Xcode 8 beta 3 Release Notes

Developer

About Xcode 8 beta 3

Supported Configurations

Xcode 8 beta 3 requires a Mac running macOS version 10.11.5 or later.

Xcode 8 beta 3 includes SDKs for iOS 10.0, watchOS 3.0, macOS version 10.12, and tvOS 10.0. To develop apps targeting prior OS versions, see "About SDKs and the Simulator" in *What's New in Xcode*, available on developer.apple.com or, select Help > What's New in Xcode when running Xcode.

Installation

Xcode 8 beta 3 can coexist with previous versions of Xcode. (Note - running multiple instances of Xcode simultaneously is not supported.)

Prerelease versions of Xcode are made available from developer.apple.com, packaged in a compressed XIP file. To install Xcode during the beta period, download the XIP file, double-click the file to expand it in place, then drag Xcode-beta.app to the Applications folder.

Upon final release, Xcode is installed via the Mac App Store.

Accessing Additional Developer Tools

To launch additional developer tools, such as Instruments and FileMerge, launch Xcode-beta and select Xcode > Open Developer Tool. You can keep these additional tools in your Dock for direct access when Xcode isn't running.

Configuring Xcode Server

macOS version 10.11.6 or later and Server 5.2 beta or later are required to perform continuous integration with Xcode 8 beta 3.

Install macOS, Server, and Xcode 8 beta 3. Next, map Xcode 8 beta 3 to Server by performing the following steps:

1. Launch the Server app and set up for first time
2. In the Services list in the Server sidebar, select the Xcode service.
3. Click the Choose Xcode button, and select Xcode 8 beta 3.
4. Click the On/Off switch to enable Xcode Server.

For additional information, see Xcode Server and Continuous Integration Guide.

Technical Support and Learning Resources

Apple provides the following web resources to support your development with Xcode:

- Apple Developer Forums. Participate in discussions about developing for Apple platforms and using developer tools.
- Bug Reporter. Report issues, enhancement requests, and feedback to Apple. Provide detailed information, including the system and developer tools version information, and any relevant crash logs or console messages.
- Apple Developer website. Get the latest development information as well as technical documentation for Xcode.

- Xcode homepage. Get high-level information about the latest release of Xcode. Download current and beta Xcode releases.

Deprecations and Removal Notices

- macOS 10.11 was the last major release of macOS that supported the previously deprecated garbage collection runtime. Applications or features that depend upon garbage collection may not function properly or will not launch in macOS Sierra. Developers should use Automatic Reference Counting (ARC) or manual retain/release for memory management instead. (20589595)
- `Debugger()` and `DebugStr()` are deprecated and the scheme editor no longer provides the option to enable these functions. If your project uses these functions, enable them by setting the environment variable `USERBREAK` with a value of 1. (24631170)
- The Automation instrument has been removed from Instruments. Use Xcode's UI Testing in its place. (26761665)
- The version of Swift 2 (2.3) used in Xcode 8 is very close to the version used in Xcode 7.3.1. However, it has been updated for the newer SDKs, and therefore is not compatible with Swift frameworks compiled in Xcode 7.3.1. Distributing binary Swift frameworks remains unsupported in Xcode 8. (25680392)

New in Xcode 8 beta 3 – IDE

Interface Builder

- The size inspector now displays frame coordinates with greater precision, and can accept fractional point values on Retina displays for iOS. (8895377)
- The "Notes" field in the Identity inspector now has "Comment For Localizer" placeholder text to more clearly indicate that it is included in XLIFF export. (26802029)
- Xcode 8 removes the ability to configure views without constraints with frame varying per size class. Views without constraints may look different after compiling with Xcode 8 when switching size classes at runtime. Going forward, use constraints and autoresizing masks with the device bar to achieve the desired layout. (25894544)

Build System

- .xcconfig files support conditional inclusion of other .xcconfig files, using the syntax `#include?` instead of the usual `#include` (which still generates a warning if the file is missing, as before). (11003005)

Xcode Server

- You can configure whether upgrade integrations will run when your server is upgraded. (27135245)

SpriteKit

- Xcode 8 contains a project template for a cross-platform SpriteKit game. When you create a project from this template, you can choose which of the 4 platforms should be supported (iOS, macOS, tvOS, watchOS). The resulting project will contain a target for each selected platform and will factor out content that is appropriate to be shared between all platforms. Shared content includes the primary `SKScene` subclass, sks files, and the main asset catalog. Platform-specific content is created for each supported platform including app delegates, view controllers, etc. (26543360)

Resolved in Xcode 8 beta 3 – IDE

General

- An app that doesn't use Swift can now contain a framework or extension that uses a legacy Swift version. (26007454)

Interface Builder

- Constraint constants with fractional (non-integer) values can now be configured in Interface Builder. (15963933)
- Fixed a crash opening documents using a pattern color with `NSPatternColorSpace`. (26964970)

Simulator

- Today extensions load in the iOS 10 simulator on subsequent installs. (26455687)

Debugging

- Xcode Debug Console will correctly show logging from system frameworks when debugging applications in the Simulator. (26652255)

Xcode Server

- Xcode Server email notifications are more legible on iPhone. (26535002)
- Test reports in Xcode and Xcode Server will remember their expanded state. (22002162)
- Addresses an issue where Xcode Server failed to start integrations for a long period of time after an upgrade and Xcode Server Builder would use 100% CPU. (26802372)
- Addresses an issue where integration logs failed to download because of a timeout. (27000597)
- Addresses an issue where failed test counts would not match the number of displayed assertions in an Integration summary report. (27039809)
- Duplicating a bot will not automatically start the first integration. (27047432)
- Addresses an issue where an issue introduced by an Xcode upgrade would also be mistakenly emailed to a committer in that integration. (27092066)

Signing

- Xcode does not require code signing for iOS frameworks, and will skip signing a framework if a team is not set. Frameworks should be signed when copied into the hosting product (using the Code Sign on Copy flag in a Copy Files build phase). (26892618)

Known Issues in Xcode 8 beta 3 – IDE

Source Editor Extensions

- Xcode Source Editor Extensions don't support adding Objective-C categories to XcodeKit classes. (26432410)
- To use Xcode Extensions on macOS version 10.11, launch Xcode and install additional system components. Then, in Terminal, run `sudo /usr/libexec/xpccachectl` and restart your Mac. (26106213)

Project Editor

- The Background Modes Capability "External Accessory" Mode is not present for tvOS projects.

