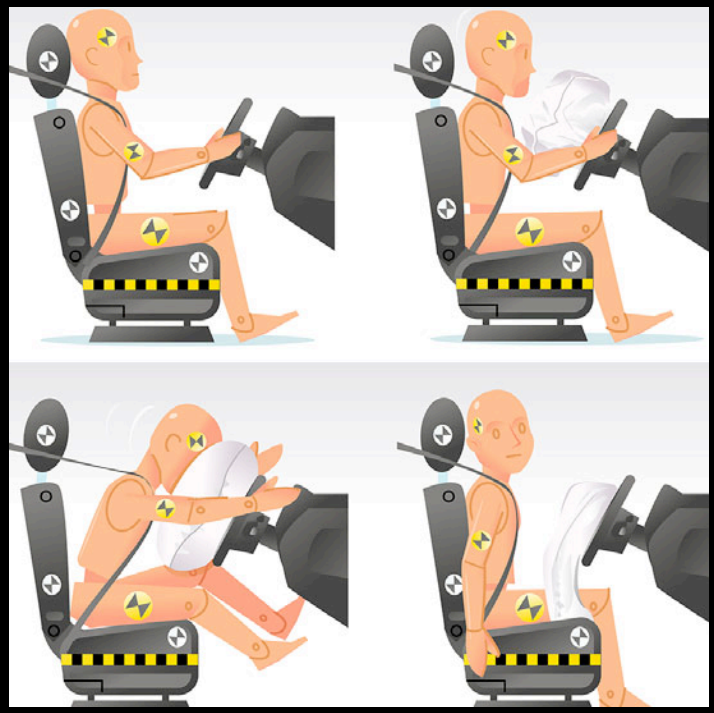
The OS X App Sandbox

Session 700/713

Ivan Krstić
Core OS Disaster Monkey

Modern Car Safety

- Mandatory standardized crash testing performed by the government
- Traction control, blind spot warnings, lane drift alerts
- But: Damage containment
- When all else fails, there are seatbelts and airbags



Traditional Desktop Security

- Defender must protect everything at all times, attacker must breach one protection at any time
- Emphasis on damage prevention (ASLR, NX, antivirus), not containment
- One thing goes wrong, game over
- No seatbelt and airbag for the computer

The Unfortunate Assumption

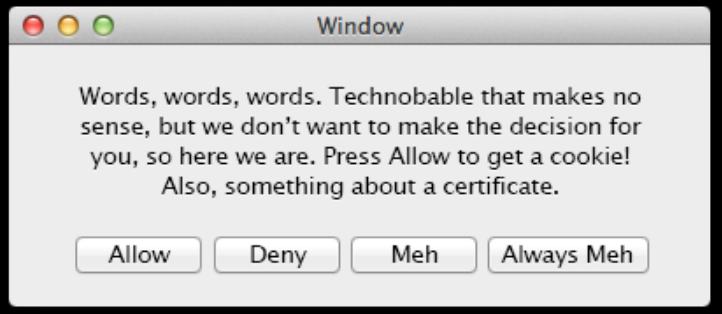
- All programs should execute with the full privileges of the executing user
  - Or: Security is a barrier between different users, not different programs
- But most modern computer devices are single-user systems
- Not every app should have access to the most sensitive data
  - Apps should only have access to the resources they need

An Unfortunate Example

- The unfortunate assumption does not work
- Compromising any app must not grant access to all user data

Security UI Does Not Work

- Security dialogs are mysterious and opaque; riddles wrapped inside enigmas
- Clicking “Permit” or “Allow” maximizes the likelihood of getting work done
- “If you’re explaining, you’re losing”
- Pavlovian conditioning to ignore security



Landscape Changes

- Many apps, many developers
- Computers are always on a network
- Easier than ever to find and run new software
- Security challenge: Isolate data between programs

Software Reality

- Complex systems will always have vulnerabilities
  - Complexity is never decreasing
- Single buffer overflow can ruin your user's day
- Frameworks and libraries you don't control
  - Every `WebView` instance: Millions of lines of code and a full-featured JavaScript engine
- No limit on exploit damage

App Sandbox

App Sandbox

- Introduced in OS X Lion
- More secure applications
- Drive security policy by user intent
- Contain exploit damage
- Reduce ability for a compromised or misbehaving application to steal, corrupt, or destroy user data

Key Concepts

- Developer expresses what an app is supposed to be able to do
- Each app runs in its own container
- User controls access to documents
  - Special cases (e.g., recent items, drag and drop) work automatically

Key Components

1. Entitlements
2. Containers
3. Powerbox
4. XPC Services

Entitlements

- What apps can do is determined by the developer-specified entitlements in the code signature
- Just a property list, editable in Xcode
- Simple, easy to understand
- About 20 total entitlements in Mountain Lion

Entitlements

- User-selected files, Downloads folder, secure bookmarks
- Personal information
  - Address book, calendars, location
- Assets: Music, movies, pictures
- Network client, server
- Devices
  - Camera, microphone, printing, USB, FireWire, Bluetooth, serial
- Application groups and scripting/automation targets

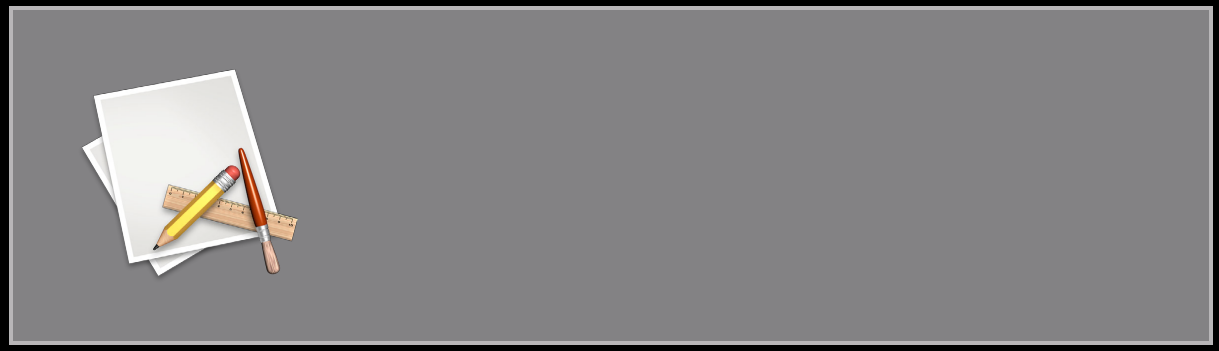
Key Components

1. Entitlements
2. Containers
3. Powerbox
4. XPC Services

```
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/
```

HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/

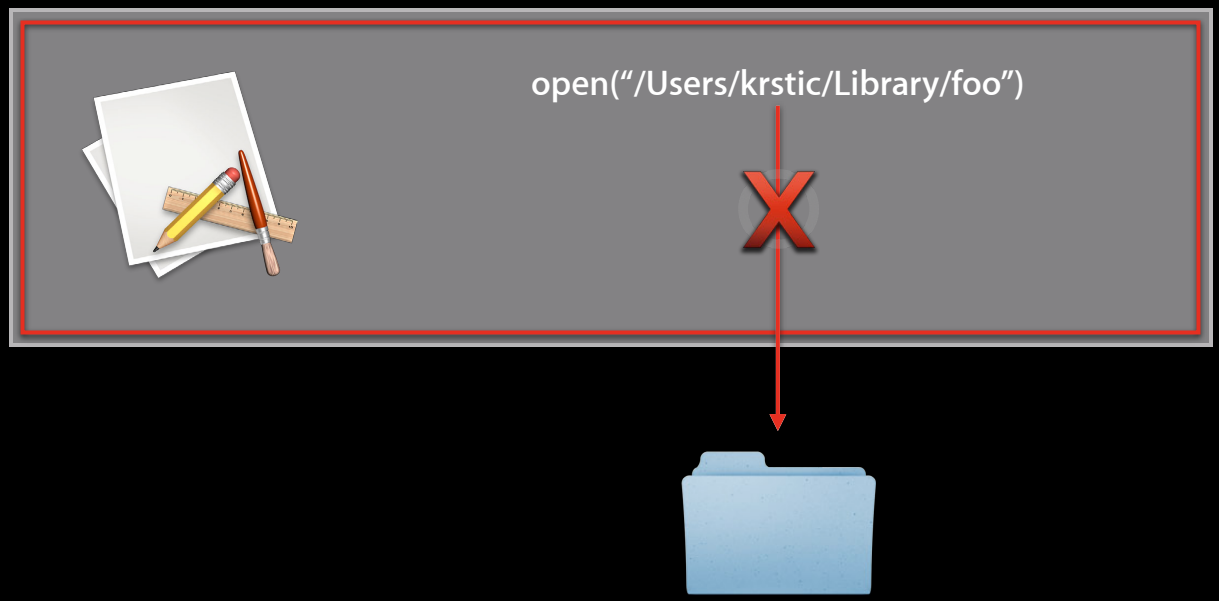
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/



```
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/
```

open(“/Users/krstic/Library/foo”)

```
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/
```



HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/

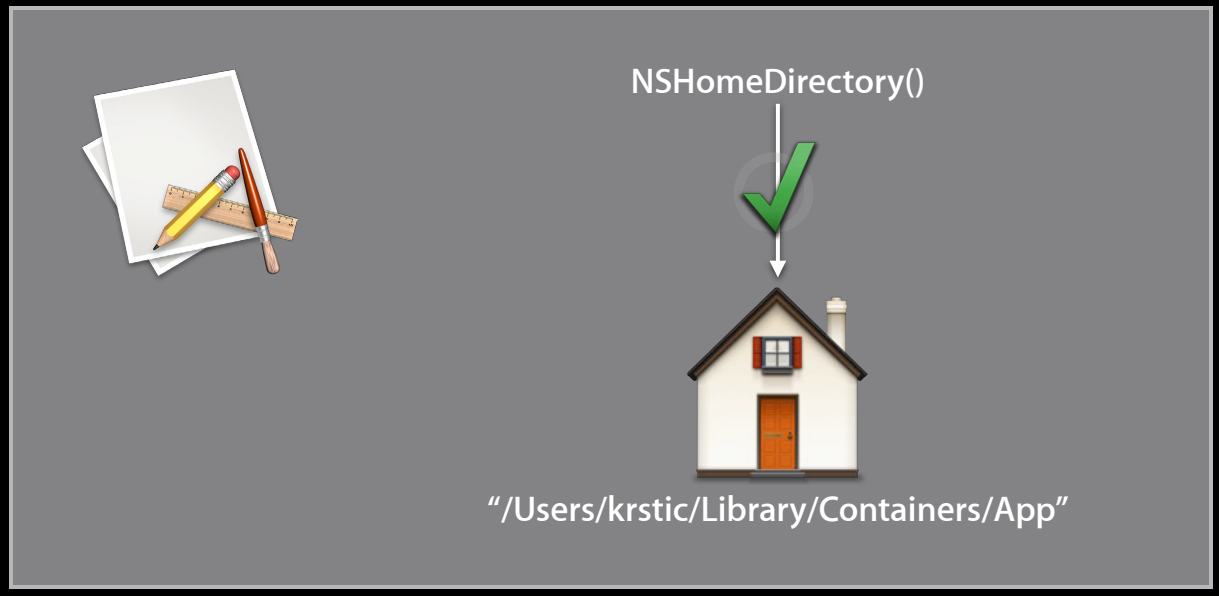
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/

HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/

NSHomeDirectory()

HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/

```
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/
```

