iOS 18 – The Developer Guide

Christian Drapatz

June 2026

Table of Contents

Apple Platforms 18 3
Overview 3
iOS 18 3
Version 18.0 3
Version 18.0.1 7
Version 18.1 7
Version 18.1.1 8
Version 18.2 8
Version 18.2.1 9
Version 18.3 9
Version 18.3.1 10
Version 18.3.2 10
Version 18.4 10
Version 18.4.1 11
Version 18.5 11
iOS 18 Migration 12
iPadOS 18 12
Highlights for iPad Developers 12
iPadOS 18 Migration 13
macOS 15 Sequoia 13
Version 15.0 13
Version 15.1 14
Further Versions (15.2 – 15.5) 14
macOS Sequoia Migration 14
watchOS 11 14
Version 11.0 14
watchOS 11 Migration 15
visionOS 2 16

Highlights for Developers 16
SwiftUI 16
New Features (iOS 18 / Xcode 16) 16
Breaking Changes SwiftUI iOS 18 19
Known Issues 20
Swift 6 20
New Features 20
Concurrency 21
Breaking Changes Swift 6 22
Xcode 16 22
IDE 22
Simulator 23
Debugging 23
Testing 23
Apple Intelligence 23
Writing Tools 24
App Intents – Improved Siri Integration 24
Foundation Models Framework (Not in iOS 18) 24
App Store & TestFlight & AppStore Connect 24
App Store SDK Deadlines 24
Privacy Manifest – Required Field 24
TestFlight 25
Known Issues 25
Swift 6 / Xcode 16 25
Apple Intelligence 25
UIKit / SwiftUI 26
Migration to Apple 18 26
Migration Checklist 26
Common Migration Issues 27
Conclusion 27
Archive 27
iOS 18 Beta History 27
iPadOS 18 Beta History 28
macOS 15 Sequoia Beta History 28

watchOS 11 Beta History 28

Apple Platforms 18

Status: June 2026 – Fully Released iOS 18 and all Apple Platforms 2024 are completely in production. iOS 18.5 is the last released version of this generation. iOS 26 (formerly iOS 19) is the successor (released September 2025).

Source: Apple Developer Releases, Apple iOS 18 Release Notes

Overview

Platform	Final Version	Successor
iOS / iPadOS	18.5	iOS/iPadOS 26
macOS	15.5 (Sequoia)	macOS 26 (Tahoe)
watchOS	11.5	watchOS 26
tvOS	18.5	tvOS 26
visionOS	2.4	visionOS 26
Xcode	16.4	Xcode 26
Swift	6.1	Swift 6.3

Key Developer Deadlines:

Date	Event
September 16, 2024	iOS 18.0 initial release
April 28, 2025	iOS 18 SDK (Xcode 16) required for new app submissions
October 2025	iOS 18 SDK required for all app updates
May 2026	iOS 18.5 – last version of this generation

Source: Apple SDK Requirements

---

iOS 18

Version 18.0

Released: September 16, 2024 **Source:** Apple iOS 18 Release Notes

Home Screen – Free Icon Placement

What's new: For the first time, app icons can be placed freely on the Home Screen — no longer constrained to a top-left grid. Icons can be positioned in the center, at the bottom, or anywhere on the screen.

What developers need to consider: No direct API intervention required. However, apps must ensure their icons look great in all system contexts.

Dark Mode and Tinted Icons: - New capability: The system automatically applies tinting to app icons - Developers can provide a custom dark mode variant and a tinted variant in the App Asset Catalog - Filenames in Asset Catalog: `AppIcon`, `AppIcon-dark`, `AppIcon-tinted`

```
<!-- Info.plist – no changes required -->
<!-- Asset Catalog: populate AppIcon.appiconset with dark/tinted variants -->
```

Why it matters: Without custom dark/tinted variants, iOS 18 auto-generates a desaturated version. This usually looks poor. Custom variants are essential for a polished appearance.

Testing: 1. Set device to Dark Mode 2. Home Screen → Customize → Select Dark 3. Also test Tinted mode

---

Control Center – Full Customization

What's new: Control Center is fully customizable. Users can add, remove, and reorder controls. Third-party developers can provide their own controls.

ControlWidgetButton / ControlWidgetToggle – new API:

```swift
import WidgetKit
import AppIntents

struct MyToggleControl: ControlWidget {
    var body: some ControlWidgetConfiguration {
        StaticControlConfiguration(
            kind: "com.example.my-toggle",
            provider: MyToggleProvider()
        ) { value in
            ControlWidgetToggle(
                "My Toggle",
                isOn: value,
                action: MyToggleIntent()
            ) { isOn in
                Label(isOn ? "On" : "Off", systemImage: isOn ? "power" : "pow
eroff")
            }
        }
    }
}
```

What developers need to consider: - Import: `WidgetKit` and `AppIntents` - Implement `ControlWidget` protocol - Use `StaticControlConfiguration` or `AppIntentControlConfiguration` - Control must live inside a Widget Extension - Deployment Target: iOS 18.0+ - Key types: `ControlWidgetButton` (one-time action) or `ControlWidgetToggle` (stateful)

Why it matters: For the first time, direct system access via Control Center without Siri. Perfect for smart home apps, timers, and app shortcuts.

