

#WWDC18

# Your Apps and the Future of macOS Security

Pierre-Olivier Martel, Security Engineering Manager

Kelly Yancey, OS Security

Garrett Jacobson, Trusted Execution

System Security Improvements

User Consent for Data Access

Enhanced Runtime Protections

Developer ID and Notarized Apps

# **System Security Improvements**

# System Security Improvements

## Extension to System Integrity Protection

- Stronger code signing enforcement for platform binaries
- Libraries/Frameworks/Plugins loaded by system processes must be signed by Apple

## Exceptions for legacy system extension points

# Trusted Event Dispatching

Most security decisions are made through system UI

- User intent (open/save dialogs, drag and drop)
- User consent (security dialogs, configuration changes)

Need to differentiate between the user deciding and software impersonating them

New approval mechanism for users to enable software controlling the UI on their behalf

Configurable in the Security and Privacy preference pane, in the Accessibility list

# Trusted Event Dispatching



# Trusted Event Dispatching

## Impacted APIs

```
IOHIDPostEvent(...);  
IOHIDSetMouseLocation(io_connect_t connect, int x, int y);  
  
CGEvent.post(tap: CGEventTapLocation)  
CGEventTap.tapCreate(..., options: CGEventTapOptions, ...) // when called without .listenOnly
```

# Safari / WebKit

## Safari

- Sandbox adoption for the app and satellite processes

## WebKit

- Audit and removal of risky dependencies from existing sandboxes
- Heap and JIT hardening (isolated heaps, ...)







**User Consent**



**Enhanced Runtime**



**Notarized Apps**

# User Consent for Data Access

Kelly Yancey, OS Security

# User Data Protections

Prompting categories

Location Services

Contacts

Calendars

Reminders

Photos

# User Data Protections

Prompting categories

Location Services

Contacts

Calendars

Reminders

Photos



# User Data Protections

Prompting categories

Location Services

Contacts

Calendars

Reminders

Photos

```
// User Data Protections - Prompting on Filesystem Access

let picturesURL = try FileManager.default.url(for: .picturesDirectory,
                                             in: .userDomainMask,
                                             appropriateFor: nil,
                                             create: false)

if let enumerator = FileManager.default.enumerator(atPath: picturesURL.path) {
    enumerator.forEach({ (entry) in
        guard let path = entry as? String else { return }
        ...
    })
}
```

```
// User Data Protections - Prompting on Filesystem Access
```

```
let picturesURL = try FileManager.default.url(for: .picturesDirectory,  
                                             in: .userDomainMask,  
                                             appropriateFor: nil,
```

```
if let enumerator = FileManager.default.enumerator(at: picturesURL, includingPropertiesForKeys: [.path),  
            enumerator.forEach({  
                guard let path =  
                    ...  
            })  
}
```



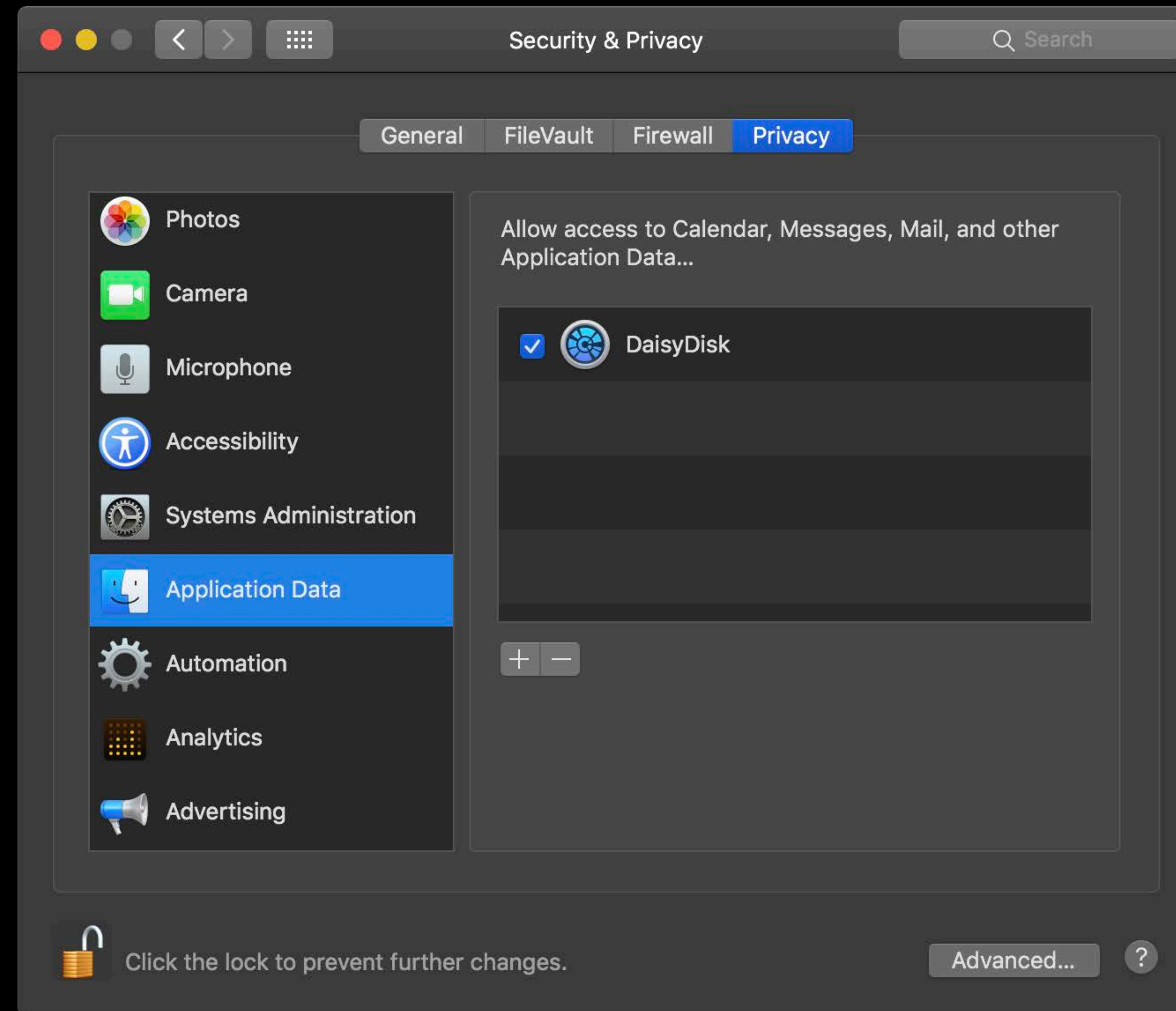
```
.path) {
```



# User Data Protections

Pre-approval

NEW



# User Data Protections

Prompting with purpose



# User Data Protections

Prompting with purpose



# User Data Protections

Info.plist keys

- `NSLocationUsageDescription`
- `NSCalendarsUsageDescription`
- `NSContactsUsageDescription`
- `NSRemindersUsageDescription`
- `NSPhotoLibraryUsageDescription`

