale)
           set rectRight to x + (itemWidth * pEllipseScale)
           set rectTop to y + (itemHeight * pEllipseScale)
           set rectBottom to y - (itemHeight * pEllipseScale)

           set ellipseRect to {rectLeft, rectTop, rectRight, rectBottom}

           make new ellipse at beginning of current document ¬
              with properties {bounds:ellipseRect, inscribed:true, ¬
                 reversed:false, stroke color:ellipseColor, ¬
                 fill color:ellipseColor}
        end repeat
     end if
  end tell
```

EPS save options

Options that can be supplied when saving a document as an Illustrator EPS file. See the save command for additional details.

This class is used to define a record containing properties that specify options when saving a document as an EPS file. `EPS save options` can only be used in conjunction with the `save` command. It is not possible to get or create an `EPS save options` object.

EPS save options object properties

Property	Value type	What it is
`CMYK PostScript`	`boolean`	If `true`, the file should be saved as `CMYK PostScript`. Default: `false`

`compatibility`	Valid values:
	`Illustrator 3`
	`Illustrator 8`
	`Illustrator 9`
	`Illustrator 10`
	`Illustrator CS`
	`Illustrator CS2`
	`Japanese 3`
		Specifies the version of the Illustrator file format to create. Default: `Illustrator CS2`

`compatible gradient printing`	`boolean`	If `true`, create a raster item of the gradient or gradient mesh so that PostScript Level 2 printers can print the object.
`embed all fonts`	`boolean`	If `true`, fonts used in the EPS file should be embedded in the file (version 7 or later). Default: `false`
`embed linked files`	`boolean`	If `true`, linked image files are to be included in the saved document. Default: `false`

`flatten output`	Valid values:
	`preserve paths`
	`preserve appearance`
		How transparency should be flattened for file formats before Illustrator 9. Default: `preserve appearance`

`included document thumbnails`	`boolean`	If `true`, the thumbnail image of the EPS artwork should be included. Default: `true`

`overprint`	Valid values:
	`discarded`
	`preserved`
		Default value is preserved.

Property	Value type	What it is

PostScript	Valid values:
	`level 2`
	`level 3`
		Specifies the PostScript level to use when saving the file (level 1 is valid for file format version 8 or older). Default: `level 3`

preview	Valid values:
	`none`
	`BW Macintosh`
	`color Macintosh`
	`BW TIFF`
	`color TIFF`
	`transparent color TIFF`
		Specifies the format for the EPS preview image. Default: `color Macintosh`

➤ Saving EPS files

This handler processes a folder of Illustrator files, saving each as an EPS file with level 2 PostScript and Illustrator CS2 compatibility. The files are save to the folder specified in the `destinationFolder` parameter. Note that the `class` property is specified in the record to ensure that Illustrator can determine the save option class.

```
-- fileList is a list of aliases to Illustrator files
-- destFolder is an alias to a folder where the EPS files are to be saved
-- sourceFolder is an alias to a folder containing more than one AI file
set sourceFolder to choose folder with prompt "Source folder?"
tell application "Finder" to ¬
   set fileList to every file of folder sourceFolder as alias list
set destFolder to choose folder with prompt "Destination folder?"
set destPath to destFolder as string

repeat with aFile in fileList

   tell application "Finder" to set fileName to name of aFile

   set newFilePath to destPath & fileName & ".EPS"
   tell application "Adobe Illustrator"
      open aFile
      save current document in file newFilePath as eps ¬
         with options {class:EPS save options, ¬
          compatibility:Illustrator 9, ¬
          preview:color Macintosh, ¬
          embed linked files:true, ¬
          include document thumbnails:true, ¬
          embed all text fonts:true, ¬
          CMYK PostScript:true, ¬
          PostScript:level 2}
      close current document saving no
   end tell
end repeat
```

Flash export options

You can supply a number of options when exporting a document as Macromedia® Flash™ (SWF). See the export command in the command reference for additional details.

This class is used to define a record containing properties that specify options when exporting a document as a Flash (SWF) file. `Flash export options` can only be supplied in conjunction with the `export` command. It is not possible to get or create a `Flash export options` object.

Flash export options object properties

Property	Value type	What it is
artboard clipping	boolean	If true, the resulting image should be clipped to the artboard. Default: false
background color	RGB color info	The background color.
background layers	list of layers	Layers to be included as the static background in all exported Flash frames.
blend animation	Valid values: in build in sequence none	How the blend art objects are animated when exported to Flash frames. Default: none
compressed	boolean	If true, the exported file should be compressed. Default: false
convert text to outlines	boolean	If true, all text should be converted to outlines. Default: false
curve quality	integer	How much curve information should be preserved Range: 0 to 10, Default: 7
export style	Valid values: Flash file layers to files layers to frames	How the Flash file should be created Default: Flash file
flatten output	Valid values: preserve paths preserve appearance	How transparency should be flattened to preserve appearance or editability on export. Default: preserve appearance
frame rate	real	When exporting layers to Flash frames Range: 0.01 to 12.0, Default: 12.0
generate HTML	boolean	If true, export as an HTML file. Default: true
image format	Valid values: lossless lossy	How the images in the exported file should be compressed. Default: lossless
JPEG method	Valid values: optimized standard	Specifies which method to use. Default: standard

Property	Value type	What it is
JPEG quality	integer	Level of compression. Range: 0 to 10, Default: 3
layer order	Valid values: bottom up top down	The order in which layers should be exported to Flash frames. Default: bottom up
looping	boolean	If true, the Flash file should be set to loop when run. Default: false
read only	boolean	If true, export as read only file. Default: false
replacing	Valid values: yes no ask	If a file with the same name already exists, should it be replaced. Default: ask
resolution	real	Pixels per inch. Range: 72 to 2400, Default: 72

flattening options

Specifies transparency flattening options when printing a document with the print command. These options are used to output artwork that contains transparency into a non-native format.

flattening options object properties

Property	Value Type	What it is
clip complex regions	boolean	If true, complex regions are clipped. Default: false
convert strokes to outlines	boolean	If true, all strokes are converted to outlines. Default: false
convert text to outlines	boolean	If true, all text items are converted to outlines. Default: false
flattening balance	integer	The flattening balance. Range: 0 to 100; Default: 100
gradient resolution	real	The gradient resolution in dots per inch. Range: 1.0 to 9600.0, Default: 300.0
overprint	Valid values: discard preserve	Overprint choice. Default: preserve
rasterization resolution	real	The rasterization resolution in dots per inch. Range: 1.0 to 9600.0, Default: 300.0

➤ **Flattening options**

```
-- Flattener Options
--
-- Activate Illustrator
-- Create a variable that holds the flattening options
-- Create a variable that holds the print options
-- Print the document
-- A document and a printer must be present
--

tell application "Adobe Illustrator"
   activate
   set flatOpts to ¬
      {class:flattening options, clip complex regions:true, ¬
       gradient resolution:360, rasterization resolution:360}
   set printOpts to {class:print options, flattener settings:flatOpts}
   if not (exists document 1) then ¬
      error "There is no document available to print."
   print document 1 options printOpts
end tell
```

font options

Font options when printing a document with the print command.

font options object properties

Property	Value Type	What it is
download fonts	Valid values: complete none subset	The font download mode. Default: subset
font substitution kind	Valid values: device substitution oblique substitution tint substitution	The font substitution policy. Default: oblique substitution

➤ Applying fonts

```
-- Text Fonts
-- Make a new document, count the number of fonts
-- For each font, make a new text frame with the name of the font
-- and that font applied to it
-- Position the text frames into three columns

tell application "Adobe Illustrator"
   activate
   make new document
   set fontCount to count text fonts
   set x to 1
   set startX to 10
   set startY to (height of document 1) - 10
   repeat with i from 1 to fontCount
      make new text frame in document 1
      set fontName to name of text font i
      try
         if fontName is not name of text font (i + 1) then
            set posX to startX
            set posY to startY - (x * 15)
            make new text frame in document 1 with properties¬
                {contents:fontName, position:{posX, posY}}
            set the text font of text of the result to text font i
            set x to x + 1
            if x > 50 then
               set startX to startX + 220
               set startY to (height of document 1) - 10
               set x to 1
            end if
         end if
      end try
   end repeat
end tell
```

GIF export options

Options that can be supplied when exporting a document as a GIF file. See the export command in the command reference for additional details.

This class is used to define a record containing properties that specify options when exporting a document as a GIF file. `GIF export options` can only be supplied in conjunction with the `export` command. It is not possible to get or create a `GIF export options` object.

GIF export options object properties

Property	Value type	What it is
antialiasing	`boolean`	If `true`, the resulting image should be anti-aliased. Default: `true`
artboard clipping	`boolean`	If `true`, the resulting image should be clipped to the artboard. Default: `false`
color count	`integer`	The number of colors in the exported color table. Range: 2 to 256. Default: `128`

color dither	Valid values:
	`none`
	`diffusion`
	`pattern dither`
	`noise`
		The method used to dither colors. Default: `diffusion`

color reduction	Valid values:
	`selective`
	`adaptive`
	`perceptual`
	`web`
		The method used to reduce the number of colors in the document. Default: `selective`

dither percent	`integer`	How much the colors should be dithered. Range: 0 to 100. Default: 88
horizontal scaling	`real`	The horizontal scaling factor to apply to the resulting image. Range: 0.0 to 100.0. Default: 100.0
information loss	`integer`	The level of information loss during compression (as a percentage). Range: 0 to 100. Default: 0
interlaced	`boolean`	If `true`, the resulting image should be interlaced. Default: `false`
matte	`boolean`	If `true`, the artboard should be matted with a color. Default: `true`
matte color	`RGB color info`	The color to use when matting the artboard. Default: `white`
saving as HTML	`boolean`	If `true`, the resulting image is saved with an accompanying HTML file. Default: `false`
transparency	`boolean`	If `true`, the resulting image uses transparency. Default: `true`

Property	Value type	What it is
vertical scaling	real	The vertical scaling factor to apply to the resulting image. Range: 0.0 to 100.0. Default: 100.0
web snap	integer	How much the color table should be changed to match the Web pallet. Range: 0 to 100, where 100 is the maximum change. Default: 0

➤ Exporting to GIF

This handler processes all Illustrator files in a specific folder, exporting each as a scaled GIF image. Note that the `class` property is specified in the record to ensure that Illustrator can determine the export option class.

```
-- fileList is assumed to be a list of aliases to Illustrator files
-- destinationFolder is assumed to be an alias to a folder where the
-- GIF files are to be exported
on ExportFilesAsGIF(fileList, destinationFolder)
   set destinationPath to destinationFolder as string
   repeat with aFile in fileList
      tell application "Finder" to set fileName to name of aFile
      set newFilePath to destinationPath & fileName & ".gif"
      tell application "Adobe Illustrator"
         open aFile
         export current document to file newFilePath as GIF with options¬
            {class:GIF export options, color count:256,¬
               color reduction:adaptive, information loss:0, ¬
               color dither:none, dither percent:100, ¬
               web snap:0, transparency:false, interlaced:false, ¬
               matte:true, matte color:{red:128, green:0, blue:60}, ¬
               horizontal scaling:50.0, vertical scaling:50.0, ¬
               antialiasing:true, artboard clipping:false, ¬
               saving as HTML:false}
         close current document saving no
      end tell
   end repeat
end ExportFilesAsGIF

-- Call handler
set sourceFolder to choose folder with prompt "Source folder?"
tell application "Finder" to ¬
   set fileList to every file of folder sourceFolder as alias list
   set destinationFolder to choose folder with prompt "Destination folder?"
   ExportFilesasGIF(fileList, destinationFolder)
```

gradient, gradients

A gradient definition or gradient definitions. Gradients are contained in documents. Scripts can create new gradients.

gradient object elements

Element	Refer to by
`gradient stop`	index, before/after, range, test

gradient object properties

Property	Value type	What it is
`best type`	`type class`	Read-only. The best type for the `gradient` object's value; always returns `reference`.
`class`	`type class`	Read-only. The object's class, which is `gradient`.
`container`	`object reference`	Read-only. A reference to the document that contains this gradient.
`default type`	`type class`	Read-only. The default type for the `gradient` object's value; always returns `reference`.
`entire gradient`	`list (of gradient stop info)`	All of the gradient stops in the `gradient`.

`gradient type`	Valid values:
	`linear`
	`radial`
		The type of the `gradient`.

`index`	`integer`	Read-only. The position of this `gradient` in the application.
`name`	`Unicode text`	The `gradient`'s name.
`properties`	`record`	All of the properties of this object returned as a record.

gradient object commands

count
delete
duplicate
exists
make

➤ Creating a gradient

This example shows how to create a linear RGB gradient.

```
-- Create a new RGB gradient with three gradient stops
property pGradientName : "RGB Hot Streak"
tell application "Adobe Illustrator"
   if not (exists gradient pGradientName in current document) then
      set newgradient to make new gradient at beginning of current document¬
```

```
            with properties {name:pGradientName, gradient type:linear}
         -- Since all new gradients are created with 2 gradient stops,
         -- create another stop for the 3-stop gradient
         make new gradient stop at beginning of newgradient
         set properties of gradient stop 1 of newgradient to ¬
            {midpoint:50.0, ramp point:0.0,¬
             color:{red:255.0, green:255.0, blue:0.0}}
         set properties of gradient stop 2 of newgradient to ¬
            {midpoint:50.0, ramp point:50.0, ¬
             color:{red:255.0, green:127.0, blue:127.0}}
         set properties of gradient stop 3 of newgradient to ¬
            {midpoint:50.0, ramp point:100.0, ¬
             color:{red:255.0, green:0.0, blue:0.0}}
      end if
   end tell
```

gradient color info

A gradient color specification, used to specify the color component values of a gradient color swatch. It is used for specifying and retrieving color information from an Illustrator document or from page items in a document.

gradient color info object properties

Note: This class inherits all properties from the color info class.

Property	Value type	What it is
angle	real	The gradient vector angle (in degrees). Default: 0.0
gradient	object reference	A reference to the gradient object that defines the gradient to use in this color definition.
hilite angle	real	The gradient highlight vector angle in degrees. Default: 0.0
hilite length	real	The gradient highlight vector length. Default: 0.0
length	real	The gradient vector length.
matrix	matrix	An additional transformation matrix to manipulate the gradient path.
origin	fixed point	The gradient vector origin.

➤ Using gradient information

This example demonstrates how to set a path item's fill color to a gradient color.

```
-- Set fill color of the first path in the current document
-- to the first gradient in the document
tell application "Adobe Illustrator"
   set the fill color of path item 1 of document 1 to ¬
      {gradient:gradient 1 of document 1}
end tell
```

gradient stop, gradient stops

A gradient stop definition or definitions contained in a specific gradient. A gradient stop is a point on a specific gradient that specifies a color change in the containing gradient.

gradient stop object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the gradient stop object's value; always returns reference.
class	type class	Read-only. The object's class, which is gradient stop.
color	color info	The color linked to this gradient stop.
container	object reference	Read-only. A reference to the gradient that contains this gradient stop.
default type	type class	Read-only. The default type for the gradient stop object's value; always returns reference.
index	integer	Read-only. The position of this gradient stop in the gradient.
midpoint	real	The midpoint of the blend between this stop's and the next stop's colors. Range: 13.0 to 87.0
properties	record	All of the properties of this object returned as a record.
ramp point	real	The location of the color in the gradient. Range: 0.0 to 100.0

gradient stop object commands

count
delete
duplicate
exists
make

➤ Reversing colors in a gradient

This example demonstrates how to reverse the colors in a gradient by getting, then switching, the colors of the contained gradient stops.

```
-- This handler reverses the colors in gradient identified
-- by the gradientRef parameter
on ReverseGradientColors(gradientRef)
  tell application "Adobe Illustrator"
     -- get a list of the gradient's colors
     set colorList to color of every gradient stop of gradientRef
     -- tell AppleScript to reverse the order of the list
     set colorList to reverse of colorList
     -- iterate over the gradient resetting its colors
     set colorCount to count items in colorList
     repeat with i from 1 to colorCount
        set color of gradient stop i of gradientRef to (item i of colorList)
     end repeat
```

```
  end tell
end ReverseGradientColors

-- call handler
tell application "Adobe Illustrator" to set gradientRef to ¬
  gradient 1 of document 1
ReverseGradientColors(gradientRef)
```

gradient stop info

Gradient stop information of a specific gradient, returned by the `entire gradient` property of a gradient.

The gradient stops for a new gradient can be specified by providing a list of `gradient stop info` records in the `entire gradient` property. The following applies when creating a gradient from a list of gradient stop info records:

- A gradient stop's location in the gradient is determined by its `ramp point` value, not the `gradient stop info` record's order in the entire gradient list.
- The `midpoint` value of the last `gradient stop info` record in the entire gradient list is not used for the newly created gradient and need not be provided. If it is present, its value must be in the valid range.

gradient stop info object properties

Property	Value type	What it is
`color`	`color info`	The color linked to this gradient stop.
`midpoint`	`real`	The midpoint of the blend between this stop's and the next stop's colors. Range: 13.0 to 87.0. Default: 50.0
`ramp point`	`real`	The location of the color in the gradient as a percentage. Range: 0.0 to 100.0. Default: 0.0

➤ Using gradient stop information

This example shows how to create a circular CMYK gradient using a list of gradient stop info records.

```
-- Create a new CMYK gradient with 4 gradient stops
property pGradientName : "CMYK Circle"
tell application "Adobe Illustrator"
   if not (exists gradient pGradientName in current document) then
      set entireGradient to {{midpoint:50.0, ramp point:0.0,¬
         color:{cyan:0.0, magenta:0.0, yellow:0.0, black:100.0}},¬
         {midpoint:50.0, ramp point:33.3, ¬
          color:{cyan:0.0, magenta:0.0, yellow:100.0, black:0.0}},¬
         {midpoint:50.0, ramp point:66.7, ¬
          color:{cyan:0.0, magenta:100.0, yellow:0.0, black:0.0}},¬
         {midpoint:50.0, ramp point:100.0, ¬
          color:{cyan:100.0, magenta:0.0, yellow:0.0, black:0.0}}}
      set gradientRef to make new gradient in current document ¬
         with properties {name:pGradientName, gradient type:radial, ¬
            entire gradient:entireGradient}
   end if
end tell
```

graph item, graph items

A graph or a list of graphs.

graph item object properties

Note: This object class inherits all properties from the `page item` class.

Property	Value type	What it is
`content variable`	`anything`	The content variable to which this `graph item` is bound It is not necessary to set the type of the `content variable` before binding. Illustrator automatically sets the type to `graph`.
`properties`	`record`	All of the properties of this object returned as a record.

graph item object commands

count
delete
duplicate
exists
move
rotate
scale
transform
translate

➤ Rotating graph items

```
-- Graph Items
-- Make sure a document is available
-- Get every page item whose class is graph item
-- For each graph item, rotate it 90 degrees

tell application "Adobe Illustrator"
   activate
   if not (exists document 1) then error "There is no available document."
   set graphItems to every page item of document 1 whose class is graph item
   if graphItems is {} then error "The doc does not contain any graph items."
   repeat with currentGraphItem in graphItems
      rotate currentGraphItem angle 90
   end repeat
end tell
```

graphic style, graphic styles

Defines a set of appearance attributes that you can apply non-destructively to page items. Graphic styles are contained in documents. The graphic styles can be accessed from a script, but cannot be created from a script. You cannot delete default `graphic styles`.

graphic style object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the `graphic style` object's value; always returns `reference`.
class	type class	Read-only. The object's class, which is `graphic style`.
container	object reference	Read-only. A reference to the document that contains this `graphic style`.
default type	type class	Read-only. The default type for the `graphic style` object, which is `reference`.
index	integer	Read-only. The index of this `graphic style`.
name	Unicode text	The name of this `graphic style`.
properties	record	All of the properties of this object returned as a record.

graphic style object commands

apply
count
delete
exists

➤ Applying a graphic style

```
-- Duplicate and group the selected path items, then apply
-- a user-selected graphic style to the items in the new group
tell application "Adobe Illustrator" to
   set selectedItems to selection of document 1

-- Check for empty selection
if selectedItems is not {} then
   tell application "Adobe Illustrator"
      -- Create the new group to contain the duplicated items
      set groupRef to make new group item at document 1
      -- Duplicate the selected items to the new group
      set newItemList to duplicate selectedItems to beginning of groupRef
      -- Get graphic style names for display in the choice list
      set styleNames to name of every graphic style of document 1
   end tell

   -- Present dialog and let user choose the style to apply
   set styleName to (choose from list styleNames
      with prompt "Style for selection?") as string
   if styleName is not "" then
```

```
        tell application "Adobe Illustrator"
           (* The chosen graphic style is applied to the list
              of items returned by the duplicate command,
              rather than to the new group itself, because the
              apply command works on individual path items,
              not groups of items *)
           apply graphic style styleName of current document to newItemList
        end tell
     end if
  end if
```

gray color info

A grayscale color specification, used to specify a gray color where a `color info` object is required.

This class is used to define a record which contains the tint value of a gray color. It is used for specifying and retrieving color information from an Illustrator document or from page items in a document.

gray color info object properties

Note: This class inherits all properties from the `color info` class.

Property	Value type	What it is
`gray value`	`real`	The tint of the gray. Range: 0.0 (white) to 100.0 (black) Default: 0.0

➤ Creating a gray color swatch

This example demonstrates how to create a gray color swatch.

```
-- Create a new gray color swatch (35% black) in the current document
property pSwatchName : "35% Gray Swatch"
tell application "Adobe Illustrator"
   if not (exists swatch pSwatchName in current document) then
      make new swatch at beginning of current document with properties ¬
         {name:pSwatchName, color:{gray value:35.0}}
   end if
end tell
```

group item, group items

A grouped set of art items. Group items can contain all of the same page items that a layer can contain, including other nested groups.

Paths contained within a group or compound path in a document are returned as individual paths when a script asks for the paths contained in the document. However, paths contained in a group or compound path are not returned when a script asks for the paths in a layer which contains the group or compound path.

A new group can be created that contains the contents of a vector art file if you provide a file specification to the vector file (EPS or PDF) in the `with data` parameter of the `make` command. The resulting group will be the same object as if the user had placed the file from the user interface using the **File > Place**… command with the embed checkbox checked.

group item object elements

Element	Refer to by
`compound path item`	name, index, before/after, range, test
`graph item`	name, index, before/after, range, test
`group item`	name, index, before/after, range, test
`legacy text item`	name, index, before/after, range, test
`mesh item`	name, index, before/after, range, test
`page item`	name, index, before/after, range, test
`path item`	name, index, before/after, range, test
`placed item`	name, index, before/after, range, test
`plugin item`	name, index, before/after, range, test
`raster item`	name, index, before/after, range, test
`symbol item`	name, index, before/after, range, test
`text frame`	name, index, before/after, range, test

group item object properties

Note: This class inherits all properties from the `page item` class.

Property	Value type	What it is
`clipped`	`boolean`	If `true`, the `group item` is clipped to its first path item.
`properties`	`record`	All of the properties of this object returned as a record.

group item object commands

`count`

delete
duplicate
exists
make
move
rotate
scale
transform
translate

➤ Group contents of a vector art file

Create a new group item from the contents of a vector art file, either EPS or PDF.

```
-- Create new group whose contents will be the contents of a vector art file
-- fileRef is an alias or file reference to the vector file to be placed
on EmbedVectorFile(fileRef)
   tell application "Adobe Illustrator"
      set groupRef to make new group item in document 1 ¬
         with data fileRef with properties {position:{0, 600}}
   end tell
   return groupRef
end EmbedVectorFile

-- Call handler
set fileRef to choose file with prompt "Select vector file to place"
set groupRef to EmbedVectorFile(fileRef)
```

➤ Create path items from a group

New groups can be easily created and populated with objects. This example demonstrates how path items can be created in a container group.

```
-- Create a new group, then add rectangles to it using
-- the available placement options
tell application "Adobe Illustrator"
   set groupRef to make new group item in document 1
   set rectRef to make new rectangle at beginning of groupRef ¬
      with properties {bounds:{150, 550, 350, 350}, fill color:{blue:255}}
   make new rectangle after rectRef with properties ¬
      {bounds:{100, 600, 300, 400}, fill color:{red:255}}
   set rectRef to make new rectangle at end of groupRef with properties ¬
      {bounds:{0, 700, 200, 500}, fill color:{green:255}}
   make new rectangle before rectRef with properties ¬
      {bounds:{50, 650, 250, 450}, fill color:{black:100}}
end tell
```

➤ Select items not in a group

This example demonstrates how to select all of the page items in a document that are not part of a group by testing the `container` property of all items with a `whose` clause.

```
-- Select only the page items in a document that are not part of
-- a group and that are not themselves groups
tell application "Adobe Illustrator"
   -- First deselect everything in the document
   set selection of current document to {}
   if (count page items of current document) > 0 then
```

```
      set layerCount to count layers in current document
      repeat with i from 1 to layerCount
         set layerRef to layer i of current document
         if (count page items of layer i of current document) > 0 then
            set selected of (every page item of current document ¬
               whose container is layerRef ¬
               and class is not group item) to true
         end if
      end repeat
   end if
end tell
```

➤ Making a clipping mask

This example shows how to create a clipping mask using the first path item in a group item. This is the same effect as you get when you use the **Object > Clipping Mask > Make** command in the user interface.

