OS X El Capitan

Core Technologies Overview
September 2015

Contents

Introduction

With more than 89 million users—consumers, scientists, animators, developers, and system administrators—OS X is the most widely used UNIX® desktop operating system. In addition, OS X is the only UNIX environment that natively runs Microsoft Office, Adobe Photoshop, and thousands of other consumer applications—all side by side with traditional command-line UNIX applications. Tight integration with hardware—from the sleek MacBook to the powerful Mac Pro—makes OS X the platform of choice for an emerging generation of power users.

This document explores the powerful industry standards and breakthrough innovations in the core technologies that power Apple's industry-leading user experiences. We walk you through the entire software stack, from firmware and kernel to iCloud and developer tools, to help you understand the many things OS X does for you every time you use your Mac.

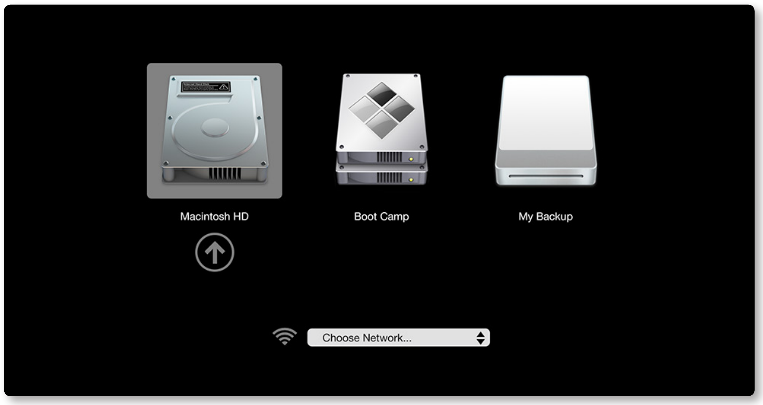
System Startup

BootROM

When you turn on the power to a Mac, it activates the BootROM firmware. BootROM, which is part of the computer's hardware, has two primary responsibilities: It initializes system hardware and it selects an operating system to run. Two BootROM components carry out these functions:

- Power-On Self Test (POST) initializes some hardware interfaces and verifies that sufficient memory is available and in a good state.
- Extensible Firmware Interface (EFI) does basic hardware initialization and selects which operating system to use.

If multiple OS installations are available, BootROM chooses the one that was last selected by the Startup Disk System Preference. The user can override this choice by holding down the Option key while the computer starts up, which causes EFI to display a screen for choosing the startup volume.



EFI boot picker screen.

EFI

EFI—a standard created by Intel—defines the interface between an operating system and platform firmware. It supersedes the legacy Basic Input Output System (BIOS) and OpenFirmware architectures.

Once BootROM is finished and an OS X partition has been selected, control passes to the boot.efi boot loader, which runs inside EFI. The principal job of this boot loader is to load the kernel environment. As it does this, the boot loader draws the "booting" image on the screen.

If full-disk encryption is enabled, the boot loader draws the login UI and prompts for the user's password, which the system needs so it can access the encrypted disk and boot from it. Otherwise, loginwindow draws the login UI.

Kernel

The OS X kernel is based on FreeBSD and Mach 3.0 and features an extensible architecture based on well-defined kernel programming interfaces (KPIs).

OS X was the first operating system to ship as a single install that could boot into either a 32-bit or 64-bit kernel, either of which could run 32-bit and 64-bit applications at full native performance. OS X now exclusively uses a 64-bit kernel, but it continues to run both 32-bit and 64-bit applications.

With its 64-bit kernel, OS X is able to address large amounts of physical RAM. OS X El Capitan has been tested to support up to 128GB of physical RAM on qualified Mac computers.

Drivers

Drivers in OS X are provided by I/O Kit, a collection of system frameworks, libraries, tools, and other resources for creating device drivers. I/O Kit is based on an object-oriented programming model implemented in a restricted form of C++ that omits features unsuitable for use within a multithreaded kernel.

By modeling the hardware connected to an OS X system and abstracting common functionality for devices in particular categories, the I/O Kit streamlines the process of device-driver development. I/O Kit helps device manufacturers rapidly create drivers that run safely in a multiprocessing, preemptive, hot-pluggable, power-managed environment.

To do this, I/O Kit provides the following:

- An object-oriented framework implementing common behavior shared among all drivers and types (families) of drivers
- Many families of drivers for developers to build upon
- Threading, communication, and data management primitives for dealing with issues related to multiprocessing, task control, and I/O transfers
- A robust, efficient match-and-load mechanism that scales well to all bus types
- The I/O Registry, a database that tracks instantiated objects (such as driver instances) and provides information about them
- The I/O Catalog, a database of all I/O Kit classes available on a system
- A set of device interfaces—plug-in mechanisms that allow applications and other software outside the kernel to communicate with drivers
- Excellent overall performance
- Support for arbitrarily complex layering of client and provider objects

Initialization

There are two phases to system initialization:

- *Boot* refers to loading the bootstrap loader and kernel.
- *Root* means mounting a partition as the root, or top-level, file system.

Once the kernel and all drivers necessary for booting are loaded, the boot loader starts the kernel's initialization procedure. At this point, enough drivers are loaded for the kernel to find the root device—the disk or network service where the rest of the operating system resides.

The kernel initializes the Mach and BSD data structures and then initializes the I/O Kit. The I/O Kit links the loaded drivers into the kernel, using the device tree to determine which drivers to link. Once the kernel finds the root device, it roots BSD off of it.

Address Space Layout Randomization (ASLR)

Many malware exploits rely on fixed locations for well-known system functions. To mitigate that risk, OS X randomly relocates the kernel, kexts, and system frameworks at system boot. This protection is available to both 32-bit and 64-bit processes.

Compressed Memory

Compressed Memory keeps your Mac fast and responsive by freeing up memory when you need it most. When your system's memory begins to fill up, Compressed Memory automatically compresses the least recently used items in memory, compacting them to about half their original size. When these items are needed again, they can be instantly uncompressed.

Compressed Memory improves total system bandwidth and responsiveness, allowing your Mac to handle large amounts of data more efficiently. Through use of the dictionary-based WKdm algorithm, compression and decompression are faster than reading and writing to disk. If your Mac needs to swap files on disk, compressed objects are stored in full-size segments, which improves read/write efficiency and reduces wear and tear on SSD and flash drives. The advantages of Compressed Memory include the following:

- **Shrinks memory usage.** Compressed Memory reduces the size of items in memory that haven't been used recently by more than 50 percent, freeing memory for the applications you are currently using.
- **Improves power efficiency.** Compressed Memory reduces the need to read and write virtual memory swap files on disk, improving the power efficiency of your Mac.
- **Minimizes CPU usage.** Compressed Memory is incredibly fast, compressing or decompressing a page of memory in just a few millionths of a second.
- **Is multicore aware.** Unlike traditional virtual memory, Compressed Memory can run in parallel on multiple CPU cores, achieving lightning-fast performance for both reclaiming unused memory and accessing seldom-used objects in memory.

Power Efficiency

The power technologies in OS X El Capitan were built with the capabilities of modern processors and the demands of modern apps in mind. The new power technologies work together to achieve substantial power savings, while maintaining—and in some cases even improving—the responsiveness and performance of your Mac.

These technologies are rooted in a few key principles:

- Just work for existing apps. No changes to applications should be needed, though small changes may facilitate additional power savings.
- Keep as many processor cores idle as possible given the demand for CPU.
- When on battery power, only do work that the user is requesting or that is absolutely essential.

App Nap

App Nap puts applications that you're not using into a special low-power state that regulates their CPU usage as well as network and disk I/O. App Nap can be automatically triggered if an app's windows are not visible and the app is not playing audio, though developers can explicitly make an app ineligible for App Nap by using the existing IOKit IOPMAssertion API (used today in OS X to prevent the system from sleeping while an app is busy). App Nap triggers a number of power-saving measures, including:

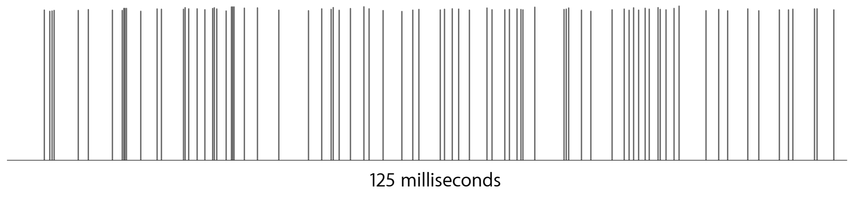
- **Timer throttling.** Reduces the frequency with which an app's timers are fired. This can mean significant improvements in CPU idle time when running applications that frequently check for data.
- **I/O throttling.** Assigns the lowest priority to disk or network activity associated with a napping app. The rate at which an application can read or write data from a device is significantly reduced. In addition, I/O throttling reduces the chances that a background process will interfere with the I/O activity of an app that you are actively using.
- **Priority reduction.** Reduces the UNIX process priority of an app so that it receives a smaller share of available processor time.

Timer Coalescing

Timer Coalescing minimizes the amount of system maintenance and background work that is performed while your Mac is running on battery power. Some tasks are set to run on battery power only after a specified amount of time has passed (for example, Software Update checks every seven days, and can defer checking by up to one day if the user is on battery power), while other tasks may be configured to never run on battery power (such as background downloads of software updates).

If allowing the CPU to spend as much time as possible idling is good for power, it stands to reason that frequently waking up a processor can hurt battery life. Typically, there are numerous applications and background processes that use timers with different intervals to schedule routine work.

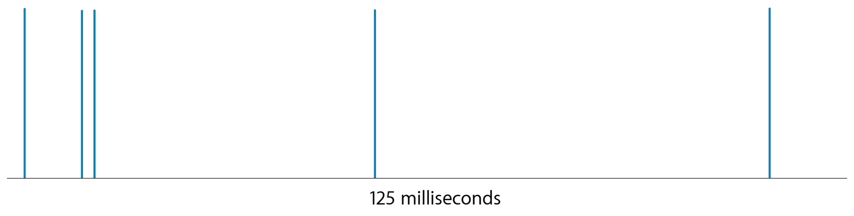
Typical timers



Typical timer execution sample showing 125 milliseconds on a system with no user interaction. Frequent execution of timers from various processes can prevent the processor from idling.

