Documentation

Article

Xcode 26 Beta 3 Release Notes

Update your apps to use new features, and test your apps against API changes.

Overview

Xcode 26 beta 3 includes SDKs for iOS 26, iPadOS 26, tvOS 26, watchOS 26, macOS Tahoe 26, and visionOS 26. The Xcode 26 beta 3 release supports on-device debugging in iOS 16 and later, tvOS 16 and later, watchOS 8 and later, and visionOS. Xcode 26 beta 3 requires a Mac running macOS Sequoia 15.4 or later.

General

Resolved Issues

- Fixed: Added support for backtick-escaped raw identifiers in Swift. (140952488)

Known Issues

- The `span` property of `UTF8View` does not support the small string representation in beta 1, and traps for small `String` instances. A future version of the Swift standard library will lift

this restriction. (137710901)

Workaround: As a workaround, reserve capacity for 16 UTF8 code units, which forces the `String` to use another representation.

```
var string = "small"
// _ = string.utf8.span     // this would trap!
string.reserveCapacity(16)
let span = string.utf8.span // no problem
assert(span[2] == 97)
```

- Simulators may fail to boot during the first build after upgrading macOS; rebuilding after a short wait typically resolves the issue. (152328794)
- Sometimes the “Cancel” button in the Coding Assistant status bar fails to stop the currently executing message. (152546155)

  Workaround: Use the Coding Assistant navigator directly, and click “Cancel”.

App Intents

Resolved Issues

- Fixed: Conforming to `TargetContentProvidingIntent` and `OpenIntent` generates a compiler error about ambiguous inference of `PerformResult`. (152099549)

Known Issues

- UISceneAppIntent.performNavigation(forScene:) not always called. (152254774)

Apple TV

Known Issues

- Apple TV might fail to authorize and attempt to launch a properly signed app that is being run immediately after installation from Xcode, with a spurious error indicating that the Developer App certificate needs to be trusted on the device. (154812309)

  Workaround: First, verify in Xcode's Signing and Capabilities editor that the app has been configured properly to be signed to run on the connected Apple TV. After excluding any configuration issues, you can work around this spurious error by editing your scheme's Run action to "Wait for the executable to be launched". This change configures Xcode's Run action to wait for the app to launch after installing it, in order for the debugger to attach. Once Xcode is waiting for the app, wait a few seconds and then launch the application manually on the Apple TV.

Asset Catalog

Resolved Issues

- Fixed: Unable to set Icon Composer icon as alternate iOS icon (153305178) (FB18025356)

Background Assets

Known Issues

- The packaging tool (`ba-package`) and the mock server (`ba-serve`) crash immediately when the selected Xcode installation isn't located at `/Applications/Xcode.app`. (148927743)

  Workaround: Install your desired Xcode version at `/Applications/Xcode.app` and use `xcode-select` to select it.

- When running the mock server (`ba-serve`), the Choose Identity panel might appear behind all other windows and be absent in Mission Control. (152256482)

  Workaround: Move all other windows out of the way to reveal the panel.

Coding Intelligence

Resolved Issues

- Fixed: When using Git LFS, restoring from a snapshot in modification history may fail. (151396769)
- Fixed: Coding assistant may send new message to a previous conversation after hiding and showing the navigator. (151819331)
- Fixed: Apple Intelligence must be turned on before launching Xcode to use coding intelligence features. (152533281)

Known Issues

- When applying changes using coding intelligence, breakpoints in affected files may move. (146888202)
- When using coding intelligence, the model is unable to view or modify files with identical names simultaneously. (147450585)
- Files created by the coding assistant are not deleted when undoing the change. (150298408)
- Models that are unavailable, have been deleted, or are disabled may continue to show as selected and available in existing conversations. (151261617)
- Coding intelligence features in Xcode require Apple Intelligence. (151423060)
- Sometimes the coding assistant may appear empty after sending a message. (152511782)

  Workaround: Scroll in the coding assistant to reveal the messages.
- Previously deleted files may appear in the modification history as having been deleted by a recent coding assistant conversation. (152545638)
- Coding intelligence provides inconsistent results when modifying files that contain thousands of lines. (152590014)

Create ML App

Deprecations in Xcode 26 Beta 3

- Removed support for creating new Style Transfer projects. (154135018)

Foundation

Known Issues

- `#bundle` does not refer to the correct resource bundle when used from a mergeable library. (150698070)

  Workaround: Use `Bundle(for:)` instead and pass a class metatype.

GameKit

Resolved Issues

- Fixed: Testing Party Codes in Game Progress Manager is currently only compatible with MacOS 16.0. Please upgrade your Mac software to use this functionality. (144568615)

Group Activities

Known Issues

- The DestinationVideo sample code doesn't compile. (147074640)

  Workaround: Replace `import GroupActivities` with `@preconcurrency import GroupActivities`.

Icon Composer

Resolved Issues

- Fixed: Icon Composer icons back deploy to older versions of iOS, macOS, and watchOS with inconsistent rendering. (152258860)

Known Issues

- User supplied background images in Icon Composer are composited at the same scale as the 2048x2048 pixel icon renderings, and thus appear much smaller than expected. (148133443)

  Workaround: Use a very large background image.

- Exporting a single PNG, or using the “Copy Icon as Image” menu item, for Clear Light, Clear Dark, Tinted Light, or Tinted Dark will produce an unexpected dark image. (148998556)

  Workaround: Use the export all functionality from the export sheet.