```
-- Create group of paths, then clip group to the first path in the group
tell application "Adobe Illustrator"
   -- Create a group to contain the paths to be clipped
   set groupRef to make new group item in document 1
   -- Add some path items to the group
   make new rectangle at end of groupRef with properties ¬
      {bounds:{200, 350, 300, 250}, fill color:{cyan:100}, stroked:false}
   make new rectangle at end of groupRef with properties ¬
      {bounds:{300, 250, 400, 150}, fill color:{magenta:100}, stroked:false}
   make new rectangle at end of groupRef with properties ¬
      {bounds:{300, 350, 400, 250}, fill color:{yellow:100}, stroked:false}
   make new rectangle at end of groupRef with properties ¬
      {bounds:{200, 250, 300, 150}, fill color:{green:255}, stroked:false}

   -- Get a little fancy and create a rotated star at the center of the group
   set pathRef to make new star at beginning of groupRef with properties¬
      {center point:{300, 250}, radius:25, inner radius:4, point count:4, ¬
       fill color:{black:100}, opacity:40, stroked:false}
   set rotationMatrix to get rotation matrix angle 45
   transform pathRef using rotationMatrix about center

   -- Create the path that the group will be clipped with
   -- The clipping path must be the first (frontmost) path in the group
   make new star at beginning of groupRef with properties ¬
      {center point:{300, 250}, radius:80, inner radius:25, ¬
       point count:4, stroked:false, filled:false}
   -- Now clip the group to the top path
   set clipped of groupRef to true
end tell
```

Illustrator preferences

Contains Adobe PDF and Adobe Photoshop® file preferences for Illustrator.

Illustrator preferences object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the object's value.
class type	type class	Read-only. The object's class.
default type	type class	Read-only. The default type for the object's value.
properties	record	All of the properties of this object returned as a record.
PDF file options	PDF options	Read-only. Options to use when opening or placing a PDF file.
Photoshop file options	Photoshop options	Read-only. Options to use when opening or placing a Photoshop file.

Illustrator save options

Options that may be supplied when saving a document as an Illustrator file. See the save command for additional details.

This class is used to define a record containing properties used to specify options when saving a document as an Illustrator file. `Illustrator save options` can only be supplied in conjunction with the `save` command. It is not possible to get or create an `Illustrator save options` object.

Illustrator save options object properties

Property	Value type	What it is

compatibility	Valid values:
	`Illustrator 3`
	`Illustrator 8`
	`Illustrator 9`
	`Illustrator 10`
	`Illustrator CS`
	`Illustrator CS2`
	`Japanese version 3`
		Specifies the version of the Illustrator file format to create. Default: `Illustrator CS2`.

compressed	`boolean`	If `true`, the saved file should be compressed. Only for Illustrator 10 or later. Default: `true`.
embed ICC profile	`boolean`	If `true`, the document's ICC profile should be embedded in the saved file. Only for Illustrator 9 or later. Default: `false`.
embed linked files	`boolean`	If `true`, include linked image files in the saved document. Only for Illustrator 7 or later. Default: `false`.

flatten output	Valid values:
	`preserve paths`
	`preserve appearance`
		How should transparency be flattened for file formats before Illustrator 9 or later.
		Default: `preserve appearance`.

font subset threshold	`real`	Include a subset of fonts when less than this percentage of characters are used. Only for Illustrator 9 or later. Range: 0.0 to 100.0. Default: 100.0.
PDF compatible	`boolean`	If `true`, the file should be saved as a PDF compatible file. Only for Illustrator 10 or later.

Valid Commands

`save`

➤ Save files in a folder

This handler processes a folder of Illustrator files, saving each with Illustrator 7 compatibility. Note that the `class` property is specified in the record to ensure that Illustrator can determine the save option class.

```
-- fileList is a list of aliases to Illustrator files
-- destFolder is an alias to a folder where the Illustrator
-- files are to be saved
```

```
on SaveFilesAsIllustrator8(fileList, destFolder)
   set destPath to destFolder as string
   repeat with aFile in fileList
      tell application "Finder" to set fileName to name of aFile
      set newFilePath to destPath & fileName
      tell application "Adobe Illustrator"
         open aFile
         save current document in file newFilePath as Illustrator ¬
            with options {class:Illustrator save options, ¬
               compatibility:Illustrator 8, ¬
               flatten output:preserve appearance}
         close current document saving no
      end tell
   end repeat
end SaveFilesAsIllustrator8

-- Call handler
set sourceFolder to choose folder with prompt "Source folder?"
tell application "Finder" to set fileList to ¬
   every file of folder sourceFolder as alias list
set destFolder to choose folder with prompt "Destination folder?"
SaveFilesAsIllustrator8(fileList, destFolder)
```

ink

Specifies the properties of the inks to be used in printing the document.

ink object properties

Property	Value Type	What it is
name	Unicode text	The ink's name.
properties	ink properties	The ink information.

➤ **List inks in a document**

```
-- List Inks

-- Get the name of every ink in document 1
-- Make a list of names for display
-- Display the list of names

tell application "Adobe Illustrator"
   set inkNames to ""
   if not (exists document 1) then error "There is no available document."
   get the name of every item of inks of document 1
   repeat with theName in the result
      set inkNames to inkNames & theName & return
   end repeat
   tell me to display dialog inkNames
end tell
```

ink properties

Information about ink use when printing a document with the print command.

ink properties object properties

Property	Value Type	What it is
angle	real	The ink's screen angle in degrees.
custom color	list of real	The custom color.
density	real	The neutral density. Minimum: 0.0
dot shape	Unicode text	The dot shape name.
frequency	real	The ink's frequency. Minimum: 0.0
kind	Valid values: black ink custom ink cyan ink magenta ink yellow ink	The ink type.
printing status	Valid values: convert ink disable ink enable ink	The ink printing status.
trapping	Valid values: ignore opaque normal opaque transparent	The trapping type.
trapping order	integer	The order of trapping for the ink. Minimum: 1

insertion point

One or more insertion points in the contents of a text frame.

An insertion point is logically located between two characters in a text frame. Each insertion point is before the corresponding character in a text frame. Insertion point 1 is before character 1, etc.

The properties of an insertion point are the same as the character at the same position in the text frame. For example, the font for insertion point 2 of text frame 1 will be the same as the font for character 2 of text frame 1.

You can set the properties for an insertion point, but only setting the contents property will have any affect on the text frame. The result of setting the contents of an insertion point to a string value is to insert the string in the text frame at the insertion point's location. Setting the contents to an empty string has no affect.

insertion point object elements

Elements	Refer to by
character style	name, numeric index, range of elements, before/after another element, satisfying a test
character	numeric index, range of elements, before/after another element, satisfying a test
insertion point	numeric index, range of elements, before/after another element, satisfying a test
line	numeric index, range of elements, before/after another element, satisfying a test
paragraph style	name, numeric index, range of elements, before/after another element, satisfying a test
paragraph	numeric index, range of elements, before/after another element, satisfying a test
text	numeric index, range of elements, before/after another element, satisfying a test
word	numeric index, range of elements, before/after another element, satisfying a test

insertion point object properties

Property	Value Type	What it is
best type	type class	Read-only. The best type for the object's value.
class	type class	Read-only.The object's class.
container	reference	Read-only.The object's container.
default type	type class	Read-only.The default type for the object's value.
index	integer	Read-only.The index of this instance of the object.
properties	record	All of the properties of this object returned as a record.
story	story	Read-only.The story of the text range.

insertion point object commands

count
exists

➤ Working with insertion points

This example shows several ways of working with insertion points.

```
tell application "Adobe Illustrator"
   -- Set insertion point karat to beginning of a text frame
   set selection to insertion point 1 of text frame 1 of document 1
   -- Add a string to end of a text frame
   set contents of insertion point -1 of text frame 1 of document 1 ¬
      to " Some new text."
   -- The default type of an insertion point is string, asking for
   -- a particular insertion point returns its contents. To get a reference
   -- to an insertion point you need to ask for a reference
   set insertionRef to ¬
      insertion point after word 3 of text frame 1 of document 1 as reference
   set contents of insertionRef to " more words"
end tell
```

➤ Add a word at the insertion point

```
-- Insertion Points
--  Make a new document
--- Make a new text frame with contents "Wouldn't you rather be scripting?"
--  Change the size of the text frame
--  Get the insertion points of the last word of the text frame
--  Add a new word at the first insertion point of the result

tell application "Adobe Illustrator"
   activate
   make new document
   make new text frame in document 1 with properties ¬
      {contents:"Wouldn't you rather be scripting?", position:{100, 400}}
   set the size of the text of the result to 20
   delay 1
   get insertion points of the last word of text frame 1 of document 1
   make new word at (item 1 of the result) with properties¬
      {contents:"AppleScript"}
end tell
```

job options

The print job options when printing a document with the print command.

job options object properties

Property	Value Type	What it is
bitmap resolution	real	The bitmap resolution. Minimum: 0.0 Default: 0.0
collate	boolean	If true, collate print pages are collated. Default: false
copies	integer	The number of copies to print. Minimum: 1 Default: 1
designation	Valid values: all layers visible layers visible printable layers	The layers/objects to be printed. Default: visible printable layers
file path	file specification	The file to which to print.
name	Unicode text	The print job name.
print area	Valid values: artboard bounds artwork bounds crop bounds	The printing bounds. Default: artboard bounds
print as bitmap	boolean	If true, the job is printed as a bitmap image. Default: false
reverse pages	boolean	If true, the pages are printed in reverse order. Default: false

➤ Printing with options

```
-- Print Job Options
--
-- Make new document
-- Make new text frame in layer 1 (visible and printable)
-- Make a new layer with properties printable false
-- Make new text frame in the new layer (visible and non-printable)
-- Make new layer
-- Make new text frame in the new layer (non-visible)
-- Make the new layer non visible
-- Print all layers
-- Print only visible layers
-- Print only visible and printable layer to file
--

set theDesktop to (path to desktop as string)
tell application "Adobe Illustrator"
   activate
```

```
    make new document
    set the name of current layer of document 1 to "VPL"
    make new text frame in document 1 with properties ¬
       {contents:"Visible and Printable", position:{200, 600}}
    make new layer in document 1 with properties ¬
       {name:"VnPL", printable:false}
    make new text frame in layer "VnPL" of document 1 with properties¬
       {contents:"Visible and Non-Printable", position:{200, 500}}
    make new layer in document 1 with properties {name:"nVPL"}
    make new text frame in layer "nVPL" of document 1 with properties¬
       {contents:"Non-Visible", position:{200, 400}}
    set visible of layer "nVPL" of document 1 to false
    set printOptions to {class:print options, ¬
       job settings:{class:job options, designation:all layers,¬
          reverse pages:true}}
    print document 1 options printOptions
    set printOptions to {class:print options, ¬
       job settings:{class:job options, designation:visible layers, ¬
          reverse pages:true}}
    print document 1 options printOptions
    set printOptions to {class:print options, ¬
       job settings:{class:job options, ¬
          file path:(theDesktop & "printfile.ps" as string),¬
          designation:visible printable layers, reverse pages:true}}
    print document 1 options printOptions
end tell
```

JPEG export options

Options that can be supplied when exporting a document as a JPEG file. See the export command in the command reference for additional details.

This class is used to define a record containing properties that specify options when exporting a document as a JPEG file. `JPEG export options` can only be supplied in conjunction with the `export` command. It is not possible to get or create a `JPEG export options` object.

JPEG export options object properties

Property	Value type	What it is
antialiasing	boolean	If true, the resulting image should be anti-aliased. Default: true
artboard clipping	boolean	If true, the resulting image should be clipped to the artboard. Default: false
blur	real	The amount of blurring to apply to the resulting image. Range: 0.0 to 2.0 Default: 0.0
horizontal scaling	real	The percent horizontal scaling factor to apply to the resulting image. Range: 0.0 to 100.0 Default: 100.0
matte	boolean	If true, the artboard should be matted with a color. Default: true
matte color	RGB color info	The color to use when matting the artboard. Default: white
optimization	boolean	If true, the resulting image should be optimized for web viewing. Default: true
quality	integer	The quality of the resulting image. Range: 0 to 100 Default: 30
saving as HTML	boolean	If true, the resulting image should be saved with an accompanying HTML file. Default: false
vertical scaling	real	The percent vertical scaling factor to apply to the resulting image. Range: 0.0 to 776.19 Default: 100.0

➤ Exporting to JPEG

This handler processes all Illustrator files in a specific folder, exporting each file as a medium-quality JPEG image. Note that the `class` property is specified in the record to ensure that Illustrator can determine the export option class.

```
-- fileList is a list of aliases to Illustrator files
-- destinationFolder is an alias to a folder
--  where the JPEGs are to be exported
on ExportFilesAsJPEGMedium(fileList, destinationFolder)
   set destinationPath to destinationFolder as string
   repeat with aFile in fileList
      tell application "Finder" to set fileName to name of aFile
      set newFilePath to destinationPath & fileName & ".jpg"
```

```
      tell application "Adobe Illustrator"
         open aFile
         export current document to file newFilePath as JPEG with options¬
            {class:JPEG export options, quality:60, blur:0.5,¬
               horizontal scaling:50.0, vertical scaling:50, matte:false}
         close current document saving no
      end tell
   end repeat
end ExportFilesAsJPEGMedium

-- Call handler
set sourceFolder to choose folder with prompt "Source folder?"
tell application "Finder" to set fileList to ¬
   every file of folder sourceFolder as alias list
set destinationFolder to choose folder with prompt "Destination folder?"
ExportFilesAsJPEGMedium(fileList, destinationFolder)
```

Lab color info

A color specification in the CIE Lab color space, used where a `color info` object is required.

Lab color info properties

Note: This class inherits all properties from the `color info` class.

Property	Value type	What it is
a	`real`	The a (red-green) color value. Range -128.0–128.0. Default: 0.0
b	`real`	The b (yellow-blue) color value. Range -128.0–128.0. Default: 0.0
l	`real`	The l (lightness) color value. Range -128.0–128.0. Default: 0.0
typename	`string`	Read-only. The class name of the referenced object.

layer, layers

A layer or list of layers. Layers may contain nested layers, which are called sublayers in the user interface.

The `layer` object contains all of the page items in the specific layer as elements. Your script can access page items as elements of either the `layer` object or as elements of the `document` object. When accessing page items as elements of a layer, only objects in that layer can be accessed. To access page items throughout the entire document, be sure to refer to them as elements of the document.

layer object elements

Element	Refer to by
`compound path item`	name, index, before/after, range, test
`graph item`	name, index, before/after, range, test
`group item`	name, index, before/after, range, test
`layer`	name, index, before/after, range, test
`legacy text item`	name, index, before/after, range, test
`mesh item`	name, index, before/after, range, test
`page item`	name, index, before/after, range, test
`path item`	name, index, before/after, range, test
`placed item`	name, index, before/after, range, test
`plugin item`	name, index, before/after, range, test
`raster item`	name, index, before/after, range, test
`symbol item`	name, index, before/after, range, test
`text frame`	name, index, before/after, range, test

layer object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the layer object's value; always returns reference.
blend mode	Valid values: color blend color burn color dodge darken difference exclusion hard light hue lighten luminosity multiply normal overlay saturation blend screen soft light	The mode used when compositing an object. An object is considered composited when its opacity is set to less than 100.0 (100%).
class	type class	Read-only. The layer object's class, which is layer.
color	RGB color info	The layer's selection mark color.
container	object reference	Read-only. A reference to the document that contains this layer.
default type	type class	Read-only. The default type for the layer object's value; always returns reference.
dim placed images	boolean	If true, placed images are to be rendered as dimmed in this layer.
has selected artwork	boolean	If true, one or more objects in this layer selected are selected; setting this property to false deselects all objects in the layer.
index	integer	Read-only. The position of this layer in the current stacking order of layers in this document, where layer 1 is always the topmost layer in the stacking order.
isolated	boolean	If true, this object is isolated
knockout	Valid values: unknown disabled enabled inherited	Is this object used to create a knockout.
locked	boolean	If true, the layer is editable.
name	Unicode text	The name of this layer.
opacity	real	The opacity of this layer, where 100.0 is completely opaque and 0.0 is completely transparent.

Property	Value type	What it is
preview	boolean	If true, this layer should be displayed using preview mode.
printable	boolean	If true, this layer should be printed when printing the document.
properties	record	All of the properties of this object returned as a record.
sliced	boolean	If true, slices should be preserved. Default: false
visible	boolean	If true, this layer is visible.

layer object commands

➤ Moving layers

The stacking order of existing layers in a document can be manipulated using the move command. The following example demonstrates how to move a layer to the top of the stacking order (index position 1).

```
-- Move the 2nd layer to the top of the stacking order
tell application "Adobe Illustrator"
   if (count layers of current document) > 1 then
      move layer 2 of document 1 to before layer 1 of document 1
   end if
end tell
```

➤ Creating a layer

Commands that deal with changes to an object's reference, including the creation of new objects with the make command, return a reference to the new or modified object in their result. This example stores the reference returned for a newly created layer and then creates a new path item in the layer using the reference.

```
-- Make a new layer at the top of the layer stack
-- then create a new path in the layer
tell application "Adobe Illustrator"
   set layerRef to make layer at document 1 ¬
      with properties{name: "Our Layer"}
   make new rectangle at beginning of layerRef
end tell
```

➤ Deleting particular layers

This example demonstrates the power of constructing simple tests (with the whose clause) to selectively delete layers in a document based on their names. In this case, the script deletes all layers in the current document that have names starting with the word "Temporary".

```
-- Delete layers that have a name which begin with a particular string
set partialName to "Temp"
```

```
tell application "Adobe Illustrator"
  delete (every layer of document 1 whose name starts with partialName)
end tell
```

legacy text item, legacy text items

A text item from a document in a pre-CS version Illustrator (earlier than version 11), or a list of such items.

legacy text item object properties

Note: This class inherits all properties from the `page item` class.

Property	Value type	What it is
`converted`	`boolean`	When `true`, the item has been updated to the current text format (a `text frame`). Read-only.
`properties`	`record`	All of the properties of this object returned as a record.

legacy text item object commands

`convert`

line

A line or lines of text in a text frame. A document's text can be accessed using the `character`, `insertion point`, `word`, `line`, `paragraph`, and `text` classes.

Lines of text cannot be created. When the `contents` property of a text frame is modified, Illustrator will create text lines as it reflows the text within the text frame.

line object elements

Elements	Refer to by
`character style`	name, numeric index, range of elements, before/after another element, satisfying a test
`character`	numeric index, range of elements, before/after another element, satisfying a test
`insertion point`	numeric index, range of elements, before/after another element, satisfying a test
`line`	numeric index, range of elements, before/after another element, satisfying a test
`paragraph style`	name, numeric index, range of elements, before/after another element, satisfying a test
`paragraph`	numeric index, range of elements, before/after another element, satisfying a test
`text`	numeric index, range of elements, before/after another element, satisfying a test
`word`	numeric index, range of elements, before/after another element, satisfying a test

line object properties

Property	Value Type	What it is
`aki left`	`real`	The amount of inter-glyph space added to the left side of each glyph in Japanese text (in thousandths of an em).
`aki right`	`real`	The amount of inter-glyph spacing added to the right side of each glyph in Japanese text (in thousandths of an em).

`alignment`	Valid values:
	`bottom`
	`center`
	`icf bottom`
	`icf top`
	`roman baseline`
	`top`
		The character alignment type.

Property	Value Type	What it is
alternate glyphs	Valid values: default expert full width half width jis78 jis83 proportional width quarter width third width traditional	Specifies the type of alternate glyphs.
alternate ligature	boolean	If true, use the alternate ligature.
auto leading	boolean	If true, use automatic leading.
baseline direction	Valid values: standard Tate Chu Yoko vertical rotated	The Japanese text baseline direction.
baseline position	Valid values: normal subscript superscript	The baseline position of text.
baseline shift	real	The amount of shift (in points) of the text baseline.
best type	type class	Read-only. The best type for the object's value.
capitalization	Valid values: all caps all small caps normal small caps	The case of the text.
character offset	integer	Offset of the first character.
class	type class	Read-only. The object's class.
connection forms	boolean	If true, use the OpenType connection forms.
container	reference	Read-only. The object's container.
contents	Unicode text	The text content of the text range.
contextual ligature	boolean	If true, use the contextual ligature.
default type	type class	Read-only. The default type for the object's value.
discretionary ligature	boolean	If true, use the discretionary ligature.

Property	Value Type	What it is

figure style	Valid values:
	`default`
	`proportional`
	`proportional oldstyle`
	`tabular`
	`tabular oldstyle`
		Specifies which figure style to use in an OpenType font.

fill color	`color info`	The color of the text fill.
font	`font`	The text font.
fractions	`boolean`	If `true`, use the OpenType fractions.
horizontal scale	`real`	The character horizontal scaling factor expressed as a percentage (100 = 100%).
index	`integer`	Read-only. The index of this instance of the object
italics	`boolean`	If `true`, the Japanese OpenType support supports the italic style.
kerning	`integer`	Controls the spacing between two characters, in thousandths of an em.

kerning method	Valid values:
	`auto`
	`none`
	`optical`
		The automatic kerning method to use.

Property	Value Type	What it is
language	Valid values: Bokmal Norwegian Brazillian Portuguese Bulgarian Canadian French Catalan Chinese Czech Danish Dutch English Finnish Greek Hungarian Icelandic Italian Japanese Nynorsk Norwegian old German Polish Romanian Russian Spanish Serbian standard French standard German standard Portuguese Swedish Swiss German Turkish UK English Ukranian	The language.
leading	real	Specifies the amount of space between two lines of text, in points.
length	integer	The length of text range in characters; Minimum: 0
ligature	boolean	If true, use the ligature.
no break	boolean	Whether break is allowed.
OpenType position	Valid values: default denominator numerator subscript superscript	The OpenType baseline position.
ordinals	boolean	If true, use the OpenType ordinals.
ornaments	boolean	If true, use the OpenType ornaments.
overprint fill	boolean	If true, overprint the fill of the text.
overprint stroke	boolean	If true, the stroke of the text may be overprinted.

Property	Value Type	What it is
properties	record	All of the properties of this object returned as a record.
proportional metrics	boolean	If true, the proportional metrics in Japanese OpenType may be used.
rotation	real	The character rotation angle.
selection	list of texts	Read-only. The selected text (ranges) in the text range.
size	real	Font size in points.
story	story	Read-only. The story of the text range.
strike through	boolean	If true, characters use strike-through style.
stroke color	color info	The color of the text stroke.
stroke weight	real	line width of stroke.
stylistic alternates	boolean	If true, use the OpenType stylistic alternates.
swash	boolean	If true, use the OpenType swash.
TCY horizontal	integer	The Tate-Chu-Yoko horizontal adjustment in points.
TCY vertical	integer	The Tate-Chu-Yoko vertical adjustment in points.
titling	boolean	If true, use the OpenType titling alternates.
tracking	integer	The tracking or range kerning amount in thousandths of an em.
Tsume	real	The percentage of space reduction around a Japanese character.
underline	boolean	If true, characters use underline style.
vertical scale	real	Character vertical scaling factor.
warichu characters after break	integer	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.
warichu characters before break	integer	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.
warichu enabled	boolean	If true, Wari-Chu is enabled.
warichu gap	integer	The Wari-Chu line gap.

Property	Value Type	What it is

warichu justification	Valid values:
	`auto justify`
	`center`
	`full justify last line center`
	`full justify`
	`full justify last line left`
	`full justify last line right`
	`left`
	`right`
		The Wari-Chu justification.

warichu lines	`integer`	The number of Wari-Chu (multiple text lines fit into a space meant for one) lines.
warichu scale	`real`	The Wari-Chu scale.

line object commands

apply character style
apply paragraph style
change case
count
delete
deselect
duplicate
exists
make
move
select

➤ Finding lines of text

Lines of text can be located with matching characteristics using the `whose` clause, as this script demonstrates.

