# AppleScript Release Notes

Developer

# Contents

# Introduction to AppleScript Release Notes

This document describes significant changes to AppleScript and related tools in each version of OS X.

Unless otherwise noted, AppleScript can use a script created in any older version of OS X. A script created using a particular version of OS X may be used on any older version of OS X, provided it does not use features that were unavailable in that version.

You should read this document if you are writing AppleScript code or if you are writing an application that uses AppleScript.

## Organization of This Document

Each article in this document covers changes in a major revision of OS X and its updates.

## See Also

For introductory information on AppleScript and related technologies, see *AppleScript Overview*.

For reference documentation on the AppleScript language, see *AppleScript Language Guide*. The current revision of the Guide incorporates information in these notes up through Mac OS X v10.5, but it may be useful to see exactly when a feature was introduced.

# 10.9 Changes

This article describes changes to AppleScript and related tools in OS X Mavericks v10.9.

## Enhancements

### AppleScript

**Script Libraries**: You can now use a script as a *script library*, and use the handlers and properties it defines from any other script, making it dramatically easier to reuse code. Libraries may also use AppleScript/Objective-C and define their own terminology.

**"use" Statements**: Scripts can now explicitly declare what other code they use, such as applications or script libraries. `use` statements can also import terminology for use throughout a script without explicit `tell` statements, simplifying script structure.

**Notifications**: Scripts can now post notifications using the new `display notification` command in Standard Additions.

**Interleaved Argument Syntax**: AppleScript now supports an interleaved-argument syntax for handlers which can make positional-argument handlers easier to read: for example, `on doThis:a withThat:b` defines a handler with two parameters, `a` and `b`. The interleaved syntax may be used in any script, but is especially useful for calling Objective-C methods with AppleScriptObjC. In a compiled script, AppleScript will automatically translate the old positional form with underscores (such as `its doThis_withThat_(a, b)`) to the interleaved form (`its doThis:a withThat:b`), and the interleaved form will appear as the old positional form if opened on an older system.

For more details on new AppleScript language features, see the *AppleScript Language Guide*.

### AppleScript Editor

**Documents in the Cloud**: AppleScript Editor now supports saving script documents to iCloud; any such documents will be automatically synchronized between all computers sharing the same iCloud account.

**Code Signing**: AppleScript Editor's Export… command can now sign exported files with your Developer Identity. This takes the place of the procedure described in the 10.8 notes under "Gatekeeper and Signing Applets" (page 11).

# Bug Fixes

The default formatting of enumerated values has changed to make them more visually distinct from properties. [6778154]

`osascript`(1) is now a "UI element" process and can therefore display its own UI, such as using `display dialog`; telling another application such as System Events is no longer necessary. [12365409]

Getting the `last word` of a string no longer leaks memory. [12959927]

Negating a numeric string such as `"4.1"` now produces the correct result. [13599507]

# Compatibility Notes

## Accessibility

Accessibility features in OS X Mavericks, including System Events' Processes suite, no longer have a single switch to enable or disable them system-wide (the "Enable access for assistive devices" checkbox). Applications must now be individually authorized to use Accessibility using the Security & Privacy preference pane in System Preferences. If an unauthorized application attempts to use Accessibility, it will fail. The system will present a dialog directing the user to the appropriate place in System Preferences; once approved, the previously denied feature will work.

System Events attributes all uses of Accessibility features to the application calling it, so if an AppleScript applet uses, for example, `UI elements`, the applet must be authorized, not System Events. System Events' `UI elements enabled` property and AppleScript Utility's `GUI Scripting enabled` property are still available, but are now read-only, and will tell you if the caller is authorized to use Accessibility.

Applets that use Accessibility features will not save their properties when run. This is because of how the system tracks which applications are authorized: applets that save their properties modify their own contents to do so, which makes them look to the system like a different application that must be re-authorized. If you have an applet that requires both Accessibility and persistent property values, you can specially sign it such that it will still work without needing constant re-authorization; see http://support.apple.com/kb/HT5914 for details.

# Developer Notes

## Scriptable Applications

The value of the `OSAScriptingDefinition` Info.plist key may now be the base name of the sdef file; the ".sdef" extension is no longer required. [12909256]

Sandboxed applications can now compile scripts that target other applications even if the target application is not in `/Applications`. Your application will need an appropriate entitlement to compile and execute the script. [13042006]

## Scripting Additions

In scripts that use `use scripting additions`, AppleScript can optimize Apple event sending for some scripting addition commands, automatically eliminating the double-send described in "Scripting Addition Security" (page 17). Currently, it can perform this optimization for any scripting addition command with a `Context` value of `Any` in its Info.plist definition: for Standard Additions, this covers `ASCII character`, `ASCII number`, `offset`, `random number`, and `round`. This optimization is expected to cover more cases in the future; for best performance, make sure that your scripting addition has the strictest context possible. For more details on writing scripting additions and the meaning of the different context values, see Tech Note 1164, *Scripting Additions for Mac OS X*.

## Other

To debug library and framework loading, define the environment variable `ASDebugLibraryLoads`. AppleScript will then log whenever it loads a script library or framework:

```
AppleScript Editor[70986] <Notice>: AppleScript loaded library into component
0x810011: /Users/somebody/Library/Script Libraries/Map Window.scptd

AppleScript Editor[70986] <Notice>: AppleScript loaded framework for component
0x810012 OSAID(3): /System/Library/Frameworks/AppKit.framework

AppleScript Editor[70986] <Notice>: AppleScript referenced loaded framework for
component 0x820011 OSAID(1): /System/Library/Frameworks/AppKit.framework
```

These messages are logged to the ASL facility `com.apple.AppleScript`.

# 10.8 Changes

This article describes changes to AppleScript and related tools in OS X Mountain Lion v10.8.

## Enhancements

### AppleScript Editor

The AppleScript Editor application in OS X 10.8 includes a number of enhancements to document handling:

- **Auto Save**: Changes to documents are automatically saved, including changes that do not currently compile.
- **Versions**: Previous revisions of a document are easily retrievable using the standard version controls in the document's title bar.
- **Exporting**: Available from the File menu, the Export… command saves a copy of a script or script application for distribution. Export… is now the only way to save a script as "run-only", reducing the risk of saving over the editable original.
- **Bundle Identifier**: The Bundle Contents drawer for script applications now includes a field to set the bundle identifier. AppleScript Editor will fill in a default value, but it is recommended you customize the identifier for any script application that is targeted for distribution. Setting the bundle identifier has always been recommended, but previously required manual editing of the Info.plist file to do so.

## Bug Fixes

### AppleScript

Coercing a list to a string will now fail if one of the list items is not convertible. Previous versions would claim success with only a partial result. [2300198]

`div` and `mod` now use standard IEEE 754 routines to compute their results. This matches other languages and the rest of OS X. Previous versions would fudge the results to make certain "wrong" answers come out "right", but this introduced inaccuracies and never worked in all cases. For further information on the issue, see *What Every Computer Scientist Should Know About Floating-Point Arithmetic* by David Goldberg. [2358365]

### Standard Additions

`time to GMT` now adjusts live to DST shifts and changes to the time zone. [7676174]

`open for access` can now create a file with a slash in its name. [10431973]

## Compatibility Notes

When sending commands to a sandboxed application, such as TextEdit in OS X Mountain Lion, parameters that refer to files must be of an explicit file-like type and not a bare string, or the target application will not be able to access the file. For example, `file "Macintosh HD:Users:me:sample.txt"`, `POSIX file "/Users/me/sample.txt"`, or the result of `choose file` would all be acceptable, but the string `"/Users/me/sample.txt"` would not.

## Developer Notes

### Gatekeeper and Signing Applets

OS X Mountain Lion includes Gatekeeper, which protects users from malicious software by applying a policy about what downloaded software is allowed to run. Gatekeeper relies on *code signing* to verify applications: a signed application is guaranteed to have been created by the signer and to have not been modified since it was signed. By default, Gatekeeper will allow running only applications that have been signed by the Mac App Store or an identified developer. If you write script applications ("applets") for distribution, then this policy applies to your applets. To sign your applets so Gatekeeper's default policy will not block them:

1. Obtain a Developer ID certificate. For details, see "Distributing Applications Outside the Mac App Store" in *App Distribution Guide*.
2. Save a copy of your applet for distribution using the Export… command. (Note: Your applet should have a unique bundle identifier, which you can set in the Bundle Contents drawer.)
3. Launch Terminal, and change the working directory to the applet bundle. (Tip: drag the applet on to Terminal, and it will create a new terminal window with the directory set correctly.)
4. Mark the main script as read-only.

```
chmod a-w ./Contents/Resources/Scripts/main.scpt
```

If your applet contains other scripts, also mark them as read-only.

5. Sign the applet using your code signing identity.

```
codesign -s 'My Signing Identity' .
```

**Note:** A code-signed applet must not rely on changed property values being persistent. Ordinarily, such changes would be written back to the script file, but in a signed applet, that would invalidate the signature and render the applet unlaunchable. Step 4 above ensures that this will not happen.

## Sandboxing and Running Scripts

Sandboxing your application may require changes to how it runs scripts. The usual method in the past has been `NSAppleScript`, but since scripts typically rely on sending Apple events and the default sandbox profile does not allow sending Apple events to any other application, this often does not work correctly when in a sandbox. Scripts run from your application will fall into one of three categories:

- *Self-targeted scripts.* Your scripts only send events to your application and never to any other application. Continue to use `NSAppleScript` as before.
- *Built-in scripts.* Your scripts are built as part of your application, and will not change after shipping. Continue to use `NSAppleScript`, but add entitlements for sending events to the target applications.
- *User scripts.* Your scripts are supplied by the end user, and may use any other application. Use `NSUserScriptTask`. The scripts must be stored in a special location (use `NSApplicationScriptsDirectory` to determine where), and will run outside of your sandbox.

## Sandboxing and Scriptability

Sandbox policy does not restrict receiving of Apple events, so in general, your application's scriptability code will not be affected by sandboxing.

> **Note:** As mentioned under Compatibility above, sandboxed applications will not be able to access files referred to in commands using a string path. Any parameters or properties in your application that refer to files should be declared as type `file`, and not type `text`. Apple Event Manager will add sandbox extensions to events that have file-like parameters, but only if the parameter is of a recognized file-like type: `typeAlias`, `typeFileURL`, `cFile`, and so on. Without these extensions, the file will not be accessible from the target application's sandbox, and the command will probably fail.

Your application's scripting definition (`sdef`), however, should be updated to add *access groups*: groups of commands, classes, and so on that a sandboxed client can request to use with the new `com.apple.security.scripting-targets` entitlement. For example, Mail defines a "compose" access group that allows creation and editing of an outgoing message, but nothing else. This is safer than the old `com.apple.security.temporary-exception.apple-events` entitlement, which allows access to the entire scripting interface. See `sdef`(5) for details of the markup format.

# 10.7 Changes

This article describes changes to AppleScript and related tools in OS X Lion v10.7.

## Enhancements

### Script Templates

AppleScript Editor now supports creating scripts from *templates* , a skeleton document designed for a particular task, using the New From Template command. Several templates are included in the system; to make your own, add an AppleScript Editor document to the folder `Library/Application Support/Script Editor/Templates`. Holding down the Command key while selecting a template menu item will reveal that template in Finder.

### AppleScriptObjC in AppleScript Editor

AppleScript Editor provides a template "Cocoa-AppleScript Applet", which is an AppleScriptObjC-based application that emulates the traditional `on run` and `on open` handlers for a script application ("applet"). As a result, you can write an applet much as you normally would, using AppleScript Editor, but with the added ability to call any Cocoa API.

AppleScriptObjC applets must be run from AppleScript Editor using the new Run Application command, which runs an applet by launching it as a separate process, as if it had been opened from the Finder. Run Application is also useful for applets that have special behavior when launched as their own process, such as `on idle` handlers.

### Other Enhancements

System Events' Processes Suite has a new `pop over` class to support the new pop-over windows in OS X Lion. [8948934]

`path to` has a new selector `services folder`, which returns the user's Services folder. [6727316]

# Bug Fixes

## AppleScript

Performance of AppleScript applets has been improved. [3434887]

AppleScript will no longer hang looking for `text item delimiters` that current `ignoring` settings say to ignore. [7346401]

AppleScript now has built-in knowledge of System Events, and will never ask where it is when running a script. [8088421]

## Standard Additions

`mount volume` now works correctly with passwords longer than 31 characters. [7154725]

`beep` now always beeps, even when used as the last command in a script. [7233358]

## AppleScript Editor

Performance of saving script bundles and applets with large embedded resources has been improved. [4404257]

Save As… will copy embedded resources correctly. [7903069]

# Compatibility Notes

For security reasons, AppleScript no longer supports initializer-based scripting additions. Scripting additions must use the Info.plist scheme introduced in Mac OS X v10.5 Leopard. If you have a now-unsupported initializer-based scripting addition, you will see a console message such as this:

```
OpenScripting.framework – scripting addition
"/Library/ScriptingAdditions/Potrzebie.osax" declares no loadable handlers.
```

For details on implementing a scripting addition, see TN1164, *Scripting Additions for Mac OS X*.

# 10.6 Changes

This article describes changes to AppleScript and related tools in Mac OS X Snow Leopard v10.6.

## Tool Reorganization

In order to streamline the `/Applications` folder, the `/Applications/AppleScript` folder has been removed, and its contents reorganized:

- Script Editor.app is now named AppleScript Editor.app, and is in `/Applications/Utilities`.
- Script Menu preferences are now in AppleScript Editor's preferences, in the General tab.
- Example scripts may be accessed from AppleScript Editor's Help menu.
- Folder Actions Setup.app is now in `/System/Library/CoreServices`; it may be accessed by control-clicking on a folder and selecting Folder Actions Setup.
- AppleScript Utility.app is now in `/System/Library/CoreServices` and has no UI; it only exists as a faceless application for the benefit of scripts that target it.

## AppleScript Editor

AppleScript Editor has several significant enhancements:

AppleScript Editor is now more responsive while running a script, and can run several scripts simultaneously. Previous versions would let you start running a second script, but the first one would pause until the second one finished. It does this by running scripts on background threads. If this causes trouble, typically because a third-party scripting addition has claimed to be thread-safe but isn't, you can force the script to run on the main thread by holding the Control key and selecting the Run command, which will show as "Run in Foreground" in the menu.

Syntax coloring for scripts now has much more detail: commands, parameters, classes, properties, and enumerated values all have their own category now, instead of sharing the "application keywords" category, and may be formatted differently depending on whether they come from an application or a scripting addition. This allows scripters to better determine what a term is simply by looking at the compiled script.

