Xcode 10 beta Release Notes

Developer

About Xcode 10 beta

Supported Configurations

Xcode 10 beta requires a Mac running macOS 10.13.4 or later.

SDK Versions

Xcode 10 beta includes SDKs for iOS 12, watchOS 5, macOS 10.14, and tvOS 12.

Installation

Xcode 10 beta can coexist with previous versions of Xcode.

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. To use command line tools from the beta Xcode, run `xcode-select -s <path to Xcode>`.

The final release of Xcode 10 will be available in the Mac App Store. Previous versions of Xcode are available from developer.apple.com/downloads/more.

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.

Technical Support and Learning Resources

Apple provides the following 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.
- 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.
- For help with using Xcode, use the built-in help by choosing Help > Xcode Help.

Deprecation Notices

- The macOS 10.14 SDK no longer contains support for compiling 32-bit applications. If developers need to compile for i386, Xcode 9.4 or earlier is required.
- Support for Subversion has been removed.
- Custom dtrace-based instrumentation that was available for macOS and Simulator only is no longer supported in Instruments. The new Custom Instruments packages based on os_signpost allow for greater flexibility and control over presentation of data and support all platforms.

Command Line Tools

The Command Line Tools package installs the macOS system headers inside the macOS SDK. Software that compiles with the installed tools will search for macOS headers at the standard include path:

```
/Applications/Xcode-beta.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.14.sdk/usr/include
```

For legacy software that looks for the macOS headers in the base system under /usr/include, please install the package file located at:

```
/Library/Developer/CommandLineTools/Packages/macOS_SDK_headers_for_macOS_10.14.pkg
```

To make sure that you are using the intended version of the command line tools, run `xcode-select -s <path to Xcode>` after installing.

Release Notes Updates

Release notes are occasionally updated after a beta is distributed. The latest version can be found on the developer.apple.com/download.

Revision: XC100B1 - RNV1

New in Xcode 10 beta – IDE

General

- Library content has moved from the bottom of the Inspector area to an overlay window, which can be moved and resized like Spotlight search. It dismisses once items are dragged, but holding the Option key before dragging will keep the library open for an additional drag.

  The library can be opened via a new toolbar button, the View > Libraries menu, or the ⇧⌘L keyboard shortcut. Content dynamically matches the active editor, so the same UI provides access to code snippets, Interface Builder, SpriteKit, or SceneKit items. The media library is available via a long press on the toolbar button, the View > Libraries menu, or the ⇧⌘M keyboard shortcut. (37318979, 39885726)

- Custom code snippets can now be added to the library via the Editor > Create Code Snippet menu item. (37946810)

- Newly-created schemes are now shared by all users of an Xcode project. To create a personal scheme, uncheck the “Shared” checkbox in the “Manage Schemes” sheet. (40223696)

- The Capabilities tab in the Project Editor provides a new Hardened Runtime capability for macOS apps and app extensions. Enabling this capability will opt your app into new security protections provided by macOS 10.14 and will be required for your app to be notarized. (39674498)

- Select Schemes and Run Destinations from the keyboard. Press “Ctrl+0” to open the Scheme popup and “Ctrl+Shift+0” to open the Run Destination popup. Once the popup appears, type enough characters to highlight the appropriate entry, use the arrow keys to highlight it, and press return to select it. (8999215)

Source Editor

- The Xcode Source Editor now supports multi-cursor editing allowing you to quickly edit multiple ranges of code at once.

  You can place additional cursors with the mouse via ^+⇧+Click or with column select (⌥+Click+Drag), or with the keyboard using ^+⇧+Up to column select up or ^+⇧+Down to column select down. (12564506)

- Xcode has increased support for code folding, including:

  1. A new code folding ribbon showing all of the multi-line foldable blocks of code in the editor
  2. A new style for folded code in the editor that allows you to edit lines with folded code
  3. Support for folding any block of code enclosed in curly braces
  4. Support for folding blocks of code from the folding ribbon, from structured selection, or from the Editor > Code Folding > Fold menu item

  To turn on the code folding ribbon, open preferences under Text Editing > Editing and select "Code folding ribbon". (33518606, 34203382, 35391767, 35932817, 36554006)

- All instances of the current symbol can be selected in the editor with the Editor > Structure > Select All Symbols command. (35064345)

- With a source control-enabled project the source editor displays changes made by a developer in the gutter and shows changes made by other developers that haven't yet been pulled into the project. (9794871)

- Find results in the source editor can now be selected using the Find > Select All Find Matches menu item. The results to select can be constrained to those in the current selection using the menu item Find > Select Find Matches in Selection. (35064581, 39283557)

- The source editor will now automatically detect and use the predominant line ending style in a file. (35343242)

- Selected text ranges in the source editor can now be split into multiple selected text ranges by line using the Editor > Structure > Split Selection By Lines menu item. (39300660)

- The source editor now supports configurable overscroll at the end of the file. The overscroll amount can be configured in Preferences > Text Editing > Editor Overscroll. (9075043)

- Updated default source editor themes for light and dark, with improved colors optimized for contrast, and taking advantage of bold and italic font traits. (40036785)

Build System

- Xcode 10 uses a new build system. The new build system provides improved reliability and build performance, and it catches project configuration problems that the legacy build system does not.

  Although the new build system is highly compatible with existing projects, some projects may require changes due to the following:

  - The new build system has stricter checks for cycles between elements in the build in order to prevent unnecessary rebuilds.
  - It is an error for any individual file in the build to be produced by more than one build command. For example, if two targets each declare the same output file from a shell script phase, factor out the declaration of the output file into a single target.
  - If an output file which is generated by a shell script is used as an input elsewhere in the build (for example, to another shell script), then that output must be declared as an explicit output by the script that generates it; otherwise the build system may attempt to search for the file before it has been generated, causing the build to fail.
  - The legacy header map that was generated when the Always Search User Paths (`ALWAYS_SEARCH_USER_PATHS`) setting was `YES` is not supported by the new build system. Instead, set `ALWAYS_SEARCH_USER_PATHS` to `NO` and migrate to using modern header include syntax. Add any needed header files that are in the project repository to the Xcode project to ensure they are available for use in #include (via the project wide header map). Use quote-style include (`"foo.h"`) for project headers, and reserve angle-bracket include (`<foo.h>`) for system headers.
  - The new build system passes `undefined_arch` as the value for the ARCH environment variable when running shell script build phases. The value was previously not well defined. Any shell scripts depending on this value must behave correctly for all defined architectures being built, available via the ARCHS environment variable.
  - The new build system uses the “clean build folder” behavior. The legacy “clean” behavior is not supported.