- Icon Composer icons added to macOS app targets do not display in Finder, Spotlight or Dock when deployed to older versions of macOS. (154784104)

  Workaround: none

Instruments

New Features

- Backtraces will now be displayed for state changes when they are the only change in an update group (152202531)

Resolved Issues

- Fixed: The left side of the call tree showing weights could have unexpected column resizing behavior, and the text in the columns may not be horizontally aligned. (151710652)
- Fixed: Recordings using the SwiftUI instrument of an app that accesses a property from an

@Observable class a large number of times could take an extremely long time to process after the recording ends (153239266) (FB17998165)

Known Issues

- No data is present when recording the CPU Counters on macOS devices in the “Streaming SME Bottlenecks” or “Instruction Processing Bottlenecks” modes. (152179757)

Interface Builder

New Features

- The “Show Library” button (″+″) has been moved from the main toolbar to the bar at the bottom of the Interface Builder canvas. (152549110)

Resolved Issues

- Fixed: Downloaded Metal Toolchain may appear unavailable in Xcode when building metal projects. (153096694)
- Fixed: Fix an issue with the xcrun cache not refreshing after the metal toolchain is downloaded when running `xcrun -f metal`. (154682120)

Localization

Resolved Issues

- Fixed: String Catalog files using comment generation or symbol generation JSON fields will auto-update their version number to 1.1. (153395850) (FB18060389)
- Fixed: Improved quality of String Catalog generated comments, including more detailed argument descriptions. (154779538)

Metal

Known Issues

- GPU Profiling in Metal Debugger is not available for Metal 4. (136244517)
- Metal 4 Sparse resources are not supported in Metal Debugger. (136326816)
- Shader Debugging in Metal Debugger is not available for Metal 4. (136626557)
- Compiling Metal shaders with the latest Xcode 26 Metal toolchain for iOS 18, macOS 15, tvOS 18, and visionOS 2 deployment targets may not produce valid GPU binaries. (150887778)

  Workaround: Use previous Xcode 16.3 Metal toolchain to compile Metal shaders to GPU binaries for those deployment targets.

Playgrounds

Resolved Issues

- Fixed: Pinned #Playground blocks do not execute properly. (150710099)

Known Issues

- Canvas doesn’t show results for #Playground blocks in “Legacy Previews Execution”. (150811580)
- Playground macro requires macOS 15.5 or later. (152008152)
- Xcode playgrounds with auxiliary Swift sources fail to compile. (154373965)

Predictive Code Completion

Resolved Issues

- Fixed: Xcode features like Predictive Code Completion and Coding Assistant may require Apple Intelligence to be enabled. (152113360)

Project Editor

Resolved Issues

- Fixed: There is an issue with the “+ Capability” button not showing any capabilities even though the Signing & Capabilities editor is showing. (151898543)

Reality Composer Pro

Resolved Issues

- Fixed: For new Reality Composer Pro projects, in the issue navigator you may see an error such as “Package realitykitcontent is using Swift tools version 6.2.0 but the installed version is 6.0.0.” (152502442)

Known Issues

- The Send to Device toolbar item down arrow does not draw correctly. (147957053)

  Workaround: You can still click the toolbar item to access the Send to Device menu. Alternatively, you can use File > Device in the menu bar.

Security

Known Issues

- The Enhanced Security capability options in Xcode to enable platform restrictions and read-only platform memory do not add those protections and instead adds an entitlements that have no effect. (152059695)

Workaround: Manually add the `com.apple.security.hardened-process.platform-restrictions` entitlement to your target with a Boolean value of YES.

Simulators

Resolved Issues

- Fixed: Installing apps from Xcode to a simulator, when build products or the simulator runtime are located in non-standard locations or on non-APFS fliesystems may fail. (152938778) (FB17867688)

Source Editor

Resolved Issues

- Fixed: The fonts and colors are incorrectly stored in the themes files. (152506293)

Known Issues

- Change bars may be out of date when reverting changes from the coding assistant. (152399120)

Swift

Resolved Issues

- Fixed: `Observations` is not available. (152888116)
- Fixed: `Data.bytes` and other `Span` API is not available. (152888512)

Swift Compiler

Known Issues

- Some incorrect uses of Span and MutableSpan do not produce compile errors. This may result in unexpected program behavior. (150275147)

  Workaround: None

- Some incorrect uses of Span and MutableSpan do not produce compile errors. This may result in unexpected program behavior. (150403948)

  Workaround: None

Swift Interoperability with C/C++

Known Issues

- Enabling the safe Swift/C++/C interoperation feature can cause a compilation error when used in a project with an older deployment target where Span isn't available. (150740330)

  Workaround: Set your project's minimum deployment target to iOS 26.0, macOS 26.0, tvOS 26.0, watchOS 26.0, or visionOS 26.0.

- With the safe Swift/C++/C interoperation feature enabled, annotating a return pointer with `__sized_by` causes a compilation error. (151799287)

- With the safe Swift/C++/C interoperation feature enabled, annotating a return pointer with both `__counted_by` and `_Nullable` causes a compilation error. (151804085)

Swift Macros Build Performance

Known Issues

- You may experience build failures when building projects with Swift macro dependencies. Common symptom is a build failure around `_SwiftSyntaxCShims`. You can work around this by disabling the swift-syntax prebuilts for macros feature. Quit Xcode. Issue the command `defaults write com.apple.dt.Xcode IDEPackageEnable`

`Prebuilts NO` at the command line. Clean out the DerivedData for the project in `~/Library/Developer/Xcode/DerivedData`. Restart Xcode and execute the build. (152253497)

- In a project that uses macros and when using the Legacy Preview Execution for Canvas, builds will fail when building for preview. You can work around this by disabling the swift-syntax prebuilts for macros feature. Quit Xcode. Issue the command `defaults write com.apple.dt.Xcode IDEPackageEnablePrebuilts NO` at the command line. Clean out the DerivedData for the project in `~/Library/Developer/Xcode/DerivedData`. Restart Xcode and execute the build. (152443472)

Swift Package Manager

Known Issues

- `.treatAllWarnings` and `.treatWarning` target APIs in SwiftPM package manifests are missing. (152340059)

  Workaround: Pass `–warnings-as-errors`, `–Wwarning`, and `–Werror` as `–Xswiftc` flags on the command line or `.unsafeFlags([...])` in your package manifest.

