Contents

Apple Developer Documentation Offline Archive

Download and maintain a complete offline archive of Apple Developer Documentation in AI-optimized formats (Markdown + JSON), with git-like incremental updates.

Features

- **Offline Access** - Complete documentation available without internet
- **AI-Optimized** - Clean Markdown format perfect for AI/LLM processing
- **Git-Like Updates** - Incremental updates with `check`, `pull`, and `status` commands
- **Multiple Formats** - Raw JSON + AI-friendly Markdown
- **Smart Caching** - Only downloads changed pages using ETags
- **Resume Support** - Continue interrupted downloads
- **Framework Selection** - Download only what you need

Quick Start

```
# Clone or download this repository
git clone <your-repo-url>
cd apple-docs-offline

# Setup
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
pip install -r scripts/requirements.txt
```

```
# Download Swift documentation (test with one framework)
python scripts/01_discover_docs.py --frameworks swift
python scripts/02_download_json.py --frameworks swift
python scripts/03_json_to_markdown.py --frameworks swift

# Later: Check for updates
python scripts/update_check.py
python scripts/update_pull.py
```

Installation

Prerequisites

- Python 3.8+
- 5-10 GB free disk space (depending on frameworks)
- Internet connection for initial download

Setup

1. **Create Virtual Environment**

```
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

2. **Install Dependencies**

```
pip install -r scripts/requirements.txt
```

3. **Optional: PDF Support** (not required for Markdown/JSON)

```
playwright install chromium
```

Usage

Initial Download

Step 1: Discover Documentation Crawls Apple's API to discover all available documentation pages:

```
# All frameworks (~50,000+ pages)
python scripts/01_discover_docs.py

# Specific frameworks only
python scripts/01_discover_docs.py --frameworks swift swiftui uikit

# Resume interrupted discovery
python scripts/01_discover_docs.py --resume
```

Duration: 1-3 hours **Output:** `index.json` with complete documentation hierarchy

Step 2: Download JSON Downloads all discovered pages as JSON files:

```
# All frameworks
python scripts/02_download_json.py

# Specific frameworks
python scripts/02_download_json.py --frameworks swift swiftui
```

Duration: 4-12 hours (rate limited at 5 req/sec) **Output:** - `raw-json/` - Original Apple JSON - `.docsync/manifest.json` - Metadata for update tracking

Step 3: Convert to Markdown Converts JSON to clean, AI-readable Markdown:

```
# All frameworks
python scripts/03_json_to_markdown.py

# Specific frameworks
python scripts/03_json_to_markdown.py --frameworks swift swiftui
```

Duration: 1-3 hours **Output:** `markdown/` - Markdown files with YAML frontmatter

Update System (Git-Like)

Check for Updates (`git fetch`)

```
python scripts/update_check.py
```

Output:

```
  Checking for updates...

Updates available:
[CHANGED] SwiftUI/TabView - Navigation updates for iOS 18
[CHANGED] Swift/Array - Performance improvements documentation
...

Total: 12 modified pages
To download: python scripts/update_pull.py
```

Show Status (`git status`)

```
python scripts/update_status.py
```

Shows current documentation state, last update check, and available updates.

Pull Updates (`git pull`)

```
# Download all updates
python scripts/update_pull.py

# Specific framework only
python scripts/update_pull.py --frameworks swiftui
```

```
# Skip markdown conversion
python scripts/update_pull.py --no-convert
```

Output: - Updated JSON and Markdown files - Changelog in `.docsync/changelog/` - Version snapshot

Directory Structure

```
apple-docs-offline/
  scripts/
     01_discover_docs.py    # Recursive documentation crawler
     02_download_json.py    # JSON downloader with manifest
     03_json_to_markdown.py # JSON → Markdown converter
     update_check.py        # Check for updates (git fetch)
     update_pull.py         # Download updates (git pull)
     update_status.py       # Show status (git status)
     requirements.txt       # Python dependencies

  .docsync/                  # Update tracking metadata
     manifest.json          # All pages with ETags/hashes
     versions/              # Version snapshots
     changelog/             # Update changelogs
     cache/                 # Update check cache

  raw-json/                  # Original Apple JSON
     swift/
     swiftui/
     ...

  markdown/                  # AI-optimized Markdown
     swift/
        array.md
        string.md
        ...
     swiftui/
        view.md
        ...
     ...

  index.json                 # Discovery index
  README.md                  # This file
  venv/                      # Python virtual environment
```

Supported Frameworks

The following frameworks are supported by default:

Framework	Description
swift	Swift Language Documentation
swiftui	SwiftUI Framework
uikit	UIKit Framework
foundation	Foundation Framework
coredata	Core Data Framework
combine	Combine Framework
swiftdata	SwiftData Framework
coreml	Core ML Framework
mapkit	MapKit Framework
avfoundation	AVFoundation Framework

You can add more frameworks by editing `FRAMEWORK_ROOTS` in `scripts/01_discover_docs.py`.

AI/LLM Integration

The Markdown output is optimized for AI processing:

- **YAML Frontmatter** - Structured metadata (title, role, platforms)
- **Clean Hierarchy** - Proper heading levels, lists, code blocks
- **Code Highlighting** - Syntax information preserved
- **Cross-References** - Relative links to related documentation
- **No HTML** - Pure Markdown for easy parsing

Example: Using with AI

