Live Activities — Technical Brief

Home Assistant iOS Companion · Branch: feat/live-activities · March 2026

iOS 17.2+ | ActivityKit | Cross-platform | Server-side PRs open

This document outlines the design decisions, architecture, and platform considerations behind the iOS Live Activities implementation. It gives reviewers full context without having to reverse-engineer intent from the code.

What This Is

Live Activities let Home Assistant push real-time state to the **iOS Lock Screen** and **Dynamic Island** — the same way your phone shows a flight tracker or a food delivery countdown.

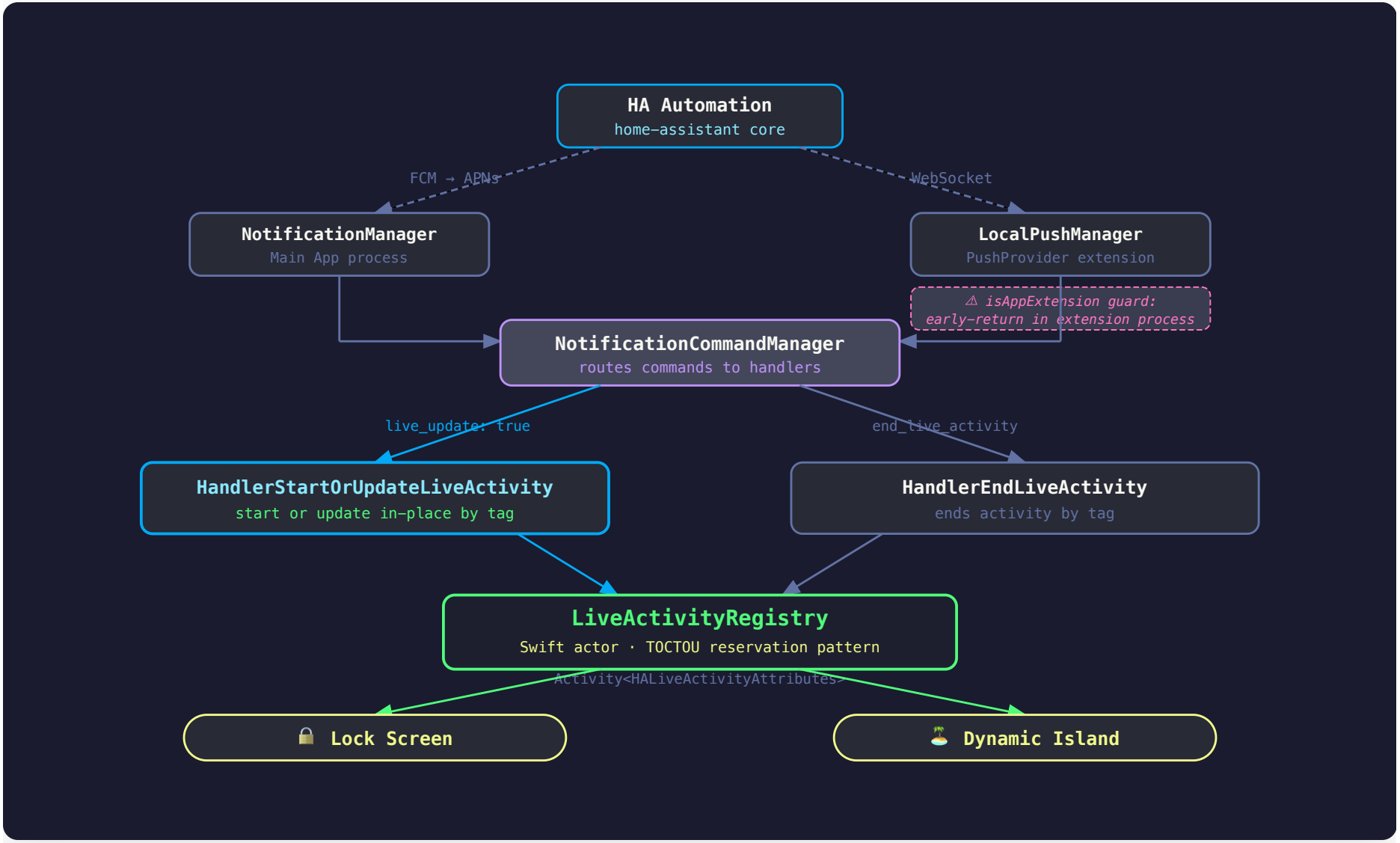
USER EXPERIENCE

A user sets up an automation in HA exactly like they would for any notification, adds `live_update: true` to the payload, and the companion app handles everything: starting the activity, keeping it updated via APNs, and ending it when HA says to.