- `swift package migrate` fails with an “expected string” error. (152533205)

Swift Packages

Resolved Issues

- Fixed: Edits made to local package dependencies are incorporated into subsequent builds. (140069965) (FB15863852)

Testing

Resolved Issues

- Fixed: When a Swift.org toolchain is selected via the “Xcode > Toolchains” menu, the copy of Swift Testing in that toolchain may not be used when running tests which may cause launch failures. (148853997)
- Fixed: When applying the `.serialized` trait to a parameterized Swift Testing `@Test` function directly, its cases will be executed serially. (154529146)

Known Issues

- Swift Testing exit tests may produce crash logs in /Library/. (47982238)
- Attachments are not recorded from within the bodies of exit tests when using Swift Testing. (149242118)
- Modules which use Swift Testing and enable the Strict Memory Safety feature introduced in Swift 6.2 encounter memory safety diagnostics from `@Test`, `@Suite`, and other macros. (151238560)
- Using a Swift.org toolchain in Xcode may cause build errors in targets that use Swift Testing due to an incompatible macro plugin being selected. (151799859)

  Workaround: In an Xcode project, add the custom Swift flags `-plugin-path $(TOOLCHAIN_DIR)/usr/lib/swift/host/plugins/testing` via the `OTHER_SWIFT_FLAGS` setting in affected targets.
- Swift Testing tests and suites whose names use raw identifiers (introduced in SE-0451: Raw identifiers) cannot be run selectively via Xcode’s Test navigator or source editor diamonds, and cannot be included or excluded in test plans. (152524780)

  Workaround: Choose “Product > Test” to run all tests, or avoid using raw identifiers for affected tests and suites.

Updates in Xcode 26 Beta 2

General

New Features in Xcode 26 Beta 2

- Hang and Launch diagnostics are enhanced with trending insights, calling attention to issues that have increased in impact over the last 4 app versions and providing additional context on the general change in impact to better highlight the prevalence. Leverage these insights, called out with a flame icon in the source list, to understand when an issue may have been introduced and to prioritize performance optimizations in new versions of your app. (135376723)

Resolved Issues in Xcode 26 Beta 2

- Fixed: Xcode may crash on launch restoring UI state. (152042209)

Build System

Resolved Issues in Xcode 26 Beta 2

- Fixed: Projects which have added support for the arm64e architecture may fail to build for the Simulator. (123839235)
- Fixed an issue which caused compilation to fail when compiler caching was enabled. (153298732) (FB18022508)

GameKit

Resolved Issues in Xcode 26 Beta 2

- Fixed: Deep links within the Game Progress Manager are currently non-functional in Beta 2. (153754027)

Playground Macros

Resolved Issues in Xcode 26 Beta 2

- Fixed: Playground macros currently cause a build failure saying that `Playgrounds` cannot be found when built for Mac Catalyst. (151029441)

Source Control

Resolved Issues in Xcode 26 Beta 2

- Fixed an issue where workspaces touching git repositories with many tags or branches might experience sudden hangs or spins. (144901032)

Source Editor

Resolved Issues in Xcode 26 Beta 2

- Resolved several major performance issues related to syntax highlighting in Swift files. (144533425)

Updates in Xcode 26 Beta

General

New Features in Xcode 26 Beta

- Added a new setting that dictates how function names are displayed in C++ frames: plugin.cplusplus.display.function-name-format

  By default this displays the entirety of a function name, but can be customized to drop various parts of a function signature (e.g., return type, scope qualifiers, etc.). See https://lldb.llvm.org/use/formatting.html#function-name-formats for more details.

  By default LLDB will now highlight the function basename when displayed in C++ frames

(e.g., in the backtrace). (150174227)

Resolved Issues in Xcode 26 Beta

- Fixed: The alert shown when a file, project, or workspace is modified by a different program while also opened in Xcode had inconsistent button labeling, and the default action did not always keep the Xcode version of the file. (86281699)

AssistantSchemas

Resolved Issues in Xcode 26 Beta

- Fixed: If you have adopted any of the following email AssistantSchemas, you will experience a compilation error due to a parameter type change: `createDraft`, `updateDraft`, `replyMail`, `forwardMail`, `message`, and `draft`. (148633307)

Build System

New Features in Xcode 26 Beta

- Compilation caching has been introduced as an opt-in feature, which speeds-up iterative build/test cycles for Swift and C-family languages. The compilation caching feature caches the results of compilations that were produced for a set of source files inputs and, when it detects that the same set of source files are getting re-compiled, it speeds-up the build by providing the prior compilation results directly from the cache. The workflows that will benefit the most from compilation caching are when switching between branches (which ends up re-compiling the same source files again) or when doing clean builds.

  Compilation caching can be enabled via “Enable Compilation Caching” build setting or via the per-user project settings. (149700201)

Resolved Issues in Xcode 26 Beta

- Fixed: When using synchronized group folders, the file order was non-deterministic, causing unnecessary rebuilds of the target from `xcodebuild` or after re-opening the Xcode project. (151472630)

C++ Standard Library

New Features in Xcode 26 Beta

- The following C++26 features have been implemented:
  - `constexpr` placement new (P2747R2) (Github)
  - A type trait for detecting virtual base classes (P2985R0) (Github)

  The following C++23 features have been implemented:
  - Relaxing Ranges Just A Smidge (P2609R3) (Github)
  - A trait for implicit lifetime types (P2674R1) (Github)
  - A Standard `flat_map` (P0429R9) (Github)

  The following C++20 features have been implemented:
  - Reviewing Deprecated Facilities of C++17 for C++20 (P0619R4) (Github)

  Performance and memory improvements:
  - The `lexicographical_compare` and `ranges::lexicographical_compare` algorithms have been optimized for trivially equality comparable types, resulting in a performance improvement of up to 40x.
  - The internal structure `__compressed_pair` has been replaced with `[[no_unique_address]]`, resulting in reduced compile times and smaller debug information as well as better code generation if optimizations are disabled.
  - The `input_iterator`-pair overload of `void assign(InputIt, InputIt)` has been optimized for `std::vector`, resulting in a performance improvement of up to 2x for trivial element types (e.g., `std::vector<int>`), and up to 3.4x for non-trivial

element types (e.g., `std::vector<std::vector<int>>`).

