OneDrive Standalone Suite on macOS — Architecture, Shared Library Sync & Duplicate Folder Diagnosis

OneDrive Standalone Suite on macOS

Architecture, Shared Library Sync & Duplicate Folder Diagnosis

> **Original conversation between Felipe Mayer Gomes and Claude (Anthropic)** Translated from Portuguese (Brazil) — September 2, 2026

Context

This document is a technical conversation exploring the internal architecture of OneDrive Standalone Suite on macOS, with a focus on diagnosing and preventing the duplication of "OneDrive - Shared Libraries" folders that occurs when adding new SharePoint synchronized libraries to a Mac that already has several libraries synced.

The team in question uses the **Sync button method** (not Add Shortcut) because they work heavily with Adobe InDesign, and the Sync method preserves absolute file paths — preventing broken image links when a task is reassigned to a different team member.

Part 1 — OneDrive Standalone Suite Architecture on macOS

Foundation: OneDrive is built on SharePoint

OneDrive is not an independent storage system. Every OneDrive account is a personal SharePoint Online site. At the physical layer, all three workloads (Teams, OneDrive, and SharePoint) store files in SharePoint document libraries. What you see in Teams, OneDrive, and SharePoint are presentation layers over the same underlying document libraries.

Group Containers on macOS

macOS uses **Group Containers** (sandboxing) to isolate app data. The OneDrive Standalone Suite uses the identifier:

```
~/Library/Group Containers/UBF8T346G9.OneDriveStandaloneSuite/
```

Depending on how OneDrive was installed:

- **Standalone version** `UBF8T346G9.OneDriveStandaloneSuite`

- **App Store version** `UBF8T346G9.OneDriveSyncClientSuite`

Inside this container, the internal cache structure (legacy engine) was:

```
~/Library/Group Containers/UBF8T346G9.OneDriveStandaloneSuite/
        OneDrive.noindex/
           OneDrive/
              Documents Sync/
              Camera Roll/
              [other synced folders]
```

The `.noindex` folder is the **internal cache** of the sync engine — it should not be confused with the files visible in Finder.

CloudStorage: the visible layer in Finder

Separate from the Group Container, OneDrive also uses the macOS **CloudStorage** folder (integration with Apple's File Provider):

```
~/Library/CloudStorage/
    OneDrive-[Organization]/              Personal OneDrive (Finder)
    OneDrive-SharedLibraries-[Org]/       Synced SP libraries
```

Sync Engine: Legacy vs. Native (2026)

Legacy Engine (2022–2026)

To support the File Provider, Microsoft built a bridge using a hidden cache folder that mirrored OneDrive content, so the sync engine could remain largely unchanged. Over time, this became the root cause of many reliability and performance issues.

Native Sync Engine (rolling out August–November 2026)

The Native Sync Engine removes the hidden cache folder entirely. It required rebuilding large parts of the codebase and creates a new sync platform that works natively with the macOS File Provider system. Key improvements:

- Faster initial sync and file change processing
- Lower CPU, memory, and battery usage
- Higher reliability through a simplified architecture that eliminates entire classes of sync problems
- Files and folders immediately visible in Finder

How to identify which engine is running: If the OneDrive version number ends with a value like `"(26K)"`, the Native Sync Engine is active.

Shared SharePoint Libraries in Finder

Sync Button method: Each synced SharePoint library appears as a subfolder under the organization's entry in Finder.

Add Shortcut to OneDrive method (Microsoft recommended): Content is accessible across all devices, not just the current Mac. Offers better performance but does not preserve absolute file paths — which is why the team uses the Sync method instead.

Part 2 — Diagnosing Duplicate "Shared Libraries" Folders

The Problem

When a new SharePoint library is added via the Sync button on a Mac that already has several libraries synced, the following duplicate entries appear in `~/Library/CloudStorage/`:

```
OneDrive - Shared Libraries - [Org]/          legitimate instance
OneDrive - Shared Libraries - [Org] 2/        DUPLICATE
   or
OneDrive - Shared Libraries - OneDrive - Shared Libraries/     nested loop name
```

The Real Root Cause: Corrupted File Provider Domain Registration

This is **not** a conflict between sync methods. It is a structural problem in the **macOS File Provider layer** — the interface between OneDrive and the operating system.

How it works internally

When OneDrive registers a SharePoint library via Sync, it creates a **File Provider Domain** in macOS. This domain is an entity managed by the OS itself (not by OneDrive), and it generates the entry in `~/Library/CloudStorage/`.