  Workaround: Add the the external accessory key to the Info.plist manually:

  ```
  <key>UIBackgroundModes</key><array><string>external-accessory</string></array>
  ```

  (25016590)

Interface Builder

- Interface Builder does not show correct bar heights (status bar, top bar, bottom bar) when picking landscape for a device in the device configuration bar. (6799670)
- Inspector tooltips in Interface Builder are not available. (26664452)
- Modal segues are currently not working correctly. The presented window may not become visible, or may become visible with missing content. (27096120)

Asset Catalog

- The "Use Asset Catalog…" button in the Project Editor's General tab does not immediately refresh after selecting an asset catalog.

  Workaround: After selecting an asset catalog, navigate away from the General tab and back. (26381374)
- If a `StickerSequence` is added to a sticker pack and left blank, or if frames of different sizes are added, the sticker pack app is blank when deployed. (26301637)

Debugging

- A Top Shelf extension is not automatically triggered when you debug from Xcode.

  Workaround: Use the remote to select a different Top Shelf item and then go back to the one you want to debug. (26131040)
- Xcode sometimes fails to attach to WatchKit notification schemes. (26352924)
- When debugging a watchOS app on a device, Xcode's debugger attaches to the app extension but may not launch the app on the device.

  Workaround: Reboot the Apple Watch and iPhone. (26484132)
- The first time you debug a Today extension on a device, it is not offered as an extension to run.

Workaround: Stop the process, then run and debug it again. (26427188)

- Build and run may not launch or attach to the app on Watch.

  Workaround: Change the watch face. (27019191)

- Siri may not recognize a request to use your Intent app extension upon first launch.

  Workaround: Wait one or two minutes before requesting Siri launch your Intent app extension. (26247696)

- In Organizer > Crashes tab, clicking "Open in Projects…" button crashes Xcode. (27356841)

Gauges

- macOS apps running with thread sanitizer may crash when profiling from the Memory gauge. (26275485)

Instruments

- Some functionality of memory graph debugging is not available for Swift code. (26565791)

Testing

- Keystroke events issued by Xcode UI tests may be dropped if TextExpander is enabled. (25203671)
- UI tests may fail to run for apps written with Swift 2.3. (26680406)

Simulator

- `UIContentSizeCategoryDidChangeNotification` is not delivered as expected in iOS 9.3 or iOS 10.0 Simulator runtimes. (25303105)
- Simulator window is sometimes shown at the wrong scale and touches are at the wrong location. (27138347)

  Workaround: Change the window scale to a different value in the Window->Scale menu, and then change it back to your desired scale again.

- Booting an iOS 10.0 Simulator Device using the `simctl` command line will result in a SpringBoard crash loop cycle.

  Workaround: Use Simulator.app to boot the device. If that is not an option, you can stop the crash loop cycle by requesting that SpringBoard relaunch suspended:

  ```
  simctl spawn booted launchctl debug system/com.apple.SpringBoard --start-suspended
  ```

  (21767449)

- Downloaded simulator runtimes sometimes appear as unavailable immediately after download. (26816587)

  Workaround: Restart CoreSimulatorService by running:

  ```
  sudo killall -9 com.apple.CoreSimulator.CoreSimulatorService
  ```

- Siri APIs are available in the simulator runtimes, but there is no way to trigger Siri from Simulator.

  Workaround: Test usage of the Siri APIs on a real device. (25957692)

Core Data

- For an attribute that allows specification of a custom class in its inspector, such as a transformable attribute, there is no way to specify the module or header from which to import that class. (25669424)
- When using Core Data automatic code generation, changes made to a data model are not immediately visible to other code in the project.

  Workaround: Building the project again updates the generated code. (25789848)
- Generated managed object classes may not be created before the first build.

  Workaround: Perform a clean build. (26564209).

Signing

- The Xcode automatic signing system may fail to sign your app if you add your account or change your password while your project is open.

  Workaround: Close and reopen the project. (25627172)
- Xcode can become unresponsive when switching signing methods, and the new signing settings don't take effect.

  Workaround: Close and reopen the project. (26101127)
- Xcode 8 migrates the deprecated PROVISIONING_PROFILE setting to PROVISIONING_PROFILE_SPECIFIER when opening projects only if the PROVISIONING_PROFILE_SPECIFIER setting is empty. (26281413)
- Downloading profiles using the "Download All Profiles" button intermittently skips some profiles.

  Workaround: Use the developer website to download any missing profiles. (26412212)
- Free provisioning accounts may experience issues launching apps on the watch for the first time.

  Workaround: Launch the app manually on the device and click "Trust" when the security sheet appears. (26629630)
- Messages extensions signed with a personal team may fail to run on a device.

  Workaround: On the device, go to Settings > General > Profiles & Device Management, tap on the profile name that authorized this app, and tap Trust. (26359174)

Help

- Quick Help popovers and the Quick Help inspector display a "Symbol not found" message for user-defined symbols with source code documentation in C or Objective-C. (26470813)
- Search queries in the documentation window containing spaces (such as "Swift Standard Library") do not return API reference results. (26643700)

Xcode Server

- Xcode Server reports an error ("Verifying that Xcode is accessible") when Xcode is installed in a non-world-readable location other than /Applications. Since the service makes use of role accounts to run various service daemons, those role accounts need to be able to read Xcode. (26920334)

  Workaround: Place Xcode in the /Applications folder.

