CloudSync Ultra Publishing Guide

This guide provides step-by-step instructions for building, signing, notarizing, and distributing CloudSync Ultra as a professional macOS application.

Target Audience: Developers releasing CloudSync Ultra

Last Updated: January 2026

App Version: 2.0.x

Table of Contents

1. Distribution Options Overview

CloudSync Ultra can be distributed through two primary channels, each with distinct advantages and trade-offs.

Direct Download Distribution

Users download the app directly from your website or GitHub Releases.

Aspect	Details
Control	Full control over releases and updates
Functionality	Complete rclone integration without sandbox restrictions
Revenue	No commission fees (100% retained)
Requirements	Code signing + Notarization required

Update Mechanism	Self-managed (Sparkle framework recommended)
User Trust	Moderate (Gatekeeper approval via notarization)

Best For: Power users, developers, users requiring full rclone functionality

Mac App Store Distribution

Users discover and download through Apple's App Store.

Aspect	Details
Control	Limited by Apple review process
Functionality	Sandbox restrictions may limit rclone features
Revenue	15-30% commission to Apple
Requirements	Full sandbox compliance, App Store review
Update Mechanism	Automatic via App Store
User Trust	Highest (Apple-vetted)

Best For: Maximum reach, mainstream users, enterprise deployment

Recommendation

Start with **Direct Download** to leverage full rclone functionality. Consider a **Mac App Store "Lite" version** later with reduced features to maximize reach.

2. Prerequisites

Before publishing, ensure all prerequisites are in place.

2.1 Apple Developer Account

Cost: $99 USD/year

Enrollment: developer.apple.com/programs

Processing Time: 24-48 hours (individual), up to 2 weeks (organization)

The Apple Developer Program provides:

- Developer ID certificates for direct distribution
- App Store distribution certificates
- Access to App Store Connect
- TestFlight beta testing
- Code signing capabilities

2.2 Xcode with Command Line Tools

Minimum Version: Xcode 15.0+

macOS Requirement: macOS 14.0+ (Sonoma)

Install command line tools if not already present:

```
xcode-select --install
```

Verify installation:

```
xcodebuild -version # Expected output: Xcode 15.x, Build version xxxxx
```

2.3 App-Specific Password

Required for notarization. Generate at appleid.apple.com:

- Sign in to your Apple ID account
- Navigate to **Security** section
- Under **App-Specific Passwords**, click **Generate Password**
- Label it "CloudSync Notarization" or similar
- **Save this password securely** - you cannot view it again

2.4 Developer ID Certificates

You need different certificates depending on distribution method:

Certificate Type	Purpose	Distribution Method
Developer ID Application	Sign apps for direct download	Direct Download
Developer ID Installer	Sign PKG installers	Direct Download
Mac App Distribution	Sign apps for App Store	App Store
Mac Installer Distribution	Sign packages for App Store	App Store

2.5 Team ID

Your Team ID is a 10-character alphanumeric identifier. Find it at:

- developer.apple.com/account > Membership Details
- Or in Xcode: **Settings > Accounts > Select Team > View Details**

3. Code Signing Setup

Code signing cryptographically validates that the app comes from a known developer and has not been tampered with.

3.1 Create Certificates

Developer ID Application Certificate (Direct Download)

- Open Xcode
- Go to **Settings > Accounts**
- Select your Apple ID and Team
- Click **Manage Certificates**
- Click **+** and select **Developer ID Application**
- Certificate is automatically downloaded to Keychain

Mac App Distribution Certificate (App Store)

Follow the same process but select **Mac App Distribution**.

Verify Certificates in Keychain

```
security find-identity -v -p codesigning
```

Expected output includes:

```
1) XXXXXXXXXX "Developer ID Application: Your Name (TEAM_ID)" 2) YYYYYYYYYY "Apple Distribution: Your Name (TEAM_ID)"
```

3.2 Configure Xcode Project

- Open `CloudSyncApp.xcodeproj`
- Select the **CloudSyncApp** target
- Go to **Signing & Capabilities** tab
- Configure signing:

For Direct Download:

- Signing Certificate: **Developer ID Application**
- Team: Select your Apple Developer team
- Uncheck "Automatically manage signing" for more control

For App Store:

