← All press kits ECOSYSTEM

# The Intelligent Notification Ecosystem

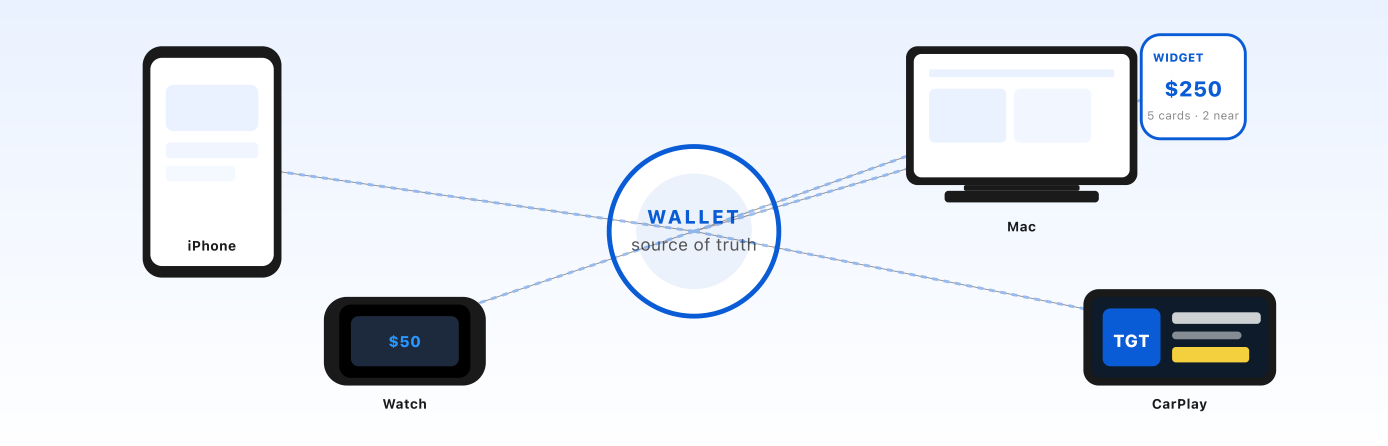
A press-kit deep-dive on how iPhone, Apple Watch, Dynamic Island, Widgets, Live Activities, and CarPlay cooperate, instead of competing, to make a gift-card wallet feel alive.

THE ONE-SENTENCE VERSION

Every Apple surface CardCue Pro touches has a single job, a single source of truth (the NotificationConductor), and a shared memory (App Group), so the same alert never fires twice, never fires in the wrong place, and never fires at the wrong time.

~220 lines · iPhone · Watch · Dynamic Island · Widgets · CarPlay

Read the white paper → Source markdown ↓




*One wallet, rendered wherever the user is paying attention — phone, wrist, desk, car, lock screen.*

## 1. The problem with most notification systems

Open any "cross-device" app today and you'll find the same pattern: each surface fires its own alerts. Your phone buzzes. Your watch buzzes a beat later because WatchConnectivity re-delivered the same payload. Your car's CarPlay dashboard shows a duplicate banner. Your Dynamic Island flashes for a card you already used. A widget reloads for the tenth time in an hour.

That is the fastest path to the "**immediate thumbs-down uninstall reaction**": users don't read a notification that has already betrayed their trust.

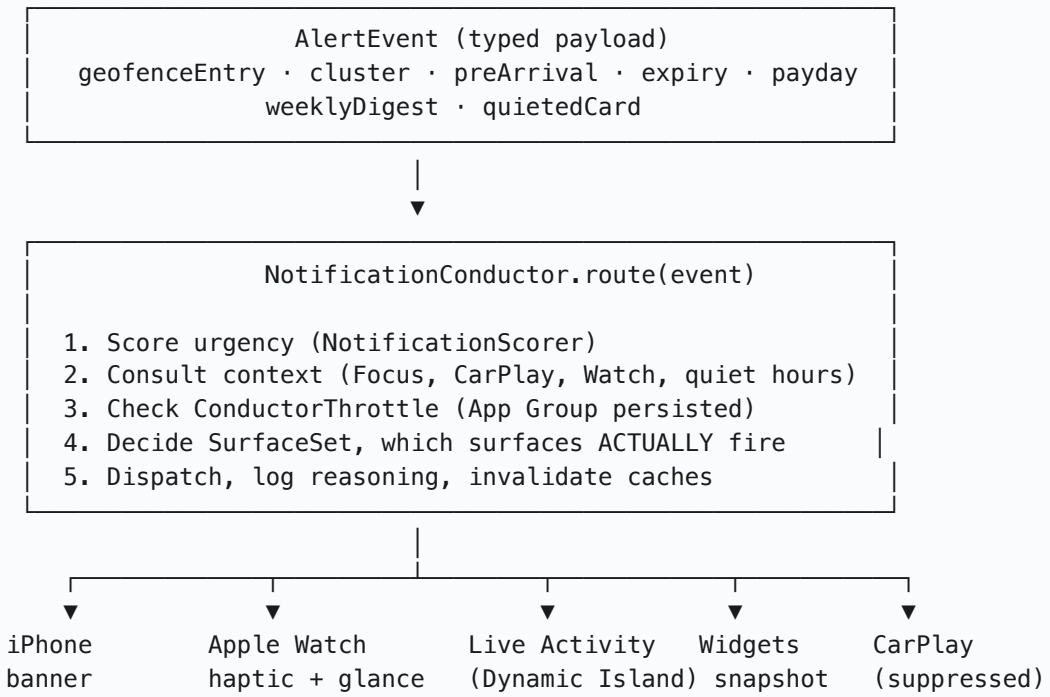
CardCue Pro was designed specifically to avoid that. Our wager: a gift-card wallet lives or dies by whether its nudges feel *helpful* instead of *noisy*. So we built an ecosystem where the surfaces coordinate before they speak.