The challenge becomes: How can the system do all of the required work while maximizing the amount of time the processor spends at idle? The solution is Timer Coalescing, which shifts the execution of timers by a small amount so that multiple applications' timers are executed at the same time. This can dramatically increase the amount of time that the processor spends idling.

OS X timers



A sample of the same scenario on OS X: 125 milliseconds of timer execution with no user activity. Executing timers at the same time reduces power use by maximizing processor idle time.

Task-Level Scheduling

OS X provides task-level scheduling and awareness to ensure the highest-priority tasks are appropriately dispatched to available CPU cores. Task-level scheduling ensures that applications and daemons that utilize a large number of threads do not unfairly monopolize CPU resources at the expense of processes that use threads more sparingly. Scheduling based on task level makes your Mac more responsive and results in a more fluid user experience.

USB Power Management

Traditionally, USB device power management is done manually, with a function driver making explicit suspend or resume calls on a USB device service. This is error-prone and results in overly complex driver code to manage power state.

USB in OS X El Capitan has been completely implemented to benefit from the same USB architecture and aggressive power management policy adopted by iOS. Each endpoint in a device can be given an idle timeout in milliseconds, and an endpoint is considered idle if an active transfer hasn't completed within that timeout. The device has an idle state as well, which is a logical AND of the idle state for the endpoints associated with the device. Only when the device idle timeout expires does the OS pause I/O on the bus and electrically suspend the device at the port.

The idling system is an important advance because drivers don't need to actively participate in power state transitions.

Disk Layout

Partition Scheme

Disk drives are divided into logical partitions, which Apple traditionally calls *volumes*. Modern Mac systems use the GUID partition table (GPT) partitioning scheme introduced by Intel as part of EFI. The partitioning scheme is formally defined by:

- Section 11.2.2 of "Extensible Firmware Interface Specification," version 1.1, available from Intel
- Chapter 5, "GUID Partition Table (GPT) Format," of the "Unified Extensible Firmware Interface Specification," version 2.0, available from the Unified EFI Forum

By default, the internal hard disk is formatted as GPT. You can explore and modify GPT disks using the `gpt` command-line tool derived from FreeBSD. You can also use Apple's GPT-aware `diskutil` utility, which provides more human-readable output.

Helper partitions
Typically, a single partition is "blessed" as the active boot volume via the `bless` command-line tool, though you can also bless specific folders or files. This partition is usually also the root volume.

However, sometimes the boot partition is not the root, such as when the root partition is encrypted using full-disk encryption or located on a device that requires additional drivers (such as a RAID array). In that case, a hidden helper partition stores the files needed to boot, such as the kernel cache. The last three known good helper partitions are maintained in case one becomes corrupted.

Recovery partitions
Your Mac includes a Recovery HD partition that provides the tools you need to do the following:

- Reinstall OS X
- Repair a hard drive
- Restore from a Time Machine backup
- Launch Safari to view documentation and search the Internet
- Create Recovery HD partitions on external drives

To boot from the Recovery HD partition, restart your Mac while holding down the Command key and the R key (Command-R). Keep holding them until the Apple icon appears, indicating that your Mac is starting up. After the Recovery HD finishes starting up, you should see a desktop with an OS X menu bar and an OS X Utilities application window.

If your Recovery HD is corrupt or unavailable and you have a network connection, your Mac will automatically use OS X Internet Recovery to download and boot directly from Apple's servers, using a pristine Recovery HD image that provides all of the same functionality.

Core Storage

Layered between the whole-disk partition scheme and the file system used for a specific partition is a new logical volume format known as Core Storage. Core Storage makes it easy to dynamically allocate partitions while providing full compatibility with existing file systems. In particular, Core Storage allows in-place transformations such as back-grounding the full-disk encryption used by File Vault and intelligent block-level data migration used by Fusion Drive.

In addition, Core Storage provides:

- **Copy-on-write B-tree catalogs.** This feature helps ensure file system consistency and durability. Even if a read or write transaction on your drive is halted while in progress, the file system remains in the most recent consistent state before the beginning of the transaction. Once a transaction is committed to disk, all file system changes are permanent even if the operating system fails right after the commit.
- **Ditto blocks.** These provide redundant copies of volume metadata (MLV blocks). Ditto blocks help prevent loss of metadata for the whole logical volume, and are used when converting volumes from one format to another (for example, enabling FileVault), when resizing, and when performing file system repair operations.

File Systems

A stackable virtual file system layer allows OS X to dynamically mount, read, and write to numerous local file systems—including HFS+, UFS, ISO 9660 CD-ROM formats, UDF for DVDs, FAT32, and NTFS (read only).

Mac OS Extended (HFS+)

OS X partitions are usually formatted using some variant of the HFS+ file system. HFS+ supports 64-bit disk space addressing, 32-bit file allocation blocks, journaling, automatic compression of files, fast B-tree lookups, robust alias support, and rich metadata—including fine-grained access controls and extended attributes.

The Mac OS Extended file system functionality includes:

- **Long filenames and international support.** HFS+ allows more descriptive filenames, with support for up to 255 characters and Unicode text encoding for international and mixed-script filenames.
- **Case sensitivity.** OS X offers an optional case-sensitive file system format for HFS+, allowing you to host files for use by UNIX applications that require case sensitivity.
- **Large volume support.** To accommodate massive databases, image archives, and video volumes larger than 16TB, OS X can configure HFS+ volumes to use block sizes larger than 4K.
- **File system journaling.** A robust journaling feature protects the integrity of the file system in the event of an unplanned shutdown or power failure. With journaling, the operating system automatically tracks file system operations and maintains a continuous record of these transactions in a separate file, called a *journal*. After an unexpected shutdown, the operating system can use the journal to return the file system to a known state—eliminating the need to perform a consistency check on the entire file system during startup. With a journaled file system, bringing a volume back online takes just seconds, regardless of the number of files or the size of the volume.

Process Control

Launchd

The kernel invokes `launchd` as the first process to run and then bootstraps the rest of the system. It replaces the complex web of `init`, `cron`, `xinetd`, and `/etc/rc` used to launch and manage processes on traditional UNIX systems. `launchd` is available as open source under the Apache license.

File-based configuration
Each job managed by launchd has its own configuration file in a standard `launchd.plist(5)` file format, which specifies the working directory, environment variables, timeout, Bonjour registration, etc. These plists can be installed independently in the standard OS X library domains (for example, /Network/Library, /System/Library, /Library, or ~/Library), avoiding the need to edit systemwide configuration scripts. Jobs and plists can also be manually managed by the `launchctl(1)` command-line tool.

Launch on demand
`launchd` prefers for processes to run only when needed instead of blocking or polling continuously in the background. These launch-on-demand semantics avoid wasting CPU and memory resources, and thus prolong battery life.

For example, jobs can be started based on the following:

- If the network goes up or down
- When a file path exists (for example, for a printer queue)
- When a device or file system is mounted

Smart scheduling
Like traditional UNIX `cron` jobs, `launchd` jobs can be scheduled for specific calendar dates with the `StartCalendarInterval` key, as well as at generic intervals via the `StartInterval` key. Unlike `cron`—which skips job invocations when the computer is asleep—`launchd` starts the job the next time the computer wakes up. If the computer sleeps through multiple intervals, those events will be coalesced into a single trigger.

User agents
`launchd` defines a daemon as a systemwide service where one instance serves multiple clients. Conversely, an agent runs once for each user. Daemons should not attempt to display UI or interact directly with a user's login session; any and all work that involves interacting with a user should be done through agents.

Every `launchd` agent is associated with a Session Type, which determines where it runs and what it can do, as shown in the following table:

Name	Session type	Notes
GUI	Aqua	Has access to all GUI services; much like a login item
Non-GUI	StandardIO	Runs only in non-GUI login sessions (for example, SSH login sessions)
Per-user	Background	Runs in a context that's the parent of all contexts for a given user
Pre-login	Loginwindow	Runs in the loginwindow context

Loginwindow

As the final part of system initialization, `launchd` launches `loginwindow`. The `loginwindow` program controls several aspects of user sessions and coordinates the display of the login window and the authentication of users.

If a password is set, OS X requires users to authenticate before they can access the system. The `loginwindow` program manages both the *visual* portion of the login process (as manifested by the window where users enter name and password information) and the *security* portion (which handles user authentication).

Once a user has been authenticated, `loginwindow` begins setting up the user environment. As part of this process, it performs the following tasks:

- Secures the login session from unauthorized remote access
- Records the login in the system's `utmp` and `utmpx` databases
- Sets the owner and permissions for the console terminal
- Resets the user's preferences to include global system defaults
- Configures the mouse, keyboard, and system sound according to user preferences
- Sets the user's group permissions (`gid`)
- Retrieves the user record from Directory Services and applies that information to the session
- Loads the user's computing environment (including preferences, environment variables, device and file permissions, keychain access, and so on)
- Launches the Dock, Finder, and SystemUIServer
- Launches the login items for the user

Once the user session is up and running, `loginwindow` monitors the session and user applications in the following ways:

- Manages the logout, restart, and shutdown procedures
- Manages Force Quit by monitoring the currently active applications and responding to user requests to force quit applications and relaunch the Finder. (Users open this window from the Apple menu or by pressing Command-Option-Escape.)
- Arranges for any output written to the standard error console to be logged using the Apple System Logger (ASL) API. (See "Log Messages Using the ASL API" in the OS X Developer Library.)

Grand Central Dispatch

Grand Central Dispatch (GCD) supports concurrent computing via an easy-to-use programming model built on highly efficient system services. This radically simplifies the code needed for parallel and asynchronous processing across multiple cores.