CROSS-PLATFORM PARITY

Field names are **intentionally identical** to Android's Live Notifications API. Both platforms share the same `live_update: true` trigger — one YAML block targets iOS 17.2+ and Android 16+ with no platform-specific keys.

Architecture



🔒 **Why a Swift `actor` for the registry?**

Push notifications can arrive on multiple background threads simultaneously. Two notifications with the same `tag` arriving 5ms apart would both try to call `Activity.request()` — without an actor, that's a race condition leading to duplicate activities. The actor serializes access and uses a reservation pattern (claim the tag slot before awaiting the async `request()` call) to close the TOCTOU window.

⚠️ **Why `isAppExtension` guarding in handlers?**

The `PushProvider` runs in a *separate OS process* that cannot call ActivityKit. Both processes initialize `NotificationCommandManager` — the handlers detect the extension context and early-return, letting the OS re-deliver to the main app. Without this, ActivityKit calls from the extension would fail silently at runtime.

Data Model

🔴 **Wire-format frozen post-ship**

`HALiveActivityAttributes` is the APNs `attributes-type` identifier — the string in push-to-start payloads must exactly match the Swift struct name. New optional fields can be added;
nothing can be renamed or removed
without breaking in-flight activities.

HALiveActivityAttributes.swift

```swift
// HALiveActivityAttributes.swift — the wire format. Never rename post-ship.
public struct HALiveActivityAttributes: ActivityAttributes {
    public let tag: String      // unique activity ID, same as Android tag
    public let title: String    // static, set at creation

    public struct ContentState: Codable, Hashable {
        public var message: String
        public var criticalText: String?
        public var progress: Int?
        public var progressMax: Int?
        public var chronometer: Bool?
        public var countdownEnd: Date?
        public var icon: String?       // MDI slug: "mdi:washing-machine"
        public var color: String?      // hex: "#2196F3"
    }
}
```

iOS Version Strategy

Feature	Minimum iOS	Notes
Live Activities (basic)	`16.1`	`ActivityKit` introduced
`ActivityContent` / stale date	`16.2`	Required for `staleDate` and `relevanceScore`
Push-to-start (remote start)	**`17.2 ← our gate`**	`pushToStartTokenUpdates`
`frequentPushesEnabled`	`17.2`	User toggle for high-frequency updates
Deployment target	`15.0`	Unchanged — `#available` guards throughout

ℹ️ **Why 17.2?**

Push-to-start (the primary server-side start mechanism) requires iOS 17.2. Gating the entire feature at 17.2 means a single `#available(iOS 17.2, *)` check covers all Live Activity APIs — `supports_live_activities`, `frequentPushesEnabled`, push-to-start token, and all ActivityKit usage — eliminating the nested 16.2/17.2 check pattern.

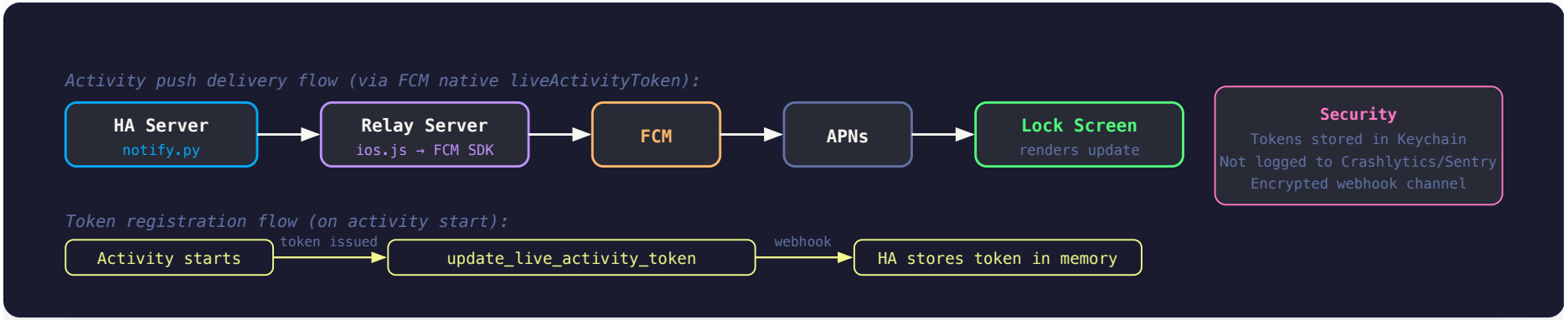
✅ **Forward compatibility**

`#available(iOS 17.2, *)` correctly covers 18.x, 19.x, etc. by Swift's availability model.

