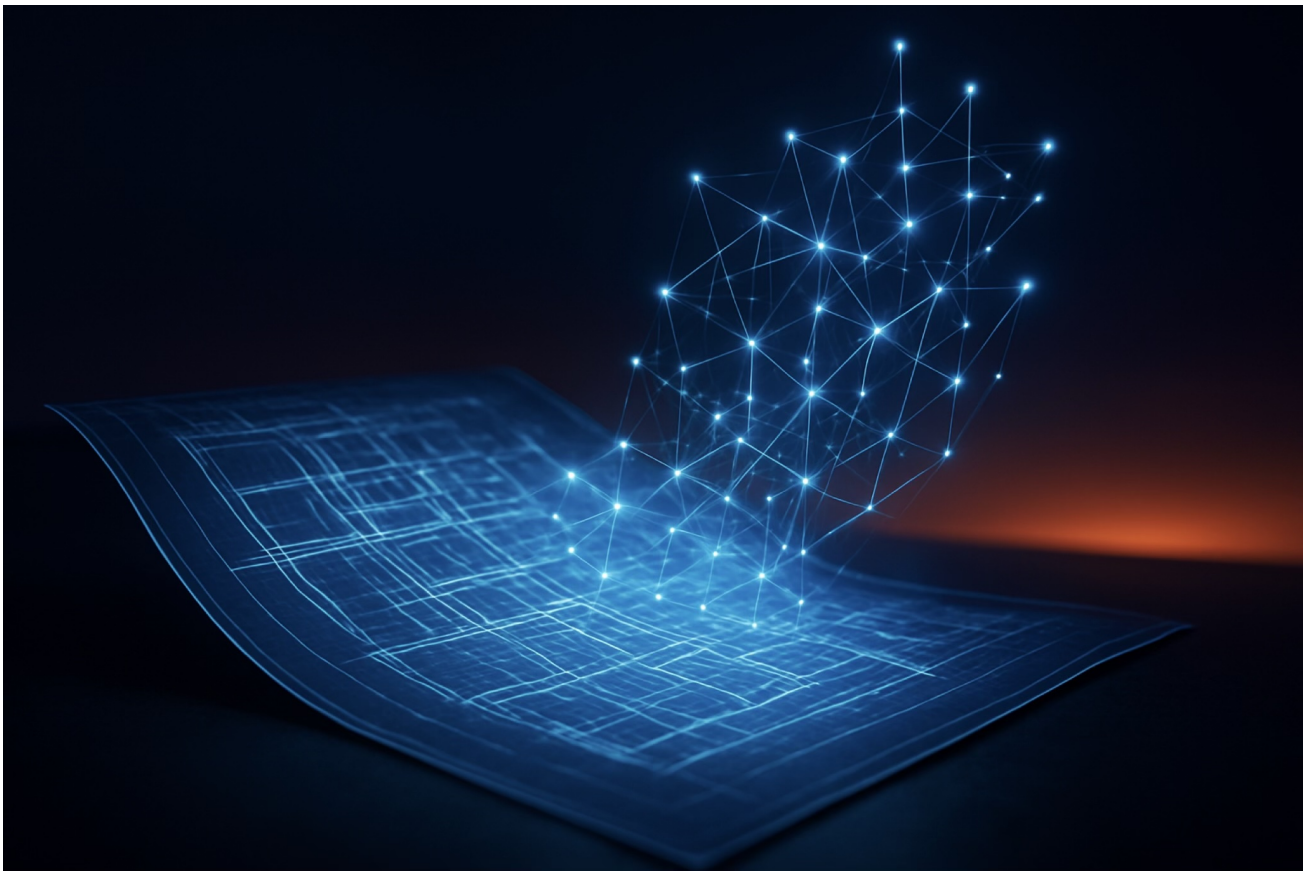

What Apple Wants You to Build This Summer (You Can Read It in the Docs)

The keynote tells you what Apple announced. The documentation and the sample code tell you what Apple wants you to actually ship. Those are not the same thing, and the second one is far more useful if you are deciding what to build this summer.

So I did the boring, revealing thing: I diffed it. Cupertino just shipped v1.4.0, a full re-crawl of Apple's developer documentation, and I compared it page by page against the version that has been on everyone's disk since May. Then I read the diff like a tea leaf.

The answer is not subtle. **Apple wants you to build Apple Intelligence into your app, and it has written you the homework.**

The sample code is the smoking gun

When Apple wants you to learn an API, it announces it. When Apple wants you to *ship* something, it gives you sample code you can paste from. So the new sample projects are the clearest signal in the whole corpus, and v1.4.0 added eleven of them. Here is the entire list:

App Intents: Integrating Your **Music** App with Apple Intelligence
App Intents: Integrating Your **Messaging** App with Apple Intelligence
App Intents: Integrating Your **Calendar** App with Apple Intelligence
App Intents: Integrating Your **Photo** App with Apple Intelligence
FoundationModels: Origami, crafting a dynamic tutorial for Apple Intelligence
Evaluations: Book Tracker, using evaluations to evaluate an intelligent feature

HealthKit: Tracking heart rate zones for workouts
Metal: Training a neural network to render irradiance in real time
Metal: Running inline ML operations in a shader with Metal 4
DeclaredAgeRange: Implementing age assurance and permissions
VisionKit: Structuring recognized text on a document

Read the top four again. Apple did not write "an example of App Intents." It wrote "Integrating **Your** Music App," "**Your** Messaging App," "**Your** Calendar App," "**Your** Photo App." That is not documentation. That is a brief. Apple has taken the four most common app categories on the platform and handed each one a template for wiring itself into Apple Intelligence through App Intents. If you ship one of those four kinds of app, your summer roadmap is sitting in a sample project with your name on it.