  More details on resolving issues which the new build system detects can be found in Xcode Help.

  If required, the legacy build system is still available in Xcode 10. To use the legacy build system, select it in the File > Project/Workspace Settings sheet. Projects configured to use the legacy build system will display an orange hammer icon in the Activity View. (35267155)

- Xcode provides improved warnings and errors for dependency cycles and other build issues detected by Xcode’s new build system. (37370377)

- While a target is performing post-compilation build steps, such as linking and code signing, Xcode's new build system will begin compiling sources from targets that depend on that target, to improve build speed by using more of your available processor cores. This compilation will not start until any "Run Script" phases from the target have completed. (38772619)

- Run Script build phases now support declaring input and output files in a `.xcfilelist` file. This file should contain a newline-seperated list of the file paths for the inputs or outputs. Build setting references may be used in these paths. The path to a copy of the `xcfilelist` file with resolved build setting references will be provided in the `SCRIPT_INPUT_FILE_LIST_(#)` and `SCRIPT_OUTPUT_FILE_LIST_(#)` environment variables of the Run Script phase. Projects which use an `.xcfilelist` require Xcode 10. (39241255)

- A new Product > Perform Action > Build With Timing Summary command produces a build log with statistics about the aggregate time taken by each command in the build. These timings aid in analyzing the time taken by the build. (40069871)

- When an `.xcconfig` file contains multiple assignments of the same build setting, later assignments using using `$(inherited)` or `$(<setting_name>)` will inherit from earlier assignments in the `.xcconfig`. The legacy build system caused every use of `$(inherited)` or `$(<setting_name>)` skip any other values defined within the `.xcconfig`. To detect whether your `.xcconfig` is affected by this improvement, running `defaults write com.apple.dt.XCBuild EnableCompatibilityWarningsForXCBuildTransition -bool YES` in Terminal will cause Xcode to generate a warning about this situation. (40283621)

Testing

- Test bundles can now be configured to execute their contents in a random order each time the tests are run. This can be enabled in the Test pane of the scheme editor as an option for each test bundle. (11719679)

- Xcode 10 supports running tests in parallel, which reduces the time it takes to run tests. Test parallelization is supported for macOS unit tests, as well as unit and UI tests on iOS and tvOS simulators. To enable parallelization, navigate to the scheme editor (Product > Scheme > Edit Scheme), select the Test action followed by the Info tab, and then next to your test target, click Options. Finally, select "Execute in parallel" (for macOS tests) or "Execute in parallel on Simulator" (for iOS and tvOS tests).

  Test parallelization occurs by distributing the test classes in a target across multiple runner processes. Use the test log to see how your test classes were parallelized. You will see an entry in the log for each runner process that was launched, and below each runner you will see the list of classes that it executed.

  When testing in parallel on Simulator, each runner process executes on a separate clone of the selected simulator. For a simulator named "iPhone X", these clones will appear in Simulator as "Clone 1 of iPhone X", "Clone 2 of iPhone X", etc. (35224733)

- `xcodebuild` has new command line options to control the behavior of parallel testing. Use `-parallel-testing-enabled` to override the per-target setting in the scheme for whether parallelization is enabled. If you want to control the number of runners that are launched, use `-parallel-testing-worker-count` or `-maximum-parallel-testing-workers`. (39648990)

- XCTest UI tests now capture the standard output and error streams from launched target apps to a file in the result bundle. (30929875)

- Schemes can now be configured to only run a fixed subset of tests in a test bundles and not automatically include new tests that are added to the bundle. This can be configured in the Test pane of the scheme editor as an option for each test bundle. (35050431)

Source Control

- Integration with Bitbucket Cloud and Bitbucket Server source control. (31156776)
- Integration with GitLab.com and GitLab self-hosted source control. (37501192)
- SSH keys can be easily managed using the new create, validate and upload workflows with GitHub, Bitbucket, and GitLab. (31798220)
- Improved source control authentication workflows provide more information and greater control to perform an authenticated operation. (33726987)
- Rebase support when pulling source control changes. (8937399)
- Tags can optionally be pushed from the push sheet. (40141815)

Interface Builder

- Canvas rendering now happens in parallel, and scales based on demand and hardware to improve the performance of edit operations, particularly for large scenes. (36999759)
- The “Stack” button in the canvas bar has been replaced with a pop-up menu containing all embedding options for the selection. (38201831)
- Some binaries used by Interface Builder have been renamed for better consistency. Instead of “Interface Builder Cocoa Touch Tool”, “Interface Builder WatchKit Tool”, and “IBAppleTVTool”, they are now called “IBAgent-[platform]”. (35400833)
- Opening a .nib or .xib file last saved in a format prior to Xcode 5 now prompts for confirmation, and automatically upgrades the document to a modern format. (37691996)
- The Editor > Embed In menu item allows embedding in a tightly-wrapped view that doesn’t add any margins to the selected content. (11515741)
- Resizing a `UIStackView` or `NSStackView` along its distribution axis in the canvas will adjust spacing between items. (21166308)
- The menu for choosing a font family in the Attributes inspector now renders a preview of each font. (37157540)
- Support for Large Title font text style (`UIFontTextStyleLargeTitle`) on iOS 11 or later, and on watchOS 5. (33150633)
- Image and color inspector properties that reference an asset catalog resource have a navigation button to jump to that resource. Option + click will show the resource in the Assistant Editor. (39661055)
- Controls using named colors from an asset catalog now update as the value of the color changes. (37613468)
- Inspector color pickers now use a standard pop-up button. The color swatch can be dragged by holding the Option key. (38582636)
- In the Identity inspector, the localization comment property has been changed to a plain string. It is still stored in the document as an attributed string to maintain compatibility with earlier versions of Xcode. (39856810)