```
-- Color red all lines of text containing more than 80 characters
tell application "Adobe Illustrator"
   if (count text frames in document 1) > 0 then
      set textItemCount to count text frames in document 1
      repeat with i from 1 to textItemCount
         set (fill color of every line of text frame i of document 1 ¬
            whose length > 80) to {red:255.0}
      end repeat
   end if
end tell
```

matrix

A transformation matrix specification, used to transform the geometry of objects. Use it to specify and retrieve matrix information from an Illustrator document or from page items in a document.

Matrices are used in conjunction with the `transform` command and as a property of a number of objects. A matrix specifies how to transform the geometry of an object. You can generate an original matrix using `get identity matrix`, `get translation matrix`, `get scale matrix`, or `get rotation matrix`.

A `matrix` is a record containing the matrix values, not a reference to a matrix object. The matrix commands listed above operate on the values of a matrix record. If a command modifies a matrix, a modified matrix record is returned as the result of the command. The original matrix record passed to the command is not modified.

matrix object properties

Property	Value type	What it is
mvalue_a	real	Matrix property a.
mvalue_b	real	Matrix property b.
mvalue_c	real	Matrix property c.
mvalue_d	real	Matrix property d.
mvalue_tx	real	Matrix property tx.
mvalue_ty	real	Matrix property ty.

matrix object commands

concatenate matrix
concatenate rotation matrix
concatenate scale matrix
concatenate translation matrix
equal matrices
get identity matrix
get rotation matrix
get scale matrix
get translation matrix
invert matrix
singular matrix

➤ Getting a matrix for scale transformation

A matrix can be generated to effect a scale transformation using the `get scale matrix` command.

```
-- Scale all art in a document to 50% vertical size
tell application "Adobe Illustrator"
   if (count page items in document 1) > 0 then
      set scaleMatrix to ¬
         get scale matrix horizontal scale 100.0 vertical scale 50.0
      transform every page item in document 1 using scaleMatrix
   end if
```

```
end tell
```

➤ Applying multiple transformations

If you need to apply multiple transformations to objects it is more efficient to use the matrix suite than to apply the transformations one at a time. The following script demonstrates how to combine multiple matrices together.

```
-- Scale, rotate, and translate all art in a document
tell application "Adobe Illustrator"
   if (count page items in document 1) > 0 then
      set matrixDef to ¬
         get scale matrix horizontal scale 100.0 vertical scale 50.0
      set matrixDef to ¬
         concatenate rotation matrix matrixDef angle -45.0
      set matrixDef to ¬
         concatenate translation matrix matrixDef delta x 50.0 delta y -50.0
      transform every page item in document 1 using matrixDef
   end if
end tell
```

mesh item, mesh items

A gradient mesh art item or list of gradient mesh art items. Scripts cannot create new mesh items, but can be duplicate, copy and paste them.

mesh item object properties

Note: This class inherits all properties from the `page item` class.

Property	Value type	What it is
properties	record	All of the properties of this object returned as a record.

mesh item object commands

no color info

Represents the "none" color. Assigning a reference to a `no color` object to a document's default fill or stroke color, or those of an art item, is equivalent to setting their `filled` or `stroked` property to `false`.

Note: This class inherits all properties from the `color info` class.

➤ Setting color to none

```
-- No Color

-- Make a new document
-- Make two overlapping rectangles with different fill colors
-- Set the fill color of the top rectangle to no color

tell application "Adobe Illustrator"
   activate
   make new document
   make new rectangle in document 1 ¬
      with properties {position:{200, 500}, width:300, height:100}
   set the fill color of the result ¬
      to {class:RGB color info, red:255, green:0, blue:0}
   make new rectangle in document 1 ¬
      with properties {position:{150, 550}, width:200, height:100}
   set the fill color of the result ¬
      to {class:RGB color info, red:0, green:255, blue:0}
   delay 1
   set the fill color of page item 1 of document 1 to ¬
      {class:no color info}
end tell
```

open options

Specifies options that can be supplied when opening a file.

open options object properties

Property	Value type	What it is
update legacy text	boolean	Read-only. If true, update all legacy text objects for documents saved with Illustrator version 10 or earlier. Default: false

➤ **Opening a file with automatic update**

```
-- Open Options
-- Allow the user to choose an Illustrator file with legacy text
-- Open the file with automatic update of legacy text

tell me
   activate
   set theFile to choose file with prompt ¬
      "Choose an Illustrator File with Legacy Text:"
end tell

tell application "Adobe Illustrator"
   activate
   open theFile with options {update legacy text:true}
end tell
```

page item, page items

Any art item or list of art items. Every art item and group in a document is a page item. You may refer to a page item as an element of a document, layer, or group item.

The `page item` class gives you complete access to every art item contained in an Illustrator document. The `page item` class is the superclass of all artwork objects in a document. The classes `compound path item`, `group item`, `mesh item`, `path item`, `placed item`, `plugin item`, `raster item`, and `text frame`, each inherit a set of properties from the `page item` class.

You cannot create a `page item` directly, you must create one of the specific `page item` subclasses, such as `path item`.

page item object elements

Element	Refer to by
`tag`	name, index, before/after, range, test

page item object properties

Property	Value type	What it is
`best type`	`type class`	Read-only. The best type for the `page item` object's value; always returns `reference`.

`blend mode`	Valid values:
	`color blend`
	`color burn`
	`color dodge`
	`darken`
	`difference`
	`exclusion`
	`hard light`
	`hue`
	`lighten`
	`luminosity`
	`multiply`
	`normal`
	`overlay`
	`saturation blend`
	`screen`
	`soft light`
		The mode to use when compositing this object. An object is considered composited when its opacity is set to less than 100.0 (100%).

`class`	`type class`	Read-only. The page item object's class, which can be any one of the specific classes that are children of the page item class, including `compound path item`, `group item`, `mesh item`, `path item`, `placed item`, `plugin item`, `raster item`, and `text frame`.
`container`	`object reference`	Read-only. A reference to the layer that contains this `page item`.
`control bounds`	`rectangle`	Read-only. The bounds of the object including stroke width and controls.

Property	Value type	What it is
default type	type class	Read-only. The default type for the page item object's value; always returns reference.
editable	boolean	Read-only. If true, this page item is editable.
geometric bounds	list	Read-only. The object's bounds excluding the stroke width.
height	real	The height of the page item, calculated from the geometric bounds.
hidden	boolean	If true, this page item is hidden.
index	integer	Read-only. The position of this page item in the current stacking order of the containing layer, where page item 1 is always topmost.
isolated	boolean	If true, this object is isolated.
knockout	Valid values: unknown disabled enabled inherited	Is this object used to create a knockout.
layer	object reference	Read-only. The layer to which this page item belongs.
locked	boolean	If true, this page item is locked.
name	Unicode text	The name of this page item.
opacity	real	The opacity of this object, where 100.0 is completely opaque and 0.0 is completely transparent.
position	fixed point	The position of the top left corner of the page item.
properties	record	All of the properties of this object returned as a record.
selected	boolean	If true, this object is selected.
sliced	boolean	If true, preserve slices.
URL	Unicode text	The value of the Adobe URL tag assigned to this page item.
visibility variable	anything	The visibility variable to which this page item path is bound.
visible bounds	rectangle	Read-only. The object's visible bounds, including stroke width of any objects in the illustration.
width	real	The width of the page item, calculated from the geometric bounds.
wrap inside	boolean	If true, the text frame object should be wrapped inside this object.

Property	Value type	What it is
wrap offset	Double	The offset to use when wrapping text around this object.
wrapped	boolean	If true, wrap text frame objects around this object (text frame must be above the object).

page item object commands

count
delete
duplicate
exists
move
rotate
scale
transform
translate

➤ Moving a page item

The stacking order of existing page items in a layer can be manipulated using the move command. This example demonstrates how to move a page item to the top of the stacking order (index position 1) in a layer.

```
-- Move the last page item of layer 1 to the top of the stacking order
tell application "Adobe Illustrator"
   if (count page items of layer 1 of document 1) > 1 then
      move last page item of layer 1 of document 1 to ¬
         beginning of layer 1 of document 1
   end if
end tell
```

page marks options

Specifies the page marks options when printing a document with the print command.

page marks options object properties

Property	Value Type	What it is
bleed offset	list	The bleed offset rectangle.
color bars	boolean	If true, color bar printing is enabled. Default: false
marks offset	list	The page marks offset rectangle.
page info marks	boolean	If true, page info marks printing is enabled Default: false
page marks style	Valid values: Japanese Roman	The page marks style Default: Roman
registration marks	boolean	If true, the registration marks are printed Default: false
trim marks	boolean	If true, printing of trim marks is enabled Default: false
trim marks weight	real	Stroke weight of trim marks. Minimum: 0.0 Default: 0.125

➤ Printing page marks

```
-- Print Page Marks
-- Make sure a document is available
-- Add page mark options
-- Print the document with the page mark options

tell application "Adobe Illustrator"
   activate
   if not (exists document 1) then error "There is no available document."
   set pageMarkOptions to {class:page marks options, color bars:true, ¬
      page info marks:true, registration marks:true, trim marks:true}
   set printOptions to {class:print options, ¬
      page marks settings:pageMarkOptions}
   print document 1 options printOptions
end tell
```

paper

This class contains information about the paper to be used when printing a document with the print command.

paper object properties

Property	Value Type	What it is
name	Unicode text	The paper name.
properties	paper properties	The paper information.

paper options

Information about the paper options when printing a document with the print command.

paper options object properties

Property	Value Type	What it is
height	real	Custom paper's height in points. Minimum 0.0. Default: 0.0
name	Unicode text	The paper's name.
offset	real	Custom paper's offset in points. Minimum 0.0. Default: 0.0
transverse	boolean	If true, transverse the artwork (rotate 90 degrees) on the custom paper. Default: false
width	real	Custom paper's width. Minimum 0.0. Default: 0.0

paper properties

Information about the paper.

paper properties object properties

Property	Value Type	What it is
custom paper	boolean	If true, it is a custom paper.
height	real	The paper's height in points.
imageable area	list	The imageable area, a rectangle.
width	real	The paper's width in points.

➤ **Getting paper size**

```
-- Paper Sizes
-- Make a new document
-- Get the name of printer 1
```

```
-- Get the paper sizes of printer 1
-- For each paper size, get the name and height/width
-- Display the printer info

tell application "Adobe Illustrator"
   activate
   make new document
   set printerName to (name of item 1 of printers) as string
   set printerProperties to properties of ¬
      (get properties of item 1 of printers)
   set paperSizes to paper sizes of printerProperties
   set textContents to printerName & return & "Paper Sizes:" & return
   repeat with paperSize in paperSizes
      set paperName to name of paperSize
      set paperHeight to height of properties of paperSize
      set paperWidth to width of properties of paperSize
      set textContents to textContents & tab & paperName ¬
         & " - (" & paperHeight & " x " & paperWidth & ")" ¬
         & return as string
   end repeat
   make new text frame in document 1 with properties ¬
      {contents:textContents, position:{20, 600}}
end tell
```

paragraph, paragraphs

A paragraph or list of paragraphs of text in the contents of a text art item. A document's text can be accessed using the `character`, `insertion point`, `word`, `line`, `paragraph` and `text` classes. All text is contained within text frames.

The `paragraph` class has additional properties that other related classes do not share, including properties for margins, tab stop settings, hyphenation, and word/letter spacing.

paragraph object elements

Elements	Refer to by
`character style`	name, numeric index, range of elements, before/after another element, satisfying a test
`character`	numeric index, range of elements, before/after another element, satisfying a test
`insertion point`	numeric index, range of elements, before/after another element, satisfying a test
`line`	numeric index, range of elements, before/after another element, satisfying a test
`paragraph style`	name, numeric index, range of elements, before/after another element, satisfying a test
`paragraph`	numeric index, range of elements, before/after another element, satisfying a test
`text`	numeric index, range of elements, before/after another element, satisfying a test
`word`	numeric index, range of elements, before/after another element, satisfying a test

paragraph object properties

Property	Value Type	What it is
`aki left`	`real`	The amount of extra space (aki) added to the left side of each glyph in Japanese text (in thousandths of an em).
`aki right`	`real`	The amount of extra space (aki) added to the right side of each glyph in Japanese text (in thousandths of an em).

`alignment`	Valid values:
	`bottom`
	`center`
	`icf bottom`
	`icf top`
	`roman baseline`
	`top`
		The character alignment type.

Property	Value Type	What it is
alternate glyphs	Valid values: default expert full width half width jis78 jis83 proportional width quarter width third width traditional	Specifies the type of alternate glyphs.
auto leading	boolean	If true, automatic leading is used.
auto leading amount	real	The auto leading amount, as a percentage.
auto TCY	integer	The automatic Tate-Chu-Yoko amount.
baseline direction	Valid values: standard Tate Chu Yoko vertical rotated	Specifies the Japanese text baseline direction.
baseline position	Valid values: normal subscript superscript	The baseline position of text.
baseline shift	real	The amount of shift (in points) of the text baseline.
best type	type class	Read-only. The best type for the object's value.
BunriKinshi	boolean	If true, BunriKinshi is enabled.
Burasagari type	Valid values: forced none standard	The Burasagari type which specifies whether punctuation is allowed to fall outside of the paragraph bounding box (not available when Kinsoku Shori is set to None).
capitalization	Valid values: all caps all small caps normal small caps	The case of the text.
character offset	integer	Offset of the first character.
class	type class	Read-only. The object's class.
connection forms	boolean	If true, use the OpenType connection forms.
container	reference	Read-only. The object's container.

Property	Value Type	What it is
contents	Unicode text	The text content of the text range.
contextual ligature	boolean	If true, use the contextual ligature.
default type	type class	Read-only. The default type for the object's value.
desired glyph scaling	real	Desired glyph scaling expressed as a percentage.
desired letter spacing	real	Desired letter spacing expressed as a percentage.
desired word spacing	real	Desired word spacing expressed as a percentage.
discretionary ligature	boolean	If true, use the discretionary ligature.
every line composer	boolean	If true, the *every line composer* is enabled.
figure style	Valid values: default proportional proportional oldstyle tabular tabular oldstyle	Specifies which figure style to use in an OpenType font.
fill color	color info	The color of the text fill.
first line indent	real	First line left indent expressed in points.
font	font	The text font.
fractions	boolean	If true, use the OpenType fractions.
horizontal scale	real	The character horizontal scaling factor expressed as a percentage (100 = 100%).
hyphenate capitalized words	boolean	If true, hyphenation is enabled for the capitalized words.
hyphenation	boolean	If true, hyphenation is enabled for the paragraph.
hyphenation preference	real	Hyphenation preference scale for better spacing (0) or fewer hyphens (1). Range: 0.0 to 1.0
hyphenation zone	real	Size of the hyphenation zone.
index	integer	The index of this instance of the object.
italics	boolean	If true, the Japanese OpenType support supports the italic style.

Property	Value Type	What it is
justification	Valid values: center full justify last line center full justify last line full full justify last line left full justify last line right left right	Paragraph justification.
kerning	integer	Controls the spacing between two characters, in thousandths of an em.
kerning method	Valid values: auto none optical	The automatic kerning method to use.
Kinsoku	Unicode text	The name of a Kinsoku Shori set (a set of characters which cannot be used to begin or end a line of Japanese text).
Kinsoku order	Valid values: push in push out first push out only	The preferred Kinsoku order.
language	Valid values: Bokmal Norwegian Brazillian Portuguese Bulgarian Canadian French Catalan Chinese Czech Danish Dutch English Finnish Greek Hungarian Icelandic Italian Japanese Nynorsk Norwegian old German Polish Romanian Russian Spanish Serbian standard French standard German standard Portuguese Swedish Swiss German Turkish UK English Ukranian	The language.

Property	Value Type	What it is
leading	real	Specifies the amount of space between two lines of text (in points).
leading type	Valid values: Japanese Roman	Auto leading type.
left indent	real	Left indent of margin expressed in points.
length	integer	The length of text range in characters. Minimum: 0
ligature	boolean	If true, the ligature should be used.
maximum consecutive hyphens	integer	Maximum number of consecutive hyphenated lines.
maximum glyph scaling	real	Maximum glyph scaling expressed as a percentage.
maximum letter spacing	real	Maximum letter spacing expressed as a percentage.
maximum word spacing	real	Maximum word spacing expressed as a percentage.
minimum after hyphen	integer	Minimum number of characters after a hyphen.
minimum before hyphen	integer	Minimum number of characters before a hyphen.
minimum glyph scaling	real	Minimum glyph scaling expressed as a percentage.
minimum hyphenated word size	integer	Minimum hyphenated word size.
minimum letter spacing	real	Minimum letter spacing expressed as a percentage.
minimum word spacing	real	Minimum word spacing expressed as a percentage.
Mojikumi	Unicode text	The name of a predefined Mojikumi set for Japanese text composition.
no break	boolean	If true, a break is allowed.
ordinals	boolean	If true, use the OpenType ordinals.
ornaments	boolean	If true, use the OpenType ornaments.
overprint fill	boolean	If true, overprint the fill of the text.
overprint stroke	boolean	If true, the stroke of the text may be overprinted.

Property	Value Type	What it is

OpenType position	Valid values:
	`default`
	`denominator`
	`numerator`
	`subscript`
	`superscript`
		The OpenType baseline position.

properties	`record`	All of the properties of this object returned as a record.
proportional metrics	`boolean`	If `true`, the proportional metrics in Japanese OpenType may be used.
right indent	`real`	Right indent of margin expressed in points.
roman hanging	`boolean`	If `true`, Roman hanging punctuation is enabled.
rotation	`real`	The character rotation angle.
selection	`list of text`	The selected text (ranges) in the text range.

single word justification	Valid values:
	`center`
	`full justify last line center`
	`full justify`
	`full justify last line left`
	`full justify last line right`
	`left`
	`right`
		Justification type for a single word.

size	`real`	Font size in points.
space after	`real`	Spacing after paragraph in points.
space before	`real`	Spacing before paragraph in points.
story	`story`	The story of the text range.
strike through	`boolean`	If `true`, characters use strike-through style.
stroke color	`color info`	The color of the text stroke.
stroke weight	`real`	line width of stroke.
stylistic alternates	`boolean`	If `true`, use the OpenType stylistic alternates.
swash	`boolean`	If `true`, use the OpenType swash.
tab stops	`list of tab stop info`	Tab stop settings.
TCY horizontal	`integer`	The Tate-Chu-Yoko horizontal adjustment in points.
TCY vertical	`integer`	The Tate-Chu-Yoko vertical adjustment in points.

Property	Value Type	What it is
titling	boolean	If true, the OpenType titling alternates should be used.
tracking	integer	The tracking or range kerning amount in thousandths of an em.
Tsume	real	The percentage of space reduction around a Japanese character.
underline	boolean	If true, characters use underline style.
vertical scale	real	Character vertical scaling factor.
warichu characters after break	integer	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.
warichu characters before break	integer	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.
warichu enabled	boolean	If true, Wari-Chu is enabled.
warichu gap	integer	The Wari-Chu line gap.
warichu justification	Valid values: auto justify center full justify last line center full justify full justify last line left full justify last line right left right	The Wari-Chu justification.
warichu lines	integer	The number of Wari-Chu (multiple text lines fit into a space meant for one) lines.
warichu scale	real	The Wari-Chu scale.

paragraph object commands

apply character style
apply paragraph style
change case
count
delete
deselect
duplicate
exists
make
move
select

➤ Changing hyphenation in text

The hyphenation of all text can be quickly changed from a script, as this example shows.

```
-- Enable hyphenation for every paragraph of the current document
tell application "Adobe Illustrator"
   if (count text frames of document 1) > 0 then
      set itemCounter to count text frames of document 1
      repeat with i from 1 to itemCounter¬
         set paragraphCounter to count paragraphs of text frame i¬
            of document 1
         repeat with j from 1 to paragraphCounter
            set hyphenation of paragraph j of text frame i ¬
               of document 1 to true
         end repeat
      end repeat
   end if
end tell
```

➤ Resize and justify paragraphs

```
-- Paragraph Attributes

-- Make a new document and a rectangle
-- Make an area-text text frame, assign the rectangle as its path
-- Set contents of the text frame to text containing three paragraphs
-- Resize and justify the paragraphs

tell application "Adobe Illustrator"
   activate
   make new document
   make new rectangle in document 1 with properties ¬
      {position:{100, 400}, width:400, height:200}
   set areaText to make new text frame in document 1 ¬
      with properties {kind:area text, text path:the result}
   set theParagraph to "Left justified paragraph." & return ¬
         & "Center justified paragraph." & return ¬
         & "Right justified paragraph."
   set the contents of areaText to theParagraph
   set the size of the text of areaText to 28
   set the justification of paragraph 1 of areaText to left
   set the justification of paragraph 2 of areaText to center
   set the justification of paragraph 3 of areaText to right
end tell
```

paragraph style, paragraph styles

A named style that remembers paragraph attributes.

Note: Paragraph attributes do not have default values, and are undefined until explicitly set.

paragraph style object properties

Property	Value Type	What it is
best type	type class	Read-only. The best type for the object's value.
class	type class	Read-only. The object's class.
default type	type class	Read-only. The default type for the object's value.
aki left	real	The amount of extra space (aki) added to the left side of each glyph in Japanese text (in thousandths of an em).
aki right	real	The amount of extra space (aki) added to the right side of each glyph in Japanese text (in thousandths of an em).
alignment	Valid values: bottom center icf bottom icf top roman baseline top	The character alignment type.
alternate glyphs	Valid values: default expert full width half width jis78 jis83 proportional width quarter width third width traditional	Specifies the type of alternate glyphs.
auto leading	boolean	If true, automatic leading is used.
auto leading amount	real	The auto leading amount, as a percentage.
baseline direction	Valid values: standard Tate Chu Yoko vertical rotated	Specifies the Japanese text baseline direction.

Property	Value Type	What it is

baseline position	Valid values:
	`normal`
	`subscript`
	`superscript`
		The baseline position of text.

baseline shift	`real`	The amount of shift (in points) of the text baseline.
best type	`type class`	Read-only. The best type for the object's value.
BunriKinshi	`boolean`	If `true`, BunriKinshi is enabled.

Burasagari type	Valid values:
	`forced`
	`none`
	`standard`
		The Burasagari type which specifies whether punctuation is allowed to fall outside of the paragraph bounding box (not available when Kinsoku Shori is set to `none`).

capitalization	Valid values:
	`all caps`
	`all small caps`
	`normal`
	`small caps`
		The case of the text.

connection forms	`boolean`	If `true`, use the OpenType connection forms.
container	`reference`	Read-only. The object's container.
contextual ligature	`boolean`	If `true`, use the contextual ligature.
desired glyph scaling	`real`	Desired glyph scaling expressed as a percentage.
desired letter spacing	`real`	Desired letter spacing expressed as a percentage.
desired word spacing	`real`	Desired word spacing expressed as a percentage.
discretionary ligature	`boolean`	If `true`, use the discretionary ligature.
every line composer	`boolean`	If `true`, the *every line composer* is enabled.

figure style	Valid values:
	`default`
	`proportional`
	`proportional oldstyle`
	`tabular`
	`tabular oldstyle`
		Specifies which figure style to use in an OpenType font.

fill color	`color info`	The color of the text fill.

Property	Value Type	What it is
first line indent	real	First line left indent expressed in points.
font	font	The text font.
fractions	boolean	If true, use the OpenType fractions.
horizontal scale	real	The character horizontal scaling factor expressed as a percentage (100 = 100%).
hyphenate capitalized words	boolean	If true, hyphenation is enabled for the capitalized words.
hyphenation	boolean	If true, hyphenation is enabled for the paragraph.
hyphenation preference	real	Hyphenation preference scale for better spacing (0) or fewer hyphens (1). Range: 0.0 to 1.0
hyphenation zone	real	Size of the hyphenation zone.
index	integer	Read-only. The index of this instance of the object.
italics	boolean	If true, the Japanese OpenType font supports italic text.
justification	Valid values: center full justify full justify last line center full justify last line left full justify last line right left right	Paragraph justification.
kerning method	Valid values: auto none optical	The automatic kerning method to use.
Kinsoku	Unicode text	The name of a Kinsoku Shori set (a set of characters which cannot be used to begin or end a line of Japanese text).
Kinsoku order	Valid values: push in push out first push out only	The preferred Kinsoku order.

Property	Value Type	What it is
KurikaeshiMojiShori	boolean	If true, the Kurikaeshi Moji Shori is enabled (controls how repeated characters are handled in Japanese text).
language	Valid values: Bokmal Norwegian Brazillian Portuguese Bulgarian Canadian French Catalan Chinese Czech Danish Dutch English Finnish Greek Hungarian Icelandic Italian Japanese Nynorsk Norwegian old German Polish Romanian Russian Spanish Serbian standard French standard German standard Portuguese Swedish Swiss German Turkish UK English Ukranian	The language.
leading	real	Specifies the amount of space between two lines of text, in points.
leading type	Valid values: Japanese Roman	Auto leading type.
left indent	real	Left indent of margin expressed in points.
ligature	boolean	If true, use the ligature.
maximum consecutive hyphens	integer	Maximum number of consecutive hyphenated lines.
maximum glyph scaling	real	Maximum glyph scaling expressed as a percentage.
maximum letter spacing	real	Maximum letter spacing expressed as a percentage.

Property	Value Type	What it is
maximum word spacing	real	Maximum word spacing expressed as a percentage.
minimum after hyphen	integer	Minimum number of characters after a hyphen.
minimum before hyphen	integer	Minimum number of characters before a hyphen.
minimum glyph scaling	real	Minimum glyph scaling expressed as a percentage.
minimum hyphenated word size	integer	Minimum hyphenated word size.
minimum letter spacing	real	Minimum letter spacing expressed as a percentage.
minimum word spacing	real	Minimum word spacing expressed as a percentage.
Mojikumi	Unicode text	The name of a predefined Mojikumi set for Japanese text composition.
name	Unicode text	The paragraph style's name.
no break	boolean	If true, no line break is allowed.
OpenType position	Valid values: default denominator numerator subscript superscript	The OpenType baseline position.
ordinals	boolean	If true, use the OpenType ordinals.
ornaments	boolean	If true, use the OpenType ornaments.
overprint fill	boolean	If true, overprint the fill of the text.
overprint stroke	boolean	If true, the stroke of the text may be overprinted.
proportional metrics	boolean	If true, the proportional metrics in a Japanese OpenType font may be used.
right indent	real	Right indent of margin expressed in points.
roman hanging	boolean	If true, Roman hanging punctuation is enabled.
rotation	real	The character rotation angle.

Property	Value Type	What it is

single word justification	Valid values:
	`center`
	`full justify last line center`
	`full justify`
	`full justify last line left`
	`full justify last line right`
	`left`
	`right`
		Justification type for a single word.

size	`real`	Font size in points.
space after	`real`	Spacing after paragraph in points.
space before	`real`	Spacing before paragraph in points.
strike through	`boolean`	If `true`, characters use strike-through style.
stroke color	`color info`	The color of the text stroke.
stroke weight	`real`	line width of stroke.
stylistic alternates	`boolean`	If `true`, use the OpenType stylistic alternates.
swash	`boolean`	If `true`, use the OpenType swash.
tab stops	`list of tab stop info`	Tab stop settings.
TCY horizontal	`integer`	The Tate-Chu-Yoko horizontal adjustment in points.
TCY vertical	`integer`	The Tate-Chu-Yoko vertical adjustment in points.
titling	`boolean`	If `true`, use the OpenType titling alternates.
tracking	`integer`	The tracking or range kerning amount in thousandths of an em.
Tsume	`real`	The percentage of space reduction around a Japanese character.
underline	`boolean`	If `true`, characters use underline style.
vertical scale	`real`	Character vertical scaling factor.
warichu characters after break	`integer`	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.
warichu characters before break	`integer`	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.
warichu enabled	`boolean`	If `true`, Wari-Chu is enabled.

Property	Value Type	What it is
warichu gap	integer	The Wari-Chu line gap.
warichu justification	Valid values: auto justify center full justify last line center full justify full justify last line left full justify last line right left right	The Wari-Chu justification.
warichu lines	integer	The number of Wari-Chu (multiple text lines fit into a space meant for one) lines.
warichu scale	real	The Wari-Chu scale.

➤ Applying paragraph styles