There is not a single exception in the eleven. Every new sample is Apple Intelligence, App Intents, or on-device ML. No new SwiftUI layout sample, no new networking sample, no new Core Data sample. The signal is monochrome.

The new frameworks are the syllabus

Sample code shows you the assignments; the new frameworks show you the curriculum. v1.4.0 added **19 frameworks that did not exist in the previous release**, and the cluster tells the same story:

Framework	Pages	Symbols
USDKit	854	806
ComputeGraph	539	273
Core AI	308	425
Evaluations	172	527
App Intents Testing	103	363
Now Playing	100	318
Music Understanding	94	77
AVSystemRouting	92	109
Media Device	92	148
Trust Insights	85	66
Media Intelligence	77	100
Accessory Access	63	53
DiskImageKit	58	87
Spatial Preview	57	60
StateReporting	38	77
CrashReportExtension	29	36
Media Intents	15	30
Suggested Actions	9	26

Framework	Pages	Symbols
Apple School and Business Manager API	3	0

Core AI. Evaluations. Media Intelligence. Music Understanding. Trust Insights. Suggested Actions. App Intents Testing. This is the connective tissue around on-device models: the pieces that let your app declare what it can do, feed a model the right context, evaluate what comes back, and route media intelligently. **Evaluations** is the quiet standout. Apple shipping a framework specifically for *evaluating* an intelligent feature, plus a sample that uses it, is Apple admitting that "add AI" is not a feature, it is an engineering discipline with a test loop. That is the most grown-up thing in the release.

And it is not only the debutants. `foundationmodels`, the on-device model framework, was not new but it more than doubled in pages (356 to 821) and more than tripled in symbols (406 to 1,383). The framework Apple gave you last year to talk to its models is the one it spent this cycle fleshing out.

The honest part: most of the bytes are a deeper re-crawl

I am not going to oversell the numbers, because the headline flatters the truth. The documentation database grew by **+12,057 pages and +67,575 symbols**. But only about **23% of the new pages** come from those 19 new frameworks. The other 77% is a deeper re-crawl of frameworks that already existed. For symbols it is even more lopsided: roughly **95%** of the gain is re-crawled existing frameworks, and the Swift standard library alone accounts for over 40% of it (+27,908 symbols).

So the accurate sentence is: the byte-weight is a fuller pass over everything, and the *signal* is the 19 new frameworks and eleven new samples on top. Both are true, and a release post that only quoted the big number would be lying by omission.

Where did the existing-framework growth concentrate? Right where you would guess given the theme:

Framework	old pages	new pages	delta
realitykit	4,456	6,133	+1,677
appintents	3,509	4,376	+867
foundation	13,649	14,282	+633
appkit	13,378	13,917	+539
foundationmodels	356	821	+465
swift	18,717	19,057	+340
metrickit	249	567	+318
corespotlight	403	719	+316
swiftui	8,679	8,938	+259

RealityKit and App Intents leading the page growth is the same story from yet another angle: spatial computing and the intent system are both load-bearing for what Apple is asking you to build. No framework shrank in page count and none were removed, so at the framework level v1.4.0 is a clean superset of v1.3.0.

What did not change, honestly

A re-crawl is not a magic content generator. Four databases came back byte-for-byte identical to v1.3.0: HIG, Swift Evolution, the Swift book, and the legacy Apple Archive guides. Three of those (HIG, Swift Evolution, the Swift book) were genuinely re-crawled this cycle and simply came back unchanged because Apple had not touched them upstream; the archive corpus was the one source carried over from the previous cycle. I verified "identical" with a content fingerprint over every page, not by trusting file dates. If your work lives in those four, v1.4.0 changes nothing for you, and I would rather say so than pad the release.

The Swift package index (`packages.db`) is the interesting middle case. The set of 185 packages is unchanged, but 183 were re-fetched from current upstream and re-indexed, so the database now reflects what those packages look like *today* rather than at the last crawl. That means it is honest rather than purely additive: Vapor lost 1,604 symbols where upstream refactored, while swift-package-manager and swift-async-algorithms each gained a few hundred. Net +2,836 symbols, but the real win is that the package picture is current.

Why your AI agent needs this, specifically now

Here is the part that ties the signal back to the tool. If Apple wants you to spend the summer wiring your app into Apple Intelligence, you are going to ask an assistant about Core AI, App Intents Testing, the Evaluations framework, and FoundationModels constantly. And an assistant working from a six-month-old corpus will confidently tell you those frameworks do not exist, because in its world they do not.

That is exactly the failure Cupertino exists to prevent. It puts Apple's documentation on your machine as a set of local, searchable databases an AI agent can read through MCP, offline, in milliseconds. v1.4.0 is not a flashy release: it does not add a feature or touch a line of the query engine. It does the quieter thing that actually matters for a documentation tool, which is make the corpus match what Apple shipped. The fact that the new frameworks map almost perfectly onto Apple's summer wishlist is not a coincidence; it is the whole point. Your agent now knows about the assignment.

Getting v1.4.0

The binary is signed and notarized and lives on the tap:

```
brew upgrade cupertino     # or: brew install mihaelamj/tap/cupertino
cupertino setup            # downloads the v1.4.0 database bundle
```

Same 8 per-source databases as v1.3.0, shipped read-only in rollback-journal mode, same schema (v18, packages on their own v5 track). Nothing in your tooling changes; the corpus underneath it just got current, and it got current in a very specific, very on-message direction.

Build the thing in the sample project. Apple already wrote the brief.