Reference: WidgetKit Documentation

Lock Screen – Replaceable Controls

What's new: The camera and flashlight buttons on the Lock Screen can now be replaced by custom app controls (requires `ControlWidget` implementation).

What developers need to consider: - Same ControlWidget API as Control Center - Lock Screen controls appear automatically when the widget has `supportsLockScreen = true`

RCS Messaging

What's new: iOS 18 supports RCS (Rich Communication Services) alongside SMS/MMS. This enables read receipts, typing indicators, and improved media transfer even when communicating with Android users.

What developers need to consider: - No direct API for app developers - Messages framework: no breaking changes - `MessageUI` framework unchanged

---

Messages – New Features

- **Scheduled Sending:** Schedule messages to be sent at a later time
- **Text Effects:** Animated text effects (Big, Small, Shake, Explode, Invisible, etc.)
- **Satellite Connectivity:** Emergency messages via satellite (iPhone 14+)
- **RCS:** Higher quality media sharing

For developers: `MessageUI` framework — no breaking changes. New text effects are system-native and not programmatically controllable.

Photos – Complete Redesign

What's new: The Photos app was completely rebuilt. New views: topic-based library, improved search, pinnable views.

What developers need to consider: - `PhotosUI` framework: `PHPickerViewController` and `PhotosPicker` (SwiftUI) unchanged - No breaking changes in the public API - PHAsset fetch operations still valid

---

Mail – Categories

What's new: Mail now automatically sorts incoming messages into categories: **Primary**, **Transactions**, **Updates**, **Promotions**.

What developers need to consider: - No API for developers - Custom mail apps: unchanged - Marketing emails automatically land in "Promotions" — may affect open rates

---

Game Mode

What's new: Game Mode automatically reduces background activity during gameplay: CPU/GPU are prioritized, AirPods audio latency is reduced, and game controller polling rate is increased.

API for developers:

```
import GameKit

// Query Game Mode status
let mode = GKGameActivityType.current
// .playing, .notPlaying, .unknown

// No explicit opt-in needed – iOS automatically recognizes active game sessi
ons
```

Why it matters: Games benefit automatically without code changes. Developers can use GKAccessPoint to optimally integrate Game Center features.

Passwords App – New Standalone App

What's new: Passwords were extracted from Settings into their own standalone app.

What developers need to consider: - AutoFill via `ASAuthorizationPasswordProvider` unchanged - `ASCredentialProviderExtension` for password manager apps unchanged - Passkey support still via `AuthenticationServices`

SharePlay – Remote Control

What's new: SharePlay now allows remote control of iOS screens (with user permission).

Version 18.0.1

Released: October 3, 2024 **Focus:** Bug fixes

- Fix: Responsiveness to touch input in certain situations
- Fix: Visual Voicemail setup could get stuck
- Fix: iPhone 16 Camera Control in certain interactions

For developers: No API changes.

Version 18.1

Released: October 28, 2024 **Focus:** Apple Intelligence – Phase 1

Source: Apple iOS 18.1 Release Notes

Apple Intelligence – First Features

Supported devices for Apple Intelligence: - iPhone 15 Pro / iPhone 15 Pro Max - iPhone 16 / 16 Plus / 16 Pro / 16 Pro Max - iPad with M1 chip or newer - Mac with Apple Silicon - **Not supported:** iPhone 15 / iPhone 15 Plus and older iPhones

Writing Tools: System-wide AI writing assistance in all text input fields. Functions: proofreading, rewriting (Neutral, Professional, Friendly, Concise), summarizing.

What developers need to consider: Writing Tools appear automatically in `UITextView` and `NSTextView` in app contexts. Opt-out available:

```
// UITextView — disable Writing Tools
textView.writingToolsBehavior = .none
// or .limited (transformation only, no AI-generated text)
```

Photo Cleanup: Objects in photos can be removed using AI (similar to Google Magic Eraser).

Notification Summaries: iOS 18.1 summarizes multiple notifications from an app into a single summary.

What developers need to consider: - No direct API control over Notification Summaries - Ensure notification content is semantically meaningful and safely summarizable - Critical notifications (`interruptionLevel: .critical` or `.timeSensitive`) are not summarized

```
// Mark notifications as time-sensitive – not summarized
content.interruptionLevel = .timeSensitive
```

Smart Reply in Messages and Mail: AI-generated reply suggestions.

Version 18.1.1

Released: November 5, 2024 **Focus:** Security updates

- Security vulnerability in JavaScriptCore (WebKit): fixes
- No API changes for developers

Version 18.2

Released: December 11, 2024 **Focus:** Apple Intelligence – Phase 2

Source: Apple iOS 18.2 Release Notes

ChatGPT Integration in Siri

What's new: Siri can now use ChatGPT (from OpenAI) for complex requests. Users decide whether to forward queries to ChatGPT.