macOS SDK

- Support for `NSGridView`, which manages Auto Layout constraints for its content in a two-dimensional table layout. `NSGridView` is backward deployable to macOS 10.12, or macOS 10.13.4 when using merged cells. (23707607)
- On macOS 10.14, the Device Bar and Preview assistant editor allow designing for and previewing in Light Appearance and Dark Appearance. (38673229, 38940117)
- Support for new `NSVisualEffectView` materials introduced in macOS 10.14. (37688179)
- Designable `NSView` subclasses are rendered with the appropriate `NSAppearance` for their use in the canvas. (39021773)
- `NSTextView` has multiple entries in the object library. This separates configurations appropriate for user interface elements that should adapt to the system appearance, and document content which has a fixed appearance that does not change with the system setting, including for print content. (39022169)
- The default colors for `NSTextField` and `NSTextView` have been changed. The new values are still appropriate for deployment on macOS 10.13 and earlier. (39577751)
- When a Mac XIB or storyboard is opened in Xcode 10, `NSTextView` that have default colors will be changed to use new recommended colors. The text color and insertion point color change from black to `NSColor.textColor`, and the background color changes from white to `NSColor.textBackgroundColor`. These should appear the same on earlier releases while also adapting to appearances on macOS 10.14. Editable NSTextField instances have their text color changed from `NSColor.textColor` to `NSColor.controlTextColor`. These changes only apply when opening and saving the document from within Xcode. (40280823)
- On macOS 10.14, `NSButton` and `NSImage` support content color tinting in the Attributes inspector. Tinting only applies to borderless buttons and template images. (39957802)
- On macOS 10.14, `NSToolbarItem` can center itself in an `NSToolbar` by checking “Is Centered Item” in the Attributes inspector. (38639941)
- An `NSToolbarItem` containing an image or custom view has an Automatic size option in the Size inspector. Use this option to automatically manage the minimum and maximum width and height of the toolbar item, for example when the item contains a segmented control. (38810182)
- Support for setting accessibility attributes on `NSWindow`. (9929792)

watchOS SDK

- Support for the new Dynamic Interactive notification interfaces. (36801793)
- Support for the new “Now Playing” and “Volume” interface objects. (36824666, 36824736)
- Notification categories now have a “Handles Grouping” property, which allows notifications to be coalesced. (38503942)
- Static notification interfaces now support connecting a subtitle label outlet. (39536566)

iOS SDK

- In iOS 12, the default value of the `UITableView` property `cellLayoutMarginsFollowReadableWidth` is now false. This means by default, table view cells will no longer have their layout margins increased automatically when the table view is wider than the maximum readable width.

  If you are using a table view to display text that spans multiple lines, in order to improve readability when the table view is very wide you should opt-in to these automatic margins by setting `cellLayoutMarginsFollowReadableWidth` to true, or consider adapting your layout to better take advantage of the additional horizontal space. (36828516)

tvOS SDK

- New `UILabel` inspector property “Marquee on Ancestor Focus” for controlling scrolling behavior. (31222028)

Asset Catalog

- Support for varying image and color assets by Light, Dark, and High Contrast appearances on macOS 10.14 and above. (38735965)
- Support for CarPlay assets. (35543584)
- Support for ARKit 3D `ARReferenceObject` assets. (38086640)
- The background of the asset catalog and view debugger can be set explicitly to light or dark so foreground elements display with sufficient contrast. (39073926)

Debugging

- The files table in the Disk Gauge Report includes files that were closed so that developers are aware of their size. (39359014)
- The Disk Gauge Report includes a new table that shows the total Read and Write sizes of the process across time. (39359147)
- The debugging Attach sheet allows developers to turn on Malloc Stack Logging (Live Allocations). (35447219)
- Named colors shown in the inspector while view debugging now indicate their names and whether they are system colors. (38193961)
- The view debugger adds `appearance` and `effectiveAppearance` properties to the `NSView` inspector. (38198002)
- Xcode's view debugger adds an option to choose between light and dark canvas background color. (39159102)
- You can change the appearance of your macOS app at runtime by using the Debug > View Debugging > Appearance menu, the Override Appearance menu in the debug bar, or the touch bar. (39448599)
- Metal Frame Debugger (33194071)

  Xcode 10's Metal Frame Debugger lets you debug your Metal shaders. Capture a Metal workload, select a draw / dispatch call and click the new debug button on the debug bar to start debugging your shaders.

  Selecting command encoder items in Debug Navigator will bring up a graph showing how encoders depend on each other through resource usage. Zoom in to get more information about attachments and encoders.

  Every draw call has new "Geometry" item where you can see 3D visualization of your post transform vertex data alongside with your vertex inputs and outputs.

Playgrounds

- Xcode playgrounds now execute code incrementally, compiling new lines of code when you hit shift-return or click the Run button next to the new line of code. This is especially useful for long tasks that you don't want to run repeatedly, like training a machine learning model or setting up your live view's state, and allows you to progressively iterate on your ideas without restarting the playground. (34313149)

Localization

- Xcode will offer to enable Base Internationalization for projects that do not yet use it but support multiple localizations. (13178091)

- Xcode 10 supports a new Xcode Localization Catalog (xcloc) export/import format for localization data that can contain both an XLIFF file and other localizable content such as image files. (28662326)

Metal

- When calling the `metal` compiler directly you must use the `-c` flag when building a `.metal` file. Not adding the option will result in an error when the resulting `.air` file is consumed by `metallib`.

  Calling the `metal` compiler via Xcode's build system will add the flag automatically. (40655432)

Signing and Distribution

- The Developer ID distribution option in Xcode's Organizer now provides support for uploading apps to Apple to be notarized. After building an archive, this option can be selected in the Organizer by clicking the Distribute App button and then selecting the Developer ID method and the Upload destination. In order to upload an app to be notarized, you must enter an Apple ID in Xcode's Accounts preferences pane with the necessary App Store Connect role and provider membership. In addition, apps uploaded to be notarized must be signed with a Developer ID certificate. The distribution workflow can create this certificate if necessary, but requires an Apple ID account with the Agent role in order to do so.

