PROTOTYPES LAB / PRACTICAL RELEASE HANDBOOK

iOS & macOS App Store Publishing

From a finished app to a reviewed release

A reusable guide for iPhone, iPad and Mac applications, with screenshot and movie tools, a privacy-policy template, and SQLCipher export-compliance worksheets.

Prepared for Ovidiu Carstea
Requirements checked on November 20, 2026. Recheck Apple's linked specifications before each release; device slots, SDK minimums and questionnaires change.

Release path

1 Prepare	2 Package	3 Publish
Audit app, data and encryption. Capture accurate UI.	Sign and archive in Xcode. Create store assets and metadata.	Upload and test the build. Submit, review and release.

Find what you need

Pages	Topic
2-5	Accounts, Xcode, App Store Connect and release steps
6-8	Required assets, screenshot sizes and video specifications
9-11	Screenshot, video resize, speed and trim scripts
12-13	Reusable privacy policy and per-app generation
14-16	SQLCipher / SQLiteTools encryption answers and Info.plist
17-18	Mac distribution, maintenance and final checklist
19-21	Primary sources and companion-file index

Keep this PDF beside the PublishingToolkit folder. The scripts operate on copies and refuse existing output files. Your original scripts and NATO Spell policy have been preserved.

PROTOTYPES LAB / 02

Before you start

Create a release worksheet

Record once	Decide for this release
Legal developer / team, app name, bundle identifier, SKU, primary language	Platforms, minimum OS, supported devices, version and build number
Support email and public support/privacy URLs	Countries, pricing, purchases, audience and account requirements
Third-party libraries / SDK versions and licenses	Data flows, storage, encryption, permissions and reviewers' test path

1. Verify access and agreements

Use an active Apple Developer Program team. Confirm your App Store Connect role can create records, upload builds and submit releases. The Account Holder must resolve agreements; paid apps and purchases also need the relevant banking and tax setup. [S7]

2. Check current toolchain rules

Apple currently requires iOS/iPadOS submissions built with Xcode 26 or later and the iOS/iPadOS 26 SDK or later. The deployment target must be iOS/iPadOS 13 or later. Use a release of Xcode accepted for App Store distribution, not merely one accepted for TestFlight. [S4] [S30]

3. Check region-specific items

Complete the current age-rating questions, content-rights and applicable compliance declarations. For EU distribution, complete and verify DSA trader information as applicable. Resolve any country-specific documentation requested by App Store Connect. [S4] [S11]

Keep a reproducible release folder

```
MyApp-1.0/
  release-notes.txt
  app-facts.json
  assets/screenshots/
  assets/previews/
  privacy/MyApp_privacy_policy.html
  compliance/encryption-assessment.txt
  review/test-instructions.txt
  build/archive-location.txt
```

Archive the source revision, dependency lockfiles, Xcode version, signing team, submitted assets and compliance decisions. Keep signing secrets and real user databases out of this folder.

PROTOTYPES LAB / 03

Prepare the app in Xcode

1. Set identity and signing
Choose the correct team and unique bundle ID for each target. Match the App Store Connect app record. Set the marketing version and a new build number. Enable only capabilities the app uses; verify Release signing and all embedded components.

2. Complete the user experience
Remove debug menus and placeholders. Test clean installation and upgrade from the previous release, database migration, errors, offline behavior, accessibility, localization and every purchase path. Use representative test data in screenshots and review materials.

3. Audit privacy integration
Provide accurate permission purpose strings. Inventory SDKs, network calls and required-reason APIs; include appropriate PrivacyInfo.xcprivacy entries and approved reasons. Verify applicable SDK signatures/manifests. A policy URL does not replace these build-time requirements. [S15]

4. Package icons and resources
Add the icon through the Xcode asset catalog or supported Icon Composer workflow. Verify release assets and localized resources are included. Make sure SQLite/SQLCipher dependencies and resources are bundled for every supported architecture. [S5] [S6]

Mac App Store checks

Enable App Sandbox and the minimum entitlements the app needs. Test file access from the sandboxed release build. For a database utility, exercise user-selected read/write access, save/export locations and security-scoped bookmarks if access persists across launches. Check any helper processes as well. [S25]

Choose architectures and minimum macOS version deliberately, and test those combinations. Supply macOS copyright metadata. Ensure bundled files do not carry the disallowed com.apple.quarantine attribute before preparing the final signed upload. [S4]

SQLiteTools release test
Use disposable database copies. Test plaintext to encrypted export, encrypted to plaintext export, wrong password, incompatible cipher settings, cancellation, disk-full behavior and migration. Verify the original database remains intact when the feature promises a new output file.

