iOS 9.0

This article summarizes the key developer-related features introduced in iOS 9, which runs on currently shipping iOS devices. The article also lists the documents that describe new features in more detail.

For late-breaking news and information about known issues, see *iOS 9 Release Notes*. For the complete list of new APIs added in iOS 9, see *iOS 9.0 API Diffs*. For more information on new devices, see *iOS Device Compatibility Reference*.

Multitasking Enhancements for iPad

iOS 9 enhances the user's multitasking experience on iPad with Slide Over, Split View, and Picture in Picture. The Slide Over feature lets users pick a secondary app and quickly interact with it. The Split View feature gives users the ability to use two apps side by side on iPad Air 2. The Picture in Picture feature (also known as PiP) lets users watch video in a window that floats above other onscreen apps.

Users decide when they want to view two apps on the screen at the same time; you have no control over when this happens. Even though Split View and Slide Over are user-controlled, there are a few things you need to do to ensure that users have a great multitasking experience.

- It's crucial that your app use system resources efficiently so that it can run well when it shares the system with another running app. Under memory pressure, the system preemptively quits the app that's consuming the most memory. For some guidance on creating energy efficient iOS apps that perform well, see *Energy Efficiency Guide for iOS Apps*.
- If you haven't already, be sure to adopt size classes so that your app looks good when the user decides to view it in a portion of the device screen.

To learn more about preparing your app to respond well when users use Split View and Slide Over, see *Adopting Multitasking Enhancements on iPad*.

As with Split View and Slide Over, users control whether they want to use PiP to view a video on top of another running app. If video playback is not your app's primary functionality, there's nothing you need to do to support the PiP experience.

To participate when users choose Picture in Picture, use AVKit or AV Foundation APIs. The video playback classes defined in the Media Player framework are deprecated in iOS 9 and do not support PiP. To learn how to prepare your video playback app for PiP, see Picture in Picture Quick Start.

3D Touch

3D Touch gives iOS 9 users an additional interaction dimension. On supported devices, people can quickly choose app-specific actions from the Home screen by pressing on the app icon. Within an app, people can use various pressures to get a preview of an item, open the item in a separate view, and access related actions.

iOS 9 provides the following 3D Touch APIs:

- The Home screen quick action API is for adding shortcuts to your app icon that anticipate and accelerate a user's interaction with your app (see the `UIApplicationShortcut`* APIs, such as `UIApplicationShortcutItem`.)
- The UIKit peek and pop APIs let you provide easy access to additional content within your app, while maintaining the user's context (see `UIViewControllerPreviewing`, `UIViewControllerPreviewingDelegate`, and new methods in `UIViewController`). Use the peek quick actions API to provide a press-enabled replacement to your app's touch-and-hold actions (see the `UIPreview`* APIs, such as `UIPreviewAction` and `UIPreviewActionItem`).
- The web view peek and pop API lets you enable previews of HTML link destinations (see `WKWebView`).
- The `UITouch` force properties let you add customized force-based user interaction to your app.

No matter which of these APIs you adopt, your app must check the availability of 3D Touch at runtime. To learn more about supporting 3D Touch, see *Adopting 3D Touch on iPhone*. For some examples of using 3D Touch APIs in your app, see *ApplicationShortcuts: Using UIApplicationShortcutItem* and *ViewControllerPreviews: Using the UIViewController previewing APIs*.

Search

Search in iOS 9 gives users great new ways to access information inside of your app, even when it isn't installed. When you make your content searchable, users can access activities and content deep within your app through Spotlight and Safari search results, Handoff, and Siri suggestions. Using APIs related to search, you decide what content gets indexed, what information to show in search results, and where the user is redirected after tapping a result from your app or website.

Integrating with search in iOS 9 is easy: You don't need any prior experience with implementing search, and most developers find that it takes only a few hours to make their content universally searchable. To learn how to make your app and website content searchable, see *App Search Programming Guide*.