GCD is built around three core pieces of functionality:

- Blocks, a concise syntax for *describing* work to be done
- Queues, an efficient mechanism for *collecting* work to be done
- Thread pools, an optimal service for *distributing* work to be done

These help your Mac make better use of all available CPU cores while improving responsiveness by preventing the main thread from blocking.

Systemwide optimization

The central insight of GCD is shifting the responsibility for managing threads and their execution from applications to the operating system. As a result, programmers can write less code to deal with concurrent operations in their applications, and the system can perform more efficiently on both single-processor and multiprocessor machines. Without a pervasive approach such as GCD, even the best-written application cannot deliver optimal performance across diverse environments because it lacks insight into everything else happening on the system.

Blocks

Block objects are extensions to C, Objective-C, and C++ that make it easy for programmers to encapsulate inline code and data for later use. Here's what a block object looks like:

```
int scale = 4;
int (^Multiply)(int) = ^(int num) {
     return scale * num;
};
int result = Multiply(7); // result is 28
```

These types of "closures"—effectively a function pointer plus its invocation context—are common in dynamically typed interpreted languages, but they were never before widely available to C programmers. Apple has published both the Blocks Language Specification and its implementation as open source under the MIT license and added blocks support to both GCC and Clang/LLVM.

Queues

GCD dispatch queues are a powerful tool for performing tasks safely and efficiently on multiple CPUs. Dispatch queues atomically add blocks of code that can execute either asynchronously or synchronously. Serial queues enable mutually exclusive access to shared data or other resources without the overhead or fragility of locks. Concurrent queues can execute tasks across multiple distinct threads, based on the number of currently available CPUs.

Thread pools

The root level of GCD is a set of global concurrent queues for every UNIX process, each of which is associated with a pool of threads. GCD dequeues blocks and private queues from the global queues on a first-in/first-out (FIFO) basis as long as there are available threads in the thread pool, providing an easy way to achieve concurrency.

If there is more work than available threads, GCD asks the kernel for more threads, which are given if there are idle logical processors. Conversely, GCD eventually retires threads from the pool if they are unused or the system is under excessive load. This all happens as a side effect of queuing and completing work so that GCD itself doesn't require a separate thread. This approach provides optimal thread allocation and CPU utilization across a wide range of loads.

Event sources

In addition to scheduling blocks directly, GCD makes it easy to run a block in response to various system events, such as a timer, signal, I/O, or process state change. When the source fires, GCD will schedule the handler block on the specific queue if it is not currently running, or—more importantly—coalesce pending events if it is running.

This provides excellent responsiveness without the expense of either polling or binding a thread to the event source. Plus, since the handler is never run more than once at a time, the block doesn't even need to be reentrant; only one thread will attempt to read or write any local variables.

OpenCL integration

Deeply integrated into OS X, OpenCL accelerates applications by tapping into the parallel computing power of modern GPUs and multicore CPUs. Using a C99-based language coupled with a flexible API for managing data-parallel workloads, OpenCL opens up a new range of computationally intensive algorithms for use in your application.

You use OpenCL to transform your apps' most performance-intensive routines into computational "kernels." Each kernel is runtime compiled by OpenCL and efficiently scheduled for execution, automatically taking best advantage of the parallel processing capabilities of the targeted GPU or CPU. On OS X, OpenCL kernels can also be run as blocks using a special GCD queue.

Sandboxing

Sandboxes ensure that processes are only allowed to perform a specific set of expected operations. For example, a web browser regularly needs to read from the network, but shouldn't write to the user's home folder without explicit permission. Conversely, a disk usage monitor may be allowed to read directories and delete files, but not talk to the network.

These restrictions limit the damage a program could potentially cause if it became exploited by an attacker. By using attack *mitigation*, sandboxes complement the usual security focus on attack *prevention*. For this reason, we recommend that sandboxes be used with all applications, and we require their use for apps distributed via the Mac App Store.

Mandatory access controls

Sandboxes are built on low-level access control mechanisms enforced in the kernel by the `kauth` subsystem. This was introduced in OS X 10.4 based on work originating in `TrustedBSD`. `kauth` identifies a valid actor (typically a process) by its credentials. It then asks one or more listeners to indicate whether that actor can perform a given action within a specified scope (authorization domain). Only the initial (default) listener can allow a request; subsequent listeners can only deny or defer. If all listeners defer, `kauth` denies the request.

Entitlements

Sandboxes collect these low-level actions into specific entitlements that an application must explicitly request by adding the appropriate key to a property list file in its application bundle. Entitlements can control access to:

- The entire file system
- Specific folders
- Networking
- iCloud
- Hardware (for example, the built-in camera or microphone)
- Personal information (for example, contacts)

In addition, entitlements control whether processes inherit their parents' permissions and can grant temporary exceptions for sending and receiving events or reading and writing files.

User intent
While it may seem that virtually all applications would need to request broad entitlements to read and write files, that isn't the case. OS X tracks user-initiated actions, such as dragging a file onto an application icon, and automatically opens a temporary hole in the sandbox, allowing the application to read just that one file. In particular, open and save panels run in a special-purpose PowerBox process that handles all user interaction. This allows applications to only request entitlements for actions they need to perform autonomously.

Code signing
Entitlements use code signing to ensure the privileges they specify only cover the code originally intended. Code signing uses public key cryptography to verify that the entity that created the entitlements (that is, the developer) is the same as the author of the executable in question, and that neither has been modified.

System Integrity Protection

System Integrity Protection is a policy introduced in OS X El Capitan to help prevent malicious software from performing harmful actions on your Mac. System Integrity Protection applies to every running process, including privileged code and code that runs out of the sandbox. The policy extends additional protections to components on disk and at runtime, only allowing system binaries to be modified by the system installer and software updates.

System Integrity Protection prevents software from modifying files and folders owned by the system, even if the software is run as the administrator or with root privileges. Examples of paths in the file system that are protected include the following:

- System folder
- /bin/
- /sbin/
- /usr/ (with the exception of /usr/local/)

In addition, System Integrity Protection prevents modification of applications and utilities that are installed with OS X, and prevents changes to your boot volume without your knowledge.

Gatekeeper

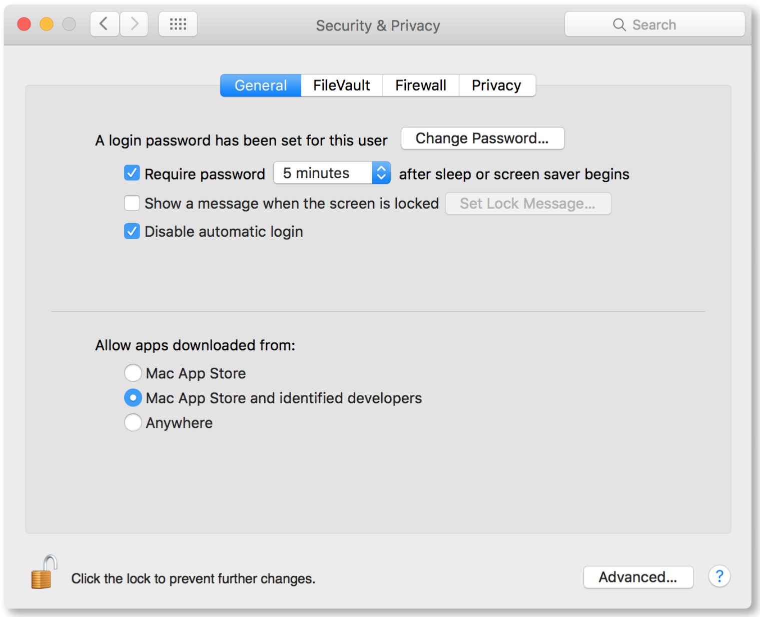
Gatekeeper is a feature in OS X that helps protect you from downloading and installing malicious software. Developers can sign their applications, plug-ins, and installer packages with a Developer ID certificate to let Gatekeeper verify that they come from identified developers.

Developer ID certificates
As part of the Mac Developer Program, Apple gives each developer a unique Developer ID for signing their apps. A developer's digital signature lets Gatekeeper verify that they have not distributed malware and that the app hasn't been tampered with.

User control
Choose the kinds of apps that are allowed to run on OS X from the following:

- Only apps from the Mac App Store, for maximum security
- Apps from the Mac App Store as well as apps that have a Developer ID
- Apps from anywhere

You can even temporarily override higher-protection settings by clicking on the app while holding down the Control key and then choosing Open from the contextual menu. This lets you install and run any app at any time. Gatekeeper ensures that you stay completely in control of your system.



A login password has been set for this user

✓ Require password 5 minutes after sleep or screen saver begins

☐ Show a message when the screen is locked

✓ Disable automatic login

Allow apps downloaded from:

○ Mac App Store

● Mac App Store and identified developers

○ Anywhere

Click the lock to prevent further changes.

You control which kinds of apps you want your system to trust.

XPC

XPC leverages `launchd`, GCD, and sandboxing to provide a lightweight mechanism for factoring an application into a family of coordinating processes. This factoring improves launch times, crash resistance, and security by allowing each process to focus on one specific task.

No configuration needed

XPC executables and `xpcservice.plist(5)` configuration files are all part of a single app bundle, so there is no need for an installer.

Launch-on-demand

XPC uses `launchd` to register and launch helper processes as they are needed.

Asynchronous communication

XPC uses GCD to send and receive messages asynchronously using blocks.

Privilege separation

XPC processes each have their own sandbox, allowing clean separation of responsibilities. For example, an application that organizes and edits photographs does not usually need network access. However, it can create an XPC helper with different entitlements, whose sole purpose is to upload photos to a photo-sharing website.

Vouchers

By default, XPC services are run in the most restricted environment possible—sandboxed with minimal file system access, network access, and so on. Vouchers allow daemons to have as few default privileges as possible, and allow them higher privileges only when absolutely needed.