This checklist is a release procedure, not a code audit of your applications. Confirm each item against the application you actually ship.

PROTOTYPES LAB / 04

Create the store record

1. Create or select the app

In App Store Connect, open Apps and choose the new-app action. Select platform(s), app name, primary language, registered bundle ID, SKU and access. For an existing app, create its next version rather than a duplicate record. [S8]

2. Fill the product page

Prepare localized copy and screenshots for each supported platform. Keep the claims aligned with the submitted build. The support page must contain working contact information; use a public HTTPS privacy page. [S10] [S13]

Field	What to prepare
Name / subtitle	Name: 2-30 characters. Subtitle: up to 30.
Description	Plain text; up to 4,000 characters. Explain benefits and actual features.
Keywords	Up to 100 bytes; comma-separated relevant terms.
Promotional text	Optional; up to 170 characters.
What's New	Updates: up to 4,000 characters; describe concrete changes.
URLs	Support and privacy policy; marketing URL if useful.
Copyright	Year and rights holder; App Store Connect adds the symbol.
Review details	Contact name/email/international phone, test instructions, working demo login if required.

Limits and field behavior: [S10] [S11] [S31]

3. Configure commercial and compliance details

Select category, price, availability and release option. Complete privacy labels, age rating, content rights and applicable questionnaires. Configure IAP/subscription products, localization, pricing and review information when used.

Public URLs for review

Your local site at prototypeslab.local:8890 is not reachable by App Review or customers. Publish the finished support and privacy pages on your public website, then test them without signing in.

PROTOTYPES LAB / 05

Upload, test, submit and release

1. Archive the release
Select the intended app scheme and a distribution destination (generic iOS device / appropriate Mac destination, not an iOS Simulator). Choose Product > Archive. In Organizer, validate and use Distribute App > App Store Connect > Upload, following the available Xcode flow. [S9]

2. Wait for processing
Open the build in App Store Connect. Resolve validation errors and Missing Compliance. Match version/build and select the processed build on the version page. Uploading alone does not submit the app for review. [S9] [S12]

3. Test the uploaded artifact
Use TestFlight for install, launch, permission, database, networking and purchase checks. External testing may require Beta App Review. TestFlight approval and production App Review are separate. Test the build you will submit, not only a debug build. [S7]

4. Finish review access
Keep any backend online. Provide non-expiring review credentials or a complete permitted demo path, sample files and exact steps for unusual features. Explain how to reach paid features and attach helpful evidence. Never give reviewers production secrets or personal user data. [S13]

5. Submit all required items
Complete the version, select the build, include relevant first IAP/subscription submissions, add the version for review, and then perform the final submission to App Review. Address any unresolved items in the submission. [S12] [S30]

6. Handle review and release
Answer reviewer questions with reproducible steps. If a new binary is needed, increment the build and retest. After approval, release manually or use your chosen automatic/date-based option. Check the live page, installation and purchases. [S10] [S12]

Three separate milestones
Upload: Apple processes a binary.
Submit: App Review evaluates the selected build and metadata.
Release: the approved version becomes available to customers.

For updates, preserve upgrade compatibility and user data. Keep the previous release archive, crash symbols and a tested remediation plan; an App Store binary is not simply rolled back with a web deploy.

PROTOTYPES LAB / 06

Plan the required assets

Asset	Required?	Preparation
App icon	Yes	1024 x 1024 design canvas; package through Xcode.
Screenshots	Yes	1-10 per supplied device-size set; cover the required supported-platform slots.
App preview movie	Optional	0-3 per supported size/language; 15-30 seconds.
IAP review screenshot	If submitting IAP	Show the purchasable item/service; use a supported screenshot size.
IAP promotional image	For promotion / relevant offers	1024 x 1024 JPG/PNG, 72 dpi, RGB, flattened, square corners.

Asset requirements: [S3] [S5] [S6] [S27]

Icon workflow

Create a crisp 1024 x 1024 master. Use the icon workflow supported by your Xcode target: a traditional asset catalog or a layered Icon Composer design. Preview the appearances your target supports. The uploaded build supplies the store icon; a marketing image alone does not configure the app icon. [S5] [S6]

For a traditional macOS AppIcon asset, fill the slots Xcode requests, including 1x/2x variants. Keep a vector/layered master. Do not flatten legitimate Icon Composer foreground layers merely because screenshots require opaque output.

Capture before resizing

Capture the correct platform and device layout at native resolution. Use stable sample data, hide sensitive content, and demonstrate the main workflow. Capture iPad and Mac interfaces separately when their layouts differ. Avoid stretching an image to simulate another device.