# User Data Protections

Prompting categories

Location Services

Contacts

Calendars

Reminders

Photos

# User Data Protections

All data categories

Location Services

Contacts

Calendars

Reminders

Photos

Mail

Messages

Safari Browsing History

HTTP Cookies

Call History

iTunes Backups

Time Machine Backups

# User Data Protections

All data categories



NEW

Location Services

Contacts

Calendars

Reminders

Photos

Mail

Messages

Safari Browsing History

HTTP Cookies

Call History

iTunes Backups

Time Machine Backups

# User Data Protections

All data categories

Location Services

Contacts

Calendars

Reminders

Photos

Mail

Messages

Safari Browsing History

HTTP Cookies

Call History

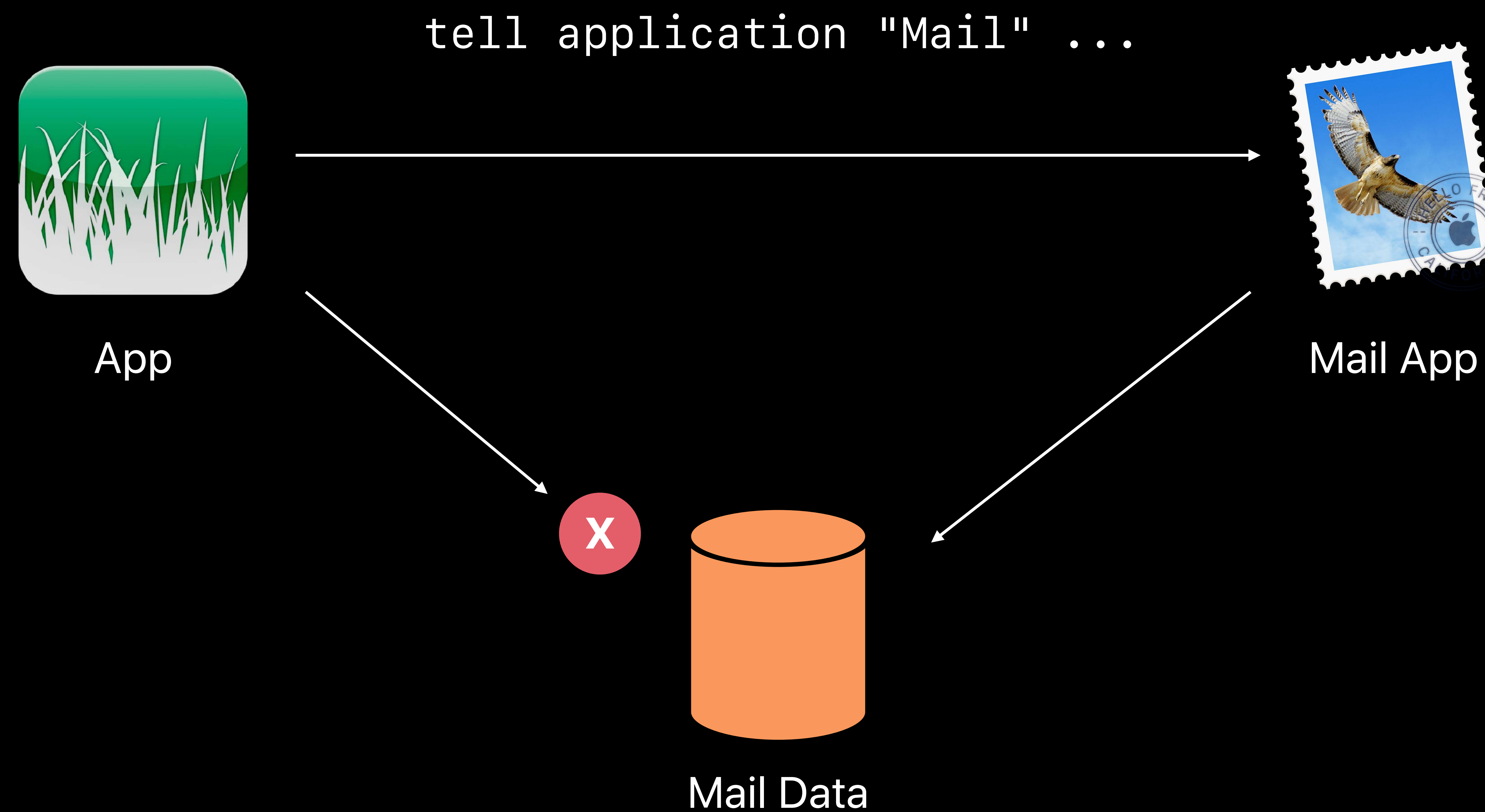
iTunes Backups

Time Machine Backups



# User Data Protections

Accessing mail data



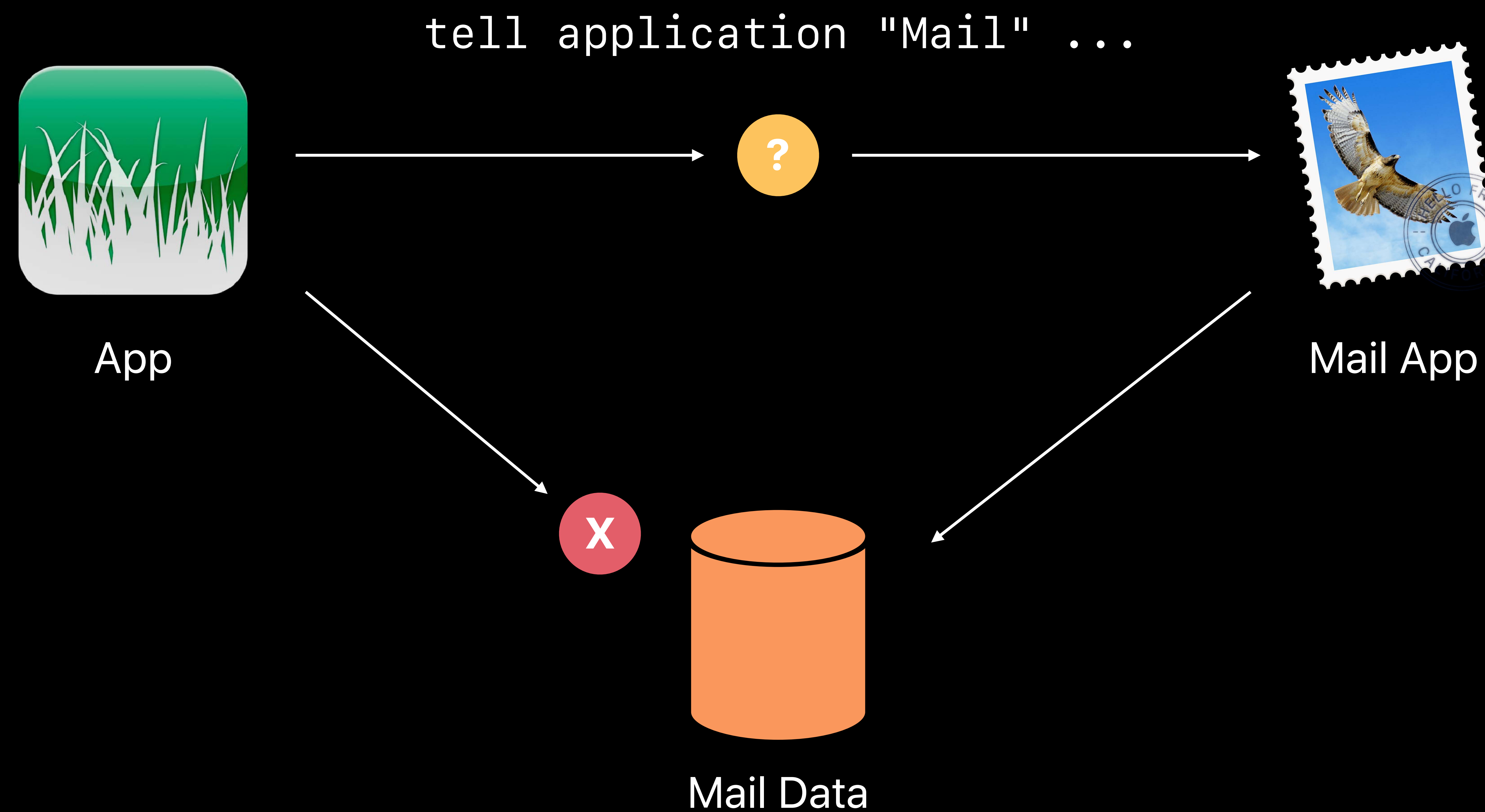
# User Data Protections

Accessing mail data



# User Data Protections

## Accessing Mail data



# User Data Protections

## Automation

NEW

User authorization required to automate other apps via Apple Events

Exceptions, including:

```
NSWorkspace.shared.openFile("/var/log/system.log")
```

```
NSWorkspace.shared.open(URL(string:"https://developer.apple.com/wwdc/")!)
```

```
NSWorkspace.shared.launchApplication("TextEdit")
```

Manageable via the Security and Privacy preference pane

# User Privacy

Camera and microphone

NEW



**"Watch Grass Grow" would like to access the camera.**

The grass watches you!

Don't Allow

OK

# User Privacy

Camera and microphone

Initiating capture requires user authorization similar to iOS

Applies to devices supported by built-in drivers

AVFoundation provides API for querying authorization status

```
let status = AVCaptureDevice.authorizationStatus(for: .video)
```

```
let status = AVCaptureDevice.authorizationStatus(for: .audio)
```

# User Privacy

## Camera and microphone

```
public enum AVAuthorizationStatus : Int {  
  
    case notDetermined  
  
    case restricted  
  
    case denied  
  
    case authorized  
  
}
```

# User Privacy

## Camera and microphone

AVFoundation provides API to pre-flight authorization

```
AVCaptureDevice.requestAccess(for: .video) { (authorized) in
    // ...
}
```

```
AVCaptureDevice.requestAccess(for: .audio) { (authorized) in
    // ...
}
```



# User Privacy

Info.plist keys

- NSCameraUsageDescription
- NSMicrophoneUsageDescription

# User Data and Privacy Protections

Recap

Location Services

Mail

Camera

Contacts

Messages

Microphone

Calendars

Safari Browsing History

Automation

Reminders

HTTP Cookies

Photos

Call History

iTunes Backups

Time Machine Backups

# User Data and Privacy Protections

Recap

Location Services

Mail

Camera

Contacts

Messages

Microphone

Calendars

Safari Browsing History

Automation

Reminders

HTTP Cookies

Photos

Call History

iTunes Backups

Time Machine Backups

# User Data and Privacy Protections

Recap

Location Services

Mail

Camera

Contacts

Messages

Microphone

Calendars

Safari Browsing History

Automation

Reminders

HTTP Cookies

Photos

Call History

iTunes Backups

Time Machine Backups

# User Data and Privacy Protections

Testing your apps

```
tccutil reset Calendars
```

```
tccutil reset Contacts
```

```
tccutil reset Reminders
```

```
tccutil reset Photos
```

```
tccutil reset AppleEvents
```

```
tccutil reset Camera
```

```
tccutil reset Microphone
```

```
tccutil reset All
```

# User Data and Privacy Protections

## Summary

Ensure approval prompts are presented in context

Add Info.plist keys explaining the reason your app needs the user's data

Access approval-gated resources from threads other than the main thread

Gracefully handle failure to access approval-gated resources

Be responsible with the user's personal data

# Enhanced Runtime Protections

Pierre-Olivier Martel, Security Engineering Manager

# Enhanced Runtime



NEW

New opt-in mechanism available with the 10.14 SDK

Enables additional protections similar to those enabled on system binaries

Configurable with unrestricted entitlements

Manageable from Xcode

Safe to enable on binaries deployed on older versions of macOS

Versioning scheme in place to support future revisions of the policies



# Enhanced Runtime

## Code signing

All executable pages must be backed by a valid code signature

```
com.apple.security.cs.allow-jit
```

Enable access to a JIT region (MAP\_JIT)

```
com.apple.security.cs.allow-unsigned-executable-memory
```

Enable executable mapping without a signature

```
com.apple.security.cs.disable-executable-page-protection
```

Disable all code signing protection

# Enhanced Runtime

## Library validation

The signature for all libraries, frameworks, and plugins validated at runtime

By default, only code signed by Apple or by the same Team ID is allowed

```
com.apple.security.cs.disable-library-validation
```

Allow loading of libraries signed by different Team IDs

# Enhanced Runtime

## Debugging

Applications cannot debug other apps or be debugged themselves unless they explicitly declare that capability

```
com.apple.security.get-task-allow
```

```
com.apple.security.cs.debugger
```

```
com.apple.security.cs.allow-dyld-environment-variables
```

Allow your app to be debugged

Allow your app to debug other apps

Enable DYLD variables for your apps

# Enhanced Runtime

## Resource access

Attempts to access protected resources without predeclaring intent will result in a crash

Add the appropriate entitlement for each protected resource that your apps needs to access

Access still subject to user approval

# Enhanced Runtime

## Resource access

```
com.apple.security.device.audio-input
```

Audio input and microphone

```
com.apple.security.device.camera
```

Any camera exposed via AVFoundation

```
com.apple.security.personal-information.location
```

Location

```
com.apple.security.personal-information.addressbook
```

Contacts

```
com.apple.security.personal-information.calendars
```

Calendars and Reminders

```
com.apple.security.personal-information.photos-library
```

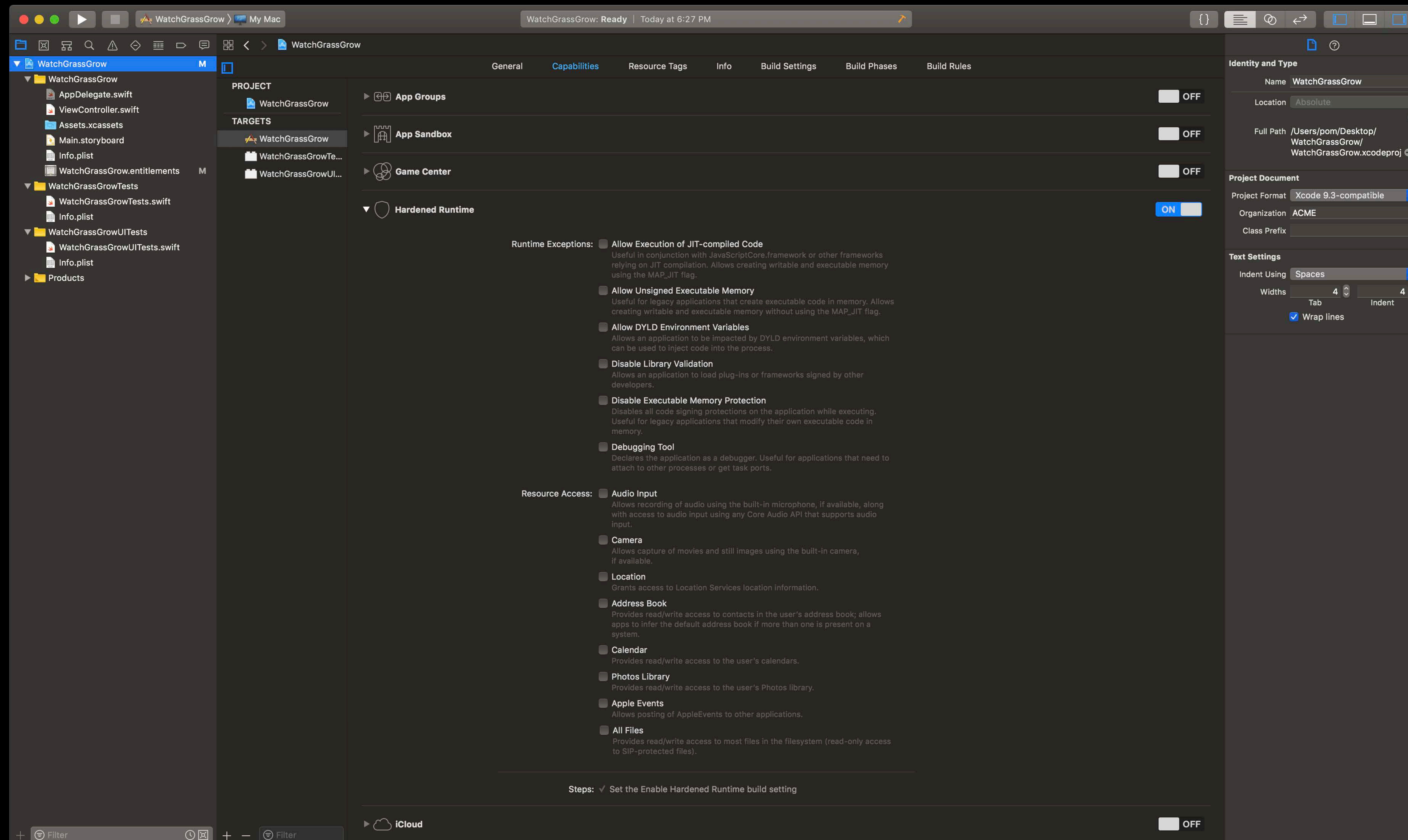
Apple Photos library

```
com.apple.security.automation.apple-events
```

Sending Apple Events to other apps


# Developer Workflow

## Xcode



# Developer Workflow

## Xcode

▼  **Hardened Runtime** ON

**Runtime Exceptions:**

- Allow Execution of JIT-compiled Code**  
Useful in conjunction with `JavaScriptCore.framework` or other frameworks relying on JIT compilation. Allows creating writable and executable memory using the `MAP_JIT` flag.
- Allow Unsigned Executable Memory**  
Useful for legacy applications that create executable code in memory. Allows creating writable and executable memory without using the `MAP_JIT` flag.
- Allow DYLD Environment Variables**  
Allows an application to be impacted by DYLD environment variables, which can be used to inject code into the process.
- Disable Library Validation**  
Allows an application to load plug-ins or frameworks signed by other developers.
- Disable Executable Memory Protection**  
Disables all code signing protections on the application while executing. Useful for legacy applications that modify their own executable code in memory.
- Debugging Tool**  
Declares the application as a debugger. Useful for applications that need to attach to other processes or get task ports.