```
-- Paragraph Styles

-- Make a new document and a rectangle
-- Make an area-text text frame, assign the rectangle as its path
-- Set contents of text frame to text containing three paragraphs
-- Resize and justify the paragraphs
-- Make a new paragraph style with a set of options
-- Apply the paragraph style to the text of the text frame

tell application "Adobe Illustrator"
  activate
  make new document
  make new rectangle in document 1 with properties¬
     {position:{100, 400}, width:400, height:200}
  set areaText to make new text frame in document 1 ¬
     with properties {kind:area text, text path:the result}
  set theParagraph to "Left justified paragraph." & return ¬
       & "Center justified paragraph." & return ¬
       & "Right justified paragraph."
  set the contents of areaText to theParagraph
  set the size of the text of areaText to 28
  set the justification of paragraph 1 of areaText to left
  set the justification of paragraph 2 of areaText to center
  set the justification of paragraph 3 of areaText to right
  delay 2
  make new paragraph style in document 1 with properties¬
     {class:paragraph style, name:"ParSty 1"}
  apply paragraph style paragraph style "ParSty 1" of document 1 ¬
     to text of text frame 1 of document 1 with clearing overrides
end tell
```

path item, path items

A path or list of paths. A path consists of path points that define its geometry. The `path items` class gives you complete access to paths in Illustrator.

path item object elements

Element	Refer to by
`path point`	index, before/after, range, test

path item object properties

Note: This object class inherits all properties from the `page item` class.

Property	Value type	What it is
`area`	`real`	Read-only. The area of this path in square points. An area may be negative or even 0. The paths winding order is determined by the sign of area. If the area is negative, the path is wound counter-clockwise. Self-intersecting paths may contain sub-areas that cancel each other out. Therefore, it is possible for a path's area to appear as zero even though it has apparent area.
`clipping`	`boolean`	If `true`, use this path as a clipping path.
`closed`	`boolean`	If `true`, this path closed.
`entire path`	`list (of path point info)`	All the path item's path points.
`evenodd`	`boolean`	If `true`, use the even-odd rule to determine insideness.
`fill color`	`color info`	The fill color of the path.
`fill overprint`	`boolean`	If `true`, the art beneath a filled object should be overprinted.
`filled`	`boolean`	If `true`, the path should be filled.
`guides`	`boolean`	If `true`, this path is a guide object.
`note`	`Unicode text`	The note text assigned to the path.

`polarity`	Valid values:
	`positive`
	`negative`
		The polarity of the path, used in the creation of compound paths.

`resolution`	`real`	The resolution of the path in dots per inch.
`selected path points`	`list (of object references)`	Read-only. All of the selected path points in the path.

Property	Value type	What it is

stroke cap	Valid values:
	`butted`
	`rounded`
	`projecting`
		The type of line capping.

stroke color	`color info`	The stroke color for the path.
stroke dash offset	`real`	The default distance into the dash pattern at which the pattern should be started
stroke dashes	`list (of real numbers)`	The lengths for dashes and gaps in dashed lines, starting with the first dash length, followed by the first gap length, and so on. Set to an empty list, `{}`, for a solid line.

stroke join	Valid values:
	`mitered`
	`rounded`
	`beveled`
		Type of join for the path.

stroke miter limit	`real`	When a default stroke join is set to `mitered`, this property specifies when the join will be converted to beveled (squared-off) by default. The default miter limit of 4 means that when the length of the point reaches four times the stroke weight, the join switches from a miter join to a bevel join. Values: 1 to 500. 1 specifies a bevel join.
stroke overprint	`boolean`	If `true`, the art beneath the stroked object should be overprinted.
stroke width	`real`	Width of stroke.
stroked	`boolean`	If `true`, the path should be stroked.

path item object commands

count
delete
duplicate
exists
move
rotate
scale
transform
translate

➤ Setting stroke width and color

The stroke width and color of a path can be easily set, as demonstrated in this example.

```
-- Set the stroke of the first path to a red 4 point line
tell application "Adobe Illustrator"
   if (count path items of document 1) > 0 then
      set properties of path item 1 of document 1 to ¬
         {stroke width:4.0, stroke color:{red:255.0}}
```

```
    end if
end tell
```

path point, path points

A point or points on a specific path. Each path point is made up of a fixed point (`anchor`) and a pair of handles (`left direction` and `right direction`). Any point can considered a corner point. Setting the `point type` property of a path point to a corner forces the left and right direction points to be on a straight line when the user attempts to modify them in the user interface.

path point object properties

Property	Value type	What it is
anchor	fixed point	The position of this point's anchor point.
best type	type class	Read-only. The best type for the path point object's value; always returns reference.
class	type class	Read-only. The path point object's class, which is path point.
container	reference	Read-only. The object's container.
default type	type class	Read-only. The default type for the path point object's value; always returns reference.
index	integer	Read-only. The position of this path point in the path item.
left direction	fixed point	The position of the path point's left direction point (in position).
point type	Valid values: smooth corner	Is this a corner path point or a curve path point.
properties	record	All of the properties of this object returned as a record.
right direction	fixed point	The position of the path point's left direction point (out position).
selected	Valid values: none anchor selected left selected right selected left right selected	Specifies which points in this path point are currently selected.

path point object commands

count
delete
duplicate
exists
make

➤ Moving a path point

The following script demonstrates how a path point of a path can be modified.

```
-- Move the first point in a path to the same spot as the last point
tell application "Adobe Illustrator"
   if (count path items of document 1) > 0 then
      set lastAnchor to anchor of last path point of path item 1 ¬
         of document 1
      set anchor of path point 1 of path item 1 of document 1 to lastAnchor
   end if
end tell
```

➤ Getting coordinates for path points

This example demonstrates how to retrieve the coordinates of every point on a path.

```
-- Returns the coordinates of each point on a path
tell application "Adobe Illustrator"
   if (count path items of document 1) > 0 then
      set anchorList to ¬
         (anchor of every path point of path item 1 of document 1)
   end if
end tell
```

path point info

Path point information for a specific path item, returned by the `entire path` property of a **`path item`**. All of the path points in a specific path item can be retrieved and specified using `entire path`, which returns a list of path point info records.

path point info object properties

Property	Value type	What it is
`anchor`	`list`	The position of a `path point`'s anchor point.
`left direction`	`list`	The position of a `path point`'s left direction point (in position).

`point type`	Valid values:
	`smooth`
	`corner`
		Specifies whether the point is a corner `path point` or a curve `path point`.

`right direction`	`fixed point`	The position of a `path point`'s left direction point (out position).

➤ Getting path point information

This example demonstrates how to get every path point for a specific path item.

```
-- Returns the path points of the first path
tell application "Adobe Illustrator"
   if (count path items of document 1) > 0 then
      set pointList to entire path of path item 1 of document 1
   end if
end tell
```

pattern, patterns

A pattern definition or list of pattern definitions contained in a document.

pattern object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the pattern object's value; always returns reference.
class	type class	Read-only. The object's class, which is pattern.
container	object reference	Read-only. A reference to the document that contains this pattern.
default type	type class	Read-only. The default type for the pattern object's value; always returns reference.
index	integer	Read-only. The position of this pattern in the application.
name	Unicode text	The pattern name.
properties	record	All of the properties of this object returned as a record.

pattern object commands

count
delete
duplicate
exists

➤ Getting the name of a pattern

The following example demonstrates how the name of a pattern can be retrieved.

```
-- Returns the name of the first pattern
tell application "Adobe Illustrator"
   set pathname to name of pattern 1 of document 1
end tell
```

pattern color info

A pattern color specification, used to specify a pattern color in conjunction with the `color` property. Pattern colors are created using a reference to an existing pattern in a document. A matrix may be specified to further transform the pattern color.

pattern color info object properties

Note: This class inherits all properties from the `color info` class.

Property	Value type	What it is
matrix	matrix	An additional transformation matrix to manipulate the prototype `pattern`.
pattern	object reference	A reference to the `pattern` object that defines the pattern to use in this color definition.
reflect	boolean	If `true`, the prototype should be reflected before filling. Default: `false`
reflect angle	real	The axis (in degrees) around which to reflect. Default: 0.0
rotation	real	The angle (in degrees) to rotate the prototype pattern before filling. Default: 0.0
scale factor	fixed point	The horizontal and vertical scaling to scale the prototype pattern expressed as a fixed point. Default: 0.0
shear angle	real	The angle (in degrees) to slant the shear by. Default: 0.0
shear axis	real	The axis (in degrees) to be used for shearing. Default: 0.0
shift angle	real	The angle (in degrees) to translate the unscaled prototype `pattern` before filling. Default: 0.0
shift distance	real	The distance to translate the unscaled prototype `pattern` before filling. Default: 0.0

➤ Using a pattern color

This example demonstrates how the default fill color of the current document can be set to a pattern color specification.

```
--Set the default fill of the document to the first pattern
tell application "Adobe Illustrator"
      set default fill color of document 1 to ¬
         {pattern:pattern 1 of document 1}
end tell
```

PDF options

Options that can be supplied when opening a PDF file.

PDF options object properties

Property	Value Type	What it is
best type	type class	Read-only. The best type for the object's value.
class	type class	Read-only. The object's class.
container	reference	Read-only. The object's container.
default type	type class	Read-only. The default type for the object's value.
page	integer	What page should be used when opening a multipage document. Default: 1
PDF crop bounds	Valid values: PDF art box PDF bleed box PDF bounding box PDF crop box PDF media box PDF trim box	What box should be used when placing a multipage document. Default: PDF media box
properties	record	All of the properties of this object returned as a record.

➤ Opening a PDF document

```
-- PDF File Options
-- Prompt the user to select a multi-page PDF file
-- Set user interaction to never interact
-- Set the page of PDF file options of the settings to 2
-- Open the file

tell me
   activate
   set theFile to choose file with prompt ¬
      "Select a multi-page PDF file to open:"
end tell

tell application "Adobe Illustrator"
   activate
   set user interaction level to never interact
   set page of PDF file options of settings to 2
   open file (theFile as string)
end tell
```

PDF save options

Options that can be supplied when saving a document as an Adobe PDF file. See the save command for additional details. This class contains properties used to specify options when saving a document to a PDF file. `PDF save options` can only be supplied in conjunction with the `save` command. It is not possible to get or create a `PDF save options` object.

Preset options can be exported from and imported to a document; see the `export PDF preset` and `import PDF preset` commands.

PDF save options object properties

Property	Value Type	What it is
acrobat layers	`boolean`	Create Adobe Acrobat® layers from top-level layers; Acrobat 6 only option Default: `false`

allow printing	Valid values:
	`pdf 128 print high res`
	`pdf 128 print low res`
	`pdf 128 print none`
	`pdf 40 print high res`
	`pdf 40 print none`
		PDF security printing permission. Default: `pdf 128 print high res`

bleed link	`boolean`	Link four bleed values. Default: `true`
bleed offset	`list`	The bleed offset rectangle

changes allowed	Valid values:
	`pdf 128 any changes`
	`pdf 128 commenting allowed`
	`pdf 128 edit page allowed`
	`pdf 128 fill form allowed`
	`pdf 128 no changes`
	`pdf 40 any changes`
	`pdf 40 commenting allowed`
	`pdf 40 no changes`
	`pdf 40 page layout allowed`
		Which PDF security changes allowed. Default: `pdf 128 any changes`

color bars	`boolean`	Draw color bars. Default: `false`

color compression	Valid values:
	`automatic JPEG high`
	`automatic JPEG low`
	`automatic JPEG maximum`
	`automatic JPEG medium`
	`automatic JPEG minimum`
	`automatic JPEG2000 high`
	`automatic JPEG2000 lossless`
	`automatic JPEG2000 low`
	`automatic JPEG2000 maximum`
	`automatic JPEG2000 medium`
		How color bitmap images should be compressed. Default: `automatic JPEG maximum`

color conversion id	Valid values:
	`repurpose`
	`color conversion to dest`
	`none`
		PDF color conversion policy. Default: `none`

Property	Value Type	What it is

color destination id	Valid values:
	`color dest doc cmyk`
	`color dest doc rgb`
	`color dest profile`
	`color dest working cmyk`
	`color dest working rgb`
	`none`
		The color destination, when color conversion is performed. Default: `none`

color downsampling	`real`	The resolution to which to downsample color image. If 0, no downsampling. Default: 150.
color downsampling threshold	`real`	Downsample if the image's resolution is above this value. Default: 450.0

color profile id	Valid values:
	`include all profiles`
	`include all rgb`
	`include dest profile`
	`leave profile unchanged`
	`none`
		PDF color profile inclusion policy. Default: `none`

color resample	Valid values:
	`average downsampling`
	`bicubic downsample`
	`nodownsample`
	`subsampling`
		How color bitmap images should be resampled. Default: `nodownsample`

color tile size	`integer`	Tile size when compressing with JPEG2000. Default: 256

compatibility	Valid values:
	`Acrobat 4`
	`Acrobat 5`
	`Acrobat 6`
		The version of the Acrobat file format to create. Default: `Acrobat 6`

compress art	`boolean`	If `true`, the line art and text should be compressed. Default: `true`
document password	`Unicode text`	A password string to open the document. Default: no string
enable access	`boolean`	If `true`, accessing 128-bit should be enabled. Default: `true`
enable copy	`boolean`	If `true`, enable copying of text 128-bit. Default: `true`
enable copy and access	`boolean`	If `true`, enable copying and accessing 40-bit. Default: `true`
enable plaintext	`boolean`	If `true`, enable plaintext metadata 128-bit; available only for Acrobat 6. Default: `false`
flattener preset	`Unicode text`	The transparency flattener preset name.

Property	Value Type	What it is
flattener settings	flattening options	The printing flattener options.
font subset threshold	real	Include a subset of fonts when less than this percentage of characters are used. Range: 0.0 to 100.0. Default: 100.0
generate thumbnails	boolean	If true, generate thumbnails for the saved document. Default: true
grayscale compression	Valid values: automatic JPEG high automatic JPEG low automatic JPEG maximum automatic JPEG medium automatic JPEG minimum automatic JPEG2000 high automatic JPEG2000 lossless automatic JPEG2000 low automatic JPEG2000 maximum automatic JPEG2000 medium automatic JPEG2000 minimum none	How grayscale bitmap images should be compressed. Default: none
grayscale downsampling	real	The resolution to which to downsample grayscale images. If 0, no downsampling. Default: 150.0
grayscale downsampling threshold	real	Downsample if the image's resolution is above this value. Default: 225.0
grayscale resample	Valid values: average downsampling bicubic downsample nodownsample subsampling	How the grayscale bitmap images should be resampled. Default: nodownsample
grayscale tile size	integer	Tile size when compressing with JPEG2000. Default: 256
monochrome compression	Valid values: CCIT3 CCIT4 none run length ZIP	How monochrome bitmap images should be compressed. Default: none
monochrome downsampling	real	The resolution to which to downsample monochrome images. If 0, no downsampling. Default: 300.0
monochrome downsampling threshold	real	Downsample if the image's resolution is above this value. Default: 450.0

Property	Value Type	What it is

monochrome resample	Valid values:
	`average downsampling`
	`bicubic downsample`
	`nodownsample`
	`subsampling`
		How monochrome bitmap images should be resampled. Default: `nodownsample`

offset	`real`	Custom offset (in points) for using the custom paper. Default: 0.0
optimization	`boolean`	If `true`, the PDF file should be saved for fast web view. Default: `false`
output condition	`Unicode text`	A comment that describes the intended printing condition. Default: no string
output condition id	`Unicode text`	The name of a registered printing condition. Default: no string
output intent profile	`Unicode text`	The color profile for the intended output. When CMS is on, this is the same as the profile selected for Destination in the Color group box. Default: no string
page info	`boolean`	If `true`, draw page information. Default: `false`

page marks style	Valid values:
	`Japanese style`
	`Roman`
		The page marks style. Default: `Roman`

PDF preset	`Unicode text`	Name of PDF preset to use. Maximum string length is 255 bytes.

pdfXstandard	Valid values:
	`PDFX 1a 2001`
	`PDFX 1a 2003`
	`PDFX 3 2001`
	`PDFX 3 2003`
	`PDFX None`
		The PDF standard, or none if not complying with any standard. Default: `PDFX None`

pdfXstandard description	`Unicode text`	A description of the selected PDF standard.
permission password	`Unicode text`	A password string to restrict editing security settings. Default: no string
preserve editability	`boolean`	If `true`, preserve Illustrator editing capabilities when saving the document. Default: `true`
printer resolution	`real`	Flattening style printer resolution. Default: 800.0
registration marks	`boolean`	If `true`, draw registration marks. Default: `false`

Property	Value Type	What it is
require doc password	boolean	If true, require a password to open the document. Default: false
require perm password	boolean	If true, a password is required to edit security settings. Default: false
trapped	boolean	If true, manual trapping has been prepared in the document. Default: false
trim mark weight	Valid values: trimmarkweight0125 trimmarkweight025 trimmarkweight05	Weight of the trim marks. Default: trimmarkweight0125
trim marks	boolean	If true, draw trim marks. Default: false
view pdf	boolean	If true, view PDF after saving. Default: false

➤ Saving to PDF

This handler processes a folder of Illustrator files, saving each file as a PDF file, with Illustrator editability and Acrobat® 6 compatibility. Note that the class property is specified in the record to ensure that Illustrator can determine the save option class.

```
-- fileList is a list of aliases to Illustrator files
-- destFolder is an alias to a folder where PDF files are saved

on SaveFilesAsPDF(fileList, destFolder)
   set destPath to destFolder as string
   repeat with aFile in fileList
      tell application "Finder" to set fileName to name of aFile
      set newFilePath to destPath & fileName
      tell application "Adobe Illustrator"
         open aFile
         save current document in file newFilePath as pdf ¬
            with options {class:PDF save options, ¬
               compatibility:Acrobat 5, preserve editability:true}
         close current document saving no
      end tell
   end repeat
end SaveFilesAsPDF

-- Call handler
set sourceFolder to choose folder with prompt "Source folder?"
tell application "Finder" to set fileList to ¬
   every file of folder sourceFolder as alias list
set destFolder to choose folder with prompt "Destination folder?"
SaveFilesAsPDF(fileList, destFolder)
```

Photoshop export options

Options that can be supplied when exporting a document as a Photoshop file. See the export command for additional details.

This class contains properties that specify options when exporting a document as a Photoshop file. `Photoshop export options` can only be supplied in conjunction with the `export` command. It is not possible to get or create a `Photoshop export options` object.

Photoshop export options object properties

Property	Value type	What it is
antialiasing	boolean	If true, the exported image should be anti-aliased. Default: true
color space	Valid values: Gray RGB CMYK	The color space of the exported file. Default: RGB
compatibility	Valid values: Photoshop 5 Photoshop 8	Which Photoshop version file format to create. Default: Photoshop 8
editable text	boolean	If true, text objects should be exported as editable text layers. Default: true
embed ICC profile	boolean	If true, an ICC profile should be embedded in the exported image. Default: false
maximum editability	boolean	If true, preserve as much of the original document's structure as possible. Default: true
resolution	real	Specifies the resolution of the exported image in dots per inch. Default: 150.0
warnings	boolean	If true, a warning dialog should be displayed because of conflicts in the export settings. Default: true
write layers	boolean	If true, the layers of the Illustrator document should be preserved in the exported image. Default: true

➤ Exporting to Photoshop format with options

This handler saves all files in a folder as layered Photoshop files. Note that the `class` property is specified in the record to ensure that Illustrator can determine the save option class.

```
-- fileList is a list of aliases to Illustrator files
-- destFolder is an alias to a folder where Photoshop files
-- are to be saved

on SaveFilesAsPhotoshop(fileList, destFolder)
   set destPath to destFolder as string
   repeat with aFile in fileList
      tell application "Finder" to set fileName to name of aFile
      set newFilePath to destPath & fileName & ".psd"
```

```
      tell application "Adobe Illustrator"
         open aFile
         export current document to file newFilePath as Photoshop ¬
            with options {class:Photoshop export options, color space:RGB,¬
               embed ICC profile:true, resolution:150}
         close current document saving no
      end tell
   end repeat
end SaveFilesAsPhotoshop

-- Call handler
set sourceFolder to choose folder with prompt "Source folder?"
tell application "Finder" to set fileList to ¬
   every file of folder sourceFolder as alias list
set destFolder to choose folder with prompt "Destination folder?"
SaveFilesAsPhotoshop(fileList, destFolder)
```

Photoshop options

You can supply options when opening a Photoshop file. See the open command in the command reference for additional details.

Photoshop options object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the object's value.
class	type class	Read-only. The object's class.
container	reference	Read-only. The object's container.
default type	type class	Read-only. The default type for the object's value.
pixel aspect ratio correction	boolean	If true, the imported images which have a non-square pixel aspect ratio should be adjusted.
preserve image maps	boolean	If true, image maps should be preserved when the document is converted. Default: true
preserve layers	boolean	If true, layers should be preserved when the document is converted. Default: true
preserve slices	boolean	If true, slices should be preserved when the document is converted. Default: true
properties	record	All of the properties of this object returned as a record.

➤ Opening a Photoshop file

```
-- Photoshop file-open options
-- Make user choose a file
-- Set open options to preserve layers, not correct aspect ratio
-- Open the file

tell me
   activate
   set theFile to choose file with prompt ¬
      "Select a Photoshop file that contains layers:"
end tell

tell application "Adobe Illustrator"
   activate
   set photoshopOptions to {class:Photoshop options, preserve layers:true, ¬
         pixel aspect ratio correction:false}
   set IllustratorPreferences to {class:Illustrator preferences, ¬
         Photoshop file options:photoshopOptions}
   set user interaction level to never interact
   open theFile
end tell
```

placed item, placed items

An artwork item (optionally stored in an external file) placed in a document. A placed item must correspond to a file containing vector-graphic data, such as a PICT, EPS or PDF file.

When you create a `placed item`, Illustrator may display a dialog. To avoid this dialog, check the box to turn the warning off the first time the dialog is displayed.

Users can place vector art files, such as EPS and PDF files, with the **File > Place** command in Illustrator. Placed items can be created from vector art files in a script using the technique illustrated in the following example. To embed the art in the document, use the `embed` command, which converts the art to an appropriate embedded art item object and removes the `placed item` object.

placed item object properties

Note: This class inherits all properties from the `page item` class.

Property	Value type	What it is
bounding box	rectangle	Read-only. Dimensions of `placed item` regardless of transformations.
content variable	anything	The content variable to which this `placed item` is bound. It is not necessary to set the type of the `content variable` before binding. Illustrator automatically sets the type to `image`.
file path	file specification	The file containing the placed artwork.
matrix	matrix	The transformation matrix applied to the `placed item`.
properties	record	All of the properties of this object returned as a record.

placed item object commands

count
delete
duplicate
embed
exists
make
move
rotate
scale
trace placed
transform
translate

➤ Placing a file in a document

This example places a vector art file in the current document.