The principle the Conductor exists to enforce has a name: the **Right-Moment Engagement Thesis**. Quiet by default. Speak once, when the moment is right. A moment is right when the user is on foot, within waking hours, near a store where the card is useful, hasn't used that card recently, and the weather won't punish the walk.

Anything less than that coincidence of signals and the app stays silent. The Conductor is the machine that checks those conditions and honors the silence when they aren't met.

## 2. The conductor pattern

At the center of CardCue Pro sits a single file, `Services/NotificationConductor.swift`, that every outbound alert flows through. Nothing writes directly to `UNUserNotificationCenter`, `ActivityKit`, or `WidgetCenter` anymore; they all go through the Conductor.




The Conductor never reaches for a surface reflexively. It always computes a `SurfaceSet` first, a tiny struct describing exactly which surfaces should light up for *this* event at *this* moment.

## 3. Each surface has one superpower

Instead of each surface being a miniature version of the app, each is a **single-question answer machine**.

### iPhone: "Should I detour?"

The iPhone banner is the only surface rich enough to carry a full rich notification (brand art, balance, distance, expiry badge, action buttons). It is also the only surface with the user's full attention. So it fires *selectively*, only when:

- The user is near a card they haven't used **and**
- Combined urgency score clears the time-sensitive threshold **and**
- No competing surface has already spoken within the last 30 minutes **and**
- The user isn't in CarPlay (driving), a Focus mode, or a quiet zone

### Apple Watch: "Don't look at your phone"

The Watch's superpower is **speed**. A wrist haptic lands a full second before a phone banner does. When the Conductor chooses `.watchHaptic` as a surface, it records a timestamp in `ConductorWatchBridge.notifyHaptic`. For the next 90 seconds, **the iPhone banner for the same event is suppressed:** the user has already been told. No double-buzz.

The Watch also carries a synced **quiet-list:** cards the user has auto-quieted on the phone show as greyed-out on the complication and Watch app. `CardStore.fetchCards()` pushes the list on every refresh via `ConductorWatchBridge.syncQuietList(_:)`.

### Dynamic Island / Live Activity: "How close am I?"

The Live Activity is the only surface designed for **continuous state:** it updates as the user walks. It is not an alert; it is a persistent readout. The Conductor starts a Live

Activity on geofence entry (only if `surfaces.contains(.liveActivity)`) and then the location manager drops **bucketed** distance updates into it, 50-meter increments, not the raw 1Hz CoreLocation stream. That preserves ActivityKit's refresh budget so the activity survives the whole 8-hour cap.

### Widgets: "What am I carrying right now?"

The widget is the only surface visible **without opening anything**. Its superpower is ambient. So the Conductor never sends an *alert* to the widget, it sends a **state refresh**. `CardCueWidgetData` is a shared App-Group payload the Conductor republishes on every route, but only if the hash changed (`ConductorWidgetBridge.cachedHash`). No-op updates don't reach `WidgetCenter.reloadAllTimelines()`. The lock-screen rectangular widget and the home-screen medium widget both read the same snapshot, neither needs to know anything about notifications.

### CarPlay: "Get out of the way"

CarPlay detection is deliberate, we don't import `CarPlay.framework`; we detect the scene via raw role string (`CPTemplateApplicationSceneSessionRoleApplication`). When `NotificationConductor.isCarPlayActive == true`, the Conductor drops iPhone banner and Watch haptic surfaces from *every* alert. The Live Activity keeps updating (for when the user glances at their phone at a stoplight), and the widget snapshot keeps refreshing. But nothing buzzes. Driving should never be louder than the road.