Privacy is an essential feature of search in iOS 9. To give users the best search experience while protecting their private data, iOS 9 makes the following indexes available:

- A private on-device index. Each device contains a private index whose information is never shared with Apple or synced between devices. When you make an item available in a user's on-device index, only that user can view the item in search results.
- Apple's server-side index. The server-side index stores only publicly available data that you've marked appropriately on your website.

iOS 9 provides the following APIs to help you make your content searchable:

- The `NSUserActivity` class includes new methods and properties that help you index items as users perform activities in your app such as visiting a navigation point or creating and viewing content. Just about every app can take advantage of the `NSUserActivity` APIs to make useful content available to users.
- The Core Spotlight framework (`CoreSpotlight.framework`) provides APIs that help you add app-specific content to the on-device index and enable deep links into your app. To learn more about Core Spotlight APIs, see *Core Spotlight Framework Reference*.
- Web markup lets you make your related web content searchable and helps you enrich the user's

search experience. To learn how to mark up your website, see Mark Up Web Content.

In addition, adding a Smart App Banner gives users an easy way to link directly to your app (to learn how to use a Smart App Banner, see Promoting Apps with Smart App Banners).

- Universal links let you replace custom URL schemes with standard HTTP or HTTPS links. Universal links work for all users: If users have your app installed, the link takes them directly into your app; if they don't have your app installed, the link opens your website in Safari. To learn more about universal links, see Support Universal Links.

Gaming

iOS 9 includes several technology improvements that make it easier than ever to implement your game's graphics and audio features. Take advantage of high-level frameworks for ease-of-development, or use new low-level enhancements to harness the power of the GPU.

GameplayKit

The GameplayKit framework (`GameplayKit.framework`) provides foundational technologies for building games. Use GameplayKit to develop gameplay mechanics, and combine it with any high-level graphics engine—such as SceneKit or SpriteKit—to build a complete game. This framework provides building blocks for creating games with modular architecture, including:

- Randomization tools for adding unpredictability to gameplay without compromising debugging
- Entity-component architecture to design gameplay code for better reusability
- State machines for untangling complex procedural code in gameplay systems

GameplayKit also includes standard implementations of common gameplay algorithms, so you can spend less time reading white papers and more time working on the mechanics that make your game unique. Several of the standard algorithm implementations in GameplayKit are listed below.

- A minmax artificial intelligence for adversarial turn-based games.
- An agent simulation that lets you describe movement behaviors in terms of high-level goals to be automatically pursued.
- Rule systems for building data-driven game logic, fuzzy reasoning, and emergent behavior.

To learn more about GameplayKit, see *GameplayKit Programming Guide* and *GameplayKit Framework Reference*. To see GameplayKit in action, download the sample code projects *FourInARow: Using the GameplayKit Minmax Strategist for Opponent AI*, *AgentsCatalog: Using the Agents System in GameplayKit*, and *DemoBots: Building a Cross Platform Game with SpriteKit and GameplayKit*.

Model I/O

The Model I/O framework (`ModelIO.framework`) provides a system-level understanding of 3D model assets and related resources. You can use this framework for several types of tasks, such as:

- Importing mesh data, material descriptions, lighting and camera settings, and other scene information from file formats used by popular authoring software and game engines

- Processing or generating such data—for example, to bake lighting information into a mesh, or create procedural sky textures
- Together with MetalKit, GLKit, or SceneKit APIs, efficiently loading asset data into GPU buffers for rendering
- Exporting processed or generated asset data to any of several file formats

To learn more about Model I/O, see *Model I/O Framework Reference*.

MetalKit

The MetalKit framework (`MetalKit.framework`) provides a set of utility functions and classes that reduce the effort required to create a Metal app. MetalKit provides development support for three key areas:

- Texture loading helps your app easily and asynchronously load textures from a variety of sources. Common file formats such as PNG and JPEG are supported, as well as texture-specific formats such as KTX and PVR.
- Model handling provides Metal-specific functionality that makes it easy to interface with Model I/O assets. Use these highly-optimized functions and objects to transfer data efficiently between Model I/O meshes and Metal buffers.
- View management provides a standard implementation of a Metal view that drastically reduces the amount of code needed to create a graphics-rendering app.

To learn more about MetalKit APIs, see *MetalKit Framework Reference*. For more information about Metal in general, see *Metal Programming Guide*, *Metal Framework Reference*, and *Metal Shading Language Guide*.