Out-of-band data
In addition to primitive data types such as booleans, strings, arrays, and dictionaries, XPC can send messages containing out-of-band data such as file descriptors and IOSurface media objects.

Data Compression

The libcompression library provides an API for two styles of data compression:

- **Block compression.** Where all of the input data is compressed or decompressed by one call to the compression or decompression function.
- **Streaming compression.** Where the compression or decompression function is called repeatedly to compress or decompress data from a source buffer to a destination buffer. Between calls, processed data is moved out of the destination buffer and new data is loaded into the source buffer.

Four different compression algorithms are supported, giving developers a choice between speed, compression rations, and interoperability with non-Apple devices.

LZFSE
LZFSE is a new algorithm in OS X El Capitan matching the compression ratio of ZLIB level 5, but with much higher energy efficiency and speed (between 2x and 3x) for both encode and decode operations.

LZFSE is available only in iOS and OS X, so it can't be used when the compressed payload has to be shared to other platforms (Linux, Windows). LZFSE is an ideal replacement for ZLIB.

LZ4
LZ4 is an extremely high-performance compressor. LZ4 provides lossless data compression algorithm that is focused on fast compression and decompression speed.

LZMA
The Lempel–Ziv–Markov chain algorithm (LZMA) is used to perform lossless data compression. OS X implements encoder support at level 6 only. It uses a dictionary compression scheme and features a high compression ratio and a variable compression dictionary size.

ZLIB
Zlib is a lossless data compression library for use on a variety of operating systems. OS X implements a ZLIB encoder at level 5, which provides a good balance between compression speed and compression ratio. The ZLIB decoder supports decoding data compressed with any compression level.

Network Access

Ethernet

Mac systems were the first mass-market computers to ship with built-in Ethernet. OS X today supports everything from 10BASE-T to 10 Gigabit Ethernet. The Ethernet capabilities in OS X include the following:

Automatic link detection
Automatic link detection brings up the network stack whenever a cable is plugged in, and safely tears it down when the cable is removed.

Auto-MDIX
This feature reconfigures the connection depending on whether you are connecting to a router or another computer, so you no longer need special crossover cables.

Autonegotiation
Autonegotiaton discovers and uses the appropriate transmission parameters for a given connection, such as speed and duplex matching.

Channel bonding
Channel bonding supports the IEEE 802.3ad/802.1ax Link Aggregation Control Protocol for using multiple low-speed physical interfaces as a single high-speed logical interface.

Jumbo frames
This capability uses Ethernet frames of up to 9000 bytes with Gigabit Ethernet switches that allow them.

Multigigabit Ethernet
Multigigabit Ethernet technology enables existing Cat 5E and Cat 6 cables to achieve speeds greater than 1 Gbps. OS X includes Ethernet mediums for multiple speeds, including 1 Gbps, 2.5 Gbps, 5 Gbps, and 10 Gbps.

TCP segmentation offload
To reduce the work required of the CPU, TCP segmentation offload lets the Network Interface Card (NIC) handle splitting a large outgoing buffer into individual packets.

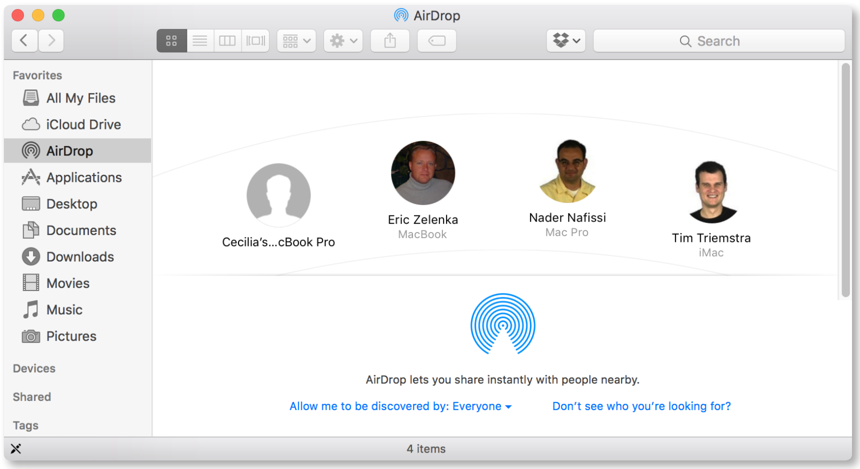
Wi-Fi

Apple brought Wi-Fi to the mass market with the original AirPort card and continues to provide cutting-edge wireless networking across our product lines.

Built into every Mac
Every Mac we ship—from the MacBook Air to the top-of-the-line Mac Pro—has 802.11 networking built right in.

AirDrop

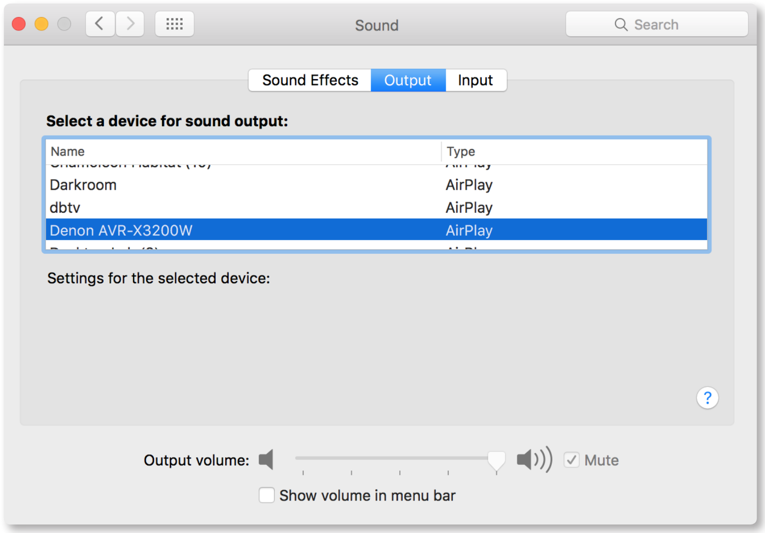
AirDrop makes it easy to safely share files wirelessly with nearby users, even if you aren't on the same network. AirDrop leverages the wireless hardware on newer Mac systems to find and connect to other computers on an ad hoc basis, even if they are already associated with different Wi-Fi networks.



Share files wirelessly with anyone around you using AirDrop.

AirPlay

AirPlay lets you stream music throughout your entire house—wirelessly. You can share audio or mirror your screen from your Mac to an Apple TV or any other AirPlay-enabled device.



Select a device for sound output:

Name	Type
Darkroom	AirPlay
dbtv	AirPlay
Denon AVR-X3200W	AirPlay

Settings for the selected device:

Output volume: [✓ Mute]

Show volume in menu bar

OS X treats AirPlay as just another audio output device.

Multihoming

OS X can have multiple network interfaces active at the same time, and dynamically determines the optimal one to use for a given connection. Here are some examples of where this is useful:

- Connecting to the Internet via Ethernet when you plug a Mac into the network, but seamlessly switching over to Wi-Fi when the network cable is unplugged
- Routing all corporate traffic through a VPN server for security, while accessing the public Internet directly to reduce latency
- Internet sharing, where one interface, such as Ethernet, is connected to the public Internet while the other, such as Wi-Fi, acts as a router for connecting your other devices

IPv6

OS X provides best-of-breed support for IPv6, the next-generation 128-bit Internet protocol.

Key features of IPv6 in OS X include:

- Full support for both stateful and stateless DHCPv6
- Happy Eyeballs algorithm (RFC 6555) for intelligently selecting between IPv6 and IPv4 addresses when both are available
- High-level APIs that resolve names directly so applications don't need to know whether they are using IPv4 or IPv6
- IPv6-enabled user applications (for example, Safari)

OS X has passed the U.S. Government v6 Testing Program and conformance tests to ensure IPv6 interoperability between computers, mobile devices, and network equipment used within the U.S. Government. USGv6 conformance tests were conducted and validated by the ISO/IEC 17025 accredited InterOperability Laboratory at the University of New Hampshire.

In addition, Apple has completed a USGv6 SDOC (Supplier's Declaration of Conformity), which will be on file with the U.S. Department of Interior.

IP over Thunderbolt

Internet Protocol over Thunderbolt allows you to connect two Mac systems via Thunderbolt cables for high-speed point-to-point data transfer.

Network File Systems

OS X includes a broad set of network file services—SMB, AFP, and NFS—for sharing files between Mac and PC.

SMB3

SMB3 is the default protocol for sharing files in OS X El Capitan. SMB3 helps protect against tampering and eavesdropping by encrypting and signing data "in flight."

- **Encryption.** SMB3 provides end-to-end encryption to protect data and secure communication on untrusted networks. SMB3 in El Capitan uses AES-CCM for encryption to ensure communications between client and server are private.
- **Signing.** To guard against tampering, SMB3 adds a signature to every packet transmitted over the wire. SMB3 uses AES-CMAC to validate the integrity of the signature, ensuring the packets have not been intercepted, changed, or replayed and that communication between hosts is authenticated and authorized.
- **Power.** Encryption and signing of SMB3 connections are fast and power efficient. Both AES-CCM for encryption and AES-CMAC for signing are dramatically accelerated on modern Intel CPUs with AES instruction support.
- **Authentication.** SMB supports Extended Authentication Security using Kerberos and NTLMv2.
- **Efficient.** SMB features Resource Compounding, allowing multiple requests to be sent in a single request. In addition, SMB can use large reads and writes to make better use of faster networks as well as large MTU support for blazing speeds on 10 Gigabit Ethernet. It aggressively caches file and folder properties and uses opportunistic locking to enable better caching of data. It's even more reliable, thanks to the ability to transparently reconnect to servers in the event of a temporary disconnect.
- **Transparent reconnect.** El Capitan supports Persistent Handles for transparent failover and reconnects to enterprise SMB3 file servers.
- **Compatible.** SMB is automatically used to share files between two Mac computers running OS X El Capitan, or when a Windows client running Windows 8 connects to your Mac. OS X El Capitan maintains support for AFP SMB2 and SMB network file-sharing protocols, automatically selecting the appropriate protocol as needed.