The Result and Event Log panes have been merged into a single pane with three visibility controls: Result, Events, and Replies. Selecting "Result" shows only the script result, much like the old Result pane. Selecting "Events" or "Replies" shows Apple events sent by the script and the values returned by the events, respectively, along with the output from any explicit `log` commands, and the final result at the end. These controls are dynamic: you may change them at any time, even while the script is running, and the appropriate bits of the log will be hidden or shown. Hovering over an event in the log will show a tooltip with the raw Apple event contents, including all the four-byte codes.

There is now an option to enable a new Tell Application pop-up in the navigation bar: this controls the default target of the script, effectively putting a `tell` block around the entire script. This is useful for exploring an application's scriptability, since there is then no need for an explicit `tell`, or for testing scripts that will eventually be run inside another application: for example, when writing a Mail rule script, you would set the default target to Mail. This setting is saved with the script, but does *not* affect the script's behavior if it is saved as an application: the default target is still that application. You can enable this feature in AppleScript Editor's "Editing" preferences. The applications in the pop-up are controlled by the contents of the Library window.

## Scripting Addition Security

For security reasons, most scripting addition commands now return a "privilege violation" error when sent between application processes. In order to preserve compatibility with existing scripts, AppleScript redirects most of these commands to the current application, that is, the process running the script. If a script sends events to a remote computer via EPPC ("Remote Apple Events"), AppleScript may redirect them to the System Events process on the target machine. AppleScript Editor's Event Log will show when this redirection happens: you will see the event sent first to the original target process, return an error, and then sent again, often to the current application.

Some scripting addition commands, such as `display dialog`, must be handled within the target process to operate correctly, and may require authenticating as an administrator if any of the following are true:

- The sender and target processes have different user or group owner ids.
- The target process is "tainted" by privilege level changes. (See `issetugid`(2) for full details.)
- The target application is not scriptable. To be considered scriptable, an application must have a terminology dictionary or the Info.plist key NSAppleScriptEnabled set to true.

If the event can be successfully redirected to the current process, then it didn't need to be sent to the target process in the first place. You can eliminate the unnecessary double-send by moving the scripting addition command outside of the `tell` block, or by adding `tell current application to` to the command. For example, this script makes a new folder on the desktop named with the current date, but does it in a way that requires `current date` to be sent twice:

```
tell application "Finder"
    set folderName to date string of (current date)
    make new folder with properties {name:folderName}
end tell
```

To eliminate the double send, move the `current date` outside of the `tell` block:

```
set folderName to date string of (current date)
tell application "Finder"
    make new folder with properties {name:folderName}
end tell
```

Or add an inner `tell` applying to the addition command:

```
tell application "Finder"
    tell current application to set folderName to date string of (current date)
    make new folder with properties {name:folderName}
end tell
```

# Other Enhancements

## AppleScript

When getting the `text items` of a string, all the values in `text item delimiters` are considered. Previous versions only considered the first item in the list. [1186965]

AppleScript itself no longer depends on being run within a login session, and may be run from outside the current user's login, such as via an `ssh`(1) session or in a root `cron`(8) job. However, external code that a script uses, such as scripting additions or applications, may still require a login session. [3192824]

The various types of `ignoring` behavior for text comparisons are now defined using Unicode General Categories, not ASCII characters:

> `ignoring punctuation` ignores category P*: for example, left- and right-quotation marks are now ignored. However, the backtick character (`) used to be ignored but is now considered, because Unicode classifies it as a symbol, not punctuation.

`ignoring hyphens` ignores category Pd: for example, em- and en-dashes are now ignored.

`ignoring whitespace` ignores category Z*, plus tab (`\t`), return (`\r`), and linefeed (`\n`): for example, non-breaking spaces are now ignored.

For further details on General Categories, see the Unicode Standard, section 4.5. [4819817]

### Standard Additions

If a `do shell script` command exits because of a signal, including a crash, `do shell script` will throw an error. The error number will be the signal number plus 1000, to allow distinguishing signal errors from exit status errors, which are always 255 or less. Formerly, the signal was ignored.

`say` now has several optional parameters for customizing voice parameters: `speaking rate`, `pitch`, `modulation`, and `volume`, and asynchronous speech no longer requires the Speech status window to be open. See the Standard Additions dictionary for more details.

### Command Line

`osascript` now supports the `log` command; the output will be sent to `stderr`. [6260159]

### Folder Actions

Folder Actions now supports attaching an Automator workflow as a folder action. Only “files added” actions are supported; the added files will be the input to the workflow.

## Bug Fixes

### AppleScript

Performance of getting elements of text has been improved. [2206026]

The startup time of the AppleScript interpreter has been improved. [3188110].

`character` elements are counted the same way `offset of` counts them, so character counts now always match between the two. [5578622]

In general, AppleScript in 10.5 and later will consider the class constants `string` and `text` equal. However, this did not apply to list containment, so `class of "foo" is in {string}` would incorrectly return `false`. It now evaluates to `true`. [5581947]

If a script uses `using terms from` blocks, the targets will not trigger a “Where is…” dialog when the script is run on 10.4 and earlier systems. [5823283]

### AppleScript Editor

“Undo” now works correctly across Compile commands.

### Folder Actions

Folder Actions now attempts to delay calling “files added” actions on files until they are done being copied. Previous versions called “files added” actions on new files as soon as they appeared in the file system. This was a problem for files that were being copied into place: if the file was sufficiently large or coming over a slow server link, the file might appear several seconds before it was done being copied, and running the folder action before it finished copying would behave badly. Folder Actions now watches if the file is changing size: if it stays the same size for more than three seconds, it is deemed “done”, and the action is called.

## Compatibility Notes

### 64-bit

AppleScript Editor and `osascript`(1) run in 64-bit mode by default on hardware that supports it. This can be an issue for scripting additions: Standard Additions is fully 64-bit capable, but third-party additions may not be, and will refuse to load with a console message such as this:

```
Error loading Potrzebie.osax/Contents/MacOS/Potrzebie:
  dlopen(Potrzebie.osax/Contents/MacOS/Potrzebie, 262): no suitable image found.
Did find:
  Potrzebie.osax/Contents/MacOS/Potrzebie: mach-o, but wrong architecture
```

This only affects scripts that rely on commands in that scripting addition; otherwise the message is effectively a warning. If a 64-bit version of the addition is not available, you can work around the issue by forcing the process running the script to run in 32-bit mode, either by using the “Open in 32-bit mode” option in Finder’s “Get Info” panel, or by using `arch`(1), such as this:

```
arch -i386 osascript 32-bit-only.scpt
```

## Script File Formats

AppleScript Editor no longer supports saving applications in the old single-file format. All scripts saved as an "Application" will be saved as an application bundle with three supported architectures: ppc, i386, and x86_64. Such applications can run on Mac OS X v10.3 and later. Old single-file applications opened in AppleScript Editor will be treated as read-only files: you must use Save As to save them, and they will be saved in the new format.

`osacompile` now outputs data-fork scripts by default. This matches AppleScript Editor. To create a resource-fork script compatible with classic Mac OS, use the option `-r scpt:128`. [6325665]

## Dates and Times

Parsing of date strings now uses `CFDateFormatterCreateDateFromString`, which means that custom date formats, Unicode-only locales such as Arabic, and non-Gregorian calendars are now all handled correctly. However, it is also much more restrictive about deviating from the system date format. Previous versions would accept anything that looked even vaguely like a date: the string could omit components, reorder others, add or omit punctuation, and so on. In AppleScript 2.1, the string must exactly match one of the system date formats (full, long, medium, or short; see System Preferences > International > Formats for examples), including all punctuation and whitespace. The only difference allowed is to use either a two- or four-digit year, regardless of what the format uses.

Some scripts used partial date specifiers as a way to get dates relative to the current date, such as `date "1/31"` for January 31 of the current year, or `date "15"` to get the 15th of the current month. To do this in AppleScript 2.1, get a base date and then set the desired components using the properties of the date object. For example, to get January 31 of the current year:

```
set d to date "1/31/2000"
set year of d to year of (current date)
d --> date "Saturday, January 31, 2009 12:00:00 AM"
```

Alternatively, start with `current date` as the base. To get the 15th of the current month:

```
set d to current date
set day of d to 15
set time of d to 0
d --> date "Friday, May 15, 2009 12:00:00 AM"
```

This technique works with all versions of AppleScript, and can be used for scripts that must run on 10.6 and 10.5 and earlier.

## Text

`character` elements are now counted using `CFStringGetRangeOfComposedCharactersAtIndex`. This mostly matches the 10.5 behavior of counting grapheme clusters, but counts a CR-LF (`\r\n`) sequence as two separate characters, as it did in 10.4 and earlier, not one.

`word` elements are now counted using `kCFStringTokenizerUnitWordBoundary`, which produces better results for non-Roman languages, and matches the definition of a "word" in the rest of the system, such as when double-clicking in text. As noted in *AppleScript Language Guide*, the word-break rules may change depending on user settings and system software updates, and should not be relied upon for deterministic parsing of text.

As described in "Other Enhancements" (page 18), `ignoring` categories now ignore relevant characters in the entire Unicode character set, not just ASCII. The results of comparisons are more correct, but may differ from previous versions.

## Apple Events

`ssh`(1) sessions are now in the same security space as other processes owned by the same user, and therefore send Apple events to each other. This means that a user can now use `ssh`(1) to run a script and control applications on a remote machine if they are also logged in as the console user on that machine.

Mac OS X v10.6 no longer supports sending remote Apple events to Mac OS 9 machines.

# Developer Notes

## 64-bit

AppleScript, Standard Additions, and all AppleScript-related system applications such as System Events are fully 64-bit capable. Scripting addition developers should make their additions 64-bit-capable as soon as possible, otherwise they will not work on 64-bit systems without extra steps by the user. (See "Compatibility Notes" (page 20).) 64-bit-capable scripting additions should use the updated format for Mac OS X v10.6; see TN1164, *Scripting Additions for Mac OS X*, for details.

## Thread Safety

OSA and AppleScript are now thread-safe: they may be safely called on a non-main thread or from multiple threads without any locking in the client code. This also applies to `NSAppleScript`. This does not mean that AppleScript is totally concurrent: AppleScript uses locking to ensure that any single connection (a

`ComponentInstance`) will only run on one thread at a time. Because of the size of the locking granularity, trying to manipulate the same script from multiple threads at once may still be subject to race conditions, and is not recommended.

Before using a scripting component on a background thread, developers should test the component's "thread-safe" bit (`cmpThreadSafe` in the `ComponentDescription`'s `componentFlags`.) Test the generic component before using OSA on a background thread, and then test the specific language component before using it on a background thread.

### Scripting Additions and Thread Safety

Since a scripting addition can contain arbitrary code, a given addition command may or may not be thread-safe: `offset of` is, for example, but `display dialog` is not. For compatibility, addition commands are presumed to be thread-unsafe, and will be executed on the main thread. Commands may be marked as thread-safe using an expanded Info.plist definition; see TN1164, *Scripting Additions for Mac OS X*, for details. Coercion handlers in scripting additions must be thread-safe; again, see TN1164 for details. Scripting addition developers should review and, if necessary, update their additions for thread-safety.

## AppleScript Identifiers

AppleScript now uses different rules for capitalizing identifiers accessed through the OSA APIs. These changes are intended to preserve capitalization information and make it unnecessary to know internal rules of AppleScript. When accessing a property or handler, such as by using `OSAGetProperty` or `OSAExecuteEvent` with a "call subroutine" event, the identifier is handled case-insensitively. Formerly, you had to lowercase the identifier before passing it to AppleScript. When getting a property or handler name, such as by using `OSAGetPropertyNames` or when a script result contains a user identifier, the identifier will be capitalized as it was in the script, which may be mixed-case. Formerly, identifiers would always come out all-lowercase.

## Scripting Definitions (sdefs)

`OSACopyScriptingDefinition` now has a sister function `OSACopyScriptingDefinitionFromURL`, which can accept either a `file:` or `eppc:` URL.

`OSACopyScriptingDefinition` now does a better job preserving information in hidden `aete` suites.

# 10.6.2 Changes

AppleScript Editor will now open certain malformed application dictionaries.

# 10.5 Changes

This article describes changes to AppleScript and related tools in Mac OS X Leopard v10.5 and its updates.

## Unicode Support

AppleScript is now entirely Unicode-based. Comments and text constants in scripts may contain any Unicode characters, and all text processing is done in Unicode, so all characters are preserved correctly regardless of the user's language preferences. For example, this script works correctly in AppleScript 2.0, where it would not have in previous versions:

```
set jp to "□□□"
set ru to "Русский"
jp & " and " & ru -- returns "□□□ and Русский"
```

There is no longer a distinction between Unicode and non-Unicode text. There is exactly one text class, named "text": that is, `class of "foo"` returns `text`. It is functionally equivalent to the former `Unicode text` class, so it may contain any Unicode character, and has two new features besides:

- `text` objects have an `id` property, which may also be used as an address.

  These allow mapping between Unicode code point values and the characters at those code points: for example, `id of "A"` returns `65`, and `character id 65` returns `"A"`. The id of text longer than one code point is a list of integers, and vice versa: for example, `id of "hello"` returns `{104, 101, 108, 108, 111}`, and `string id {104, 101, 108, 108, 111}` returns `"hello"`. (Because of a bug, `text id ...` does not work; you must use one of `string`, `Unicode text`, or `character`.) These obsolete the older `ASCII character` and `ASCII number` commands, since, unlike those, they cover the full Unicode character range and will return the same results regardless of the user's language preferences. [2915643]

- `character` elements of text count a grapheme cluster as a single character.

  Some "characters" may be composed of a series of Unicode code points, what Unicode defines as a *grapheme cluster*. For example, "é" may be encoded as U+0065 (LATIN SMALL LETTER E), U+0301 (COMBINING ACUTE ACCENT). AppleScript 2.0 will count the cluster as one character, where older versions counted the base character and combining mark separately. For a complete definition of grapheme clusters, see UAX #29: Unicode Text Segmentation. [4192557]

Matching of `text item delimiters` now respects `considering` and `ignoring` attributes, where previous versions ignored them. The `expansion` attribute is no longer supported.

The change to all-Unicode necessitated a change in the `read` and `write` commands and how they deal with text encodings. Formerly, they would write `string` objects using the primary encoding and `Unicode text` objects as UTF-16, unless instructed otherwise using an `as` parameter. Now that there is only one text class, the `read` and `write` commands rely solely on the `as` parameter to determine the encoding: with no `as` parameter, `as text`, or `as string`, they use the primary encoding; with `as Unicode text`, they use UTF-16. For the most reliable results when creating scripts that will run on both 2.0 and pre-2.0, always specify the encoding explicitly using `as text` or `as Unicode text`, as appropriate. [4421553]

## Compatibility

For compatibility with pre-2.0 AppleScript, `string` and `Unicode text` are still defined, but are considered synonyms for `text`. For example, all three of these statements have the same effect:

```
someObject as text
someObject as string
someObject as Unicode text
```

In addition, `text`, `string`, and `Unicode text` will all compare as equal. For example, `class of "foo" is string` is true, even though `class of "foo"` returns `text`. It is still possible for applications to distinguish between the three different types, even though AppleScript itself does not.

