

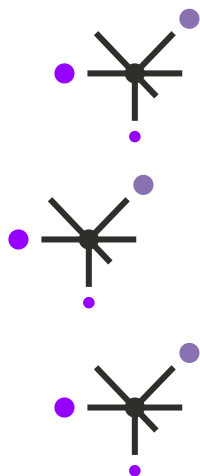
URGENT INCIDENT REVIEW

# The paypal went blank. Do we need an emergency release?

An urgent one-day investigation into falling purchases after an iOS release, contained the same day without shipping a new build.

|             |                                                                           |
|-------------|---------------------------------------------------------------------------|
| Company     | Harborline, an event ticketing app (fictional)                            |
| Question    | Why did purchases fall 32% on version 5.2, and how do we stop it?         |
| Turnaround  | One working day, from first call to containment                           |
| Data        | Analytics events, server and StoreKit logs, remote config. All synthetic. |
| Prepared by | Alex Severianov                                                           |

This is a sample. The company, people, code and numbers are invented. The kind of bugs, the checks and the reasoning come from real production investigations I have done.



00\_

# Executive summary

## Background

Three days into the phased release of iOS 5.2, purchases of the Plus membership fell 32% per paywall view. Version 5.2 also had new paywall copy, so the team suspected the copy and prepared an emergency release to revert it.

## Objective

Find why purchases fell on 5.2, stop the loss as fast as possible, and decide whether an emergency release is needed.

## What was done

Treated as urgent: every day on 5.2 cost purchases. Within one working day I compared 5.2 with 5.1 over the same days, added server-side logging of paywall load failures, traced each failure through remote config, StoreKit and the SwiftUI view lifecycle, and tested each mitigation before applying it.

## Key findings

- **The copy was not the problem.** 30.1% of 5.2 paywalls showed no products at all. People who saw a full paywall bought at the usual rate.
- **Three separate bugs caused the blank paywalls:** a config timeout cut from 10 to 3 seconds (50%), one missing App Store product that hid all products (34%), and a product load cancelled when the view was re-created (16%).
- **Analytics hid it.** The paywall view event fired before products loaded, so blank paywalls were counted as normal views.
- **All three were contained on the server the same day.** Blank paywalls fell from 30.1% to 1.3% within two hours, and conversion recovered by the next morning.

## Conclusion

No emergency release was needed. Reverting the copy would not have fixed anything, and would have taken longer than the server-side containment.

## Recommendations

Ship the code fixes in the next regular release. Fire the paywall view event only when products are on screen, and log load failures with their reason. Add tests for slow networks and partial product lists, and an alert on blank paywalls.

01\_

What was reported.  
What was actually wrong.

| Reported                        | What I found                                                                                         | Fixed where      |
|---------------------------------|------------------------------------------------------------------------------------------------------|------------------|
| Purchases down 32% on 5.2       | 30.1% of paywalls had no products. Full paywalls converted at 3.0%, close to 5.1's 3.1%.             | Not the copy     |
| Some users see an empty screen  | Remote config timed out on slow networks after the networking refactor cut its timeout to 3 seconds. | Server, then app |
| Nothing loads in some countries | The new intro offer wasn't approved in 9 storefronts. One missing product made the app show none.    | Server, then app |
| Endless spinner for new users   | The experiment variant arrived late, the view was re-created, and the product load was cancelled.    | Server, then app |
| Paywall views look normal       | The view event fired on appear, before content. Blank paywalls counted as views.                     | Analytics        |