📱 **iPad**

`areActivitiesEnabled` is always `false` on iPad — Apple system restriction. Registry silently no-ops. Settings shows "Not available on iPad". HA receives `supports_live_activities: false`.

Push Token Flow



Push-to-start (iOS 17.2+): A single token lets HA start a brand-new Live Activity without the app open. Stored in Keychain, reported via the registration update payload. Push-to-start from a terminated app has ~50% success rate due to iOS power management — it's best-effort, not the primary flow.

Notification Payload → Live Activity Mapping

🔄 **One YAML block, both platforms**

`clear_notification` + `tag` ends both the Live Activity AND the delivered `UNNotification` with the same identifier in a single YAML block.

Home Assistant Automation YAML

```yaml
# This YAML works on Android 16+ and iOS 17.2+
data:
  title: "Washing Machine"
  message: "45 minutes remaining"
  data:
    tag: washer_cycle           # same tag = update in-place on both platforms
    live_update: true           # Android 16+ and iOS 17.2+
    critical_text: "45 min"     # Dynamic Island compact trailing / Android status bar
    progress: 2700
    progress_max: 3600
    chronometer: true
    when: 2700                  # seconds remaining
    when_relative: true         # treat as duration from now, not Unix timestamp
    notification_icon: mdi:washing-machine
    notification_icon_color: "#2196F3"
        # Android-only fields below — iOS silently ignores them
    alert_once: true
    sticky: true
    visibility: public

# Dismiss — identical on both platforms:
message: clear_notification
data:
  tag: washer_cycle
```

Companion Server-Side PRs

The iOS client is complete. End-to-end push updates require corresponding server-side work — all PRs are now open:

Component	Status
`supports_live_activities` field in device registration	✓ core#166072
`update_live_activity_token` webhook handler	✓ core#166072
`live_activity_dismissed` webhook handler	✓ core#166072
`notify.py` routing: includes `live_activity_token` alongside FCM token	✓ core#166072
Relay server: Live Activity delivery via FCM `apns.liveActivityToken`	✓ relay#278

ℹ️ The relay server uses FCM's native `apns.liveActivityToken` field (Firebase Admin SDK v13.5.0+) — FCM automatically sets `apns-push-type: liveactivity` and handles APNs delivery. No custom APNs client, JWT signing, or environment routing needed. The iOS client is fully functional for
local push (WebSocket, LAN)
and manual testing via the in-app debug scenarios.

Rate Limiting (iOS 18)

⚡ **~15 second minimum between rendered updates in iOS 18**
(down from ~1 Hz in iOS 17). The OS silently accepts `Activity.update()` calls above this rate — the HTTP 200 response comes back, but the device ignores the update. The simulator renders all updates regardless.

Automations should trigger on **state change events**, not polling timers. A sensor updating at 1 Hz should not fire a Live Activity push on every reading.

Key Files

```
Sources/Shared/LiveActivity/
  ├── HALiveActivityAttributes.swift   ← wire format — frozen post-ship
  └── LiveActivityRegistry.swift       ← Swift actor, TOCTOU reservation pattern

Sources/Extensions/Widgets/LiveActivity/
  ├── HALiveActivityConfiguration.swift ← ActivityConfiguration + DI island
  ├── HALockScreenView.swift            ← Lock Screen (max 160pt height)
  └── HADynamicIslandView.swift         ← compact / minimal / expanded

Sources/Shared/Notifications/NotificationCommands/
  └── HandlerLiveActivity.swift         ← start/update and end handlers

Sources/App/Settings/LiveActivity/
  └── LiveActivitySettingsView.swift    ← status, active activities, debug scenarios

Tests/Shared/LiveActivity/
  ├── MockLiveActivityRegistry.swift
  ├── HandlerLiveActivityTests.swift              ← 36 tests
  └── NotificationsCommandManagerLiveActivityTests.swift ← 9 tests
```