**Resource Access:**

- Audio Input**  
Allows recording of audio using the built-in microphone, if available, along with access to audio input using any Core Audio API that supports audio input.
- Camera**  
Allows capture of movies and still images using the built-in camera, if available.
- Location**  
Grants access to Location Services location information.
- Address Book**  
Provides read/write access to contacts in the user's address book; allows apps to infer the default address book if more than one is present on a system.

```
// Developer Workflow - Terminal
```

```
# Signature
```

```
$> codesign --sign "Developer ID" --options runtime WatchGrassGrow.app
```

```
WatchGrassGrow.app: signed app bundle with Mach-O thin (x86_64) [com.acme.WatchGrassGrow]
```

```
# Verification
```

```
$> codesign --display --verbose=2 WatchGrassGrow.app
```

```
Executable=WatchGrassGrow.app/Contents/MacOS/WatchGrassGrow
```

```
Identifier=com.acme.WatchGrassGrow
```

```
Format=app bundle with Mach-O thin (x86_64)
```

```
CodeDirectory v=20500 size=566 flags=0x10000(runtime) hashes=11+3 location=embedded
```

```
Signature size=4605
```

```
Info.plist entries=22
```

```
TeamIdentifier=XXXXXXXXXX
```

```
Runtime Version=10.14.0
```

```
Sealed Resources version=2 rules=13 files=20
```

```
Internal requirements count=0 size=12
```



# Developer ID and Notarized Apps

Garrett Jacobson, Trusted Execution

# Identify and Block Malicious Software

# App Notarization

Block malicious software before distribution

Keep the flexibility of the Developer ID program

# Notary Service



NEW

Performs automated security checks on Developer ID content

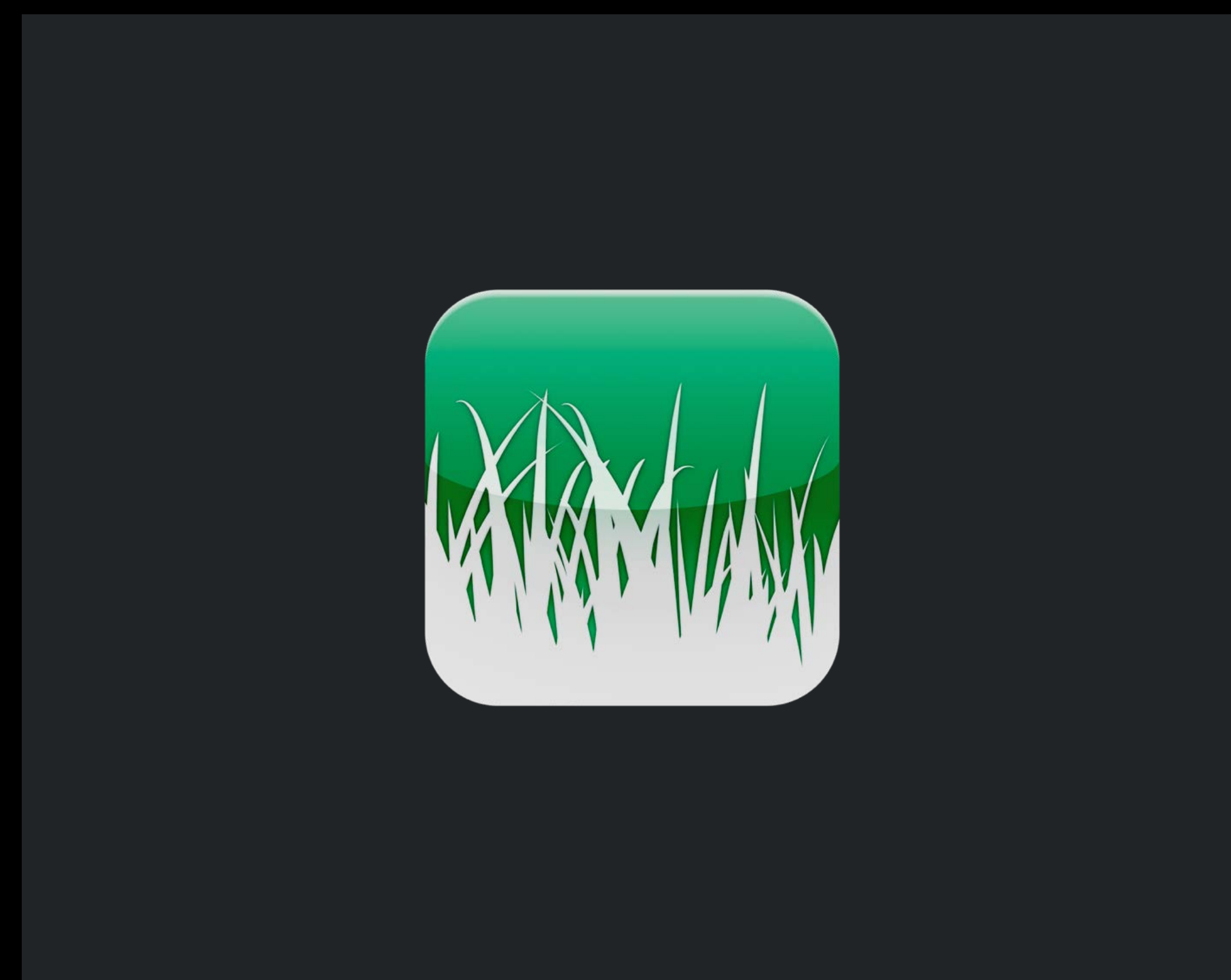
Optional extension to Developer ID program

Developers upload distribution-ready content

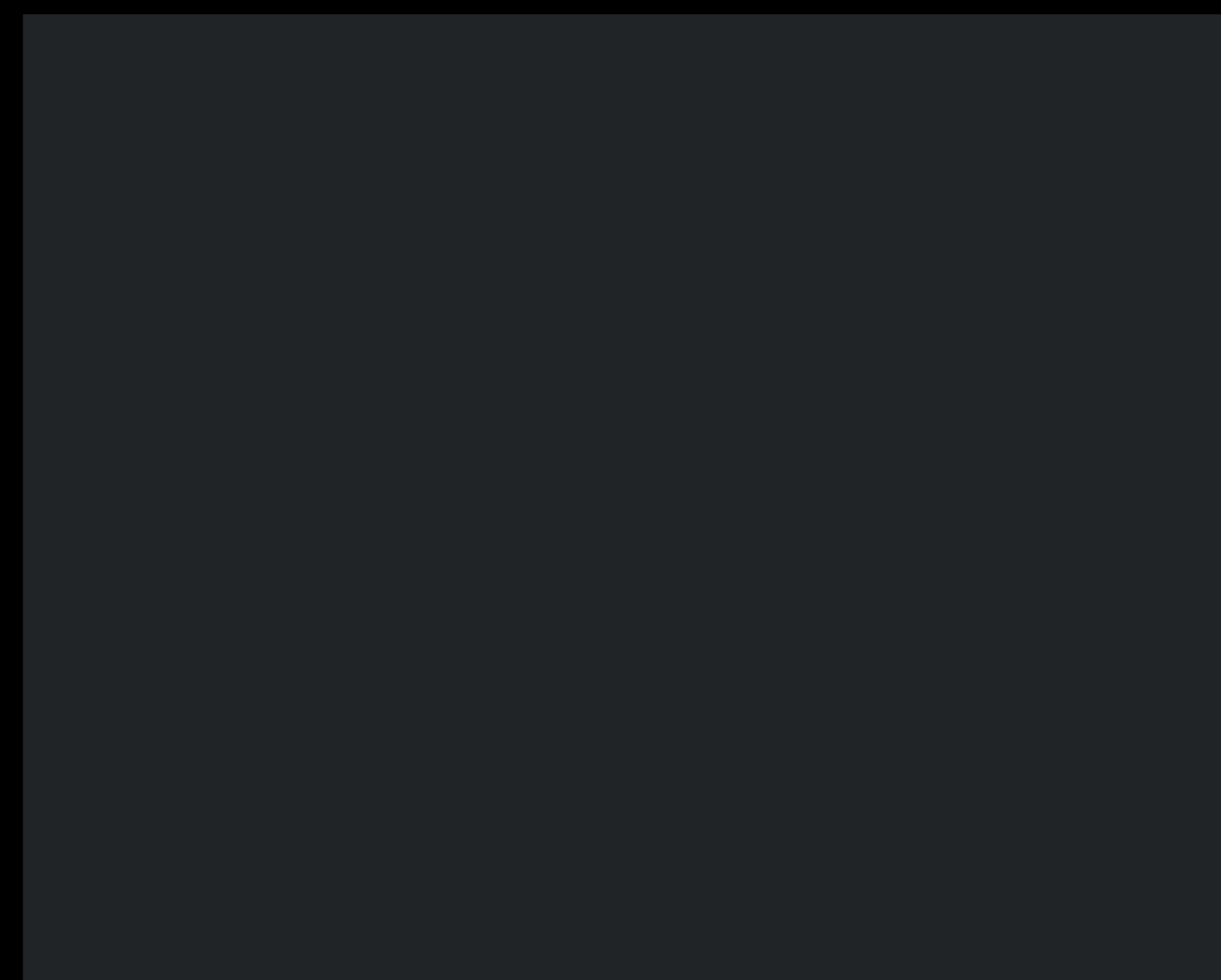
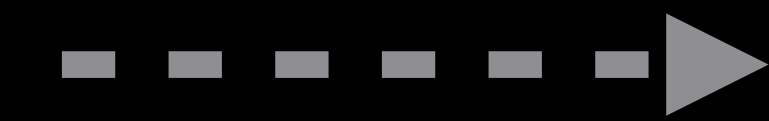
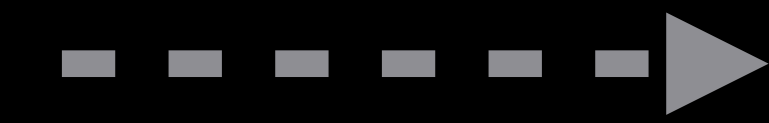
Development workflow prior to distribution is unchanged