- The `input_iterator`-pair overload of `iterator insert(const_iterator, InputIt, InputIt)` has been optimized for `std::vector`, resulting in a performance improvement of up to 10x for `std::vector<int>`, and up to 2.3x for `std::vector<std::vector<int>>`.
- `std::stable_sort` uses radix sort for integral types now, which can improve the performance up to 10 times, depending on type of sorted elements and the initial state of the sorted array.
- Reduced the amount of debug information generated for internal typedefs. This reduces the size of debug builds.

Hardened libc++ related improvements:

- The `_LIBCPP_ABI_BOUNDED_ITERATORS_IN_STD_ARRAY` ABI configuration was added, which allows storing valid bounds in `std::array::iterator` and detecting OOB accesses when the appropriate hardening mode is enabled.
- The `_LIBCPP_ABI_BOUNDED_UNIQUE_PTR` ABI configuration was added, which allows `std::unique_ptr<T[]>` to detect out-of-bounds accesses in certain circumstances. `std::unique_ptr<T[]>` can now also detect out-of-bounds accesses for a limited set of types (non-trivially destructible types) when the ABI configuration is disabled.
- Added (hardening mode) support for `forward_list` and `bitset`.

General Improvements:

- `std::jthread` and `<stop_token>` are not guarded behind `-fexperimental-library` anymore
- The `_LIBCPP_ENABLE_CXX20_REMOVED_TEMPORARY_BUFFER` macro has been added to make `std::get_temporary_buffer` and `std::return_temporary_buffer` available.
- The `std::uncaught_exception` function was marked as deprecated since C++17 and removed since C++20. The `_LIBCPP_ENABLE_CXX20_REMOVED_UNCAUGHT_EXCEPTION` macro has been added to make `std::uncaught_exception` available

in C++20 and later modes.

ABI Affecting Changes:

- The ABI breaks for removing undefined behaviour in `std::forward_list`, `std::list`, `std::map`, `std::set`, `std::multimap`, `std::multiset`, `std::unordered_map`, `std::unordered_set`, `std::unordered_multimap` and `std::unordered_multiset` are now applied unconditionally. This only affects fancy pointers which have a different value representation when pointing at the base of an internal node type instead of the type itself. A size or alignment difference is diagnosed, but more subtle ABI breaks may result in unexpected behaviour.
- The internal structure `__compressed_pair` has been replaced with `[[no_unique_address]]`. The ABI impact is:
- When using the Itanium ABI (most non-MSVC platforms), empty types are now placed at the beginning of the enclosing object instead of where the beginning of the `__compressed_pair` subobject was. This is only observable by checking the address of the empty allocator, equality comparator or hasher.
- Additionally, using an overaligned empty type as an allocator, comparator or hasher in the associative containers (and only those containers) may result in the container's object object size and data layout changing beyond only the address of the empty member.
  - The changes for `ranges::zip_view` from (P2165R4) have been implemented. This changes the element type of `zip_view` from a `std::pair` to a `std::tuple` in some cases. This is technically an ABI break, however since `zip_view` is generally not an ABI sensitive type, we don't expect users to encounter any issues and we don't provide a way to change this behavior, which would make libc++ non-conforming. (151879824)

Deprecations in Xcode 26 Beta

- The following items have been deprecated or removed:
  - The `_LIBCPP_ENABLE_ASSERTIONS` macro that were used to enable the safe mode

have been removed in LLVM 20. Please use (support for hardening) instead.

- Support for the C++20 synchronization library (`<barrier>`, `<latch>`, `atomic::wait`, etc.) has been removed in language modes prior to C++20. If you are using these features prior to C++20, you will need to update to `-std=c++20`.
- The relational operators for `std::chrono::weekday` have been removed entirely, and the `_LIBCPP_ENABLE_REMOVED_WEEKDAY_RELATIONAL_OPERATORS` macro is now ignored.
- The `_LIBCPP_ENABLE_REMOVED_ALLOCATOR_CONST` macro no longer has any effect. `std::allocator<const T>` is not supported as an extension anymore, please migrate any code that uses e.g. `std::vector<const T>` to be standards conforming.
- Non-conforming member typedefs `base`, `iterator`, `const_iterator`, `size_type`, `difference_type`, and `const_reference` of `std::bitset`, and member typedef `base` of `std::forward_list` and `std::list` are removed. Previously, these member typedefs (except `const_reference`) were private but could cause ambiguity in name lookup. Code that expects such ambiguity will possibly not compile in LLVM 20.
- The function `__libcpp_verbose_abort()` is now `noexcept`, to match `std::terminate()`. (The combination of `noexcept` and `[[noreturn]]` has special significance for function effects analysis.) For backwards compatibility, the `_LIBCPP_VERBOSE_ABORT_NOT_NOEXCEPT` macro can be defined to make the function non-`noexcept`. That macro will be removed in LLVM 21.
- `<ccomplex>`, `<cstdalign>` (previously missing), `<cstdbool>`, and `<ctgmath>` are deprecated since C++17 as specified by the standard. They, together with `<ciso646>`, are removed in C++20, but libc++ still provides these headers as an extension and only deprecates them. The `_LIBCPP_DISABLE_DEPRECATION_WARNINGS` macro can be defined to suppress deprecation for these headers.
- The pointer safety functions `declare_reachable`, `declare_no_pointers`, `undeclare_no_pointers` and `__undeclare_reachable` have been removed from the library. These functions were never implemented in a non-trivial way, making it very unlikely that any binary depends on them.