```
-- Create a new placed vector art item
-- fileRef is an alias or file reference to the vector file to be placed
-- itemPosition is a fixed point at which to position the placed item
```

```
property itemPosition: {100.0, 200.0}
set fileRef to choose file with prompt "Select vector art file"

tell application "Adobe Illustrator"
   set placedRef to make new placed item in document 1 ¬
      with properties {file path:fileRef, position:itemPosition}
end tell
```

plugin item, plugin items

An art item or objects created by an Illustrator plug-in. Scripts cannot create plug-in items, but can duplicate, copy, and paste them.

plugin item object properties

Note: This class inherits all properties from the `page item` class.

Property	Value type	What it is
`properties`	`record`	All of the properties of this object returned as a record.
`is tracing`	`boolean`	If `true`, this plugin group was created by tracing a raster art item.
`tracing`	`tracingobject`	If this object was created by tracing a raster art item, the `tracingobject` that associates the resulting vector art with tracing options. Use the expand tracing and release tracing commands with this object to convert this plugin group to a `group item`, or to revert to the original raster art.

plugin item object commands

`count`
`delete`
`duplicate`
`exists`
`move`
`rotate`
`scale`
`transform`
`translate`

PNG8 export options

Options that can be supplied when exporting a document as a PNG file with 8-bit color. See the export command for additional details.

This class contains properties that specify options when exporting a document as a PNG8 file. `PNG8 export options` can only be supplied in conjunction with the `export` command. It is not possible to get or create a `PNG8 export options` object.

PNG8 export options object properties

Property	Value type	What it is
antialiasing	boolean	If true, the resulting image should be anti-aliased. Default: true
artboard clipping	boolean	If true, the resulting image should be clipped to the artboard. Default: false
color count	integer	The number of colors in the exported color table. This value can range from 2 to 256. The default value is 128 if the property is not set explicitly.
color dither	Valid values: none diffusion pattern dither noise	The method used to dither colors. Default: diffusion
color reduction	Valid values: selective adaptive perceptual web	The method used to reduce the number of colors in the document. Default: selective
dither percent	integer	How much should the colors be dithered as a percentage. Range: 0 to 100. Default: 88
horizontal scaling	real	The percentage horizontal scaling factor to apply to the resulting image. Range: 0.0 to 100.0 Default: 100.0
interlaced	boolean	If true, the resulting image be should be interlaced. Default: false
matte	boolean	If true, the artboard should be matted with a color. Default: true
matte color	RGB color info	The color to use when matting the artboard. Default: white
saving as HTML	boolean	If true, the resulting image should be saved with an accompanying HTML file. Default: false
transparency	boolean	If true, the resulting image should use transparency. Default: true

Property	Value type	What it is
vertical scaling	real	The percentage vertical scaling factor to apply to the resulting image. Range: 0.0 to 100.0. Default: 100.0
web snap	integer	How much should the color table be changed to match the web pallet as a percentage. Range: 0 to 100. Default: 0

➤ Exporting to PNG8

This handler saves all files in a folder as 8 bit PNG files in HTML format with dithering and interlacing. Note that the `class` property is specified in the record to ensure that Illustrator can determine the save option class.

```
-- fileList is a list of aliases to Illustrator files
-- destFolder is an alias to a folder where PNG files are saved

on SaveFilesAsPNG8HTML(fileList, destFolder)
   set destPath to destFolder as string
   repeat with aFile in fileList
      tell application "Finder" to set fileName to name of aFile
      set newFilePath to destPath & fileName & ".png"
      tell application "Adobe Illustrator"
         open aFile
         export current document to file newFilePath as PNG8 ¬
            with options {class:PNG8 export options, color count:64,¬
               color reduction:web, color dither:pattern dither,¬
               dither percent:50, interlaced:true}
         close current document saving no
      end tell
   end repeat
end SaveFilesAsPNG8HTML

-- Call handler
set sourceFolder to choose folder with prompt "Source folder?"
tell application "Finder" to set fileList to ¬
   every file of folder sourceFolder as alias list
set destFolder to choose folder with prompt "Destination folder?"
SaveFilesAsPNG8HTML(fileList, destFolder)
```

PNG24 export options

Options that can be supplied when exporting a document as a PNG file with 24-bit color. See the export command for additional details.

This class contains properties that specify options to be used when exporting a document as a PNG24 file. PNG24 export options can only be supplied in conjunction with the export command. It is not possible to get or create a PNG24 export options object.

PNG24 export options object properties

Property	Value type	What it is
antialiasing	boolean	If true, the resulting image should be anti-aliased. Default: true
artboard clipping	boolean	If true, the resulting image should be clipped to the artboard. Default: false
horizontal scaling	real	The percent horizontal scaling factor to apply to the resulting image. Range: 0.0 to 100.0. Default: 100.0
matte	boolean	If true, the artboard should be matted with a color. Default: true
matte color	RGB color info	The color to use when matting the artboard. Default: {255.0, 255.0, 255.0}
saving as HTML	boolean	If true, the resulting image be saved with an accompanying HTML file. Default: false
transparency	boolean	If true, the resulting image should use transparency. Default: true
vertical scaling	real	The percentage vertical scaling factor to apply to the resulting image. Range: 0.0 to 100.0. Default: 100.0

➤ Exporting to PNG24

This handler saves all files in a folder as 24 bit PNG files in HTML format scaled to 50%. Note that the class property is specified in the record to ensure that Illustrator can determine the save option class.

```
-- fileList is a list of aliases to Illustrator files
-- destFolder is an alias to a folder where PNG files are saved

on SaveFilesAsPNG24HTML(fileList, destFolder)
   set destPath to destFolder as string
   repeat with aFile in fileList
      tell application "Finder" to set fileName to name of aFile
      set newFilePath to destPath & fileName & ".png"
      tell application "Adobe Illustrator"
         open aFile
         export current document to file newFilePath as PNG24 ¬
            with options {class:PNG24 export options, ¬
               horizontal scaling:50.0, vertical scaling:50.0, ¬
```

```
                saving as HTML:true}
            close current document saving no
        end tell
    end repeat
end SaveFilesAsPNG24HTML

-- Call handler
set sourceFolder to choose folder with prompt "Source folder?"
tell application "Finder" to set fileList to ¬
    every file of folder sourceFolder as alias list
set destFolder to choose folder with prompt "Destination folder?"
SaveFilesAsPNG24HTML(fileList, destFolder)
```

polygon

A class used to create a multi-sided path item in an Illustrator document. This object is available only in the context of a `make` command, which creates an instance of the `path item` class. This special class allows you to quickly create complex path items using the properties provided. Properties usually associated with path items, such as `fill color`, can also be specified at the time of creation.

If you do not specify any properties when making a new polygon, default values are used.

polygon object properties

Property	Value type	What it is
center point	`fixed point`	Write-once. The center point for the polygon. Default: {200.0, 300.0}
radius	`real`	Write-once. The radius of the polygon's points. Default: 50.0
reversed	`boolean`	Write-once. If `true`, the polygon path is reversed. Default: `false`
sides	`integer (unsigned)`	Write-once. The number of sides for the polygon. Default: 8

polygon object commands

`make`

➤ Creating a polygon

This sample demonstrates how to create a polygon.

```
-- Make an octagon in document 1
tell application "Adobe Illustrator"
   set pathRef to make new polygon in document 1 with properties ¬
      {center point:{200.0, 200.0}, radius:40.0, sides:8}
end tell
```

postscript options

Specifies the options for printing to a PostScript language printer or image setter when printing a document with the print command.

postscript options object properties

Property	Value Type	What it is
binary printing	boolean	If true, job is to be printed in binary mode. Default: false
compatible shading	boolean	If true, use PostScript language level 1 compatible gradient and gradient mesh printing. Default: false
force continuous tone	boolean	If true, force continuous tone. Default: false
image compression	Valid values: JPEG none RLE	The image compression type. Default: none
negative printing	boolean	If true, print in negative mode. Default: false
PostScript	Valid values: level 1 level 2 level 3	The PostScript language level. Default: level 2
shading resolution	real	The shading resolution in dots per inch. Range: 1.0 to 9600.0; Default: 300.0

PPD file

Associates properties with a PPD file to be used in printing to a PostScript language printer or image setter. The properties are not available unless a document is open.

PPD file object properties

Property	Value Type	What it is
name	Unicode text	The PPD name.
properties	PPD properties	The PPD file information.

➤ **Saving to PPD**

```
-- PPD File
-- Make a new document
-- Get the PPDs
-- Get the name, PS Level, and file path of the first PPD
-- Make a new text frame with the PPD info as its contents

tell application "Adobe Illustrator"
   activate
   make new document
   set PPDFiles to PPDs
   set PPDName to name of item 1 of PPDFiles
   set PPDProperties to get properties of item 1 of PPDFiles
   set PPDLevel to language level of PPDProperties
   set PPDPath to file path of PPDProperties
   set textContents to PPDName & return & "PostScript Level " ¬
      & PPDLevel & return & "PPD Path: " & PPDPath as string
   make new text frame in document 1 with properties ¬
      {contents:textContents, position:{20, 600}}
end tell
```

PPD properties

Specifies information about a PPD file.

PPD properties object properties

Property	Value Type	What it is
file path	Unicode text	Path specification for the PPD file.
language level	Unicode text	The PostScript language level.
screens	list of separation screen	List of color separation screens.
spot functions	list of screen spot functions	List of color separation screen spot functions.

➤ Using PPD information

```
-- PPD File Combined
-- Make a new document
-- Get the PPDs
-- Get name, PS Level, screens, screen spot functions, file path of first PPD
-- For each screen, get the name, angle, and frequency
-- For each spot function, get the name and the function
-- Make a new text frame with the PPD info as its contents

tell application "Adobe Illustrator"
   activate
   make new document
   set PPDFiles to PPDs
   set PPDName to name of item 1 of PPDFiles
   set PPDProperties to get properties of item 1 of PPDFiles
   set PPDLevel to language level of PPDProperties
   set PPDPath to file path of PPDProperties
   set PPDScreens to screens of PPDProperties
   set screensText to "Screens" & return
   repeat with PPDScreen in PPDScreens
      set PPDScreenName to name of PPDScreen
      set PPDScreenAngle to angle of properties of PPDScreen
      set PPFScreenFrequency to frequency of properties of PPDScreen
      set screensText to screensText & tab & PPDScreenName & " - Angle: "¬
         & PPDScreenAngle & ", Frequency: " & PPFScreenFrequency ¬
         & return as string
   end repeat
   set PPDSpotFunctions to spot functions of PPDProperties
   set PPDSpotFunctionText to "Spot Functions" & return
   repeat with PPDSpotFunction in PPDSpotFunctions
      set PPDSpotFunctionName to name of PPDSpotFunction
      set PPDSpotFunctionTX to spot function of PPDSpotFunction
      set PPDSpotFunctionText to PPDSpotFunctionText & tab ¬
         & PPDSpotFunctionName & ": " & PPDSpotFunctionTX ¬
         & return as string
   end repeat
   set textContents to PPDName & return & "PostScript Level " ¬
      & PPDLevel & return & "PPD Path: " & PPDPath & return & return ¬
      & screensText & return & return & PPDSpotFunctionText as string
   make new text frame in document 1 with properties ¬
      {contents:textContents, position:{20, 700}}
end tell
```

print options

Collects all print options when printing a document with the print command.

print options object properties

Property	Value Type	What it is
color management settings	color management options	The printing color management options.
color separation settings	color separation options	The printing color separation options.
coordinate settings	coordinate options	The printing coordinate options.
flattener preset	Unicode text	The transparency flattener preset name.
flattener settings	flattening options	The printing flattener options.
font settings	font options	The printing font options.
job settings	job options	The printing job options.
page marks settings	page marks options	The printing page marks options.
paper settings	paper options	The paper options.
postscript settings	postscript options	The printing PostScript options.
PPD name	Unicode text	The name of the PPD file.
print preset	Unicode text	The name of the printer preset to use.
printer name	Unicode text	The printer name.

➤ Printing with options

```
-- Print Options
-- Make new document. Add 6 symbol items
-- Set job options, color management options, coordinate options,
-- and flattening options
-- Print the document using these options

set theDesktop to (path to desktop as string)
tell application "Adobe Illustrator"
   activate
   make new document
   repeat with i from 1 to 6
      round (i / 2 - (round (i / 2) rounding down)) rounding up
      make new symbol item in document 1 with properties ¬
         {symbol:symbol i of document 1, ¬
          position:{100 + (the result * 150), (50 + i * 70)}}
   end repeat
   set jobOptions to {class:job options, designation:all layers,¬
      reverse pages:true}
```

```
    set colorOptions to {class:color management options, ¬
       name:"ColorMatch RGB", intent:saturation}
    set coordinateOptions to {class:coordinate options, fit to page:true}
    set flatteningOptions to {class:flattening options, ¬
       clip complex regions:true, gradient resolution:60, ¬
       rasterization resolution:60}
    set printOptions to {class:print options, job settings:jobOptions, ¬
       color management settings:colorOptions,¬
       coordinate settings:coordinateOptions,¬
       flattener settings:flatteningOptions}
    print document 1 options printOptions
end tell
```

printer

Associates an installed printer with a printer configuration object.

printer object properties

Property	Value Type	What it is
name	Unicode text	The printer name.
properties	printer properties	The printer information.

➤ **Listing printers**

```
-- Printer List
-- Make a new document
-- Get the name of every printer
-- Display the list of names

tell application "Adobe Illustrator"
   set printerList to ""
   activate
   make new document
   name of every item of printers as list
   repeat with theName in the result
      set printerList to printerList & theName & return
   end repeat
   set user interaction level to interact with all
   display dialog printerList
   set user interaction level to never interact
end tell
```

printer properties

Specifies configuration information for a printer.

printer properties object properties

Property	Value Type	What it is
binary printing	boolean	If true, the printer supports binary printing.
color support	Valid values: black and white output color output grayscale output	The printer color capability.
custom paper sizes	boolean	If true, the printer supports custom paper sizes.
custom paper transverse	boolean	If true, the printer supports custom paper transverse.

Property	Value Type	What it is
default resolution	real	The printer default resolution. Minimum: 0.0
InRIP separation support	boolean	If true, the printer supports InRIP color separation.
maximum height offset	real	Custom paper's maximum height offset.
maximum paper height	real	Custom paper's maximum height.
maximum paper width	real	Custom paper's maximum width.
maximum resolution	real	The printer maximum device resolution. Minimum: 0.0
maximum width offset	real	Custom paper's maximum width offset.
minimum height offset	real	Custom paper's minimum height offset.
minimum paper height	real	Custom paper's minimum height.
minimum paper width	real	Custom paper's minimum width.
minimum width offset	real	Custom paper's minimum width offset.
paper sizes	list of paper	The list of supported paper sizes.
PostScript	Valid values: level 1 level 2 level 3	The PostScript language level.
printer type	Valid values: non PostScript printer PostScript printer unknown	The printer type.

raster item, raster items

A bitmap art item or list of objects. You can create `raster items` from a script if you use an external file. You can also create new raster items by duplicating or copying and pasting an existing `raster item`.

raster item object properties

Note: This class inherits all properties from the `page item` class.

Property	Value type	What it is
bounding box	`rectangle`	Dimensions of `raster item` regardless of transformations.

color space	Valid values:
	`Gray`
	`RGB`
	`CMYK`
		Read-only. The color space of the `raster item`.

content variable	`anything`	The contents of the variable to which this raster item is bound. It is not necessary to set the type of the `content variable` before binding. Illustrator automatically sets the type to image.
embedded	`boolean`	If `true`, the `raster item` is embedded within the illustration.
file path	`file specification`	The file containing the `raster item`, if it is stored externally.
matrix	`matrix`	The transformation matrix of the raster art item.
properties	`record`	All of the properties of this object returned as a record.

status	Valid values:
	`no data`
	`data from file`
	`modified data`
		Read-only. The status of the linked image, if the image is stored externally.

raster item object commands

count
delete
duplicate
exists
move
rotate
scale
trace raster
transform
translate

rectangle

A class used to create a rectangular path in an Illustrator document. This object is available only in the context of a `make` command, which creates an instance of the `path item` class. This special class allows you to quickly create complex path items. Properties associated with `path items`, such as `fill color` and `note`, can also be specified at the time of creation.

A rectangle is stored as a list of four real numbers, where the first item is the leftmost horizontal coordinate of the rectangle, the second item is the top vertical coordinate of the rectangle, the third item is the rightmost horizontal coordinate, and the fourth item is the bottom vertical coordinate of the rectangle.

In the Illustrator coordinate system, vertical coordinates increase from bottom to top, which is the opposite of screen coordinates. This mean that the top coordinate value in a rectangle is larger than the bottom coordinate value.

rectangle object properties

Property	Value type	What it is
`bounds`	`list`	Write-once. The bounds of the rectangle. Default: {100.0, 200.0, 175.0, 100.0}
`reversed`	`boolean`	Write-once. If `true`, the path is reversed. Default: `false`

rectangle object commands

`make`

➤ Creating a rectangle

This example demonstrates how to create a square rectangle with a note.

```
-- Make a square in document 1
tell application "Adobe Illustrator"
   set pathRef to make new rectangle at beginning of document 1 ¬
      with properties {bounds:{50.0, 200.0, 200.0, 50.0}, note:"square"}
end tell
```

➤ Using rectangle values

The values in a `rectangle` can be used in a number of ways in a script.

```
tell application "Adobe Illustrator"
   -- Get the bounds of a page item
   set itemBounds to geometric bounds of page item 1 of document 1
   -->  {100.0, 400.0, 300.0, 200.0}

   -- Assigns the four values in a rectangle point to four variables
   set {l, t, r, b} to itemBounds
   -->  l = 100.0, t = 400.0, r = 300.0, b = 200.0
   -- or assign to four variables directly
   set {l, t, r, b} to geometric bounds of page item 1 of document 1
   -->  l = 100.0, t = 400.0, r = 300.0, b = 200.0

   -- Calculate center of page item from its bounds
   set xCenter to ((item 1 of itemBounds) + (item 3 of itemBounds)) / 2
```

```
    set yCenter to ((item 2 of itemBounds) + (item 4 of itemBounds)) / 2
    -->  xCenter = 200.0, yCenter = 300.0

    -- or calculate the center using the individual coordinate variables
    set xCenter to (l + r) / 2
    set yCenter to (t + b) / 2
    -->  xCenter = 200.0, yCenter = 300.0

    -- Change the left value in a fixed rectangle
    set item 1 of itemBounds to (item 1 of itemBounds) + 100.0
    -->  {200.0, 400.0, 300.0, 200.0}
end tell
```

RGB color info

An RGB color specification, used to specify a RGB color where a `color info` object is required.

If the color space of a document is CMYK and you specify the color value for a page item in that document using `RGB color info`, Illustrator will translate the RGB color specification into a CMYK color specification. The same thing happens if the document's color space is RGB and you specify colors using `CMYK color info`. Since this translation can cause information loss you should specify colors using the color info class that matches the document's color space.

RGB color info object properties

Note: This class inherits all properties from the `color info` class.

Property	Value type	What it is
red	real	The red color value. Range: 0.0 to 255.0 Default: 0.0
green	real	The green color value. Range: 0.0 to 255.0 Default: 0.0
blue	real	The blue color value. Range: 0.0 to 255.0 Default: 0.0

➤ Setting to an RGB color

The following example demonstrates how the default stroke color of the current document can be set to a RGB color specification.

```
-- Set the default stroke color of document 1 to yellow
tell application "Adobe Illustrator"
   set default stroke color of document 1 to {red:255, green:255, blue:0}
end tell
```

rounded rectangle

A class used to create a rectangular path with rounded corners in an Illustrator document. This object is available only in the context of a `make` command, which creates an instance of the `path item` class. This special class allows you to quickly create complex path items. Properties associated with `path items`, such as `fill color` and `note`, can also be specified at the time of creation.

If you do not specify any properties when making a new rounded rectangle, default values are used.

rounded rectangle object properties

Property	Value type	What it is
bounds	rectangle	Write-once. The bounds of the rectangle to create. Default: {100.0, 100.0, 150.0, 200.0}
horizontal radius	real	Write-once.The horizontal radius of the rectangle's rounded corners. Default: 15.0
reversed	boolean	Write-once.If `true`, the rectangle path reversed. Default: `false`
vertical radius	real	Write-once. The vertical radius of the rectangle's rounded corners. Default: 20.0

rounded rectangle object commands

make

➤ Creating a rounded rectangle

The following script demonstrates how to create a rounded rectangle that is square.

```
-- Make a rounded rectangle
tell application "Adobe Illustrator"
   set pathRef to make new rounded rectangle in document 1 ¬
      with properties {bounds:{50.0, 200.0, 200.0, 50.0}, ¬
         horizontal radius:20.0, vertical radius:25.0}
end tell
```

screen properties

Contains screen information.

screen properties object properties

Property	Value Type	What it is
angle	real	The screen's angle in degrees.
default screen	boolean	If true, it is the default screen.
frequency	real	The screen's frequency.

➤ Getting screen properties

```
-- PPD Screens
-- Make a new document, get the PPDs
-- Get the name, and screens of the first PPD
-- For each screen, get the name, angle, and frequency
-- Display the results of the PPD info in a text frame

tell application "Adobe Illustrator"
   activate
   make new document
   set PPDFiles to PPDs
   set PPDName to name of item 1 of PPDFiles
   set PPDProperties to get properties of item 1 of PPDFiles
   set PPDScreens to screens of PPDProperties
   set screensText to "Screens" & return
   repeat with PPDScreen in PPDScreens
      set PPDScreenName to name of PPDScreen
      set PPDScreenAngle to angle of properties of PPDScreen
      set PPFScreenFrequency to frequency of properties of PPDScreen
      set screensText to screensText & tab & PPDScreenName & " - Angle: "¬
         & PPDScreenAngle & ", Frequency: " & PPFScreenFrequency ¬
         & return as string
   end repeat
   set textContents to PPDName & return & screensText
   make new text frame in document 1 with properties ¬
      {contents:textContents, position:{20, 600}}
end tell
```

screen spot function

Information about the color separation screen spot function.

screen spot function object properties

Property	Value Type	What it is
name	Unicode text	The color separation screen spot function name.
spot function	Unicode text	The spot function in terms of the PostScript commands.

➤ Getting screen spot function information

```
-- PPD Screen Spot Functions
-- Make a new document, get the PPDs
-- Get the name, and spot functions of the first PPD
-- For each spot function, get the name and the function
-- Display the results of the PPD info in a text frame

tell application "Adobe Illustrator"
   activate
   make new document
   set PPDFiles to PPDs
   set PPDName to name of item 1 of PPDFiles
   set PPDProperties to get properties of item 1 of PPDFiles
   set PPDSpotFunctions to spot functions of PPDProperties
   set PPDSpotFunctionText to "Spot Functions" & return
   repeat with PPDSpotFunction in PPDSpotFunctions
      set PPDSpotFunctionName to name of PPDSpotFunction
      set PPDSpotFunctionTX to spot function of PPDSpotFunction
      set PPDSpotFunctionText to PPDSpotFunctionText & tab ¬
         & PPDSpotFunctionName & ": " & PPDSpotFunctionTX ¬
         & return as string
   end repeat
   set textContents to PPDName & return & PPDSpotFunctionText
   make new text frame in document 1 with properties ¬
      {contents:textContents, position:{20, 600}}
end tell
```

separation screen

Represents a color separation screen.

separation screen object properties

Property	Value Type	What it is
name	Unicode text	The color separation screen name.
properties	screen properties	The color separation screen information.

spot, spots

A custom color definition, or list of definitions, contained in a document. All Illustrator documents contain the spot color "[Registration]" which can be used to print to all plates of a separation.

If no properties are specified when creating a new spot, default properties will be provided. However, if specifying the color, you must use the same color space as the document, either CMYK or RGB. Otherwise, an error will result. When created, the spot is inserted into the swatch palette at the end.

spot object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the spot object; always returns reference.
class	type class	Read-only. The object's class, which is spot.
color	spot color info	The color information for this spot color.
color type	Valid values: process color registration color spot color	The color model for this spot color.
container	object reference	Read-only. A reference to the document that contains this spot color.
default type	type class	Read-only. Default type for the spot; always returns reference.
index	integer	Read-only. The position of this spot in the document.
name	Unicode text	The spot color's unique name.
properties	record	All of the properties of this object returned as a record.

spot object commands

count
delete
duplicate
exists
make

➤ Creating a spot color

This script demonstrates how a spot color can be created.