General

- For developers on macOS Sierra it is recommended that you move to Xcode 8. You can continue to develop with the Swift 2.x release with the included Swift 2.3 compiler until you are ready to migrate to Swift 3. Xcode 7.3.1 is not supported on macOS Sierra. (27224895)
- Expanding Xcode beta's .xip requires macOS version 10.11.5 or later. Attempting to expand the archive on earlier versions of macOS will result in an error. (26756225)
- You must unlock devices when first connecting them to Xcode in order to use them for development. (25771635)
- Swift watch apps written prior to this release may fail to launch with a missing dylib or missing symbol error. When migrating to a new version of Swift, Xcode unselects the watchOS app target.

  Workaround: Select the watchOS app target for upgrade and click "Upgrade" in the Swift migrator. (26106054)
- If you store Xcode projects in iCloud Drive we recommended that you disable "Optimize Mac Storage" for iCloud Drive on Macs that you use for your projects. Xcode does not automatically detect iCloud Drive sync conflicts for projects or for files involved in a build. Note that the Documents and Desktop folders may optionally be stored in iCloud Drive on macOS 10.12. (18161353) (26521147)
- On macOS Sierra beta 2, administrator authentication is required in order to install watchOS debugging symbols. If you connect an iPhone with a paired Apple Watch to your Mac, and you haven't already downloaded debugging symbols for the version of watchOS running on your device, Xcode will prompt you for authorization to install Apple-signed software. Your Apple Watch will be unavailable for running or debugging apps until you've successfully authenticated. (27174407)

New in Xcode 8 beta 3 – Swift Compiler

Swift Language

- Function parameters with default arguments must now be specified in declaration order. A call site must always supply the arguments it provides to a function in their declared order:

```
func requiredArguments(a: Int, b: Int, c: Int) {}
func defaultArguments(a: Int = 0, b: Int = 0, c: Int = 0) {}

requiredArguments(a: 0, b: 1, c: 2)
requiredArguments(b: 0, a: 1, c: 2) // error
defaultArguments(a: 0, b: 1, c: 2)
defaultArguments(b: 0, a: 1, c: 2) // error
```

  Arbitrary labeled parameters with default arguments may still be elided, as long as the specified arguments follow declaration order:

```
defaultArguments(a: 0) // ok
defaultArguments(b: 1) // ok
defaultArguments(c: 2) // ok
defaultArguments(a: 1, c: 2) // ok
defaultArguments(b: 1, c: 2) // ok
defaultArguments(c: 1, b: 2) // error
```

  (SE-0060)

- When a Swift program traps at runtime due to an unexpected null value, the error message reports the source location of the "!" operator or implicit unwrapping in the user source code. (26919123)
- Earlier Swift 3 betas implicitly renamed the property, getter, and setter for `@objc` properties of type `Bool` whose names started with "is", to match the Objective-C naming conventions. (For example, a Swift boolean property named `isMagical` generated an Objective-C declaration of the form `@property (nonatomic, getter=isMagical) BOOL magical`.) This turned out to cause more trouble than it was worth and has been removed. (26847223)
- Methods whose names start with `init` are no longer considered to be part of the Objective-C "init" method family. (25759260)
- Condition clauses in `if`, `guard`, and `while` statements now use a more regular syntax. Each pattern or optional binding must be prefixed with case or let respectively, and all conditions are separated by , instead of where.

```
// before
if let a = a, b = b where a == b { }


// after
if let a = a, let b = b, a == b { }
```

  (SE-0099)

Resolved in Xcode 8 beta 3 – Swift Compiler

Swift Compiler

- Changing a private type now causes other files in a target to be recompiled. This is because the private type may be the type of a private property in a non-private type. (SR-1030) (25316242)
- Code coverage support in Swift 2.3 (Legacy) is now available. (26677353)

Known Issues in Xcode 8 beta 3 – Swift Compiler

Swift Compiler

- Attempting to use `NSManagedObjectContext`'s `delete(_:)` method may result in calling the UIKit-added `delete(_:)` method on `NSObject` instead (part of the `UIResponderStandardEditActions` category) if the argument is optional (including `ImplicitlyUnwrappedOptional`). (27206368)

  Workaround: Manually unwrap the optional value using **`if let`** **or** **`!`.**

- If a change to a declaration in one file causes errors, and all of the errors are within function bodies in the same file, other files may not get rebuilt, resulting in possible memory corruption at run-time and other issues. (Note: This only affects incremental builds; it does not affect the Archive action.) (25405605)

  Workaround: Add or remove a top-level function in the file that originally changed to force recompilation. At worst, do a clean build. Removing the build folder is not necessary.

- The generated Objective-C header for a Swift target may include invalid uses of Objective-C generics where another Swift class is used as a generic parameter before it is defined (i.e. before its `@interface`). (27109377)

  Workaround: If you don't need that particular method or property to be exposed to Objective-C, you can mark it as `@nonobjc`.

New in Xcode 8 beta 2 – IDE

Project Editor

- If your CODE_SIGN_ENTITLEMENTS value is conditionalized per-SDK, add a setting for the appropriate simulator SDK(s) explicitly. Previous versions of Xcode fall back to the iOS SDK's value; this is no longer the case. (26923346)

Instruments

- Memory graph debugging is available for WatchKit extensions. (26563396)

Xcode Server

- Queued integrations that are not yet running can be canceled. (13393973)
- When Xcode identifies issues in your code running an Upgrade Integration, Xcode displays the version of Xcode that first detected the issue. (25992006)
- When configuring email notification triggers, you can specify a custom CC and Reply-To address. (25175521, 26050339)
- When integrations are pending, Xcode displays the expected start time and number of integrations ahead of that integration in the queue. When integrations are running, Xcode displays how long the integration has been running. (26374589)
- Bots can be configured to disable application thinning when exporting an installable product. (26452081)

Playgrounds

- Xcode playgrounds support running with open-source Swift toolchains from Swift.org. (26200406)