A practical starting set
Prepare modern iPhone medium (1206 x 2622) and large (1320 x 2868) sets, iPad 13-inch (2064 x 2752) if supported, and Mac (2560 x 1600) if supported. Then verify the required slots in Media Manager and its permitted scaling behavior. These are suggested choices within Apple's accepted sizes, not a requirement to make every size.

Product-page headers, search creatives, In-App Events and custom pages are separate optional marketing surfaces with their own specifications. They are not prerequisites for a standard app listing.

PROTOTYPES LAB / 07

Screenshot size reference

Pixels, width x height. Use JPG/JPEG or opaque RGB PNG. P = portrait; L = landscape. For rows marked "swap", reverse width/height for landscape. [S1]

Device group	Accepted dimensions
iPhone Dynamic Island, large	1260 x 2736; 1290 x 2796; 1320 x 2868 (swap)
iPhone Dynamic Island, medium	1179 x 2556; 1206 x 2622 (swap)
iPhone Face ID, large	1284 x 2778; 1242 x 2688 (swap)
iPhone Face ID, medium	1170 x 2532; 1125 x 2436; 1080 x 2340 (swap)
iPhone Home Button, large / medium	1242 x 2208 / 750 x 1334 (swap)
iPhone 4-inch (legacy)	P: 640 x 1096 or 640 x 1136 L: 1136 x 600 or 1136 x 640
iPhone 3.5-inch (legacy)	P: 640 x 920 or 640 x 960 L: 960 x 600 or 960 x 640
iPad 13-inch / 12.9-inch	13: 2064 x 2752 or 2048 x 2732 (swap) 12.9: 2048 x 2732 (swap)
iPad 11-inch group	1488 x 2266; 1512 x 2316; 1668 x 2420; 1668 x 2388; 1640 x 2360 (swap)
iPad 10.5-inch	1668 x 2224 (swap)
iPad 9.7-inch (legacy)	P: 1536 x 2008 / 2048; 768 x 1004 / 1024 L: 2048 x 1496 / 1536; 1024 x 748 / 768
Mac (landscape, 16:10)	1280 x 800; 1440 x 900; 2560 x 1600; 2880 x 1800

Legacy paired heights mean without / with status bar. Required slots currently include iPhone Dynamic Island medium, iPad 13-inch when supported, and Mac. Large iPhone fallback rules also apply; verify Media Manager. Identical UI may use Apple's automatic scaling. [S1] [S3]

Future checkpoint: Apple lists iPhone Duo outer 1398 x 2034 and inner 2007 x 2853 (swap), with an April 2027 requirement tied to iOS 27.1 SDK or later. Recheck before that release window. [S1]

PROTOTYPES LAB / 08

App preview size reference

Device group	Portrait	Landscape
Modern iPhone (Face ID / Dynamic Island)	886 x 1920	1920 x 886
Home Button large / 4-inch	1080 x 1920	1920 x 1080
Home Button medium	750 x 1334	1334 x 750
iPad 13 / 11 / 10.5-inch	1200 x 1600	1600 x 1200
iPad 12.9-inch	1200 x 1600 or 900 x 1200	1600 x 1200 or 1200 x 900
iPad 9.7-inch	900 x 1200	1200 x 900
Mac	Not supported	1920 x 1080

iPhone 3.5-inch previews are unsupported. Preview dimensions are different from screenshot dimensions. [S2]

Recommended H.264 delivery settings

Property	Setting
Duration / size / frame rate	15-30 seconds; at most 500 MB; up to 30 fps
Video	H.264, progressive, High Profile Level 4.0; target 10-12 Mbps
Container	MP4, M4V or MOV for H.264
Audio	AAC, stereo, 256 kbps; 44.1 or 48 kHz
Poster frame	Default is 5 seconds; choose the clearest frame in App Store Connect

ProRes 422 HQ in MOV is another accepted route. This toolkit uses H.264 MP4. Allow for processing time; Apple notes previews may take up to 24 hours. [S2]

Show a readable, authentic demonstration

Use real app footage, deliberate cursor/touch actions and legible text. Cut pauses before accelerating. Keep captions inside the frame and verify audio rights. A 5x speed-up may meet the duration limit while making the product impossible to understand.

App Store files and YouTube are separate
Upload the actual preview file to App Store Connect. A YouTube URL is not a replacement. Your Prototypes Lab blog can embed a separate YouTube demo, including a longer version.

PROTOTYPES LAB / 09

Resize screenshots safely

The supplied resize_app_screenshots.sh processes PNG/JPG/JPEG files into a separate folder. It preserves aspect ratio, removes transparency, writes 8-bit RGB PNGs and refuses existing output files. It does not recurse into subfolders.