What developers need to consider: - No direct API for third-party LLM integration in Siri - App Intents remain the correct way to implement Siri integration

Genmoji – Custom Emoji

What's new: Users can create custom AI-generated emoji (Genmoji) based on their description.

What developers need to consider: - Genmoji are transmitted as image attachments in messages - Apps that process `NSAttributedString` with emoji should check for compatibility - `UITextView` with Genmoji: rendered correctly

Image Playground

What's new: New app: Image Playground. Users create AI images in three styles (Animation, Illustration, Sketch).

Integration in your own apps:

```
import ImagePlayground

// Open Image Playground sheet from an app
struct ContentView: View {
    @State private var showImagePlayground = false
    @State private var generatedImage: UIImage?

    var body: some View {
        Button("Generate Image") {
            showImagePlayground = true
        }
        .imagePlaygroundSheet(
            isPresented: $showImagePlayground,
```

```
            concept: "A friendly robot holding coffee"
        ) { url in
            // Generated image as URL
            if let image = UIImage(contentsOfFile: url.path) {
                generatedImage = image
            }
        }
    }
}
```

What developers need to consider: - Import `ImagePlayground` framework - Deployment Target: iOS 18.2+ - Users must have Apple Intelligence enabled - `ImagePlaygroundViewController` available for UIKit apps

Reference: ImagePlayground Framework

Visual Intelligence (iPhone 16)

What's new: Camera Control on iPhone 16 opens Visual Intelligence: recognition of objects, text, plants, animals, restaurant info (via Yelp/Google), QR codes.

What developers need to consider: - No direct API for Visual Intelligence - Apps can appear in Visual Intelligence results via App Intents - `SFSafariViewController` and web links are opened from Visual Intelligence results

---

Version 18.2.1

Released: January 6, 2025 **Focus:** Bug fixes

- Fix: iPhone with iOS 18.2 could not make/receive calls in rare cases
- Security fixes
- For developers: no API changes

Version 18.3

Released: January 27, 2025 **Focus:** Apple Intelligence Corrections, Bug Fixes

Source: Apple iOS 18.3 Release Notes

Notification Summaries – Disabled by Default for News Apps

Background: Following complaints about incorrectly summarized news headlines (particularly misinformation), Apple disabled Notification Summaries by default for news and entertainment apps.

What developers need to consider: - Your own app's notifications are not directly affected - Design notifications with clear, unambiguous meaning — summaries can be

semantically incorrect - **Recommendation:** Mark time-sensitive and safety-relevant notifications with `.timeSensitive` or `.critical`

```
let content = UNMutableNotificationContent()
content.title = "Balance Warning"
content.body = "Your account has less than $50"
content.interruptionLevel = .timeSensitive // Not summarized
```

Memory Movie Improvements

Improved film generation through Apple Intelligence in the Photos app.

AirPlay Hotel Support

Hotels can offer AirPlay without app installation on their Apple TV devices.

Version 18.3.1

Released: March 12, 2025 **Focus:** Security updates

- Critical fix: USB Restricted Mode could be bypassed under certain circumstances (CVE-2025-24200)
- For developers: no API changes
- **Recommendation:** Promptly test and submit app updates with this minimum requirement

---

Version 18.3.2

Released: March 12, 2025 **Focus:** WebKit/Safari security vulnerability

- Fix: Cross-site scripting vulnerability in WebKit (CVE-2025-24201)
- Affects all apps using `WKWebView` or `SFSafariViewController`
- **Important for developers:** Apps with web content should test on iOS 18.3.2+

Version 18.4

Released: March 31, 2025 **Focus:** Apple Intelligence Extensions, Priority Notifications

Source: Apple iOS 18.4 Release Notes

Priority Notifications

What's new: Apple Intelligence automatically prioritizes notifications. Important notifications appear at the top; less important ones are summarized.

What developers need to consider: Developers can explicitly control notifications using interrupt levels:

```
// Interrupt levels for notifications
content.interruptionLevel = .passive       // Not shown when screen is off
content.interruptionLevel = .active        // Default
content.interruptionLevel = .timeSensitive // Appears despite Focus mode
content.interruptionLevel = .critical      // Ignores muting (requires entitl
ement)
```

Apple Intelligence – More Languages

- French, German, Italian, Portuguese (Brazil), Spanish, Japanese, Korean, Chinese (Simplified/Traditional)
- **For developers:** Writing Tools and Siri integrations are now usable in more languages

Additional New Features

- **Satellite Messages in all countries** with iPhone-14-compatible satellite network
- **CarPlay improvements:** Recently used apps prominently displayed
- **TV app:** Content improvements
- **Accessibility:** Improved Live Captions

Version 18.4.1

Released: April 16, 2025 **Focus:** Security updates

- Security fixes for critical WebKit vulnerabilities (actively exploited)
- **Recommendation:** Immediate response — test apps with WKWebView
- CarPlay connectivity issue fixed

Version 18.5

Released: May 12, 2025 **Focus:** Bug fixes, security updates, Pride wallpaper

- New Pride wallpapers for June 2025
- Security fixes (WebKit, Kernel)
- Mail stability improvements
- **Last version of the iOS 18 generation**

For developers: No API changes. iOS 18.5 is the final stable version before iOS 26.