- Non-conforming extension `packaged_task::result_type` is deprecated. It will be removed in LLVM 21.
- The changes for `ranges::zip_view` from (P2165R4) have been implemented. This can lead to code assuming that `zip_view` produces `std::pair` to stop compiling now that it produces `std::tuple`. The cases are rare since `tuple` and `pair` are compatible for the most part, but this can lead to code that was previously accepted now being rejected. This is necessary for libc++ to be conforming, so we don't provide any way to opt-out of that behavior. (151880074)

Debugger

Resolved Issues in Xcode 26 Beta

- Fixed: The debugger is now able to follow a Swift Task when a step operation causes the Task to be migrated to a different thread. (141825713)
- Fixed: The payload of a Swift enum or Optional SBValue is now represented as a synthetic child rather than a direct child. Custom Python data formatters that unwrap Optional values continue to work with this change, as long as do not turn SetPreferSyntheticValue() off. (152662881)

Editors

Resolved Issues in Xcode 26 Beta

- Fixed: In Xcode 26 you can control when Editor Tabs are pinned and added. By default Xcode will only create tabs you ask for (File > New > Tab or Option+Click navigation). Automatic pinning is available in Xcode > Settings > Navigation > Pin Tabs Any Editor Tab can be pinned or unpinned by clicking the pin icon or using View > [Pin, Unpin] Editor Tab. (135835457)
- Fixed: The new Start Page in Xcode 26 provides the familiar Open Quickly queries and Related Items in an editor. The Start Page is shown for new Editor Tabs and new Editor

Panes without other content. This can be configured in Xcode > Settings > Navigation. (137163041)

Foundation

New Features in Xcode 26 Beta

- The `#bundle` macro allows referring to the resource bundle associated with the current Xcode target. You can pass this to any Foundation API expecting a `Bundle`, such as when looking up images or localized strings. (79247613)
- NotificationCenter can post and observe new `Message` types with custom properties and isolation checking (132023811)

GameKit

New Features in Xcode 26 Beta

- In addition to Leaderboards and Achievements, you can now also adopt, configure and modify Challenges and Activities directly in Xcode making it easier to enhance engagement in your game. Use Xcode to pull, modify and push Challenges and Activities updates to and from App Store Connect. The Game Progress Manager in Xcode allows you to test your Party Codes locally during development by modifying properties, reporting updates and resetting resources. These real-time updates can be used to verify that the Activity changed as expected. (144816182)

Instruments

New Features in Xcode 26 Beta

- Target chooser has been redesigned from the ground up to improve the device and process selection experience. Changes include:

- Move of the control from the toolbar to the Next Recording section, unifying all the settings relevant for the next run in one place.
- Ability to filter available set of processes to launch or attach, including apps, launch agents and daemons.
- Quick access to modify the environment or arguments for a process being launched by Instruments. (79987176)

- CPU Counters has been reworked to support a performance optimization methodology called Bottleneck Analysis, which uses preset modes that provide a “guided” path to using counters. The CPU Bottlenecks template configures the counters to initially break down down sustainable CPU bandwidth into four broad categories:
  - Useful: CPU resources were used to make forward progress in your code
  - Instruction Delivery: the CPU could not provide enough instructions for processing
  - Instruction Processing: instructions are not executing quickly due to their relative cost or data not being ready
  - Discarded: incorrect speculative execution wasted bandwidth.

  Each category aside from Useful has one or more further recommended modes to narrow in on instruction sequences that are preventing the CPU from executing efficiently. There are also traditional modes that measure the characteristics of executed instructions or metrics about their execution, like the number of L1 Data Cache misses.

  Manual mode for configuring Counters is still available and can be accessed from the instrument recording options. (86470694)

- Power Profiler is a new instrument designed to visualize system power usage and the power impact of an application on different subsystems like CPU, GPU or networking. To use the tool, drag it from the instruments library and record a trace on an iOS or iPadOS device. Process metrics are only present when using “Launch” or “Attach” recording modes in Instruments. Instruments also supports analysis of traces taken on-device using the Power Profiler feature in the Developer settings. To visualize captured trace, open the resulting .atrc file in Instruments, which will present both power data and CPU samples captured

alongside. (135759149)

- The SwiftUI template has been updated with a next-generation SwiftUI instrument. The new instrument captures the duration of all of the updates SwiftUI performs, making it easy to identify long updates that may be negatively impacting app performance. It also tracks the causes of each update, allowing you to understand why view bodies are running, using the new Cause & Effect Graph. (137637030)
- Timeline content tooltips are now rendered in multiline lines when applicable. (139212962)
- Instruments General settings features a new “Attach Supporting files” toggle, allows Instruments to attach imported files or underlying Apple Trace files as part of the .trace container when enabled. (144313007)
- Added ability to name runs with `xctrace record` using `run-name` argument. Run names are presented when trace is subsequently opened in Instruments. (144883457)
- Animation Hitches Instrument has been redesigned to address correctness and performance issues. New version supports multiple displays across the platforms, includes application update intervals and more reliably reports hitching frames. Volume of recorded data has been reduced significantly, contributing to faster trace processing times. (146033792)
- Instruments Settings view has been redesigned for readability. (146418991)
- The Foundation Models instrument is a new tool designed to help developers profile their app’s usage of the FoundationModels framework. The instrument provides insights into the app’s requests and different phases, such as asset loading, prompt processing, and inference details. To use it, drag it from the Instruments library and record a new trace. (148055761)
- Call Tree has an ability to view all the aggregated samples using the “Focus” submenu for a node. The “Focus” button available on hover now defaults to showing the submenu with multiple actions instead of invoking “Focus on Subtree”. (148098893)
- Secondary clicking on a track in the timeline now offers a context menu with actions that allow for track pinning, copying represented value or adding it to various filters across the application. (148666063)

- Tracks in the timeline are now automatically sized to fit the plots inside. Tracks can still be resized manually using cursor or CMD+, CMD- keyboard shortcuts (148943810)
- Swift Concurrency template now surfaces names of tasks given using the new API available in Swift 6.2. (149797014)

Resolved Issues in Xcode 26 Beta

- Fixed: Copy of unsymbolicated frames from a call tree view now includes names of libraries. (141645741)
- Fixed: Copy and Copy with Header actions are now properly handled in the aggregation detail views across Instruments. (146333462)
- Fixed: Resolves a large memory leak happening when recording with the User Space Sampler Instrument. (146506463)
- Fixed issues with the metric calculation in the CPU Counters Instrument happening during thread context switches (147776170)
- Fixed: Set Inspection Range and Zoom action is now available as a standalone menu item instead of an alternate. (149986438)
- Fixed an issue where pinned timeline wouldn't be sized appropriately when presented. (150055669)
- Fixed: Main menu of Instruments has been simplified. Document menu has been removed and its actions have been moved under `View`. (150067894)
- Fixed: Pause / Resume button has been removed from the toolbar. It is still present in the main menu of the application. (150984980)
- Fixed an issue where resizing the timeline track by dragging would sometimes get stuck or not be responsive to user's action. (151040978)
- Fixed: Plot titles in the timeline will now show tooltips when the text is truncated. (151046422)