- Signing Certificate: **Apple Distribution**
- Team: Select your Apple Developer team
- Check "Automatically manage signing"

3.3 Verify Code Signing

After building, verify the app is properly signed:

```
# Basic verification codesign -dv /path/to/CloudSyncApp.app # Detailed verification with
entitlements codesign -dv --verbose=4 /path/to/CloudSyncApp.app # Verify with requirements
codesign --verify --deep --strict /path/to/CloudSyncApp.app # Check entitlements codesign -d
--entitlements :- /path/to/CloudSyncApp.app
```

Expected output for properly signed app:

```
Executable=/path/to/CloudSyncApp.app/Contents/MacOS/CloudSyncApp
Identifier=com.yourcompany.CloudSyncApp Format=app bundle with Mach-O universal (x86_64 arm64)
CodeDirectory v=20500 size=xxxxx flags=0x10000(runtime) hashes=xx+x location=embedded Signature
size=xxxx Authority=Developer ID Application: Your Name (TEAM_ID) Authority=Developer ID
Certification Authority Authority=Apple Root CA Timestamp=...
```

3.4 Hardened Runtime

CloudSync Ultra must use hardened runtime for notarization. Verify this is enabled:

- In Xcode, select target > **Signing & Capabilities**
- Click **+ Capability**
- Add **Hardened Runtime** if not present

Or verify in build settings:

- `ENABLE_HARDENED_RUNTIME = YES`

4. Building for Release

4.1 Version Configuration

Before building, update version numbers:

- Open `CloudSyncApp.xcodeproj`
- Select project > **General** tab
- Update:
- **Version** (CFBundleShortVersionString): e.g., `2.0.17`
- **Build** (CFBundleVersion): e.g., `217`

Or edit via `Info.plist`:

```
<key>CFBundleShortVersionString</key> <string>2.0.17</string> <key>CFBundleVersion</key>
<string>217</string>
```

4.2 Archive Build (Xcode GUI)

- In Xcode, select **Product > Archive**
- Wait for build to complete
- Organizer window opens automatically
- Archive appears in the list

4.3 Archive Build (Command Line)

Navigate to the project directory and run:

```
# Clean previous builds xcodebuild clean \ -project CloudSyncApp.xcodeproj \ -scheme CloudSyncApp
\ -configuration Release # Create archive xcodebuild archive \ -project CloudSyncApp.xcodeproj \
-scheme CloudSyncApp \ -configuration Release \ -archivePath build/CloudSyncApp.xcarchive
```

4.4 Export App from Archive

For Direct Download

Create `ExportOptions-DirectDownload.plist`:

```
<?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>method</key>
<string>developer-id</string> <key>teamID</key> <string>YOUR_TEAM_ID</string>
<key>signingStyle</key> <string>automatic</string> <key>destination</key> <string>export</string>
</dict> </plist>
```

Export the archive:

```
xcodebuild -exportArchive \ -archivePath build/CloudSyncApp.xcarchive \ -exportPath build/export \
-exportOptionsPlist ExportOptions-DirectDownload.plist
```

For App Store

Create `ExportOptions-AppStore.plist`:

```
<?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>method</key>
<string>app-store</string> <key>teamID</key> <string>YOUR_TEAM_ID</string>
<key>uploadSymbols</key> <true/> <key>signingStyle</key> <string>automatic</string>
<key>destination</key> <string>upload</string> </dict> </plist>
```

Export for App Store:

```
xcodebuild -exportArchive \ -archivePath build/CloudSyncApp.xcarchive \ -exportPath build/appstore
\ -exportOptionsPlist ExportOptions-AppStore.plist
```

5. Notarization Process

Notarization is **required** for apps distributed outside the Mac App Store. Apple scans your app for malicious content and issues a ticket that Gatekeeper trusts.

5.1 Store Credentials in Keychain

Store your notarization credentials securely to avoid entering them repeatedly:

```
xcrun notarytool store-credentials "CloudSyncUltra" \ --apple-id "your@email.com" \ --team-id
"YOUR_TEAM_ID" \ --password "app-specific-password"
```

When prompted, enter your app-specific password. This creates a keychain profile named "CloudSyncUltra".