---

iOS 18 Migration

What Developers Need to Adjust

1. App Icon – Dark / Tinted Variants

```
Asset Catalog → AppIcon → Add Dark Appearance
Asset Catalog → AppIcon → Add Tinted Appearance
```

2. Control Center Widget (optional but recommended) - New Widget Extension with `ControlWidget` protocol - Users can place custom app controls in Control Center

3. Swift 6 Concurrency (Breaking Change) - Gradual migration via `SWIFT_STRICT_CONCURRENCY` build setting - Start: `minimal` → `targeted` → `complete`

4. Writing Tools Opt-out (if necessary)

```
textView.writingToolsBehavior = .none    // Disable
textView.writingToolsBehavior = .limited // Transformation only
textView.writingToolsBehavior = .default // System default
```

5. Set Notification Interrupt Levels - `timeSensitive` for important notifications (bypass summarization) - `critical` for life-critical alerts (requires entitlement)

---

iPadOS 18

iPadOS 18 shares the iOS 18 version history. Additional iPad-specific features:

Highlights for iPad Developers

Tab Bar + Sidebar (SwiftUI / UIKit)

What's new: iPad apps can now adaptively switch between Tab Bar (phone-style) and Sidebar (Mac-style).

SwiftUI:

```
TabView {
    Tab("Home", systemImage: "house") {
        HomeView()
    }
    Tab("Search", systemImage: "magnifyingglass") {
        SearchView()
    }
}
.tabViewStyle(.sidebarAdaptable) // Automatically Tab or Sidebar depending on
 device
```

UIKit:

```
// UITabBarController with UITabBarControllerDelegate
// New property: tabBarController.mode = .tabSidebar
```

Why it matters: Apps that don't use `.tabViewStyle(.sidebarAdaptable)` look suboptimal on iPad. Mac Catalyst apps benefit directly.

Reference: TabView sidebarAdaptable

iPhone Mirroring

Macs with macOS 15 Sequoia can mirror and control the iPhone screen. For developers: apps don't need extra adjustments for the Mac.

Calculator App for iPad (first time)

No developer impact.

iPadOS 18 Migration

- Implement `TabView.sidebarAdaptable`
- Review `UITabBarController` customizations
- iPhone Mirroring: test apps for compatibility with mouse/trackpad gestures

---

macOS 15 Sequoia

Released: September 16, 2024 **Source:** Apple macOS Sequoia Release Notes

Version 15.0

Highlights:

iPhone Mirroring

- Mac shows iPhone screen in a window
- Full mouse/keyboard control of iPhone from Mac
- Drag & Drop between Mac and iPhone mirror (18.1+)
- For app developers: no code changes needed

Window Tiling

- Windows can be automatically arranged side-by-side via key+drag or menu
- Similar to Windows Snap feature
- **For developers:** `NSWindow` handling unaffected

Safari 18

- New **Highlights** feature: AI-extracted key information from a website
- **Web Inspector:** New tools for responsive design
- **Reader Mode:** AI summary at the top

What web developers need to consider: - `<meta name="theme-color">` is used by Safari 18 on macOS for window chrome coloring - `View Transitions API` now supported - Scroll-animated elements: improved performance

Passwords App

Identical to iOS 18 — standalone app for passwords and passkeys.

Apple Intelligence on Mac

From macOS 15.1: Apple Intelligence available on M1+ Macs. Same features as iOS 18.1.

Version 15.1

Released: October 28, 2024 (simultaneous with iOS 18.1)

- Apple Intelligence enabled
- Writing Tools system-wide
- Siri improved

Further Versions (15.2 – 15.5)

Parallel to iOS 18 updates. Same Apple Intelligence extensions.

macOS Sequoia Migration

Minimum macOS for Xcode 16: macOS 14.5 (Sonoma) **Minimum macOS for Xcode 16.3+:** macOS 15.0

- **macOS Deployment Target:** Apps targeting macOS 15 should consider Window Tiling and new UI conventions
- **Mac Catalyst:** Test `.tabViewStyle(.sidebarAdaptable)`
- **Menu Bar Extras:** No breaking changes

---

watchOS 11

Released: September 16, 2024 **Source:** Apple watchOS 11 Release Notes

Version 11.0

Highlights:

Training Load & Vitals App

New health apps directly on Apple Watch. Relevant for developers: new `HealthKit` APIs.