  After uploading an app to be notarized, you can view your app's status in the Organizer window by selecting your archive and clicking the Show Status Log button. When you receive notification that your app has been notarized, you can export it from the Organizer window by selecting your archive and clicking the Export App button. The exported app contains a stapled ticket and is ready for distribution. (36409604)

- Support for uploading apps to Apple via the command line. The `xcodebuild -exportArchive` command will perform an upload if the provided `ExportOptions.plist` contains a key named "destination" with value "upload". In addition, an Apple ID account with the necessary App Store Connect role and provider membership must be added in Xcode's Accounts preference pane. The "app-store", "developer-id", and "validation" distribution methods are supported for use from `xcodebuild`. (28555930)

Instruments

- You can now build and distribute your own custom instruments. (37987666)
- The support for system trace instrumentation prior to Instruments 8 has been removed. Some trace files containing data from these instruments will no longer open in Instruments. A version of Xcode prior to Xcode 10 can be downloaded from developer.apple.com in order to view these older files. (38506479)
- There is a new graph mode available for threads when using System Trace instrumentation in Instruments 10 – Thread States – which shows only the thread state lane. The previous OS Fundamentals graph style (system calls, VM faults, and thread state lanes) is still available as well. (37727022)

Documentation Viewer

- Quick Help has been updated to use a single column layout, making better use of available space that it is presented in. (39518057)
- Code listings in the documentation viewer and Quick Help match the user's currently selected theme colors. (39435799)

Resolved in Xcode 10 beta – IDE

General

- The New File template for Objective-C header (.h) files includes `NS_ASSUME_NONNULL_BEGIN/END` macros. (22753521)
- Project settings validation understands build setting values set by reference to other build settings. (30549576)
- Double Click Navigation now defaults to Same as Click. This can be changed in the Navigation pane of Preferences. (37294346)
- Schemes for which “Show” has been unchecked in the Manage Schemes sheet no longer appear in the Product > Scheme submenu. (24579413)

Source Editor

- Improved the performance of many editing operations, including Undo and Redo. (30874294)
- Fixed a problem where the "Automatically balance brackets in Objective-C method calls" preference was not respected. (34195824)
- Live issues will now be cleared when renaming a file. This resolves an issue where live issues may remain after a file has been renamed and edited. (36186038)
- Fixed a problem that would cause wide content in the "Jump to Definition" popover to be clipped. (39060697)
- Resolved a crash when editing a file using Classic Mac OS (CR) or Windows (CRLF) line endings. (39137414)
- Resolved an issue where attempting to present the Action menu immediately after selecting the next find result may fail. (39884637)
- Fixed a problem with code completion using code snippets starting with @ character, such as @autoreleasepool. (31768452)
- The source editor now provides more accurate code completion when typing initializers in Swift. (38265505)
- Enhanced reliability of cmd+click jump to definition, now showing results more reliably in situations with syntax errors or other errors in the code. (34905255)
- Resolved an issue where breakpoints may appear at the incorrect line in folded code. (37288192)
- Fixed crash of image literal quick editor. (38281672)
- Fixed problems where the "Insert Newline without Extra Action" and "Insert Newline and Leave Selection Before It" commands didn't work correctly. (38477237)
- Improved the scrolling estimation for large files. (39344677)
- Fixed a problem with occasional hang/spin during code completion. (40088929)
- Fixed an issue where comments in XML files that contained MARK: or TODO: would cause syntax coloring of the rest of the file to fail. (12051726)

Testing

- Xcode and xcodebuild now build all targets in the active scheme when clicking a test gem in the Test navigator or source editor (Xcode), or when using `-only-testing: (xcodebuild)`. Previously, only the relevant test bundle target and its dependencies would be built. Xcode and xcodebuild will now also disallow you from running a test or test class whose test bundle is not part of the active (Xcode) or specified (xcodebuild) scheme. (38935442)

- Errors that may lead to missing or no coverage data are now displayed in the test log. If you notice any such errors, please file a bug at https://bugreport.apple.com, and attach a zipped archive of your project's folder in DerivedData. (39930570)

Source Control

- The source control authors editor submode performance is improved to load and scroll faster. (40179372)

Interface Builder

- Typing delete (⌫) when an edge is selected in the constraint filter of the Size inspector now removes the applicable constraints, rather than the view itself. (25373426)

- Corrected an issue where previews and drag images did not always render at the correct scale factor if multiple displays with different scale factors were connected. (38775753)

- Fixed an issue where connect-to-source mistakenly allowed adding code to Swift generated interfaces. (35263074)

- When a default color is selected in the inspector, the name of the color is shown in addition to the text "Default". (39663381)

- Improved performance compiling documents, particularly those containing multiple `NSComboBox` controls. (30468986)

Asset Catalog

- Editing fractional color values in an asset catalog works correctly for non-English locales. (39666878)

Debugging

- Options in the Debug > View Debugging menu, like "Show View Frames", now continue the target application as expected. (26649378)
- The Memory Gauge Report now reports the physical footprint consistently across all the platforms. (20472364)

Playgrounds

- A crash that sometimes occurred when creating a new playground page, especially if the playground was locked, has been fixed. (38281509)
- Code completion is now supported in auxiliary source files in playgrounds. (34363732)
- A common crash that occurred in playgrounds with inline results has been fixed. (38281379)
- Resolved an issue that prevented the iOS Game playground template from working correctly. (38828600)

Localization

- In an XLIFF produced by the Export For Localization command, content from a file outside the project directory will be referenced via a path relative to the project directory rather than an absolute path. (38680116)

Instruments

- Resolved an issue that would cause most templates to fail when profiling on iOS versions prior to iOS 11.3. (39759266)

Simulator

- Simulator now supports a "Point Accurate" snap-to scale factor in addition to the existing "Physical Size" and "Pixel Accurate" options. (38232400)
- "Pixel Accurate" scaling is now correct for tvOS Simulator devices. (36341177)
- CarPlay is now supported in the iOS Simulator. (37416934)
- Copy/Paste in Simulator's Edit menu is no longer used for synchronization with a simulator devices's pasteboard. The Edit menu now has explicit menu items to handle these actions. (38225290)
- Rendering is now always done at full resolution. As a result, screenshots are always at screen size. Simulator's "Optimize Rendering for Window Scale" option has been removed. (38232185)

Known Issues in Xcode 10 beta – IDE

Build System

- When a target that contains Swift code adds a new dependency on a Swift system library, Xcode will not copy that runtime library into the app until the next full build. This will cause a runtime error such as "dyld: Library not loaded: @rpath/libswiftos.dylib". (40456595)

  Workaround: After adding code which requires another Swift system library, perform the Clean Build Folder action before building.

- `xcodebuild clean` does not delete all build products. (40460666)

  Workaround: Remove the build folder for the project manually.

- The new build system does not yet support On Demand Resources. If your project uses ODR, the legacy build system can be reenabled in File > Workspace/Project Settings. (31508570)

- When performing `xcodebuild clean` on a project which uses a customized build location outside the derived data directory, and which has older build products produced prior to Xcode 10, Xcode may report an error indicating that it will not delete directories not created by the new build system. (40427159)

  Workaround: Delete the build folder manually.