```
-- Make a new spot with name and color properties
tell application "Adobe Illustrator"
   -- set up the appropriate color record for the document color space
   set docColorSpace to color space of document 1
   if (docColorSpace is CMYK) then
      set newSpotColor to {cyan:25.0, magenta:75.0, yellow:0.0, black:0.0}
   else
      set newSpotColor to {red:255.0, green:0.0, blue:25.0}
```

```
    end if
    -- now create the new spot
    make new spot in document 1 with properties ¬
       {name:"My Spot", color:newSpotColor}
end tell
```

spot color info

A spot color specification, used to specify a spot color in the `spot` object's `color` property.

spot color info object properties

Note: This class inherits all properties from the `color info` class.

Property	Value type	What it is
spot	`object reference`	A reference to the `spot` object which defines the color. Must be set to a reference to an existing spot color definition
tint	`real`	The tint of the color. Range: 0.0 to 100.0 Default: 100.0

➤ Setting to a spot color

This script demonstrates how the default stroke color of the current document can be set to a new spot color specification.

```
-- Make a new spot color and apply a 50% tint to the default stroke color
tell application "Adobe Illustrator"
   -- create a document with RGB color space
   make new document with properties {color space:RGB}
   set newSpot to make new spot in document 1 with properties ¬
         {name:"Big Blue", color:{red:0.0, green:0.0, blue:255.0}}
   set default stroke color of document 1 to {spot:newSpot, tint:50.0}
end tell
```

star

A class used to create a star-shaped path in an Illustrator document. This object is available only in the context of a `make` command, which creates an instance of the `path item` class. This special class allows you to quickly create complex path items. Properties associated with `path items`, such as `fill color` and `note`, can also be specified at the time of creation.

star object properties

Property	Value type	What it is
`center point`	`fixed point`	Write-once. The center point of the `star`. Default: {200.0, 300.0}
`inner radius`	`real`	Write-once. The inner radius of the `star`. Default: 20.0
`point count`	`integer`	Write-once. The number of points on the `star`. Default: 5
`radius`	`real`	Write-once. The radius of the `star`'s points. Default: 50.0
`reversed`	`boolean`	Write-once. If `true`, the `star` path is reversed. Default: `false`

star object commands

`make`

➤ Creating a star

This script demonstrates how to create a star.

```
-- Make a 16-pointed star
tell application "Adobe Illustrator"
   make new star in document 1 with properties ¬
      {center point:{200.0, 500.0}, inner radius:70, ¬
       radius:100, point count:16}
end tell
```

story, stories

A contiguous block of text with a specified text range. A story can contain one or more text frames; if more—the multiple text frames are linked together to form a single story.

story object elements

Elements	Refer to by
character	numeric index, range of elements, before/after another element, satisfying a test
insertion point	numeric index, range of elements, before/after another element, satisfying a test
line	numeric index, range of elements, before/after another element, satisfying a test
paragraph	numeric index, range of elements, before/after another element, satisfying a test
text	numeric index, range of elements, before/after another element, satisfying a test
text frame	name, numeric index, range of elements, before/after another element, satisfying a test
word	numeric index, range of elements, before/after another element, satisfying a test

story object properties

Property	Value Type	What it is
best type	type class	Read-only. The best type for the object's value.
class	type class	Read-only. The object's class.
container	reference	Read-only. The object's container.
default type	type class	Read-only. The default type for the object's value.
index	integer	Read-only. The index of this instance of the object.
length	integer	Read-only. The length of text range in characters. Minimum: 0
properties	record	All of the properties of this object returned as a record.
selection	list (of texts)	Read-only. The selected text ranges in the story.
text range	text	Read-only. The text range of the story.

➤ Using stories

```
-- Story
-- Make a new document and two text frames
-- Set the previous frame of the second text frame to text frame 1
-- Add a story to text frame 1, long enough to overflow to text frame 2
-- Count the number of stories
-- Add a new text frame
-- Count the number of stories

tell application "Adobe Illustrator"
```

```
    activate
    make new document
    make new rectangle in document 1 with properties ¬
       {position:{200, 600}, height:30, width:50}
    make new text frame in document 1 with properties ¬
       {name:"Text1", kind:area text, text path:the result}
    make new rectangle in document 1 with properties ¬
       {position:{300, 550}, height:200, width:50}
    make new text frame in document 1 with properties ¬
       {name:"Text2", kind:area text, text path:the result}
    set previous frame of text frame "Text2" of document 1 ¬
       to text frame "Text1" of document 1
    set the contents of text frame "Text1" of document 1 ¬
       to "This is two text frames linked together as one story"
    set user interaction level to interact with all
    display dialog "Number of Stories in Document: " ¬
       & (count stories of document 1) as string
    make new rectangle in document 1 with properties ¬
       {position:{200, 300}, height:30, width:150}
    make new text frame in document 1 with properties ¬
       {name:"Text3", kind:area text, text path:the result}
    set the contents of text frame "Text3" of document 1 ¬
       to "Each unlinked textFrame adds a new story"
    display dialog "Number of Stories in Document: " ¬
       & (count stories of document 1) as string
    set user interaction level to never interact
end tell
```

SVG export options

Options that can be supplied when exporting a document as an SVG file. See the export command in the command reference for additional details.

This class is used to define a record containing properties that specify options when exporting a document as a SVG file. `SVG export options` can only be supplied in conjunction with the `export` command. It is not possible to get or create an `SVG export options` object.

SVG export options object properties

Property	Value type	What it is
compressed	boolean	If true, the exported file should be compressed. Default: false
coordinate precision	integer	The decimal precision for element coordinate values. Range: 1 to 7 Default: 3
CSS properties	Valid values: entities style attributes style elements presentation attributes	How should the CCS properties of the document be included in the exported file. Default: style attributes
document encoding	Valid values: ASCII UTF8 UTF16	How the text should be encoded in the document. Default: ASCII
DTD	Valid values: SVG 1.0 SVG 1.1 SVG Basic 1.1 SVG Tiny 1.1 SVG Tiny 1.1 Plus	The DTD version to which the exported file conforms. Default: SVG 1.1
embed auto kerning	boolean	If true, SVG automatic kerning is allowed for the file. Default: false
embed raster images	boolean	If true, the raster images used in the document should be included in the exported file. Default: false
embed text on path	boolean	If true, the SVG text-on-path construct is allowed for the file. Default: false
font subsetting	Valid values: none all glyphs glyphs used common english glyphs used plus english common roman glyphs used plus roman	Specifies which font glyphs should be included in the exported file. Default: all glyphs

Property	Value type	What it is
font type	Valid values: CEF font outline font SVG font	The type of font to be included in the exported file.
include file info	boolean	If true, the XMP metadata should be included in the exported file. Default: false
include variables and datasets	boolean	If true, variables and datasets should be included. Default: false
optimize for SVG Viewer	boolean	If true, the Adobe namespace should be included. Default: false
preserve editability	boolean	If true, Illustrator editing capabilities should be preserved when exporting the document. Default: false
slices	boolean	If true, slice data should be preserved in exported document. Default: false

➤ Exporting to SVG

This handler saves all files in a folder as SVG files with linked raster imaged embedded in the exported files. Note that the class property is specified in the record to ensure that Illustrator can determine the save option class.

```
-- fileList is a list of aliases to Illustrator files
-- destFolder is an alias to a folder where SVG files are saved
on SaveFilesAsSVG(fileList, destFolder)
   set destPath to destFolder as string
   repeat with aFile in fileList
      tell application "Finder" to set fileName to name of aFile
      set newFilePath to destPath & fileName
      tell application "Adobe Illustrator"
         open aFile
         export current document to file newFilePath as SVG ¬
            with options {class:SVG export options, ¬
               embed raster images:true}
         close current document saving no
      end tell
   end repeat
end SaveFilesAsSVG

-- Call handler
set sourceFolder to choose folder with prompt "Source folder?"
tell application "Finder" to set fileList to ¬
   every file of folder sourceFolder as alias list
set destFolder to choose folder with prompt "Destination folder?"
SaveFilesAsSVG(fileList, destFolder)
```

swatch, swatches

A color swatch or list of swatches contained in a document. The swatches correspond to the swatch palette in the Illustrator user interface. Additional swatches can be created either manually by a user or by a script. The swatch can hold all types of color data (such as pattern, gradient, CMYK, RGB, gray, or spot).

swatch object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the swatch; always returns reference.
class	type class	Read-only. The swatch object's class, which is swatch.
color	color info	The color information for this swatch.
container	object reference	Read-only. A reference to the document that contains this swatch.
default type	type class	Read-only. The default type for the swatch; always returns reference.
index	integer	Read-only. The position of this swatch in the document.
name	Unicode text	The unique name of the swatch.
properties	record	All of the properties of this object returned as a record.

swatch object commands

count
delete
duplicate
exists
make

➤ Creating a swatch

This script demonstrates how to create a swatch with a specified name.

```
-- Make a new swatch
tell application "Adobe Illustrator"
   make new swatch in document 1 with properties ¬
      {name:"My Swatch", color:{red:175.0, green:50.0, blue:0.0}}
end tell
```

symbol, symbols

A symbol or list of symbols. A `symbol` is an art item that is stored in the symbols palette, and can be reused one or more times in the document without duplicating the art data. Symbols are contained in documents.

symbol object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the `symbol` object's value; always returns `reference`.
class	type class	Read-only. The `symbol`'s class, which is `symbol`.
container	object reference	Read-only. A reference to the object that contains this `symbol`.
default type	type class	Read-only. The default type for the `symbol`.
index	integer	Read-only. The index of this `symbol`.
name	Unicode text	Read-only. The name of the `symbol`. Defaults to "`New Symbol nnn`" where `n` is an integer, starting at 1 and increasing with each newly created symbol.
properties	record	All of the properties of this object returned as a record.
source art	anything	Read-only. The source art is only used when creating a new `symbol`.

symbol object commands

count
delete
duplicate
exists
make

➤ Using symbols

```
-- Symbol Items
-- Make a new document
-- Add 4 rectangles, apply different graphic style to each
-- Add delay of at least a second (allow UI to catch up to scripting plug-in)
-- Make a new symbol for each page item, use the page item as the source art

tell application "Adobe Illustrator"
   activate
   make new document
   repeat with i from 1 to 4
      round (i / 2 - (round (i / 2) rounding down)) rounding up
      make new rectangle in document 1 with properties ¬
         {position:{100 + (the result * 150), (50 + i * 70)},¬
          height:20, width:20}
      apply graphic style (i + 1) of document 1 to the result
   end repeat
```

```
    delay 2
    repeat with i from 1 to 4
       make new symbol in document 1 with properties ¬
          {name:("symbol" & i as string), ¬
           source art:page item i of document 1}
    end repeat
end tell
```

symbol item, symbol items

An instance of a `symbol` in a document. Symbol items are linked to the `symbol` from which they are created and change with any modification of that `symbol`.

symbol item object properties

Note: This class inherits all properties from the `page item` class.

Property	Value type	What it is
`properties`	`record`	All of the properties of this object returned as a record.
`symbol`	`symbol`	The symbol that was used to create this symbol item.

symbol item object commands

count
delete
duplicate
exists
move
rotate
scale
transform
translate

➤ Creating symbol items

```
-- Symbol Items
-- Make a new document, add 4 symbol items

tell application "Adobe Illustrator"
   activate
   make new document
   repeat with i from 1 to 4
      round (i / 2 - (round (i / 2) rounding down)) rounding up
      make new symbol item in document 1 with properties ¬
         {symbol:symbol i of document 1, ¬
          position:{100 + (the result * 150), (50 + i * 70)}}
   end repeat
end tell
```

tab stop info, tab stops

Tab stop information for a paragraph. All of the tab stops in a paragraph can be retrieved and specified using `tab stops`, which returns a list of `tab stop info` records.

tab stop info object properties

Property	Value type	What it is

`alignment`	Valid values:
	`left`
	`center`
	`right`
	`decimal`
		The alignment of the tab stop. Default: `left`

`decimal character`	`Unicode text`	The character to use for decimal tab stops.
`leader`	`Unicode text`	The leader dot.
`position`	`real`	The position of the tab stop expressed in points. Default: 0.0

➤ Getting tab stops

This script demonstrates how to get the tab stops for a paragraph.

```
-- Return the tab stops of the first paragraph
tell application "Adobe Illustrator"
   set allTabs to tab stops of paragraph 1 of text frame 1 of document 1
end tell
```

tag, tags

A tag or list of tags associated with a specific page item. Tags allows you to assign an unlimited number of key-value pairs to any page item in a document.

tag object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the tag object; always returns reference.
class	type class	Read-only. The object's class, which is tag.
container	object reference	Read-only. A reference to the page item that contains this tag.
default type	type class	Read-only. The default type for the tag; always returns reference.
index	integer	Read-only. The index of this tag in the page item.
name	Unicode text	The tag's name.
properties	record	All of the properties of this object returned as a record.
value	Unicode text	The data stored in this tag.

tag object commands

count
delete
duplicate
exists
make

➤ Getting tags

This script demonstrates how to get the tags for a page item.

```
-- Get the tags for the first page item in the document
tell application "Adobe Illustrator"
make rectangle in document 1 with properties {name:"rectPath"}
set AdobeURL to make tag of path item "rectPath" of document 1
set name of AdobeURL to "AdobeURLTag"
set value of AdobeURL to "www.adobe.com"
count tags of path item "rectPath" of document 1
get properties of tags of path item "rectPath" of document 1
end tell
```

text

Any text in the contents of a text frame. Text can be accessed using the `character`, `insertion point`, `word`, `line`, `paragraph` and `text` classes. All text is contained within text frames.

text object elements

Elements	Refer to by
`character style`	name, numeric index, range of elements, before/after another element, satisfying a test
`character`	numeric index, range of elements, before/after another element, satisfying a test
`insertion point`	numeric index, range of elements, before/after another element, satisfying a test
`line`	numeric index, range of elements, before/after another element, satisfying a test
`paragraph style`	name, numeric index, range of elements, before/after another element, satisfying a test
`paragraph`	numeric index, range of elements, before/after another element, satisfying a test
`text`	numeric index, range of elements, before/after another element, satisfying a test
`word`	numeric index, range of elements, before/after another element, satisfying a test

text object properties

Property	Value Type	What it is
`best type`	`type class`	Read-only. The best type for the object's value.
`character offset`	`integer`	Offset of the first character.
`class`	`type class`	Read-only. The object's class.
`container`	`reference`	Read-only. The object's container.
`contents`	`Unicode text`	The text content of the text range.
`default type`	`type class`	Read-only. The default type for the object's value.
`index`	`integer`	Read-only. The index of this instance of the object.
`kerning`	`integer`	Controls the spacing between two characters, in thousandths of an em.
`length`	`integer`	Read-only. Length of text range in characters. Minimum: 0
`properties`	`record`	All of the properties of this object returned as a record.
`selection`	`list of text`	Read-only. The selected text (ranges) in the text range.
`story`	`story`	Read-only. The story of the text range.

text object commands

➤ Changing point size of text

In this example, all characters set to 12-point type in the current document will be changed to 18-point type.

```
-- Change all 12pt text to 18pt
tell application "Adobe Illustrator"
   set textArtItemCount to count text frames of document 1
   -- Loop through all the text frames
   repeat with itemCount from 1 to textArtItemCount
      set textRef to text of text frame itemCount of document 1 as reference
      set sizeList to size of textRef
      repeat with indexCount from 1 to length of sizeList
         if (item indexCount of sizeList = 12) then
            set size of character indexCount of text frame itemCount ¬
               of document 1 to 18
         end if
      end repeat
   end repeat
end tell
```

text font, text fonts

An installed font.

text font object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the object's value; always returns reference.
class	type class	Read-only. The object's class, which is text font.
default type	type class	Read-only. The default type for the object; always returns reference.
family	Unicode text	Read-only. The font's family name.
index	integer	Read-only. The index of this object in the art item.
name	Unicode text	The name of the variable.
properties	record	All of the properties of this object returned as a record.
style	Unicode text	Read-only. The font's style name.

text frame, text frames

A text art item or items. From the user interface, this is text created with the Text tool. There are three types of text art in Illustrator, as specified by the text frame's `kind` property. See 'Working with text art' on page 53 for more information on working with the three kinds of text frames.

text frame object elements

Element	Refer to by
`character`	index, before/after, range, test
`insertion point`	index, before/after, range, test
`line`	index, before/after, range, test
`paragraph`	index, before/after, range, test
`text`	index, before/after, range
`word`	index, before/after, range, test

text frame object properties

Note: This class inherits all properties from the `page item` class.

Property	Value type	What it is
`anchor`	`list`	The position of the anchor point (start of base line for point text).
`area`	`real`	Read-only. The area of this path is square points.
`best type`	`type class`	Read-only. The best type for the object's value; always returns `reference`.
`class`	`type class`	Read-only. The object's class, which is `text font`.
`column gutter`	`real`	The column gutter in the text frame (area text only).
`content variable`	`anything`	The content variable to which this `text frame` is bound. It is not necessary to set the type of the `content variable` before binding. Illustrator automatically sets the type to be the same as the `page item` to which it is bound.
`contents`	`Unicode text`	The textual contents of the `text frame`, represented as a string.
`default type`	`type class`	Read-only. The default type for the object; always returns `reference`.
`column count`	`integer`	The column count in the text frame (area text only).
`end T value`	`real`	The end position of text along a path, as a value relative to the path's segments (path text only).

Property	Value type	What it is
flow links horizontally	boolean	If true, the text flows horizontally first between linked frames.
kind	Valid values: point text area text path text	The type of text frame.
matrix	matrix	Read-only. The transformation matrix of the text frame.
next frame	text frame	The linked text frame following this one.
optical alignment	boolean	If true, the optical alignment is active.
previous frame	text frame	The linked text frame preceding this one.
properties	record	All of the properties of this object returned as a record.
row count	integer	The row count in the text frame (area text only).
row gutter	real	The row gutter in the text frame (area text only).
selection	object reference	Read-only. The reference to the text range in this text frame's current selection, if any.
spacing	real	The amount of spacing.
start T value	real	The start position of text along a path, as a value relative to the path's segments (path text only).
story	story	Read-only. The story of the text frame.
text orientation	Valid values: horizontal vertical	The orientation of the text in the frame.
text path	list of path point info	The path points defining the path for the text frame (area and path text).
text range	Unicode text	Read-only. The text range of the text frame.

text frame object commands

transform
translate

➤ Scaling area text frames

This script scales only text frames that are area text, which means they are rectangular regions of text.

```
-- Scale all area text frames to 50% wide
tell application "Adobe Illustrator"
   set textArtItemCount to count text frames in document 1
   repeat with itemCount from 1 to textArtItemCount
      set textKind to kind of text frame itemCount of document 1
      if (textKind = area text) then
         set curwidth to the width of text frame itemCount of document 1
         set width of text frame itemCount of document 1 to curwidth / 2
      end if
   end repeat
end tell
```

➤ Creating and manipulating text frames

```
-- Text Frames
-- Make a new document, one text frame of each type: Area, Point, and Path
-- Display the count of text frames
-- Change the contents of each text frame
-- Delete the point text frame
-- Display the count of text frames

tell application "Adobe Illustrator"
   activate
   make new document
   make new rectangle in document 1 with properties ¬
      {position:{100, 700}, height:100, width:100}
   make new text frame in document 1 with properties ¬
      {name:"AreaText", kind:area text, text path:the result,¬
       contents:"Text Frame 1"}
   set pathPoint1 to {class:path point info, anchor:{250, 700}}
   set pathPoint2 to {class:path point info, anchor:{350, 550}}
   make new path item in document 1 with properties ¬
      {entire path:{pathPoint1, pathPoint2}}
   make new text frame in document 1 with properties ¬
      {name:"PathText", kind:path text, text path:the result, ¬
       contents:"Text Frame 2"}
   make new text frame in document 1 with properties ¬
      {name:"PointText", contents:"Text Frame 3"}
   set the position of text frame "PointText" of document 1 to {400, 700}
   delay 1
   set user interaction level to interact with all
   display dialog ("Text Frame Count: " & (count text frames of document 1)¬
      as string)
   set the contents of text frame "AreaText" of document 1 ¬
      to "Area Text is cool"
   set the contents of text frame "PathText" of document 1 ¬
      to "Path Text is cooler"
   set the contents of text frame "PointText" of document 1 ¬
      to "Point Text is not"
```