Deprecations in Xcode 26 Beta

- Support for profiling WatchKit 1.0 applications has been removed. (139213888)
- The Zombies template has been removed. To record Zombies, select the “Allocations” template and enable the “Enable NSZombie detection” checkbox. (146791790)
- The SceneKit template has been removed. SceneKit Instrument remains available in the library. (146791805)
- Previous SwiftUI template containing View Body and View Properties Instruments have been replaced. The instruments are now deprecated and still available in the instruments library. (148596828)

Localization

New Features in Xcode 26 Beta

- Xcode can now generate type-safe Swift symbols for manually-managed strings in String Catalogs. For example, a string in Localizable.xcstrings with key “Landmarks” and value “%(count)lld landmarks” can be accessed via `LocalizedStringResource.landmarks(count: 42)`. You can enable this via the build setting “Generate String Catalog Symbols”. (110427968) (FB12264530)
- The String Catalog Editor now supports selecting and performing actions on multiple strings at once. (110457331) (FB12271972)
- Xcode can now use source code context to generate translation comments in a String Catalog. Use “Generate Comment” in the context menu to generate comments for existing strings. To ensure newly-extracted strings get comments by default, turn on “Automatically generate string catalog comments” in Settings > Editing. This requires Xcode’s on-device predictive code completion model. (111708155)
- The Refactor menu in String Catalogs and the Source Editor now lets you change a string key across all code files and String Catalogs. (129261942)
- A new Refactor menu in the String Catalog Editor allows converting one or more strings

between automatically-extracted string literals and manually-managed strings referenced via Swift symbols. This allows you to update source code at the same time as making String Catalog changes. These operations are also accessible via the Refactor menu in the Source Editor. (129261981)

- Expressive Format Specifiers in String Catalogs allow you to add a name to any format specifier. For example, `%(landmarkCount)lld` clearly communicates that the integer passed in at runtime represents a count of landmarks. These format specifiers will be included in the exported Localization Catalog (.xcloc) to help guide translators. (136543666)
- When typing "%" in a String Catalog, an auto-completion menu suggests relevant format specifiers for your string. (139033592)
- The String Catalog filter bar now allows filtering for strings with generated comments and strings using symbol generation. (151563807)

Metal

New Features in Xcode 26 Beta

- Metal Debugger in Xcode 26 has been updated to support Metal 4 in most debugging workflows. This includes support for Machine Learning tensor visualization, Machine Learning graph visualization and debugging, and an updated Dependency Viewer that now also visualizes per stage dependencies. (136174499)
- Metal System Trace in Instruments has been updated to support Metal 4 (136242467)
- Metal Debugger in Xcode 26 has been updated with the ability to export performance data into a GPU Trace than can be later visualized in Xcode using the same or a different Mac, without the need to replay the trace. (137163147)
- Metal Debugger in Xcode 26 has been updated with improved profiling workflows. This includes a new Summary tab that shows an overview of the profiling session, a new Shaders tab providing statistics for shaders in the trace, and an updated visualization for counters in the GPU Timeline. (137242382)

- Metal Performance HUD has been updated with new metrics and an enhanced user interface that supports customization. It also now has the ability to identify performance insights and generate performance reports for games running natively or within the evaluation environment. (151481481)

Resolved Issues in Xcode 26 Beta

- Fixed: Xcodebuild may fail to find the downloaded metal toolchain when building a project with metal shaders by command line. (152810434)

Objective-C Runtime

New Features in Xcode 26 Beta

- Concurrent mutation of nonatomic properties in Objective-C will now sometimes produce more actionable crashes. Synthesized setters will briefly store the sentinel value 0x400000000000bad0 (0xbad0 on 32-bit watchOS) which may be read by another thread accessing the property unsafely. A crash on this sentinel value indicates a thread safety issue with the property it came from. (148109501)

Organizer

New Features in Xcode 26 Beta

- Metric recommendations are now available for the launch time metric in the Xcode Organizer. When there is enough information, the Organizer will display a recommended value for a metric on the chart associated with your app's metrics. Use this data to plan and prioritize performance engineering work. (141243138)

Previews

Resolved Issues in Xcode 26 Beta

- Fixed: Previews could fail to find libraries linked through symlnked paths in some cases. This should be resolved. (142639854) (FB16276100)
- Fixed: Previewing code that used libraries asserting they were loaded on the main thread could crash. Previews now runs library initializers on the main thread. (145534532) (FB16599189)
- Fixed: Previews now handles static libraries that were resulting in a “does not contain an archive” error message. If you find an example of this that still is not resolved, please file a feedback and attach the static library for analysis. (146364028) (FB16747894)
- Fixed: Previews now works better with weak linked libraries on recent OSes. (146794084)
- Fixed: Previews could fail when targeting a physical device if the path to derived data crossed a symlink. This is now resolved. (149032016)

RealityKit

New Features in Xcode 26 Beta

- RealityKit is not supported on the Apple TV HD (4th generation). (145292590)
- RealityKit is available from tvOS 19.0 and supports all Apple TV 4K models. (145952172)

Resolved Issues in Xcode 26 Beta

- Fixed: `ParticleEmitterComponent` does not render properly on iOS, macOS, and tvOS. (152201501)

SceneKit

New Features in Xcode 26 Beta

- The scntool CLI now supports converting SCN files (including animations) to USD formats. (148582191)

Deprecations in Xcode 26 Beta

- SceneKit is now deprecated across all Apple platforms. Developers can continue to use SceneKit in their projects, but no new features or optimizations are coming to this framework. For new projects, Apple recommends using RealityKit. (147454720)