Now that AppleScript preserves all characters correctly worldwide, it is also stricter about the text used in scripts. AppleScript syntax uses several non-ASCII characters, such as "□a" and "¬". These characters must be typed exactly as the *AppleScript Language Guide* describes. For compatibility with Asian national encodings, "□a" and "□a" are allowed as synonyms for "«" and "»", since the latter do not exist in some Asian encodings.

Because all text is Unicode text, scripts now always get the Unicode text behavior. This may be different from the former `string` behavior for some locale-dependent operations, in particular `word` elements. To get the same behavior with 2.0 and pre-2.0, add an explicit `as Unicode text` coercion, for example, `words of (someText as Unicode text)`.

Because `text item delimiters` now respect `considering` and `ignoring` attributes, they now are case-insensitive by default. Formerly, they were always case-sensitive. To get the same behavior with 2.0 and pre-2.0, add an explicit `considering case` statement.

Because AppleScript 2.0 scripts store all text as Unicode, any text constants count as a use of the former `Unicode text` class, which will work with any version of AppleScript back to version 1.3. A script that contains Unicode-only characters such as Arabic or Thai will run, but will not be correctly editable using pre-2.0 AppleScript: the Unicode-only characters will be lost.

Use of the new `id` property requires AppleScript 2.0.

## Application Objects

`application` objects in AppleScript have some new features designed to reduce the need for awkward or obscure workarounds. In other words, the capabilities are not new, but they are now directly supported and easier to use.

- Is an application running?

  While System Events is capable of telling you whether or not an application is running, it takes several lines to do it. `application` objects now have a `running` property that can give you the answer directly, without invoking System Events. For example, the following script pauses iTunes, but will not launch it if it is not already running:

  ```
  tell application "iTunes"
      if it is running -- same thing as "if running" or "if running is true"
          pause
      end if
  end tell
  ```

  `running` does not need to appear inside a tell block; the above script could also be written like this:

  ```
  if application "iTunes" is running
      tell application "iTunes" to pause
  end if
  ```

- Is an application frontmost?

  `application` objects now have a `frontmost` property similar to `running`, including that the information is available from System Events, but it tells you whether or not the application is in front. The value of `frontmost` for background-only applications, UI element applications such as System Events, and applications that are not running is always `false`.

- Get an application's version

Most applications have a version property of their own, but using it requires that the application be running, and sometimes you specifically want to avoid that. The new built-in `version` property will return the application version as text without launching the application or sending it an event. There is also a corresponding built-in `name` property; the value is usable as the name of an application to `tell`.

- Target applications by signature or bundle identifier [3858040]

  Doing this in older versions requires a multi-line incantation using Finder. It is now possible to simply address an `application` object by id, where the id is either the bundle identifier or the four-character signature code. For example, `application "Mail"`, `application id "com.apple.mail"`, and `application id "emal"` all refer to Mail.app, and will all work as the target of a `tell` block, as in this example:

```
tell application id "com.apple.mail"
    get unread count of inbox
end tell
```

  There is a corresponding `id` property, so you can get the necessary id to tell:

```
get id of application "TextEdit"
-- returns "com.apple.TextEdit"
-- Now we know that 'tell application id "com.apple.TextEdit"' will work.
```

  This does not require that the application be running.

  When running a script, if an application specified by id is not found, AppleScript will not ask where it is. It will throw an error, which can be caught using a `try` block.

  Scripts intended for distribution should use the id form. That way, the script will continue to work even if the user has changed the name of the application.

In addition, there are several changes to application behavior to make them easier to deal with:

- Applications launch hidden.

  AppleScript has always launched applications if it needed to in order to send them a command. However, they would always launch visibly, which could be visually disruptive. AppleScript now launches applications hidden by default. They will not be visible unless the script explicitly says otherwise using `launch` or `activate`.

- Applications are located lazily.

When running a script, AppleScript will not attempt to locate an application until it needs to in order to send it a command. This means that a compiled script or script application may contain references to applications that do not exist on the user's system, but AppleScript will not ask where the missing applications are until it encounters a relevant `tell` block. Older versions of AppleScript would attempt to locate every referenced application before running the script.

When opening a script, AppleScript will attempt to locate all the referenced applications in the entire script, which may mean asking where one is. Pressing the Cancel button only cancels the search for that application; the script will continue opening normally, though custom terminology for that application will display as raw codes. In older versions, pressing Cancel would cancel opening the script.

- Applications are located and re-located dynamically.

  `application` object specifiers, including those in `tell` blocks, are evaluated every time the script runs. This alleviates problems with scripts getting "stuck" to a particular copy of an application. [4356296]

### Compatibility

Uses of the built-in application properties will fall back to sending an event to the application in older versions of AppleScript, but the application may not handle them the same, or handle them at all. (Most applications will handle `name`, `version`, and `frontmost`; `id` and `running` are uncommon.) The other new features above require AppleScript 2.0.

## Scriptability and VoiceOver

AppleScript Utility and Folder Actions Setup are now scriptable, and Script Editor's scriptability has been enhanced thanks to improvements in Cocoa Scripting. All three applications are now VoiceOver-aware.

## Scriptable Network Preferences

Networking preferences are now accessible via scripting using a new suite in System Events, where they were previously only scriptable via accessibility. A script can inspect various networking settings, and can tell a service to connect or disconnect. For example, the following script locates a PPPoE service in the current location and tells it to connect if it exists.

```
tell application "System Events"
    tell network preferences
        tell current location
            set aPPPoEService to a reference to (first service whose kind is 10)
```

```
            if exists aPPPoEService then
                connect aPPPoEService
            end if
        end tell
    end tell
end tell
```

# Command Line Support

AppleScript now allows # as a comment-to-end-of-line token, in addition to --. This means that you can make a plain text AppleScript script into a Unix executable by beginning it with the line `#!/usr/bin/osascript` and giving it execute permission. For details of `osascript` usage, including how to access command line arguments from the script, see the `osascript`(1) man page. [2468788]

There is a command-line tool to display compiled scripts as text, `/usr/bin/osadecompile`. [4501123]

`osascript`(1) and `osacompile`(1) now correctly handle text scripts encoded as UTF-8, in addition to UTF-16 and the primary encoding. If a script is neither UTF-8 nor UTF-16, it is presumed to be encoded using the primary encoding.

## Compatibility

Compiled scripts that use # will run normally on pre-2.0 systems, and if edited will display using --. Executable text scripts using `#!/usr/bin/osascript` as above will not run on pre-2.0 systems, since the # will be considered a syntax error.

# Other Enhancements

## Script Editor

- The dictionary viewer allows class definitions to show all the properties and elements they inherit from other classes. To turn this on and off, use the "Show inherited items in dictionary viewer" setting in the General preferences. [4950321]
- Script Editor can show tabs, carriage returns, and linefeeds in text constants as an escape sequence rather than a literal tab, carriage return, or linefeed. To turn this on and off, use the "Escape tabs and line breaks in strings" setting in the Editing preferences. [4911918]

- When run from a script document in Script Editor, `path to me` will return the location of the document file. If the document has not been saved, `path to me` will return the location of Script Editor. [3148582, 5363659]

### System Events

- The `disk` class has `server` and `zone` properties for use with AppleShare volumes. [3554247]
- The `file` class has a `default application` property, corresponding to `info for`'s `default application` property and Finder's `open with` setting in Get Info. [4796981]
- System Events has a `downloads folder` property. [5255406]
- System Events has a new Security Suite, which controls various settings from the Security preference pane. [4358335]
- The `process` class has a `bundle identifier` property. [4782866]
- The `process` class has an `architecture` property, which specifies the processor architecture of that process. [5237251]
- The Property List Suite allows creating new property list files and property list items. [4728058, 5392953]
- If a `XML element` object has a "name" or "id" attribute, it will be reflected in the element's `name` or `id` property, respectively. `XML element named ...` and `XML element id ...` will also work. [4437103, 4441834]

### Image Events

- Image Events supports RAW images. [4250666]
- The `pad` command has an optional `with pad color` parameter. [5234464]
- The `save` command has an optional `with compression level` parameter to specify the JPEG compression level. [3614780]

## Bug Fixes

### AppleScript

- The `delay` command uses less CPU. [3178086]
- Impossible object specifiers in math expressions, such as `1 + character 2 of "9"`, produce an error instead of a random result. [4029175]

- The numerical limits for `repeat` loops accept real numbers; they will be rounded to integers. [4215670]
- Object specifiers other than `application` and `date` specifiers, in particular `alias "..."` and `POSIX file "..."`, are not evaluated at compile time, and will be left exactly as originally typed. [4444698]
- AppleScript no longer limits script memory usage to 32MB. [4511477]
- Counting the paragraphs of an empty string gives a result of zero. [4588706]
- Raw data literals (`«data ...»`) are no longer limited to 127 bytes. [4986420]

## Standard Additions

- `choose from list` correctly sizes the dialog if given more than 2000 items. [4102349]
- `display dialog` correctly sizes the dialog if given more than 2000 lines of text. [4314839]
- `display dialog` correctly displays "^0", "^1", "^2", and "^3" in the text. [4831383]
- A script may not tell a remote application to `do shell script`. [4241641]

## Script Editor

- Script Editor accurately reports the position of errors in scripts that contain Asian characters. [3457168]
- The Library window correctly handles applications with Asian names. [5080569]

## System Events

- `disk` objects distinguish between disks with identical names. [4141496]
- Setting the `file type` and `creator type` properties works correctly on Intel systems. [4788442]
- Setting the `focused` property of a `UI element` works. [4756520]
- `UI element` objects have more informative names. [4886664]
- The `frontmost` property of the `process` class will always be false for UI element applications such as System Events. This means that `first process whose frontmost is true` will return the same application as `path to frontmost application`. [4175274, 5100612]
- The `process` class will report the correct name for an AppleScript application bundle. [4381260]
- Attempting to use GUI Scripting commands when Universal Access is off will produce a more informative error message. [4774412]

## Others

- Database Events' performance has been improved. [4124666]

- Folder Actions have been re-architected to improve their stability and performance. [3663310, 3693421, 4614337]
- Folder Actions Setup allows attaching a script bundle as an action. [4036743]
- Image Events handles files whose name contains "/" correctly. [4275156]

# Developer Notes

## Scripting Definitions (sdefs)

AppleScript can now read an application's sdef directly. This means that an application with only an sdef no longer needs to be launched to get its scripting terminology. [4569425]

There are a few changes to the sdef format itself; see the sdef(5) man page for details. In particular, sdefs now support the use of XInclude. Applications that process sdefs should be prepared to handle them. The 10.4 sdef format is still supported, but you should migrate when you have the opportunity.

## API changes

Several Open Scripting Architecture (OSA) APIs have new replacements, mostly intended for improved handling of Unicode script content. Except as noted, the new APIs are available in Mac OS X version 10.5 and later, and the old APIs are still available and supported, though they do not support the new functionality. For additional details, see the *Open Scripting Architecture Reference*.

| old | new |
| --- | --- |
| `OSAGetSource`, `OSADisplay` | `OSACopySourceString`, `OSACopyDisplayString`<br>These return their output as a `CFAttributedStringRef`, which means they can handle styled Unicode text, where the older two cannot. If a language component does not support `OSACopySourceString` or `OSACopyDisplayString` directly, calls to them will be passed through to `OSAGetSource` and `OSADisplay`, respectively. |
| `ASGetSourceStyles`, `ASSetSourceStyles` | `ASCopySourceAttributes`, `ASSetSourceAttributes`<br>These use a CFArray of CFDictionaries of attributes to apply to the `CFAttributedStringRef` returned by `OSACopySourceString` and `OSACopyDisplayString`. |

| old | new |
| --- | --- |
| `OSAGetAppTerminology` | `OSACopyScriptingDefinition`<br>`OSACopyScriptingDefinition` is available in Mac OS X 10.4 and later; `OSAGetAppTerminology` is not available in 64-bit. |

`OSALoadFile` and `OSADoScriptFile` now correctly handle text scripts encoded as UTF-8. [4490939]

The various `OSADebugger` APIs, which have been marked as "not implemented" for some time, have been completely removed. [3918369]

## Scripting Additions

Scripting additions may be written using a new architecture which is both easier to write and improves performance. See *Scripting Additions for Mac OS X* for details. [4236732]

The scripting addition loader will search the directory specified by the environment variable `DYLD_FRAMEWORK_PATH`. Xcode sets this variable to the build product directory when you run your executable, which means you can run and debug a scripting addition project directly from Xcode without first installing the scripting addition. [5027805]

# 10.4 Changes

This article describes changes to AppleScript and related tools in Mac OS X v10.4 and its updates.

## Mac OS X v10.4 - AppleScript 1.10

AppleScript 1.10 is included with Mac OS X version 10.4 and requires Mac OS X version 10.4 or later. It contains many new features and enhancements, and corrects problems found with AppleScript 1.9.3 and earlier versions.

AppleScript 1.10 can use scripts developed for any version of AppleScript from 1.1 through 1.9.3, any scripting addition created for AppleScript 1.5 or later for Mac OS X, and any scriptable application for Mac OS 7.1 or later.

A script created with AppleScript 1.10 can be used by any version of AppleScript back to version 1.1, provided it does not use features of AppleScript, scripting additions, or scriptable applications that are unavailable in that version.

### Special Notes

- Scripts that compare AppleScript version numbers using string comparisons will need to be changed in order to work correctly with the new AppleScript version number. A `considering numeric strings` statement must be added before the string comparison so that the string "1.10" will be greater than the string "1.9.3".
- Scripts that compare version numbers by coercing the version class to a real number and then doing numeric comparisons will need to be changed. The coercion from `version` to `real` has been changed to allow the second and third parts of the version number to be 0..15 instead of 0..9. As a result, the version 1.2.3 will now coerce to the real number 1.0203. Previously, it would coerce to the real number 1.23.
- Scripts that compare AppleScript version numbers, and which need to run on AppleScript 1.10 as well as earlier versions of AppleScript, should do numeric comparisons using the Gestalt version number. The Gestalt version number can be obtained using the `system attribute "ascv"` command.
- String concatenation now uses the richer of the two operands. If either operand is Unicode, the result will be Unicode.
- The `path to me` command now returns the correct result for compiled scripts, including those that are run using the `run script` command.