```
    delay 1
    delete text frame "PointText" of document 1
    delay 1
    display dialog ("Text Frame Count: " & (count text frames of document 1)¬
       as string)
    set user interaction level to never interact
end tell
```

text path item, text path items

A path or list of paths for area or path text. A path consists of path points that define its geometry.

text path item object elements

Element	Refer to by
`path point`	index, range of elements, before/after another element, satisfying a test

text path item object properties

Note: This object class inherits all properties from the `page item` class.

Property	Value type	What it is
`area`	`real`	Read-only. The area of this path in square points. An area may be negative or even 0. The paths winding order is determined by the sign of area. If the area is negative, the path is wound counter-clockwise. Self-intersecting paths may contain sub-areas that cancel each other out. Therefore, it is possible for a path's area to appear as zero even though it has apparent area.

`blend mode`	Valid values:
	`color blend`
	`color burn`
	`color dodge`
	`darken`
	`difference`
	`exclusion`
	`hard light`
	`hue`
	`lighten`
	`luminosity`
	`multiply`
	`normal`
	`overlay`
	`saturation blend`
	`screen`
	`soft light`
		The mode to use when compositing this object. An object is considered composited when its opacity is set to less than 100.0 (100%).

`clipping`	`boolean`	If `true`, use this path as a clipping path.
`closed`	`boolean`	If `true`, this path closed.
`container`	`reference`	Read-only. A reference to the art item that contains this path.
`editable`	`boolean`	If `true`, this path can be modified.
`entire path`	`list (of path point info)`	All the path item's path points.
`evenodd`	`boolean`	If `true`, use the even-odd rule to determine insideness.
`fill color`	`color info`	The fill color of the path.

Property	Value type	What it is
fill overprint	boolean	If true, the art beneath a filled object should be overprinted.
filled	boolean	If true, the path should be filled.
guides	boolean	If true, this path is a guide object.
height	real	The height of the path in points. Range: 0.0 to 16348.0
note	Unicode text	The note text assigned to the path.
opacity	real	The object's opacity, expressed as a percentage. Range: 0.0 to 100.00
polarity	Valid values: positive negative	The polarity of the path, used in the creation of compound paths.
position	list	The position of the top left corner of the text path.
resolution	real	The resolution of the path in dots per inch.
selected path points	list (of object references)	Read-only. All of the selected path points in the path.
stroke cap	Valid values: butted rounded projecting	The type of line capping.
stroke color	color info	The stroke color for the path.
stroke dash offset	real	The default distance into the dash pattern at which the pattern should be started
stroke dashes	list (of real numbers)	The lengths for dashes and gaps in dashed lines, starting with the first dash length, followed by the first gap length, and so on. Set to an empty list, {}, for a solid line.
stroke join	Valid values: mitered rounded beveled	Type of join for the path.
stroke miter limit	real	When a default stroke join is set to mitered, this property specifies when the join will be converted to beveled (squared-off) by default. The default miter limit of 4 means that when the length of the point reaches four times the stroke weight, the join switches from a miter join to a bevel join. Values: 1 to 500. 1 specifies a bevel join.
stroke overprint	boolean	If true, the art beneath the stroked object should be overprinted.
stroke width	real	Width of stroke.

Property	Value type	What it is
fill overprint	boolean	If true, the art beneath a filled object should be overprinted.
filled	boolean	If true, the path should be filled.
guides	boolean	If true, this path is a guide object.
height	real	The height of the path in points. Range: 0.0 to 16348.0
note	Unicode text	The note text assigned to the path.
opacity	real	The object's opacity, expressed as a percentage. Range: 0.0 to 100.00
polarity	Valid values: positive negative	The polarity of the path, used in the creation of compound paths.
position	list	The position of the top left corner of the text path.
resolution	real	The resolution of the path in dots per inch.
selected path points	list (of object references)	Read-only. All of the selected path points in the path.
stroke cap	Valid values: butted rounded projecting	The type of line capping.
stroke color	color info	The stroke color for the path.
stroke dash offset	real	The default distance into the dash pattern at which the pattern should be started
stroke dashes	list (of real numbers)	The lengths for dashes and gaps in dashed lines, starting with the first dash length, followed by the first gap length, and so on. Set to an empty list, {}, for a solid line.
stroke join	Valid values: mitered rounded beveled	Type of join for the path.
stroke miter limit	real	When a default stroke join is set to mitered, this property specifies when the join will be converted to beveled (squared-off) by default. The default miter limit of 4 means that when the length of the point reaches four times the stroke weight, the join switches from a miter join to a bevel join. Values: 1 to 500. 1 specifies a bevel join.
stroke overprint	boolean	If true, the art beneath the stroked object should be overprinted.
stroke width	real	Width of stroke.

Property	Value type	What it is
stroked	boolean	If true, the path should be stroked.
width	real	The width of the text path in points. Range: 0.0 to 16348.0

text path item object commands

count
delete
duplicate
exists
move
rotate
scale
transform
translate

tracingobject, tracings

Associates source raster art item with a vector-art plugin group created by tracing. Scripts can initiate tracing using the `trace placed` command for a `placed item` or `raster item`. The resulting `plugin item` object represents the vector art group, and has this object in its `tracing` property.

A script can force the tracing operation by calling the document's `redraw` command. The operation is asynchronous, so a script should call `redraw` after creating the tracing object, but before accessing its properties or expanding the tracing to convert it to an art item group.

The read-only properties that describe the tracing result have valid values only after the first tracing operation completes. A value of 0 indicates that the operation has not yet been completed.

tracingobject object properties

Property	Value type	What it is
anchor count	integer	Read-only. The number of anchors in the tracing result.
area count	integer	Read-only. The number of areas in the tracing result.
best type	type class	Read-only. The best type for the object's value; always returns `reference`.
class	type class	Read-only. The object's class, which is `text font`.
container	object reference	Read-only. A reference to the object that contains this tracing group.
default type	type class	Read-only. The default type for the object; always returns `reference`.
image resolution	real	Read-only. The resolution of the source image in pixels per inch.
original art	placed item or raster item object	Read-only. The raster art used to create the associated vector-art plugin group.
path count	integer	Read-only. The number of paths in the tracing result.
properties	record	All of the properties of this object returned as a record.
tracing options	tracing options object	Read-only. The options used to convert the raster artwork to vector art.
used color count	integer	Read-only. The number of colors used in the tracing result.

tracingobject object commands

expand tracing
release tracing

tracing options, multiple tracing options

A set of options used in converting raster art to vector art by tracing.

tracing options object properties

Property	Value type	What it is
best type	type class	Read-only. The best type for the object's value; always returns reference.
class	type class	Read-only. The object's class, which is text font.
container	object reference	Read-only. A reference to the object that contains this tracing group.
corner angle	real	The sharpness, in degrees of a turn in the original image that is considered a corner in the tracing result path. Range: 0 to 180
default type	type class	Read-only. The default type for the object; always returns reference.
fills	boolean	If true, trace with fills. At least one of fills or strokes must be true.
live paint output	boolean	If true, result is LivePaint art. If false, it is classic art. **Note:** A script should only set this value in preparation for a subsequent expand operation. Leaving a tracing on the artboard when this property is true can lead to unexpected application behavior.
maximum colors	integer	The maximum number of colors allowed for automatic palette generation. Used only if tracing mode is color or grayscale. Range: 2 to 256
maximum stroke weight	real	The maximum stroke weight, when strokes is true. Range: 0.01 to 100.0
minimum area	integer	The smallest feature, in square pixels, that is traced. For example, if it is 4, a feature of 2 pixels wide by 2 pixels high is traced.
minimum stroke length	real	The minimum length in pixels of features in the original image that can be stroked, when strokes is true. Smaller features are omitted. Range: 0.0 to 200.0 Default: 20.0

Property	Value type	What it is
output swatches	boolean	If true, named colors (swatches) are generated for each new color created by the tracing result. Used only if tracing mode is color or grayscale.
palette	string	The name of a color palette to use for tracing. If the empty string, use the automatic palette. Used only if tracing mode is color or grayscale.
path fitting	real	The distance between the traced shape and the original pixel shape. Lower values create a tighter path fitting. Higher values create a looser path fitting. Range: 0.0 to 10.0
preprocess blur	real	The amount of blur used during preprocessing. Blurring helps reduce small artifacts and smooth jagged edges in the tracing result. Range: 0.0 to 2.0
preset	string	Read-only. The name of a preset file containing these options.
properties	record	All of the properties of this object returned as a record.
resample	boolean	If true, resample when tracing. (This setting is not captured in a preset file.) Always true when the raster source art is placed or linked.
resample resolution	real	The resolution to use when resampling in pixels per inch (ppi). Lower resolution increases the speed of the tracing operation. (This setting is not captured in a preset file.)
strokes	boolean	If true, trace with strokes. At least one of fills or strokes must be true. Used only if tracing mode is black-and-white.
threshold	integer	The threshold value of black-and-white tracing. All pixels with a grayscale value greater than this are converted to black. Used only if tracing mode is black-and-white. Range: 0 to 255
tracing mode	Valid values: bw tracing mode color tracing mode gray tracing mode	The color mode for tracing.

Property	Value type	What it is
view raster	Valid values: view adjusted image view no image view original image view transparent image	The view for previews of the raster image. (This setting is not captured in a preset file.)
view vector	Valid values: view no tracing result view outlines view outlines tracing view tracing result	The view for previews of the vector result. (This setting is not captured in a preset file.)

tracing options object commands

load preset
store preset

variable, variables

A class of variables that can be imported and exported. Variables are document-level, created in the document object.

variable object elements

Element	Refer to by
`page item`	name, numeric index, range of elements, before/after another element, satisfying a test

variable object properties

Property	Value type	What it is
`best type`	`type class`	Read-only. The best type for the `variable` object's value; always returns `reference`.
`class`	`type class`	Read-only. The object's class, which is `variable`.
`container`	`object reference`	Read-only. A reference to the art item that contains this variable.
`default type`	`type class`	Read-only. The default type for the variable; always returns `reference`.
`index`	`integer`	Read-only. The index of this `variable` in the art item.

`kind`	Valid values:
	`graph`
	`image`
	`textual`
	`unknown`
	`visibility`
		The kind of variable.

`name`	`Unicode text`	The name of the variable.
`properties`	`record`	All of the properties of this object returned as a record.

variable object commands

`count`
`delete`
`exists`
`make`

view, views

A document view or list of views in an Illustrator document. The `view` object represents a window view onto a document. Scripts cannot create new views, but can modify some properties of existing views, including the center point, screen mode, and zoom.

view object properties

Property	Value type	What it is
best type	`type class`	Read-only. The best type for the `view` object; always returns `reference`.
bounds	`rectangle`	Read-only. The bounding rectangle of this `view` relative to the current document's bounds
center point	`fixed point`	The center point of this `view` relative to the current document's bounds
class	`type class`	Read-only. The object's class, which is `view`.
container	`object reference`	Read-only. A reference to the document that contains this `view`.
default type	`type class`	Read-only. The default type for the `view` object; always returns `reference`.
index	`integer`	Read-only. The index of the view in the document.
properties	`record`	All of the properties of this object returned as a record.

screen mode	Valid values:
	`multiwindow`
	`desktop`
	`full screen`
		The mode of display for this `view`.

zoom	`real`	The zoom factor of this `view`, where 1.0 is 100%.

view object commands

count
exists

➤ Centering a view

This example demonstrates how a view can be centered to the currently selected page item.

```
-- Center the view on the first selected object
tell application "Adobe Illustrator"
   set selectedItems to the selection
   if selectedItems is not {} then
      set firstObject to item 1 of selectedItems
      set newPosition to position of firstObject
      set center point of view 1 of document 1 to newPosition
   end if
end tell
```

➤ Making a view full screen

This example shows how a view can be toggled to fill the entire screen.

```
-- Fill the entire screen with the first view
tell application "Adobe Illustrator"
   if (count documents) > 0 then
      set screen mode of view 1 of document 1 to full screen
   end if
end tell
```

word

A string of text in a `text frame` that is separated by whitespace. A document's text can be accessed using the `character`, `insertion point`, `word`, `line`, `paragraph`, and `text` classes. All text is contained within `text frames`.

word object elements

Elements	Refer to by
`character style`	name, numeric index, range of elements, before/after another element, satisfying a test
`character`	numeric index, range of elements, before/after another element, satisfying a test
`insertion point`	numeric index, range of elements, before/after another element, satisfying a test
`line`	numeric index, range of elements, before/after another element, satisfying a test
`paragraph style`	name, numeric index, range of elements, before/after another element, satisfying a test
`paragraph`	name, numeric index, range of elements, before/after another element, satisfying a test
`text`	numeric index, range of elements, before/after another element, satisfying a test
`word`	name, numeric index, range of elements, before/after another element, satisfying a test

word object properties

Property	Value Type	What it is
`aki left`	`real`	The amount of extra space (aki) added to the left side of each glyph in Japanese text (in thousandths of an em).
`aki right`	`real`	The amount of extra space (aki) added to the right side of each glyph in Japanese text (in thousandths of an em).

`alignment`	Valid values:
	`bottom`
	`center`
	`icf bottom`
	`icf top`
	`roman baseline`
	`top`
		The character alignment type.

Property	Value Type	What it is

alternate glyphs	Valid values:
	`default`
	`expert`
	`full width`
	`half width`
	`jis78`
	`jis83`
	`proportional width`
	`quarter width`
	`third width`
	`traditional`
		Specifies the type of alternate glyphs.

auto leading	`boolean`	If `true`, use automatic leading.

baseline direction	Valid values:
	`standard`
	`Tate Chu Yoko`
	`vertical rotated`
		Specifies the Japanese text baseline direction.

baseline position	Valid values:
	`normal`
	`subscript`
	`superscript`
		The baseline position of text.

baseline shift	`real`	The amount of shift (in points) of the text baseline.
best type	`type class`	Read-only. The best type for the object's value.

capitalization	Valid values:
	`all caps`
	`all small caps`
	`normal`
	`small caps`
		Specifies whether the text is normal, all uppercase, all small caps, or a mix of small caps and lowercase.

character offset	`integer`	Offset of the first character.
class	`type class`	Read-only. The object's class.
connection forms	`boolean`	If `true`, use the OpenType connection forms.
container	`reference`	Read-only. The object' s container.
contents	`Unicode text`	The text string content of the text range.
contextual ligature	`boolean`	If `true`, use the contextual ligature.
default type	`type class`	Read-only. The default type for the object's value.
discretionary ligature	`boolean`	If `true`, use the discretionary ligature.

Property	Value Type	What it is

figure style	Valid values:
	`default`
	`proportional`
	`proportional oldstyle`
	`tabular`
	`tabular oldstyle`
		Specifies which figure style to use in an OpenType font.

fill color	`color info`	The color of the text fill.
fractions	`boolean`	If `true`, use the OpenType fractions.
horizontal scale	`real`	The character horizontal scaling factor expressed as a percentage (100 = 100%).
index	`integer`	Read-only. The index of this instance of the object.
italics	`boolean`	If `true`, the Japanese font supports italics.
kerning	`integer`	Controls the spacing between two characters, in thousandths of the em space.

kerning method	Valid values:
	`auto`
	`none`
	`optical`
		The type of automatic kerning method to use.

Property	Value Type	What it is
language	Valid values: Bokmal Norwegian Brazillian Portuguese Bulgarian Canadian French Catalan Chinese Czech Danish Dutch English Finnish Greek Hungarian Icelandic Italian Japanese Nynorsk Norwegian old German Polish Romanian Russian Spanish Serbian standard French standard German standard Portuguese Swedish Swiss German Turkish UK English Ukranian	The language.
leading	real	The amount of space between two lines of text, in points.
length	integer	Read-only. Length of text range in characters. Minimum: 0
ligature	boolean	If true, use the ligature.
no break	boolean	If true, no line break is allowed in this word.
OpenType position	Valid values: default denominator numerator subscript superscript	The OpenType baseline position.
ordinals	boolean	If true, use the OpenType ordinals.
ornaments	boolean	If true, use the OpenType ornaments.
overprint fill	boolean	If true, overprint the fill of the text.
overprint stroke	boolean	If true, overprinting of the stroke of the text is allowed.

Property	Value Type	What it is
properties	record	All of the properties of this object returned as a record.
proportional metrics	boolean	If true, the Japanese OpenType supports proportional fonts.
rotation	real	The character rotation angle in degrees.
selection	list of text	Read-only. The selected text ranges in the text range.
size	real	The font size in points.
story	story	Read-only. The story of the text range.
strike through	boolean	If true, characters use strike-through style.
stroke color	color info	The color of the text stroke.
stroke weight	real	Line width of stroke.
stylistic alternates	boolean	If true, use OpenType stylistic alternates.
swash	boolean	If true, use the OpenType swash character.
TCY horizontal	integer	The Tate-Chu-Yoko horizontal adjustment in points.
TCY vertical	integer	The Tate-Chu-Yoko vertical adjustment in points.
text font	text font	The text font.
titling	boolean	If true, use the OpenType titling alternates.
tracking	integer	The tracking or range kerning amount in thousandths of an em.
Tsume	real	The percentage of space reduction around a Japanese character.
underline	boolean	If true, characters use underline style.
vertical scale	real	Character vertical scaling factor. 100 = 100%
warichu characters after break	integer	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.

Property	Value Type	What it is
warichu characters before break	integer	How the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.
warichu enabled	boolean	If true, Wari-Chu is enabled.
warichu gap	integer	The Wari-Chu line gap.
warichu justification	Valid values: auto justify center full justify last line center full justify full justify last line left full justify last line right left right	The Wari-Chu justification.
warichu lines	integer	The number of Wari-Chu (multiple text lines fit into a space meant for one) lines.
warichu scale	real	The Wari-Chu scale.

word object commands

apply character style
apply paragraph style
change case
count
delete
deselect
duplicate
exists
make
select

➤ Finding specific words

This example demonstrates how to use the matching abilities of the whose clause in conjunction with word properties to modify words that match a specific string.

```
-- Change the color of every occurrence of a specific
-- word in all text frames
set searchString to text returned of
   (display dialog "Word to set color of?" default answer "the")
tell application "Adobe Illustrator"
   set textArtItemCount to (count text frames in document 1)
   if (textArtItemCount > 0) then
      repeat with itemCounter from 1 to textArtItemCount
         if (((contents of text frame itemCounter of document 1) as string)¬
            contains searchString) then
               set fill color of ¬
                  (words of text frame itemCounter of document 1 ¬
                   whose contents = searchString) to ¬
```

```
                    {red:100, green:0, blue:0}
            end if
        end repeat
    end if
end tell
```

5 AppleScript Commands

This chapter provides a complete reference for the commands in the Illustrator AppleScript dictionary. The commands are presented alphabetically.

The commands supported by each object, with links to the detailed descriptions here, are listed in "AppleScript Objects" on page 61.

Overview

This chapter describes the commands in the Illustrator AppleScript dictionary, as well as some of the important standard AppleScript commands. The AppleScript dictionary itself shows only that the command returns an object, or that the command takes an object reference as a parameter; it does not show the specific objects that can respond to a particular command. Not all Illustrator objects can respond to all commands; this reference details which objects respond to which commands, and what type of object each command returns (if any).

The following information is given for each command.

Column heading	What it means
Parameters	Constants, keywords, and values needed by the command. • Variable values to be supplied are shown in bold. • Literal terms and constants are shown in plain type. • Items surrounded by brackets [ ] are optional.
What it is	An explanation of the parameters.
Objects supported	Which objects support the command and/or which objects the command can operate upon. The `document` object, for example, supports the command `close`, but not the command `quit`.
Returns	Many commands return values (text, numbers, lists, and object references). This column shows you what kind of reference you can expect the command to return (if any).

activate

Makes an application active; that is, makes Illustrator the frontmost application.

Parameters	What it is	Objects supported	Returns
none		`application`	nothing

Notes

Illustrator must be the frontmost application when executing any command that deals with the clipboard. Use this command to ensure this. See the clipboard commands for examples.

apply

Applies a brush or graphic style to one or more page items.

Parameters	What it is	Objects supported	Returns
object reference	The brush or graphic style to apply to the referenced objector objects.	graphic style brush	nothing
to **anything**	The page item or items to which to apply a brush or graphic style.	compound path item group item mesh item page item path item placed item plugin item raster item text frame	

Notes

Use `apply` to affect one or more page items by applying an existing brush or graphic style. Brushes and graphic styles can be created in the user interface, but not from a script.

➤ Applying an art style

```
-- Draws a circle in the center of the document
-- and applies an art style to it
tell application "Adobe Illustrator"
   make new document with properties {color space:CMYK}
   set docWidth to (width of document 1) / 2
   set docHeight to (height of document 1) / 2
   set pathItemRef to make new ellipse in document 1 with properties ¬
      {bounds:{docWidth - 50, docHeight + 50, docWidth + 50, docHeight - 50}}
   apply graphic style 2 of document 1 to pathItemRef
end tell
```

apply character style

Applies a character style to a specified text object(s).

Parameters	What it is	Objects supported	Returns
character style	The character style object or objects to be operated upon.	character style	nothing
to **anything**	The text object or objects to which to apply the style.	text	
[clearing overrides **boolean**]	Whether to clear any text attributes before apply the style. Default: `false`		

apply paragraph style

Applies the paragraph style to text object(s).

Parameters	What it is	Objects supported	Returns
`paragraph style`	The paragraph style object or objects to be operated upon.	`paragraph style`	nothing
`to` **`anything`**	The text object or objects to which to apply the style.	`text`	
`[clearing overrides` **`boolean`**`]`	If `true`, text attributes are cleared before apply the style. Default: `false`		

change case

Changes the capitalization of the selected text.

Parameters	What it is	Objects supported	Returns
text	The text object or objects to be operated upon.	text	nothing
to lower case/ sentence case/ title case/ upper case	The type of case.		

close

Closes a document.

Parameters	What it is	Objects supported	Returns
document	The document to close.	document	nothing
[saving yes/no/ask]	Whether to save the document before closing.		

➤ Closing a document

```
-- Close the first document and prompt the user with a "Save as" dialog
tell application "Adobe Illustrator"
   activate
   close document 1 saving ask
end tell
```

colorize

Colorizes a raster item.

Parameters	What it is	Objects supported	Returns
object reference	The raster item to colorize.	raster item	nothing
raster color **color info reference**	The color to use when coloring the TIFF image.	CMYK color info/ gradient color info/ gray color info/ pattern color info/ RGB color info/ spot color info	

concatenate matrix

Concatenates two transformation matrices to form a single resulting matrix.

Parameters	What it is	Objects supported	Returns
matrix	The first matrix.	matrix	matrix
with **matrix**	The second matrix.	matrix	

➤ **Concatenating matrices**

```
-- This script concatenates 2 matrices
tell application "Adobe Illustrator"
   set someMatrix to get identity matrix
   set anotherMatrix to get rotation matrix angle 30.0
   set newMatrix to concatenate matrix someMatrix with anotherMatrix
end tell
```

concatenate rotation matrix

Concatenates a rotation angle together with a matrix and returns the resulting matrix.

Parameters	What it is	Objects supported	Returns
matrix	The matrix.	matrix	matrix
angle **real**	Rotation angle in degrees.		

➤ **Rotating matrices**

```
-- This script adds a 45 degree rotation to an existing matrix
tell application "Adobe Illustrator"
   set someMatrix to get identity matrix
   set newMatrix to concatenate rotation matrix someMatrix angle 45.0
end tell
```

concatenate scale matrix

Concatenates a horizontal and/or vertical scaling with a matrix to form a new, rescaled matrix.

Parameters	What it is	Objects supported	Returns
matrix	The matrix.	matrix	matrix
[horizontal scale **real**]	Horizontal scaling factor, 100.0 is 100%. Default: 100.0		
[vertical scale **real**]	Vertical scaling factor, 100.0 is 100%. Default: 100.0		

➤ **Scaling a matrix**

```
-- This script combines a 75% horizontal scaling with an existing matrix
```

```
tell application "Adobe Illustrator"
   set someMatrix to get identity matrix
   set newMatrix to concatenate scale matrix someMatrix ¬
      horizontal scale 75 vertical scale 25.0
end tell
```

concatenate translation matrix

Concatenates a positional translation factor (specified by a horizontal and/or vertical offset) with a matrix to form a new, repositioned matrix.

Parameters	What it is	Objects supported	Returns
matrix	The matrix.	matrix	matrix
[delta x **real**]	Horizontal translation offset. Default: 0.0		
[delta y **real**]	Vertical translation offset. Default: 0.0		

➤ Translating a matrix position

```
--This script combines a 25 point horizontal offset with an existing matrix
tell application "Adobe Illustrator"
   set someMatrix to get identity matrix
   set newMatrix to concatenate translation matrix someMatrix delta x 25.0
end tell
```

convert

Creates a native text frame from a legacy text item. The original legacy text item is deleted.

Parameters	What it is	Objects supported	Returns
legacy text item	The legacy text item object or objects to be operated upon.	legacy text item	group item

convert to paths

Converts a text item to path items. Creates an outline for the frame text.

Parameters	What it is	Objects supported	Returns
text frame	The text frame object or objects to be operated upon.	text frame	group item or null

➤ **Creating outlines for text frames**

```
--This script converts all text art
tell application "Adobe Illustrator"
   convert to paths (every text frame of document 1)
end tell
```

copy

Copies the selection in the current document to the clipboard.

Parameters	What it is	Objects supported	Returns

none		`compound path item`
		`group item`
		`mesh item`
		`path item`
		`placed item`
		`plugin item`
		`raster item`
		`text frame`
			nothing

Notes

Commands that manipulate the clipboard (`cut`, `copy`, and `paste`) require that Illustrator be the frontmost application during these operations. Use `activate` to bring Illustrator to the front before executing the `copy` command. No error is returned if there is no selection to copy. If the application is not frontmost, an error is returned.

➤ Copying selected objects

```
--This script copies the selected objects (if any)
tell application "Adobe Illustrator"
   activate
   copy
end tell
```

count

Counts the elements of a specified type contained in a specified object.

Parameters	What it is	Objects supported	Returns

`count` **`reference`**	The object or list of objects whose elements are to be counted.	`graphic style`
		`brush`
		`character`
		`compound path item`
		`document`
		`gradient`
		`gradient stop`
		`group item`
		`insertion point`
		`layer`
		`line`
		`mesh item`
		`page item`
		`paragraph`
		`path item`
		`path point`
		`pattern`
		`placed item`
		`plugin item`
		`raster item`
		`spot`
		`tag`
		`text frame`
		`view`
		`word`
			`integer`

[`each` **`type class`**]	The class of the objects to count.		
[`whose` **`property`** `is` **`value`**]	A condition that objects must meet to be counted.		

Notes

With the optional `each/every` term, use the singular form for the object type to be counted; for example, `brush` rather than `brushes`. Otherwise, you can use the singular or plural form.

➤ Counting filled path items in a document

```
-- This script shows the user how many paths
-- are filled out of the total number in document 1
tell application "Adobe Illustrator"
   set pathCount to count every path item of document 1
   set numberFilled to ¬
      count (path items of document 1 whose filled is true)
end tell
display dialog numberFilled & " of " & pathCount ¬
   & " paths are filled in this document." as string
```

cut

Cuts the current selection from the current document and places it in the clipboard.

Parameters	What it is	Objects supported	Returns
none	nothing	compound path item group item mesh item path item path point placed item plugin item raster item text text frame	nothing

Notes

Commands that manipulate the clipboard (`cut`, `copy`, and `paste`) require that Illustrator be the frontmost application. Use `activate` to bring Illustrator to the front before executing the `cut` command. No error is returned if there is no selection to cut. If the application is not frontmost, an error is returned.

➤ Cutting selected objects to the clipboard

```
--This script cuts the selected objects (if any)
tell application "Adobe Illustrator"
   activate
   cut
end tell
```

delete

Removes one or more elements from a container, or deletes one or more objects.

Parameters	What it is	Objects supported	Returns
object reference	Contained object or objects to delete or remove.	compound path item gradient gradient stop group item layer mesh item page item point pattern placed item plugin item raster item spot swatch tag text text frame text path	nothing
[of **object reference**]	Optional. Container object. If supplied, removes the specified object or objects from this container. If not supplied, deletes the specified object or objects.	document group layer compound path item path item story	

➤ Deleting a layer

```
-- This script deletes the second layer in the document
tell application "Adobe Illustrator"
   if (count layers of document 1) > 1 then
      delete layer 2 of document 1
   end if
end tell
```

deselect

Deselects a text range.

Parameters	What it is	Objects supported	Returns
text	The text object or objects to be deselected.	text	nothing

display

Displays the dynamic data that has been captured in a `dataset` object.

Parameters	What it is	Objects supported	Returns
`dataset`	The `dataset` object or objects to be displayed.	`dataset`	`boolean`

do javascript

Executes a JavaScript script and returns the result of execution.

Parameters	What it is	Objects supported	Returns
anything	JavaScript code to execute.	N/A	Unicode text
[with arguments **list of anything**]	A list of suitable arguments to pass to the Javascript routine.		
[show debugger before running/ never/ on runtime error]	When a debugger should be shown. Default is never.		

do script

Plays an action from the Actions palette.

Parameters	What it is	Objects supported	Returns
Unicode text	The name of the action to play. Case-sensitive.	N/A	nothing
from **Unicode text**	The name of the Action Set containing the action. Case-sensitive.		
[dialogs **boolean**]	If true, dialog boxes should be associated with the action presented to the user. Default is true.		

Notes

If the action is selected in the Actions palette in the Illustrator user interface, this command returns an error.

➤ Executing an action