**Not an App Review**

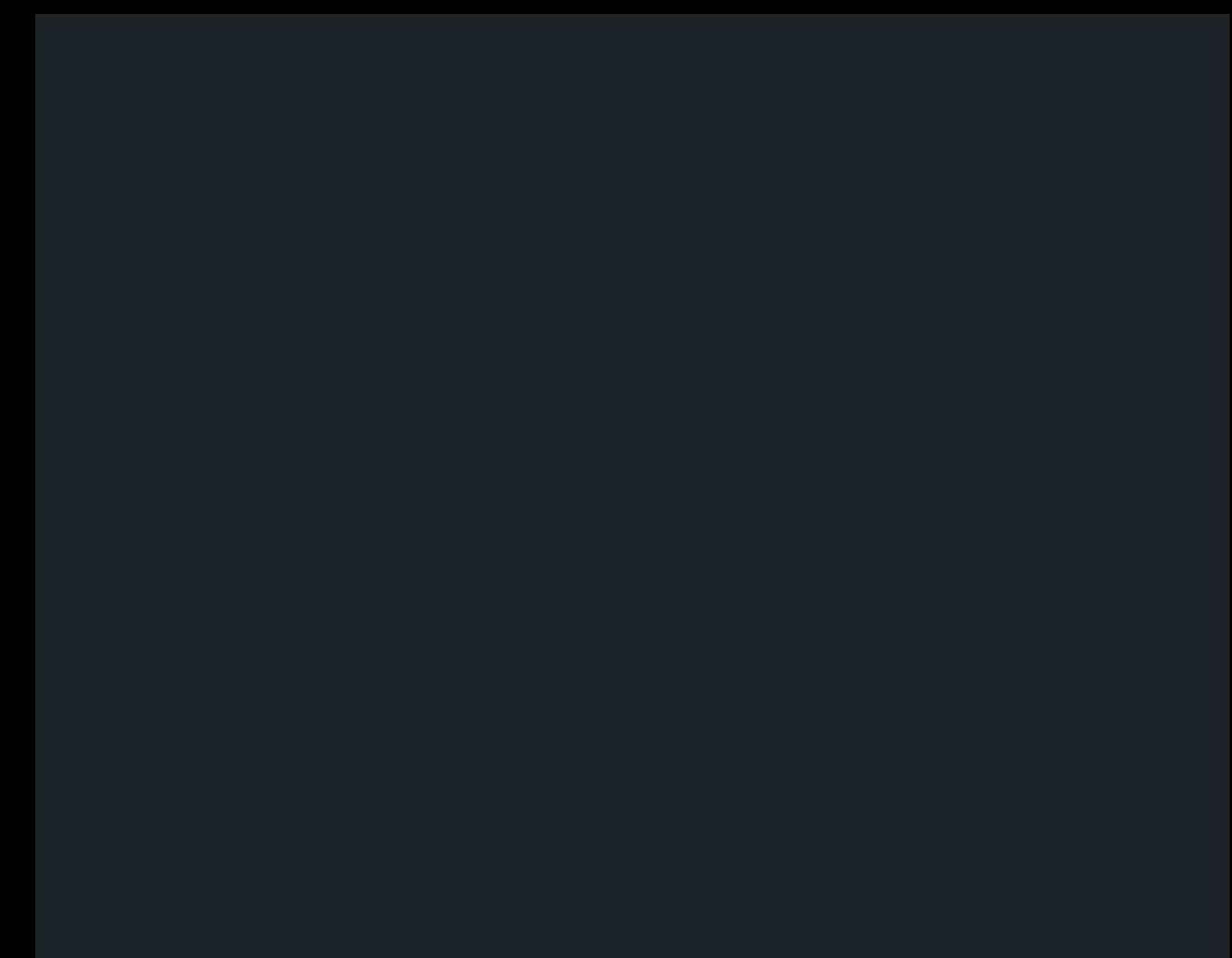
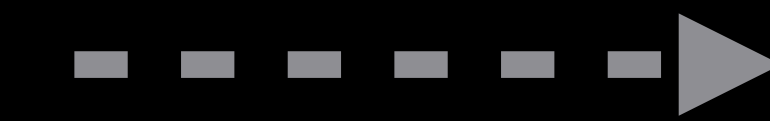
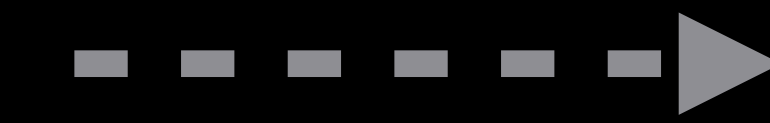
# Development Process



Local development



Distribution signing  
and testing

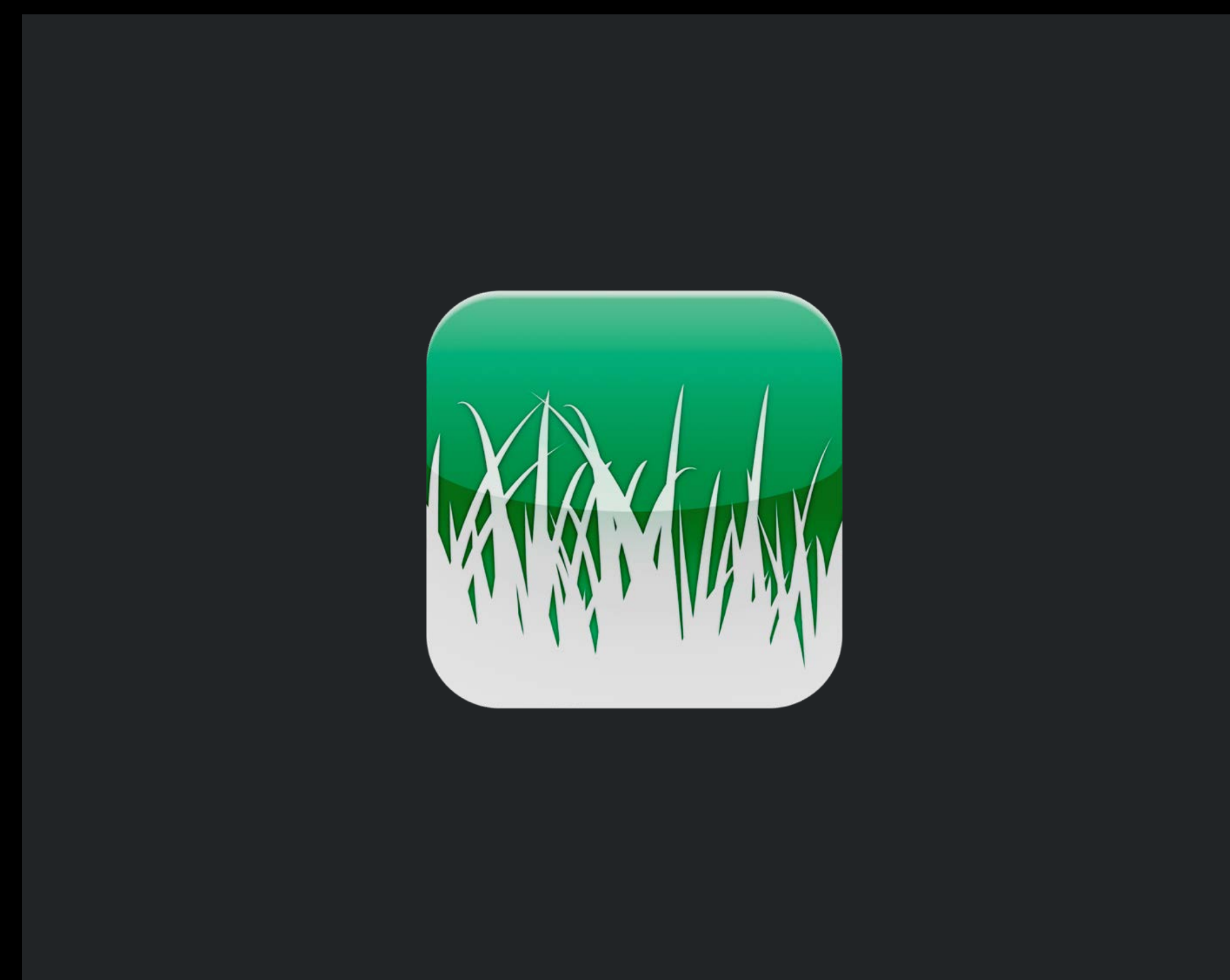


Distribute via  
website, etc

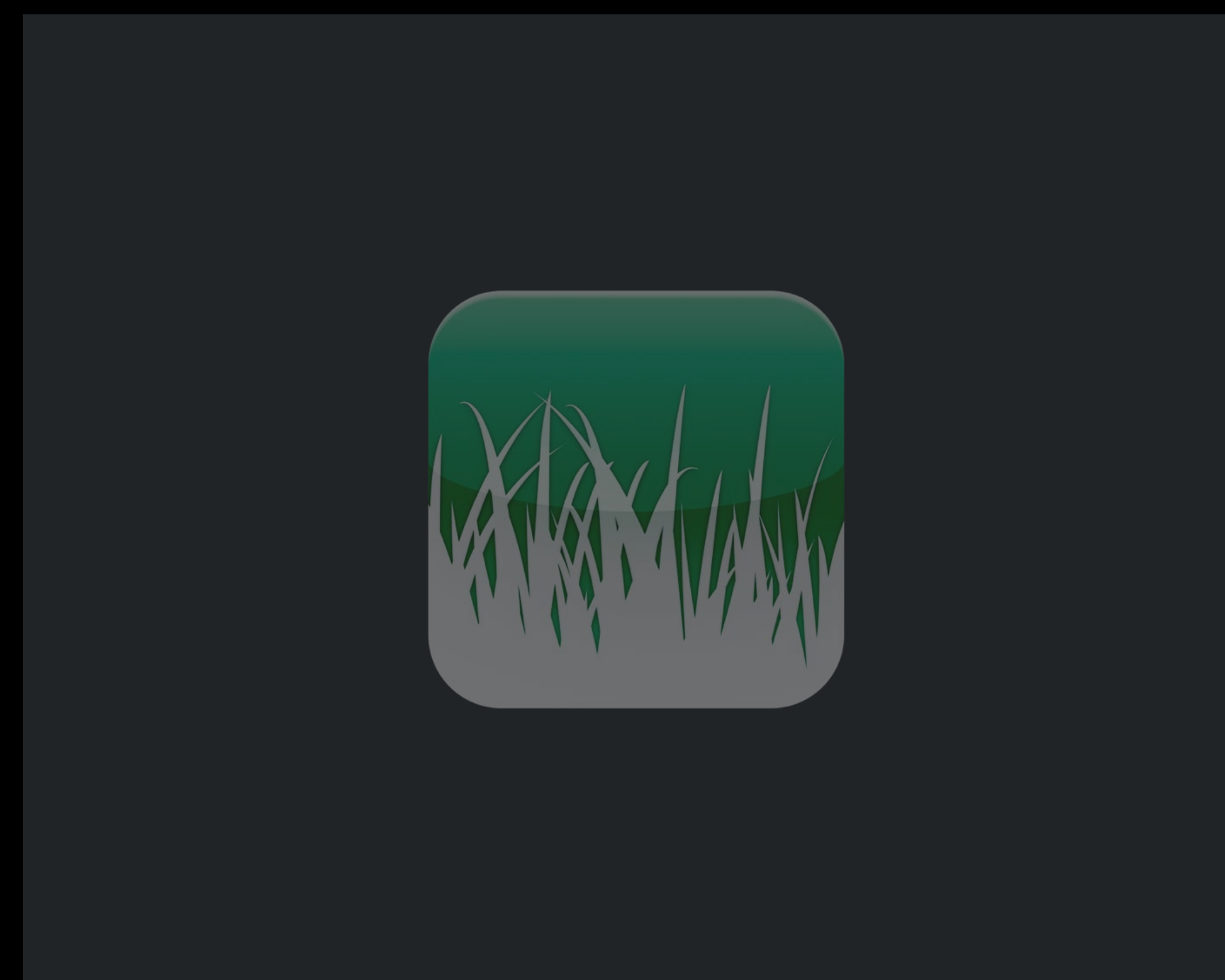
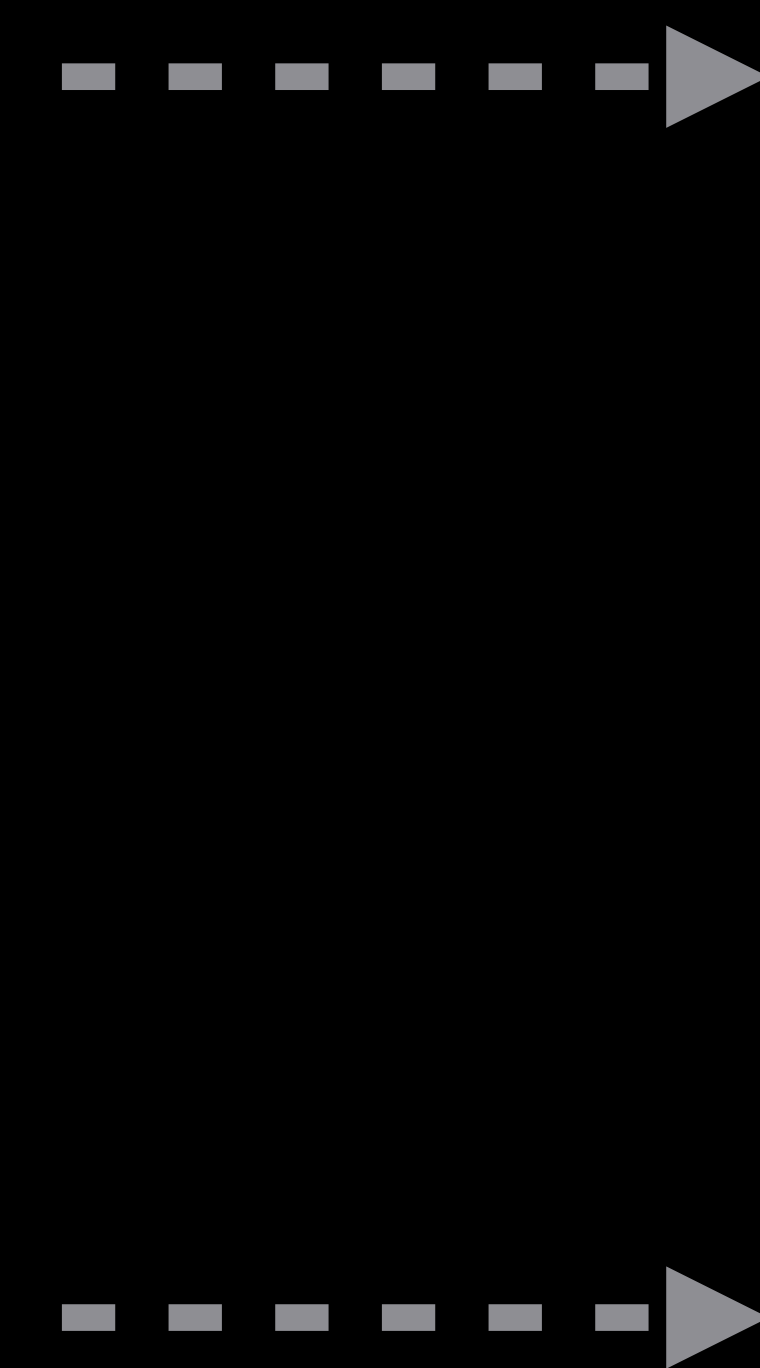
# Development Process