Install missing command-line tools

```
# With Homebrew already installed:
brew install imagemagick ffmpeg python

cd "/Users/ovidiucarstea/Desktop/AppStore Distribution"
cd PublishingToolkit
```

Run the reusable script

```
bash resize_app_screenshots.sh \
  "$HOME/Desktop/AppScreenshots/iPhone" \
  "$HOME/Desktop/AppAssets/iPhone-medium" \
  1206 2622 fit white

bash resize_app_screenshots.sh \
  "$HOME/Desktop/AppScreenshots/Mac" \
  "$HOME/Desktop/AppAssets/Mac" \
  2560 1600 fit black
```

fit keeps the entire image and adds padding. **fill** fills the canvas by cropping; use only after inspecting the edges. Change width/height using the reference table. Output names retain the source extension to prevent collisions (for example, capture.JPG.png).

Single-image equivalent

```
magick "input.png" -auto-orient -colorspace sRGB \
  -background white -alpha remove -alpha off \
  -resize 1206x2622 -gravity center \
  -extent 1206x2622 -depth 8 "PNG24:output.png"
```

The one-line command can overwrite its output; prefer the supplied wrapper for batch work. ImageMagick resize/extent behavior is documented in its command reference. [S29]

Your existing scripts
resize_iphone_screenshots.sh targets 1242 x 2688.
resize_iphone_screenshots_1280x800.sh targets a **Mac** screenshot size despite its name. Both originals are unchanged. Use fresh captures when the intended platform/layout differs.

PROTOTYPES LAB / 10

Resize and encode a movie

Run commands from PublishingToolkit. Keep make_app_preview.sh and app_preview.py together. The tool requires FFmpeg, FFprobe and Python 3; input/output paths can contain spaces.

Preserve an already suitable duration

```
bash make_app_preview.sh resize \
  "$HOME/Desktop/preview.mp4" \
  "$HOME/Desktop/iphone-preview.mp4" \
  886 1920
```

Resize mode accepts a source already 15-30 seconds long. Change the output canvas to 1920 886 for iPhone landscape, 1200 1600 for iPad portrait, or 1920 1080 for Mac. Choose the destination using Apple's device slot, not the input recording dimensions.

A direct FFmpeg resize command

```
# For an SDR source with audio and duration 15-30 seconds:
vf='scale=1920:1080:force_original_aspect_ratio=decrease'
vf="$vf:force_divisible_by=2"
vf="$vf,pad=1920:1080:(ow-iw)/2:(oh-ih)/2"
vf="$vf,setsar=1,fps=30,format=yuv420p"
ffmpeg -n -i "input.mp4" \
  -map 0:v:0 -map 0:a:0 -vf "$vf" \
  -c:v libx264 -profile:v high -level:v 4.0 \
  -b:v 10M -maxrate 12M -bufsize 24M \
  -c:a aac -b:a 256k -ar 48000 -ac 2 \
  -movflags +faststart "output.mp4"
```

Use the wrapper for silent movies, non-square source pixels and validation. It adds silent stereo audio when needed; --mute replaces original audio. HDR masters are rejected when identified; export a deliberately tone-mapped SDR master first.

Inspect the result

```
ffprobe -v error -show_streams -show_format \
  -of json "output.mp4"
open "output.mp4"
```

Check dimensions, 30 fps, H.264/yuv420p, AAC stereo at 48 kHz, duration and file size. Target bitrate is an encoder setting; simple images or silence can produce a lower measured average. Watch the full export for framing, sharpness, color and sound.

Filter/encoding behavior: [S28]

PROTOTYPES LAB / 11

Shorten a long movie

Option A: keep the whole sequence, increase speed

```
bash make_app_preview.sh speed \
  "$HOME/Desktop/long-demo.mp4" \
  "$HOME/Desktop/mac-preview.mp4" \
  1920 1080 --seconds 29.8
```

The tool measures the source and computes **speed factor = source duration / target duration**. A 120-second source to 30 seconds needs 4x speed. A 63-second source to 30 seconds needs 2.1x. A fixed factor of 2.1 is not a general 30-second conversion.

```
# Core transformation, shown for a 4x speed-up:
# Video timestamp filter:
setpts=(PTS-STARTPTS)/4
# Matching audio tempo filter:
atempo=2.0,atempo=2.0
```

The script chains audio tempo stages within 0.5-2.0, so high speed factors do not rely on one extreme atempo stage. It fits/pads the image, emits 30 fps, and validates the completed movie. It rejects speed targets longer than the source. [S28]

Option B: select a normal-speed excerpt

```
bash make_app_preview.sh trim \
  "$HOME/Desktop/long-demo.mp4" \
  "$HOME/Desktop/excerpt.mp4" \
  1920 1080 --start 45 --seconds 20
```

This retains seconds 45-65 and discards everything else. It checks that the requested range exists. For a better preview, edit several key actions into a short narrative before encoding.