Resolved in Xcode 8 beta 2 – IDE

General

- Xcode can exist at a location with spaces in the path. (26392825)
- Resolved an issue that caused Xcode 8 beta to crash when opening certain projects. (27068797)
- Xcode provides a cancellation banner if an Xcode Source Editor extension takes too long to return to Xcode. (26560416)
- The "Top Shelf Image Wide" slot is available from Assets.xcassets in template tvOS projects. (26633275)
- Reduced assertion failures from `logd` when running the iOS 10.0 Beta, tvOS 10.0 Beta, or watchOS 3.0 Beta simulators. (25731362)
- SKVideoNode is usable in the iOS and tvOS Simulator Runtimes. (26433285)
- Application responsiveness is significantly improved when a device is attached. If you have a device attached and continue to experience unresponsive behavior while the device window is open, while detaching a device, or while changing the selected scheme, file a bug report via https://bugreport.apple.com. (26060887, 25657432)

Debugging

- Debugging applications in the tvOS Simulator does not affect audio quality. (26602360)
- When debugging an app on the Simulator, logs are visible. (26457535)
- Debugging with address sanitizer does not result in a runtime error about dyld inserting the ASan library. (23805032)
- The view debugger renders `UIVisualEffectViews` properly. (24185948)
- The view debugger functions normally when debugging on the iOS 8 simulator. (25802025)
- `CALayer` transformations do not cause the view debugger to incorrectly space out views in 3D. (26867813)
- Xcode can launch Intent extensions and invoke Siri for debugging. (26431846)

Interface Builder

- A `UIDatePicker` configured in Interface Builder does not have double width at run time. (26511770)
- Xcode records custom font names once per document, rather than once for each use in a document. (18389741)
- XLIFF export no longer includes accessibilityIdentifier strings configured in Interface Builder. (24548071)
- Using a custom font in UILabel/UITextField/UITextView with size classes enabled now correctly encodes the font for use at runtime. (18535469)

Simulator

- `NSUserDefaults` function correctly in the iOS 10 Simulator, regardless of whether you have previously booted the iOS 8 or iOS 9 Simulator. (25974372)
- After erasing a device's settings, the hardware keyboard does not appear to be toggled and the menu state will be correct. (26447487)
- The Simulator's hardware keyboard appears connected on first launch. (26510928)
- Host applications for an extension load once the Simulator has booted. (26283355)
- The Simulator "Authorize Apple Pay" menu item works for iOS. (25786372)
- Image assets with wide-gamut color display properly while running on the iPad Pro (9.7 inch) Simulator. (26235542)
- The Simulator supports copy/paste between the macOS pasteboard and iOS 10.0 pasteboard. (26589804)

Testing

- iOS devices do not lock themselves before running unit tests. (26162579)
- Unit tests do not hang on devices running iOS 9.3.2. (26585821)

Project Editor

- The App Icons Source selection pop-up button in the Project Editor's General tab correctly indicates that `.stickericonset` asset catalog type is eligible. (26381332)

Xcode Server

- Xcode notifies you if integrations are not running because the custom integration user is not currently logged in. (26563837)
- Bots are visible in the Report Navigator if Xcode is not connected to a network at launch time. (26782665)
- Post-integration emails send when emails are configured to only send on certain types of issues. (27006738)

Instruments

- The Points of Interest instrument does not support recording in immediate mode on earlier OS versions. (26400658)
- The Points of Interest instrument is available on macOS, iOS, tvOS, and watchOS. (26376142)

Build System

- Making code changes to a Swift project and building does not result in a binary that fails to run and errors about "dyld: Library not loaded". (26532737)
- The Swift migrator builds the correct targets when workspaces contain multiple targets that build products with the same name for different platforms. (26579521)

Core Data

- Disabling automatic code generation for a specific entity in the Xcode Core Data data model editor persists. (27044493)

Siri

- Unlocalized `AppIntentVocabularly.plist` files work correctly. (26293393)

Source Control

- Xcode can load revisions with the Git color configurations enabled in the local user config. (26871959)
- The file navigator in the commit sheet does not appear collapsed when opening a sheet for the first time. (26962361)
- Xcode does not crash opening certain Interface Builder files with the Version Editor open. (27006872)
- HTTPS repositories added to Xcode do not lose their password after restart. (27031964)
- Switching between modes of the comparison editor behaves normally for SVN repositories. (27043637)
- History for projects containing files with umlaut characters displays correctly. (26861162)

New in Xcode 8 beta 2 – Swift Compiler

Swift Language

- Slice types have a 'base' property that allows public readonly access to their base collections. [SE-0093]
- Nested generic functions may capture bindings from the environment. For example:

  ```
  func outer<T>(t: T) -> T {
    func inner<U>(u: U) -> (T, U) {
      return (t, u)
    }
    return inner(u: (t, t)).0
  }
  ```

  (22051279)
- Initializers are inherited even if the base class or derived class is generic:

  ```
  class Base<T> {
    let t: T

    init(t: T) {
      self.t = t
    }
  }

  class Derived<T> : Base<T> {
    // init(t: T) is now synthesized to call super.init(t: t)
  }
  ```

  (17960407)
- When a Swift program traps at runtime due to an unexpected null value, the error message reports the source location of the "!" operator or implicit unwrapping in the user source code. (26919123)
- Xcode imports Objective-C protocol-qualified classes (e.g. `Class <NSCoding>`) into Swift as protocol type values (`NSCoding.Type`). (15101588)

Resolved in Xcode 8 beta 2 – Swift Compiler

Swift Compiler

- File literals in playgrounds have the URL struct type rather than NSURL. (26561852)

New in Xcode 8 beta – IDE

Source Editor Extensions

- Xcode 8 adds support for Xcode Source Editor Extensions: Application Extensions provide additional commands in the Xcode Editor menu. These extensions can manipulate both text and selections. To create them, use the new Xcode Source Editor Extension target template in the macOS Application Extensions section when creating a project. (23194974)