New HealthKit metrics:

```
import HealthKit

// New types in watchOS 11 / iOS 18
let trainingLoad = HKQuantityType(.runningStrideLength)
let vo2MaxEstimate = HKQuantityType(.vo2Max)
```

```
// Vitals App data queries (new)
let physicalEffort = HKQuantityType(.physicalEffort)
```

Double Tap – Extended

Double Tap gesture on Apple Watch Series 9+ can now be used by third-party apps.

```
// watchOS 11: Double Tap in Custom Views
struct ContentView: View {
    @State private var doubleTapCount = 0

    var body: some View {
        Text("Taps: \(doubleTapCount)")
            .handGestureShortcut(.primaryAction) {
                doubleTapCount += 1
            }
    }
}
```

Why it matters: Programmatically usable for custom Watch apps for the first time.

Smart Stack

Watch Face Smart Stack automatically shows relevant widgets. Apps can set `WidgetKit` relevance scores:

```
WidgetCenter.shared.setCurrentConfigurations(...)
// Set TimelineEntry with .relevance
TimelineEntryRelevance(score: 10, duration: 3600)
```

Live Activities on Apple Watch

Live Activities (since iOS 16.1) are now displayed on watchOS 11 as well.

```
// Existing Live Activity – no code changes needed
// watchOS 11 automatically shows them in the Smart Stack
```

What developers need to consider: - Live Activity `ContentState` is displayed on the watch - Consider compact layout for Watch display

Pause & Resume Workouts

Watch workouts can now be interrupted and resumed. New `HealthKit` API.

watchOS 11 Migration

- Implement `handGestureShortcut(.primaryAction)` for Double Tap
- HealthKit data: consider new types in queries
- Live Activity: optimize compact layout for Watch

---

visionOS 2

Released: September 16, 2024 **Source:** Apple visionOS 2 Release Notes

Highlights for Developers

Volumetric Apps – Improved

```
// RealityKit / SwiftUI: New gesture APIs
// RealityView with improved Entity manipulation
```

SharePlay – Persona

Users can show their Spatial Persona during SharePlay. For developers: no code changes for existing SharePlay apps.

Travel Mode

Improved usage in transportation (airplane, train). For developers: no adjustments needed.

Wide and Ultra-Wide Windows

```
// New .windowStyle options
WindowGroup {
    ContentView()
}
.defaultSize(width: 1.5, height: 1.0, depth: 0, in: .meters)
.windowStyle(.volumetric)
```

SwiftUI

New Features (iOS 18 / Xcode 16)

TabView – Sidebar Adaptable

Already described under iPadOS 18. Critical for all iPad/Mac apps.

```
struct ContentView: View {
    var body: some View {
        TabView {
            Tab("Home", systemImage: "house") { HomeView() }
            Tab("Search", systemImage: "magnifyingglass") { SearchView() }
            Tab("Profile", systemImage: "person") { ProfileView() }
        }
        .tabViewStyle(.sidebarAdaptable)
    }
}
```

@Entry Macro

New shorthand for `EnvironmentValues`, `FocusedValues`, and `Transaction`.

Without @Entry (old way):

```swift
struct MyValueKey: EnvironmentKey {
    static let defaultValue: String = "Default"
}
extension EnvironmentValues {
    var myCustomValue: String {
        get { self[MyValueKey.self] }
        set { self[MyValueKey.self] = newValue }
    }
}

// With @Entry (iOS 18):
extension EnvironmentValues {
    @Entry var myCustomValue: String = "Default"
}
```

Saves: The `EnvironmentKey` boilerplate struct is no longer needed.

@Previewable Macro

```swift
#Preview {
    @Previewable @State var isOn = false
    Toggle("Setting", isOn: $isOn)
}
```

Before: State had to be extracted into a separate PreviewWrapper view. `@Previewable` allows direct use in `#Preview` blocks.

MeshGradient

New, high-quality gradient type with control points on a 2D grid.

```swift
struct ContentView: View {
    var body: some View {
        MeshGradient(
            width: 3,
            height: 3,
            points: [
                [0, 0], [0.5, 0], [1, 0],
                [0, 0.5], [0.5, 0.5], [1, 0.5],
                [0, 1], [0.5, 1], [1, 1]
            ],
            colors: [
                .blue, .purple, .indigo,
                .orange, .pink, .red,
                .green, .teal, .cyan
            ]
        )
        .ignoresSafeArea()
    }
}
```

Why it matters: Enables high-quality, naturally-looking backgrounds without external assets.

