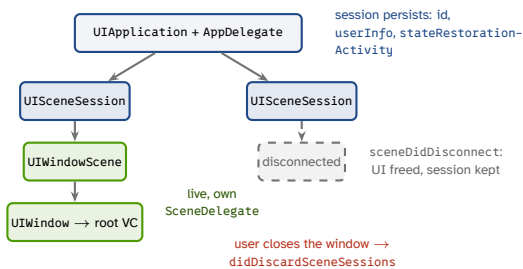
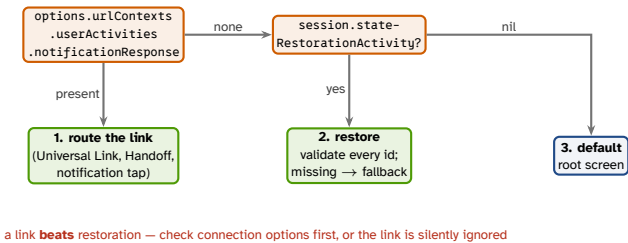


In one line: Since iOS 13 one **process** drives many **UI instances**: each window is a UIWindowScene with its own lifecycle, backed by a persistent UISceneSession. The system may **disconnect** a background scene to save memory and reconnect it later; restoration hands the scene back its own NSUserActivity so it rebuilds **UI state (ids, not model data)** — unless a deep link says where to go.

One process, many scenes



scene(_:willConnectTo:options:) — who decides the first screen?



How it works — scenes and windows

- AppDelegate** = process: didFinishLaunching, push token, services, configurationForConnecting, didDiscardSceneSessions. **SceneDelegate** = one window: willConnectTo, sceneDidBecomeActive/WillResignActive/WillEnterForeground/DidEnterBackground, sceneDidDisconnect, openURLContexts, continue (links while running). With scenes, the app delegate's applicationDidBecomeActive family is **not called**.
- Info.plist UIApplicationSceneManifest: UIApplicationSupportsMultipleScenes = YES or the app stays single-window on iPad. Open a window with UIApplication.shared.requestSceneSessionActivation(_:userActivity:options:errorHandler:): nil session = new window; an existing one = bring it forward — prefer that to spawning duplicates.
- Disconnect** ≠ **discard**: disconnect frees a background scene's UI (session kept, may reconnect via willConnectTo); **discard** = the user closed it — clean up per-window data (may arrive at the next launch).
- keyWindow / windows are ambiguous with several scenes: use view.window?.windowScene or connectedScenes.

How it works — iPad multitasking

- Split View, Slide Over, Stage Manager (iPadOS 16), external display: the **system** sizes your window, live. Adapt to **size classes** (horizontalSizeClass is .compact in a narrow pane on the biggest iPad), never to userInterfaceIdiom.
- React to traits: registerForTraitChanges (iOS 17; replaces traitCollectionDidChange), viewWillTransition(to:with:); UISplitViewController / NavigationSplitView collapse to a stack in compact width.
- Two windows share the process: a global “current document” breaks. Shared model at app level, per-window state in the scene.

Likely questions

- Where do you restore? — willConnectTo, after checking connection options.
- Save when? — stateRestorationActivity(for:), on entering background.
- Open item X in a new iPad window? — activity with X's id → requestSceneSessionActivation; SwiftUI openWindow(value:).

Example — restore, but let links win

```

func scene(_ scene: UIScene,
willConnectTo session: UISceneSession,
options: UIScene.ConnectionOptions) {
    guard let ws = scene as? UIWindowScene else { return }
    window = UIWindow(windowScene: ws)
    if let url = options.urlContexts.first?.url { router.open(url) }
    else if let ua = options.userActivities.first {
        router.continue(ua) // UL / Handoff
    } else if let id = session.stateRestorationActivity?.
        .userInfo[“itemID”] as? String, store.exists(id) {
        router.show(id) // validated
    }
    window?.rootViewController = router.root
    window?.makeKeyAndVisible()
}
// asked on entering the background: keep it cheap
func stateRestorationActivity(for scene: UIScene)
-> NSUserActivity? {
    let ua = NSUserActivity(activityType: “com.your.viewing”)
    ua.addUserInfoEntries(from: [“itemID”: router.currentID])
    return ua
}
  
```

Restoration + SwiftUI

- userInfo = property-list values: ids, offsets, tab, draft text. One NSUserActivity can also drive Handoff (isEligibleForHandoff) and Spotlight.
- Restores after a **system** kill. Discarded after the user closes the window / swipes the app away, or a crash during restore. **Test**: background the app, then Stop in Xcode, relaunch.
- Legacy: restorationIdentifier + encodeRestorableState(with:), opt-in shouldSaveSecureApplicationState.
- SwiftUI**: WindowGroup (multi-instance on iPadOS/macOS); @Environment(\.openWindow) → openWindow(id:) / openWindow(value:). @SceneStorage(“k”) = small per-scene values, restored automatically; @AppStorage = one UserDefaults for all windows. scenePhase is per scene (combined when read in App).

Interview traps

- ❗ Archiving model objects into restoration — store **ids** and re-fetch; the record may be gone.
- ❗ @AppStorage for per-window UI state: two windows clobber each other. @StateObject inside WindowGroup = one per window.
- ❗ NavigationPath is not Codable itself — persist path.codable (a CodableRepresentation) as Data.

Remember

Session persists, scene comes and goes · link > restore > default · ids, not models.