Interface Builder

- Interface Builder makes it easier to migrate to auto layout when you are ready to do so by no longer generating implicit constraints for views without constraints. (13728545, 24274870)
- Storyboard and xib files support smooth zooming across iOS, tvOS, and watchOS, as well as editing at any zoom level. When using a mouse, operating the scroll wheel while holding down the option key allows you to zoom in and out. (5215027, 9635866, 11579591, 15011038, 21256576, 23379420)
- A completely redesigned workflow for working with trait variations (for example, size classes), supports designing UI in terms of a real device size rather than by using intentionally abstract rectangles. Combined with a new mode for quickly introducing variations, it's easier than ever to see what your UI looks like across multiple devices, orientations, and adaptations like Slide Over and Split View on iPad. (16901225, 23804474, 24012599)
- Creating an `@IBAction` method by control-dragging from Interface Builder to a Swift 3 source file includes an underscore before the "sender" parameter, causing the corresponding Objective-C selector to match Swift 2 behavior. Converting to Swift 3 syntax modifies `@IBAction` methods by adding an underscore, preserving connections in existing IB documents. (26120871)
- The canvas in Interface Builder renders visual interactions between iOS views as they appear at runtime, including accurate compositing of `UIVisualEffectView`. Designable views can visually interact with their subviews. Rendering performance is significantly improved for tvOS documents. (21922006)
- Interface Builder supports customizing UI elements for Dark interface style on tvOS and previewing in the assistant editor. This support is enabled by default for new xib and storyboard files; it can be enabled for existing documents by checking "Uses Trait Variations" in the Identity inspector. (23343849)
- Interface Builder supports colors in the Display P3 color space. In the system color panel, select RGB or HSB sliders, click the gear menu to the right of the popup button, and choose "Display P3" from the list. (23612133)
- Color values in Interface Builder documents correctly use color space during rendering and compilation. Earlier versions of Xcode mishandled color spaces saved in iOS and tvOS documents. Xcode 8 converts existing colors in a way that preserves their perceptual appearance on device, and updates either the color space or component values in the source document as appropriate. (7645087)
- While the canvas is varying for traits (indicated by a blue bottom bar) and has focus, the Delete key uninstalls the selected objects only for the current configuration. If the document outline has focus, or when the canvas is not in varying mode, the Delete key removes the selected objects from the document, affecting all configurations. (26153578)

Runtime Sanitizers

- The new Thread Sanitizer feature combines compiler instrumentation and run time monitoring to help find and understand data races and other concurrency bugs in Swift and Objective-C. Thread Sanitizer is supported on 64-bit macOS and 64-bit simulators. (19432893)
- Address Sanitizer is supported for Swift code. It can catch memory corruption errors that get triggered by using types such as `UnsafeMutablePointer`. (21888114)

Static Analyzer

- The static analyzer check for nullability violations supports both aggressive and less aggressive levels of checking. The more aggressive level checks for nullability violations in all calls. It is enabled by default for new projects. The less aggressive level checks for nullability violations in calls to project headers but not system headers. It is enabled by default for existing projects. You can turn on the aggressive level for existing projects by selecting "Yes (Aggressive)" for "Misuse of 'nonnull'" in the "Static Analyzer - General Issues" section of the target build settings. (25120516)
- The clang static analyzer now checks for improper cleanup of synthesized instance variables in `–dealloc` methods. (6927496)

Playgrounds

- Xcode playgrounds which target macOS now support running with open-source Swift toolchains from Swift.org. Playgrounds which target iOS or tvOS require the Xcode 8 toolchain. (23287417, 26704661)
- The playground video tag supports remote URLs. (24886083)

Build System

- Xcode supports both Swift 3 and Swift 2.3. Swift 2.3 is very similar to Swift 2.2 in Xcode 7.3.1, but has been updated for the newer SDKs, and therefore is not compatible with Swift frameworks compiled in Xcode 7.3.1. Please note that several new features in Xcode 8 are not supported for projects that use Swift 2.3. These include:
  - Memory graph debugging
  - Address Sanitizer
  - Thread Sanitizer
  - Core Data build-time code generation

  To be able to use these features you must migrate your Swift code to Swift 3. (26003920)

  Distributing binary Swift frameworks remains unsupported in Xcode 8, and are not compatible between Swift 2.2 and Swift 2.3 or 3.0. (25680392)
- Xcode takes build settings set in xcconfig files into account when suggesting updates to your build settings. (13433596)
- "Active Compilation Conditions" is a new build setting for passing conditional compilation flags to the Swift compiler. Each element of the value of this setting passes to swiftc prefixed with '-D', in the same way the elements of "Preprocessor Macros" pass to clang with the same prefix. (22457329)
- Two new build settings have been added to enable Swift compiler options: -suppress-warnings (SWIFT_SUPPRESS_WARNINGS) and -warnings-as-errors (SWIFT_TREAT_WARNINGS_AS_ERRORS). These are independent of the build settings for the corresponding clang options. (24213154)

- The PNG compression build settings "Compress PNG Files" and "Remove Text Metadata From PNG Files" are available for macOS in addition to iOS, watchOS, and tvOS. For macOS, they are both off by default. (24638927)
- The build setting ENABLE_ADDRESS_SANITIZER controls whether the address sanitizer is enabled across both the clang and swiftc compilers. Previously only CLANG_USE_ADDRESS_SANITIZER existed; it is subordinate to ENABLE_ADDRESS_SANITIZER and should not be used by itself. If a project was setting CLANG_USE_ADDRESS_SANITIZER as a build setting to enable the address sanitizer separate from the scheme option, then ENABLE_ADDRESS_SANITIZER should be used instead. (24912445)
- The new build setting ALWAYS_EMBED_SWIFT_STANDARD_LIBRARIES replaces the use of EMBEDDED_CONTENT_CONTAINS_SWIFT. This setting indicates that Xcode should always embed Swift standard libraries in a target for which it has been set, whether or not the target contains Swift code. A typical scenario for using this setting is when a target directly uses or embeds another product which contains Swift code.