### Share Sheet: "Catch what's coming in"

The Share Sheet is the one surface in the set that faces the other direction. It doesn't tell the user anything; it **listens**. Mail sends you a gift card, you tap Share, the wallet catches it. That makes the Share Sheet the intake edge of the ecosystem, the counterpart to the camera. The iOS Share Extension writes the email text and attachments into the shared App Group, deep-links the main app open, and the Add

Card form appears pre-filled. No banner, no haptic, no network call leaves the extension. A Share Sheet that respects the ecosystem is one that knows the right surface for "a new card just arrived" is the form itself, not an alert.

## 4. The intelligence moments

This is where the ecosystem earns the word "intelligent." These are real branches inside `NotificationConductor.decideSurfaces(for:)`:

### "User is walking toward a Target store at 4mph"

- **Trajectory predictor** sees bearing alignment + speed → fires a `.preArrival` AlertEvent ~3 minutes before geofence entry.
- Conductor routes to **Live Activity only:** enough to be helpful, not enough to interrupt a conversation.
- 15 minutes later, when the user crosses the geofence, the `.geofenceEntry` event fires. Conductor sees the pre-arrival cooldown is still active **and** the Live Activity is already running → suppresses the banner. The Live Activity simply updates to "You're here."

### "User is already inside the store"

- Geofence fires, Conductor checks `recentlyPreArrived` and sees the same card in the last 15 minutes → the banner is suppressed.
- If the user used the card (barcode shown), `fireSingleCardNotification` records usage via `ConductorThrottle.markUsed`, which **pre-emptively shortcuts the next 60 minutes of re-entries** and ends the Live Activity.

### "User is driving through a shopping plaza"

- 4 geofences trip in 90 seconds. Without coordination, you'd get 4 banners.

- Conductor's cluster logic rolls them into a single `AlertEvent.cluster` payload: "*Shopping plaza, 4 cards, $183 available*".
- CarPlay detection suppresses the banner entirely anyway, the Live Activity silently shows the cluster, and the widget refreshes so the total is current when the user parks.

### "User dismissed a Starbucks alert 3 times this month"

- `NotificationReasonStore` has logged every dismiss.
- On the 4th approach, Conductor checks `autoQuietedCards` and routes a one-time `.quietedCard` event ("CardCue Pro stopped bugging you about Starbucks, tap to undo") instead of firing the banner.
- The Watch and widget pick up the new quiet-list via `ConductorWatchBridge`.

### "It's 10pm and payday tomorrow"

- `schedulePaydayReminderIfEnabled()` in `CardCueApp.swift` creates a `UNCalendarNotificationTrigger` for 9:05am on payday. The notification content is built by the **same** `NotificationContentFactory.paydayContent` used for Conductor-imperative payday alerts, no copy drift.
- At 9:05am, the banner fires. Conductor's `cardsTouchedInLast24h()` helper is consulted by the morning scan loop, any card already mentioned in the payday banner is skipped by the geofence notifier for the next 24 hours. No "hey your Starbucks has $3" banner on top of "reload your Starbucks, payday's here."

### "User is in Focus mode"

- Conductor queries `UNUserNotificationCenter.notificationSettings()`.
- If `alertSetting == .disabled`, only the Live Activity and widget update. No banner, no haptic.
- When the user exits Focus, the *next* significant geofence re-checks and resumes normally. Nothing is "replayed", we don't believe in stale alerts.

## 5. The shared-memory architecture

Coordination requires shared memory. Three stores, all in the `group.com.cardcue.app` App Group, are the system's nervous system:

| Store | Lives in | Readers |
|---|---|---|
| `ConductorThrottle` per-card, per-cluster, pre-arrival, daily-cap timestamps | `UserDefaults(suiteName: appGroupID)` | Conductor (iPhone), BGAppRefreshTask, Widget extension |
| `CardCueWidgetData` full card snapshot + balance totals | App Group shared file | Widget timeline, Watch complication, BGTask scheduler |
| `NotificationReasonStore` every decision + signals | App Group plist | Conductor, `NotificationReasoningView`, "Why did I get this?" action |

Because the throttle state lives on disk in the App Group, a `BGAppRefreshTask` cold-launch at 3am can't bypass the 30-minute cooldown the way a runtime-only dictionary would. The widget extension, the Watch, and the main app all share the same accounting ledger.