Metal Performance Shaders

The Metal Performance Shaders framework (`MetalPerformanceShaders.framework`) provides highly-optimized compute and graphics shaders that are designed to integrate easily and efficiently into your Metal app. These data-parallel shaders are specially tuned to take advantage of the unique hardware characteristics of each Metal-supported iOS GPU.

Use the Metal Performance Shader classes to achieve optimal performance for all supported hardware, without having to target or update your shader code to specific iOS GPU families. `MetalPerformanceShader` objects fit seamlessly into your Metal apps and can be used with Metal resource objects such as buffers and textures.

Common shaders provided by the Metal Performance Shader framework include:

- Gaussian blur—provided by the `MPSImageGaussianBlur` class.
- Image histogram—provided by the `MPSImageHistogram` class.
- Sobel edge detection—provided by the `MPSImageSobel` class.

New Features in Metal

The Metal framework (`Metal.framework`) adds new features to make your graphics-rendering apps look even better and be more performant. These features include:

- Improvements to the Metal Shading Language and Metal Standard Library

- Compute shaders can now write to a wider range of pixel formats
- The addition of private and depth stencil textures to align with OS X
- The addition of depth clamping and separate front and back stencil reference values for improved shadow quality

New Features in SceneKit

The SceneKit framework (`SceneKit.framework`) includes new features in iOS 9, including:

- Metal rendering support. See the `SCNView` and `SCNSceneRenderer` classes to enable high-performance Metal rendering on supported devices.
- A new Scene Editor in Xcode. Build games and interactive 3D apps in less time and with less code by designing scenes in Xcode (for a related sample code project, download *Fox: Building a SceneKit Game with the Xcode Scene Editor*).
- Positional audio. See the `SCNAudioPlayer` and `SCNNode` classes to add spatial audio effects that automatically track the listener's position in a scene.

For details on these and many other new features, see *SceneKit Framework Reference*.

New Features in SpriteKit

The SpriteKit framework (`SpriteKit.framework`) includes new features in iOS 9, such as:

- Metal rendering support. On devices that support Metal, metal rendering is automatically used, even in cases where you are using custom OpenGL ES shaders.
- An improved Scene Editor and a new Action Editor in Xcode. Build games and interactive 2D apps in less time and with less code by designing scenes in Xcode (for a related sample project, download *DemoBots: Building a Cross Platform Game with SpriteKit and GameplayKit.*)
- Camera nodes (that is, `SKCameraNode` objects) make it even easier to create scrolling games. Simply drop a camera node into your scene and set the scene's camera property.
- Positional audio. To learn how to add spatial audio effects that automatically track the listener's position in a scene, see *SKAudioNode Class Reference*.

For details on these and many other new features, see *SpriteKit Framework Reference*.

App Thinning

App thinning helps you develop apps for diverse platforms and deliver an optimized installation automatically. App thinning includes the following elements:

- Slicing. Artwork incorporated into the Asset Catalog and tagged for a platform allows the App Store to deliver only what is needed for installation.
- On-Demand Resources. Host additional content for your app in the iTunes App Store repository, allowing it to fetch resources as needed using asynchronous download and installation. To learn more about this technology, see *On-Demand Resources Guide*.
- Bitcode. Archive your app for submission to the App Store in an intermediate representation, which is compiled into 64- or 32-bit executables for the target devices when delivered.

To learn more about app thinning, see App Thinning (iOS, watchOS).

Support for Right-to-Left Languages

iOS 9 brings comprehensive support for right-to-left languages, which makes it easier for you to provide a flipped user interface. For example:

- Standard UIKit controls automatically flip in a right-to-left context.
- `UIView` defines semantic content attributes that let you specify how particular views should appear in a right-to-left context.
- `UIImage` adds the `imageFlippedForRightToLeftLayoutDirection` method, which makes it easy to flip an image programmatically when appropriate.

To learn more about providing a flipped user interface, see Supporting Right-to-Left Languages.

App Transport Security

App Transport Security (ATS) enforces best practices in the secure connections between an app and its back end. ATS prevents accidental disclosure, provides secure default behavior, and is easy to adopt; it is also on by default in iOS 9 and OS X v10.11. You should adopt ATS as soon as possible, regardless of whether you're creating a new app or updating an existing one.