  Note: EMBEDDED_CONTENT_CONTAINS_SWIFT has been deprecated. (26158130)
- `xcodebuild` supports a `-quiet` flag to suppress output other than errors and warnings. (5732994)

Indexing

- The indexing component moved to an XPC service and out of the Xcode process, providing the benefit of process isolation and accurate resource usage attribution. (24330820)
- The indexing subsystem for unit tests is dramatically faster. It now populates the Test navigator shortly after opening a project for the first time, eliminating the need to wait for indexing to parse all the unit test source files. It supports both Swift and Objective-C unit tests. (25716884)

Source Control

- Includes an updated version editor with improved support for unversioned files, correctly tracking file renames in Git , an updated diff count and stepper to make reviewing diffs easier and a simplified file listing to show changes in both flat and hierarchical views. Xcode Server bots and integrations will now display if the version editor is enabled (22895232, 25098748).
- Full support for unicode filenames in Git (22814717).
- Includes support for Subversion 1.9.
- Improved log and blame modes in the version editor, and faster SCM status for file modifications.

Xcode Server

- Bots can be configured with a custom toolchain (23167923).
- Xcode 8 allows you to customize the user that will run integrations of your project (_xcsbuildd has been removed). (24516345)
- Xcode 8 will now run perform upgrade integrations when you upgrade Xcode on your Xcode Server (an upgrade integration is a re-integration of the last known revision of your projects against the new tools), and can attribute new issues (e.g. deprecation warnings) to that Xcode upgrade. (20918669)
- Improvements to issue tracking and blame in Xcode Server, including more accurate author identification, the ability to track bot configuration changes over time and the ability to attribute new issues to those changes. (22814617)

- Improvements to email notifications, including daily and weekly digest email reports, the ability to set a custom From, CC and Reply-To addresses, optionally include server configuration information, and filter repositories and domains for which emails will be sent.
- Integration summary reports will now show the scheme, branch and duration information for a particular integration.
- Xcode 8 includes a new trigger editor for bots, allowing trigger renaming, reordering and duplication.
- xcscontrol has been improved to support relocating service data to an alternate volume (--relocate and --data-dir), and includes a --graceful-shutdown flag that will stop Xcode Server after the current integration has completed.

AppleScript

- Xcode 8 provides completely rewritten AppleScript support.

  A new scripting dictionary provides the ability to automate Xcode workflows with AppleScript. New functionality includes, for example, the ability to perform "run" and "test" actions. Some previous functionality, such as manipulating files and groups in a workspace, has been curtailed for enhanced reliability.

  Open Script Editor.app and select File > Open Dictionary to access the Xcode scripting library.

  Note: Xcode scripts can be used in Automator workflows with the "run script" command. (22819339)

General

- Terminal can be used for macOS process input and output in addition to the Xcode console. Open the scheme editor to edit your scheme's Run action and then select "Use Terminal" in the Options tab. (23565724)
- Xcode 8 provides numerous accessibility improvements, including VoiceOver enhancements to the design canvas, inspectors, document outline, and object library. (5306382)
- On macOS 10.12 and later, Xcode cleans up stale derived data, precompiled headers, and module caches. (23282174)
- Xcode 8 supports switching toolchains, such as those from swift.org, without relaunching Xcode. (23135507)
- For projects with localizations in more than one language, Xcode automatically turns on the static analyzer check for missing localizability. (22807459)
- Xcode automatically generates classes or class extensions for the entities and properties in a Core Data data model. Automatic code generation is enabled and disabled on an entity by entity basis, and is enabled for all entities in new models using the Xcode 8 file format. This feature is available for any data model that has been upgraded to the Xcode 8 format. You specify whether Xcode generates Swift or Objective-C code for a data model using the data model's file inspector.

  When automatic code generation is enabled for an entity, Xcode creates either a class or class extension for the entity as specified in the entity's inspector: the specified class name is used and the sources are placed in the project's Derived Data. For both Swift and Objective-C, these classes are directly usable from the project's code. For Objective-C, an additional header file is created for all generated entities in your model: The file name conforms to the naming convention "DataModelName+CoreDataModel.h".

  The Xcode attribute inspector also allows a choice of whether to generate scalar or object properties. New attributes default to generating scalar properties where appropriate.

For attributes that previously relied on filling in class information after code was manually generated (for example, transformable attributes and transient attributes with an undefined attribute type), specify the class to use in the attribute's declaration directly in the attribute inspector. (22223273)

- A new target type for Metal libraries enables creating new projects, or new targets in existing projects, for platforms that support Metal. Targets of this type only compile Metal source files and produce Metal libraries. A Metal library produced in this manner can be added to the Copy Bundle Resources build phase of another target to make the Metal content available to that target's products. (17877026)
- Xcode 8 introduces a new signing system.

  In Xcode 8, the signing system has been rewritten to include a new mode for automatically managing signing assets in addition to a dedicated manual mode where the profiles for the target must be explicitly selected. When automatically managing signing assets, Xcode will create signing certificates, update app IDs, and create provisioning profiles. For manual mode, only custom created profiles can be selected and Xcode will not modify or create any signing assets..

  Xcode now encodes profiles in the target using the PROVISIONING_PROFILE_SPECIFIER build setting. This setting allows specifying both the team ID and the name or identifier of the profile.