AFP

The Apple Filing Protocol (AFP) is the traditional network file service used on the Mac. Built-in AFP support provides connectivity with older Mac computers and Time Machine–based backup systems.

NFS

NFS v3 and v4 support in OS X allows for accessing UNIX and Linux desktop and server systems. With AutoFS, you can now specify automount paths for your entire organization using the same standard automounter maps supported by Linux and Solaris. For enhanced security, NFS can use Kerberos authentication as an alternative to UNIX UID-based authentication.

Access Control Lists

OS X supports both traditional UNIX file permissions and access control lists (ACLs), offering administrators an unprecedented level of control over file and folder permissions.

POSIX permissions

Standard UNIX file permissions, also known as Standard Portable Operating System Interface (POSIX) permissions, allow you to assign one access privilege to the file's owner, one to a group, and one to everyone on the network. Multiple users and multiple groups are not allowed, nor is ownership by a group. The traditional UNIX model supports three permissions—read, write, and execute.

Access control lists
For flexibility in complex computing environments, OS X includes file system access control lists. With file system ACLs, any file object can be assigned multiple users and groups, including groups within groups. Each file object can also be assigned both allow and deny permissions, as well as a granular set of permissions for administrative control, read, write, and delete operations. OS X supports a file permission inheritance model, ensuring that user permissions are inherited when files are moved, rewritten, or copied. ACLs may be set to allow a user to modify a file, but not to delete or even rename it.

App Transport Security

App Transport Security (ATS) helps enforce best practices in securing connections between applications and backend web services. App Transport Security sets default connection requirements for HTTS that conform to best practices for secure connections.

All connections that use NSURLConnection, CFURL, or NSURLSession APIs use App Transport Security by default in OS X. App Transport Security ensures network connections use TLS, are encrypted, use connection ciphers that provide forward secrecy, and use certificates with a SHA256 fingerprint with either a 2048-bit or greater RSA key, or a 256-bit or greater elliptic curve cryptography (ECC) key.

Directory Services

Directory services allow organizations to centralize information about users, groups, authentication, and computing resources. OS X is designed to be easily integrated with the most common directory servers including Microsoft Active Directory, Open Directory, and standard LDAP-based servers using the RFC 2307 schema standard.

Active Directory integration
OS X integrates with Microsoft Active Directory services. With Active Directory support, you need only keep a single user record to support OS X systems. Users can access their Mac using the same credentials they would use to access Windows PCs. Mac systems are fully integrated with Active Directory for password policies, single sign-on, and directory resources.

LDAPv3 support
To simplify integration into existing directory services, OS X supports the LDAP RFC 2307 standard. For greater flexibility, OS X supports client-side schema mappings so attributes in an LDAP-based directory can be easily mapped to settings in OS X.

Single sign-on
OS X uses Kerberos for single sign-on in networks using either Open Directory or Active Directory. Kerberos uses encryption keys to provide strong authentication for client/server applications, allowing authorized users to access secure network services—without exchanging passwords or requiring users to type in their passwords repeatedly.

Remote Access

Captive networks
Like iOS, OS X automatically detects the presence of a captive network and prompts for the authentication necessary to reach the public Internet.

VPN client
OS X includes a virtual private network (VPN) client that supports the Internet standard Layer 2 Tunneling Protocol (L2TP) over IPSec (the secure version of IPv4), as well as the older Point-to-Point Tunneling Protocol (PPTP). OS X also includes a VPN framework that developers can use to build additional VPN clients.

Firewalls
In addition to the ipfw2-based systemwide firewall, OS X includes an application firewall that can be configured to allow incoming access to preapproved applications and services only.

Self-tuning TCP
OS X sets the initial maximum TCP window size according to the local resources and connection type, enabling TCP to optimize performance when connecting to high-bandwidth/high-latency networks.

Port mapping
NAT-PMP enables you to export Internet services from behind a NAT gateway, while Wide Area Bonjour lets you register the resulting port number with Back to My Mac. This enables you to easily and securely access your home printer and disk drives remotely, even from the public Internet.

Bonjour

Bonjour is Apple's implementation of the Zero Configuration Networking standard. It helps applications discover shared services such as printers on the local network. It also enables services to coordinate within and across machines without requiring well-known port numbers. The ability of Bonjour to painlessly find other computers over a network is critical to many Apple technologies, such as AirPlay, AirDrop, and Back to My Mac.

Link-local addressing
Any user or service on a computer that needs an IP address benefits from this feature automatically. When your host computer encounters a local network that lacks DHCP address management, it finds an unused local address and adopts it without you having to take any action.

mDNS Responder
The mDNS Responder daemon is responsible for unicast DNS resolution, multicast DNS resolution, and Service Discovery on the system. It performs queries and registrations, and provides answers on behalf of other clients through the DNS Service Discovery API as documented in dns_sd.h.

The design of mDNS Responder is modular, separated by function into several different subsystems that operate somewhat independently but also have a cooperative relationship. Their operation is coordinated by a lightweight event system that allows arbitrary data to be passed between subsystems, and also facilitates arbitrary connection topologies between clients. A publish-subscribe system built on top allows the subsystems to communicate without direct knowledge of interested clients.

Service discovery
Service discovery allows applications to find all available instances of a particular type of service and to maintain a list of named services. The application can then dynamically resolve a named instance of a service to an IP address and port number. Concentrating on services rather than devices makes the user's browsing experience more useful and trouble-free.

Unicast DNS

One of the main subsystems is unicast DNS, an implementation of the domain name resolution system and protocol as described in RFC 1034. It is a caching DNS forwarder that reduces outgoing network traffic by caching DNS records locally for their time to live, or TTL, lifetimes. It also provides higher performance since records that are still live in the cache can enable answers to be returned to clients immediately, without waiting for an answer from the network. Multiple name servers as well as multiple simultaneous resolver configurations are supported to allow for VPN configurations. The unicast subsystem also supports dynamic DNS registrations for Wide-Area Bonjour.

Multicast DNS

Multicast DNS (mDNS) uses DNS-format queries over IP multicast to resolve local names not handled by a central DNS server. Bonjour goes further by handling mDNS queries for any network service on the host computer. This relieves your application of the need to interpret and respond to mDNS messages. By registering your service with the Bonjour daemon, OS X automatically directs any queries for your name to your network address.

Bonjour supports the protocol described in RFC 6762; and for service registration and discovery, as described in RFC 6763. This implementation has a specialized DNS caching system for multicast DNS querying, and supports continuous querying. For registering records, probing, announcing, and name conflict resolution are supported. Several traffic reduction strategies are employed to coalesce records into single DNS messages, minimizing the number of multicast frames.

Wide-Area Bonjour

Bonjour now uses Dynamic DNS Update (RFC 2136) and unicast DNS queries to enable discovery and publishing of services to a central DNS server. These can be viewed in the Bonjour tab of Safari in addition to other locations. This feature can be used by companies to publicize their intranet or by retailers to advertise promotional websites.

Wake on Demand

Wake on Demand allows your Mac to sleep yet still advertise available services via a Bonjour Sleep Proxy (typically an AirPort Extreme Base Station) located on your network. The proxy automatically wakes your machine when clients attempt to access it. Your Mac can even periodically do a maintenance wake to renew its DHCP address and other leases.

High-level APIs

OS X provides multiple APIs for publication, discovery, and resolution of network services, as follows:

- NSNetService and NSNetServiceBrowser classes, part of the Cocoa Foundation framework, provide object-oriented abstractions for service discovery and publication.
- The CFNetServices API declared in the Core Services framework provide Core Foundation–style types and functions for managing services and service discovery.
- The DNS Service Discovery API, declared in `</usr/include/dns_sd.h>`, provides low-level BSD socket communication for Bonjour services.

Open source

The complete Bonjour source code is available under the Apache License, Version 2.0, on Apple’s open source website, where it’s called the mDNSResponder project. You can easily compile it for a wide range of platforms, including UNIX, Linux, and Windows. We encourage hardware device manufacturers to embed the open source mDNSResponder code directly into their products and, optionally, to pass the Bonjour Conformance Test so they can participate in the Bonjour Logo Licensing Program.

Document Lifecycle

Auto Save

Thanks to Auto Save, you no longer need to manually save important documents every few minutes. Applications that support Auto Save automatically save your data in the background whenever you pause or every five minutes, whichever comes first. If the current state of your document has been saved, OS X won't even prompt you before quitting the application, making logouts and reboots virtually painless.

Automatic Versions

Versions automatically records the history of a document as you create and make changes to it. OS X automatically creates a new version of a document each time you open it and every hour while you're working on it. You can also manually create snapshots of a document whenever you like.

OS X uses a sophisticated chunking algorithm to save only the information that has changed, making efficient use of space on your hard drive (or iCloud). Versions understands many common document formats, so it can chunk documents between logical sections, not just at a fixed number of bytes. This allows a new version to store, for example, just the one chapter you rewrote instead of a copy of the entire novel.

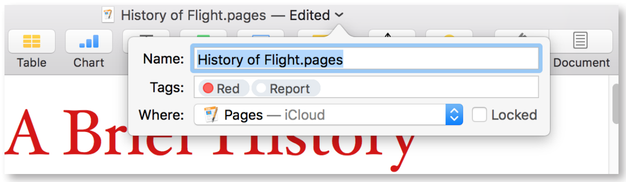
OS X automatically manages the version history of a document for you, keeping hourly versions for a day, daily versions for a month, and weekly versions for all previous months.

To further safeguard important milestones, OS X automatically locks documents that were edited more than two weeks ago. You can change the interval by clicking the Options button in the Time Machine System Preferences pane, then choosing the interval you want from the Lock documents pop-up menu.

When you share a document—for example, through email, Messages, or AirDrop—only the latest, final version is sent. All other versions and changes remain safely on your Mac.