If you're developing a new app, you should use HTTPS exclusively. If you have an existing app, you should use HTTPS as much as you can right now, and create a plan for migrating the rest of your app as soon as possible. In addition, your communication through higher-level APIs needs to be encrypted using TLS version 1.2 with forward secrecy. If you try to make a connection that doesn't follow this requirement, an error is thrown. If your app needs to make a request to an insecure domain, you have to specify this domain in your app's `Info.plist` file.

Extension Points

iOS 9 introduces several new extension points (an extension point defines usage policies and provides APIs to use when you create an app extension for that area). Specifically:

- Network extension points:
  - Use the Packet Tunnel Provider extension point to implement the client side of a custom VPN tunneling protocol.
  - Use the App Proxy Provider extension point to implement the client side of a custom transparent network proxy protocol.
  - Use the Filter Data Provider and the Filter Control Provider extension points to implement dynamic, on-device network content filtering.

  Each of the network extension points requires special permission from Apple.
- Safari extension points:
  - Use the Shared Links extension point to enable users to see your content in Safari's Shared

Links.

    - Use the Content Blocking extension point to give Safari a block list describing the content that you want to block while your users are browsing the web.

- The Index Maintenance extension point to support the reindexing of app data without launching the app.
- The Audio Unit extension point allows your app to provide musical instruments, audio effects, sound generators, and more for use within apps like GarageBand, Logic, and other Audio Unit host apps. The extension point also brings a full audio plug-in model to iOS and lets you sell Audio Units on the App Store.

To learn more about creating app extensions in general, see *App Extension Programming Guide*.

Contacts and Contacts UI

iOS 9 introduces the Contacts and Contacts UI frameworks (`Contacts.framework` and `ContactsUI.framework`), which provide modern object-oriented replacements for the Address Book and Address Book UI frameworks. To learn more, see *Contacts Framework Reference* and *ContactsUI Framework Reference*.

Watch Connectivity

The Watch Connectivity framework (`WatchConnectivity.framework`) provides two-way communication between an iPhone and a paired Apple Watch. Use this framework to coordinate activities between your iOS app and your corresponding Watch app. The framework supports immediate messaging between the apps when they are both running, and background messaging in other cases. To learn more, see *Watch Connectivity Framework Reference*.

Keychain

The keychain provides more item protection options and a new type of encryption keys owned by the secure enclave. Specifically:

- New constraints for access control lists that allow creating constraints with Touch ID only or passcode only.
- A new Touch ID constraint that invalidates keychain items when a fingerprint is added or removed.
- Support for app-provided entropy for keychain item encryption using the Application Password option of the access control list.
- Support for an authentication context that lets you invoke the authentication separately from `SecItem` calls.
- Support for keys generated and used inside the secure enclave using the `kSecAttrTokenIDSecureEnclave` attribute. Note that access to these keys can be controlled by all constraints supported by access control lists.

Swift Enhancements

To learn about what’s new in Swift, see Swift Language.

Additional Framework Changes

In addition to the major changes described above, iOS 9 includes many other improvements.

AV Foundation Framework

The AV Foundation framework (`AVFoundation.framework`) adds new `AVSpeechSynthesisVoice` API that lets you specify a voice by identifier, instead of by language. You can also use the `name` and `quality` properties to get information about a voice.

AVKit Framework

The AVKit framework (`AVKit.framework`) includes the `AVPictureInPictureController` and `AVPlayerViewController` classes, which help you participate in Picture in Picture. For more information about Picture in Picture, see Multitasking Enhancements for iPad.

CloudKit Framework

If you have a CloudKit app, you can use CloudKit web services or CloudKit JS, a JavaScript library, to provide a web interface for users to access the same data as your app. You must have the schema for your databases already created to use a web interface to fetch, create, update, and delete records, zones, and subscriptions. For more information, see *CloudKit JS Reference*, *CloudKit Web Services Reference*, and *CloudKit Catalog: An Introduction to CloudKit (Cocoa and JavaScript)* .

Foundation Framework

The Foundation framework (`Foundation.framework`) includes the following enhancements:

- APIs for on-demand loading of `NSBundle` resources.
- Strings file support for context-dependent variable width strings.
- `NSProcessInfo` APIs for power and thermal management.