```
Read the SwiftUI View documentation:
markdown/swiftui/view.md

Explain the main protocol requirements.
```

Example: Vector Database Indexing

```python
import os
from pathlib import Path

# Load all markdown files
docs_dir = Path("markdown")
for md_file in docs_dir.rglob("*.md"):
    with open(md_file) as f:
        content = f.read()
        # Add to your vector database
        # vector_db.add(content, metadata={"source": str(md_file)})
```

Storage Requirements

Component	Size
JSON (`raw-json/`)	~2-3 GB
Markdown (`markdown/`)	~1-2 GB
Total	**~3-5 GB**

Rate Limiting & Performance

All scripts implement respectful rate limiting:

- **Discovery:** 5 requests/second
- **Download:** 5 requests/second with exponential backoff
- **Update Check:** 10 requests/second (HEAD requests only)

Troubleshooting

"Manifest file not found"

The manifest is created during the first download:

```
python scripts/02_download_json.py
```

"No update check found"

Run an update check first:

```
python scripts/update_check.py
```

Rate Limiting (429 errors)

Scripts automatically retry with backoff. If issues persist: - Wait 10-15 minutes - Resume with `--resume` flag (for discovery)

Missing Dependencies

```
source venv/bin/activate  # On Windows: venv\Scripts\activate
pip install -r scripts/requirements.txt
```

Advanced Usage

Test with Single Framework

For faster testing, download only one framework:

```
python scripts/01_discover_docs.py --frameworks swift
python scripts/02_download_json.py --frameworks swift
python scripts/03_json_to_markdown.py --frameworks swift
```

Resume Interrupted Downloads

Discovery supports resume:

```
python scripts/01_discover_docs.py --resume
```

Download automatically skips existing files.

Framework-Specific Updates

```
python scripts/update_check.py --frameworks swiftui
python scripts/update_pull.py --frameworks swiftui
```

Command Help

All scripts support `--help`:

```
python scripts/01_discover_docs.py --help
python scripts/02_download_json.py --help
python scripts/03_json_to_markdown.py --help
python scripts/update_check.py --help
python scripts/update_pull.py --help
python scripts/update_status.py --help
```

How It Works

Documentation Discovery

1. Starts with framework root URLs (e.g., `swift.json`)
2. Recursively extracts references from `topicSections` and `references`
3. Builds complete documentation hierarchy
4. Saves to `index.json`

JSON Download

1. Reads index from discovery step
2. Downloads each page with rate limiting
3. Stores ETag and content hash in manifest
4. Creates `.docsync/manifest.json` for update tracking

Markdown Conversion

Converts Apple's JSON schema to Markdown:

```
JSON Structure          →  Markdown Output

metadata                →  YAML frontmatter
primaryContentSections  →  Main content
  heading              →  ## Headers
  paragraph            →  Text paragraphs
  codeListing          →  ```swift code blocks
  unorderedList        →  - Bullet lists
  orderedList          →  1. Numbered lists
topicSections          →  ## Topics section
references             →  Internal links
```

Update Detection

1. Checks ETags via HTTP HEAD requests
2. Compares with stored ETags in manifest
3. Only downloads changed pages
4. Generates changelog

Maintenance

Weekly/Monthly Updates

```
# Activate environment
source venv/bin/activate

# Check status
python scripts/update_status.py

# Check for updates
python scripts/update_check.py

# Download if available
python scripts/update_pull.py

# Review changes
cat .docsync/changelog/*.md
```

Future Features

- ☐ PDF generation (`04_markdown_to_pdf.py`)
- ☐ Update history viewer (`update_history.py`)
- ☐ Version rollback (`update_rollback.py`)
- ☐ Swift Book integration (`05_swift_book_download.sh`)
- ☐ Full-text search index
- ☐ Web UI for browsing

License & Disclaimer

This tool downloads publicly available Apple Developer Documentation for offline use.

Important: - Documentation content © Apple Inc. - For personal, non-commercial use only - Respects Apple's servers via rate limiting - Not affiliated with or endorsed by Apple

Contributing

Contributions welcome! Areas for improvement:

- Additional framework support
- PDF generation templates
- Search functionality
- Performance optimizations
- Bug fixes

Support

1. Check this README
2. Review script output for error messages
3. Use `--help` flag for script-specific documentation
4. Check existing issues

Credits

Built with: - aiohttp - Async HTTP - BeautifulSoup - HTML parsing - tqdm - Progress bars - PyYAML - YAML processing

---

Made for the developer community *Download once, reference forever*