- When build settings are defined in an `xcconfig` file that inherit values using `$(inherited)` or `$(<setting_name>)`, the Build Settings editor shows the last assigned value for the setting instead of the properly composed value. (28572636)

- The "Clean Build Folder" command will not clear errors and warnings. (39659924)

  Workaround: Fix the errors and warnings and rebuild. Alternatively, close the workspace and delete the workspace's derived data directory inside `~/Library/Xcode/DerivedData`.

- The command timings listed in Xcode's build log will list times for each Swift subtask which may be inaccurate when using Multi-File Mode compilation. (40424339)

- After installing a Swift toolchain, Xcode will not immediately detect the toolchain. (31360319)

  Workaround: Close and re-open the workspace.

- Custom build system plugins are not yet supported by Xcode's new build system. (32795438)

  Workaround: If required, switch to using the legacy build system in the File > Project/Workspace Settings sheet.

- The Intel ICC compiler is not yet supported by Xcode's new build system. (33286594)

  Workaround: If required, switch to the legacy build system in the File > Project/Workspace Settings sheet.

- The new "Build with Timing Summary" command does not show timings for some tasks run within the build system during the build. (39801746)

- Targets which have multiple asset catalogs that aren't in the same build phase may produce an error regarding a "duplicate output file". (39810274)

  Workaround: Ensure that all asset catalogs are processed by the same build phase in the target.

- The `Clean Build Folder` action is disabled for projects which use Custom or Legacy build locations. (40108679)

- Xcode's new build system does not support build setting expansions in the advanced build location options found in Location Preferences or Project Settings. (40236969)

- watchOS targets may fail to install due to misconfigured project settings. Please check the VALID_ARCHS for watchOS targets to ensure they are correct. (40531052)

- Previously broken input references may cause the new build system to generate spurious "Build input file cannot be found" errors on subsequent builds. (40667135)

Testing

- Running macOS UI tests on macOS 10.14 from Xcode may result in a password prompt being shown for allowing the debugger to attach to the test runner. (40530732)

  Workaround: Enter your password to allow the debugger to attach, or uncheck the Debug Executable checkbox in the Test action pane of the scheme editor.

- When tests are run on an iOS or tvOS simulator, Simulator may hang after the OS has finished booting. (40147579)

  Workaround: Quit and relaunch Simulator, and testing should resume normally.

- When configuring destinations for an Xcode Server bot, selecting "Run tests in parallel" causes the selected destinations to run the scheme's tests simultaneously. It does not cause an individual destination to run tests in parallel. To allow each destination to run tests in parallel, deselect this checkbox and either enable parallel testing in the scheme or add `-parallel-testing-enabled YES` as an xcodebuild argument. (39985444)

Source Control

- A pull or merge will pause file operations for all projects, not just the current project. (39994122)
- Source control accounts and "Add Documentation" may not appear or be enabled or available in Xcode. Restarting macOS should re-enable these features. (40704433)

Interface Builder

- On macOS 10.13, attempting to drag an `NSTextView` object from the library to the canvas will crash Xcode. (40529925)
- When deploying apps to iOS 10 or earlier, a constraint between a UIScrollView's layout margin and one of its descendant views will cause a crash at runtime. (40540860)
- Color attributes in the user-defined runtime attributes table cannot be edited. (40035743)

  Workaround: Declare an `IBInspectable` property for the color key path, and edit the value in the Attributes inspector.
- Hovering over a watchOS notification category in the document outline will crash. The row has no icon or text, and displaying the tooltip will result in a crash. (40561230)

  Workaround: Select the category using arrow keys in the document outline, or using the Jump Bar.

Asset Catalog

- Dragging an `ARReferenceObject` to the empty region in the Asset catalog does not import the asset. (40379198)

  Workaround: Create an AR Resource group and drag the `ARReferenceObject` into that group.

- When importing an `ARReferenceObject` into the asset catalog, the preview image is not shown. (39959348)

  Workaround: Close and reopen the asset catalog.

- Compiling an asset catalog with an empty `ARReferenceObject` slot will fail. (40382246)

  Workaround: Either remove the empty `ARReferenceObject` slot or populate the empty slot.

- When deploying apps to iOS 11 or earlier, certain image assets are not included in compiled car file. (40507731)

- Auto scaling is turned on by default in asset catalogs for watchOS PDF assets, regardless of selection. (35788204)

- When both Universal and Mac dark slots are defined for a color, the asset catalog compiler fails to compile the catalog. (39506690)

  Workaround: Only specalize the color for either Mac or Universal idioms, but not both.

- Building for Dark Appearance is not supported on macOS 10.13. (40624707)

Debugging

- In the GPU Frame Debugger, when debugging shaders of offline-compiled libraries with included sources and multiple Xcode installations, users will need to switch to their active Xcode version by using the `xcode-select` command in Terminal. (40340429)

- When view debugging a SceneKit scene the inspectors may not display the properties for the selected scene node. (39801376)

- In the sidebar of the Shader Debugger, the button for accessing the detail view may be positioned so it cannot be pressed. (40665537)

  Workaround: The sidebar width can be increased, or hovering over the variable in source code will display the same information.

- The GPU Frame Debugger's Geometry Viewer does not support issue detection in Xcode 10 beta. (40224469)

Playgrounds

- macOS Single View playgrounds do not execute. (40533054)

  Workaround: Replace the line:

  ```
  let nibFile = NSNib.Name(rawValue:"MyView")
  ```

  with:

  ```
  let nibFile = NSNib.Name("MyView")
  ```

- Playgrounds can sometimes fail to run with the “couldn’t lookup symbols” error message. (38505726)

  Workaround: Rerunning the playground will usually fix this issue.

Create ML

- After renaming an image classifier in the UI and then dragging out a mlmodel does not always use the specified name. (40535700)

  Workaround: Click “Save” in the save options view first to set the name.

- Evaluation won’t start when dragging in a folder with fewer than half of the number of trained classes. (40558219)

  Workaround: Make sure the evaluation folder has the number of sub-folders greater than or equal to half of the number of the trained classes.

- Create ML is not supported for use in Command Line Tools. (40637430)

SpriteKit

- The playback bar for the SpriteKit Scene Editor and Action Editor may sometimes have a black background instead of white. (40706702)

SiriKit Intent Editor

- You can not set a key image for an intent with no parameters. (40395951)

  Workaround: Add a parameter and provide an image for the parameter.