```
-- Executes an action in the default set without displaying any dialogs
tell application "Adobe Illustrator"
   do script "Opacity 60 (selection)" from "Default Actions" without dialogs
end tell
```

duplicate

Duplicates an object or objects.

Parameters	What it is	Objects supported	Returns

`object reference`	The object or objects to duplicate	all objects *except*:
		`application`
		`mesh item`
		`plugin items`
			`object reference` or `list (of object references)`

`[to` **`location reference`**`]`	The location for the new object or objects		
`[with properties` **`record`**`]`	New values for specified properties of the new object or objects		

Notes

You can duplicate page items from one document to another. This is equivalent to setting the selection, performing a cut or copy, bringing another document to the front, and then pasting. When duplicating objects from one document to another, you must specify the location reference.

➤ Duplicating to another document

```
-- Duplicate the first page item in document 1 to document 2
tell application "Adobe Illustrator"
   set pageItemRef to duplicate page item 1 of document 1 ¬
      to beginning of document 2
end tell
```

equal matrices

Compares two matrices for equality.

Parameters	What it is	Objects supported	Returns
matrix	The first matrix for the comparison.	matrix	boolean
with **matrix**	The second matrix for the comparison.		

➤ Comparing matrices

```
-- This script compares 2 matrices and beeps if they are equal
tell application "Adobe Illustrator"
   set someMatrix to get identity matrix
   set anotherMatrix to get identity matrix
   if (equal matrices someMatrix with anotherMatrix) then beep
end tell
```

embed

Embeds art in a document. Applied to a `placed item`, it converts the art to art item objects as needed and deletes the `placed item` object.

Parameters	What it is	Objects supported	Returns
`object reference`	The placed item to embed.	`placed item`	nothing

exists

Determines whether an object exists.

Parameters	What it is	Objects supported	Returns
`object reference`	The object to test for existence.	Any object except `application`	`boolean`

➤ Check if a document exists

```
-- Check if a document exists and beep twice if one does
tell application "Adobe Illustrator"
   if exists document 1 then beep 2
end tell
```

expand tracing

Converts the vector art associated with a `tracingobject` into a new group item. The new `group item` object replaces the `plugin item` object in the document. Deletes this object and its associated `plugin item` object. Any group-level attributes that were applied to the plugin item are applied to the top level of the new group item.

Parameters	What it is	Objects supported	Returns
`tracingobject`	The `tracingobject` object to operate on.	`tracingobject`	`group item object reference`
`with viewed`	By default the new group contains only the tracing result (the filled or stroked paths). If `with viewed` is specified, the new group retains additional information that was specified for the viewing mode, such as outlines and overlays.		

export

Exports the specified document to a specified file type.

Parameters	What it is	Objects supported	Returns
object reference	The document to export.	document	nothing
to **file specification**	The file to export to, specified as a string containing the full file path or an alias.		
as JPEG/Photoshop/ SVG/PNG8/PNG24/ GIF	The file type to which to export the document.		
with options **anything**	The export options for the specified file type.		

➤ **Exporting a document to JPEG**

```
-- This script exports the current document as JPEG
set newFilePath to (path to desktop folder as string) & "Sample.jpg"
tell application "Adobe Illustrator"
   export current document to newFilePath as JPEG with options ¬
      {class:JPEG export options, quality:60}
end tell
```

export PDF preset

Exports PDF presets for a document and saves them to a file.

Parameters	What it is	Objects supported	Returns
document	The document object or objects to be operated on.	document	nothing
to **file specification**	The file to export to, specified as a string containing the full file path or an alias.		

export print preset

Exports Illustrator print presets for a document to a file.

Parameters	What it is	Objects supported	Returns
document	The document object or objects to be operated on.	document	nothing
to **file specification**	The file to export to, specified as a string containing the full file path or an alias.		

export variables

Saves datasets containing variables and their associated dynamic data into an XML library.

Parameters	What it is	Objects supported	Returns
document	The document object or objects to be operated on.	document	nothing
to **file specification**	The file to export to, specified as a string containing the full file path or an alias.		

get

Gets data from an object.

Parameters	What it is	Objects supported	Returns
object reference *or* **property**	The object or property to get a reference to or data from.	Any object	The property value or object reference as the specified type.
[as **class** *or* list (of **classes**)]	The type of data to retrieve.		

Notes

This standard AppleScript command is included because it illustrates AppleScript's ability to coerce values from one value type to another. You do not need to use `get` to assign values to variables.

➤ Using the get command

```
-- Get the contents of a text frame as a string
tell application "Adobe Illustrator"
set textString to contents of text frame 1 of document 1
set theContent to (get textString)
display dialog "The content is " & theContent
end tell
```

get identity matrix

Returns an identity matrix.

Parameters	What it is	Objects supported	Returns
none	nothing	matrix	matrix

Notes

The identity matrix is a transformation matrix that causes no transformation. Use it to get a base matrix to use with the matrix concatenation commands.

➤ **Using an identity matrix**

```
-- This script gets the identity matrix,
-- combines with rotation and scale and applies to object
tell application "Adobe Illustrator"
   set transformMatrix to get identity matrix
   set transformMatrix to concatenate rotation matrix ¬
      transformMatrix angle 45.0
   set transformMatrix to concatenate scale matrix ¬
      transformMatrix horizontal scale 60
   transform page item 1 of document 1 using transformMatrix
end tell
```

get rotation matrix

Returns a rotation matrix based on a specified rotation angle.

Parameters	What it is	Objects supported	Returns
[angle **real**]	The rotation angle in degrees. Default is 0.0, which returns the standard identity matrix.	matrix	matrix

➤ **Getting a rotation matrix**

```
-- Get a 30-degree rotation matrix
tell application "Adobe Illustrator"
   set rotateMatrix to get rotation matrix angle 30.0
end tell
```

get scale matrix

Returns a scale matrix based on specified horizontal and vertical scaling factor.

Parameters	What it is	Objects supported	Returns
[horizontal scale **real**]	The horizontal scaling factor as a percentage. Default is 100.0, which is 100%	matrix	matrix
[vertical scale **real**]	The vertical scaling factor. Default is 100.0, which is 100%		

Notes

If no parameters are supplied, returns the standard identity matrix.

➤ **Getting a scale matrix**

```
-- This script gets a scale matrix
tell application "Adobe Illustrator"
   set scaleMatrix to get scale matrix horizontal scale 100.0 ¬
      vertical scale 50.0
end tell
```

get translation matrix

Returns a translation matrix based on a single movement with horizontal and vertical offsets.

Parameters	What it is	Objects supported	Returns
[delta x **real**]	The horizontal offset. Default: 0.0	matrix	matrix
[delta y **real**]	The vertical offset. Default: 0.0		

Notes

If no parameters are supplied, returns the standard identity matrix.

➤ **Getting a translation matrix**

```
-- This script gets a translation matrix
tell application "Adobe Illustrator"
   set translateMatrix to get translation matrix delta x 10.0 delta y 100.0
end tell
```

import character styles

Loads character styles from a file.

Parameters	What it is	Objects supported	Returns
document	The document object or objects to be operated on.	document	nothing
from **file specification**	File from which to import.		

import paragraph styles

Loads paragraph styles from a file.

Parameters	What it is	Objects supported	Returns
document	The document object or objects to be operated on.	document	nothing
from **file specification**	File from which to import.		

import PDF preset

Loads all PDF presets from a file.

Parameters	What it is	Objects supported	Returns
document	The document object or objects to be operated on.	document	nothing
from **file specification**	File from which to import.		
{replacing preset **boolean**]	Whether existing editable presets should be replaced. Default is false		

import print preset

Loads a print preset from a file.

Parameters	What it is	Objects supported	Returns
document	The document object or objects to be operated on.	document	nothing
print preset **Unicode text**	The name of the print preset to import.		
from **file specification**	The file to import from, specified as a string containing the full file path or an alias.		

import variables

Loads a library from a file that contains datasets, variables, and the associated dynamic data. The imported data overwrites any existing variables and datasets.

Parameters	What it is	Objects supported	Returns
document	The document object or objects into which to import variables	document	nothing
from **file specification**	The file from which to import variables, specified as a string containing the full file path or an alias.		

invert matrix

Returns an inverted matrix.

Parameters	What it is	Objects supported	Returns
matrix	The matrix to invert.	matrix	matrix

Notes

A singular matrix cannot be inverted. Use the `singular matrix` command to test if a matrix is singular.

➤ Inverting a matrix

```
-- This script gets the inverse matrix of a 50% vertical scale matrix
-- When applied, the inverse matrix scales the object 200% vertically
tell application "Adobe Illustrator"
   set transformMatrix to get scale matrix vertical scale 50.0
   set transformMatrix to invert matrix transformMatrix
   transform page item 1 of document 1 using transformMatrix
end tell
```

launch

Launches Illustrator.

Parameters	What it is	Objects supported	Returns
none		`application`	nothing

load preset

Loads a set of preset tracing options from a file into a `tracing options` object.

Parameters	What it is	Objects supported	Returns
`tracingoptions`	The `tracing options` object to operate on.	`tracing options`	`boolean`
`presetname` **`Unicode text`**	The preset name, as found in the `application tracing presets` list.		

make

Creates a new object and returns a reference to newly created object. To place new art in a document, use this command to create a `placed item`, then use the `embed` command on the resulting `placed item` object to convert it to embedded art items.

Parameters	What it is	Objects supported	Returns

`new` **`type class`**	The class of object to create. The term `new` is optional.	all objects *except*:
		`application`
		`mesh item`
		`plugin item`
			`object reference`

`at` **`location reference`**	Location at which to insert new object.		
`[with properties` **`record`**`]`	Any property of the object you wish to set at creation.		
`[with data` **`anything]`**	Any data needed for creation that is not a property.		

➤ Creating layer objects

An open document must exist before this script is executed.

```
-- Make 2 layers in the open document, one at the top and one at the bottom
-- demonstrating the power of location references like beginning and end
tell application "Adobe Illustrator"
   set topLayer to make new layer ¬
      at beginning of document 1 with properties {name:"Top Layer"}
   set bottomLayer to make new layer ¬
      at end of document 1 with properties {name:"Bottom Layer"}
end tell
```

move

Moves one or more objects to a new location; returns references to the moved object or objects at the new location.

Parameters	What it is	Objects supported	Returns
object reference	Object or objects to move.	compound path item group item layer mesh item page item path item placed item plugin item raster item text frame	object reference or list (of object references)
to **location reference**	New location of the object or objects.		

Notes

Objects cannot be moved between documents.

➤ Moving objects to a layer

```
-- This script moves all objects in a document to the first layer
tell application "Adobe Illustrator"
   set allPageItems to every page item of document 1
   move allPageItems to beginning of layer 1 of document 1
end tell
```

➤ Moving layers

```
-- This script moves the bottommost layer to after the first layer
tell application "Adobe Illustrator"
   tell document 1 to move last layer to after first layer
end tell
```

open

Opens one or more specified documents.

Parameters	What it is	Objects supported	Returns
file specification	The file to be opened.	N/A	nothing
[forcing RGB/CMYK]	Pre-Illustrator 9 files only. Opens the document using the specified color space, converting if necessary. If not supplied, and the document contains both color spaces, displays a dialog for the user to choose one.		
[dialogs **boolean**]	If true, show warning and error dialogs when opening the file or files. Default is true.		
[with options **anything**]	Options for opening a particular type of file.		

➤ Opening a PDF file

```
-- This script opens a PDF document
-- selected by the user and forcing the use of the RGB color space
set fileToOpen to choose file with prompt "Select CMYK file to open as RGB"

tell application "Adobe Illustrator"
   open fileToOpen forcing RGB
end tell
```

paste

Pastes the clipboard contents into the current layer of the current document.

Parameters	What it is	Objects supported	Returns

none		`compound path item`
		`group item`
		`mesh item`
		`path item`
		`path point`
		`placed item`
		`plugin item`
		`raster item`
		`text`
		`text frame`
			nothing

Notes

Commands that manipulate the clipboard (`cut`, `copy`, and `paste`) require that Illustrator be the frontmost application. Use `activate` to bring Illustrator to the front before executing the `paste` command. No error is returned if there is no selection to paste. If the application is not frontmost, an error is returned.

➤ Pasting from the clipboard

```
-- Paste the contents of the clipboard into the current document
tell application "Adobe Illustrator"
   activate
   paste
end tell
```

print

Prints one or more documents or files.

Parameters	What it is	Objects supported	Returns
anything	Document or list of documents, or file or list of files to be printed.	document	nothing
[options **print options**]	A print options object.	print options	

➤ Print a document

```
-- Print the current document without displaying a dialog
tell application "Adobe Illustrator"
   print document 1 without dialog
end tell
```

➤ Print with options

```
-- Print Options
-- Make new document. add 6 symbol items
-- Set job options, color management options, coordinate options,
-- flattening options
-- Print the document using these options

set theDesktop to (path to desktop as string)
tell application "Adobe Illustrator"
   activate
   make new document
   repeat with i from 1 to 6
      round (i / 2 - (round (i / 2) rounding down)) rounding up
      make new symbol item in document 1 with properties ¬
         {symbol:symbol i of document 1, ¬
         position:{100 + (the result * 150), (50 + i * 70)}}
   end repeat
   set jobOptions to {class:job options, designation:all layers,¬
      reverse pages:true}
   set colorOptions to {class:color management options, ¬
      name:"ColorMatch RGB", intent:saturation}
   set coordinateOptions to {class:coordinate options, fit to page:true}
   set flatteningOptions to {class:flattening options, ¬
      clip complex regions:true, gradient resolution:60, ¬
      rasterization resolution:60}
   set printOptions to {class:print options, job settings:jobOptions, ¬
      color management settings:colorOptions, ¬
      coordinate settings:coordinateOptions, ¬
      flattener settings:flatteningOptions}
   print document 1 options printOptions
end tell
```

quit

Forces Illustrator to quit.

Parameters	What it is	Objects supported	Returns
none		application	nothing

Notes

If there is Illustrator data on the clipboard, Illustrator displays a dialog asking if you want to save the clipboard for other applications. To prevent this dialog from being displayed, send the following command to the frontmost application:

```
set the clipboard to {}
```

➤ Quitting Illustrator

```
-- Quit Illustrator after clearing the clipboard and closing documents
tell application "Adobe Illustrator"
   activate
   set the clipboard to {}
   close every document saving no
   quit
end tell
```

redraw

Forces Illustrator to redraw its window or windows.

Parameters	What it is	Objects supported	Returns
none		application	nothing

➤ Redrawing

```
-- This script redraws all windows in Illustrator
tell application "Adobe Illustrator" to redraw
```

release tracing

Reverts vector artwork in the document that was created by tracing to the original source raster art, and removes the traced vector art. Returns the original object used to create the tracing, and deletes the `tracingobject` object and its associated `plugin item` object.

Parameters	What it is	Objects supported	Returns
`tracingobject`	The `tracingobject` object to operate on.	`tracingobject`	`placed item or raster item object reference`

rotate

Rotates one or more page items counterclockwise by a specified rotation angle.

Parameters	What it is	Objects supported	Returns

`page item`	The `page item` object or objects to rotate.	`compound path item`
		`group item`
		`mesh item`
		`page item`
		`path item`
		`path point`
		`placed item`
		`plugin item`
		`raster item`
		`text frame`
			nothing

`angle` **`real`**	The rotation angle in degrees. Rotation is counterclockwise.		
`[transforming objects` **`boolean`**`]`	If `true`, the page item positions and their orientations are affected. Default is `true`.		
`[transforming fill patterns` **`boolean`**`]`	If `true`, the fill patterns assigned to paths are affected. Default is `true`.		
`[transforming fill gradients` **`boolean`**`]`	If `true`, the fill gradients assigned to paths are affected. Default is `true`.		
`[transforming stroke patterns` **`boolean`**`]`	If `true`, the stroke patterns assigned to paths are affected. Default is `true`.		

`[about`
`document origin/`
`top left/`
`left/`
`bottom left/`
`top/`
`center/`
`bottom/`
`top right/`
`right/`
`bottom right]`
	The point on the bounding box to which the rotation is applied. Default is `center`.		

Notes

The `rotate` command provides many variations when used with the `about` parameter. Experiment with different choices for `about` to see what the results are for each setting.

➤ Rotating about the bottom left corner

```
-- Rotate the first page item by 45 degrees using the
-- bottom left corner as the rotation pivot point
tell application "Adobe Illustrator"
   rotate page item 1 of document 1 angle 45.0 about bottom left
end tell
```

save

Saves an Illustrator document. Returns a reference to the saved document.

Parameters	What it is	Objects supported	Returns
document	The document to save.	document	object reference
[in **file specification**]	The file to save to, specified as a string containing the full file path or an alias. If not specified, the document is saved to its existing file.		
[as Illustrator/eps/pdf]	The file type to which to save.		
[with options **anything**]	The save options for the specified file type.		

➤ Saving PDF files

This example shows to batch process folders of Illustrator documents, saving each as a PDF file with specific settings.

```
-- Process all files in folders dropped on this script
-- (when saved as an applet)
-- Save each Illustrator file as a PDF file.
on run
   tell me to open {choose folder}
end run

on open droppedItems
   set destFolder to choose folder with prompt "Destination folder?"
   repeat with anItem in droppedItems
      tell application "Finder"
         -- Make sure each item processed by this script is a folder
         if class of item anItem is not folder then
            -- Not a folder, notify the user of the error
            display dialog "Please drop only folders on this script"
         else
            -- A folder, get the Illustrator files and process them
            set fileList to (every file of anItem whose creator type is ¬
                "ART5") as alias list
         end if
      end tell
      SaveFilesAsPDF(fileList, destFolder)
   end repeat
end open

-- fileList is a list of aliases to Illustrator files
-- destFolder is an alias to a folder where the PDF files are to be saved
on SaveFilesAsPDF(fileList, destFolder)
   set destPath to destFolder as string
   repeat with aFile in fileList
      tell application "Finder" to set fileName to name of aFile
      set newFilePath to destPath & fileName & ".pdf"
      tell application "Adobe Illustrator"
```

```
            open aFile
            save current document in file newFilePath as pdf ¬
               with options {class:PDF save options, ¬
                  compatibility:Acrobat 5, preserve editability:true}
            close current document saving no
         end tell
      end repeat
   end SaveFilesAsPDF
```

scale

Scales one or more page items by the specified horizontal and vertical amounts.

Parameters	What it is	Objects supported	Returns
page item	The page item object or objects to scale.	compound path item group item mesh item page item path item path point placed item plugin item raster item text frame	nothing
horizontal scale **real**	The horizontal scaling factor. 100.0 is 100%		
vertical scale **real**	The vertical scaling factor. 100.0 is 100%		
[transforming objects **boolean**]	If true, the page item positions and their orientations are affected. Default is true.		
[transforming fill patterns **boolean**]	If true, the fill patterns assigned to paths are affected. Default is true.		
[transforming fill gradients **boolean**]	If true, the fill gradients assigned to paths are affected. Default is true.		
[transforming stroke patterns **boolean**]	If true, the stroke patterns assigned to paths are affected. Default is true.		
[line scale **real**]	The amount that line widths are to be scaled. 100.0 is 100%. Default is 100.0.		
[about document origin/ top left/ left/ bottom left/ top/ center/ bottom/ top right/ right/ bottom right]	The point in the bounding box of the page item or items to which the scaling is applied. Default is center.		

Notes

The scale command provides many variations when used in conjunction with the about parameter. Experiment with different choices for the about parameter to see what the results are for each setting.

➤ Scaling a page item

```
-- Scale a page item by 50% horizontally resizing to the right
tell application "Adobe Illustrator"
   tell document 1
      scale page item 1 horizontal scale 50.0 vertical scale 100.0 about left
   end tell
end tell
```

select

Selects the text range.

Parameters	What it is	Objects supported	Returns
text	The text object or objects to select,	text	nothing
[extending selection **boolean**]	If true, the text range is added to the document's existing text selection. Default is false.		

set

Changes the value of a variable or an object's property or data. This is a standard AppleScript command used to assign values to variables and object properties.

Parameters	What it is	Objects supported	Returns
property *or* **variable**	The object property or script variable to modify.	any property or variable	nothing
to **anything**	Any valid value.		

➤ Setting a property

```
-- Set the zoom property of the frontmost view window to 100%
tell application "Adobe Illustrator"
  set zoom of view 1 of document 1 to 1.0
end tell
```

show presets

Returns presets from a file as a list of Unicode text items.

Parameters	What it is	Objects supported	Returns
`from` **`file specification`**	The file to import from, specified as a string containing the full file path or an alias.	N/A	`list (of Unicode text)`

singular matrix

Tests an existing matrix to see if it is singular. A singular matrix cannot be inverted.

Parameters	What it is	Objects supported	Returns
matrix	The matrix to test.	matrix	boolean

➤ **Inverting a matrix**

```
-- This script gets an identity matrix and then
-- test to see if it can be inverted (if not singular)
-- If it can, then it inverts it
tell application "Adobe Illustrator"
   set someMatrix to get identity matrix
   if (not singular matrix someMatrix) then
         set someMatrix to invert matrix someMatrix
   end if
end tell
```

store preset

Saves a set of preset tracing options from a `tracing options` object. For an existing preset, overwrites an unlocked preset and returns `true`. Returns `false` if the preset is locked.

Parameters	What it is	Objects supported	Returns
`tracingoptions`	The `tracing options` object to operate on.	`tracing options`	`boolean`
`presetname` **`Unicode text`**	The preset name. Use a name found in the `application tracing presets` list, or a new name to create a new preset.		

trace placed

Converts the raster art for the art item to vector art, using default options. Reorders the raster art into the source art of a plugin group, and converts it into a group of filled and/or stroked paths that resemble the original image.

Creates and returns a `plugin item` object that references a `traceobject` object.

Parameters	What it is	Objects supported	Returns
`placed item object`	The object to operate on.	`placed item`	`plugin item object reference`

trace raster

Converts the raster art for the art item to vector art, using default options. Reorders the raster art into the source art of a plugin group, and converts it into a group of filled and/or stroked paths that resemble the original image.

Creates and returns a `plugin item` object that references a `traceobject` object.

Parameters	What it is	Objects supported	Returns
`placed item object`	The object to operate on.	`placed item`	`plugin item object reference`

transform

Transform one or more page items by a specified matrix.

Parameters	What it is	Objects supported	Returns
page item	The page item object or objects to transform.	compound path item group item mesh item page item path item path point placed item plugin item raster item text frame	nothing
using **matrix**	The matrix to use for the transformation.		
[transforming objects **boolean**]	If true, the page item positions and their orientations are affected. Default is true.		
[transforming fill patterns **boolean**]	If true, the fill patterns assigned to paths should are affected. Default is true.		
[transforming fill gradients **boolean**]	If true, the fill gradients assigned to paths are affected. Default is true.		
[transforming stroke patterns **boolean**]	If true, the stroke patterns assigned to paths are affected. Default is true.		
[line scale **real**]	The amount that line widths are to be scaled. Default is 100.0, which is 100%.		
[about document origin/ top left/ left/ bottom left/ top/ center/ bottom/ top right/ right/ bottom right]	The point in the bounding box to which the transformation is applied. Default is center.		

Notes

This command can be used to generate any combination of transformations contained in a matrix, making it possible to skew objects among other modifications. The command provides many variations when used with the about parameter. Experiment with different choices for about to see what the results are for each setting.

➤ Transforming an object

```
-- This script skews an object 45 degrees to the right horizontally
-- by generating a rotation matrix and setting the appropriate matrix values
tell application "Adobe Illustrator"
   set baseMatrix to get rotation matrix angle 45.0
   set mvalue_b of baseMatrix to 0
   transform page item 1 of document 1 using baseMatrix
end tell
```

translate

Moves one or more page items from their existing position in a document to a new position defined by relative coordinates.

Parameters	What it is	Objects supported	Returns
page item	The page item object or objects to translate.	compound path item group item mesh item page item path item path point placed item plugin item raster item text frame	nothing
[delta x **real**]	The horizontal coordinate of the new position. Default is 0.0.		
[delta y **real**]	The vertical coordinate of the new position. Default is 0.0.		
[transforming objects **boolean**]	If true, the object positions and orientations are affected. Default is true.		
[transforming fill patterns **boolean**]	If true, the fill patterns are affected. Default is true.		
[transforming fill gradients **boolean**]	If true, the fill gradients are affected. Default is true.		
[transforming stroke patterns **boolean**]	If true, the stroke patterns are affected. Default is true.		

Notes

Use translate to move objects relatively from their existing position. Set the position property of an object to move the object to absolute coordinates.

➤ Moving an item to a new position

```
--This script moves the first page item to new relative coordinates
tell application "Adobe Illustrator"
   tell document 1 to translate page item 1 delta x 20.0 delta y -10.0
end tell
```

translate placeholder text

Translate the placeholder text to regular text. This allows you to enter Unicode characters as hex values.

Parameters	What it is	Objects supported	Returns
`Unicode text`	The placeholder text to be translated.	`text`	Unicode text or null

update

Reapplies the dynamic data of the active dataset to the artboard.

Parameters	What it is	Objects supported	Returns
dataset	Dataset to be updated.	dataset	dataset

Index

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z