Zoom Navigation Transition

```
struct ContentView: View {
    @Namespace var namespace
    @State var showDetail = false

    var body: some View {
        if !showDetail {
            Image("preview")
                .navigationTransition(.zoom(sourceID: "img", in: namespace))
                .onTapGesture { showDetail = true }
        } else {
            DetailView()
                .navigationTransition(.zoom(sourceID: "img", in: namespace))
        }
    }
}
```

ScrollView Improvements

```
@State private var position = ScrollPosition(edge: .top)

ScrollView {
    ForEach(items) { item in
        ItemRow(item: item)
    }
}
.scrollPosition($position)

// Scroll programmatically:
Button("To Top") {
    position.scrollTo(edge: .top)
}
```

PresentationSizing

```
.sheet(isPresented: $showSheet) {
    MySheetContent()
        .presentationSizing(.fitted)    // Adapts to content
        // .form → Form-like layout
        // .page → Full screen width
}
```

SF Symbol Effects

```
Image(systemName: "bell.fill")
    .symbolEffect(.bounce)

Image(systemName: "wifi")
    .symbolEffect(.variableColor.iterative.reversing)

Button("Load") { load() }
    .symbolEffect(.rotate, isActive: isLoading)
```

```swift
Image(systemName: "heart.fill")
    .symbolEffect(.breathe)

Image(systemName: "trash")
    .symbolEffect(.wiggle)
```

TextRenderer Protocol

```swift
struct GradientTextRenderer: TextRenderer {
    func draw(layout: Text.Layout, in ctx: inout GraphicsContext) {
        for line in layout {
            ctx.draw(line)
        }
    }
}

Text("Hello")
    .textRenderer(GradientTextRenderer())
```

Container Views

```swift
struct CardGroup<Content: View>: View {
    @ViewBuilder var content: Content

    var body: some View {
        Group(subviewsOf: content) { subviews in
            VStack {
                ForEach(subviews) { subview in
                    subview
                        .padding()
                        .background(Color.secondary.opacity(0.1))
                        .cornerRadius(12)
                }
            }
        }
    }
}
```

Breaking Changes SwiftUI iOS 18

Change	iOS 17	iOS 18
`onChange(of:)`	Second closure parameter optional	New signature: `(oldValue, newValue)`
`TabView` on iPad	Standard Tab Bar	`.sidebarAdaptable` recommended
`NavigationSplitView`	Stable	No breaking changes

onChange Breaking Change:

```swift
// Old (iOS 16/17 – deprecated)
.onChange(of: value) { newValue in ... }
```

```swift
// New (iOS 17+ / iOS 18 recommended)
.onChange(of: value) { oldValue, newValue in
    print("From \(oldValue) to \(newValue)")
}
```

Known Issues

- **MeshGradient on older hardware:** Performance may be worse on A12 chips and older
- **TabView Sidebar:** On iPhone, unchanged Tab Bar; Sidebar only on iPad/Mac

---

Swift 6

New Features

Strict Concurrency Checking (Complete)

Swift 6 changes the default for concurrency checking to `complete`. This means: **the compiler checks all potential data races at compile time**.

Migration Build Setting:

```
Build Settings → Swift Compiler – Language → Swift Concurrency Checking
minimal    → No warnings (backward compatible)
targeted   → Warnings for new code
complete   → Errors like Swift 6 (target)
```

Gradual migration: 1. Start with `SWIFT_STRICT_CONCURRENCY=minimal` 2. Switch to `targeted` and fix warnings 3. Raise to `complete` (equivalent to Swift 6) 4. Set `swift-tools-version: 6.0` in Package.swift for full Swift 6 semantics

Typed Throws

```swift
enum NetworkError: Error {
    case timeout
    case unauthorized
    case serverError(Int)
}

func fetchUser(id: Int) throws(NetworkError) -> User {
    guard id > 0 else { throw .unauthorized }
    // ...
}

do {
    let user = try fetchUser(id: 42)
} catch let error {
    switch error {
    case .timeout: handleTimeout()
    case .unauthorized: handleAuth()
    case .serverError(let code): handleServer(code)
```

```
    }
}
```

Noncopyable Types (~Copyable)

```
struct FileHandle: ~Copyable {
    let fd: Int32

    init(path: String) {
        fd = open(path, O_RDONLY)
    }

    deinit {
        close(fd)
    }
}
```

Swift Testing Framework

```
import Testing

@Suite("User Tests")
struct UserTests {

    @Test("User can be created")
    func testUserCreation() {
        let user = User(name: "Max")
        #expect(user.name == "Max")
    }

    @Test("Various age values", arguments: [0, 18, 65, 120])
    func testAge(age: Int) {
        let user = User(name: "Test")
        user.age = age
        #expect(user.age == age)
    }
}
```

Differences from XCTest: | Feature | XCTest | Swift Testing | |———|———|————| | Test marking | `func testXxx()` | `@Test` | | Assertion | `XCTAssertEqual(a, b)` | `#expect(a == b)` | | Error test | `XCTAssertThrowsError` | `#expect(throws:)` | | Parameterized | Manual | `arguments:` | | Suite | `class XCTestCase` | `@Suite struct` |

Reference: Swift Testing Documentation

Concurrency

Sendable Enforcement

```
// Solution 1: @MainActor
@MainActor
class SharedState {
    var data: [String] = []
```

```swift
}

// Solution 2: Actor
actor SharedState {
    var data: [String] = []
    func append(_ value: String) { data.append(value) }
}

// Solution 3: Sendable Value Type
struct SharedState: Sendable {
    let data: [String]
}
```