Document Management

You can use the pop-up menu next to the document title to edit the document title, tags, location, and lock status.


History of Flight.pages — Edited

Name: History of Flight.pages

Tags: Red Report

Where: Pages — iCloud ☐ Locked

Continuity

Continuity features let you move seamlessly between your Mac, iPhone, and iPad, or use them together.

You can start an activity on one device and seamlessly resume the activity on another device, receive a call on your iPhone and answer it on your Mac, send and receive SMS messages on your Mac through your iPhone, or activate the hotspot on your iPhone right from your Mac.

Continuity uses multiple technologies to hand off information between a Mac running OS X and a device running iOS.

Bluetooth Low Energy

When you sign in to iCloud, your Mac and iOS device recognize when they're near each other. Bluetooth Low Energy (BTLE) is used for discovery of nearby devices and triggering the Handoff-related app icons to appear in Dock and on the iOS device.

iCloud

iCloud is used to handle transferring data from device to device. For example, a map location or a draft of an email started on one device is handed off to another by transferring it over the user's iCloud account. In this situation, iCloud acts as an intermediary, so Continuity knows which devices work with each other, and which devices belong to which user.

Wi-Fi (peer to peer)

Wi-Fi is used when apps need to hand off large amounts of data (such as when using Mail, Pages, or Keynote). OS X attempts to set up a peer-to-peer Wi-Fi connection with the other device to transfer the data automatically over the connection.

Cellular and Wi-Fi

When a phone call is transferred from an iPhone to a Mac, the iPhone continues to handle the cellular connection. Caller ID and incoming and outgoing audio are transmitted between the iPhone and your Mac or iPad using Wi-Fi. In turn, the iPhone transmits the audio to the cell tower for cellular-based calls.

Both cellular and Wi-Fi are used as part of the SMS/MMS messaging transfer. Forwarding of an SMS or MMS message from an iPhone to another iOS device or Mac uses both cellular and Wi-Fi.

Extensions

Extensions let you share photos, videos, and links; manipulate content; and even enter passwords from third-party applications without leaving the application you are in. App developers can extend their applications to enable specific tasks, such as posting a remark to a social network, or displaying current sports scores in Notification Center.

Extension types

OS X defines several types of app extensions, each tied to an area of the system. A system area that supports extensions is called an *extension point*, and each has specialized usage policies and APIs that developers can use when creating an extension for that area.

- **Today.** Get a quick update or perform a quick task in the Today view of Notification Center.
- **Sharing.** Post to a sharing website or share content with others.
- **Action.** Manipulate or view content originating in a host app.
- **Finder Sync.** Present information about file sync state directly in Finder.

iCloud Storage

iCloud Storage APIs enable apps to store documents and key value data in iCloud. iCloud wirelessly pushes documents to your devices and updates them whenever they are changed on any of your devices—automatically.

iCloud Drive

You can access the latest versions of your files in iCloud Drive from any device. On the Mac, iCloud Drive appears in Finder and works like any other folder: Just drag in documents and organize them any way you like with folders or tags, and iCloud keeps everything up to date across all of your devices.

Files in iCloud Drive are available on your Mac even when you don't have an Internet connection. Any changes you make to a file while offline are automatically synced as soon as your Mac is back online.

With iCloud Drive, you can safely store any kind of document in iCloud and access it from all your devices.

CloudKit

The CloudKit APIs enable apps to store data in the cloud so users can access it on multiple devices. The CloudKit framework provides interfaces for moving data between your app and your iCloud containers. You can also store data in a public area where all users can access it.

Ubiquitous storage

Making a document ubiquitous using iCloud means you can view or edit those documents from any device without having to sync or transfer files explicitly. Storing documents in your iCloud Drive also provides an extra layer of protection. Even if you lose a device, those documents are still available on your iCloud Drive.

File coordination

Because the file system is shared by all running processes, conflicts can occur when two processes (or two threads in the same process) try to change the same file at the same time. To avoid this type of contention, OS X includes support for file coordinators, which enable developers to safely coordinate file access between different processes or different threads.

File coordinators mediate changes between applications and the daemon that facilitates the transfer of the document to and from iCloud Drive. In this way, the file coordinator acts as a locking mechanism for the document, preventing applications and the daemon from modifying the document simultaneously.

Safe versions
Versions automatically stores iCloud Drive documents. This means you don't have to resolve conflicts or decide which version to keep. It automatically chooses the most recent version. You can always use the Browse Saved Versions option if you want to revert to a different one. Versions' chunking mechanism also minimizes the information that needs to be sent across the network.

Ubiquitous metadata, lazy content
iCloud immediately updates the metadata (that is, the file name and other attributes) for every document stored or modified in the cloud. However, iCloud may not push the actual content to devices until later, perhaps only when actively requested. Devices always know what's available but defer loading the data in order to conserve storage and network bandwidth.

Web access
iCloud provides a range of powerful web applications to let you work directly with your data from a web browser. These include the usual personal information tools such as Mail, Calendar, and Contacts, as well as Pages, Keynote, and Numbers.

Two-Factor Authentication

Two-factor authentication is an additional layer of security for your Apple ID that is designed to prevent unauthorized access to your account and protect the photos, documents, and other data that you store with Apple. Two-factor authentication can keep your account secure even if someone has your password.

Whenever you sign in with your Apple ID on a new device or browser, you will verify your identity by entering your password plus a six-digit verification code. The verification code will be displayed automatically on any Apple devices you are already signed in to that are running iOS 9 or OS X El Capitan. Just enter the code to complete sign-in. If you don't have an Apple device handy, you can receive the code on your phone via a text message or phone call instead.

Once signed in, you won't be prompted for a verification code again on that device unless you erase your device, remove it from your device list, or need to change your password for security reasons. When signing in on the web, you can choose to trust your browser so you won't be prompted for a verification code the next time you sign in from that computer.

Data Management

Spotlight

Spotlight is a fast desktop search technology that helps you organize and search for files based on either contents or metadata. It's available to users via the Spotlight window in the upper right of the screen. Developers can embed Spotlight in their own applications using an API available from Apple.

Standard metadata
Spotlight defines standard metadata attributes that provide a wide range of options for consistently storing document metadata, making it easier to form consistent queries. These include POSIX-style file attributes, authoring information, and specialized metadata for audio, video, and image file formats.

Extensible importers
Using OS X Launch Services, Spotlight determines the uniform type identifier of a new or modified file and attempts to find an appropriate importer plug-in. If an importer exists and is authorized, OS X loads it and passes it the path to the file.

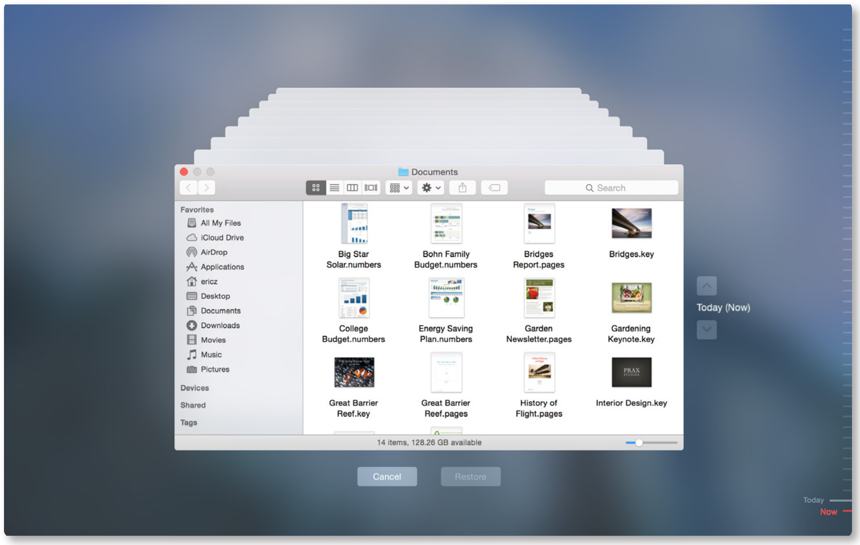
Third parties can create custom importers that extract both standard and custom metadata for a given file type and return a dictionary that is used to update the Spotlight datastore.

Dynamic datastore
Every time you create, modify, or delete a file, the kernel notifies the Spotlight engine that it needs to update the system store. OS X accomplishes this with the high-performance `fsevents` API.

Live update
Whenever OS X updates the datastore, it also notifies the system results window and any third-party client applications if the update causes different files to match or not match the query. This ensures that the Mac always presents the latest real-time information to the user.

Time Machine

Time Machine makes it easy to back up and restore either your entire system or individual files.



Browse by date to see how your system looked at any previous time. And restore your Mac with a click.

Easy setup

To set up Time Machine, all you need to do is select a local disk or Time Capsule to store the backups. With OS X, you can select multiple backup destinations for Time Machine. OS X immediately starts backing up all your files in the background. After the initial backup, it automatically creates new incremental backups every hour.

Coalescing changes

Time Machine leverages the `fsevents` technology developed for Spotlight to continuously identify and track any folders (what UNIX calls *directories*) that contain modified files. During the hourly backup, it creates a new folder that appears to represent the entire contents of your hard drive. In reality, it uses a variant of UNIX hard links that mostly point to trees of unmodified folders already on the disk. Those trees are effectively copy-on-write, so that future changes never affect the backup version.

Time Machine creates new trees inside a backup for any path that contains modified folders. Time Machine creates new versions of those folders that contain links to the current files, thus automatically capturing any changes that occurred in the past hour. This avoids the overhead of either scanning every file on disk or capturing each and every change to a file.

This technique allows each backup to provide the appearance and functionality of a full backup while only taking up the space of an incremental backup (plus some slight overhead for the metadata of modified trees). This makes it easy to boot or clone a system from the most recent Time Machine backup.