Devices

- The Open Console button in the Devices and Simulators window may fail to open the Console app with a timeout error. (39955550)

  Workaround: Open the Console app from the /Applications/Uitilities folder.

- Unpairing a network device from Xcode may not disable network development between the device and your Mac. (39226136)

  Workaround: Uncheck the “Connect via Network” checkbox on the device before unpairing the device.

- Running a WatchKit App with Xcode 10 that was built with a previous version of Xcode can give an installation error “The WatchKit app has an invalid stub executable.” (40567857)

  Workaround: Clean the build folder and Run again.

Server

- Certain Xcode Server logs, such as trigger logs, are not visible in the integration Logs report. (40462372)

  Workaround: Click “Download Logs” in the integration Logs report and the resulting archive includes all of the logs produced by the integration.

Signing and Distribution

- `xcodebuild -exportNotarizedApp` may crash when checking the status of an archive. (40625436)

  Workaround: Export your notarized app using Xcode. In the Organizer window, select your archive and then click the "Show Archive Status" button in the sidebar to update the status of your archive. If your app has been notarized and is ready for distribution, then click the "Export App" button in the Organizer sidebar.

- Ad-Hoc signed apps containing Swift code will crash on launch in macOS 10.14 if the Hardened Runtime capability is enabled. (40571112)

  Workaround: Enable development signing using a Mac Developer cert, then clean and rebuild the app.

- Developer ID distribution may fail in the re-signing stage with the error "signature size too large to embed". This can occur if you've enabled the Hardened Runtime capability in Xcode. (40665475)

  Workaround: Disable the Hardened Runtime capability, rebuild your archive, and then try distributing the app again.

Instruments

- os_signpost and os_log recording misses most data if recording from a device that has rebooted recently. (40654502)

  Workaround: macOS devices are affected for the first 18min 20sec after boot, after which time recording should work properly again. iOS devices may be affected for up to the first ~9.5 hours after boot.

- Custom Instruments based on os_signpost and os_log instrumentation do not work in the Simulator. (40661069)

- The event-type field of os-signpost point events in Instruments incorrectly appears as "Emit" instead of "Event". (40591556)

  Workaround: Match against (event-type "Emit"|"Event")

- Profiling unit tests with no host application fails for the iOS simulator. (39334812)

  Workaround: Configure the unit test target to have a host application.

Simulator

- iOS 9.2 and earlier simulator runtimes will not function correctly on macOS 10.14 Beta. (40335891)
- When Xcode tries installing an improperly constructed app to the Simulator, Xcode shows the error "This app could not be installed at this time." (40505189)
- Shortcuts are not available on Simulator. (40380495, 40380729)

  Workaround: Test on a device.
- When WatchApp is run in a Simulator that isn't currently running, it will fail with "Couldn't communicate with a helper application." (39116182)

  Workaround: Try again in a moment or use the menu Xcode > Open Developer Tools > Simulator to observe when the Simulators have finished launching and then run again.
- iOS 12's user interface unexpectedly disappears when activating an external display with an iOS Simulator including a CarPlay window, or when switching external displays. (39728482)
- The OS can take several minutes to boot for the first time in a Simulator. (40535421)
- On iOS 12, News will crash shortly after launch. (40383020)

Documentation Viewer

- On macOS 10.14 beta, light font colors are used when printing documentation viewer content, leading to text having low contrast between it and the background. (40274975)

New in Xcode 10 beta – Apple LLVM and Swift Compilers

Apple LLVM Compiler

- The static analyzer is more efficient and will report additional issues on most programs. (36672459)

- The static analyzer checks for a common performance anti-pattern when using Grand Central Dispatch, which involves waiting on a callback using a semaphore:

```
+ (NSString *)requestCurrentTaskName {
    __block NSString *taskName = nil;
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    NSXPCConnection *connection =
        [[NSXPCConnection alloc] initWithServiceName:@"MyConnection"];
    id remoteObjectProxy = connection.remoteObjectProxy;
    [remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
        taskName = task;
        dispatch_semaphore_signal(sema);
    }];
    dispatch_semaphore_wait(
        sema,
        dispatch_time(DISPATCH_TIME_NOW, 100)
    );
    return taskName;
}
```

  Such a pattern can degrade performance and cause hangs in your application. The check is currently disabled by default, but can be enabled using the build setting "Performance Anti-Patterns with Grand Central Dispatch". (37312818)

Swift Language

- The new `CaseIterable` protocol describes types which have a static `allCases` property that is used to describe all of the cases of the type. Swift will synthesize this `allCases` property for enums that have no associated values. (SE–0194) (17102392). For example:

```
enum Suit: CaseIterable {
    case heart
    case club
    case diamond
    case spade
}

print(Suit.allCases)
// prints [Suit.heart, Suit.club, Suit.diamond, Suit.spade]
```

- As part of fully implementing (SE–0054), `ImplicitlyUnwrappedOptional<T>` is now an unavailable typealias of `Optional<T>`. Declarations annotated with '`!`' have the type `Optional<T>`. If an expression involving one of these values will not compile successfully with the type `Optional<T>`, it is implicitly unwrapped, producing a value of type `T`. In some cases this will cause code that previously compiled to require updating. Please see this blog post for more information: (Reimplementation of Implicitly Unwrapped Optionals). (33272674)

- Public classes may now have internal `required` initializers. The rule for `required` initializers is that they must be available everywhere the class can be subclassed, but previously `required` initializers on public classes were forced to be public themselves. (This limitation was a holdover from before the introduction of the open/public distinction in Swift 3.) (22845087)

- The standard library now uses a high-quality, randomly seeded, universal hash function, represented by the new public `Hasher` struct. "Random seeding" varies the result of `hashValue` on each execution of a Swift program, improving the reliability of the standard library's hashed collections such as `Set` and `Dictionary`. In particular, random seeding enables better protection against (accidental or deliberate) hash-flooding attacks.

  This change fulfills a long-standing prophecy in `Hashable`'s documentation: *Hash values are not guaranteed to be equal across different executions of your program. Do not save hash values to use during a future execution.*

  As a consequence of random seeding, the elements in `Set` and `Dictionary` values may have a different order on each execution. This may expose some bugs in existing code that accidentally relies on repeatable ordering.