@MainActor Best Practices

```swift
@MainActor
class UserViewModel: ObservableObject {
    @Published var users: [User] = []

    func loadUsers() async {
        let fetched = await UserService.shared.fetch()
        users = fetched
    }
}
```

Breaking Changes Swift 6

Topic	Impact
Strict Sendable checking	Classes without `Sendable` or `@MainActor` cannot be passed across Tasks
Global mutable state	Global `var` must be `@MainActor` or `nonisolated(unsafe)`
Protocol conformances	Protocols with `Sendable` requirements strictly checked
Closures in Tasks	Must be `@Sendable`

Reference: Swift 6 Migration Guide

Xcode 16

IDE

Predictive Code Completion

Xcode 16 features new AI-based code completion directly in the IDE, running entirely locally (no internet required).

- Multi-line suggestions

- Context-sensitive: understands Swift types, frameworks, and project context
- No cloud upload of your codebase
- Automatic on Apple Silicon Macs

Explicit Modules

```
# Build Settings → Module Build (Explicit)
# SWIFT_MODULE_BUILD = explicit
```

20–40% faster builds for large projects.

Simulator

Simulator Push Notifications

```
xcrun simctl push booted com.example.app notification.apns

{
  "aps": {
    "alert": {
      "title": "Test Notification",
      "body": "Sent via Terminal"
    },
    "badge": 1,
    "sound": "default"
  }
}
```

Debugging

Thread Performance Checker – Extended

- Detects `@MainActor` violations at runtime
- Shows which function is running on the wrong thread
- Activation: Scheme → Diagnostics → Thread Performance Checker

Testing

Swift Testing Integration

```
@Test("Database test", .tags(.database), .serialized)
func testDatabase() async {
    // Executed serially within the suite
}

extension Tag {
    @Tag static var database: Self
    @Tag static var network: Self
}
```

Apple Intelligence

Availability: iOS 18.1+ (feature rollout across multiple versions)

Supported Devices: | Device | Apple Intelligence | |———|————————-| | iPhone 15 Pro / Pro Max | Yes (from iOS 18.1) | | iPhone 16 / Plus / Pro / Pro Max | Yes (active from 18.1) | | iPhone 14 and older | No | | iPhone 15 / Plus | No | | iPad with M1+ | Yes | | Mac with Apple Silicon | Yes (from macOS 15.1) |

Writing Tools

```
// UIKit
textView.writingToolsBehavior = .none

// SwiftUI
TextField("Input", text: $text)
    .writingToolsBehavior(.none)
```

App Intents – Improved Siri Integration

```
import AppIntents

struct CreateTaskIntent: AppIntent {
    static var title: LocalizedStringResource = "Create Task"

    @Parameter(title: "Task Title") var title: String

    func perform() async throws -> some IntentResult & ProvidesDialog {
        let task = try await TaskManager.shared.create(title: title)
        return .result(dialog: "Task \"\(task.title)\" has been created.")
    }
}
```

Reference: App Intents Documentation

Foundation Models Framework (Not in iOS 18)

The Foundation Models Framework was introduced in iOS 26. In iOS 18, access to the on-device LLM is only possible through Siri/Apple Intelligence features — no direct API for developers.

App Store & TestFlight & AppStore Connect

App Store SDK Deadlines

Date	Requirement
From April 28, 2025	New app submissions must use Xcode 16 (iOS 18 SDK)
From October 2025	All app updates must use Xcode 16

Privacy Manifest – Required Field

Apps using certain APIs must include a **Privacy Manifest** (`PrivacyInfo.xcprivacy`).

Affected APIs: `UserDefaults`, `NSFileManager` (certain methods), `NSProcessInfo`, `AppTrackingTransparency`, CoreData, CloudKit, Crashlytics and others.

```
<key>NSPrivacyTracking</key>
<false/>
<key>NSPrivacyCollectedDataTypes</key>
<array>
    <dict>
        <key>NSPrivacyCollectedDataType</key>
        <string>NSPrivacyCollectedDataTypeUserID</string>
        <key>NSPrivacyCollectedDataTypeLinked</key>
        <true/>
        <key>NSPrivacyCollectedDataTypeTracking</key>
        <false/>
        <key>NSPrivacyCollectedDataTypePurposes</key>
        <array>
            <string>NSPrivacyCollectedDataTypePurposeAppFunctionality</string>
        </array>
    </dict>
</array>
```

Reference: Privacy Manifest

TestFlight

- TestFlight for Mac: Generally available for macOS 12+
- Up to 10,000 external testers
- Xcode Cloud: Direct deployment to TestFlight from CI/CD pipelines

---

Known Issues

Swift 6 / Xcode 16

Issue	Affected Apps	Solution
`@Observable` + `@MainActor` conflicts	MVVM + Combine apps	Explicitly set `@MainActor` on `@Observable` class
Swift Testing + XCTest mix	Mixed test targets	Separate targets for each framework
Predictive Code Completion too aggressive	All apps	Disableable in Xcode Preferences