- Beginning in Mac OS X version 10.4, the `mount volume` command can no longer mount Mac OS 9 server volumes if the server is specified using the AppleTalk server name. The TCP server name or IP address must be used instead.
- Do not use `sudo(8)` with the `administrator privileges` parameter of the `do shell script` command. Due to a bug, if `sudo` thinks it needs a password, it will prompt for it on a non-existent terminal and wait forever for a response, causing the script to hang. This may break existing scripts.
- The `do shell script` command now runs the shell script as seteuid-root, not actually root. This creates certain differences; for instance, perl will refuse to accept `–e` options when seteuid-root.
- The error codes returned by the `do shell script` command have changed.
- If the `with icon` parameter to the `display dialog` command specifies an icon that doesn't exist, an error is now reported. Previously, the error was ignored.
- The return value from the `list disks` command is now a list of Unicode strings. Previously, it was a list of plain strings.
- Getting the `count` of the `paragraphs` or `text items` of an empty Unicode string now returns 1 instead of 0.
- Strings that contain only a plus or minus sign, can no longer be coerced to a number.
- The implicitly encoded text types, `typeText`, `typeCString`, and `typePString`, are all deprecated as of AppleScript 1.9.2, since they are incapable of representing international characters and may be reinterpreted in unpredictable ways. Additionally, `typeCString` and `typePString` do not support the full range of text coercions, and will be removed entirely in a future release. `typeStyledText` and `typeIntlText`, while they have explicit encodings, are not recommended, since they are incapable of representing Unicode-only characters like Hungarian, Arabic, or Thai. The recommended text type is `typeUnicodeText`.

## Developer Notes

- A new Open Scripting Architecture (OSA) API has been added that lets you get the scripting dictionary of an application as an sdef (see ASDebugging.h and `sdef(5)`). If the target application does not have a true sdef but does have an `'aete'` or Cocoa-generated dictionary, it will translate that, so the API will work on any scriptable application. [3657719]

  ```
  OSAError OSACopyScriptingDefinition(const FSRef *ref, SInt32 modeFlags, CFDataRef *sdef);
  ```

  **Parameter Descriptions**

  `ref` An FSRef to the application file or bundle.

  `modeFlags` There are no flags currently defined; pass 0.

  `sdef` If the result is noErr, a CFDataRef containing the sdef XML, which you are responsible for releasing; otherwise undefined.

**Discussion**

To provide an sdef in your application, put the sdef in the Resources folder of your bundle, and add the key `OSAScriptingDefinition` to your Info.plist with the value of the sdef name (e.g., MyApplication.sdef).

**Known Bugs and Limitations**

Dynamic generation of sdefs is not supported. sdefs in single-file applications are not supported.

- A crash that occurred when calling `OSAScriptError()` with the `kOSAErrorApp` selector has been fixed. [3547609]
- The `osascript` command line tool now passes the `kOSAModeCompileIntoContext` mode flag when compiling a text script. [3684436]
- The `osascript` command line tool can now call scripts that take parameters. Any arguments following the script will be passed as a list of strings to the direct parameter of the run handler. [3165225]

## New Features and Enhancements

### AppleScript

- A new Considering/Ignoring attribute has been added that affects string comparisons. The `numeric strings` attribute specifies that numeric substrings should be collated by their numeric value. For example, "version 1.9" is less than "version 1.10". [3614412]
- A new string constant named `quote` has been added whose value is `"\""`. [3618573]
- String concatenation now uses the richer of the two operands. If either operand is Unicode, the result will be Unicode. [3375227]
- The `date` class now has `hours`, `minutes`, and `seconds` properties. The `hours` property always uses a 24-hour clock. [3516702]
- The `month` property of the `date` class can now be set to an integer. [3525546]
- The `weekday` constants can now be coerced to numbers. `Sunday` is 1. [3639194]
- The coercion from `version` to `real` has been changed to allow the second and third parts of the version number to be 0..15 instead of 0..9. As a result, the version 1.2.3 will now coerce to the real number 1.0203. Previously, it would coerce to the real number 1.23. [3621631]
- A `tell application "AppName"` statement can now use the application's `short name` (CFBundleName) if the application is currently running. [3655554]
- If a script application (applet) encounters an error, the Edit button in the error dialog will now open the applet in the default script editor. The default script editor can be set using the new AppleScript Utility application. [3084519]

- If an applet has a custom startup screen, the startup screen will now be displayed when the user selects the "About appletName" menu item. Previously, the generic applet About Box was displayed. [3537429]
- If a bundled applet has a custom startup screen stored in `Contents/Resources/description.rtfd/TXT.rtf`, that startup screen will be displayed instead of the startup screen stored in the bundle's `'TEXT'` and `'styl'` resources. Note that non-text startup screens are not currently supported. [3545405]
- Bundled applets now display localized menus and dialogs. [3847125] [3933933]
- The applet startup screen dialogs and error dialogs now conform to the Aqua Human Interface Guidelines. [3545405] [3535392]
- The applet About Box has been updated. [2632525]

## Standard Additions

- The `with prompt` and `with title` parameters to the `choose application` command now support Unicode text. [3526680]
- The `choose application` dialog now remembers the column widths. [2680232]
- The `choose file` and `choose folder` commands now have a `showing package contents` parameter that specifies whether packages should be treated as folders. The default is false. [3616391]
- The `of type` parameter to the `choose file` command now accepts either a list of file types or a list of type identifiers. [3226364]
- The `choose from list` command now has a `with title` parameter that specifies the title of the dialog. By default, the dialog does not have a title. [3722157]
- The `choose from list` command now supports Unicode text. [3318667]
- The `choose from list` command now supports long prompts and multi-line prompts. [2641867]
- The `choose from list` dialog now allows a text item to be selected by typing the first few characters of its name. [3497349]
- The `choose from list` dialog now supports full keyboard access. [2748218]
- A new `choose remote application` command has been added that enables the user to choose a running application on a remote machine or on the local machine. [2763939]
- The `choose URL` command now supports CIFS file servers. [3444824]
- A new `display alert` command has been added that displays a standard alert dialog that conforms to the Aqua Human Interface Guidelines. [3580135] Note that standard alert dialogs have the following restrictions:
    - A default button is required. The right-most button will be the default button unless the user specifies otherwise using the `default button` parameter.

    - The cancel button cannot be the same as the default button.
- The `display dialog` command now has a `cancel button` parameter that specifies the name or number of the cancel button. [3422345]
- The `display dialog` command now has a `hidden answer` parameter that specifies whether editable text should be displayed as bullets. The default is false. [1622240]
- The `display dialog` command now has a `with title` parameter that specifies the title of the dialog. By default, the dialog does not have a title. [3601825]
- The `default answer` parameter to the `display dialog` command now supports Unicode text and linefeeds. [3332444] [3316708]
- The `with icon` parameter to the `display dialog` command now accepts an alias or file reference to a `.icns` file. [3437405]
- The `display dialog` dialog now supports full keyboard access. [3272416]
- The `display dialog` dialog now uses Aqua icons for `note`, `caution`, and `stop`. [3437405]
- The `display dialog` dialog now conforms to the Aqua Human Interface Guidelines. [3438825]
- The `do shell script` command now displays the Mac OS X authentication dialog in order to obtain administrator privileges. [3004723]
- The `do shell script` command now has a `user name` parameter that specifies an administrator account. This parameter can be used along with the `password` parameter and the `administrator privileges` parameter in order to execute commands as an administrator without displaying an authentication dialog. [3004723]
- The `info for` command now has a `size` parameter that specifies whether the size of the item should be returned. The default is true. [3566020]
- The `info for` command now returns the `type identifier` for files. [3226360]
- The `info for` command now returns the `short name` (CFBundleName) for applications. [3655608]
- The `info for` command now returns the `busy status` for packages. The busy status is set to true if the package is marked as incomplete (kExtendedFlagObjectIsBusy). [3675978]
- The `info for` command now sets the `busy status` for non-bundled files to true if either the file is marked as open (kioFlAttribFileOpenBit) or the file is marked as incomplete (kExtendedFlagObjectIsBusy). [3976012]
- The `list disks` command now supports disk names that contain Unicode characters. The return value from the `list disks` command is now a list of Unicode strings. Previously, it was a list of plain strings. [3513241]
- A new `localized string` command has been added that returns a localized string for a specified key. [2857256]

- The `path to` command now has a folder constant for the Automator `workflows folder`. [3748836]
- The `path to me` command now returns the correct result for compiled scripts, including those that are run using the `run script` command. [3634573]
- A new `path to resource` command has been added that returns an alias to the specified resource. [3177652]
- The `say` command now accepts Unicode text and numbers as the direct parameter and as the `displaying` parameter. [3380562]
- A new `system info` command has been added that returns a record containing information about the system. [3611016]

## Bug Fixes

### AppleScript

- Strings that contain only a plus or minus sign, can no longer be coerced to a number. [2604425] [3759051]
- The numeric value classes, the enumerated value class (`typeEnumerated`), and the type value class (`typeType`) can now be coerced to and from Unicode text and styled text. [3333079]
- The Unit Type value classes (e.g., `miles`, `gallons`) can now be coerced to and from Unicode text. [3567987]
- The version class (`typeVersion`) can now be coerced to Unicode text. [3333229]
- Plain text (`typeText`) can now be coerced to styled text (`typeStyledText`). [3333079]
- A coercion from `typeFileURL` to object specifier has been added. [3244564]
- Coercing a file specification (`typeFSS`) to an object specifier now works correctly for filenames that are longer than 31 characters. [3078972]
- Coercing an alias to an object specifier would crash in certain circumstances if the alias could not be resolved. This has been fixed. [3556365]
- Distance and weight conversions are now more accurate. [3627006]
- `"file://localhost" as URL` no longer causes a crash. [3580636]
- Getting the first word of an empty Unicode string no longer causes a crash. [3619870]
- Getting an arbitrary element of Unicode text using `some` (e.g., `some word of theUnicodeText`) now produces more random results. [3479349]
- Getting the `count` of a non-existent element of Unicode text (e.g., `count files of theUnicodeText`) now returns 0. Previously, it returned the number of characters. [3735260]
- Getting the `count` of the `paragraphs` or `text items` of an empty Unicode string now returns 1 instead of 0. [3978194]

- A problem that caused some styled text comparisons to fail when the primary language was set to Japanese has been fixed. [3528969]
- `tell application "AppName"` will now look for a Mac OS X application named AppName.app before looking for a Classic application named AppName. Previously, the Classic application was found first. [2868512]
- Long application names are no longer truncated when used in a `tell` statement. [2854838] [3736065]
- Long application names are no longer truncated in the "Where is xxx?" dialog. [3874872]
- Long volume/folder/file names no longer cause errors. [3625871]
- Getting the `POSIX path` of a path string that is already POSIX now returns the correct results. [3405197]
- Scriptable Cocoa applications that contain Resource Manager resources, but do not contain an `'aete'` resource, are now recognized as scriptable. [3553719]
- If a `with timeout` statement specifies a timeout of more than 8947848 seconds, the timeout will be set to 8947848 seconds. Previously, an error was reported. [3570429]
- A compilation problem that caused the last character of an enhanced application URL specification to be removed has been fixed. [3577364]
- Backslash characters and Yen sign characters will now compile correctly when the primary language is set to Japanese. [3765766]
- The `'¬'`, `'÷'`, `'≠'`, `'≤'`, and `'≥'` characters will now compile correctly when the primary language is set to Japanese, Korean, or Chinese. [3427926] [3758459] [3828128]
- Bundled applets created using AppleScript 1.9.x always appear to be named "applet" when targeted from a remote machine. This has been fixed for bundled applets created with AppleScript 1.10 and later. [3625938]
- Bundled applets created using AppleScript 1.9.x erroneously display the Open in the Classic Environment checkbox in the Finder's Get Info window. This has been fixed for bundled applets created with AppleScript 1.10 and later. [3560457]
- The size of the internal symbol table has been increased which enables larger scripts to be compiled. [3465382]
- The size of the internal runtime stack has been increased which enables larger static lists to be created. [3612848]
- The default formatting is now localizable. [3381966]
- The References formatting category is no longer used. [3394917]
- A divide by 0 within a `try` block within a `repeat` block no longer causes a hang. [3735291]
- Crashes that would occur in certain circumstances after garbage collection have been fixed. [3410634] [3783112]

- Memory leaks have been fixed. [3400572] [3542919]
- Several error reporting problems have been fixed. [3537493] [2361457] [3481518] [3382032] [3753807] [3756205] [3775760]

## Standard Additions

- The `choose application`, `choose from list`, `choose URL`, and `display dialog` commands will no longer present dialogs on remote machines. [3342807]
- The `choose application` command no longer leaks memory. [3505723]
- In Mac OS X version 10.3.x, the `choose application` dialog is not responsive to mouse clicks within the content area when inactive. This has been fixed. [3467405]
- The `of type` parameter to the `choose file` command now works correctly for packages that do not contain a PkgInfo file (e.g., scripting additions). [3742878]
- `choose file of type {"MooV"}` now works as expected. Previously, it allowed the user to choose any file that QuickTime could open. [3595938]
- The `choose file name` dialog now recognizes packages. [4003078]
- The `display dialog` command will no longer create a dialog that is too tall for the screen when given a very long string to display. [3464996]
- The `display dialog` command now displays Japanese text correctly when an applet is running on a Japanese-primary system. [2812677]
- The `do shell script` command now works correctly with multiple commands when the `administrator privileges` parameter is specified. [3466509]
- When the `administrator privileges` parameter is specified to the `do shell script` command, the 5 minute authentication timestamp now applies only to the current script. [3004723]
- The `do shell script` command now returns the correct error code (-128) if the user cancels the authentication dialog. [3446874]
- The `info for` command now returns the correct `file type` and `file creator` for packages that do not contain a PkgInfo file (e.g., scripting additions). [3153056]
- `read before <delimiter> as date` no longer causes an error. [3383413]
- The `run script` command no longer causes a stack overflow error if the direct parameter is not specified. [3514354]
- The `using` parameter to the `say` command now works correctly when Speech Recognition is turned on. [3055088]
- The `using` parameter to the `say` command no longer causes an error if the specified voice is not capitalized correctly. [2361457]

- `the clipboard as Unicode text` now works correctly when the clipboard contains styled text but no Unicode text. [3668725]
- Error checking and reporting has been improved. [3523658] [3522599] [2361457] [3413730] [3446766]
- Several errors in the Standard Additions dictionary have been fixed. [3512009] [3431746] [3632368] [3608653] [3505597]

# Mac OS X v10.4.2 - Standard Additions 1.10.1

Standard Additions 1.10.1 is included with Mac OS X version 10.4.2 and requires Mac OS X version 10.4 or later. It contains changes to the `do shell script` command. Note that only Standard Additions has changed in this release, the AppleScript runtime has not changed and its version number remains 1.10.

## Bug Fixes

### Standard Additions

- `do shell script ... with administrator privileges` now sets both the real and effective user ids of the script to 0. This matches the behavior in Mac OS X version 10.3.x, and corrects a number of permission troubles in Mac OS X versions 10.4.0 and 10.4.1. [4126949]
- Using `sudo(8)` with the `administrator privileges` parameter of the `do shell script` command will no longer cause the script to hang. In general, adding `sudo` to `with administrator privileges` is redundant, but some scripts rely on it. [4126949]

# Mac OS X v10.4.3 - AppleScript 1.10.3

AppleScript 1.10.3 is included with Mac OS X version 10.4.3 and requires Mac OS X version 10.4 or later.