Duration compression versus file compression
Speed: retain the sequence in less time.
Trim: retain only an excerpt.
Bitrate/codec compression: reduce bytes without changing runtime.
FFmpeg -t 30 limits output duration; by itself it does not accelerate a long movie.

Use 29.8 seconds as a practical margin below the 30-second cap. Append --mute if accelerated narration is unusable. Existing outputs are never replaced; choose a new name or intentionally remove an obsolete export yourself.

PROTOTYPES LAB / 12

Build an accurate privacy policy

PublishingToolkit/Application_privacy_policy.html is a reusable draft. It contains explicit placeholders, not assertions that every future app is offline or collects no data. The original NATO Spell policy in the parent folder remains intact.

Audit the shipped app before writing

Area	Facts to establish
Data and local storage	Files, text, databases, history, settings, Keychain; purpose and retention.
Networking and SDKs	Every endpoint, analytics/crash SDK, advertising, tracking and recipient.
Cloud and backups	iCloud/sync, device backups, exported files and user-directed sharing.
Identity and purchases	Accounts, account deletion, StoreKit transactions and receipts.
Permissions	Files, camera, microphone, photos, contacts, location; user controls.
Support and legal scope	Support mail/attachments, providers, contact details, applicable rights, children and markets.

Keep three disclosures consistent

Public privacy policy: describe real practices, including local storage, support messages, retention and deletion.
App Privacy labels: answer Apple's data questions for the app and integrated third parties.
Privacy manifests: supply the applicable machine-readable declarations for app/SDK behavior and required-reason APIs. [S13] [S14] [S15]

For Apple's labels, purely on-device processing generally is not collection. Off-device data retained beyond servicing a real-time request can be collection. Audit the actual flows and Apple's exceptions; neither an offline-looking UI nor SQLCipher proves "Data Not Collected". [S14]

Use precise security language
Describe exactly which database/files are encrypted and how keys are handled if verified. Do not promise that all data is encrypted, that deleted data vanishes from every backup, or that the developer cannot access keys unless the implementation supports it.

A reusable template cannot choose legal bases, child-data rules or jurisdiction-specific rights for every app. Complete those sections from the app's actual markets and operations before publication.

PROTOTYPES LAB / 13

Generate the per-app HTML

1. Make a facts file
Copy privacy-policy.example.json to a new app-specific JSON file. Replace every [REPLACE ...] value. State verified absence explicitly when a category does not apply; do not leave sections blank.

```
cp privacy-policy.example.json MyApp-privacy.json
# Edit MyApp-privacy.json with verified app facts.
python3 generate_privacy_policy.py \
  MyApp-privacy.json MyApp_privacy_policy.html
open MyApp_privacy_policy.html
```

2. Review the generated page
The generator escapes inserted HTML, requires all fields, checks the date/email format and refuses overwriting. It cannot establish whether supplied facts are true. Read the final policy as a customer, including all support-data and deletion statements.

3. Publish and connect it
Host the finished HTML on a publicly accessible HTTPS page. Add the URL in App Store Connect and an easily accessible in-app link. Test from a signed-out browser and a device outside your local network. [S13]

Reusable prompt for a future application

Copy and complete the bracketed fields
Prepare an application-specific privacy policy from [APP FACTS FILE] using the reusable template. Inspect [APP SOURCE FOLDER] and dependencies when available. Cover on-device data, network services, SDKs, analytics, diagnostics, accounts, purchases, cloud/backups, permissions, support messages, recipients, retention/deletion, security, children, privacy rights and contact details. Distinguish local processing from off-device collection. Do not invent privacy promises or contact details. List missing facts and keep the policy marked draft until resolved. Save a new [APP_NAME]_privacy_policy.html and provide matching App Privacy label guidance.

Before making it public

Check that the app name, developer identity, effective date and monitored contact are correct. Verify account deletion where applicable, retention periods, exported files and backup behavior. Read every third-party claim against the enabled SDK configuration. Revisit the policy whenever features or data flows change.

The full prompt and field-by-field instructions are also in README.txt and privacy-policy.example.json. No policy has been published as part of creating this guide.