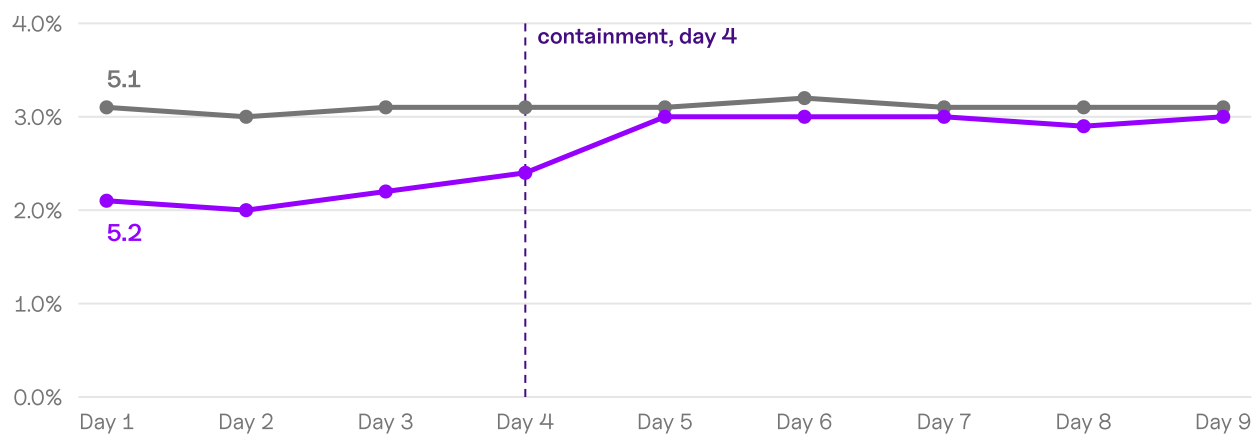
DECISION

Cancel the emergency release. Contain all three causes on the server today, and ship the code fixes in the next regular release.

02\_

# A third of the purchases disappeared. The dashboard looked normal.

The alert compared purchases per paywall view. On 5.2 that rate dropped from the first day of the rollout. Paywall views themselves looked normal, so the obvious explanation was that people didn't like the new copy.



Purchases per paywall view, by app version and day of the rollout

I added server-side logging of paywall load failures on day 4. It showed that 30.1% of 5.2 paywalls never displayed any product, against 0.6% on 5.1. Each failure logged its reason:

| Why the paywall was blank, 5.2, days 1–3            | Paywalls | Share          |
|-----------------------------------------------------|----------|----------------|
| Remote config timed out, no product IDs             | 7,250    | 50%            |
| One product missing, all products hidden            | 4,900    | 34%            |
| Product load cancelled when the view was re-created | 2,380    | 16%            |
| All blank paywalls                                  | 14,530   | 30.1% of views |

**Excluding blank paywalls, 5.2 converted at 3.0%.** The copy was fine. About 482 purchases were lost in three days, and the loss would have grown with every day of the phased rollout.

03\_

# Bug one: a 3-second timeout on slow networks.

The paywall gets its product IDs from remote config. Version 5.2 included a networking refactor that moved every request to one shared client with a 3-second timeout. Remote config used to have 10 seconds. Nobody noticed, because on office Wi-Fi it rarely takes more than a second.

```
// NetworkClient.swift, 5.2: one client for everything
let session = URLSession(configuration: .default.with(timeout: 3))
// RemoteConfig used this client, so its 10 s timeout became 3 s
```

| Remote config fetch, 5.2 | Median | 90th percentile | Over 3 seconds |
|--------------------------|--------|-----------------|----------------|
| Wi-Fi                    | 0.7 s  | 2.4 s           | 4.2%           |
| Cellular                 | 1.6 s  | 4.6 s           | 21.0%          |

On cellular, one fetch in five timed out. With no product IDs, the paywall rendered its empty container. There was no cached config to fall back on and no error state.

Part of the delay was the response itself: 410 KB, mostly translations the paywall never uses. Trimming it was the server-side lever.

```
// Fix in 5.2.1: own timeout, then last known config, then bundled defaults
let config = try? await remoteConfig.fetch(timeout: .seconds(10))
    ?? remoteConfig.lastKnown
    ?? .bundledDefault
```

## 04.

## Bug two: one missing product hid all of them.

Version 5.2 added a new annual intro offer to the product list in remote config. Its intro pricing was still waiting for App Store approval in 9 storefronts. There, StoreKit returned every product except the new one.

```
// PaywallModel.swift, 5.2
let products = try await Product.products(for: ids)
guard products.count == ids.count else {
    state = .failed // one missing product hid all of them
    return
}
```

The check was written to catch broken configs. In practice, it turned one product that wasn't available yet into a blank paywall for every user in those storefronts: 4,900 paywalls in three days, every paywall shown there.

**This is why some countries looked completely broken** while others only had occasional failures. Two causes overlapped, and each on its own looked like a different bug.

```
// Fix in 5.2.1: show what loaded, report what didn't
let products = try await Product.products(for: ids)
let missing = Set(ids).subtracting(products.map(\.id))
if !missing.isEmpty { log.paywallProductsMissing(missing) }
state = products.isEmpty ? .failed : .loaded(products)
```

App Store product availability differs by storefront. A product can exist in App Store Connect and still be missing from StoreKit responses in some countries while a price or offer is in review.