The normal flow:

1. User clicks "Sync" on a new library in SharePoint
2. OneDrive calls the macOS File Provider API to register a new domain
3. macOS creates the entry in ~/Library/CloudStorage/
4. OneDrive writes the domain metadata to its internal database
   (inside UBF8T346G9.OneDriveStandaloneSuite/)
5. Sync begins normally

The problem occurs at **step 4**: when OneDrive tries to register the new domain, macOS detects that a domain with the same tenant/organization identifier already exists. Instead of reusing the existing domain, it creates a **second domain with a numeric suffix** (`2`, `3`...) or with a nested name.

Why the legacy architecture caused this

Each synced library maintained state in **two places simultaneously**: in the macOS File Provider database **and** in the OneDrive internal cache inside the Group Container. When adding a new library, OneDrive needed to reconcile these two states. Any inconsistency — even a small one — between what macOS had registered and what OneDrive had in cache caused macOS to create a new domain instead of completing the existing one.

The Three Specific Triggers

Trigger 1: Automatic OneDrive app update during an active sync session

When OneDrive auto-updates in the background while libraries are being synced, the new binary may have a different version identifier for the File Provider Extension. macOS then treats the new File Provider as a different entity and creates a second domain for the same tenant.

How to verify: Ask affected users if the problem occurred shortly after an OneDrive update:

```
mdls -name kMDItemContentModificationDate /Applications/OneDrive.app
```

Trigger 2: Corrupted SQLite internal database in the Group Container

OneDrive maintains a SQLite database at:

```
~/Library/Group Containers/UBF8T346G9.OneDriveStandaloneSuite/
  Library/Application Support/OneDrive/settings/
```

This database registers all active sync domains. If a token for a shared library is expired, revoked, or corrupted during the process of adding a new library, OneDrive may write an incomplete domain record. On the next registration attempt, it creates a new one instead of completing the previous one.

Trigger 3: Stale credentials in the Keychain

OneDrive stores its authentication tokens in the macOS Keychain under `com.microsoft.OneDrive`. If this entry contains stale or malformed data, OneDrive cannot complete the handshake with the Microsoft 365 authentication service. When this happens during new library registration, OneDrive may create a "provisional" domain in the File Provider that cannot later be reconciled with the tenant's main domain.

Why the Reset Fixes Some Users but Not Others

The `ResetOneDriveApp.command` clears OneDrive's internal database and the Group Container cache, but **does not clear the File Provider state registered in macOS**. In some cases, the problem is stale File Provider state inside macOS itself, not inside OneDrive. The reset has no access to that layer.

For users where the reset **does not resolve** the issue, the duplicate domain is registered at the macOS level. The only supported route in that case is to collect a `sysdiagnose` and open a case with Apple, since macOS no longer exposes a reliable public tool to manually purge broken File Provider domains.

Part 3 — Prevention Strategy

Prevention 1: Control OneDrive auto-updates via LaunchDaemon scheduling

There is **no official Microsoft-supported plist key** to disable OneDrive auto-updates on macOS. The `DisableAutoUpdate` key does not exist as a supported option.

OneDrive Standalone uses two auto-update mechanisms registered as LaunchAgents/LaunchDaemons:

File	Location	Function
`com.microsoft.OneDriveStandaloneUpdater.plist`	`/Library/LaunchAgents/`	Per-user updater
`com.microsoft.OneDriveStandaloneUpdaterDaemon.plist`	`/Library/LaunchDaemons/`	System updater

To schedule updates to run only at specific times, replace the original plists with modified versions using the `StartCalendarInterval` key:

```
 <?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
 "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
   <key>Label</key>
   <string>com.microsoft.OneDriveStandaloneUpdater</string>
   <key>ProgramArguments</key>
   <array>
      <string>/Applications/OneDrive.app/Contents/StandaloneUpdater.app/Contents/Ma
   </array>
   <key>StartCalendarInterval</key>
   <array>
      <!-- Monday through Friday at 22:00 -->
      <dict><key>Hour</key><integer>22</integer><key>Minute</key><integer>0</intege
      <dict><key>Hour</key><integer>22</integer><key>Minute</key><integer>0</intege
      <dict><key>Hour</key><integer>22</integer><key>Minute</key><integer>0</intege
      <dict><key>Hour</key><integer>22</integer><key>Minute</key><integer>0</intege
      <dict><key>Hour</key><integer>22</integer><key>Minute</key><integer>0</intege
   </array>
   <key>RunAtLoad</key>
   <false/>
</dict>
</plist>
```

> **Important caveat:** Each time OneDrive updates, it may **recreate the original plists**, overwriting your modified version. For corporate environments, the most robust approach is to manage this via **MDM (Intune/Jamf)**, which can redistribute the modified plists automatically after each app update.