PROTOTYPES LAB / 14

Encryption: answer the actual question

Both use cases use encryption. SQLCipher is third-party encryption in the app, even when iOS/macOS also encrypts the device. Apple's questionnaire, your Info.plist declaration and any export filing must describe the actual released product. [S16] [S20]

Question / decision	SQLCipher in either case
Does the app use encryption?	Yes.
Proprietary or non-standard algorithms?	No, only if all cryptography is standard and unmodified; audit other code/SDKs too.
Standard algorithms beyond Apple OS encryption?	Yes: SQLCipher is bundled, not solely an Apple OS encryption API.
Exempt from documentation / export rules?	App-specific assessment. Internal protection and an exposed encryption utility are materially different.
France / U.S.-only availability?	Use the actual selected distribution territories; never a convenient answer.

Complete the App Store Connect assessment

Set the app description and availability. Open App Information > App Encryption Documentation, or resolve Missing Compliance on the build. Follow the current questions, attach any requested documents, and retain the decision for that app/version. [S17]

Apple's documentation matrix lists a French declaration for standard cryptography outside Apple OS encryption when distributing in France. It lists CCATS and, for France, the French declaration for proprietary algorithms. Follow the assessment for your app rather than copying a fixed yes/no sequence. [S18]

Do not inherit the library's approval
Zetetic publishes SQLCipher classification information, but the developer is responsible for the application's classification. Its library declaration does not automatically satisfy a French declaration for your app. Do not submit a vendor reference as if Apple had approved your product. [S22]

This is technical preparation, not a legal export determination. Where the classification is unclear, verify current BIS/ANSSI rules or obtain export advice before selecting an exemption or distribution countries.

PROTOTYPES LAB / 15

Encryption descriptions to adapt

A. SQLCipher protects the app's own database

Paste-ready technical description
[APP_NAME] uses SQLCipher [VERSION] to encrypt its own local SQLite application database at rest using standard AES-256 encryption. This protects locally stored application data. The app does not expose database encryption/decryption as a user-facing general utility. It does not implement proprietary or non-standard cryptographic algorithms.

Optional, only if verified: "The app does not transmit database contents, encryption keys or passwords to the developer or to third-party services."

Confirm this against all features, SDKs, sync and backup behavior. Internal use is relevant to classification, but it is not by itself proof of an exemption. Record the SQLCipher version and actual cipher configuration. [S21] [S22]

B. SQLiteTools encrypts and decrypts user databases

Paste-ready technical description
SQLiteTools uses SQLCipher [VERSION] to let users encrypt and decrypt SQLite databases locally. Users select a database file and supply the required password or key. Encryption produces a SQLCipher-compatible encrypted database using standard AES-256 encryption; decryption requires the correct key and compatible cipher settings. The app does not implement proprietary or non-standard cryptographic algorithms.

Optional, only if verified: "Database files, passwords and encryption keys are processed on the user's device and are not uploaded to the developer or to third-party services."

Do not describe this as merely protecting the app's internal database: encryption is a user-facing capability. Do not claim compatibility with every encrypted SQLite format; other vendors' formats can differ. Adapt the text to the actual supported operations and algorithms.

Facts to keep with either description

Library/version and cipher settings; the data being encrypted; who chooses/holds keys; local/network/cloud behavior; other cryptographic components; territories; app-specific classification and supporting documents. If using standard SQLCipher 4 defaults, confirm them in the shipped build before adding algorithm details. [S21]

Copyable text and conditions are supplied separately in Encryption_Answers.txt. These paragraphs are drafts for verified app facts, not preapproved responses.

PROTOTYPES LAB / 16

Info.plist and export records

Set the key from the confirmed result

Confirmed app result	Info.plist declaration
No encryption, or all encryption is exempt	ITSAppUsesNonExemptEncryption = NO
The app uses non-exempt encryption	ITSAppUsesNonExemptEncryption = YES
Assessment not yet complete	Do not guess. Leaving the key absent causes App Store Connect to ask.

The presence of SQLCipher alone does not make YES universally correct, and absence of requested upload documents does not automatically justify NO. [S19] [S20]

```
<!-- Only when all app encryption is confirmed exempt: -->
<key>ITSAppUsesNonExemptEncryption</key>
<false/>

<!-- If the app uses non-exempt encryption instead: -->
<key>ITSAppUsesNonExemptEncryption</key>
<true/>
```

Use only the applicable entry, not both. For generated Info.plist builds, set the equivalent target build property so it reaches the archived app. Inspect the archive's Info.plist before uploading.

If Apple supplies an approval code

```
<key>ITSEncryptionExportComplianceCode</key>
<string>APPLE_PROVIDED_CODE</string>
```

Replace the placeholder only with the exact code Apple provides after the relevant documentation approval. An ECCN, SQLCipher ANSSI number or invented value is not a substitute. [S20]

Separate the store workflow from legal obligations

Apple's result does not replace applicable U.S. or French export obligations. Establish the product classification, destination restrictions, exceptions and any reporting duties. Applicable BIS annual self-classification reports are generally due February 1 for the previous year; scope and exceptions matter. [S23] [S24]

Reassess on change
Revisit the answers when adding encryption tools, changing SQLCipher/configuration, exposing keys, introducing cloud/network features, adding other cryptographic SDKs or changing countries. Keep the assessment and any approvals with the release record.