HealthKit Framework

The HealthKit framework (`HealthKit.framework`) includes the following enhancements:

- New support for tracking areas such as reproductive health and UV exposure. To learn about the new constants that describe characteristics, quantities, and other items, see *HealthKit Constants Reference*.

- New support for bulk-deleting entries and tracking deleted entries. For more information, see `HKDeletedObject`, `HKAnchoredObjectQuery`, and the `deleteObjects:withCompletion:` and `deleteObjectsOfType:predicate:withCompletion:` methods in *HKHealthStore Class Reference*.

Local Authentication Framework

The Local Authentication framework (`LocalAuthentication.framework`) includes the following enhancements:

- The ability to get a representation of the current set of enrolled fingers so that apps can change behavior when a finger is enrolled or removed.
- Support for canceling a user prompt from code.
- Support for evaluating keychain access control lists and the use of an authentication context in keychain calls.
- Support for reusable Touch ID matches. A match from the previous phone unlock can be used by `evaluateAccessControl:` and `evaluatePolicy:localizedReason:reply:`.

MapKit Framework

The MapKit framework (`MapKit.framework`) includes several additions that help you provide a richer user experience. Specifically:

- MapKit supports querying transit ETAs and launching Maps into transit directions.
- Map views support a 3D flyover mode.
- Annotations can be fully customized.
- Search results for MapKit and `CLGeocoder` can provide a time zone for the result.

PassKit Framework

The PassKit framework (`PassKit.framework`) includes several additions that support enhancements in Apple Pay. Specifically:

- In iOS 9, Apple Pay supports Discover cards and store debit and credit cards. For more information, see "Payment Networks" in *PKPaymentRequest Class Reference*.
- Card issuers and payment networks can add cards to Apple Pay directly in their apps. For more information, see *PKAddPaymentPassViewController Class Reference*.

Safari Services Framework

The Safari Services framework (`SafariServices.framework`) includes the following enhancement.

`SFSafariViewController` can be used to display web content within your app. It shares cookies and other website data with Safari, and has many of Safari's great features, such as Safari AutoFill and Safari Reader. Unlike Safari itself, the `SFSafariViewController` UI is tailored for displaying a single page, featuring a Done button that takes users back to where they were in your app.

If your app displays web content, but does not customize that content, consider replacing your `WKWebView` or `UIWebView`-based browsers with `SFSafariViewController`.

UIKit Framework

The UIKit framework (`UIKit.framework`) includes many enhancements, such as:

- The `UIStackView` class, which helps you manage a set of subviews as a stack that can be arranged vertically or horizontally.
- New layout anchors in `UIView` (such as `leadingAnchor` and `widthAnchor`), `NSLayoutAnchor`, and `NSLayoutDimension`, all of which help make layout easy.
- New layout guides that help you adopt readable content margins and define where within a view the content should draw. For more information, see `UILayoutGuide`.
- A new `UIApplicationDelegate` method you can use to open a document (and modify it) in place, instead of working with a copy of the document. To support the open-in-place functionality, an app also adds to its `Info.plist` file the `LSSupportsOpeningDocumentsInPlace` key with a value of `YES`.
- The `UITextInputAssistantItem` class, which helps you lay out bar button groups in the shortcuts bar.
- Enhancements to touch events, such as the ability to get access to intermediate touches that may have occurred since the last refresh of the display and touch prediction.
- Enhancements to UIKit Dynamics, such as support for nonrectangular collision bounds, the new `UIFieldBehavior` class, which supports various field types in addition to being customizable, and additional attachment types in `UIAttachmentBehavior`.
- The new `behavior` property in `UIUserNotificationAction`, which lets you support text input from users in notifications.
- The new `NSDataAsset` class, which makes it easy to fetch content tailored to the memory and graphics capabilities of your device.
- All standard UIKit controls flip appropriately to support right-to-left languages. In addition, navigation, gestures, collection views, and table cell layouts also flip appropriately.

Deprecated APIs

The following APIs are deprecated:

- The Address Book and Address Book UI frameworks. Use the Contacts and Contacts UI frameworks instead.
- The `NSURLConnection` API in the Foundation framework. Use `NSURLSession` APIs instead.

For a complete list of specific API deprecations, see *iOS 9.0 API Diffs*.