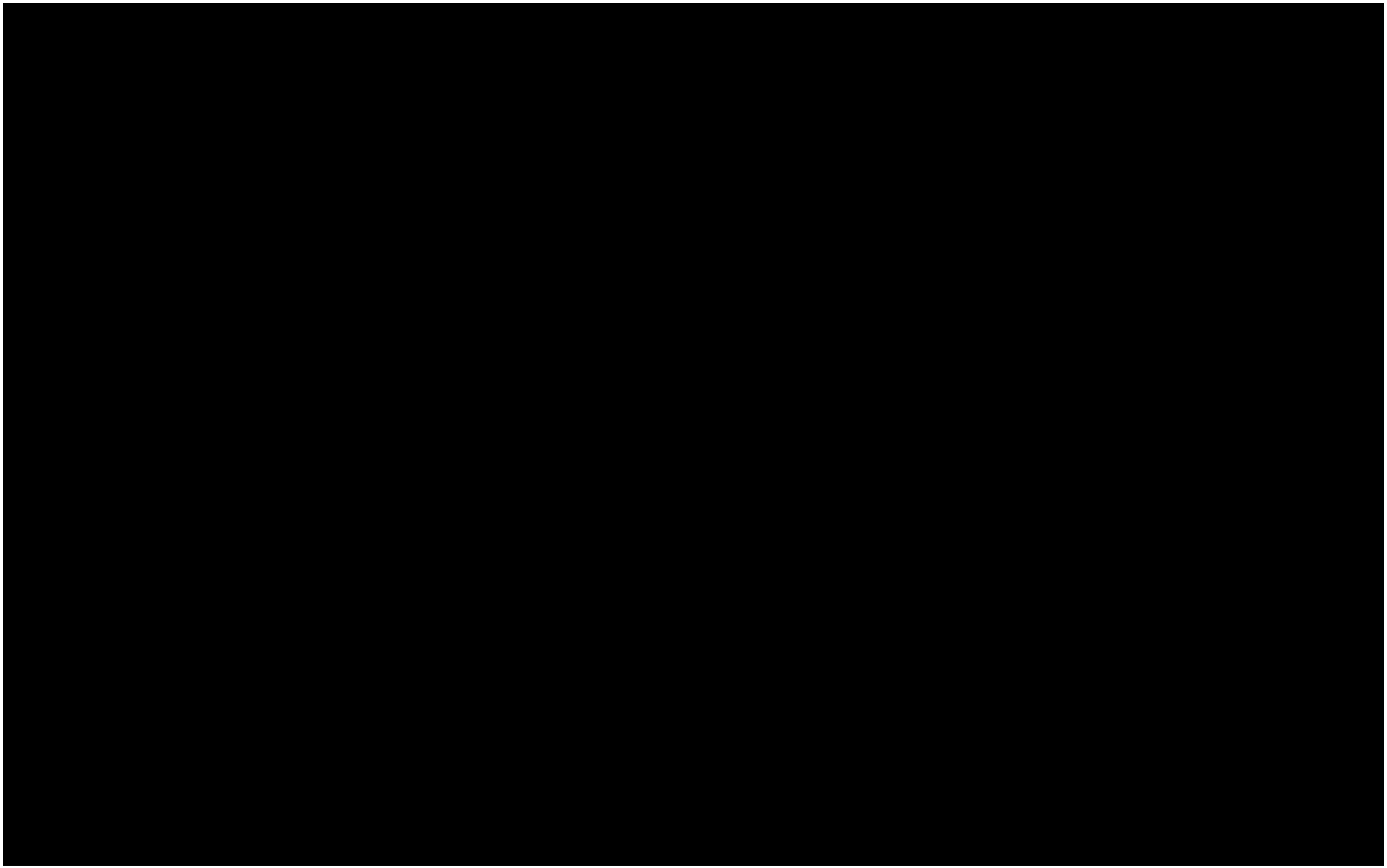


Key Components

1. Entitlements
2. Containers
3. Powerbox
4. XPC Services

Powerbox

- Cocoa NSOpenPanel/NSSavePanel
- Trusted mediator process
- Clear declaration of user intent
  - Drives security policy
  - Sandboxed apps cannot synthesize user input events