Security

New Features in Xcode 26 Beta

- Xcode now provides an “Enhanced Security” extension template, which you can use to separate sensitive calculations, such as handling data from untrusted sources, into an extension that the system runs as a separate process from your app. For more information see https://developer.apple.com/documentation/xcode/creating-extensions-with-enhanced-security (141308470)
- Xcode now provides an “Enhanced Security” capability, which enables additional runtime and compile-time protections for your application and builds it with pointer authentication enabled. For more information see https://developer.apple.com/documentation/xcode/enabling-enhanced-security-for-your-app (143278278)
- Clang now supports a language extension for C, `-fbounds-safety`, that guarantees bounds safety. The extension also provides bounds annotations that you can use to specify bounds information for pointers and arrays. The compiler informs you where to add bounds annotations through errors and warnings and automatically inserts guaranteed bounds checks at run time. To enable this extension, set the `ENABLE_CLANG_BOUNDS_SAFETY` build setting to `YES`. For more information see -fbounds-safety: Enforcing bounds safety for C. (149099879)
- Clang now supports a strict programming model that guarantees bounds safety in C++ by rejecting raw pointer arithmetic and requiring the use of hardened C++ Standard Library APIs for buffer manipulation. To enable this mode set the `ENABLE_CPLUSPLUS_BOUNDS_SAFE_BUFFERS` build setting to `YES`. (150517840)

Simulator

Resolved Issues in Xcode 26 Beta

- Fixed: The visionOS Simulator now uses a 4K resolution when running visionOS 3.0. (121031014)

Source Editor

New Features in Xcode 26 Beta

- Predictive Code Completion in Xcode now supports progressively accepting completions in smaller segments by holding the ^ key. (137459125)
- An annotation displaying the #if condition is displayed at the end of a line starting with #endif. The annotation will only be visible if there are no #else or #elseif branches. (143072375)

Swift

Resolved Issues in Xcode 26 Beta

- Fixed: Starting from Xcode 26, Swift explicit modules will be the default mode for building all Swift targets. When encountering severe issues, projects could opt-out of Swift explicit modules by specifying a build setting SWIFT_ENABLE_EXPLICIT_MODULES=NO. (125244407)
- Fixed: When building projects using Swift language versions prior to Swift 5, Swift explicit modules remains to be disabled by default. (146404229)
- Fixed: Swift explicit modules hasn’t been enabled by default for project targets that use Swift/C++ interoperability. (152118892)

Swift Compiler

New Features in Xcode 26 Beta

- MutableSpan may now be passed as an inout function parameter without enabling an experimental feature. (150557314)

Swift Interoperability with C/C++

New Features in Xcode 26 Beta

- Swift compiler now automatically infers `SWIFT_SHARED_REFERENCE` annotation for C++ types inheriting from a base that is annotated with `SWIFT_SHARED_REFERENCE`. The derived type is imported as a Swift class or reference-counted type with the same `retain` and `release` operations as the base type. (97914474)

- You can now annotate C and C++ functions with bounds and lifetime annotations that enable Swift to import them without the need for unsafe boilerplate. This feature is currently experimental. To enable it, add the `-enable-experimental-feature SafeInteropWrappers` to your `OTHER_SWIFT_FLAGS` build setting.

  Add `__noescape` to a C or C++ function parameter to indicate that the parameter doesn't escape the function. If you annotate a parameter of `std::span` type with `__noescape`, the compiler generates a safe overload that directly takes a Span in Swift.

  Add `__lifetimebound` on a C or C++ function parameter to indicate that the return value of the function does not outlive the objects referred to by that parameter. If you annotate a parameter with `__lifetimebound` and the return value is of `std::span` type, the compiler will generate a safe overload that directly returns a Span.

  You can annotate raw C and C++ pointers with these lifetime annotation and also with bounds annotations, such as `__counted_by(N)`, which indicate that the raw pointer points to memory containing at least `N` elements. If you add both a lifetime annotation and a bounds annotation to a raw pointer in a function parameter, Swift will import it as a Span. Here is an example:

```
// C/C++
#include <lifetimebound.h>
#include <ptrcheck.h>
void initializeBuffer(int * __counted_by(len) p __noescape, size_t len);

// Unsafe overload in Swift
func initializeBuffer(_ p: UnsafeMutablePointer<CInt>, _ len: Int)

// Safe overload in Swift
func initializeBuffer(_ p: inout MutableSpan<CInt>)
```

Note that the safe overload of initializeBuffer doesn't contain a separate len parameter because MutableSpan already contains its count.

Similarly, if you annotate a parameter with `__lifetimebound` and annotate a function return type with a bounds annotation, Swift will import the return type as a Span.

For `void *` or pointers to type with an unknown size, you can use the `__sized_by(N)` bounds annotation which indicates that the pointer points to memory of at least N bytes. (97942270)

- Constructors of C++ `SWIFT_SHARED_REFERENCE` types are now callable from Swift as initializers, allowing these types to be constructed directly from Swift. (131557219)
- Objective-C and Objective-C++ APIs that return C++ `SWIFT_SHARED_REFERENCE` types can now be annotated with `SWIFT_RETURNS_RETAINED` or `SWIFT_RETURNS_UNRETAINED` annotations. (135360972)
- C++ classes can be annotated with `SWIFT_PRIVATE_FILEID` to allow Swift extensions of those classes to access their `private` and `protected` members. This annotation is defined in the `swift/bridging` header and takes a fileID as its argument (e.g., `SWIFT_PRIVATE_FILEID("MyModule/MyFile.swift")`). For C++ classes with this annotation, non-`public` members are treated as if they were defined as `private` in the Swift source file designated by the fileID. (137764620)
- Objective-C and Objective-C++ functions and methods that return C++ `SWIFT_SHARED`

_REFERENCE types can now use the new `SwiftReturnOwnership:` API Notes attribute to specify ownership. Set the value to `retained` or `unretained` to apply the corresponding `SWIFT_RETURNS_RETAINED` or `SWIFT_RETURNS_UNRETAINED` annotation. (142504115)