## Developer Notes

- The `OSAGetPropertyNames()` and `OSAGetHandlerNames()` APIs now return all of the properties and handlers of a script object when the script object contains a parent property declaration. Previously they would only return properties and handlers that were declared before the parent property. [4179456]
- AppleScript 1.10.3 now respects the normal rules when trying to fetch terminology from the host application: it loads an `'aete'` resource if one is present, or sends an event if the application is marked as having dynamic terminology. Previously, it would always send an event. This only affects scriptable applications that run scripts in their own process. [4250843]

## Bug Fixes

### AppleScript

- The `delay` command now yields more time to other processes. [4179466]

### Standard Additions

- `do shell script ... with administrator privileges` no longer slows down when called more than once in the same script. [4179474]
- The `get volume settings` command now returns the correct `output volume` when the balance is set to the right of center. [4242153]
- The performance of the `system info` command has been improved. [4182950]

# Mac OS X v10.4.6 - AppleScript 1.10.6

AppleScript 1.10.6 is included with Mac OS X version 10.4.6 and requires Mac OS X version 10.4 or later.

## Bug Fixes

### AppleScript

- Changes to `property` values are now preserved when a script is run from the Script menu. In previous versions of AppleScript, the property values were only preserved if the script returned a result. [4400567]
- A problem that caused the `path to`, `path to it`, and `path to application "AppName"` commands to return the wrong result in certain situations has been fixed. [4400329]
- A compilation problem that caused the `after` reference form to be converted to `before` on Intel-based Macs has been fixed. [4460185]
- Crashes triggered by various Dashboard widgets and applications in certain situations have been fixed. [4400300] [4400304] [4400306] [4400309] [4400557]
- A memory leak has been fixed. [4400420]

### Standard Additions

- A problem that could cause the `choose application` dialog to be invisible has been fixed. [4400378]
- A problem that could cause `choose remote application` to be unable to find applications on remote machines using Bonjour has been fixed. [4400386]

# Mac OS X v10.4.7 - AppleScript 1.10.7

AppleScript 1.10.7 is included with Mac OS X version 10.4.7 and requires Mac OS X version 10.4 or later.

## Bug Fixes

### AppleScript

- Quitting a read-only bundled applet no longer causes an error to be reported. [4505459]
- Compiled scripts containing dates now decompile and run correctly when transferred from PowerPC-based Macs to Intel-based Macs (and vice-versa). [4525261]
- `delay` commands and `on idle` handlers no longer conflict. [4508927]
- Concatenating an empty record no longer causes a crash. [4510216]
- A crash that could occur in certain circumstances when copying large nested script objects has been fixed. [4510228]
- A memory leak has been fixed. [4505510]

### Standard Additions

- Specifying a data type of `''TEXT''` for a `read` or `write` command (e.g., `read fileRef as ''TEXT''`) no longer causes an error on Intel-based Macs. [4508534]

# 10.3 Changes

This article describes changes to AppleScript and related tools in Mac OS X v10.3 and its updates.

## Mac OS X v10.3 - AppleScript 1.9.2

AppleScript 1.9.2 is included with Mac OS X version 10.3 and requires Mac OS X version 10.3 or later. It contains many new features and enhancements, and corrects problems found with AppleScript 1.9.1 and earlier versions.

AppleScript 1.9.2 can use scripts developed for any version of AppleScript from 1.1 through 1.9.1, any scripting addition created for AppleScript 1.5 or later for Mac OS X, and any scriptable application for Mac OS 7.1 or later.

A script created with AppleScript 1.9.2 can be used by any version of AppleScript back to version 1.1, provided it does not use features of AppleScript, scripting additions, or scriptable applications that are unavailable in that version.

### Special Notes

- The implicitly encoded text types, `typeText`, `typeCString`, and `typePString`, are all deprecated as of AppleScript 1.9.2, since they are incapable of representing international characters and may be reinterpreted in unpredictable ways. Additionally, `typeCString` and `typePString` do not support the full range of text coercions, and will be removed entirely in a future release. `typeStyledText` and `typeIntlText`, while they have explicit encodings, are not recommended, since they are incapable of representing Unicode-only characters like Hungarian, Arabic, or Thai. The recommended text type is `typeUnicodeText`.
- The `offset` command now ignores case by default and honors `considering` and `ignoring` statements with regards to `case` and `diacriticals`. See the offset section below for more information.
- Resource-fork script files can no longer be created using `store script`. See the store script section below for more information.
- The `path to me` command will return a path that ends with a colon when used within a script application that has been saved in the new bundled format.

## Developer Notes

- Four new functions have been added to the Open Scripting Architecture (OSA) API in Mac OS X version 10.3. The new functions enable applications to load, execute, and store scripts using any of the old (data/resource fork based) or new (bundled) script formats. The new functions are `OSALoadFile()`, `OSALoadExecuteFile()`, `OSAStoreFile()`, and `OSADoScriptFile()`. For details, see the Open Scripting Architecture Reference.
- AppleScript now sends the subject parameter in some cases where it previously failed to. This allows applications to make the `at` parameter for `make new` optional. [3039070]
- System terminology is now read into memory only when needed for compilation or decompilation. This reduces the memory cost for applications that simply run compiled scripts. [3142099]
- AppleScript now preserves international characters better when compiling Unicode source. [3157261]

## New Features and Enhancements

### AppleScript

- AppleScript 1.9.2 supports a new bundled format for compiled scripts and script applications (applets) that uses standard Mac OS X bundles. Compiled script bundles have the filename extension `.scptd`, while applet bundles have the filename extension `.app`. [3150267] [3150248]
- Scripting additions can be embedded within bundled applets by placing them in a folder named `Scripting Additions` inside the bundle's `Contents/Resources/` folder. Note that Script Editor does not look for embedded scripting additions when editing bundled applets. Any required scripting additions must be properly installed in the normal locations during script development so that Script Editor can find them. [3314117]
- AppleScript applets now handle incoming events in a first-in, first-out manner. Prior to AppleScript 1.9.2, applets handled incoming events in a last-in, first-out manner. [2553648]
- The Idle handler return value can now be a real number. [3199911]
- Real numbers can now be coerced to integers even if they have a non-zero fractional part. Rounding is used to do the coercion. [2849518]
- The `month` constants can now be coerced to numbers. [2396328]
- The `date` class now has a `short date string` property. The format of the short date string is specified by the International preference pane. [2396328]
- Two new types were added: `JPEG picture` and `GIF picture`. [3251772]
- The `zone` class has been removed since EPPC over AppleTalk is not supported in Mac OS X. [2967694]

- When a non-URL `machine` name is specified, it is now assumed to be a Rendezvous name instead of an AppleTalk name. [3113390]

## Standard Additions

- The `choose application` dialog box now allows an application to be selected by typing the first few characters of its name. [3064133]
- The `choose application` dialog box now displays localized application names. [2970962]
- The `choose application` dialog box now remembers its size and position. [3141005]
- A `choose color` command has been added which displays the standard color picker dialog box and returns the selected color as a list of the RGB values. The `default color` parameter specifies the initial color. By default, the initial color is black. [3160380]
- The `choose file`, `choose file name`, and `choose folder` commands now have a `default location` parameter that specifies which directory should be displayed in the dialog box. [3143207]
- The `choose file` and `choose folder` commands now have an `invisibles` parameter that specifies whether invisible files and folders should be displayed. The default is true for `choose file` and false for `choose folder`. [2690829]
- The `choose file` and `choose folder` commands now have a `multiple selections allowed` parameter that specifies whether multiple items can be selected. The default is false. [3254466]
- The `choose file` and `choose folder` commands now support Unicode and styled text prompts. [2836896] [3318664]
- The `choose file` and `choose folder` dialog boxes no longer use a default prompt. The dialog box will only contain a prompt if one has been specified using the `with prompt` parameter. [2519050]
- The `choose file name` dialog box now has a new title and default prompt. [2519050]
- The `delay` command now accepts real numbers. [2812515]
- The `display dialog` command now supports Unicode and can now display more than 255 characters. [3077453]
- The `do shell script` command now has an `as` parameter that specifies the desired type of the result. The default is to return the result as UTF-8 if possible. If the `as` parameter is not specified and the result is not valid UTF-8, the result will be returned as `typeText`. [3080630] [3157877]
- Two new properties were added to the `file information` class. The `kind` property returns the item's kind string. The `bundle identifier` property returns the item's bundle identifier, if the item is a package. [2884166] [3274781]
- The `info for` command now returns the `short version` and `long version` for packages. [2913766]
- The `load script` command now supports the new bundled format for compiled scripts and applets. [3177719]

- The `offset` command now ignores case by default and honors `considering` and `ignoring` statements with regards to `case` and `diacriticals`. As a result of these changes, the offset command now behaves the same as other AppleScript string comparisons with regards to case and diacriticals. In AppleScript 1.9.1 and earlier, the offset command always considers case and does not honor considering and ignoring statements. Scripts that are expecting `offset` to always consider case will need to add a `considering case` statement to maintain the previous behavior. [1068386]
- The `path to` command now has a `folder creation` parameter that specifies whether the folder should be created if it does not exist. The default is to create the folder if possible. If `without folder creation` is specified, an error will be returned if the folder does not exist. [3315615]
- The `path to` command now has the following new folder constants: applications folder, documents folder, favorites folder, home folder, library folder, movies folder, music folder, pictures folder, public folder, shared documents, shared documents folder, sites folder, utilities folder. [2650692]
- The `path to` command now has a `Classic domain` constant that can be used with the `from` parameter. [2836501]
- The `run script` command now supports the new bundled format for compiled scripts and applets. [3177719]
- The `shutdown` folder constant has been renamed `shutdown folder` to prevent confusion with the `shut down` Finder command. [2748241]
- The `store script` command can now store into any format script file including the new bundled format script files. In addition, `store script` can now create a new script file in bundled format (compiled script or applet) or data-fork format (compiled script only). When creating a new script file, the filename extension determines which format is created. Resource-fork script files can no longer be created using `store script` but existing ones can be updated. When storing a script into an existing file, the file must be a script file; previously, the file could contain anything and an `'scpt'` resource was simply added to the file. [3177719]
- The `store script` dialog box is now resizable and moveable. [3148202]
- The `store script` dialog box now has a new title and default prompt. [2519050]

## Command Line Tools

- The `osacompile` tool now supports the new bundled format for compiled scripts and applets. If the `-o` option is specified and the file does not already exist, `osacompile` uses the filename extension to determine what type of file to create. If the filename ends with `.app`, a bundled applet will be created. If the filename ends with `.scpt`, a bundled compiled script will be created. Otherwise, the resulting script will be placed in the resource fork and/or data fork of the output file depending upon what other options are specified. [3150246]

- The `osacompile` tool now has a `–s` (stay open) option. This option is only valid when a new bundled applet is being created. The option specifies that the applet should stay open after running. [3231791]
- The `osacompile` tool now has a `–u` (use startup screen) option. This option is only valid when a new bundled applet is being created. The option specifies that the applet should display a startup screen when launched. [3231800]
- The `osascript` tool now supports the new bundled format for compiled scripts and applets. [3150246]
- The `osascript` tool now reports character range information when an error occurs. [3310364]

## Bug Fixes

### AppleScript

- In AppleScript 1.9.1 and earlier, pathnames of `alias` objects may have problems if they contain non-ASCII characters. This has been fixed. [2765349]
- Coercing `miles`, `yards`, or `feet` to `inches` now produces the correct result. Coercing `cubic yards` or `cubic feet` to `cubic inches` now produces the correct result. [3251787]
- Coercions from `typeUnicodeText` to `typeText` or `typeStyledText` now handle byte-order-mark characters correctly. [3115555]
- Coercions from `typeStyledText` to `typeUnicodeText` now handle coercion failures and warnings correctly. As a result of this change, the example scripts can now be opened in Script Editor when the primary language is set to Korean. [3309458]
- An empty `with timeout` statement no longer causes a crash or stack overflow. [2354124]
- An empty `considering` or `ignoring` statement no longer causes a crash or stack overflow. [3243780]
- The statement `paragraphs 2 thru –1 of ""` no longer causes a crash. [3342312]
- Getting the `characters` of a very long string no longer causes a stack overflow. [3016960]
- Getting the `text items` of a Unicode string now works correctly. [3136465]
- Getting the `text items` or `paragraphs` of an empty Unicode string now returns {""} instead of {}. [3198302]
- Getting the `text items` of a string now works correctly when the `text item delimiters` are set to {}. [3146794]
- The arbitrary element reference form `some item` wasn't very random. This has been fixed. [3044302]
- The `contains` and `ends with` operators now work correctly on strings with more than 32,767 characters. Previously, they would always return false if they did not find a match within the first 32,767 characters of the string. [3126655]

- The containment operators now work correctly on Unicode strings when used within an `ignoring punctuation` statement. Previously, they did not ignore punctuation marks correctly when doing the string comparisons. [3328867]
- The containment and comparison operators now work correctly on Unicode strings when used within an `ignoring hyphens` statement. Previously, they did not ignore hyphens when doing the string comparisons. [3329084]
- The containment and comparison operators now work correctly on Unicode strings when used within an `ignoring white space` statement. Previously, they did not ignore white space when doing the string comparisons. [3330635]
- The containment operators are now much faster when used on Unicode strings. [3204625]
- Constant Unicode strings that are specified using the `data` class (i.e. `«data utxt»`) can now be coerced to `typeUnicodeText` and no longer cause a crash when used with the `&` operator. [3222690]
- In certain circumstances, a `run script` command within a `tell application "Finder"` statement would cause a crash in previous versions of AppleScript. This has been fixed. [3380899]
- AppleScript can now compile URLs that begin with `https`. [3113313]
- AppleScript now compiles numbers larger than 4294967295 correctly. Older versions of AppleScript compile numbers larger than 4294967295 to their value minus 4294967296. [3247815]
- Exponent math is now correct when the power is greater than 32767 and less than 65536. [3396181]
- Adding a negative number to any date between 1/1/1904 0:00:00 and 1/19/1972 3:14:07 now produces the correct result. [3160911]
- Subtraction operations involving dates earlier than 1/1/1904 0:00:00 now produce the correct result. [2861317]
- Date arithmetic involving date object specifiers now works as expected instead of reporting an error that says the object specifier can't be coerced to a date. [3187787]
- If an error occurs while attempting to connect to a remote machine, the actual error is now reported. Previously, connection errors were converted into a generic message saying that the application's dictionary could not be read. [3209734]
- If a compilation error occurs near the end of a long script (more than 32,767 characters) the correct line is now highlighted in the script editor. [3089998]
- Backslashes in strings now work correctly on Japanese systems. [3390330]
- If a script tells an application that is not running to `quit`, nothing will happen. In previous versions of AppleScript, the application would be launched and then immediately be told to quit. [2589455]
- AppleScript no longer crashes while attempting to unload an OSAX written in Objective-C. [3193941]
- A crash that would occur in certain circumstances after garbage collection has been fixed. [3369338]