Resolved Issues in Xcode 8 beta – IDE

Interface Builder

- Storyboard and xib files can be scrolled while dragging objects or scenes, and during control-dragging to create connections. Dragging to the edge of the canvas autoscrolls. (25221374, 25631320)
- Interface Builder's design canvas draws correctly when layer-backing is enabled in macOS xib and storyboard documents. This includes canvas decorations such as resize knobs, selection indicators, bounds rectangles, constraints, and segue arrows. (5565950)

Playgrounds

- The first link included in a playground markup comment does not have its hit target reduced to the very start of the linked text. (24920637)

Debugging

- Resolved stability issues with debugging extensions or watch apps in the simulator. (25338888)
- The view debugger shows the debug description for views in the inspector. (23976841)
- The view debugger shows the frame for views in the size inspector. (24314615)
- Resolved an issue where View Debugging could fail when not using auto layout, due to an unhandled assertion in an underlying system framework. (25311044)

General

- Changes made to projects from outside Xcode (for instance, from git) do not cause Xcode to select a different active scheme. (16762297)

New in Xcode 8 beta – Swift and Apple LLVM Compilers

Swift Language

- Xcode 8 includes a prerelease of both Swift 3 and Swift 2.3. Unless otherwise specified, these release notes apply to Swift 3. The "SE-XXXX" numbers refer to swift-evolution proposals, which you can read more details about at this web site: https://github.com/apple/swift-evolution. If you have existing code, please use the Swift Migration workflow to update your code to the latest Swift syntax, by using the menu option "Edit -> Convert -> To Latest Swift Syntax …". This will handle the majority of changes necessary to update your code from Swift 2.2 to Swift 2.3 or Swift 3.0 syntax, and is also a great way to update your code between beta releases of Xcode 8.
- The Swift Objective-C importer and standard library have been improved to conform to the new API Guidelines (https://swift.org/documentation/api-design-guidelines/) in order to provide clear and consistent APIs. [SE-0023] [SE-0005] [SE-0006]
- The `swift_name` attribute has been added to allow C API authors to specify how their APIs should be imported into Swift. [SE-0044]
- Given a list of constants in an Objective-C file, added an attribute that will enable Swift to import them as either an `Enum` or a `Struct`, using `RawRepresentable` to convert to their original type. [SE-0033]
- The behavior of implicitly-unwrapped optionals has changed. The `!` syntax (e.g. `NSObject!`) is now only allowed directly on the type of a variable, constant, parameter, or result. An implicitly-unwrapped optional value can still be used immediately as a value of its non-optional type, but generic features such as type inference will consider it to have an ordinary optional type. [SE-0054]

  `For example:`

  ```
    var x: Int! = nil   // x is an implicitly-unwrapped optional value
  of type Int?
    var y = x           // y is an ordinary optional value of type Int?
    var z = x ?? 0      // x will be treated as an optional; since it
  is nil, z will be bound to 0
    print(x + 0)        // x will be implicitly unwrapped, causing a
  trap
  ```

- Function parameters have consistent labelling across all function parameters. With this update the first parameter declarations match the existing behavior of the second and later parameters. This change makes the language simpler. [SE-0046]

  For example, functions that were previously written and called as follows:

  ```
  func foo(x: Int, y: Int) {}
  foo(1, y: 2)
  func bar(a a: Int, b: Int) {}
  bar(a: 3, b: 4)
  ```

  should be written as:

  ```
  func foo(_ x: Int, y: Int) {}
  foo(1, y: 2)
  func bar(a: Int, b: Int) {}
  bar(a: 3, b: 4)
  ```

in order to achieve the same behavior.

- The compiler warns about unused results by default. [SE-0047]
- Currying func declaration syntax has been removed. [SE-0002]
- ++ and -- operators have been removed. [SE-0004]
- C-style for loop has been removed. [SE-0007]
- Implicit tuple expansion in calls has been removed. [SE-0029]
- Most language keywords are now usable in member references without backticks. For example, you can now write `foo.default` if `foo` has a member with that name. [SE-0071]
- The `Ref` forms of CF type names have been removed from Swift. For example, use `CFString` instead of `CFStringRef`. One exception is if there are types named both `Foo` and `FooRef`, where `FooRef` is a Core Foundation type. In this case, both types continue to be imported as they always have. (16888940)
- New `#file`, `#sourceLocation`, `#column`, and `#function` expressions have been introduced to replace the existing `__FILE__`, `__LINE__`, `__COLUMN__`, and `__FUNCTION__` symbols. The `__FILE__`-style symbols have been removed. [SE-0028] [SE-0034]
- The order of defaulted parameters at the call site is always required to match a function's declared parameter order. [SE-0060]
- Function type syntax has been standardized to always require parentheses around the parameter list. [SE-0066]
- Objective-C lightweight generic classes are imported as generic types in Swift. Because Objective-C generics are not represented at runtime, there are some limitations on what can be done with them in Swift:
  - If an ObjC generic class is used in a checked `` `as?` ``, `` `as!` ``, or `` `is` `` cast, the generic parameters are not checked at runtime. The cast succeeds if the operand is an instance of the ObjC class, regardless of parameters.

```
let x = NSFoo<NSNumber>(value: NSNumber(integer: 0))
let y: AnyObject = x
let z = y as! NSFoo<NSString> // Succeeds
```

  - Swift subclasses can only inherit an ObjC generic class if its generic parameters are fully specified.

```
// Error: Can't inherit ObjC generic class with unbound parameter T
class SwiftFoo1<T>: NSFoo<T> { }

// OK: Can inherit ObjC generic class with specific parameters
class SwiftFoo2<T>: NSFoo<NSString> { }
```

  - Swift can extend ObjC generic classes, but the extensions cannot be constrained, and definitions inside the extension do not have access to the class's generic parameters.

```
extension NSFoo {
// Error: Can't access generic param T
func foo() -> T {
      return T()
}
}

// Error: extension can't be constrained
extension NSFoo where T: NSString {
}
```

- Foundation container classes NS[Mutable]Array, NS[Mutable]Set, and NS[Mutable]Dictionary are still imported as nongeneric classes for the time being.

[SE-0057]

- The types UnsafePointer, UnsafeMutablePointer, AutoreleasingUnsafeMutablePointer, OpaquePointer, Selector, and NSZone represent non-nullable pointers, that is, pointers that are never nil. A nullable pointer is represented using Optional, for example, UnsafePointer<Int>?. For types imported from C, non-object pointers (such as int *) have their nullability taken into account.
  - A pointer marked as _Nonnull, or within an NS_ASSUME_NONNULL_BEGIN/_END block, is imported as a non-optional value. Example: NSInteger * _Nonnull is imported as UnsafeMutablePointer<Int>.
  - A pointer marked as _Nullable is imported as an optional value. Example: NSInteger * _Nullable is imported as UnsafeMutablePointer<Int>?.
  - A pointer marked as _Null_unspecified, or an unannotated pointer outside of any NS_ASSUME_NONNULL_BEGIN/_END block, is imported as an implicitly-unwrapped optional value. Example: NSInteger * _Null_unspecified is imported as UnsafeMutablePointer<Int>!.