Apple Notary Service

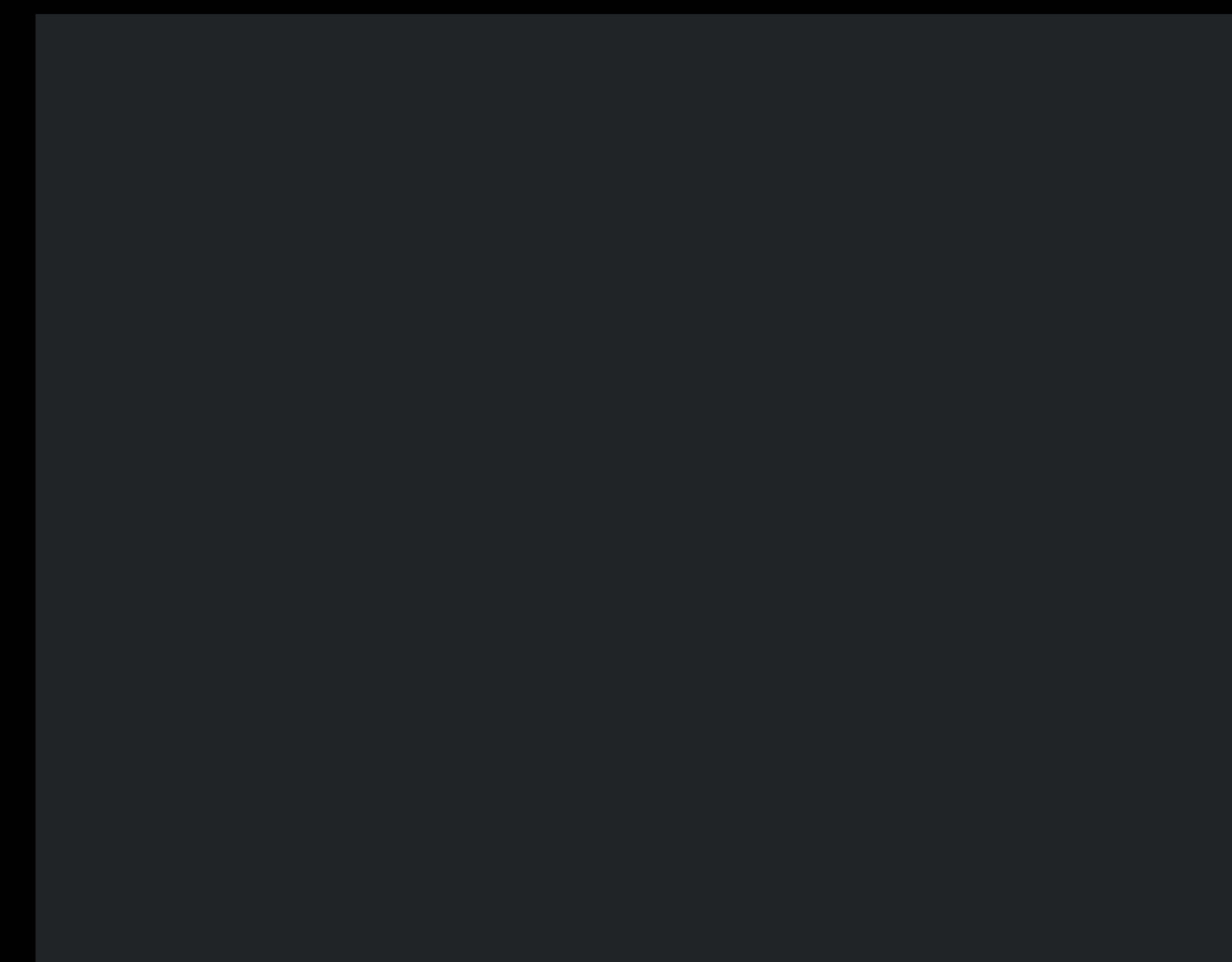
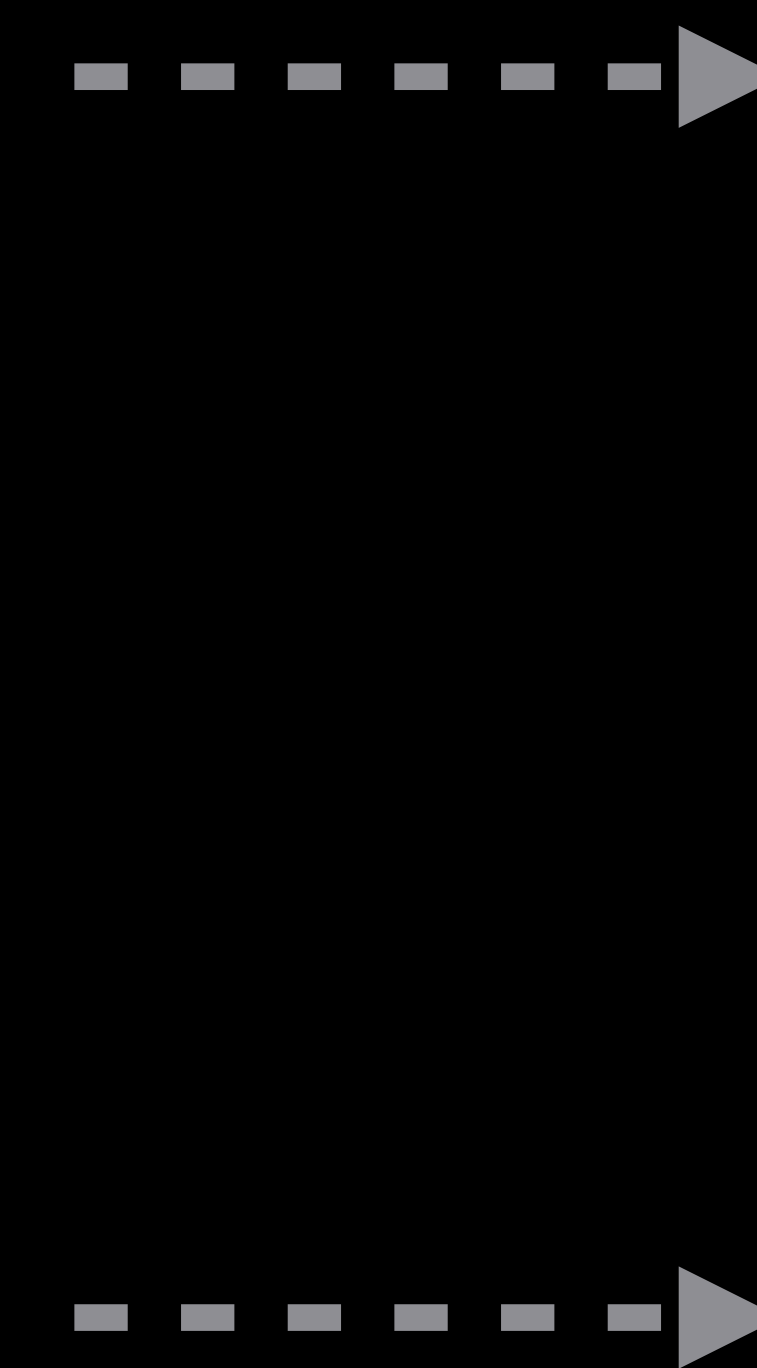
NEW