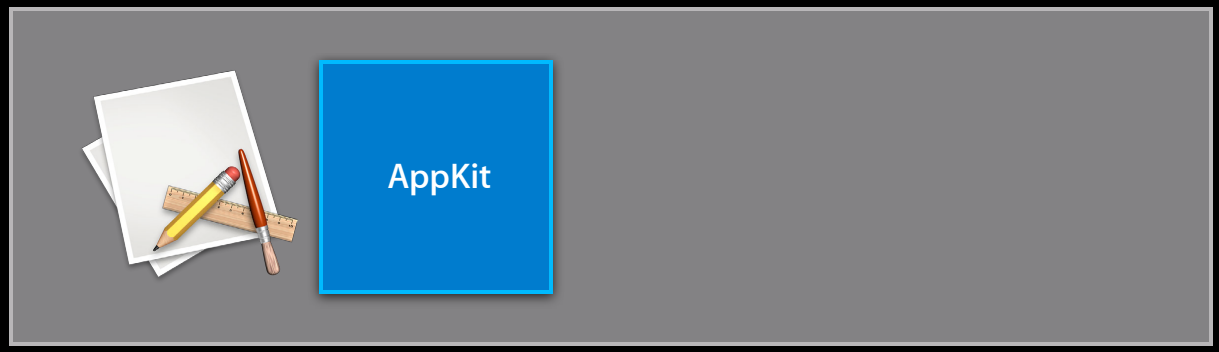


AppKit

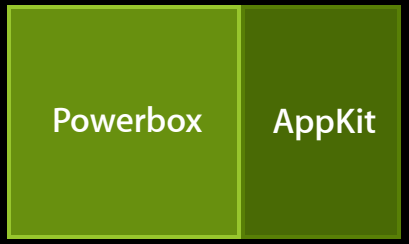
~/Documents

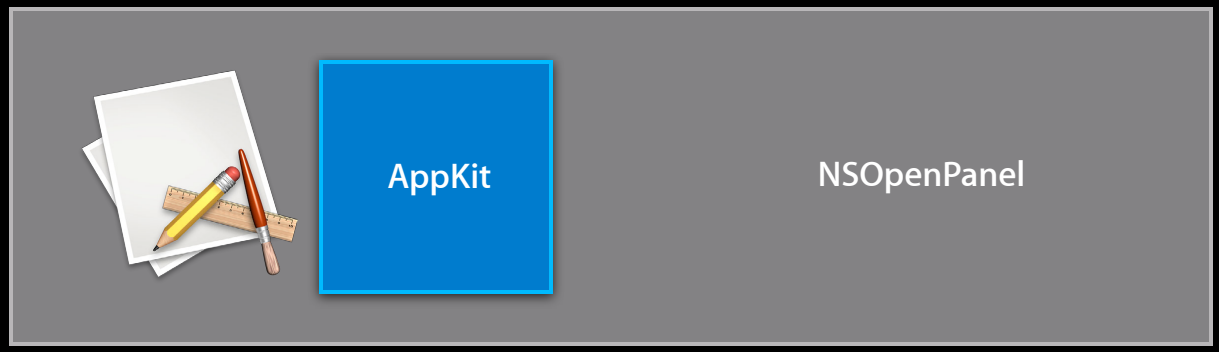
Powerbox

AppKit

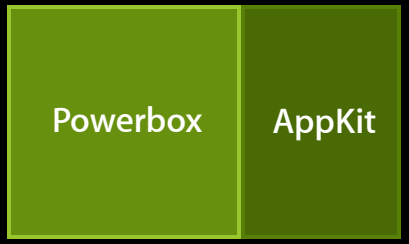


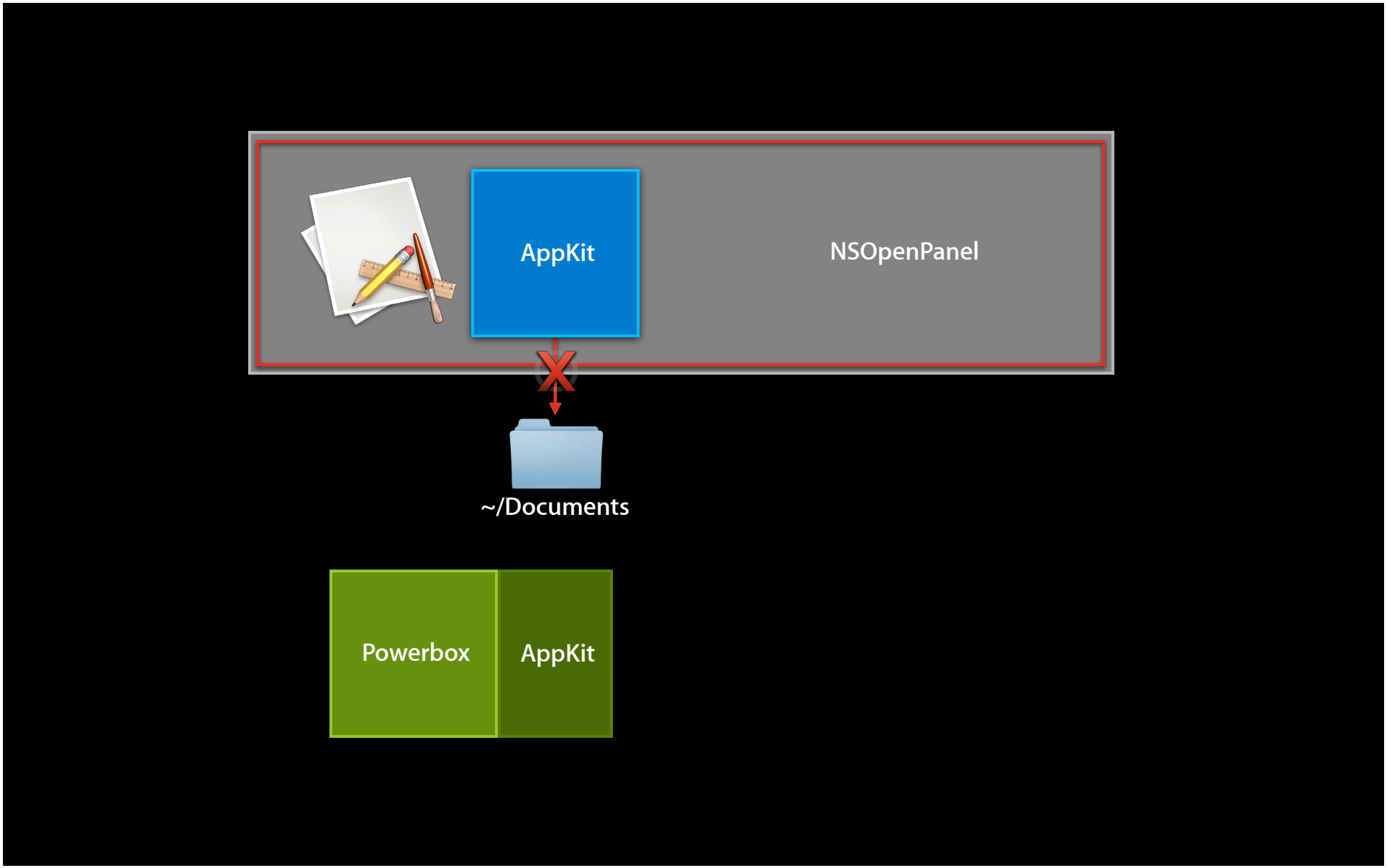
~/Documents

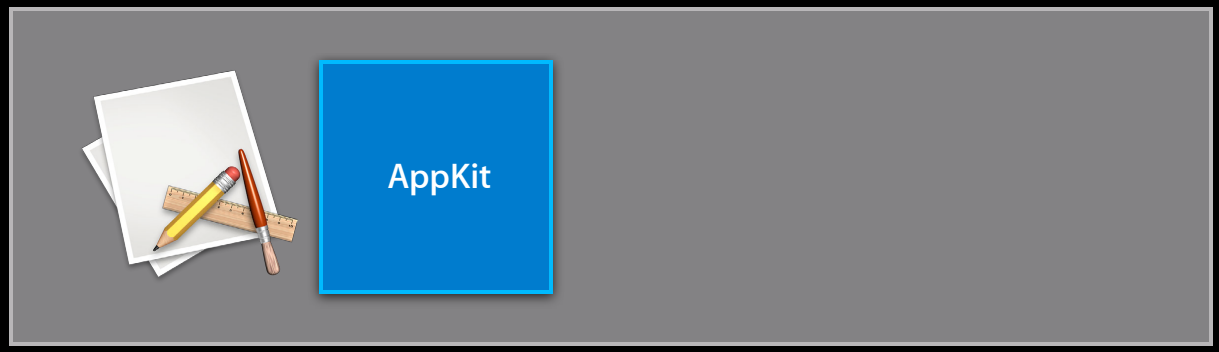




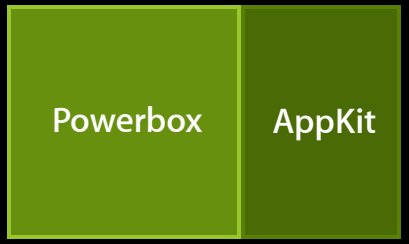
~/Documents

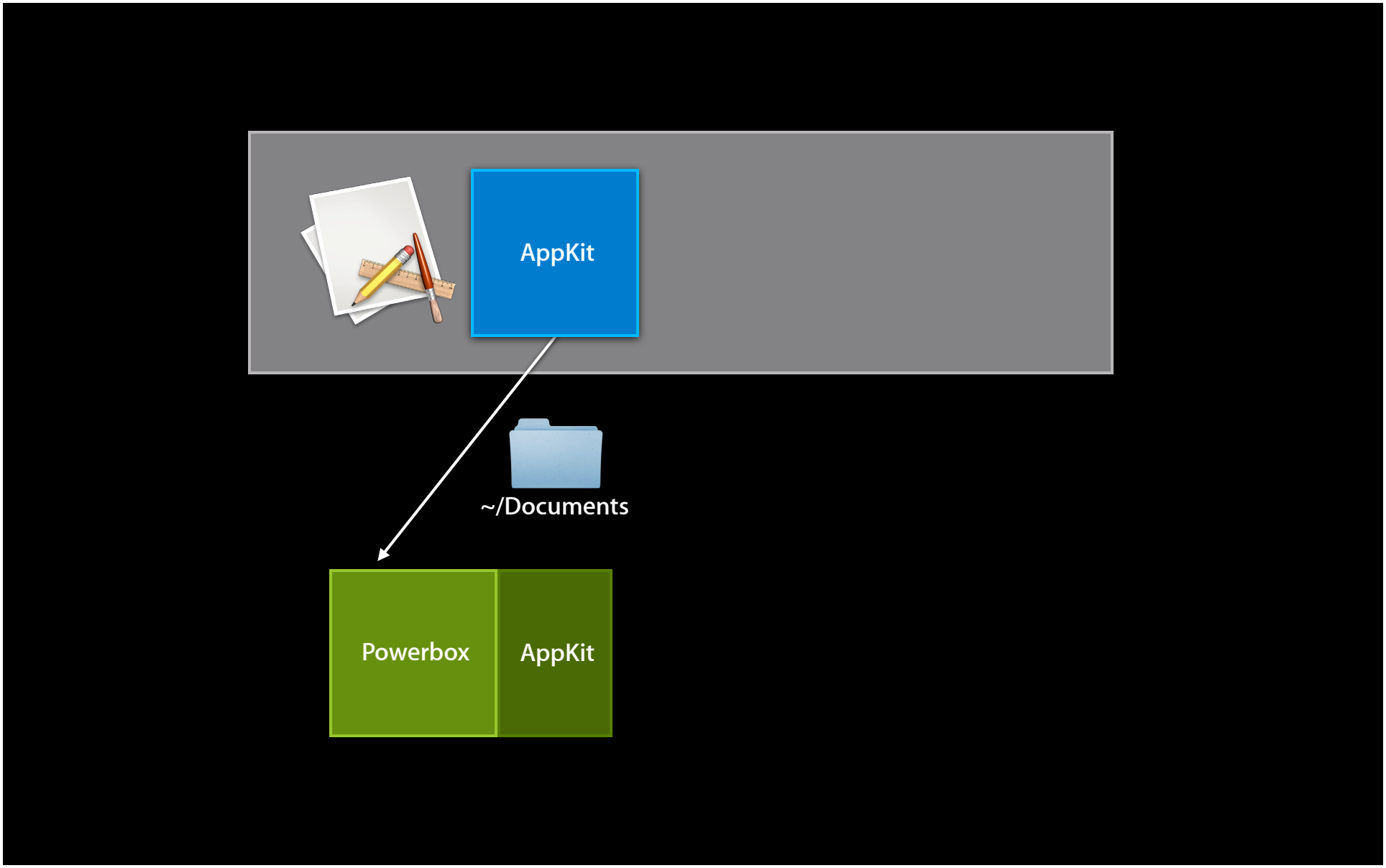


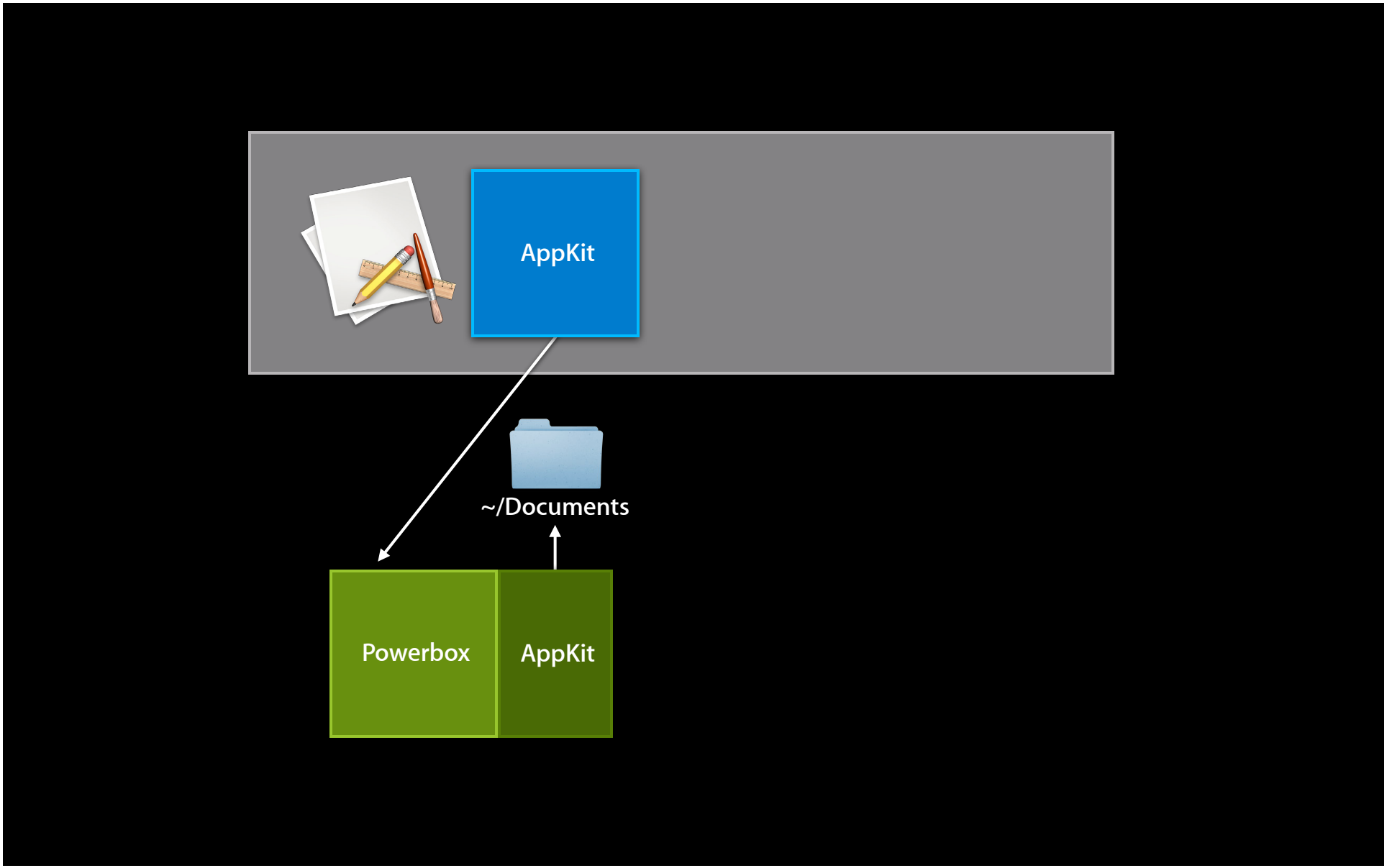

AppKit
NSOpenPanel
~/Documents
Powerbox
AppKit

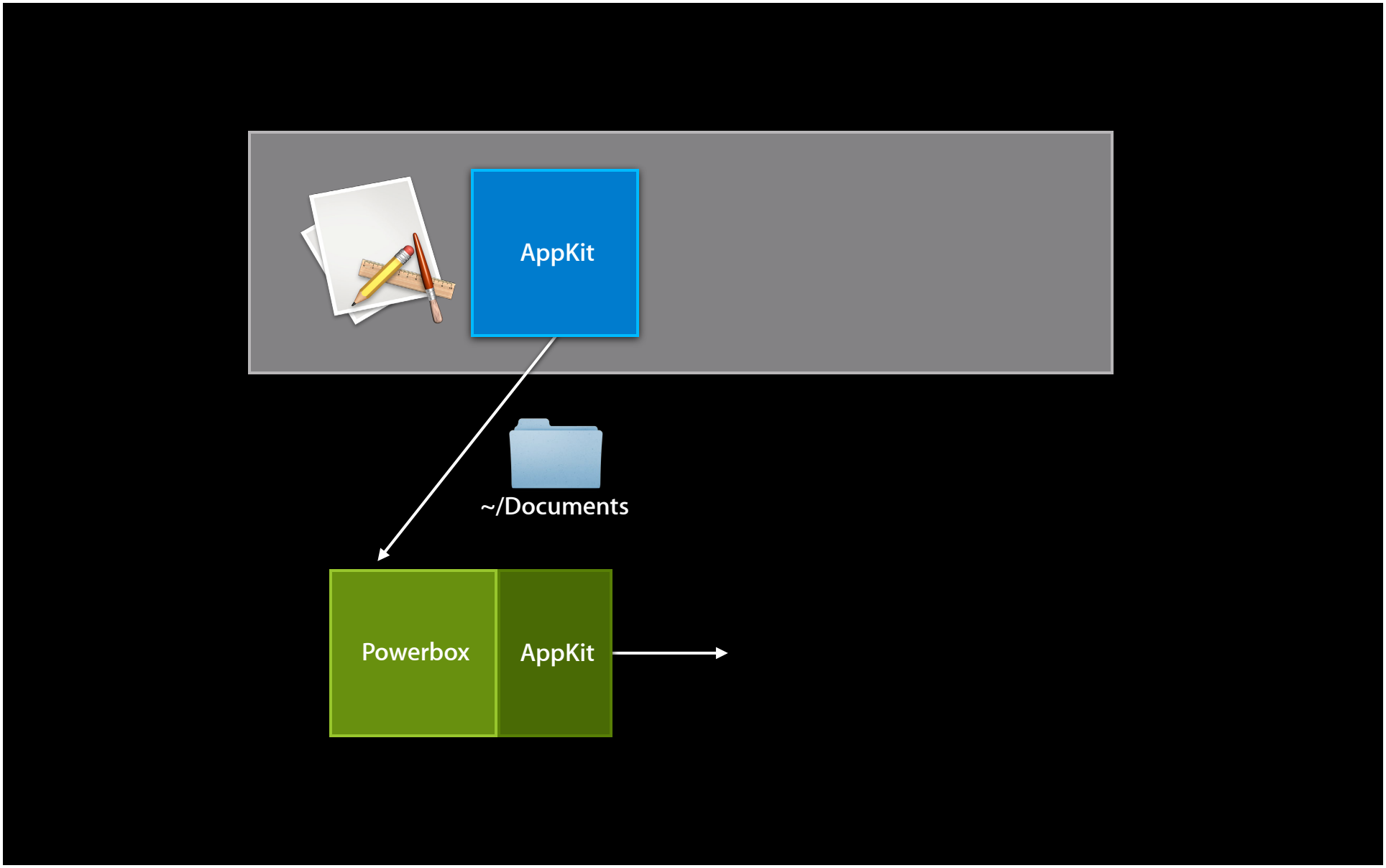


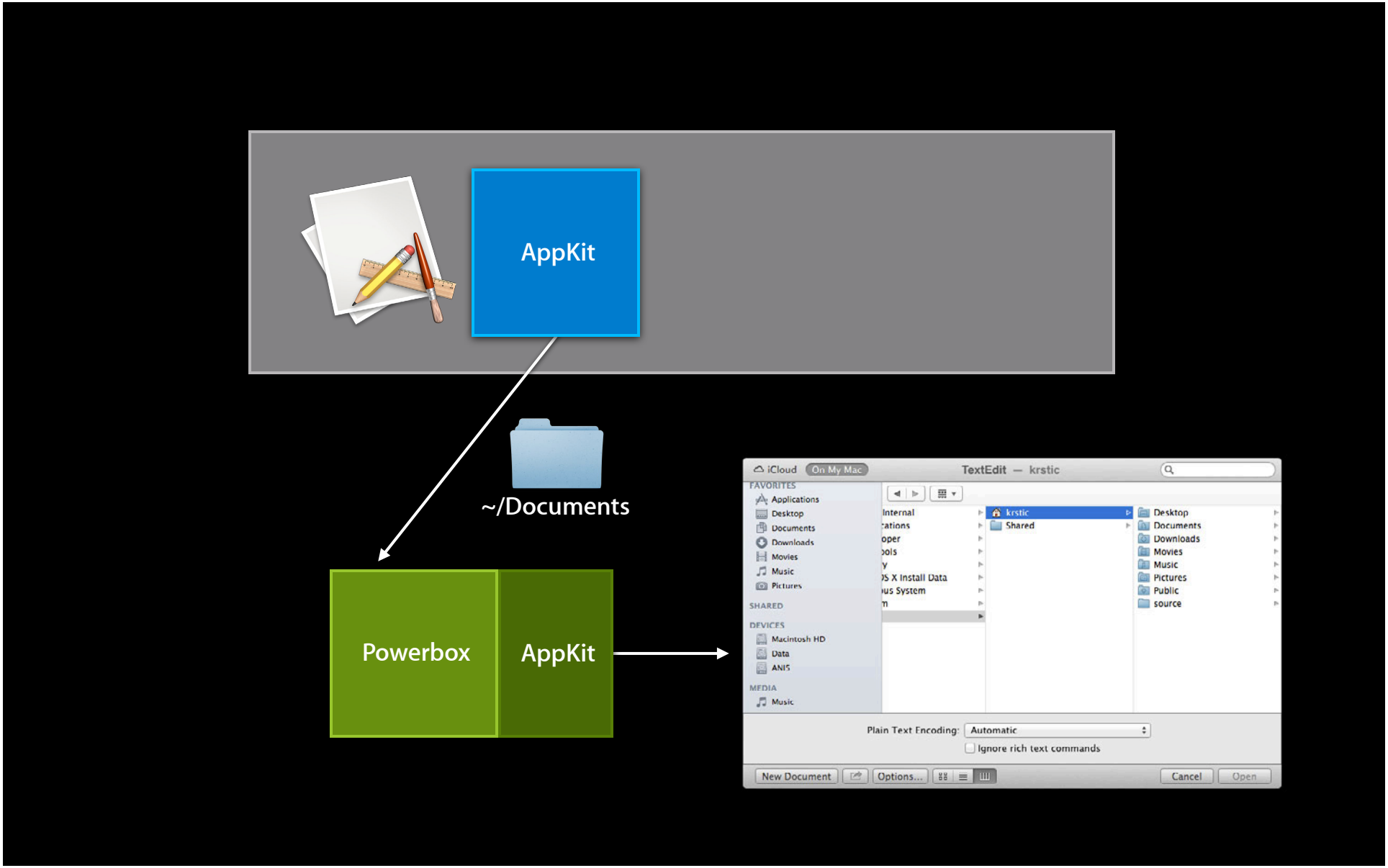
~/Documents

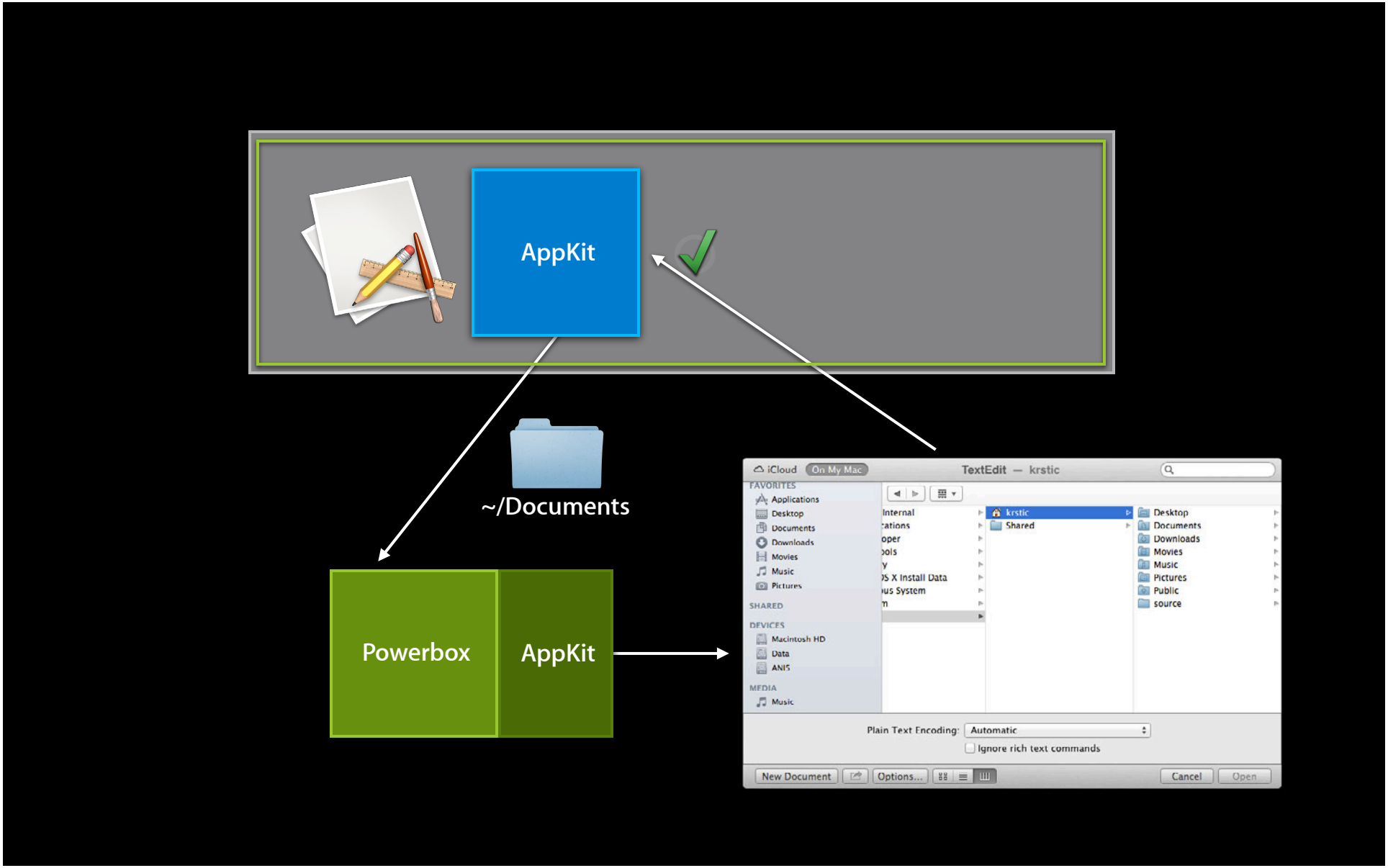



AppKit
~/Documents
Powerbox
AppKit


AppKit
~/Documents