  The Swift standard library has been adjusted to match.

  One possible area of difficulty is passing a nullable pointer to a function that uses C variadics. Swift will not permit this directly, so as a workaround please use the following idiom to pass it as a pointer-sized integer value instead:

```
Int(bitPattern: nullablePointer)
```

  [SE-0055] (15189170)

- Comments are treated as whitespace when determining whether an operator is prefix, postfix, or binary. For example, these expressions work:

```
if /*comment*/!foo { ... }
1 +/*comment*/2
```

  This also means that comments can no longer appear between a unary operator and its argument.

```
foo/* comment */! // no longer works
```

Note: Any parse errors resulting from this change can be resolved by moving the comment outside of the expression. [SE-0037]

- The location of the `inout` attribute has been moved to after the `:` and before the parameter type. [SE-0031]

  For example:

```
func foo(inout x: Int) {
}
```

is written as:

```
func foo(x: inout Int) {
}
```

- The `@noescape` and `@autoclosure` attributes must now be written before the parameter type instead of before the parameter name. [SE-0049]
- `let` is no longer accepted as a parameter attribute for functions. The compiler provides a fixit to remove it from the function declaration. [SE-0053]
- `var` is no longer accepted as a parameter attribute for functions. The compiler provides a fixit to create a shadow copy in the function body.

```
func foo(var x: Int) {
}
```

  is written as:

```
func foo(x: Int) {
  var x = x
}
```

  [SE-0003]

- The `none` member of an imported `NS_OPTIONS` option set is marked as unavailable when it is imported. Use `[]` to make an empty option set, instead of a `none` member.
- The ability to declare variables in multiple patterns in cases has been added. [SE-0043]
- Attributes now use `:` instead of = to introduce their attribute arguments. This aligns their syntax with function call syntax. [SE-0040]
- Generic typealiases are supported, for example:

```
typealias StringDictionary<T> = Dictionary<String, T>
typealias IntFunction<T> = (T) -> Int
typealias MatchingTriple<T> = (T, T, T)
typealias BackwardTriple<T1, T2, T3> = (T3, T2, T1)
```

  [SE-0048]

- The `@noescape` attribute has been extended to be a more general type attribute. You can declare values of `@noescape` function type, e.g. in manually curried function signatures. You can now also declare local variables of `@noescape` type, and can use `@noescape` with the `typealias` keyword.

  For example, this is valid code:

```
func apply<T, U>(f: @noescape T -> U,
                g: @noescape (@noescape T -> U) -> U) -> U {
  return g(f)
}
```

- Generic signatures can now contain superclass requirements with generic parameter types.

- `catch` blocks in `rethrows` functions may now throw errors. For example:

```
 func process(f: () throws -> Int) rethrows -> Int {
   do {
     return try f()
   } catch is SomeError {
throw OtherError()
   }
 }
```

- Throwing closure arguments of a rethrowing function may be optional. For example:

```
func executeClosureIfNotNil(closure: (() throws -> Void)?) rethrows {
  try closure?()
}
```

- The Objective-C selectors for the getter or setter of a property can be referenced with `#selector`. For example:

```
let sel1 = #selector(getter: UIView.backgroundColor) // sel1 has type
Selector
let sel2 = #selector(setter: UIView.backgroundColor) // sel2 has type
Selector
```

  [SE-0064]

- A key-path can be formed with `#keyPath`. For example:

```
person.valueForKeyPath(#keyPath(Person.bestFriend.lastName))
```

  [SE-0062]

- The "domain" property of an `NSError` containing an error thrown from Swift includes the module name of the type, matching the constant shown in the generated header. [SR-700] (24999546)

Swift Standard Library

- The standard library have been improved to conform to the new API Guidelines in order to provide clear and consistent APIs. [SE-0023] [SE-0005] [SE-0006]
- There is a new collections indexing model, in which collections move indexes, instead of indexes moving themselves. [SE-0065]
- The Floating Point protocols have been enhanced. [SE-0067]
- A lazy `flatMap` has been added for sequences of optionals. [SE-0008]
- Conversions from `Unsafe[Mutable]Pointer` to `Int` and `UInt` are now supported. [SE-0016]
- In Swift 2.2 the `Unmanaged` type had a static method `fromOpaque(_:)` and an instance method `toOpaque()`, which converted the unmanaged reference from and to the `COpaquePointer` type. In Swift 3 these have been changed to convert from an `UnsafePointer<Void>` and to an `UnsafeMutablePointer<Void>` to match the common use of being passed as the "context pointer" for a C API. In most cases, you will be able to simply remove uses of `COpaquePointer` (now renamed to `OpaquePointer`). [SE-0017] (26611811)
- The `first(where:)` method has been added to `Sequence`. [SE-0032]
- Added generic result and error handling to `autoreleasepool()`. [SE-0061]
- Failable numeric conversion initializers have been added. [SE-0080]
- `sequence(first:next:)` and `sequence(state:next:)` have been added. [SE-0094]

clang

- Incremental Link-Time Optimization (LTO) has been added and is a major redesign of the existing Monolithic LTO mode. It addresses the three main limitations of Monolithic LTO by making it parallel, memory lean, and incremental. (22252706)
- C++ now supports the `thread_local` keyword, which declares thread-local storage (TLS) and supports C++ classes with non-trivial constructors and destructors. (9001553)
- Objective-C now supports class properties, which interoperate with Swift type properties. They are declared as: `@property (class) NSString *someStringProperty;`. They are never synthesized. (23891898)

General

- The Swift Package Manager tool for managing the distribution of Swift code is available on the command line. Help for the Swift Package Manager can be found by running 'swift package --help' in Terminal. (25582703)