5.2 Prepare App for Notarization

Create a ZIP archive of the app:

```
# Navigate to export directory cd build/export # Create ZIP for notarization ditto -c -k
--keepParent CloudSyncApp.app CloudSyncApp.zip
```

5.3 Submit for Notarization

Submit and wait for completion:

```
xcrun notarytool submit CloudSyncApp.zip \ --keychain-profile "CloudSyncUltra" \ --wait
```

The `--wait` flag blocks until notarization completes (usually 5-15 minutes).

Without stored credentials:

```
xcrun notarytool submit CloudSyncApp.zip \ --apple-id "your@email.com" \ --team-id "YOUR_TEAM_ID"
\ --password "app-specific-password" \ --wait
```

5.4 Check Notarization Status

If you submitted without `--wait`, check status:

```
xcrun notarytool info <submission-id> \ --keychain-profile "CloudSyncUltra"
```

View submission history:

```
xcrun notarytool history \ --keychain-profile "CloudSyncUltra"
```

5.5 Retrieve Notarization Log

If notarization fails, retrieve the detailed log:

```
xcrun notarytool log <submission-id> \ --keychain-profile "CloudSyncUltra" \ notarization-log.json
```

Common issues in the log:

- Missing hardened runtime
- Unsigned nested binaries (like rclone)
- Invalid entitlements
- Missing timestamp

5.6 Staple the Ticket

After successful notarization, staple the ticket to the app:

```
xcrun stapler staple CloudSyncApp.app
```

Stapling embeds the notarization ticket so Gatekeeper can verify the app offline.

5.7 Verify Notarization

Verify the app passes Gatekeeper:

```
# Check Gatekeeper assessment spctl -a -v CloudSyncApp.app # Expected output: # CloudSyncApp.app: accepted # source=Notarized Developer ID
```

Verify stapling:

```
xcrun stapler validate CloudSyncApp.app
```

6. Creating DMG Installers

DMG (Disk Image) files provide a professional installation experience with drag-to-Applications functionality.

6.1 Using create-dmg (Recommended)

Install create-dmg:

```
# Via npm npm install -g create-dmg # Or via Homebrew brew install create-dmg
```

Create DMG:

```
create-dmg \ --volname "CloudSync Ultra" \ --volicon "CloudSyncApp.app/Contents/Resources/AppIcon.icns" \ --window-pos 200 120 \ --window-size 600 400 \ --icon-size 100 \ --icon "CloudSyncApp.app" 150 190 \ --hide-extension "CloudSyncApp.app" \ --app-drop-link 450 185 \ --no-internet-enable \ "build/CloudSyncUltra-2.0.17.dmg" \ "build/export/CloudSyncApp.app"
```

6.2 Using hdiutil (Manual)

For more control, use hdiutil directly:

```
# Create temporary DMG hdiutil create -volname "CloudSync Ultra" \ -srcfolder build/export/CloudSyncApp.app \ -ov -format UDRW \ build/CloudSyncUltra-temp.dmg # Mount the DMG hdiutil attach build/CloudSyncUltra-temp.dmg # Customize (add background, icon positions, symlink to Applications) # Use Finder or command line to arrange # Unmount hdiutil detach "/Volumes/CloudSync Ultra" # Convert to compressed, read-only DMG hdiutil convert build/CloudSyncUltra-temp.dmg \ -format UDZO \ -o build/CloudSyncUltra-2.0.17.dmg # Clean up rm build/CloudSyncUltra-temp.dmg
```

6.3 Notarize the DMG

Important: The DMG itself must also be notarized.

```
# Submit DMG for notarization xcrun notarytool submit build/CloudSyncUltra-2.0.17.dmg \
--keychain-profile "CloudSyncUltra" \ --wait # Staple the DMG xcrun stapler staple
build/CloudSyncUltra-2.0.17.dmg # Verify spctl -a -v build/CloudSyncUltra-2.0.17.dmg
```

6.4 DMG Best Practices

- **Consistent naming:** Use `AppName-Version.dmg` format
- **Include version in volume name:** Helps users identify outdated DMGs
- **Professional background:** Add a subtle background image with installation instructions
- **Applications symlink:** Always include a link to /Applications
- **Test installation:** Verify drag-and-drop works correctly