Powerbox
AppKit


AppKit
~/Documents
Powerbox
AppKit


AppKit
~/Documents
Powerbox
AppKit
iCloud
On My Mac
TextEdit — krstic
FAVORITES
Applications
Desktop
Documents
Downloads
Movies
Music
Pictures
SHARED
DEVICES
Macintosh HD
Data
ANIS
MEDIA
Music
krstic
Shared
Desktop
Documents
Downloads
Movies
Music
Pictures
Public
source
Plain Text Encoding:
Automatic
Ignore rich text commands
New Document
Options...
Cancel
Open


AppKit
~/Documents
Powerbox
AppKit
iCloud
On My Mac
TextEdit — krstic
FAVORITES
Applications
Desktop
Documents
Downloads
Movies
Music
Pictures
SHARED
DEVICES
Macintosh HD
Data
ANI5
MEDIA
Music
krstic
Shared
Desktop
Documents
Downloads
Movies
Music
Pictures
Public
source
Plain Text Encoding:
Automatic
Ignore rich text commands
New Document
Options...
Cancel
Open

Key Components

1. Entitlements
2. Containers
3. Powerbox
4. XPC Services

XPC Services

- Very easy app and framework privilege separation
- Services have their own entitlements
- No fork/exec, process lifecycle managed by XPC
- Only available to the containing app

Putting It All Together

Adium

Adium

- Popular open source IM client
- Full featured
- 250 source files
- 75,000 lines of code for the main app
- 65,000 lines in app's own frameworks

Adium

Process

- Prepare entitlements
- Code sign program
- Run and verify App Sandbox status
- Look for violations

Demo

Adium

Exploitation