## 05.

## Bug three: the experiment re-created the view.

Version 5.2 also started a payroll experiment. On a first launch, the experiment variant often arrived after the payroll was already on screen. The parent view used the variant as the payroll's identity, so SwiftUI threw the old view away and built a new one.

```
PaywallView(model: model)           // model is owned by the parent
    .id(experiment.variant)          // variant arrives late: view re-created

// PaywallView.swift, 5.2
.task {
    guard model.state != .loading else { return } // the new view skips loading
    model.state = .loading
    do {
        model.state = .loaded(try await model.loadProducts())
    } catch {
        log.error(error) // cancelled: state stays .loading forever
    }
}
```

The old view's task was cancelled mid-load and left the shared model in *loading*. The new view saw *loading* and waited for a result that would never come. 19% of first-session paywalls on 5.2 ended in an endless spinner.

```
// Fix in 5.2.1: no .id(), the view stays and the task restarts per variant
PaywallView(model: model)

.task(id: experiment.variant) {
    model.state = .loading
    do {
        model.state = .loaded(try await model.loadProducts())
    } catch is CancellationError {
        // the replacement task takes over
    } catch {
        model.state = .failed(error)
    }
}
```

06\_

# Contained without an app release.

An emergency release would have taken at least a day for review, and then days more for users to update. All three causes could be reduced from the server, and each change could be checked within minutes on the failure logs.

|           |                                                                                                                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Bug one   | Trimmed the remote config response from 410 KB to 58 KB by moving unused translations out. Cellular 90th-percentile fetch time fell from 4.6 s to 1.1 s, inside the 3-second limit. |
| Bug two   | Removed the new intro product from the product list until its pricing is approved everywhere. Every storefront loaded products again.                                               |
| Bug three | Stopped the paywall experiment and served the control to everyone. The variant no longer changes after the paywall appears.                                                         |

|                |                                                                                                                               |
|----------------|-------------------------------------------------------------------------------------------------------------------------------|
| Timeline       |                                                                                                                               |
| Day 1          | Version 5.2 starts its phased release.                                                                                        |
| Day 3, evening | Purchase alert: conversion on 5.2 is a third lower. The team suspects the new paywall copy and prepares an emergency release. |
| Day 4, 08:30   | Investigation starts. Load-failure logging added server-side within the first hour.                                           |
| Day 4, 13:00   | Three causes identified and confirmed in the logs.                                                                            |
| Day 4, 15:10   | Containment: config trimmed, intro product removed, experiment stopped. No app release.                                       |
| Day 4, 17:10   | Blank paywalls down from 30.1% to 1.3%.                                                                                       |
| Day 12         | Version 5.2.1 ships the code fixes in the regular release.                                                                    |

In the five days after, 5.2 converted at 3.0% against 3.1% on 5.1 (p = 0.34). The code fixes went out a week later in the regular release, with tests, instead of as a rushed build.

07\_

# What was ruled out.

Before changing anything, I checked the explanations that would have needed a different fix.

- ✓ **Not an App Store outage.** 5.1 users bought normally on the same days, in the same countries.
- ✓ **Not the backend.** The config endpoint answered 99.9% of requests successfully. The failures happened on the device, after the response was late.
- ✓ **Not a device or iOS version.** Blank paywalls were spread across iOS 16 to 18 and every iPhone model in proportion to usage.
- ✓ **Not the copy.** Users who saw a complete 5.2 paywall bought at 3.0%, close to 5.1.
- ! **Containment is not a fix.** A bigger config or a new product would break 5.2 again until 5.2.1 is installed widely.

08\_

# What I would do next.

|            |                                                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| Release    | Ship the three code fixes in 5.2.1. Keep the config small and the new product off until most users have updated.                           |
| Analytics  | Fire the paywall view event when products are on screen. Add a load-failure event with its reason, so this is visible on day 1, not day 3. |
| Tests      | Unit tests for a partial product list and a config timeout. A UI test on a throttled network profile.                                      |
| Monitoring | Alert when blank paywalls exceed 2% of views for any app version.                                                                          |
| Process    | Product and offer changes in remote config wait until App Store approval is complete in every storefront.                                  |

## 09\_

## How this investigation was done.

### Data used

Product analytics by app version, server logs for remote config, a temporary server-side load-failure log, StoreKit product responses by storefront, and the 5.1 to 5.2 code diff.

### Tools

Xcode with the network link conditioner, Instruments, SwiftUI view lifecycle tracing, the App Store Connect product status, and log queries joined by session.

### Statistics

Two-sided tests on proportions comparing versions over the same days, so seasonality and campaigns affect both equally.

### Limits

The cause of each failure comes from logs added on day 4, applied back to days 1–3 by matching sessions. A small share of failures had no clear reason and is not shown. Lost purchases are an estimate at 5.1's conversion rate.

### Time

One working day: the morning to find and confirm the three causes, the afternoon to contain them and verify recovery, the evening to write this up. Code fixes were done separately.

### What you get

An executive summary with the decision, a report like this one, and the fixes themselves if you want me to build them.



### Production problem nobody can pin down?

Send it to me. I'll tell you what I would check first.

x91tech.com · Alex Severianov