7. Mac App Store Submission

7.1 App Store Connect Setup

- Log in to App Store Connect
- Click **My Apps** > **+** > **New App**
- Complete the form:
- **Platform:** macOS
- **Name:** CloudSync Ultra
- **Primary Language:** English (U.S.)
- **Bundle ID:** com.yourcompany.CloudSyncApp
- **SKU:** CloudSyncUltra (unique identifier)
- **User Access:** Full Access

7.2 Required Assets

App Icon

- **Size:** 1024x1024 pixels
- **Format:** PNG
- **Requirements:** No alpha channel, no rounded corners (Apple adds them)

Screenshots

Required sizes for macOS:

Size	Display
1280 x 800	MacBook (13")
1440 x 900	MacBook Air (13")
2560 x 1600	MacBook Pro (13" Retina)
2880 x 1800	MacBook Pro (15" Retina)

Minimum: 3 screenshots

Maximum: 10 screenshots

App Description

- **Promotional Text:** Up to 170 characters (can be updated without new build)
- **Description:** Up to 4000 characters
- **Keywords:** Up to 100 characters, comma-separated
- **What's New:** Required for updates

Example description structure:

```
CloudSync Ultra is a powerful multi-cloud management app for macOS. KEY FEATURES: - Support for 40+ cloud providers - Dual-pane file browser for easy transfers - End-to-end encryption - Scheduled sync tasks - Native macOS design SUPPORTED SERVICES: Google Drive, Dropbox, OneDrive, Amazon S3, MEGA, Box, pCloud, Proton Drive, and 30+ more cloud storage providers. REQUIREMENTS: - macOS 14.0 or later - rclone (installed via Homebrew)
```

7.3 Upload Build

Using Xcode Organizer

- In Xcode, select **Window > Organizer**
- Select your archive
- Click **Distribute App**
- Select **App Store Connect**
- Choose **Upload**
- Follow prompts to upload

Using Command Line

```
# Export for App Store upload xcodebuild -exportArchive \ -archivePath build/CloudSyncApp.xcarchive \ -exportPath build/appstore \ -exportOptionsPlist ExportOptions-AppStore.plist # Upload using altool (deprecated but still works) xcrun altool --upload-app \ -f build/appstore/CloudSyncApp.pkg \ -t macos \ -u "your@email.com" \ -p "@keychain:AC_PASSWORD"
```

Or use the **Transporter** app from the Mac App Store for GUI-based uploads.

7.4 Submit for Review

- In App Store Connect, navigate to your app
- Select the uploaded build under **Build**
- Complete **App Review Information**:
- Contact information
- Demo account (if app requires login)
- Notes for reviewer
- Answer **Export Compliance** questions
- Click **Submit for Review**

7.5 Review Timeline

Submission Type	Expected Duration
New app	24-72 hours
App update	12-24 hours
First-time developer	May take longer

Tips for faster approval:

- Ensure all metadata is complete
- Provide clear demo account if needed
- Include notes explaining complex functionality
- Test thoroughly before submission

8. Troubleshooting

8.1 Code Signing Issues

Error: "No signing certificate found"

```
# List available certificates security find-identity -v -p codesigning # If empty, create
certificate in Xcode: # Settings > Accounts > Manage Certificates > +
```

Error: "Code signature invalid"

```
# Re-sign the app codesign --deep --force --verify --verbose \ --sign "Developer ID Application:
Your Name (TEAM_ID)" \ --options runtime \ CloudSyncApp.app
```

Error: "Embedded binary not signed"

Ensure all binaries (including rclone) are signed:

```
# Sign embedded binary codesign --force --verify --verbose \ --sign "Developer ID Application:
Your Name (TEAM_ID)" \ --options runtime \ CloudSyncApp.app/Contents/MacOS/rclone
```

8.2 Notarization Failures

Error: "The executable does not have the hardened runtime enabled"

Enable hardened runtime in Xcode:

- Select target > **Signing & Capabilities**
- Add **Hardened Runtime** capability

Error: "The signature of the binary is invalid"

Re-sign with timestamp:

```
codesign --force --deep --timestamp \ --sign "Developer ID Application: Your Name (TEAM_ID)" \
--options runtime \ CloudSyncApp.app
```

Error: "The binary uses an SDK older than the 10.9 SDK"

Update deployment target in Xcode to macOS 10.9 or later.