Local development



Distribution signing  
and testing



Distribute via  
website, etc

# Security Requirements

For new apps

No malicious software

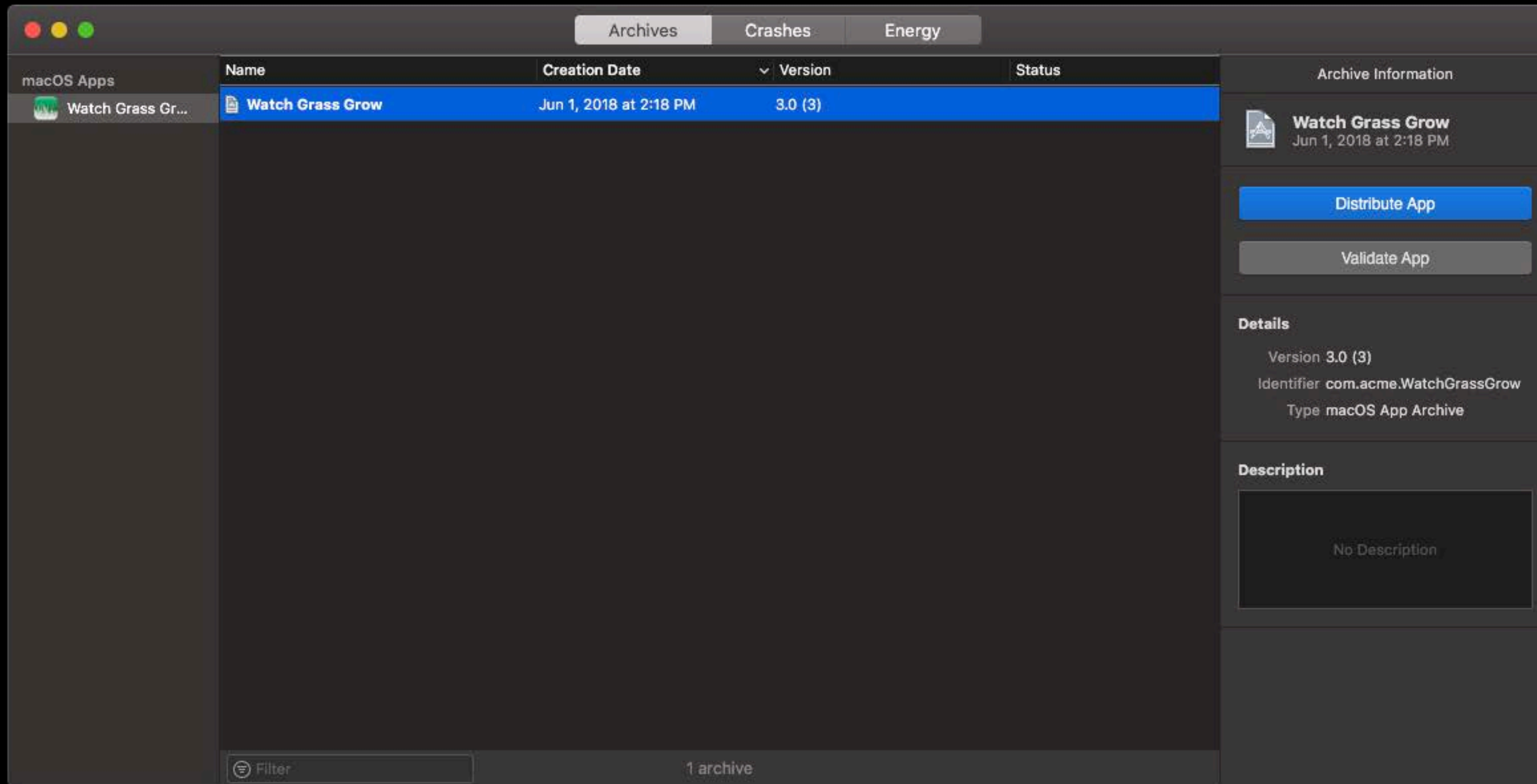
All executables properly signed

Opted into the enhanced runtime



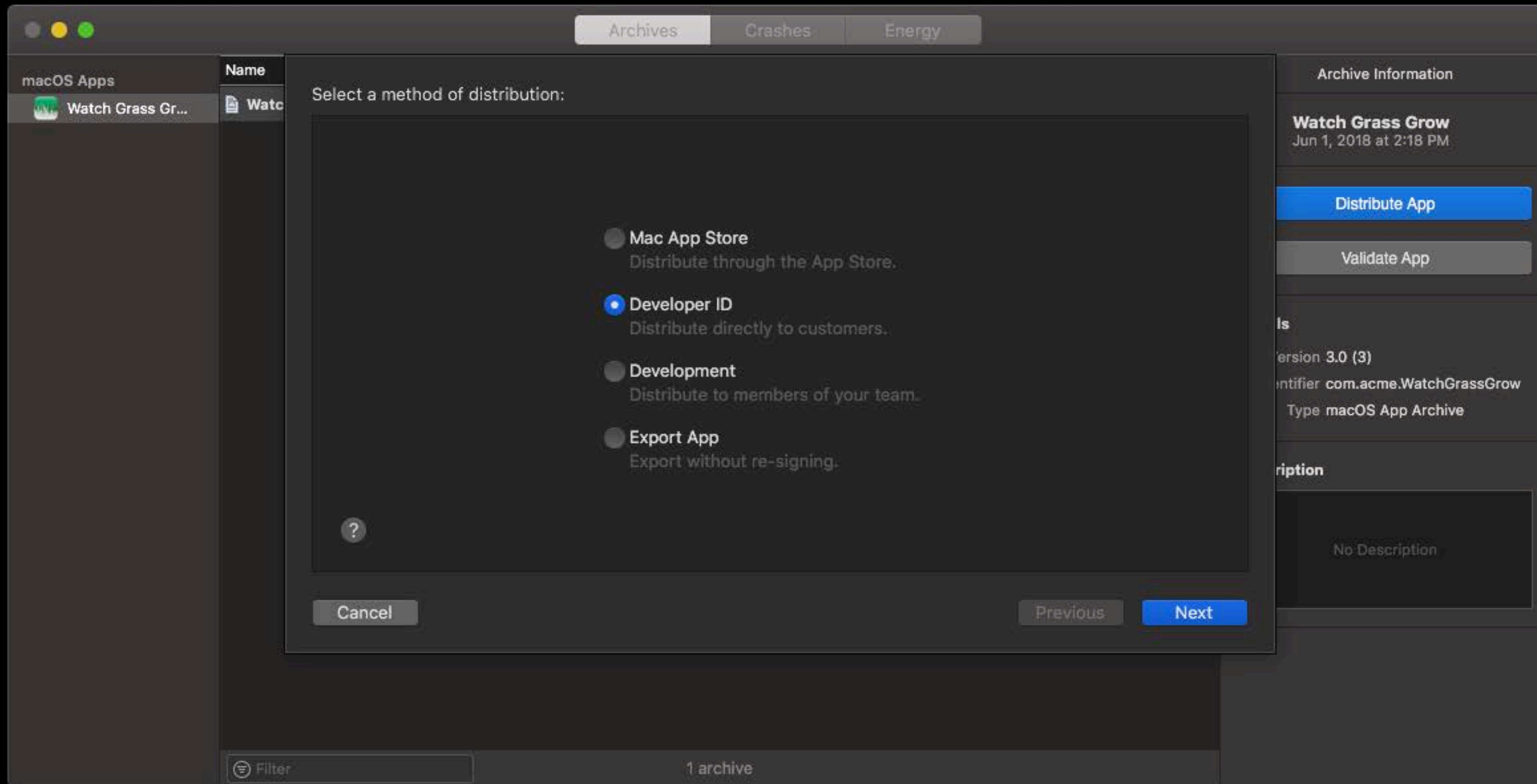
# Notarizing Applications

Xcode



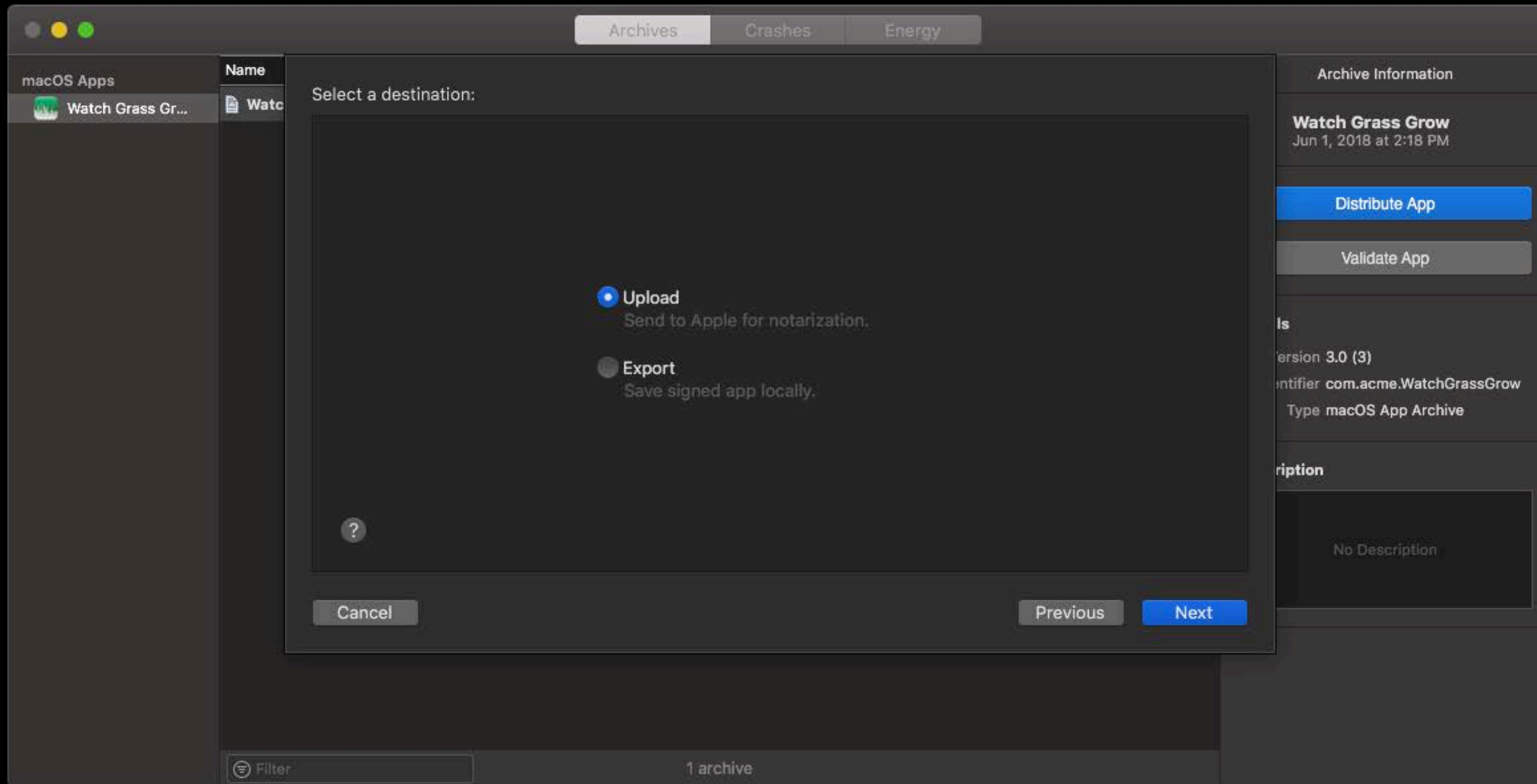
# Notarizing Applications

Xcode



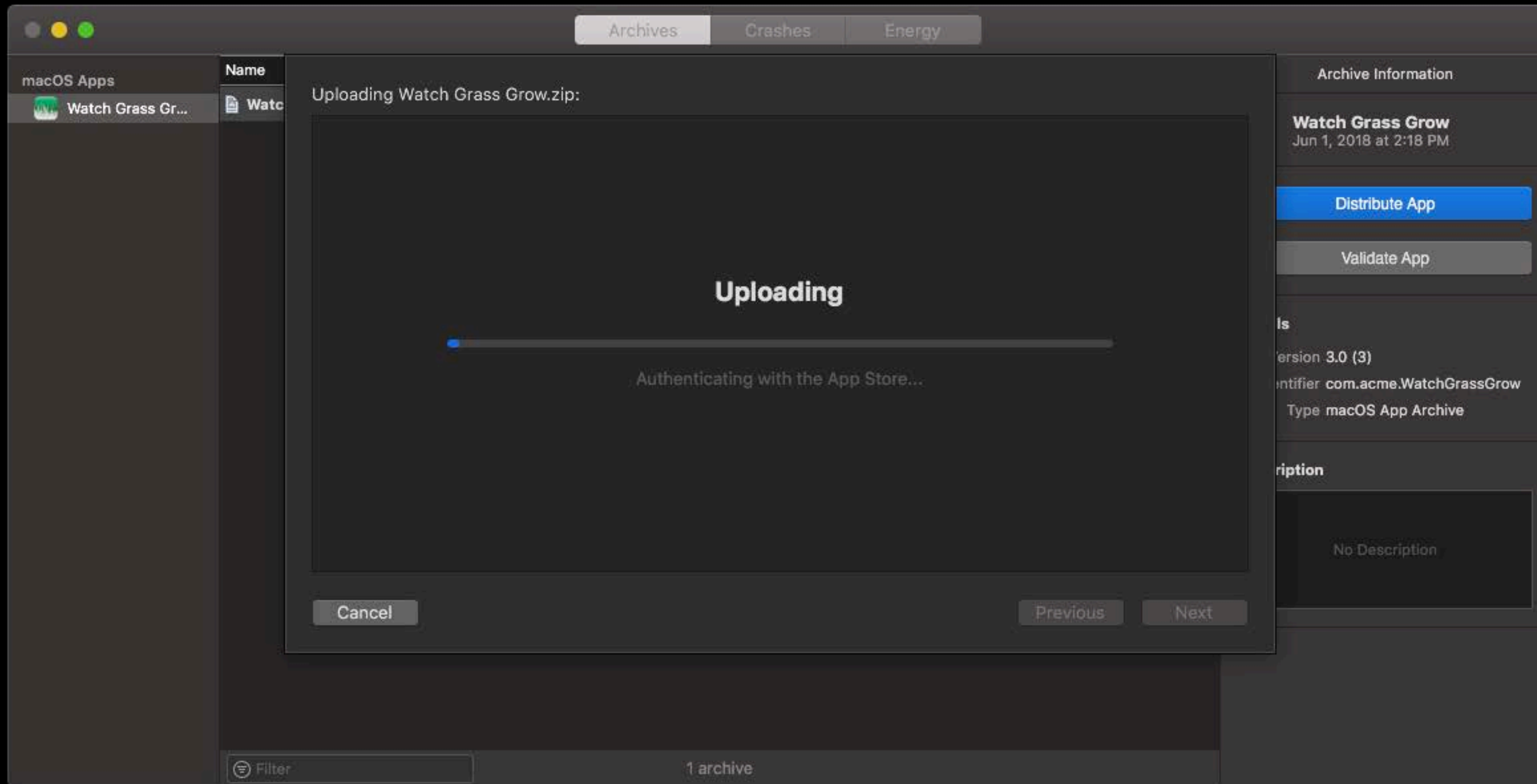
# Notarizing Applications

Xcode



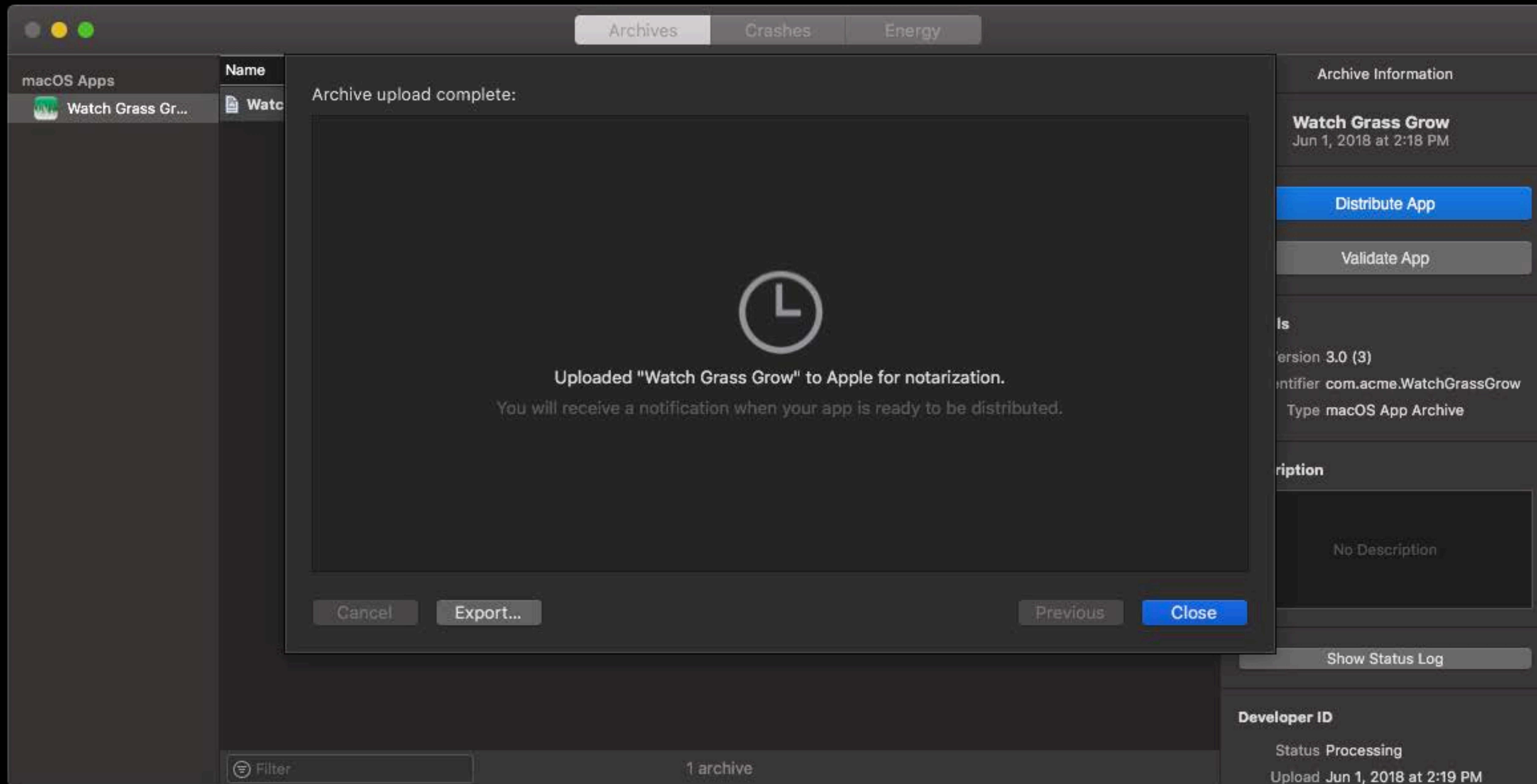
# Notarizing Applications

Xcode



# Notarizing Applications

Xcode



# Notarizing Applications

Xcode

The screenshot shows the Xcode interface with the 'Archives' tab selected. A table lists the archives, and a right-hand pane provides details for the selected archive.

Name	Creation Date	Version	Status
Watch Grass Grow	Jun 1, 2018 at 2:18 PM	3.0 (3)	Processing

**Archive Information**  
Watch Grass Grow  
Jun 1, 2018 at 2:18 PM

**Actions:**  
Distribute App  
Validate App

**Details**  
Version 3.0 (3)  