- Memory leaks have been fixed. [3254901]
- A couple of incorrect error messages have been fixed and a missing error message has been added. [3020705] [3290768]
- AppleScript's default formatting has been changed. [3032595] If you have already used AppleScript on a given machine, you will not notice this change unless you do any of the following:
  - choose Use Defaults from the Formatting dialog box in Script Editor
  - delete your AppleScript preferences
  - set up a new user (or log in on a user account that has never used AppleScript)
  - do a clean operating system install without saving or retaining user preferences

## Standard Additions

- The `display dialog` dialog box now handles the Return, Enter, and Esc keys correctly. [3044290] [3096706]
- The `display dialog` command now reports an error if the `default button` parameter is set to 0, a negative number, or a number that is greater than the total number of buttons. [3051124]
- The `display dialog` command now reports an error if a record is specified as the text to display. [3142086]
- `do shell script with administrator privileges` now distinguishes between a cancelled authentication and a failed authentication. [3070647]
- `do shell script with administrator privileges` no longer reports an authentication error if the admin user has no password. [2882577]
- The `info for` command no longer returns the `folder window` for packages. The `visible` property is now correct for items whose name begins with a period. [2884166]
- The `list folder` command no longer crashes when given a file instead of a folder. [3267269]
- `list folder without invisibles` no longer lists items whose name begins with a period. [2767663]
- In certain situations, the `open for access` command would try to create a file that already existed which would result in a "duplicate file name" error. This has been fixed. [2827608]
- The `open for access` command can now create a file whose name contains slashes or colons (whichever is not the native path separator for the path style in use). [3267270]
- The `path to` command now works correctly with the following folder constants: `startup`, `startup items`, `startup items folder`. [2650692]
- Scripts that use the `path to frontmost application` command will now get the correct path when run from the Script menu. Previously, the command would return the path to System Events. [3084984]

- The `read` command now works correctly when the file contains Unicode text and the `before`, `until`, or `using delimiter` parameters have been specified. [3013987]
- If the `run script` command encounters an error while compiling the script, it now reports the actual error instead of a generic script error. [2304013]
- A problem that caused the `say` command to hang has been fixed. [3196145]
- The `set volume` command now sets the volume level correctly. [3268130]
- `the clipboard` now works correctly if the `as` parameter is specified. Previously, the `as` parameter was almost completely ignored. [3255134]
- `the clipboard` now works correctly if the `as` parameter is not specified. If the clipboard contains only one item, that item is returned. If the clipboard contains multiple items and at least one of the items is text, the highest quality text item will be returned. Unicode text is preferred over styled text which is preferred over plain text. If the clipboard contains multiple items but none of the items is text, the items are returned as a record. [3334818]
- Several errors in the Standard Additions dictionary have been fixed. [2696269] [3034675] [3152145] [3292454] [3311842]

# Mac OS X v10.3.2 - AppleScript 1.9.3

AppleScript 1.9.3 is included with Mac OS X version 10.3.2 and requires Mac OS X version 10.3 or later.

## Developer Notes

- In AppleScript 1.9.2, coercing an AEDesc to Unicode text will fail if the text is a null string. This has been fixed. [3439141]

## Bug Fixes

### AppleScript

- In AppleScript 1.9.2, the `with timeout` statement does not return an error if the timeout expires. This has been fixed. [3457829]
- When using a Japanese-primary system, scripts that compile correctly with AppleScript 1.9.1 may fail to compile with AppleScript 1.9.2. This has been fixed. [3419243]
- In AppleScript 1.9.2, saving a blank script with Script Editor 2.0 will result in a crash. This has been fixed. [3421468]

- In AppleScript 1.9.2, using Xcode to open a new `.applescript` file that was created by Interface Builder will result in a crash. This has been fixed. [3454301]
- AppleScript Studio debugging has been improved. [3467973] Note, AppleScript Studio debugging requires Xcode 1.1.

### Standard Additions

- In AppleScript 1.9.2, the `mount volume` command will fail if the user name is not specified and the volume does not allow guest access. This has been fixed. [3464919]

## Mac OS X v10.3.5 - Standard Additions 1.9.4

Standard Additions 1.9.4 is included with Mac OS X version 10.3.5 and requires Mac OS X version 10.3 or later. It contains improvements to the `set volume` command as well as a new `get volume settings` command. Note that only Standard Additions has changed in this release, the AppleScript runtime has not changed and its version number remains 1.9.3.

### New Features and Enhancements

#### Standard Additions

- A new `get volume settings` command has been added that returns a record containing information about the current volume settings. [3467967]
- The `set volume` command now has four new parameters that enable the user to set the `output volume`, the `input volume`, the `alert volume`, and the mute setting (`output muted`). The direct parameter is now deprecated. If the direct parameter is specified, all other parameters are ignored. [3563760]

### Bug Fixes

#### Standard Additions

- The `set volume` command now sets the volume level correctly. [3268130] [3479316]

# 10.2 Changes

This article describes changes to AppleScript and related tools in Mac OS X v10.2 and its updates.

## Mac OS X v10.2 - AppleScript 1.9

AppleScript 1.9 is included with Mac OS X version 10.2 and requires Mac OS X version 10.2 or later. It contains several new features and enhancements, and corrects problems found with AppleScript 1.8.3 and earlier versions.

AppleScript 1.9 can use scripts developed for any version of AppleScript from 1.1 through 1.8.3, any scripting addition created for AppleScript 1.5 or later for Mac OS X, and any scriptable application for Mac OS 7.1 or later.

A script created with AppleScript 1.9 can be used by any version of AppleScript back to version 1.1, provided it does not use features of AppleScript, scripting additions, or scriptable applications that are unavailable in that version.

### Developer Notes

- In AppleScript 1.7 and earlier, the `OSASetProperty()` call would fail frequently. This has been fixed. [2786611]
- Object references of the form <class> <string> have historically been sent to applications as by name object specifiers. In AppleScript 1.5 through 1.8.3, if the <string> was `typeUnicodeText` the object specifier would be sent as `formAbsolutePosition`. In AppleScript 1.9 and later, object references using Unicode strings are always sent to applications using `formName`. (Note that all other scalar types, including `typeUTF8`, are sent as `formAbsolutePosition`). [2768306]
- Developers can now define application scriptability using an XML description, rather than old-style `'aete'` resources or Cocoa `.scriptSuite` and `.scriptTerminology` plist files. This makes it easier to write and edit an application terminology. A new developer tool, `/Developer/Tools/sdp`, converts XML files with the `.sdef` extension into either `.r` files (for Carbon applications) or `.scriptSuite` and `.scriptTerminology` files (for Cocoa). In the future, AppleScript will read `.sdef` files directly, making this conversion unnecessary. See the man pages for `sdp` and `sdef` for more information. Some sample `.sdef` files are supplied in `/Developer/Examples/Scripting Definitions`. [2912639] [2913622]

# New Features and Enhancements

## Standard Additions

- The `choose URL` command now displays SMB and Samba servers when the File servers pulldown menu is selected. [2810506]
- The `mount volume` command can now connect to any file server that is supported by the Finder's Connect To... command, including Windows (smb), Samba, and FTP servers. On some kinds of servers, the `as user name` and `with password` parameters may not bypass the login dialog, but encoding the name and password in the URL (e.g. `smb://myname:passwd@server.domain.com/sharename`) will mount it silently. [2855789]
- The `say` command supports a new optional parameter, `saving to`, which specifies a file in which to save the spoken text. If this parameter is provided, the command will be silent and will instead create an audio file with the data of the spoken text. For best results, the file extension of this file should be `.aiff`, which will make it useable with iTunes and other audio applications. [2898296]

## Script Menu

- The Script Menu menu extra that has been available for download from the AppleScript web page is now included with Mac OS X. To activate the Script menu, open the `AppleScript` folder (in the `Applications` folder of your startup disk) and double-click on the `ScriptMenu.menu` folder. The Script menu will appear on the right side of the menu bar in all applications.
- Like Script Runner in previous versions of Mac OS X, the Script menu lets you execute any script while using any application. It lists all files that are present in the `Scripts` folder of the `Library` folder of both your startup disk and your Home directory. Unlike Script Runner, the Script menu will execute both OSA scripts (saved by Script Editor) and UNIX scripts (shell scripts, perl scripts, etc.).

## Folder Actions

- Folder Actions are now supported in Mac OS X. Folder Actions let you attach a script to a folder, and when that folder is opened or closed in the Finder or files are changed in it, a handler in that script will be executed.
- In Mac OS X, Folder Actions are handled by the System Events application, not a separate Folder Actions extension. A property in the System Events application turns Folder Actions on and off; when they are on, System Events is added to your Login Items list and starts up when you log in. Folder Actions will not be performed if the System Events application quits.

- Actions are attached and removed from folders by sending commands to System Events. The same events that work with the Folder Actions extension on Mac OS 9 will work the same with the System Events application in Mac OS X; in addition, System Events supports an object model for managing actions on a folder. See the System Events dictionary for the complete set of Folder Actions events and classes.
- The action events that are sent to action scripts are the same as in Mac OS 9. Terminology for these events can be found in the Standard Additions scripting addition.
- There are some functional differences between Folder Actions in Mac OS X from the Mac OS 9 version. In Mac OS X, actions are assigned to folders on a per-user basis, so different users of the same machine can have different sets of actions. That means that actions attached to removable or shared disks will not take effect unless that action has been assigned on your specific user account. This is a security precaution. This also means that actions attached to folders in Mac OS 9 will not take effect in Mac OS X unless they've been specifically attached for a particular user account in X.
- Folder badging works differently in Mac OS X than in Mac OS 9. When an action is assigned to a folder, no badge appears in Mac OS X; folders that have had actions attached in Mac OS 9 will have the Folder Action badge, but will not execute their actions unless they've been assigned to that user.
- The contextual menu that assigns, removes, and edits Folder Actions in Mac OS 9 is not present in this version of Mac OS X. Example scripts to assign and remove Folder Actions are supplied in the `/Library/Scripts/Folder Actions` directory, and can be used directly from the Script menu.
- Because of changes in the Finder, many Folder Action scripts from Mac OS 9 do not work in Mac OS X. [2865598]

## Bug Fixes

### AppleScript

- In AppleScript 1.8.2 and later, certain errors in handlers of scripts loaded via the `load script` command would cause a crash. In AppleScript 1.9 these errors produce error messages instead. [2891612]
- Operations that create large AppleScript objects (such as lists with thousands of elements) may fail when trying to display the object (for example, in the Results or Log window of Script Editor), though the operation itself succeeds. In AppleScript 1.8.3 and earlier this usually crashes; in AppleScript 1.9 it will signal an "out of memory" error. [2897872]
- AppleScript automatically converts any result value that is returned as Unicode text in UTF-8 format into Unicode text. To convert a string value to the UTF-8 format, use the coercion `as «class utf8»` [2936516]

### Standard Additions

- The `do shell script` command in Standard Additions 1.8.3 and earlier could not handle file paths that contained accented or non-English characters. This has been fixed. [2848082]
- In Standard Additions 1.8.2 and later, the `display dialog` command does not activate the default text field when the dialog is displayed, so the user must click in it before typing. This has been fixed. [2915953]
- With Standard Additions 1.8.3 and earlier, the `the clipboard` command would return an erroneous value when the text on the Clipboard did not have style information. If passed to other applications, that erroneous value could cause those applications to crash. This has been fixed. [2811798]

## Mac OS X v10.2.3 - AppleScript 1.9.1

AppleScript 1.9.1 is included with Mac OS X version 10.2.3 and requires Mac OS X version 10.2 or later.

### Developer Notes

- A new `modeFlag` has been added to the OSA API:
- When this `modeFlag` is set on a call to `OSACoerceToDesc()`, the resulting descriptor will be fully qualified, that is, it will have as its outermost container (innermost 'from' object) an application descriptor. Historically this function assumes all object specifiers are rooted at the current application, which is often incorrect. [2879211]

### Bug Fixes

#### AppleScript

- In AppleScript 1.0 through 1.9, coercions from strings to numbers (e.g. `"12" as integer`) would fail if the string contained leading or trailing white space characters (e.g. space, tab, return, linefeed). This has been fixed. [2849044]
- In AppleScript 1.7, the `paragraph` element of strings was improved to recognize Mac-style, UNIX-style, and Windows-style line breaks. The same changes were not, however, made to the `Unicode text` class. This has been done in AppleScript 1.9.1. [3013979]
- In AppleScript 1.6 through 1.9, getting a `paragraph` or `text item` of an empty Unicode string fails with an error. In AppleScript 1.9.1 an empty string is now returned as the result. [3044226]
- With AppleScript 1.7 through 1.9, some script-editing applications cannot open their own scripting dictionaries. This has been fixed. [2688067]

- In AppleScript 1.9, long scripts (from about 3,000 lines, depending on complexity) would cause an out of memory error when being displayed. This has been fixed. [3016146]
- In AppleScript 1.0 through 1.9, a long script (greater than 32,767 characters) could cause a crash when being compiled. This has been fixed. [2800857]
- In AppleScript 1.0 through 1.9, an execution error that occurred near the end of a long script (greater than 28,671 characters) could result in multiple lines being highlighted in the script editor instead of just the line that caused the error. This has been fixed. [1342368]
- In AppleScript 1.9 and earlier, comparing two strings will fail if one string contains more than 32,767 characters, even if the strings are obviously different. This has been fixed. [2655090]
- In AppleScript 1.9 and earlier, the length of a long literal string (greater than 32,767 characters) is considered to be its actual length mod 32,768. This has been fixed. [2731384]
- In AppleScript 1.6 through 1.9, compiling a script might cause it to unnecessarily launch an application referred to in a `tell` block of that script. This has been fixed. [2838223]
- In AppleScript 1.9 and earlier, canceling out of an EPPC authentication dialog can cause compilation errors. This has been fixed. [3006076]
- In AppleScript 1.8 through 1.9, the `POSIX path` property of a `file` or `alias` object would return "//" for the startup disk object. While not technically incorrect this has been disturbing to those familiar with UNIX, and has been fixed. [3006675]
- In AppleScript 1.6 through 1.9, recording a script would crash on relatively rare occasions. This has been fixed. [3006033]
- In AppleScript 1.9 and earlier, an AppleScript Studio application would crash in certain circumstances. This has been fixed. [3024472]
- A crash in AppleScript 1.6 and later that occurred when using Unicode text has been fixed in AppleScript 1.9.1. [3025031]
- In AppleScript 1.0 through 1.9, the expression `end of {}` has caused a crash. This has been fixed. [3014615]
- The expression `get name of parent` would crash AppleScript 1.9 and earlier in certain circumstances. This has been fixed. [2865830]

## Standard Additions

- Because of changes in the Sound Manager in Mac OS X, multiple beeps produced by the `beep` command in Standard Additions 1.6 through 1.9 have sounded like one long beep. This has been fixed. [2812503]
- When the `display dialog` command is used with three buttons and an icon, the dialog box is now tall enough to correctly display both the icon and the leftmost button. [3052663]

- In Standard Additions 1.6 through 1.9, the `mount volume` command could occasionally crash when supplied with `user name` and/or `password` parameters. This has been fixed. [3025436]

# 10.1 Changes

This article describes changes to AppleScript and related tools in Mac OS X v10.1 and its updates.