Prevention 2: Clean the Keychain before adding a new library

Run this script **before** any team member adds a new synced library:

```
 #!/bin/bash
# Clean stale OneDrive OAuth tokens from Keychain
# Run BEFORE adding a new library

echo "Cleaning OneDrive tokens from Keychain..."
security delete-generic-password -s "OneDrive Cached Credential" 2>/dev/null
security delete-generic-password -s "OneDrive Business Cached Credential" 2>/dev/null

echo "Restarting OneDrive..."
killall OneDrive 2>/dev/null
sleep 3
open /Applications/OneDrive.app

echo "Wait for OneDrive to re-authenticate before adding the new library."
```

Prevention 3: Verify domain integrity before adding a new library

```
 #!/bin/bash
# Preventive diagnostic: check registered File Provider domains
# Run before adding a new library

echo "=== Registered OneDrive File Provider Domains ==="
pluginkit -m -i com.microsoft.OneDrive.FileProvider 2>/dev/null || \
pluginkit -m -i com.microsoft.OneDrive-mac.FileProvider 2>/dev/null

echo ""
echo "=== CloudStorage entries ==="
ls -la ~/Library/CloudStorage/ | grep -i "onedrive"

echo ""
echo "=== Checking for duplicates ==="
SHARED=$(ls ~/Library/CloudStorage/ | grep -i "shared" | wc -l)
if [ "$SHARED" -gt 1 ]; then
    echo "    WARNING: $SHARED 'Shared Libraries' entries detected. Do NOT add a new lib
else
```

```
    echo "   Healthy state. You may proceed with adding the new library."
fi
```

Prevention 4: Safe protocol for adding a new library

Establish this protocol for the entire team:

Step	Action	Why
1	Pause sync (2h) from the OneDrive menu	Prevents incomplete state writes
2	Wait for all sync icons to show	Ensures the internal database is consistent
3	Run the diagnostic script above	Confirms no duplicate domains already exist
4	Add the new library from SharePoint	Clean domain registration
5	Wait for the new library to appear in Finder	Confirms the domain was registered correctly
6	Resume sync	Only after confirming healthy state

Part 4 — Native Sync Engine: Does It Solve the Problem Permanently?

Structurally: Yes

The hidden cache folder was identified as the root cause of many reliability and performance issues reported since 2022. The Native Sync Engine removes it entirely.

Instead of rewriting the sync engine to match the behavior expected by the File Provider, Microsoft bridged the two systems with a hidden cache folder that mirrored OneDrive content. The existing sync engine kept talking to this mirror, and a translation layer kept the File Provider satisfied. This decision reduced delivery risk and unlocked features like Known Folder Move — but also became the biggest source of reliability and performance problems that Mac users reported in the years that followed. Duplicate data, stuck syncs, Finder windows showing "Loading..." while folders materialized, and apps hanging while waiting for files to appear on disk — all traced back to this architecture.

Building the Native Sync Engine required rebuilding large parts of the codebase and creating a new sync platform that allows OneDrive to work natively with the macOS File Provider system.

Important Caveat for InDesign-Heavy Teams

Organizations that sync SharePoint document libraries will notice a **visual change in Finder**. Each synced SharePoint library will appear as a **separate location in the Finder sidebar**, instead of being grouped under a single organization entry.

Before (legacy engine):

```
OneDrive - Shared Libraries - [Org]/
        Library A/
```

```
        Library B/
```

After (Native Sync Engine):

```
 Library A          own entry in sidebar
Library B           own entry in sidebar
```

This means the **absolute file paths on disk will change** after migration to the Native Sync Engine. Any InDesign file that references the old path will need to have its image links relinked — which is exactly the problem the team was trying to avoid by using the Sync method in the first place.

Recommendation: Plan the migration to the Native Sync Engine as a coordinated team event, with time allocated to relink InDesign files after the transition.

Summary Table

Topic	Finding
Root cause of duplicate folders	Corrupted File Provider Domain registration in macOS
Why reset fixes some but not all	Reset clears OneDrive cache but not macOS File Provider state
Main triggers	App auto-update during sync, corrupted SQLite DB, stale Keychain tokens
`DisableAutoUpdate` plist key	Does NOT exist as an official supported option on macOS
Scheduling updates by time	Possible via `StartCalendarInterval` in LaunchDaemon plists
Native Sync Engine (Nov 2026)	Structurally eliminates the root cause
Impact on InDesign teams	Finder paths change after migration — InDesign links will need relinking

---

Document generated from a technical conversation on September 2, 2026. Original language: Portuguese (Brazil). Translated to English.