Cupertino v0.5.0: 234K Docs, 287 Frameworks, and the Roadmap Ahead

TL;DR: Documentation database nearly doubled. New release tooling. A roadmap for what's next.

Two days after v0.4.0, we're shipping v0.5.0. Why so fast? Because I finally finished a 5-day deep crawl of Apple's documentation.

The Numbers

Metric	v0.4.0	v0.5.0	Change
Total Documents	138,000	234,331	+70%
Frameworks	263	287	+24
Database Size	~1.2GB	~1.9GB	+58%
Crawl Time	20-24 hours	32-48 hours	Deeper

The crawl ran for 5 days because I wanted *everything*. Every framework, every page, every deprecated API. The result: nearly 100,000 more documents than before.

Top Frameworks by Document Count

1. **Kernel** - 24,747 docs
2. **Matter** - 22,013 docs
3. **Swift** - 17,466 docs
4. **Foundation** - 14,892 docs
5. **UIKit** - 12,341 docs

Kernel documentation alone has almost 25,000 pages. That's IOKit, BSD interfaces, Mach APIs - the stuff that's usually impossible to find.

What This Means for AI

When Claude searches Cupertino now, it has access to:

Low-level APIs - Kernel programming, IOKit drivers, Mach ports
Hardware interfaces - Matter protocol (24k+ docs for smart home development)
Complete Swift coverage - Every proposal, every standard library type
Obscure frameworks - The ones you forget exist until you need them

This is deterministic access to Apple's documentation. No hallucination, no guessing from training data. Just the actual docs.

Why I Built Release Automation

I remember the release oopsie from v0.4.0 like it was just a few days ago. Wait, it *was* just a few days ago. ? I tagged before committing the version bump, had to delete tags, rebuild everything, update SHA256 checksums twice...

Almost happened again with v0.5.0. Caught it in time, but the close call was enough.

So I finally wrote the automation I'd been putting off. Not because users need it - they don't. But because *I* was losing hours to release mechanics instead of building features.

The release process now touches 4 repositories (main repo, cupertino-docs for databases, Homebrew tap, and the blog). Miss one step, wrong order, typo in a version number - and you're starting over.

Now it's one command. I run it, go make coffee, come back to a published release. That's the dream, anyway. We'll see how v0.6.0 goes.

The Roadmap

The full roadmap is tracked in the Cupertino Roadmap GitHub project. Here's what's coming next, in dependency order:

Phase 1: Complete the Crawl (#88)

The crawl is extensive but not complete. Some frameworks have deep hierarchies that weren't fully traversed. The goal is 100% coverage of Apple's public documentation.

Status: In progress. Currently at depth level 8. No idea when it finishes - some frameworks go deep.

Phase 2: Prune the Database (#87)

Once we have everything, we need to clean it up. The database has:

- Duplicate entries (same doc indexed multiple times)
- Misclassified documents (wrong framework attribution)
- Orphaned entries from URL changes

This is blocked by #88 - no point pruning until the crawl is complete.

Phase 3: API Version Tracking (#99)

This is the big one. Apple's documentation includes version information - which iOS/macOS version introduced each API. We can extract this.

Imagine searching with:

```
cupertino search "async" --min-os ios16
```

Or having Claude automatically filter out deprecated APIs when you're targeting iOS 17+.

I've been investigating Apple's `/tutorials/data/documentation/` endpoint. It returns JSON with version metadata for most frameworks. Some older frameworks (Kernel, IOKit) don't have it, but the coverage is good.

Status: Research complete. Implementation pending.

Phase 4: Version-Filtered Search (#100)

Once we have version data in the database, we can expose it as a search filter. Both CLI and MCP tool would support filtering by minimum OS version.

Blocked by: #99 (need the data first)

Phase 5: Semantic Code Search (#81)

The end goal: search sample code by what it *does*, not just by keyword.

```
cupertino search "async image loading with caching"
```

This requires:

1. Complete documentation (#88)
2. Clean database (#87)
3. SwiftSyntax AST indexing of sample code

It's ambitious, but the foundation is being laid with each release.

Install or Update

```
# New install
brew install mihaelamj/tap/cupertino

# Update
brew upgrade cupertino

# Or instant setup
bash <(curl -sSL
https://raw.githubusercontent.com/mihaelamj/cupertino/main/install.sh)
```

After installing, databases download automatically:

```
cupertino setup
```

Takes about 30 seconds to download the pre-built 1.9GB database.

The Vision: Human-Curated Developer Docs

Here's what I've learned: having *all* the docs isn't enough. We need them *in context*.

When you're building an app targeting iOS 15+, you don't want to see iOS 12 APIs. When you're learning Core Animation, you want the conceptual guides from Apple Archive, not just the API reference. When you're debugging a Matter integration, you need the 22,000 Matter docs filtered to what's actually relevant.

Raw documentation is noise. Curated documentation is signal.

That's where Cupertino is heading. Not just "here's 234,331 docs" but "here's what you actually need for your specific context." Version filtering is coming - ask for iOS 16+ APIs only. Source awareness is coming - archive guides surfaced alongside modern reference. Framework relationships are coming - search UIKit and get relevant Foundation results too.

The crawl is the foundation. The curation is the value.

234,331 documents today. Context-aware, version-filtered, human-curated search tomorrow.

Cupertino is an Apple Documentation MCP Server. Install with `brew install mihaelamj/tap/cupertino`*. Source at github.com/mihaelamj/cupertino.*