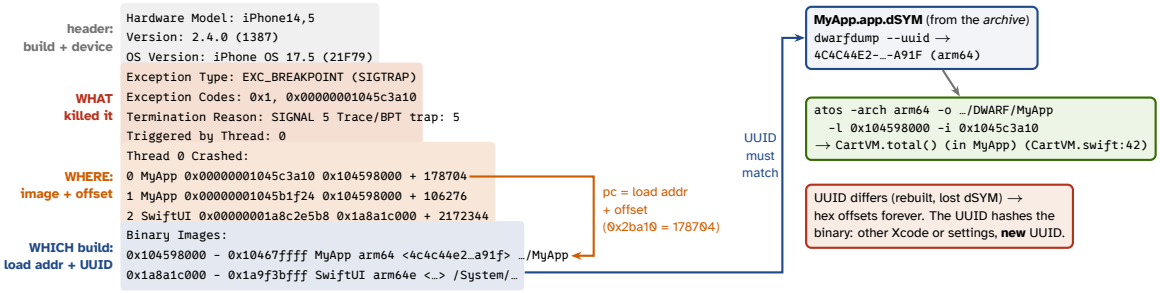
Crashes & symbolication

In one line: A crash report says **what** killed the process (*Exception Type* + *Termination Reason*), **where** (the crashed thread's backtrace of *image* + *offset*), and **which build** (the *UUID* of every binary image). Symbolication turns offsets into `function + file:line` using the **dSYM whose UUID matches** — no matching dSYM, no names, ever.



Exception type → cause → first move

Type (signal)	Meaning	First move
`EXC_BAD_ACCESS` (SIGSEGV / SIGBUS)	bad pointer: freed object, unowned, unsafe pointer; `KERN_INVALID_ADDRESS`	Zombies, ASan
`EXC_BREAKPOINT` (SIGTRAP)	Swift trap: `!` on nil, index out of range, `fatalError`, `precondition`, `as!`, overflow	top app frame (Release drops `precondition`'s message)
`EXC_CRASH` (SIGABRT)	`abort()`: uncaught `NSException`, C `assert`	*Last Exception Backtrace*
`EXC_CRASH` (SIGKILL) `0x8badf00d`	watchdog: main thread blocked at launch / resume / suspend	main stack; I/O off main
`0xdead10cc`	held a file or SQLite lock while being suspended	`beginBackgroundTask`, release first
no crash log	**jetsam** (memory limit)	iOS 27 MetricKit `.memoryException`

How it works

- Reports are JSON `.ips` (iOS 15+). Always read in order: exception type → termination reason → crashed thread (main?) → first frame in *your* image.
- **dSYM** = DWARF debug info split out of the stripped binary (`DEBUG_INFORMATION_FORMAT = dwarf-with-dsym`); the archive keeps it. Find one by UUID: `mdfind "com_apple_xcode_dsym_uuids == <UUID>"`.
- **Bitcode** is gone (Xcode 14): the App Store no longer recompiles, and App Store Connect has no dSYM to download: the *archive's* is your only copy.
- Where crashes arrive: Xcode **Organizer** (opt-in users, delayed, auto-symbolicated if the dSYM is known) · Crashlytics/Sentry (upload dSYMs from a build phase or CI) · **MetricKit**.
- **MetricKit** (iOS 27): one `MetricManager` instance; `diagnosticReports` is an `AsyncSequence` of `DiagnosticReport` (`Codable`); `result`: `.crash` · `.hang` · `.cpuException` · `.diskWriteException` · `.appLaunch` · `.memoryException` (memory-limit kills). Before: `MXMetricManager` + subscriber `didReceive(_:)` — deprecated in 27.2. Call stacks come *unsymbolicated*. Diagnostics arrive immediately (iOS 15+), metrics at most daily.

Reproduce and triage

- Memory: **Zombies**, **ASan**; races: **TSan** (not with ASan). OOM: Memory Graph; downsample images.
- `8badf00d`: read *Elapsed application CPU time* (all threads) before the stack. High = working: Time Profiler. Low = *waiting*, e.g. a synchronous network call, even one already off the stack (`String(contentsOf:)` on a URL, `SCNetworkReachabilityGetFlags`). Office Wi-Fi never shows it.

Example

```
let metrics = MetricManager()        // ONE; 2 loops split reports
Task {
  for await report in metrics.diagnosticReports { // never throws
    switch report.result {
    case .crash, .hang, .memoryException:
      send(report)                   // Codable; stacks
      unsymbolicated
    default: break
    }
  }
}
```

Traps

- ! Swift traps are **not** SIGABRT: on arm64 they are `EXC_BREAKPOINT (SIGTRAP)` (`EXC_BAD_INSTRUCTION`/SIGILL on Intel simulators).
- ! "No crash log" ≠ no crash — a jetsam event report holds every process's memory but *no* backtrace of your threads.
- ! `0x8badf00d` is not a bug at the top frame: it is where the main thread *was* when time ran out.
- ! Inlined calls have no frame of their own: `atos -i` brings them back from the dSYM.
- ! Never trust "I rebuilt the same commit": another Xcode version or build setting means another UUID. Keep every shipped archive.

Remember

"What · Where · Which build." Type tells the *category*, the backtrace the *place*, the UUID the *dSYM*. `8badf00d` = too slow, `dead10cc` = lock on suspend.

Related: linking & launch time · code signing & iOS CI/CD (archive keeps dSYMs) · ARC (retain cycles) · Instruments · Concurrency (GCD, threads)