Mobile Time Machine
Mobile Time Machine keeps track of modified files even while you are disconnected from your backup drive. When you reconnect, it will automatically record the hourly snapshots to ensure you don't lose your version history.

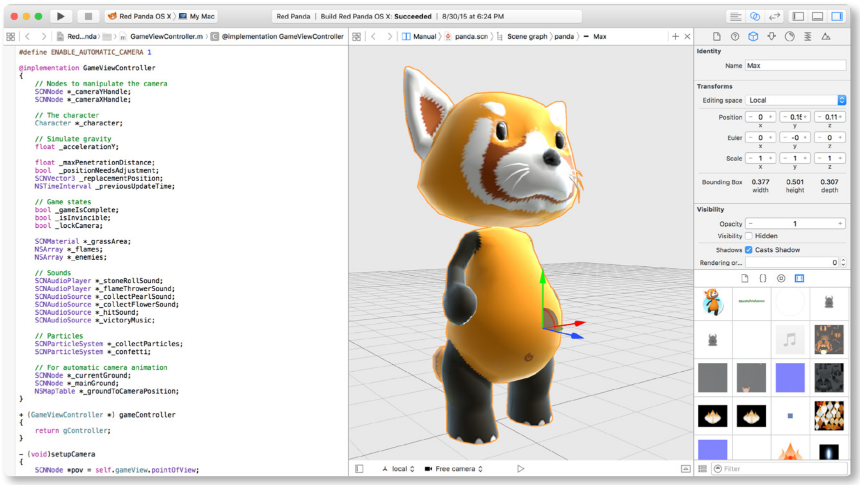
Preserving backups
Time Machine keeps hourly backups for the past 24 hours, daily backups for the past month, and weekly backups until your backup drive is full. At that point, OS X warns you that it is starting to delete older backups. To be notified whenever OS X deletes an older backup, open Time Machine preferences, click the Options button, and select the checkbox next to "Notify after old backups are deleted."

Developer Tools

Xcode

Xcode 7 is the latest version of Apple's integrated development environment (IDE). Xcode provides everything developers need to create great applications for Mac, iPhone, and iPad. Xcode brings user interface design, coding, testing, and debugging all into a unified workflow. The Xcode IDE combined with the Cocoa and Cocoa Touch frameworks, and the Swift programming language make developing apps easier and more fun than ever before.

Xcode includes the Xcode IDE, Swift, and Objective-C compilers, the Instruments analysis tool, iOS Simulator, and the latest OS X and iOS SDKs.



```
#define ENABLE_AUTOMATIC_CAMERA 1

@implementation GameViewController
{
    // Nodes to manipulate the camera
    SCNNode *_cameraYHandle;
    SCNNode *_cameraXHandle;

    // The character
    Character *_character;

    // Simulate gravity
    float _accelerationY;

    float _maxPenetrationDistance;
    bool  _positionNeedsAdjustment;
    SCNVector3 _replacementPosition;
    NSTimeInterval _previousUpdateTime;

    // Game states
    bool _gameIsComplete;
    bool _isInvincible;
    bool _lockCamera;

    SCNMaterial *_grassArea;
    NSArray *_flames;
    NSArray *_enemies;

    // Sounds
    SCNAudioPlayer *_stoneRollSound;
    SCNAudioPlayer *_flameThrowerSound;
    SCNAudioSource *_collectPearlSound;
    SCNAudioSource *_collectFlowerSound;
    SCNAudioSource *_hitSound;
    SCNAudioSource *_victoryMusic;

    // Particles
    SCNParticleSystem *_collectParticles;
    SCNParticleSystem *_confetti;

    // For automatic camera animation
    SCNNode *_currentGround;
    SCNNode *_mainGround;
    NSMapTable *_groundToCameraPosition;
}

+ (GameViewController *) gameController
{
    return gController;
}

- (void)setupCamera
{
    SCNNode *pov = self.gameView.pointOfView;
```

The Xcode 7 integrated development environment.

Swift

Swift is a new programming language for creating iOS and OS X apps. Swift builds on the best of C and Objective-C, without the constraints of C compatibility. Writing code is interactive and fun, the syntax is concise yet expressive, and apps run lightning-fast. Swift adopts safe programming patterns and adds modern features to make programming easier, more flexible, and more fun.

Swift 2.0 is the latest release of this modern, powerful, and expressive language. It extends Swift 1.2 with the following:

- **Error handling.** Now you can create routines that throw, catch, and manage errors in Swift. You can surface and deal with recoverable errors like "file-not-found" or network timeouts. The Swift 2.0 error handling interoperates seamlessly with NSError.

- **Availability.** Availability features enable you to mark up methods with the operating system that they are available on, providing compile-time checking and preventing unavailable methods from being used. You can adopt new APIs while still deploying back to older OS versions, and provide compile-time errors when you're using an API that isn't guaranteed to be available on the deployment target.
- **Protocol extensions.** You can now add methods and properties to any class that conforms to a particular protocol, allowing you to reuse more of your code. For example, in the standard library, instead of having to use global functions for all of the generic algorithms (such as map, filter, and sort), those generic algorithms are now available as methods on all of the collection types.
- **Testability.** With testability, you are now able to write tests of Swift 2.0 frameworks and apps without having to make all of your internal routines public. Use @testable import {`ModuleName`} in your test source code to make all public and internal routines usable by XCTest targets, but not by other framework and app targets.

Modern
Swift is the result of the latest research on programming languages, combined with decades of experience building Apple platforms. Named parameters brought forward from Objective-C are expressed in a clean syntax that makes APIs in Swift even easier to read and maintain.

```
var
sortedStrings = sorted(stringArray
) {
     $0.
uppercaseString < $1.uppercaseString
}
```

Inferred types make code cleaner and less prone to mistakes, while modules eliminate headers and provide namespaces. Memory is managed automatically, and you don't even need to type semicolons.

Swift has many other features to make your code more expressive:

- Closures unified with function pointers
- Tuples and multiple return values
- Generics
- Fast and concise iteration over a range or collection
- Structs that support methods, extensions, and protocols
- Functional programming patterns; for example, map and filter

Interactive playgrounds
Playgrounds make writing Swift code incredibly simple and fun. Type a line of code and the result appears immediately. If your code runs over time—for instance, through a loop—you can watch its progress in the timeline assistant. The timeline displays variables in a graph, draws each step when composing a view, and can play an animated SpriteKit scene. When you've perfected your code in the playground, simply move that code into your project. Following are some uses for playgrounds:

- Design a new algorithm, watching its results every step of the way.
- Create new tests, verifying they work before promoting into your test suite.
- Experiment with new APIs to hone your Swift coding skills.

Read-Eval-Print-Loop (REPL)
The debugging console in Xcode includes an interactive version of the Swift language built right in. Use Swift syntax to evaluate and interact with your running app, or write new code to see how it works in a script-like environment. Available from within the Xcode console or in Terminal.

Designed for safety
Swift eliminates entire classes of unsafe code. Variables are always initialized before use, arrays and integers are checked for overflow, and memory is managed automatically. Syntax is tuned to make it easy to define your intent—for example, simple three-character keywords define a variable (var) or constant (let).

The safe patterns in Swift are tuned for the powerful Cocoa and Cocoa Touch APIs. Understanding and properly handling cases where objects are nil is fundamental to the frameworks, and Swift code makes this extremely easy. Adding a single character can replace what used to be an entire line of code in Objective-C. This all works together to make building iOS and Mac apps easier and safer than ever before.

Fast and powerful
From its earliest conception, Swift was built to be fast. Using the incredibly high-performance LLVM compiler, Swift code is transformed into optimized native code, tuned to get the most out of modern Mac, iPhone, and iPad hardware. The syntax and standard library have also been tuned to make the most obvious way to write your code also perform the best.

Swift is a successor to the C and Objective-C languages. It includes low-level primitives such as types, flow control, and operators. It also provides object-oriented features such as classes, protocols, and generics, giving Cocoa and Cocoa Touch developers the performance and power they demand.

Testing and continuous integration
The Xcode Test Navigator makes it easy to add or edit tests, and execute them one at a time or as a group. The XCTest framework enables you to write tests that run on Mac, iPhone, iPad, or iOS Simulator, from within Xcode or the command line.

Once you have a test harness created, Xcode can configure a continuous integration "bot" on any Mac in your network running OS X Server. The Xcode bot on the remote Mac will perform an integration—build, analyze, test, and archive your app—when anyone on your team commits to source control, or at a defined interval.

Static analysis
You can think of static analysis as providing you advanced warnings by identifying bugs in your code before it is run—hence the term *static*. The Xcode static analyzer gives you a much deeper understanding of your code than do traditional compiler warnings. The static analyzer leverages the Clang code processor to travel down each possible code path, identifying logical errors such as unreleased memory—well beyond the simple syntax errors normally found at compile time.

Fix-it
Fix-it brings autocorrection from the word processor to your source code. The Xcode Fix-it feature checks your symbol names and code syntax as you type, highlights any errors it detects, and even fixes them for you. Fix-it marks syntax errors with a red underbar or a caret at the position of the error and a symbol in the gutter. Clicking the symbol displays a message describing the possible syntax error and, in many cases, offers to repair it automatically.

Source control
Xcode also provides integrated support for Git and Subversion repositories, including an option to create a local Git repository when you create a new project. Once your project is under source control, Xcode provides a special editor, called the *version editor*, that compares different versions of files back through time so you can watch as code changes, and determine who made each change.

LLVM

The next-generation Apple LLVM compiler is based on the open source LLVM.org project, which employs a unique approach of building compiler technologies as a set of libraries. Capable of working together or independently, these libraries enable rapid innovation and provide the ability to attack problems never before solved by compilers.

Apple's compiler, runtime, and graphics teams are extensive contributors to the LLVM.org community. They use LLVM technology to make Apple platforms faster and more secure.

Clang front-end
Clang is a high-performance front-end for parsing C, Objective-C, and C++ code as part of the Apple LLVM compiler. It supports the latest C++ standards, including full C++11 support. Clang is also implemented as a series of libraries, allowing its technology to be embedded within the Xcode IDE to enable static code analysis and Fix-it.