Apple Intelligence

Issue	Version	Status
Notification summaries with incorrect	18.1–18.2	Disabled for news apps in 18.3

Issue	Version	Status
information		
Writing Tools active in wrong contexts	18.1	API opt-out available
ChatGPT integration English-only (initial)	18.2	More languages in 18.4

UIKit / SwiftUI

Issue	Impact	Solution
`TabView` layout regression on iOS 18	Tab bar sometimes disappears	Explicitly set `tabViewStyle(.automatic)`
`onChange` deprecation warnings	Compiler warnings	Use new `(oldValue, newValue)` signature

---

Migration to Apple 18

Migration Checklist

Priority HIGH (Mandatory):

- ☐ **Install Xcode 16** (macOS 14.5 minimum)
- ☐ **Create Privacy Manifest** (`PrivacyInfo.xcprivacy`) – App Store required
- ☐ **App Icon Variants**: Add Dark Mode + Tinted variant in Asset Catalog
- ☐ **Test iOS 18 SDK Build**: Check UIKit apps for layout regressions
- ☐ **Update `onChange` signature**: Use new `(oldValue, newValue)` variant
- ☐ **Swift Concurrency Level**: Set build setting to `targeted` and fix warnings

Priority MEDIUM (Recommended):

- ☐ **Swift Testing**: Write new tests in Swift Testing instead of XCTest
- ☐ **TabView Sidebar**: Modernize iPad apps with `sidebarAdaptable`
- ☐ **Control Center Widget**: Provide app actions as `ControlWidget`
- ☐ **Expand App Intents**: Prepare for Apple Intelligence
- ☐ **Typed Throws**: Make error handling more precise

Priority LOW (Optional):

- ☐ **MeshGradient**: Enhance backgrounds
- ☐ **SF Symbol Effects**: Add animations
- ☐ **@Entry Macro**: Reduce EnvironmentValues boilerplate
- ☐ **@Previewable**: Simplify previews

Common Migration Issues

Issue 1: Too many Swift 6 errors

```
Set SWIFT_STRICT_CONCURRENCY=minimal → No errors
Gradually increase to targeted
```

Issue 2: Missing Privacy Manifest

```
File → New → Privacy Manifest (PrivacyInfo.xcprivacy)
Enter all used APIs with their purpose
```

Issue 3: Tab Bar disappears on iOS 18

```
.tabViewStyle(.automatic)
```

Issue 4: onChange warnings

```
// Old (Deprecated):
.onChange(of: value) { newValue in ... }
// New:
.onChange(of: value) { oldValue, newValue in ... }
```

Conclusion

iOS 18 (September 2024) marks the beginning of the Apple Intelligence era. Key takeaways for developers:

What stays the same: - UIKit and SwiftUI grow in parallel and converge - App Store fundamentals unchanged - Deployment target freedom remains

What has changed: - **Swift 6 Concurrency**: Largest syntactic change since Swift 4. Migration is time-consuming but safety-relevant. - **Apple Intelligence**: New surface for app integration via App Intents. Investing now pays off long-term. - **Privacy Manifest**: Mandatory for all new submissions — implement immediately. - **Control Center Widgets**: First-time deep system access for third parties.

Outlook: iOS 26 (September 2025, successor) brings the Foundation Models Framework for direct on-device LLM access and the Liquid Glass design system. Developers who have well-positioned their apps for iOS 18 have a solid foundation for migrating to iOS 26.

Archive

iOS 18 Beta History

Beta	Build	Date	Highlights
Beta 1	22A5282m	June 10, 2024	Initial announcement WWDC 2024

Beta	Build	Date	Highlights
Beta 2	22A5295h	July 1, 2024	Stability improvements
Beta 3	22A5307m	July 15, 2024	SwiftUI fixes
Beta 4	22A5316k	July 29, 2024	App Icon Dark/Tinted stabilized
Beta 5	22A5326g	August 12, 2024	Performance improvements
Beta 6	22A5337a	August 26, 2024	Final beta fixes
RC	22A3354	September 9, 2024	Release Candidate
Release	22A3354 / 22A3370	September 16, 2024	Public release

> **Note:** Build numbers from the beta phase are based on publicly available community reports and developer forums. For final build numbers, the Apple release documentation is authoritative.

iPadOS 18 Beta History

Identical to iOS 18 beta cycle — iPadOS and iOS share the beta cycle.

macOS 15 Sequoia Beta History

Beta	Date	Highlights
Beta 1	June 10, 2024	iPhone Mirroring first appearance
Beta 2	July 1, 2024	Window Tiling improved
Beta 3–6	July – August 2024	Stabilization
RC	September 9, 2024	Release Candidate
Release	September 16, 2024	macOS 15.0

watchOS 11 Beta History

Beta	Date	Highlights
Beta 1	June 10, 2024	Training Load, Double Tap API
Beta 2–6	July – August 2024	Stabilization
Release	September 16, 2024	watchOS 11.0