Identifier com.acme.WatchGrassGrow  
Type macOS App Archive

**Description**  
No Description

**Developer ID**  
Status Processing  
Upload Jun 1, 2018 at 2:19 PM

1 archive

# Notarizing Applications

Xcode

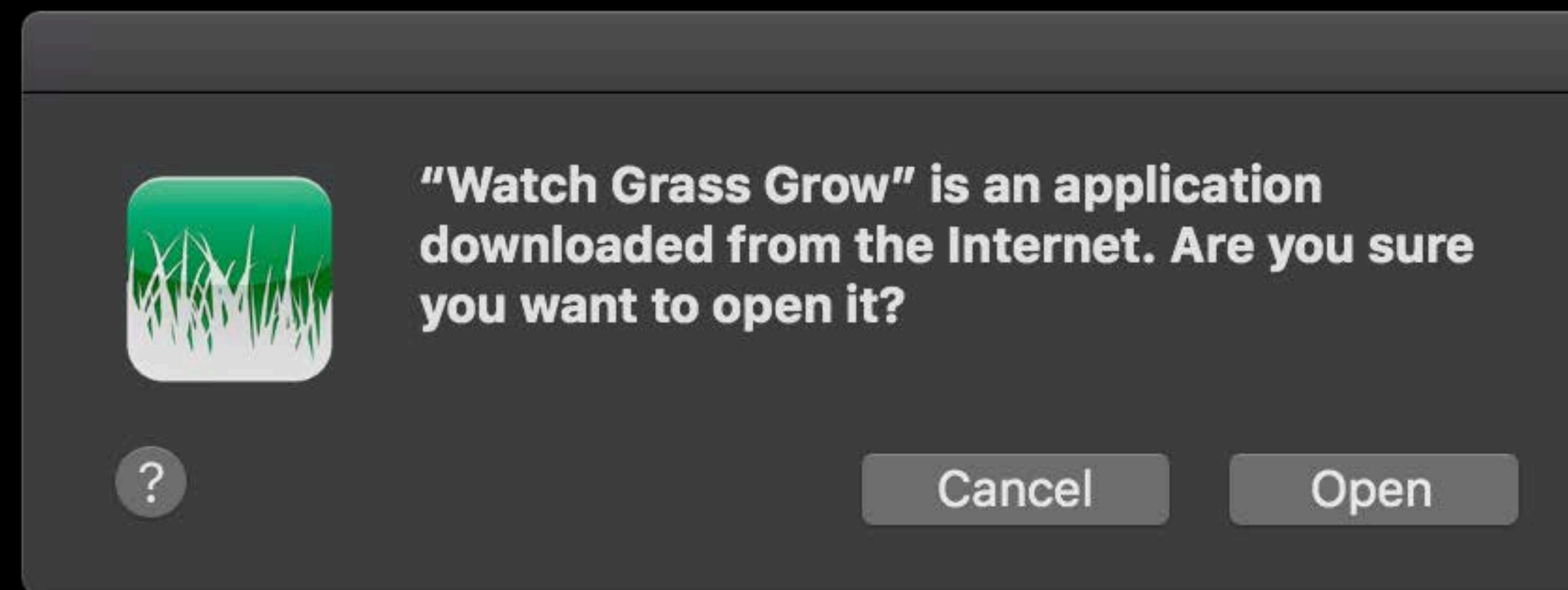


## Mac App Successfully Notarized

com.acme.WatchGrassGrow was notarized and can now be exported from the Organizer.

# Notarizing Applications

Xcode





# Notarizing Applications

## Upload

Notary service accepts zip files, packages, and disk images

```
# From Terminal
$> xcrun altool --eval-app "Watch Grass Grow.dmg" --primary-bundle-id com.acme.WatchGrassGrow
    -u developer@acme.com -p @keychain:AppleIDAccount
...
RequestUUID = 892d6d9e-2f64-495e-b027-8e7bd73fa674.
$>
```

# Notarizing Applications

## Status

Status output includes a log URL with helpful details

```
# From Terminal
$> xcrun altool --eval-info --uuid <UUID> -u developer@acme.com -p @keychain:AppleIDAccount
RequestUUID: 892d6d9e-2f64-495e-b027-8e7bd73fa674
      Status: 0
      StatusStr: success
      LogFileURL: <very long link>
$>
```