PROTOTYPES LAB / 17

Mac distribution and maintenance

Choose the distribution route deliberately

Mac App Store	Direct distribution outside the store
App Sandbox and appropriate entitlements. Archive and upload to App Store Connect. App Review and store release.	Developer ID signing and Hardened Runtime as applicable. Notarize the distributable and staple its ticket. Host an installer/archive and manage updates.

The Mac App Store route does not require a separate Developer ID notarization submission. Direct notarization is an automated security assessment and does not replace App Review or create an App Store listing. [S25] [S26]

Optional direct-distribution example

```
# For an already correctly signed distributable:
# Store credentials in a named Keychain profile first.
xcrun notarytool submit "MyApp.dmg" \
  --keychain-profile "MyNotaryProfile" --wait

# Continue only after an Accepted result:
xcrun stapler staple "MyApp.dmg"
xcrun stapler validate "MyApp.dmg"
```

Follow Apple's signing/notarization instructions for the chosen package type. Test Gatekeeper behavior on a clean Mac. Do not embed passwords or API private keys into scripts. Apple's notarization service uses notarytool, not the obsolete altool upload route. [S26] [S4]

After every release

1. Check the public result

Inspect the store page, all localizations, screenshots, preview poster, privacy/support links, prices and availability. Download the public app on a clean device.

2. Monitor and support

Review crash reports and support messages. Confirm database upgrades, purchases and restore behavior. Keep review contact and demo access current for future updates.

3. Prepare the next version

Increment version/build appropriately, write precise release notes, rerun data/encryption audits and refresh assets where UI changed. Retain archives, symbols, dependency versions and compliance evidence.

PROTOTYPES LAB / 18

Final submission checklist

Use this as the release gate. Mark items only after checking the uploaded candidate and the live App Store Connect record.

Check	Evidence
App identity and signing	Correct team, bundle ID, platform, version and unique build.
Toolchain / compatibility	Current accepted SDK/Xcode; tested minimum OS and architectures.
Functional test	Fresh install, upgrade, database migration, errors, offline and accessibility.
Privacy in the build	Purpose strings, manifests, SDK audit and permissions match behavior.
Public policy and support	Working HTTPS URLs; policy facts and contact verified; in-app access.
Privacy labels	App and third-party data flows declared consistently.
Store images	Correct device sets, opaque RGB, accurate readable UI and localization.
Preview movie (if used)	Correct size/codec/frame rate; 15-30 seconds; file under 500 MB; watched.
Store text / audience	Description, categories, keywords, age rating, rights and compliance.
Purchases (if used)	Products, review screenshots, paywall/restore tests and review inclusion.
Encryption	App-specific answers, countries, documents and Info.plist agree.
Review access	Working credentials/sample files, contact and clear test instructions.
Pricing and release	Correct territories, price, agreements and release timing.
Submission state	Processed build selected; final Submit to App Review completed.

Keep a record
Save the source revision, archive location, submitted assets, metadata, privacy-policy version, encryption determination, test notes and review correspondence for this release.

REFERENCE / VERIFIED NOVEMBER 20, 2026

Primary references

Source labels throughout the guide are clickable. Apple's current interface and linked specifications take precedence if requirements change.

S1 - Screenshot specifications
https://developer.apple.com/help/app-store-connect/reference/app-information/screenshot-specifications

S2 - App preview specifications
https://developer.apple.com/help/app-store-connect/reference/app-information/app-preview-specifications

S3 - Upload app previews and screenshots
https://developer.apple.com/help/app-store-connect/manage-app-information/upload-app-previews-and-screenshots/

S4 - Upcoming requirements
https://developer.apple.com/news/upcoming-requirements/

S5 - Add an app icon
https://developer.apple.com/help/app-store-connect/manage-app-information/add-an-app-icon

S6 - Human Interface Guidelines: App icons
https://developer.apple.com/design/human-interface-guidelines/app-icons/

S7 - App Store Connect workflow
https://developer.apple.com/help/app-store-connect/get-started/app-store-connect-workflow/

S8 - Add a new app
https://developer.apple.com/help/app-store-connect/create-an-app-record/add-a-new-app/

S9 - Upload builds
https://developer.apple.com/help/app-store-connect/manage-builds/upload-builds/