- You can now annotate function parameters and return values of clang modules with bounds attributes using APINotes. This lets you to create safe Swift interfaces when using the SafeInteropWrappers feature without modifying the headers of external modules. Example:

```
// C header
void foo(int * p, size_t len);
void * bar(size_t size, size_t count);

// APINotes
Name: MyBoundsModule
Functions:
  - Name: foo
    Parameters:
      - Position: 0
        BoundsSafety:
          Kind: counted_by
          BoundedBy: "len"
  - Name: bar
    BoundsSafety:
      Kind: sized_by
      BoundedBy: "size * count"
```

(143701027)

- The `SWIFT_RETURNS_RETAINED` and `SWIFT_RETURNS_UNRETAINED` annotations can now be applied to C++ APIs with templated return types. (143732201)
- C++ `std::string` now conforms to Swift `ExpressibleByStringInterpolation` (147249169)

- The Swift compiler no longer emits a warning for C++ functions and methods that return `SWIFT_SHARED_REFERENCE` types without an explicit `SWIFT_RETURNS_RETAINED` or `SWIFT_RETURNS_UNRETAINED` annotation. (150937617)
- Static C++ factory functions returning `SWIFT_SHARED_REFERENCE` types are now imported as Swift initializers, as long as they are annotated with `SWIFT_NAME("init(_:_:_:)")`, with the underscore placeholders matching the factory function's parameter count. (152457366)

Resolved Issues in Xcode 26 Beta

- Fixed: Swift compiler would sometimes generate a C++ header for a Swift module that had nested types declared out of order, causing compilation errors. (147882976)

Swift Macros Build Performance

New Features in Xcode 26 Beta

- Build for Swift macro targets is accelerated by downloading a prebuilt library for swift-syntax from `swift.org` and integrating it into the build. This feature is enabled automatically and will improve build times for these projects. (151701829)

Swift Package Manager

New Features in Xcode 26 Beta

- `swift test` now accepts an `--attachments-path` argument for specifying a directory where attachments should be saved when using Swift Testing. See ST-0009: Attachments. (147753584)

Swift Packages

New Features in Xcode 26 Beta

- Xcode 26 contains a preview of a new implementation for Swift package builds based on shared code with Swift Package Manager, which will improve consistency and stability of builds across Swift Package Manager and Xcode and will become the default in a future release. To try the new implementation, set the user default: `defaults write com.apple.dt.Xcode IDEEnableNewPackagePIFBuilder -bool YES` and please report any issues or differences in behavior via Feedback Assistant. (132025293)

Resolved Issues in Xcode 26 Beta

- Fixed an issue where the Add Package assistant would occasionally not load if you had a private package from GitHub in your list of recently used packages. (136229497) (FB15166959)

Test Report

Resolved Issues in Xcode 26 Beta

- Fixed: XCUIElementQuery suggestions in the Automation Explorer are more accurate and concise when there are multiple similar elements in the UI hierarchy. (133085864)

Testing

New Features in Xcode 26 Beta

- Swift Testing now supports exit tests which allow you to test code that calls `precondition()` or `fatalError()` or might otherwise terminate the test process. (126206192)
- New UI Test Recording experience. To start recording UI interactions, navigate to the test case function and click the record button at the edge of the editor. (135593667)
- Swift Testing now supports attaching files or data to help diagnose test failures.

Attachments are shown in Xcode's Test Report and included in .xcresult bundles. (ST-0009) (139098583)

- Runtime issue detection in Swift Testing and XCTest. When running tests through Xcode and xcodebuild users will be notified of runtime issues that occur during their tests. This detection can be configured in the Test Plan by either promoting the runtime issues to failures or turning them off. (141704793)
- Swift Testing's `ConditionTrait` type now includes an `evaluate()` method which evaluates its underlying condition, allowing these traits to be composed more easily. (ST-0010) (142690932)
- The total number of arguments passed to a parameterized Swift Testing test function is now shown in its console output message when it finishes. (144250498)
- The `xcodebuild build-for-testing` command now includes information about disabled test plan configurations in generated `.xctestrun` files. Disabled test plan configurations are not run by default, but may be run selectively by passing the `-only-test-configuration <name>` flag one or more times to `xcodebuild test-without-building`. (144809893)
- When verbose output is enabled, a console message will be printed each time an invocation of a parameterized Swift Testing test function finishes including its pass/fail status and the total number of issues it recorded. (146863942)
- API added to XCTest for creating non failing issues. Test Authors can create a `XCTIssue` with a severity level as warning in order to record a non failing issue. In order to handle these non failing issues we also added API on `XCTIssue` to determine if the issue `is Failure`. (148547346)

Resolved Issues in Xcode 26 Beta

- Fixed: Test diamonds shown in the source editor no longer disappear after changing branches or otherwise modifying Xcode's project file. (109443495)
- Fixed: If a parameterized Swift Testing test function includes two or more arguments that either don't have a stable representation (e.g. they don't conform to `Codable` or any other

supported protocol) or have an identical stable representation, they will each now run even if parallelization is enabled. Previously this situation could crash, and most often affected arguments which had equal values for their `description` property. (121455205)

- Fixed: Comments passed to the `withKnownIssue` family of APIs are now displayed in the Test Report next to the known issue. (126711065)
- Fixed: If an issue is recorded during an invocation of a parameterized Swift Testing test function, and then the arguments of the failing run are modified and the issue is resolved upon rerun, the outdated issues will be removed from the Issue navigator and source editor. (134298578)
- Fixed: When an expectation such as `#expect(...)` fails, the amount of automatically-collected data representing its subexpressions is now limited to avoid poor performance when a value references a large object or collection. (138208832)
- Fixed: Swift Testing no longer crashes if an expectation such as `#expect` has an empty comment. (149482060)
- Fixed: When an expectation such as `#expect(...)` has `try` or `await` keywords before it, code comments on the preceding lines are now collected. (151825393)
- Fixed: The `@Test` macro no longer produces a compilation error in contexts where there is a concrete type named `Actor`. (152039067)