  Additionally, the `Hashable` protocol now includes an extra function requirement, `hash(into:)`. The new requirement is designed to be much easier to implement than the old `hashValue` property, and it generally provides better hashing. To implement `hash(into:)`, simply feed the exact same components of your type that you compare in `Equatable`'s `==` implementation to the supplied `Hasher`:

```
struct Foo: Hashable {
    var a: String?
    var b: [Int]
    var c: [String: Int]

    static func ==(lhs: Foo, rhs: Foo) -> Bool {
        return lhs.a == rhs.a && lhs.b == rhs.b && lhs.c == rhs.c
    }

    func hash(into hasher: inout Hasher) {
        hasher.combine(a)
        hasher.combine(b)
        hasher.combine(c)
    }
}
```

Automatic synthesis for `Hashable` (SE–0185) has been updated to generate `hash(into:)` implementations. For example, the `==` and `hash(into:)` implementations above are equivalent to the ones synthesized by the compiler, and can be removed without changing the meaning of the code.

Synthesis has also been extended to support deriving `hashValue` from `hash(into:)`, and vice versa. Therefore, code that only implements `hashValue` continues to work in Swift 4.2. This new compiler functionality works for all types that can implement `Hashable`, including classes.

Note that these changes don't affect Foundation's hashing interface. Classes that subclass `NSObject` should override the `hash` property, like before.

In certain controlled environments, such as while running particular tests, it may be helpful to selectively disable hash seed randomization, so that hash values and the order of elements in `Set`/`Dictionary` values remain consistent across executions. You can disable hash seed randomization by defining the environment variable `SWIFT_DETERMINISTIC_HASHING` with the value of `1`. The Swift runtime looks at this variable during process startup and, if it is defined, replaces the random seed with a constant value. (SE–0206) (35052153)

- The standard library now has a Random generator API. The Swift standard library now supports generating and using random numbers. Swift collections now support `shuffle`/`shuffled` operations as well as choosing a `randomElement()`. Numeric types can now generate a uniform value in a range e.g. `Int.random(in: 1...10)` or `Double.random(in: 0..<100)`, and `Bool` provides a random coin-flip: `Bool.random()`.

  A `RandomNumberGenerator` protocol has been introduced as a source of randomness to drive these methods, with a default implementation for each platform. On Apple platforms, this implementation wraps `arc4random_buf`. (40564129)

- Custom compile-time warnings or error messages can be emitted using the `#warning(_:)` and `#error(_:)` directives (SE–0196). (16015824)

```
#warning("this is incomplete")

#if BUILD_CONFIG && OTHER_BUILD_CONFIG
#error("BUILD_CONFIG and OTHER_BUILD_CONFIG can't both be set")
#endif
```

Swift Compiler

- The Swift compiler defaults to a new compilation strategy that can greatly speed up debug builds. This strategy allows each compilation job to process a batch of files instead of just a single file.

  Projects that had sped up debug builds by opting into building with Whole Module Optimization in Debug mode (at `-Onone`) should try the new compilation style. To do so, choose the Incremental option for the Compilation Mode build setting and ensure that your project is not using older whole-module settings for Debug builds such as `SWIFT_WHOLE_MODULE_OPTIMIZATION=YES` or `SWIFT_OPTIMIZATION_LEVEL=-Owholemodule`.

  The new compilation strategy should be transparent in terms of the build products. However, in case it causes a problem, you can revert to the previous single file strategy with a custom Build Setting `SWIFT_ENABLE_BATCH_MODE=NO`. (39253613)

- C macros containing casts are no longer imported to Swift if the type in the cast is unavailable or deprecated, or produces some other diagnostic when referenced. These macros were already only imported under very limited circumstances with very simple values, so this is unlikely to affect real-world code. (36528212)

- Swift 4.1 warnings about 'overlapping accesses' are now errors in Swift 4 mode. These new errors most commonly arise when a generic mutating method that modifies a variable is passed a non-escaping closure that reads from the same variable. Code that previously compiled with a warning will need to be updated. Please see this forum post for more information: (Upgrading exclusive access warning to be an error in Swift 4.2). (34669400)

- Various function-like declarations can now be marked as `@inlinable`, making their bodies available for optimizations from other modules. Inlinable function bodies must only reference public declarations, unless the referenced declaration is marked as `@usableFromInline` (SE-0193). (40566899)

  Note that the presence of the attribute itself does not force inlining or any other optimization to be performed, nor does it have any effect on optimizations performed within a single module.

- The C `long double` type is now imported as `Float80` on i386 and x86_64 macOS and Linux. The tgmath functions in the Darwin and glibc modules now support `Float80` as well as `Float` and `Double`. Several tgmath functions have been made generic over `[Binary]FloatingPoint` so that they will automatically be available for any conforming type. (SR-2046) (27196587)

Swift Standard Library

- `Range` is now conditionally a `RandomAccessCollection`. Through conditional conformance, the `Range` type is now conditionally a `RandomAccessCollection` when its `Bound` is `Strideable` with a `SignedInteger` stride. This means that you can now use use `Range` as a collection when these criteria are met. (40564272)

  `CountableRange` is now a constrained generic type alias for a `Range`, rather than an independent type. This means that you can still use `CountableRange` in the same ways as before: extend it, take it as an argument to a function, or declare a variable of that type. But you can also now use ranges and countable ranges interchangeably:

  ```
  let countableRange: CountableRange<Int> = 0..<10
  var range: Range = countableRange

  func f<T: SignedInteger>(_ countableRange: CountableRange<T>) { }
  f(range)
  ```

- The standard library types `Optional`, `Array`, `ArraySlice`, `ContiguousArray`, `Dictionary`, `DictionaryLiteral`, `Range`, and `ClosedRange` now conform to the `Hashable` protocol when their element or bound types (as the case may be) conform to `Hashable`. This makes synthesized `Hashable` implementations available for types that include stored properties of these types (SE0143). (40567748)

- The behavior of the `description` and `debugDescription` properties for floating-point numbers use a new algorithm that is both more accurate and significantly faster than the previous code. In particular, we now guarantee that `Double(String(d)) == d` for every Double value `d`. Previously these unconditionally printed a fixed number of decimal digits (e.g. 15 and 17 for Double, respectively). They now print exactly as many digits as are needed for the resulting string to convert back to the original source value, and no more (SR–106). (40565639)

Swift Package Manager

- As per SE–0209, `swiftLanguageVersions` property in `PackageDescription` manifest API for tools version 4.2 is changed from an array of Integers to an array of `SwiftVersion` enum. (38721967)