## 6. The per-event data contract

`AlertEvent` is a strongly-typed enum with one case per reason the app might want to talk to the user:

```
enum AlertEvent {
    case geofenceEntry(GeofencePayload)
    case cluster(ClusterPayload)
    case preArrival(PreArrivalPayload)
    case expiryReminder(ExpiryPayload)
    case payday(PaydayPayload)
```

```
    case weeklyDigest(WeeklyDigestPayload)
    case quietedCard(QuietedCardPayload)
}
```

Every code path that used to call `UNUserNotificationCenter.current().add()` directly now builds an `AlertEvent` and calls `NotificationConductor.shared.route(event)`. There are no remaining bypasses in the main app module.

Each payload carries everything the content factory might need, card ID, card name, formatted balance, distance in meters, days-until-expiry, `isDigitalReady`, etc., so the factory can produce rich copy without having to hit the data layer from inside the notification pipeline.

## 7. How we test that the ecosystem coordinates

Three kinds of tests:

1. **Determinism tests:** given `AlertEvent` X and context Y, assert the `SurfaceSet` is exactly Z. Covers every cell in the CarPlay × Focus × cooldown × quiet-zone × urgency matrix.
2. **Double-fire tests:** simulate a Watch haptic, then inject a phone geofence 10 seconds later; assert the phone banner is suppressed.
3. **Live-Activity budget tests:** feed 60 seconds of 1Hz fake location updates; assert exactly one ActivityKit update per 50m bucket.

These run as XCTest cases in `CardCueTests`, with fake implementations of `CLLocationManager`, `UNUserNotificationCenter`, and `WCSession`.

## 8. Why this matters for the product story

Every notification that *doesn't* fire is a kindness. Every notification that *does* fire is a promise kept, it landed on the right surface, at the right moment, with the right copy.

A user walking into a Target store with a $50 balance on a card expiring in two days should feel like the app noticed. Not because it buzzed six times. Because it buzzed **once**, on the surface they were already looking at, their wrist if they were out for a walk, their Dynamic Island if their phone was already in hand, the widget if they'd glanced at their lock screen on the way in.

That is the difference between an app that ships notifications and an app that respects the ecosystem it lives in.

## 9. Appendix: Files of record

| File | Role |
|---|---|
| `Services/NotificationConductor.swift` | Central routing, scoring, surface decisions |
| `Services/NotificationContentFactory.swift` | Single source of truth for copy |
| `Services/NotificationIntelligence.swift` | Urgency score + trajectory predictor |
| `Services/NotificationReasonStore.swift` | Per-event decision log |
| `Services/ConductorThrottle.swift` | Persistent cross-launch cooldowns |
| `Services/ConductorWatchBridge.swift` | Watch haptic dedup + quiet-list sync |
| `Services/ConductorWidgetBridge.swift` | Hashed widget snapshot republishing |
| `Services/Services.swift` | Geofence scanner, pre-arrival, cluster detector (all route through Conductor) |
| `ViewModels/CardStore.swift` | Expiry scheduler, Watch quiet-list sync |

| File | Role |
| --- | --- |
| `CardCueApp.swift` | Payday + weekly digest calendar triggers (shared content factory) |
| `LiveActivity/CardCueLiveActivity.swift` | Dynamic Island + bucketed distance updates |
| `Widgets/CardCueWidget.swift` | Ambient state readout |
| `Models/Constants.swift` | `DistanceFormatter.formatNearby` shared across surfaces |

THE FIRST TIME

*The first time the Watch delivers a warmth-coded haptic while the Dynamic Island shows the same card and CarPlay stays politely silent because the driver never asked for it, the ecosystem chapter stops reading as an integration list. It becomes what "where you already are" actually felt like, the invisible payoff for Shortcuts, Live Activities, and App Intents all agreeing on one user.*


**Christian Sorensen**, Founder · BigUnit Digital LLC
Read the founder's origin story →

*CardCue Pro, by BigUnit Digital LLC. Built with SwiftUI, SwiftData, ActivityKit, WidgetKit, WatchConnectivity, UserNotifications, and deep respect for the user's attention.*

**A note on the writing.** The thinking, the stories, and the product opinions are mine. I used AI to edit for grammar, tighten prose, and keep the voice consistent across the series. If a sentence lands cleanly, some of that credit goes to the machine. I figured you should know.

## MORE PRESS KITS

COMPLETE KIT

**CardCue Pro: The Whole App in One Read**

SCANNER

**Shutter-less Card Capture**

GEOLOCATION

**The Geolocation Brain**

PRIVACY

**Your Gift Cards Are Nobody Else's Business**

VOICE

**The Voice at the Counter**

## FOLLOW CARDCUE PRO

X / Twitter Instagram TikTok YouTube Threads Mastodon RSS Email

 Back to press