- The attacker only has access to documents that the user exchanged with buddies during this Adium run
- No ability to access or modify other apps or documents
- Need multiple vulnerabilities for a successful exploit

New Since Lion

New Since Lion

- Security-scoped bookmarks (10.7.3 and later)
- Application groups (10.7.3 and later)
- Related items
- Automation

New Since Lion

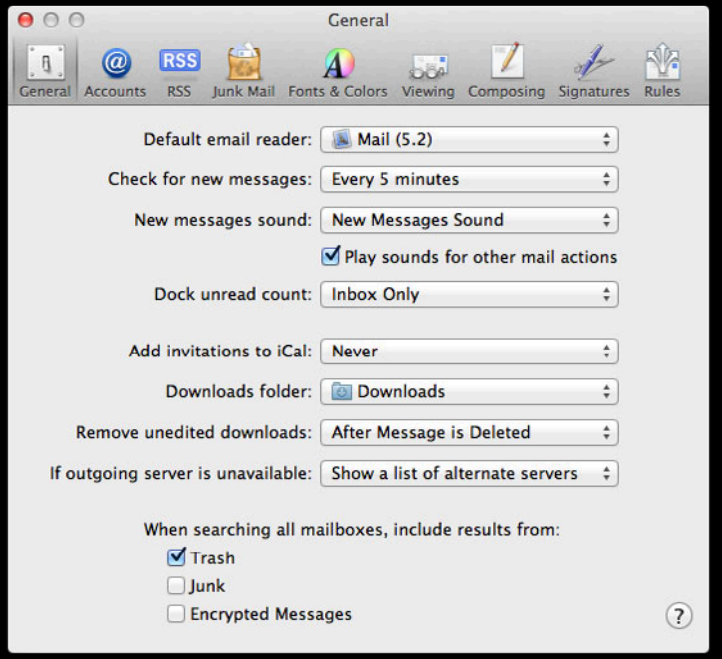
Security-scoped bookmarks

Security-Scoped Bookmarks

- Preserve access to user-chosen files and folders across system reboot
- Per-user app configuration: Input and output folders, commonly accessed files
- Document formats that contain references to files