Example usage:

```swift
// swift-tools-version:4.2
import PackageDescription

let package = Package(
    ...
    swiftLanguageVersions: [.v3, .v4]
)
```

Resolved in Xcode 10 beta – Swift Compiler

Swift Compiler

- When a property satisfies a requirement in an `@objc` protocol, the selectors for its getter and setter are in all circumstances now inferred from the protocol requirement. (37363245)
- When switching over an `@objc` enum, if the value is not one of the recognized cases and the switch does not have a default case, the program will trap at run time. (20420436)
- The compiler now correctly handles when a nested type in a `Codable` type has the same name as a property of the `Codable` type (SR–6569). (37570349)
- Protocol conformances are now able to be synthesized in extensions in the same file as the type definition, allowing automatic synthesis of conditional conformances to `Hashable`, `Equatable` and `Codable` (both `Encodable` and `Decodable`). For instance, if there is a generic wrapper type that can only be `Equatable` when its wrapped type is also `Equatable`, the `==` method can be automatically constructed by the compiler:

```
struct Generic<Param> {
    var property: Param
}

extension Generic: Equatable where Param: Equatable {}
// Automatically synthesized inside the extension:
// static func ==(lhs: Generic, rhs: Generic) -> Bool {
//   return lhs.property == rhs.property
// }
```

  Code that wants to be as precise as possible should generally not conditionally conform to `Codable` directly, but rather its two constituent protocols `Encodable` and `Decodable`, or else one can only (for instance) decode a `Generic<Param>` if `Param` is `Encodable` in addition to `Decodable`, even though `Encodable` is likely not required:

```
// Unnecessarily restrictive:
extension Generic: Codable where Param: Codable {}

// More precise:
extension Generic: Encodable where Param: Encodable {}
extension Generic: Decodable where Param: Decodable {}
```

  Finally, due to `Decodable` having an `init` requirement, it is not possible to conform to `Decodable` in an extension of a non-final class: such a class needs to have any `init`s from protocols be `required`, which means they need to be in the class definition. (SR–6803) (39199726)

- Runtime query of conditional conformances is now implemented. Therefore, a dynamic cast such as `value as? P`, where the dynamic type of `value` conditionally conforms to `P`, will succeed when the conditional requirements are met. (SE–0143) (40349058)

```
protocol P {}

struct DoesntConform {}
struct Conforms: P {}

struct Generic<Param> {}
extension Generic: P where Param: P {}

print(Generic<DoesntConform>() as? P as Any) // nil
print(Generic<Conforms>() as? P as Any)
// Optional(main.Generic<main.Conforms>())
```

- When a Swift method is exposed to Objective-C with a custom selector beginning with "new", "copy", or "mutableCopy", it will correctly return an owned (+1) value rather than an autoreleased (+0) one when called from Objective-C, as expected by Objective-C ARC. (SR–6065) (34834291)

- Properties in the header generated by the Swift compiler are now correctly annotated with OS availability information. (37090774)

- A conditional conformance does not imply conformances to any inherited protocols, unlike unconditional ones. This is because the optimal conditional requirements are likely to be different between the two protocols if the inherited protocol can use less restrictive requirements (SR–6569). For instance:

```
protocol Base {}
protocol Sub: Base {}
struct Generic<Param> {}

extension Generic: Base where Param: Base {}
extension Generic: Sub where Param: Sub {}
```

  The first extension has to exist, because the second one does not imply a conformance of `Generic` to `Base` (if it did, it would imply it with the same conditional requirements as the `Sub` conformance, that is, `Param: Sub`). This particularly affects hierarchies like `Hashable: Equatable`, and `BidirectionalCollection: Collection`, and helps achieve precision in the bounds of types conforming to them. (36499373)

Known Issues in Xcode 10 beta – Apple LLVM and Swift Compilers

Apple LLVM Compiler

- When importing some frameworks, the Swift REPL will fail with symbol lookup errors. (40458863)

- The debugger cannot display the values of very small resilient types with inline storage. (39722386)

Swift Compiler

- The Foundation API `Bundle.init(for: AnyClass)` always returns the application bundle when used on a class that inherits from a generic class, even if that class itself is not generic. (40367300)

- `type(of: object)` may produce an unexpected result if `object` is being observed by KVO. For instance, `Bundle(for: type(of: object))` may unexpectedly fail. (37319860)

  Workaround: Get the class of the object with `object.value(forKey: "class")`.

- The compiler might emit code that references the type metadata for a private or internal type that should not be accessible, e.g., because it's from another module or (in non-WMO mode) source file. Such problems manifest as build failures and can be identified by a reference to a "type metadata accessor" symbol, whose mangled name has the suffix "Ma". Specifically, you will see a failure coming from a compiler invocation that looks like this:

  ```
  Global is external, but doesn't have external or weak linkage!
  %swift.metadata_response (i64)*
      @"$S6Second8Property33_241EE7586AF57F5A142D57A4DBC25F8ALLOMa"
  <unknown>:0: error: fatal error encountered during compilation;
      please file a bug report with your project and the crash log
  ```

  Note that the mangled symbol in the error message ends with suffix "Ma". (40229755)

  Workaround: To work around this problem, add a `OTHER_SWIFT_FLAGS` build setting to your project and pass `-emit-public-type-metadata-accessors` compiler flag to it. This compiler flag will be phased out once the problem is fixed.

- Swift 4.2 gates important enhancements to the SDK, with some of those enhancements landing in a later beta. While all the new APIs and language features are available without migration, developers must migrate to incorporate the enhancements. Migrating early will identify issues

that can be corrected in later betas, but will forego automatic migration of those later SDK changes. (39498127)

Workaround: Wait until a later beta to try migration or re-run the migrator with non-migrated sources.

- Assigning between properties of a value of protocol type may raise incorrect inout exclusivity errors in some circumstances. (39524104) For example:

```
protocol P {
    var x: Int { get set }
    var y: Int { get set }
}

func foo(x: inout P) {
    x.x = x.y
}
```

Workaround: Break up the left and right side of the assignment using a temporary variable:

```
func foo(x: inout P) {
    let sadTrombone = x.x
    x.y = sadTrombone
}
```

- The Swift REPL shipped in the Command Line Tools package will fail to import any Objective-C frameworks, or Swift frameworks depending on Objective-C frameworks. For example `import Darwin` will fail with a missing header error. (40537961)