## Mac OS X v10.1 - AppleScript 1.7

AppleScript 1.7 is included with Mac OS X version 10.1 and requires Mac OS X version 10.1 or later.

### Developer Notes

- The return value from the `OSAScriptError()` function historically returned an error range that was a record of two short integers, and therefore could not accurately report errors in scripts longer than 32K. AppleScript 1.7 returns a record of two `typeLongInteger` values. Existing code that gets these values as `typeLongInteger` will now work correctly; applications that asked for short integers will continue to get erroneous results on long scripts and should be revised to accommodate the new return value.
- To support a wider variety of sources of script text, the compiler in AppleScript 1.7 accepts all three prevalent styles of line endings: Macintosh style (carriage return, ASCII 13), UNIX style (linefeed, ASCII 10) and Windows style (CRLF, ASCII 13-ASCII 10). Decompiled source will continue to be Macintosh-style.
- Every event sent by AppleScript to an application or scripting addition now includes an `enumConsidsAndIgnores ('csig')` attribute whose value is a `typeUInt32` bit field with bits set for every consideration or ignore currently in effect. Mask constants are defined in `ASRegistry.h`. This attribute obsoletes the older `enumConsiderations` attribute, which is difficult to interpret, incomplete, and is only sent with events that contain a `whose` clause.
- The `kAESave` and `kAEClose` Apple events have been defined in the Apple Event Registry to accept a `typeAlias` value in their `in` and `saving in` parameters. This is technically inaccurate as scripters may specify files that do not exist for these parameters, and `typeAlias` cannot represent a file that does not exist. The historical solution has been to supply a `typeFSS` value (or a `cFile` value which is coerced to a `typeFSS`) and create the file using the file specification. In Mac OS X, however, file specifications are not valid values in Apple events, as they cannot represent long or Unicode file names and are not necessarily valid across processes. For AppleScript 1.7 and later, applications should get these parameters as `typeFileURL` and use the sample code provided in Tech Note TN2022 to find or create the file it represents.

## New Features and Enhancements

- Program Linking: Program Linking now works between Mac OS X machines as well as between Mac OS 9 and Mac OS X machines. When targeting applications running on a Mac OS X machine, you must use TCP/IP networking and addressing; AppleTalk is not supported. (In addition, AppleScript 1.7 now supports encoding the login name and password into the eppc URL.) For a Mac OS X machine to respond to Program Linking requests, Allow Remote Apple Events must be enabled in the Sharing panel of System Preferences.
- SOAP and XML-RPC Support: AppleScript can now use Internet applications as targets of `tell` blocks. The URL for the application reference must be a correct URL for a SOAP (Simple Object Access Protocol) or XML-RPC (eXtensible Markup Language Remote Procedure Call) Web server. The syntax is:
- An alternate syntax is:
- but this form will be converted to the form above when the script is compiled.
- File Extensions: Like most other Mac OS X functions, AppleScript now supports file extensions (as well as the historical file types) on all the file kinds it deals with. The file types for AppleScript are:
- Applets: Prior to AppleScript 1.4.3, script applications (applets) were stored in a universal format that worked on both 680x0-based and Power PC-based Macintosh models. Because that format was 680x0-based, it would not work on Mac OS X, so a Mac OS X applet format was introduced in AppleScript 1.4.3. With improvements in Mac OS X since that time, it is now possible to produce a script application that works on 680x0-based, Power PC-based Mac OS 9, and Mac OS X configurations. Script Editor 1.7 again offers only one choice for saving script applications. Older formats continue to work on the configurations they were saved for.
- Command Line Tools: The `osacompile` and `osascript` tools now have an include option (`-i`) that allows you to specify application terminology to be used when compiling the given scripts. This frees you from having to specify applications in `tell` blocks in the script itself. You can also use this to specify the static terminology for an application with dynamic terms, so you can prevent that application from being launched when you compile the script.

## Bug Fixes

### AppleScript

- In AppleScript 1.5 through 1.6, a script's global variables could be lost under certain rare circumstances. If a script contains a `tell` block that targets a variable, and that variable specifies an application that does not exist, the "Where is application AppName?" dialog would be presented when the script is executed. After that point, further references to global variables in the handler that included the `tell` block would fail. This has been fixed.

- In AppleScript 1.5 and 1.6 for Mac OS X, scripting additions behaved fundamentally differently than in Mac OS 9: they were only installed in certain applications that were AppleScript-aware. If a scripting addition command (such as `display dialog`) was sent to an application that was not AppleScript-aware, the event would instead be executed in the application running the script (e.g. Script Editor); that means the dialog would come up in the Script Editor layer, not the `tell` target application. In AppleScript 1.7 for Mac OS X, the scripting additions mechanism has been redesigned to behave more like Mac OS 9, and scripting addition commands sent to applications will be executed in the process space and window layer of the target application.
- AppleScript 1.5 through 1.6 often launched applications unnecessarily in order to get their terminology, and often used the terminology of the script editing application (such as Script Debugger) instead of application terminology. This has been fixed.
- In AppleScript 1.5 through 1.6 for Mac OS X, the `version` property of AppleScript or an application would be displayed as raw data rather than a string (unless the version property was explicitly converted to a string). This has been fixed.
- In AppleScript 1.5 and 1.6 for Mac OS X, certain operations with `file` object specifiers (e.g. `file "Mac OS X:Applications:Terminal.app:" as alias`) would fail, when similar operations would succeed on AppleScript 1.3 through 1.6 for Mac OS 9. These operations now work in AppleScript 1.7 for Mac OS X. Note that, as always, while you can pass a `file` object specifier as a parameter to an event, or perform certain operations on it (like assignment and coercion), simply evaluating `file "pathname"` is itself an AppleScript error.
- AppleScript 1.5 and 1.6 for Mac OS X could not target Mac OS 9 control panels running in the Classic environment. This has been fixed. Note that the Classic environment only supports a small number of control panels from Mac OS 9 (such as the Launcher).
- Accessing `paragraphs` of AppleScript strings did not work correctly in AppleScript 1.5 through 1.6 if the line endings were Windows-style (CRLF). The result would be only the first half of the text paragraphs, interspersed with an equal number of blank strings. This has been fixed.
- Occasionally with AppleScript 1.5 and 1.6, recompiling a script would get an “end of file” error reading an application’s terminology. This has been fixed.
- In AppleScript 1.6, under certain circumstances calling a handler in an applet that has not been launched could cause a “Cannot continue” error. This has been fixed.
- In AppleScript 1.6, Unicode text values stored in script properties of a saved script could not be reloaded reliably. This has been fixed.
- In AppleScript 1.6, Unicode text received from certain applications that included a Unicode byte-order mark would behave incorrectly; the text would appear to begin with the Euro symbol, and some operations would fail. AppleScript 1.7 handles Unicode text with byte-order marks correctly.

- In somewhat rare circumstances, comparing two references to the same file for equality can fail even if both references refer to the same file. This is true in AppleScript 1.0 through 1.6 and has been fixed in AppleScript 1.7.
- In AppleScript 1.5 and 1.6, a running AppleScript applet will become non-responsive to any events (except command-period) after processing more than one incoming event. This has been fixed.

## Standard Additions

- All disks formatted as Enhanced Macintosh File System (HFS Plus) are able to store files with file names greater than 32 characters and stored as Unicode text. When older software accesses such files, those names are represented in an encoded form (with something like `#28AF` in the name). With AppleScript 1.6 and earlier, the results of many Standard Additions commands (like `path to`, `info for`, and `list folder`), as well as creation and display of `alias` objects, would require and return encoded names. In AppleScript 1.7, long and Unicode file names in aliases and for scripting additions are represented by styled or Unicode text. Encoded file names are no longer supported.
- In Mac OS X v10.0 through 10.0.4, the `choose file name` command in Standard Additions 1.6 would crash if given an empty string ("") as the default file name. This has been fixed. In addition, on Mac OS X the default file name can be a Unicode string.
- The `choose from list` command in Scripting Additions 1.4 through 1.6 allowed the scripter to create a situation where it would return an empty selection even if the scripter specifically disallowed that case. (This can be done by supplying a default item that does not appear in the list). In Scripting Additions 1.7 and later, the OK button will not be enabled in such cases.
- In versions of Standard Additions prior to 1.7, the `choose URL` command's `using editable URL` parameter was present in the Scripting Additions dictionary but was nonfunctional (URLs were always editable regardless of the setting of the parameter). It now behaves correctly: if set to false, the user can choose a URL only by browsing, not by editing the URL in the text box.
- In Mac OS X, the `info for` command now returns additional record items that provide information about the file's extensions. The `displayed name` item returns the name of the item as the user sees it in the Finder or file dialogs, and the `name extension` item now returns the file extension (if any) that is on the indicated file (without the ".", which is considered a separator, and not part of either the name or the extension). Both of these fields (along with the `name` field itself) are Unicode text values.
- Mac OS 9.1 and Mac OS X Public Beta introduced the concept of a package, that is, a file composed of a folder of several files and folders, which is treated as an integral file by the Finder and other applications. The `info for` command in AppleScript 1.6 and earlier treated such packaged data application files as folders, not files, and would not return file- or application-related information about them. In Standard Additions 1.7 and later, getting the information for any directory (including package folders) returns an additional record item `package folder` (if this Boolean value is true, then the directory is a package), and the file's type and creator.

- In Standard Additions 1.6 and earlier, the `info for` scripting addition would fail if any piece of information in the requested file or directory was unavailable. On UFS (Mac OS X) disks and mounted NFS volumes, the file creation date is unavailable, and accessing it causes a -8850 error (in trying to convert it to Universal Time). Starting with Standard Additions 1.7, any missing piece of information in the `info for` result is provided as the value `missing value` without failure. In AppleScript 1.6 and earlier, magnitude comparisons (<, □>, □) against the `missing value` would signal a coercion error for most types; in 1.7 and later, these comparisons always return `false` without error. These two changes ensure that scripts that use `info for` against UFS and NFS volumes do not fail, though they may not get correct results because any comparison to the creation date is always false.
- In Standard Additions 1.6 on both Mac OS 9 and Mac OS X, the `info for` command would return an error -1401 for a nonexistent file, rather than the normal -35 returned in previous versions. In Standard Additions 1.7, trying to get the info for a nonexistent file (or a file in a nonexistent directory or inaccessible disk) will return the -35 error.
- From AppleScript 1.1, the `offset` command has not worked correctly when the direct parameter is styled text. It also has not worked correctly with the newly-introduced `Unicode text` type. It now produces correct results with both styled and Unicode text.
- Standard Additions 1.5 through 1.6 changed the way that the random number seed worked in the `random number` command. Older versions used the system clock as the seed value if no seed value (or a seed value of 0) was provided; Scripting Additions 1.5 and 1.6 always use a seed value of 0 in these cases. In Scripting Additions 1.7 the older behavior has been restored. In addition, the seed value is now taken from the 60Hz "ticks" counter rather than the system clock, because the increased speed of new machines makes it more likely that two consecutive `random number` calls may receive the same system clock time as their seed value.
- The `read` command in Scripting Additions 1.6 had specific problems with the `using delimiters` parameter. In previous versions, `read f as list using delimiters {tab}` would return a list, but in AppleScript 1.6, it returns a list of one-item lists. In AppleScript 1.7 and later it behaves as it did prior to version 1.6
- The `read` command in Scripting Additions 1.6 had specific problems with the `before` and `until` parameters if used in conjunction with the `as` parameter. Instead of returning a value of a given type, it would return random data or an error. This has been fixed.

## Command Line Tools

- For the `osacompile` and `osascript` tools, the requirement that they be invoked with full paths (i.e. `/usr/bin/osacompile`) has been eliminated. They can now be invoked with implicit paths like any other command-line tool.

- In Mac OS X v10.0.0 through 10.0.4, the `-l` and `-d` options for the `osalang` tool did not work as documented. In Mac OS X v10.1 and later, `-l` prints the subtype, manufacturer, flags, and name; and `-L` also prints the description in parentheses after the name.
- The `osacompile` and `osascript` tools previously had a limit of 32K of input text. This limit has been lifted in Mac OS X v10.1.
- The `osacompile` and `osascript` tools execute as background applications; they cannot involve any user interface. If you want to call `display dialog`, `choose from list`, or other user-interface scripting additions from a script being executed by `osascript`, you must enclose it in a `tell` block targeting a running application (such as the Finder).

# December 2001 Developer Tools Update - AppleScript 1.8

AppleScript 1.8 for Mac OS X is included with the December 2001 Developer Tools Update and requires Mac OS X version 10.1 or later.

## Developer Notes

- In Mac OS X v10.1 and 10.1.1, scripting additions would have their initialization function called at application launch time, but their termination functions would not be called when the application quit. Scripting additions that used system resources would not get a chance to release those resources. Every scripting addition termination function is now called when an application exits. [2811696]

## New Features and Enhancements

### AppleScript

- To provide better access to Mac OS X's underlying UNIX functionality, AppleScript 1.8 adds a new Standard Additions command (`do shell script`), a new file property (`POSIX path`), and a new object (`POSIX file`).

### Standard Additions

- The new `do shell script` command and the scriptable Terminal allow AppleScript scripts to execute strings of text as UNIX shell scripts. Many interesting shell commands act on files, and require POSIX-style file paths as input parameters. To create POSIX-style file paths from AppleScript file objects, the property `POSIX path` has been added to the `file` and `alias` classes. All file types, including `file specification` and `file reference`, support this property. Its use is straightforward:

- The `do shell script` command and the scriptable Terminal allow AppleScript to execute UNIX shell scripts and get the results as strings of text. Frequently these strings are the pathnames of files in the UNIX format, i.e. starting at root with directory names separated by forward slashes. To convert these POSIX paths into an AppleScript file object, you can use the new `POSIX file` object:

- The new `do shell script` command takes a string as its direct parameter and executes that string in a UNIX shell. The text can be anything from a single command with parameters to a long script. The results of the script execution (standard out) are returned as the result of the command; if there is an error (or no output), the error output (standard error) are returned as the error message and can be processed in an AppleScript `on error` handler. Note that the environment in which a shell script is executed is a separate process in the user's process group, and does not have a terminal set up, so commands that require interaction (`vi`, `top`) will not function. There is no way at present to direct any information to standard in of a shell script. Note also that `do shell script` uses the `sh` shell, which is different than the default `tcsh` shell used in the Terminal. Command paths and environment variables are different in `sh`; you will need to adjust your shell scripts accordingly.
- The `system attribute` command has been extended to provide information on the user's environment variables. These are set up at user login time and cannot be altered. If the direct parameter to system attribute is a four-character Gestalt code, it is executed as in Standard Additions 1.7; if not, it attempts to get an environment variable by that name, returning either the value of the variable or an empty string. (Note that `system attribute` now will not fail with an error; it simply returns an empty string.) If called with no parameters, `system attribute` returns a list of all currently-defined environment variables.