S10 - Platform version information
https://developer.apple.com/help/app-store-connect/reference/app-information/platform-version-information

S11 - App and compliance information
https://developer.apple.com/help/app-store-connect/reference/app-information/app-and-compliance-information

S12 - Overview of submitting for review
https://developer.apple.com/help/app-store-connect/manage-submissions-to-app-review/overview-of-submitting-for-review/

S13 - App Review Guidelines
https://developer.apple.com/app-store/review/guidelines/

S14 - App privacy details
https://developer.apple.com/app-store/app-privacy-details/

S15 - Describing use of required reason API
https://developer.apple.com/documentation/BundleResources/describing-use-of-required-reason-api

S16 - Overview of export compliance
https://developer.apple.com/help/app-store-connect/manage-app-information/overview-of-export-compliance

REFERENCE / VERIFIED NOVEMBER 20, 2026

Primary references - continued

Source labels throughout the guide are clickable. Apple's current interface and linked specifications take precedence if requirements change.

S17 - Determine and upload app encryption documentation
https://developer.apple.com/help/app-store-connect/manage-app-information/determine-and-upload-app-encryption-documentation

S18 - Export compliance documentation for encryption
https://developer.apple.com/help/app-store-connect/reference/app-information/export-compliance-documentation-for-encryption

S19 - ITSAppUsesNonExemptEncryption
https://developer.apple.com/documentation/bundleresources/information-property-list/itsappusesnonexemptencryption

S20 - Complying with encryption export regulations
https://developer.apple.com/documentation/security/complying-with-encryption-export-regulations

S21 - SQLCipher design
https://www.zetetic.net/sqlcipher/design/

S22 - SQLCipher export compliance
https://www.zetetic.net/sqlcipher/export-compliance/

S23 - BIS: Annual self-classification
https://media.bis.gov/learn-support/encryption-controls/annual-self-classification

S24 - ANSSI: Controle de moyen de cryptologie
https://cyber.gouv.fr/reglementation/reglementation-identite-confiance-numerique/controles-reglementaires-cryptographie/controle-moyen-de-cryptologie/

S25 - Distributing software on macOS
https://developer.apple.com/macos/distribution/

S26 - Notarizing macOS software before distribution
https://developer.apple.com/documentation/security/notarizing-macos-software-before-distribution

S27 - In-App Purchase information
https://developer.apple.com/help/app-store-connect/reference/in-app-purchases-and-subscriptions/in-app-purchase-information

S28 - FFmpeg filter reference
https://ffmpeg.org/ffmpeg-filters.html

S29 - ImageMagick command-line options
https://imagemagick.org/command-line-options/

S30 - App Store Connect release notes
https://developer.apple.com/help/app-store-connect/release-notes/

S31 - App Store product page
https://developer.apple.com/app-store/product-page/

PROTOTYPES LAB / 21

Companion files and usage notes

File in PublishingToolkit	Purpose
README.txt	Setup, command examples and reusable writing prompt.
resize_app_screenshots.sh	Batch resize/crop/pad and flatten PNG/JPG/JPEG screenshots.
make_app_preview.sh app_preview.py	Movie resize, automatic speed-to-duration and excerpt modes. Keep together.
Application_privacy_policy.html	Reusable, unfinished HTML template with explicit placeholders.
privacy-policy.example.json	App facts worksheet; every placeholder must be completed.
generate_privacy_policy.py	Render a new application policy without overwriting existing files.
Encryption_Answers.txt	Editable descriptions, conditional answers and export worksheet.

How the supplied commands differ from your originals

The original screenshot scripts remain available. The new tool supports more image extensions, explicitly removes alpha, accepts different canvas sizes and refuses replacement of existing outputs. Use accurate device captures even when a resize technically matches Apple's dimensions.

The movie tool calculates the duration factor instead of assuming 2.1x. Resize, speed and trim are separate modes. Aspect-preserving fit avoids stretching. The export uses an explicit bitrate target and stereo AAC settings, then checks key output properties.

Boundaries of the automation

The tools do not log in to Apple, submit an app, classify exports or certify a privacy policy. Movie processing uses the first video/audio stream and handles standard SDR recordings. Edit unusual color, audio or timeline material in a media editor first. Review every asset after conversion.

Preserved source material

Your original ShellScripts.txt, resize_iphone_screenshots.sh, resize_iphone_screenshots_1280x800.sh, Application_privacy_policy.html and App Store Connect Encryption.md were consulted and preserved. Their older assumptions are clarified in this guide; their contents were not rewritten.

Location: /Users/ovidiucarstea/Desktop/AppStore Distribution
Companions: PublishingToolkit/ and PublishingToolkit.zip