# Notarizing Applications

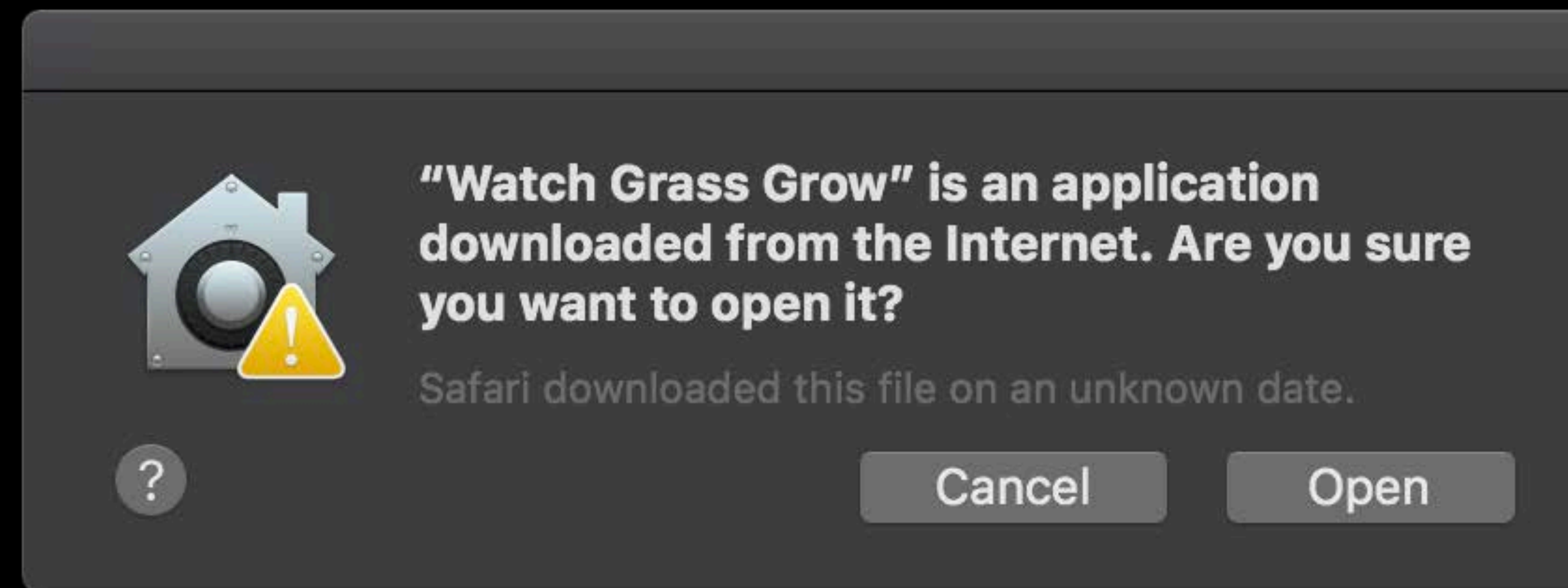
## Stapling

Staple tickets directly to applications, disk images, and installer packages

Stapler combines ticket retrieval and attaching in one step

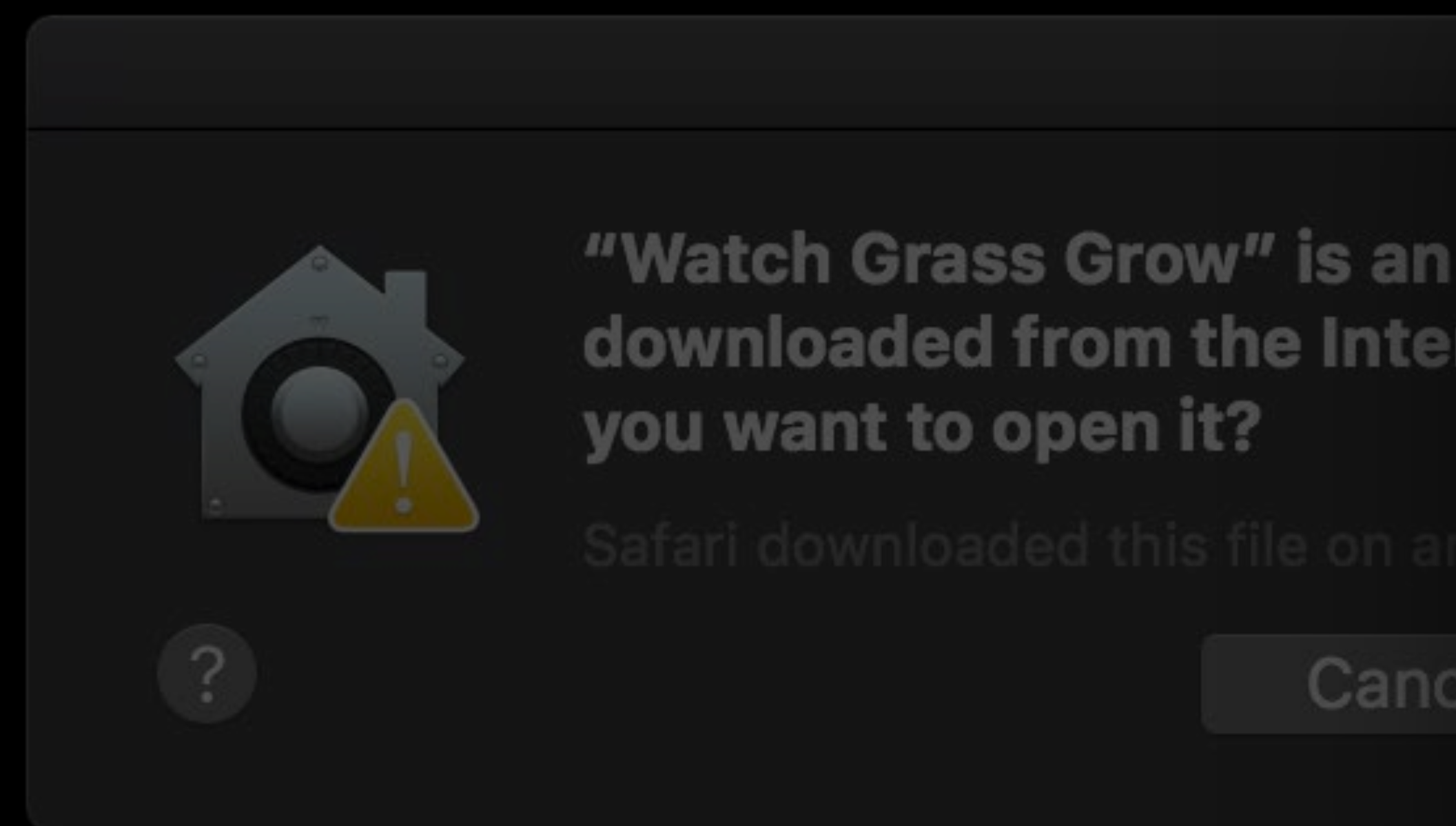
```
# From Terminal
$> xcrun stapler staple "Watch Grass Grow.dmg"
...
The staple and validate action worked!
$>
```

# First Launch

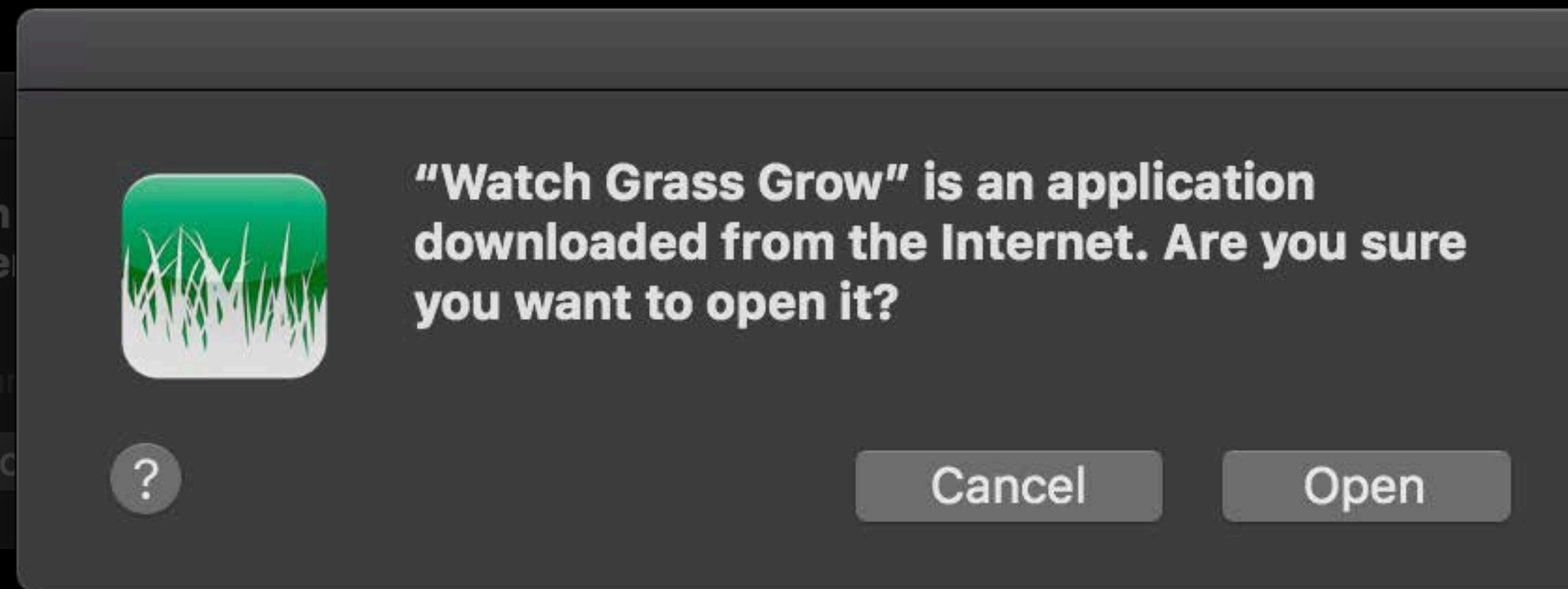


Developer ID App

# First Launch

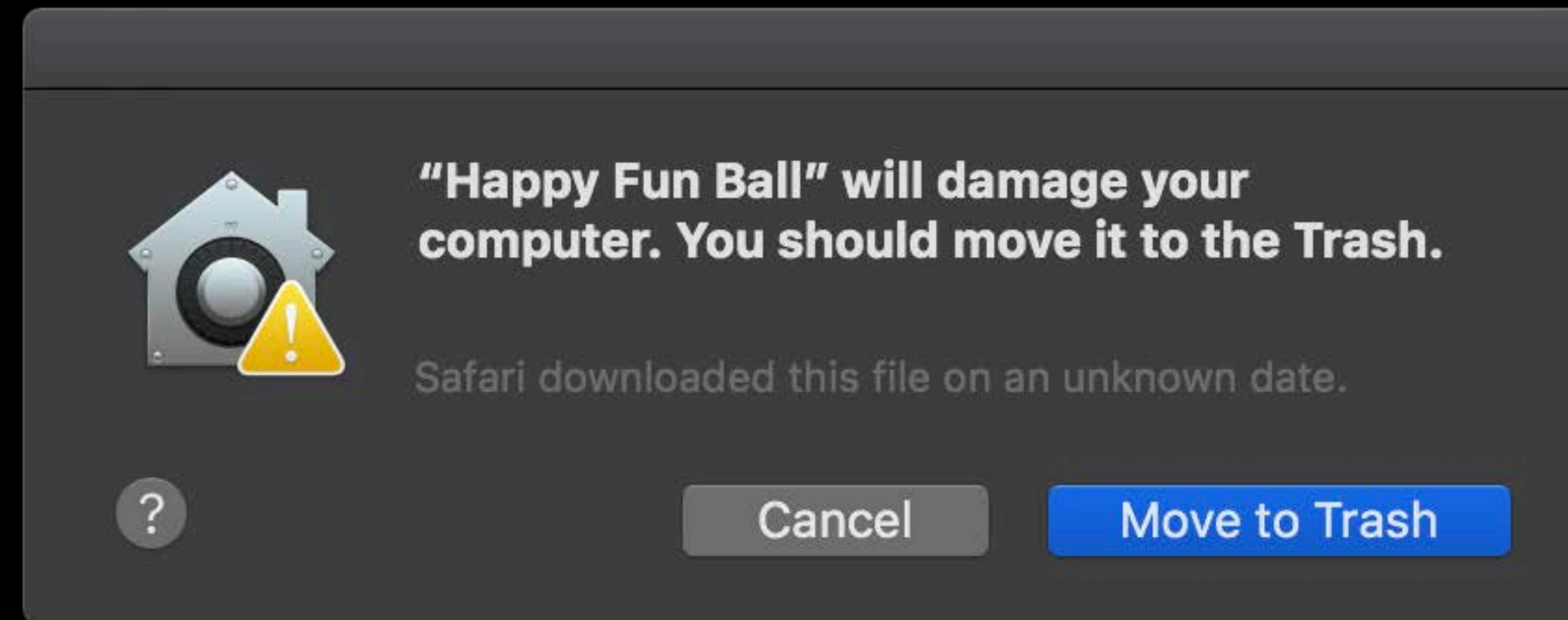


Developer ID App



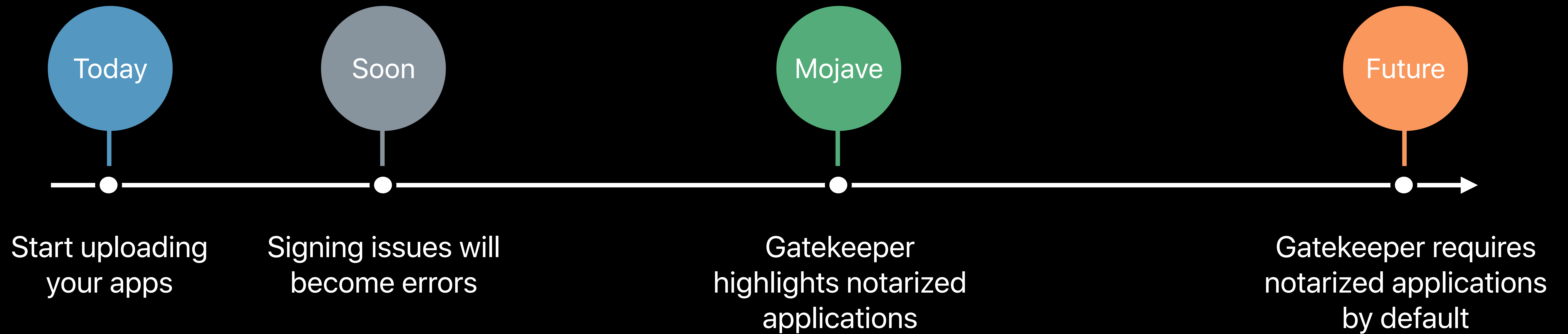
Notarized Developer ID App

# First Launch



Malicious App

# Notarization Timeline







# Summary

Test your app against the new user data protections

Be transparent to the user about the data you need to access

Adopt the new hardened runtime for additional protection

Start submitting your apps to the notary service today!

# More Information

<https://developer.apple.com/wwdc18/702>

---

Security Lab

Technology Lab 1

Tuesday 3:00PM

---

Security Lab

Technology Lab 2

Wednesday 9:00AM

---

Signing and Distribution Lab

Technology Lab 10

Thursday 9:00AM

---

 **WWDC18**