## Bug Fixes

### AppleScript

- AppleScript 1.7 could not perform certain operations on file names or file paths that contain certain characters (such as accented characters like é). This has been fixed. [2816363]
- In AppleScript 1.7, references to the `process` class of the Finder on a remote machine did not work. This has been fixed. Note that you need to be running Mac OS X v10.1 or later on the target machine, and Mac OS X v10.1.2 or later on the machine executing the script. [2775419]
- With AppleScript 1.6 and 1.7, applications that attempted to convert Unicode text to plain text would get an error (though AppleScript itself could perform such conversions successfully). This prevented older applications from being able to use information (such as file names) that is now supplied as Unicode text. This has been fixed. [2816500]
- Certain development environments can create scripts whose parent is an application object, rather than a top-level script. Attempting to get any parent properties of such objects (including the information returned by `OSAGetScriptInfo`) would cause AppleScript to crash. This has been fixed. [2755096]

- With AppleScript 1.7 in Mac OS X v10.1, applets that are set to show their startup (splash) screens occasionally failed to do so. This has been fixed in AppleScript 1.8. The applications do not need to be recompiled or re-saved. [2756226]

### Standard Additions

- In Standard Additions 1.7, the description of the `choose application` command stated that it allows users to choose a running application. On Mac OS X it allows users to choose any application, running or not. [2785713]
- In Standard Additions 1.7, using the `mount volume` command with an AppleTalk server name in the `on server` parameter did not work. This has been fixed. [2656820]
- In Standard Additions 1.7, if the `path to` command was requested to return the path of a folder that did not exist (such as `scripting additions folder from local domain`), it would fail. In previous versions, and now in 1.8 as well, it creates the folder and returns the path to the created folder. Note that if the user does not have authorization to create the folder (e.g. is not an administrator and requests a nonexistent folder in the System domain), the operation will still fail. [2751119]
- In Standard Additions 1.7, the `store script` command would crash if no `in` parameter was supplied. This has been fixed. [2782590]
- Previous versions of Standard Additions improperly list the parameters for the `store script` command. Both the direct parameter (the script) and the `in` parameter (the file) are optional. If the script is omitted, an empty script is stored in the file; if the `in` parameter is omitted, the user is prompted for a file name. [2784735]

### Command Line Tools

- The `osacompile` tool would fail if provided with a `-i` parameter specifying a nonexistent file. It is now more lenient and issues a warning, but continues compilation. [2796415]

## Mac OS X v10.1.2 - AppleScript 1.8.1

AppleScript 1.8.1 is included with Mac OS X version 10.1.2 and requires Mac OS X version 10.1 or later.

### Bug Fixes

### AppleScript

- In Mac OS X version 10.1 and 10.1.1, installing certain multimedia software (such as Roxio Toast or QuickTime 5) would cause all AppleScript applets to crash when launching. This has been fixed. [2785067]

# April 2002 Developer Tools Update - AppleScript 1.8.2

AppleScript 1.8.2 for Mac OS X is included with the April 2002 Developer Tools Update and requires Mac OS X version 10.1 or later.

## New Features and Enhancements

### AppleScript

- The `do shell script` command and the scriptable Terminal allow AppleScript scripts to execute strings of text as UNIX shell scripts. The UNIX shell interprets certain characters specially (including single-quote, double-quote, dollar sign, slash, and backslash). In order to pass text containing these characters to the shell, they need to be quoted to let the shell know to process them literally, rather than apply its special interpretation. The property `quoted form` has been added to the `string` and `Unicode text` classes. The value of this property is the text enclosed in single quotes, with any single quotes in the string (such as apostrophes in filenames) handled by concatenation. In general, all POSIX file paths and most text parameters provided to shell commands should be quoted. A common use of this is in commands like:

### Standard Additions

- The dialogs for `display dialog` and `choose from list` are now moveable modal dialogs and can be moved by the user. [2845800]
- The `do shell script` command has a new `altering line endings` optional parameter. When this optional parameter is omitted or true, the result of `do shell script` will have Mac-style line endings (CR, ASCII character 13) and always be terminated by a CR; with this optional parameter set to false (i.e. `without altering line endings`), the results will have UNIX-style line endings (LF, ASCII character 10) and may or may not have a terminating LF. The default results are compatible with AppleScript 1.8.1, while the `without altering line endings` form returns the unaltered output of the UNIX script. [2850410]

## Bug Fixes

### AppleScript

- In AppleScript 1.8.1 and earlier, using `text item delimiters` to extract text items from Unicode text can cause the application to crash. This has been fixed. [2848981]
- In AppleScript 1.6 through 1.8.1, a script comparing two Unicode strings could occasionally crash the application running the script. This has been fixed. [2885328]

- In AppleScript 1.8.1 and earlier, certain applications executing scripts can become unstable when coercing identifiers to strings in tight loops. This includes FaceSpan and AppleScript Studio applications. This has been fixed. [2858324]
- In AppleScript 1.8 and 1.8.1, the Use Startup Screen menu item in script applications does not work. This has been fixed. [2844978]
- In AppleScript 1.8 and 1.8.1 for Mac OS X, getting the `POSIX path` property of a directory would omit the terminating colon. This has been fixed. [2851963]
- In AppleScript 1.1 and later, loading any script would cause a small memory leak. This has been fixed. [2852866]
- With AppleScript 1.8 and 1.8.1, some applications that create and/or execute scripts (such as Script Debugger) will occasionally crash when trying to get certain information about a script. This has been fixed. [2811897]
- In AppleScript 1.8 and 1.8.1, some applications that execute scripts that target themselves will crash unexpectedly. This has been fixed. [2839367]
- On some rare occasions in AppleScript 1.8 (often during recording), the name of an application in a `tell` statement is changed to `AEServer`. This has been fixed. [2822232]
- In all versions of AppleScript 1.5.5 and earlier, using the `some` operator on a `record` to select a type that does not exist in the record (e.g. `some string of {a:2, b:3, c:{5, 7, 11}}`) would return the entire record, rather than an error. This was fixed in AppleScript 1.6 for Mac OS 9, but has not been fixed in Mac OS X until now. [2842496]
- AppleScript 1.7 could not open scripts saved on Mac OS 9 by any version of AppleScript if those scripts contained references to applications on remote machines. AppleScript 1.8 can open those scripts, but omits the first two characters of the host name. AppleScript 1.8.2 and later will correctly read and display an application reference created on Mac OS 9. [2781523]

## Standard Additions

- Many commands in Standard Additions 1.7 through 1.8.1 adopted a new style of processing input parameters that refer to files. This was found to fail when used inside application `tell` blocks with application objects (such as Finder `file` objects). Those commands have been modified to work harder to understand file references in applications. [2840037]
- The `display dialog` command now recognizes a user-defined button with the button name “Cancel” as the cancel button when the computer is using a non-English system. [2765410]
- In Standard Additions 1.7 through 1.8.1, using lengthy button titles in the `display dialog` command could cause the prompt text to be truncated. This has been fixed. [2844857]
- The performance of the `do shell script` command has been significantly improved when executing lengthy shell scripts. [2853370]

- Prior to Standard Additions 1.7, the `list folder` command had a special case to accept a path string as its direct parameter (though it is documented to accept an `alias`). In Standard Additions 1.7, this special case was omitted, causing scripts that used the form `list folder "Macintosh HD:"` to fail. This special case has been restored. [2850766]
- In Standard Additions 1.7 through 1.8.1, the `open for access` command would not set the file type and creator of newly-created files. This has been fixed. [2836047]
- The `run script` command in Standard Additions 1.5 through 1.8.1 did not handle the `in` parameter correctly when choosing an OSA scripting component. It would in fact select the first OSA scripting component that was not the one named. This has been fixed. [2889646]
- In Standard Additions 1.7, the `write` command signals an error -50 (parameter error) when a script attempts to write an empty string to a file. This has been fixed. [2823197]

### Command Line Tools

- In AppleScript 1.7 through 1.8.1, the `osascript` tool truncates output text if it contains a null character. This has been fixed. [2827241]

## AppleScript 1.8.3

AppleScript 1.8.3 for Mac OS X is available as an independent Software Update and requires Mac OS X version 10.1 or later.

### Bug Fixes

#### AppleScript

- In AppleScript 1.8.2 and earlier, operations using AppleScript's `text item delimiters` fail if the delimiters are set to Unicode string values. This has been fixed. [2895572]
- In AppleScript 1.8.2 and earlier, converting a list of values to a Unicode string will ignore any text encoding information in those values, so the resulting Unicode string will not be correct. This has been fixed. [2895579]

#### Standard Additions

- In Standard Additions 1.6 through 1.8.2, the `as` parameter for the `read` and `write` commands was not handled correctly (especially if the value of the parameter was `short` or a list). This has been fixed. [2751784, 2892057]

# 10.0 Changes

This article describes changes to AppleScript and related tools in Mac OS X v10.0.

## Mac OS X v10.0 - AppleScript 1.6

AppleScript 1.6 is included with Mac OS X version 10.0 and requires Mac OS X version 10.0 or later.

### Developer Notes

- A new call, `OSACopyScript()`, will create an independent clone of an existing OSAID. `OSACopyID()`, by comparison, creates a new reference to an existing object.
- Event handlers have always been able to accept an `as` parameter, but have had no way to indicate to AppleScript that they handled the coercion. This creates problems when the handler actually returns a list of the requested type -- AppleScript thinks the list needs to be coerced, and tends to either fail or do the wrong thing. The workaround has been to define a "type list" type and a coercion handler from that to a plain list that simply duplicates the descriptor. The Finder's `alias list` type, for example, is implemented this way. There is now a better way: if you take an `as` parameter, and you do not want AppleScript to try to coerce your result, put a Boolean true value into the `keyAppHandledCoercion` parameter of the result. AppleScript will sense this parameter and leave the result unchanged.
- Mac OS X does not support scripting additions (OSAXen) built for Mac OS 9. Because only applications can safely link to CarbonLib on Mac OS 9, scripting additions on 9 must link to InterfaceLib, which is not supported on Mac OS X. Scripting additions for Mac OS X come in two flavors: CFM and Mach-O. For CFM, follow the directions in Tech Note TN1164 to build it as a shared library, but link to CarbonLib instead of InterfaceLib.
- Mach-O scripting additions are packaged as bundles. Using Project Builder, create a Bundle target, and make sure that it has either a `CFBundleSignature` of `'osax'` or a name ending with `.osax`. The code layout is essentially the same as a CFM scripting addition, but instead of using the CFM initialization and termination routines plus the `gAdditionReferenceCount` global, AppleScript looks for three entry points: `SAInitialize`, `SATerminate`, and `SAIsBusy`, with the following prototypes:
- `additionBundle` is a reference to your own addition; you may retain it for future use or ignore it and rely on your own bundle identifier. `SAIsBusy` is equivalent to testing for `gAdditionReferenceCount` not equal to zero.

- It is possible to build a bundled scripting addition that contains both 9 and X versions, thus giving you a single file distribution, but you need to have AppleScript 1.6 and CarbonLib 1.3 installed on Mac OS 9.1 for it to work with Mac OS 9.
- An important difference from Mac OS 9 is that in Mac OS X, scripting additions are loaded separately into each application that connects to AppleScript. This means that (1) there may be multiple instances of your addition running on the system, so be careful about shared resource management; and (2) you cannot assume that any random application has access to your addition, so sending raw AppleEvents to an application will not necessarily work. (e.g., sending `syso/beep` to the Finder will return not handled.) AppleScript has a workaround built into it so that scripts will still work, but applications that use AppleEvents directly may have trouble.

## New Features and Enhancements

### AppleScript

- The `Unicode text` data type has been growing in functionality since AppleScript 1.3 and is now comparable in functionality with the `string` data type. Not only can Unicode text strings be received from and sent to applications, they display properly in the Results and Event Log windows of the Script Editor (and third-party editors); they can be accessed with the `character`, `word`, `paragraph`, and `text item` element classes and the `first`, `last`, `middle`, `some`, and `every` access forms; they can be operated on with the `&` (concatenation) operator; and they can be compared with the `less than`, `greater than`, `less than or equal to`, `greater than or equal to`, `equals`, `contains`, `is contained by`, `begins with`, and `ends with` operators. For `text item` elements of a Unicode string, the `text item delimiters` can be either string or Unicode text values. When comparing Unicode strings, the `considering` and `ignoring` constructs are supported for `case`, `diacriticals` and `punctuation`.
- AppleScript 1.6 supports many changes in general system operation introduced in Mac OS X. It can use applications and scripting additions that are stored as bundles or packages for Mac OS X. It can load scripts and application terminologies from files that do not have traditional Mac OS resource forks. It accommodates the fact that many Mac OS X applications actually have a `.app` filename extension that is hidden from the user.

### Command Line Tools

- There are two new command line tools, `osascript` and `osacompile`, that let you execute and compile scripts from a Unix shell. For full details, see their respective man pages.

# Bug Fixes

## AppleScript

- AppleScript 1.6 fixes a long-standing issue in AppleScript in which scripts may claim to run out of memory, even when plenty of memory is available.
- AppleScript 1.4.3 and 1.5.5 had rare issues with applets losing track of global variables; these issues have been fixed in AppleScript 1.6.
- AppleScript 1.6 returns more correct results from certain unusual operations (`item 0` of a list, `set text item delimiters to {}`, `strings of` a string, `strings of` a list that doesn't contain strings) where previous versions of AppleScript would return unhelpful results or fail to signal appropriate errors.

## Standard Additions

- AppleScript 1.6 has improvements to the `random number` command, to make random numbers more evenly distributed and less predictable.
- The `read` and `write` commands can now work with files greater than 2 GB in size. In addition, all of the issues introduced in the Read/Write commands in AppleScript 1.5 and 1.5.5 have been fixed, and several long-standing issues with reading and writing lists and records have been fixed.
- Copyright © 2001-2007 Apple Inc.

# Document Revision History

This table describes the changes to *AppleScript Release Notes*.

| Date | Notes |
| --- | --- |
| 2013-10-22 | Updated for OS X Mavericks. |
| 2013-06-25 | Updated with changes for OS X Lion v10.7 and OS X Mountain Lion v10.8. |
| 2009-05-29 | Unifies older separate documents, adds changes for Mac OS X v10.6. |

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010