Comprehensive optimization
LLVM's flexible architecture makes it easy to add sophisticated optimizations at any point during the compilation process. For example, LLVM performs whole-program analysis and link-time optimizations to eliminate unused code paths.

Automatic Reference Counting
Automatic Reference Counting (ARC) for Objective-C lets the compiler take care of memory management. By enabling ARC with the new Apple LLVM compiler, you never need to manually track object lifecycles using `retain` and `release`, dramatically simplifying the development process while reducing crashes and memory leaks. The compiler has a complete understanding of your objects, and releases each object the instant it is no longer used. Apps run as fast as ever, with predictable, smooth performance.

Instruments

Instruments is an application for dynamically tracing and profiling OS X and iOS code. It is a flexible and powerful tool that lets you track one or more processes, examine the collected data, and track correlations over time. In this way, Instruments helps you understand the behavior of both user programs and the operating system.

Synchronized tracks
The Instruments Track pane displays a graphical summary of the data returned by the current instruments. Each instrument has its own track, which provides a chart of the data collected by that instrument. You use this pane to select specific data points you want to examine more closely.

Multiple traces
Each time you click the Record button in a trace document, Instruments starts gathering data for the target processes. Rather than appending the new data to any existing data, Instruments creates a new trace run to store that data. This makes it easy to compare behavior between different configurations.

A trace run consists of all of the data gathered between the time you click the Record button and the Stop button. By default, Instruments displays only the most recent trace run in the Track pane, but you can view data from previous trace runs in one of two ways:

- Use the Time/Run control in the toolbar to select which trace run you want to view.
- Click the disclosure triangle next to an instrument to display the data for all trace runs for that instrument.

DTrace
DTrace is a dynamic tracing facility available for Mac systems. Because DTrace taps into the operating system kernel, you have access to low-level information about the kernel itself and about the user processes running on your computer. DTrace is used to power many of the built-in instruments.

DTrace probes make it easy to use Instruments to create custom instruments. A probe is a sensor you place in your code that corresponds to a location or event (such as a function entry point) to which DTrace can bind. When the function executes or the event is generated, the associated probe fires and DTrace runs whatever actions are associated with the probe.

Most DTrace actions simply collect data about the operating system and user program behavior at that moment. It is possible, however, to run custom scripts as part of an action. Scripts let you use the features of DTrace to fine-tune the data you gather. That data is then available as an Instruments track to compare with data from other instruments or other trace runs.

Accelerate

Accelerate is a unique framework of hardware-optimized math libraries that provides the following:

- **Vector digital signal processing (vDSP).** Optimized Fast Fourier Transforms (FFTs), convolutions, vector arithmetic, and other common video and audio processing tasks for both single- and double-precision data.
- **Vector image processing (vImage).** Optimized routines for convolutions, compositing, color correction, and other image-processing tasks, even for gigapixel images.
- **vForce.** Designed to wring optimal efficiency from modern hardware by specifying multiple operands at once, allowing only default IEEE-754 exception handling.
- **Linear Algebra Package (LAPACK).** Industry-standard APIs written on top of BLAS for solving common linear algebra problems.
- **Basic Linear Algebra Subprograms (BLAS) Levels I, II, and III.** High-quality “building block” routines that perform basic vector and matrix operations using standard APIs.
- **vMathLib.** A vectorized version of libm that provides transcendental operations, enabling you to perform standard math functions on many operands at once.

Automation

AppleScript
AppleScript is Apple’s native language for application automation, as used by the AppleScript Editor. Its English-like syntax generates Apple events, which use a scripting dictionary (provided by most Mac applications) to programmatically create, edit, or transform their documents. AppleScript and other Open Scripting Architecture (OSA) scripts can be activated by contextual menus, user interface elements, Calendar events, and even folder actions, such as drag and drop.

Automator

Automator provides a graphical environment for assembling actions (typically built from AppleScript or shell scripts) into sophisticated workflows, which can be saved as either stand-alone applications or as custom services, print plug-ins, folder actions, Calendar alarms, and Image Capture plug-ins.

Apple events

The Apple Event Bridge framework provides an elegant way for Cocoa applications (including bridged scripting languages) to generate Apple events based on an application's dictionary, even generating appropriate header files if necessary.

Services

The Services menu lets you focus on only those actions relevant to your current selection, whether in the menu bar, the Finder action menu, or a contextual menu. Individual services can also be disabled and assigned shortcuts from the Keyboard pane in System Preferences.

JavaScript OSA

Automation in OS X has always been about power and choice. Scriptable applications, such as Pages, Keynote, Numbers, and the Finder, are automated using a variety of languages, including AppleScript, Objective-C, Perl, Python, and Ruby. With OS X El Capitan, application scripting support has been bolstered with the inclusion of another popular language: JavaScript.

JavaScript for Automation (JXA) extends the JavaScriptCore framework with support for querying and controlling all of the scriptable applications running in OS X. In addition, JavaScript for Automation incorporates an Objective-C bridge providing inline access to all of the Cocoa frameworks.

And to make it even better, JXA scripts are integrated into all aspects of the system and can be invoked from the command line, from the systemwide Script Menu, and can even be distributed as code-signed applets and droplets.

iCloud Drive support

iCloud Drive support has been added to both the AppleScript Editor and Automator applications. As with other Apple applications, users can store their automation documents—including AppleScript scripts, applets, droplets, and Automator workflow files and applets—in iCloud Drive. This service ensures that your favorite automation tools can be easily shared across all of your Mac laptop and desktop computers.

Code signing

Support for code signing has been added as an export feature to both the AppleScript Editor and Automator applications, enabling Apple developers to easily generate signed copies of their applets and droplets. Such signed automation applets will not trigger OS security warnings on computers that are using the default Gatekeeper setting of allowing only applications downloaded from the Mac App Store and identified developers to execute. This feature is a boon for developers of automation-supporting Mac applications and for solution providers, allowing them to distribute smooth-launching automation solutions to their customers and clients.

AppleScript Libraries

AppleScript Libraries provide a new plug-in architecture for extending the power and abilities of AppleScript. AppleScript Libraries are user-created script files and bundles, written using AppleScript or AppleScript/Objective-C, that can be referenced in scripts to provide specialized handlers and functionality. In addition to providing access to standard Cocoa classes and methods through the use of AppleScript/Objective-C, AppleScript Libraries can publish their own scripting terminology, making it easier for scripters to remember and incorporate custom commands in their scripts. AppleScript

Libraries have been made universally available through placement in new Script Libraries directories created in the standard OS Library domains, enabling AppleScript Libraries to be easily distributed between multiple computers running OS X.

The "use" clause
An AppleScript construct called the "use" clause imports the terminology and functionality of AppleScript Libraries and scriptable applications through a simple single-line declaration placed at the top of a script, such as "use application Finder," or "use My AppleScript Text Library." The handlers, scriptable objects, properties, and terminology of AppleScript Libraries and scriptable applications imported via a use clause are automatically available globally throughout the hosting script—no longer requiring that numerous "tell statements" or "tell blocks" be compiled and executed. Scripts taking advantage of the use clause are more streamlined and clearer than similar scripts not implementing this construct.

Notification support
AppleScript and Automator now allow system notifications to be created and displayed. Users can incorporate notification banners and alerts into scripts and workflows with the default AppleScript Display Notification command or the Automator Display Notification action. Automation processes requiring extended time to execute no longer need to display notification dialogs to communicate their progress or completion. Custom notifications—containing relevant information about the automation processing—can be placed at the start, end, or throughout an automation workflow, providing current feedback and status to the user.

Speakable-Workflows
If the Speakable Items feature is activated in the Accessibility System Preferences pane, Automator will now present an option in the Automator Save dialog: to save applets as Speakable Items, automatically added to the Speakable Items Speech Recognition architecture, and available for execution by simply speaking the names of the saved applets. Speakable-Workflows extend the scope of the built-in Accessibility Speech Recognition commands with the expansive automation architecture of OS X.

WebKit

WebKit is an open source web browser engine developed by Apple. WebKit's HTML and JavaScript code began as a branch of the KHTML and KJS libraries from KDE.

WebKit is also the name of the OS X system framework version of the engine that's used by Safari, Dashboard, Mail, and many other OS X applications. Key features include:

- Lightweight footprint
- Great mobile support
- Rich HTML5 functionality
- Easy to embed in Cocoa and Cocoa Touch applications
- Available as open source at webkit.org

For More Information

- Extensible Firmware Interface (EFI): See *www.uefi.org*
- I/O Kit: See *Kernel Programming Guide: I/O Kit Overview*
- Partition Schemes: See *Technical Note TN2166: Secrets of the GPT*
- Recovery Partitions: See *OS X: About OS X Recovery*
- Full-Disk Encryption: See *OS X: About FileVault 2*
- Backup: See *Mac Basics: Time Machine*
- File System Events: See *Spotlight Overview*
- Launchd: See *Daemons and Services Programming Guide*
- Grand Central Dispatch (GCD): See *Concurrency Programming Guide*
- Sandboxes: See *Code Signing Guide*
- About System Integrity Protection: *https://support.apple.com/kb/HT204899*
- Gatekeeper: See *Distributing Outside the Mac App Store*
- Bonjour: See *Bonjour Overview*
- XPC: See *Daemons and Services Programming Guide: Creating XPC Services*
- App Transport Security: *https://developer.apple.com/library/prerelease/ios/technotes/App-Transport-Security-Technote/*
- iCloud: See *iCloud Design Guide*
- LLVM: See *LLVM Compiler Infrastructure Project*
- Xcode: See *Xcode User Guide*
- Instruments: See *Instruments User Guide*
- Automation: See *AppleScript Overview*
- WebKit: See *WebKit Objective-C Programming Guide*

For More Information

For more information about OS X El Capitan, visit www.apple.com/osx.