8.3 Gatekeeper Rejection

Error: "App cannot be opened because developer cannot be verified"

The app is either:

- Not signed correctly
- Not notarized
- Notarization ticket not stapled

Verify with:

```
spctl -a -vvv CloudSyncApp.app
```

8.4 App Store Rejection

Common rejection reasons:

- **Guideline 2.1:** App does not function as expected
- **Guideline 2.3:** Inaccurate metadata
- **Guideline 4.2:** Minimum functionality
- **Guideline 5.1.1:** Data collection without consent

Always read Apple's rejection message carefully and address each point.

9. Quick Reference

Complete Release Process (Direct Download)

```
#!/bin/bash # CloudSync Ultra Release Script VERSION="2.0.17" TEAM_ID="YOUR_TEAM_ID"
KEYCHAIN_PROFILE="CloudSyncUltra" # 1. Clean and Archive echo "Building archive..." xcodebuild
clean archive \ -project CloudSyncApp.xcodeproj \ -scheme CloudSyncApp \ -configuration Release \
-archivePath build/CloudSyncApp.xcarchive # 2. Export App echo "Exporting app..." xcodebuild
-exportArchive \ -archivePath build/CloudSyncApp.xcarchive \ -exportPath build/export \
-exportOptionsPlist ExportOptions-DirectDownload.plist # 3. Create ZIP for notarization echo
"Creating ZIP..." cd build/export ditto -c -k --keepParent CloudSyncApp.app CloudSyncApp.zip # 4.
Notarize App echo "Notarizing app..." xcrun notarytool submit CloudSyncApp.zip \
--keychain-profile "$KEYCHAIN_PROFILE" \ --wait # 5. Staple App echo "Stapling app..." xcrun
stapler staple CloudSyncApp.app # 6. Verify echo "Verifying..." spctl -a -v CloudSyncApp.app # 7.
Create DMG echo "Creating DMG..." cd ../.. hdiutil create -volname "CloudSync Ultra" \ -srcfolder
build/export/CloudSyncApp.app \ -ov -format UDZO \ "build/CloudSyncUltra-${VERSION}.dmg" # 8.
Notarize DMG echo "Notarizing DMG..." xcrun notarytool submit
"build/CloudSyncUltra-${VERSION}.dmg" \ --keychain-profile "$KEYCHAIN_PROFILE" \ --wait # 9.
Staple DMG xcrun stapler staple "build/CloudSyncUltra-${VERSION}.dmg" # 10. Final verification
echo "Final verification..." spctl -a -v "build/CloudSyncUltra-${VERSION}.dmg" echo "Release
complete: build/CloudSyncUltra-${VERSION}.dmg"
```

Essential Commands Reference

```
# Check signing identity security find-identity -v -p codesigning # Sign app codesign --force
--deep --timestamp \ --sign "Developer ID Application: Name (TEAM_ID)" \ --options runtime \
CloudSyncApp.app # Verify signing codesign -dv --verbose=4 CloudSyncApp.app # Store notarization
credentials xcrun notarytool store-credentials "ProfileName" \ --apple-id "email" --team-id
"TEAM_ID" # Submit for notarization xcrun notarytool submit App.zip \ --keychain-profile
"ProfileName" --wait # Staple notarization ticket xcrun stapler staple CloudSyncApp.app # Verify
Gatekeeper approval spctl -a -v CloudSyncApp.app # Create DMG hdiutil create -volname "App Name" \
-srcfolder CloudSyncApp.app \ -ov -format UDZO output.dmg
```

Appendix: File Locations

File	Purpose
`CloudSyncApp.xcodeproj`	Xcode project file
`CloudSyncApp/Info.plist`	App metadata and version
`CloudSyncApp/CloudSyncApp.entitlements`	Sandbox permissions
`ExportOptions-DirectDownload.plist`	Export settings for direct download
`ExportOptions-AppStore.plist`	Export settings for App Store
`build/CloudSyncApp.xcarchive`	Build archive
`build/export/CloudSyncApp.app`	Exported application
`build/CloudSyncUltra-X.X.X.dmg`	Final distributable DMG

CloudSync Ultra Publishing Guide

Created by Tech Writer

January 2026