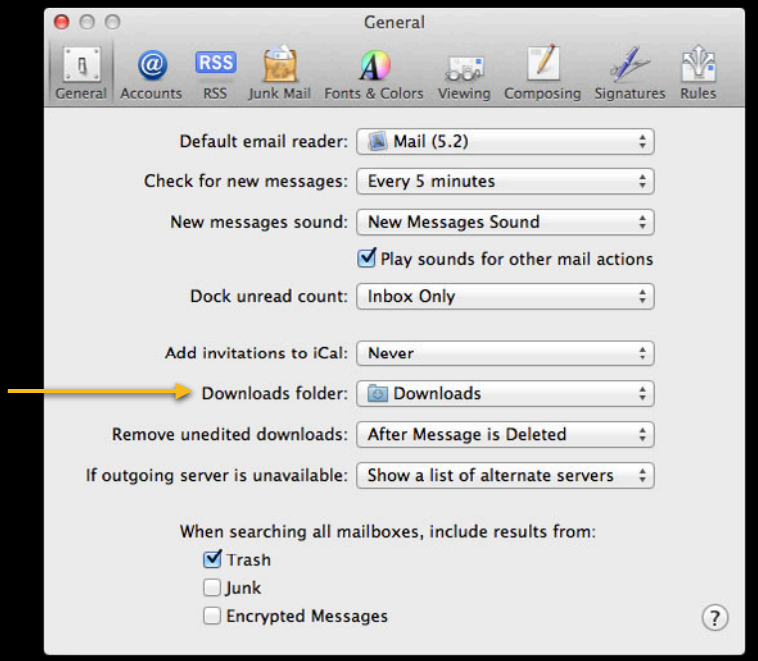
Security-Scoped Bookmarks

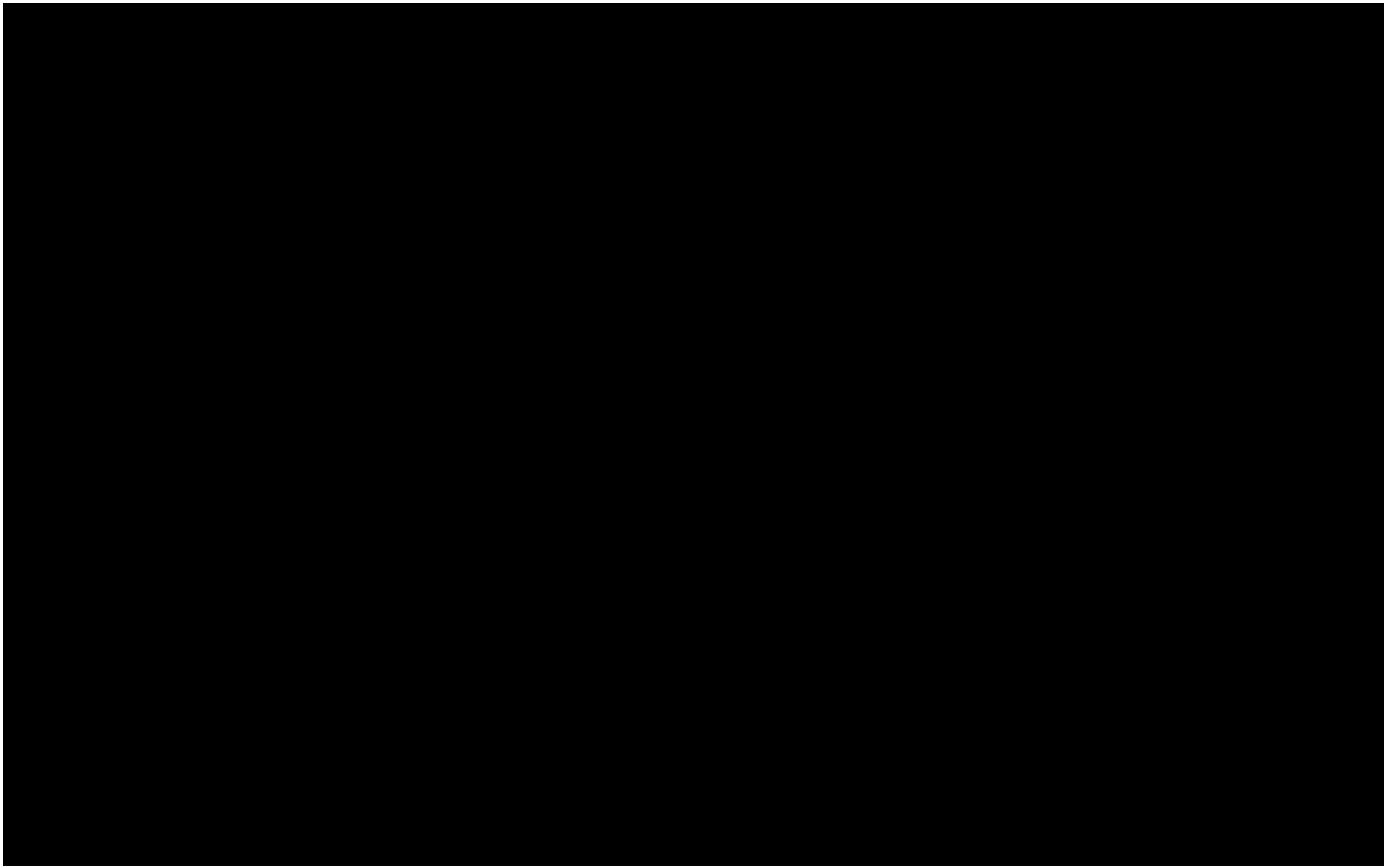
- App scope
  - `com.apple.security.files.user-selected.read-{write,only}`
  - Locked to the app and user that created them

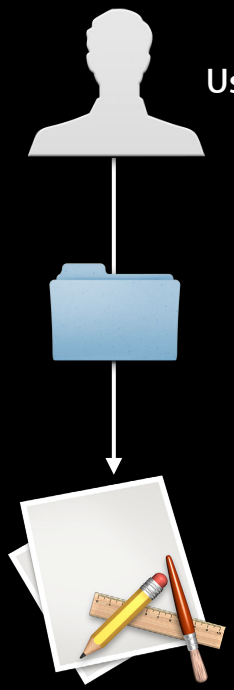


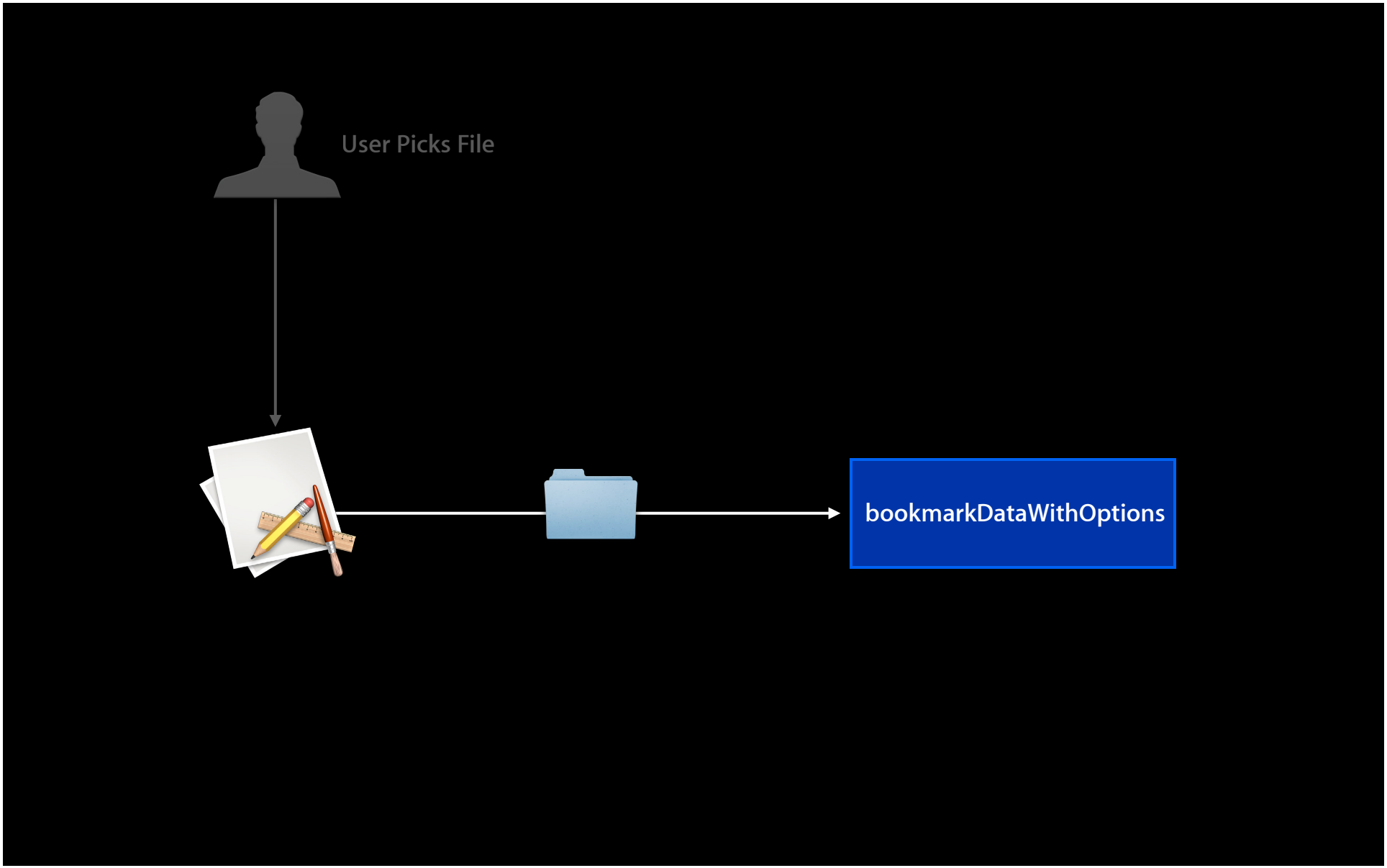
Security-Scoped Bookmarks

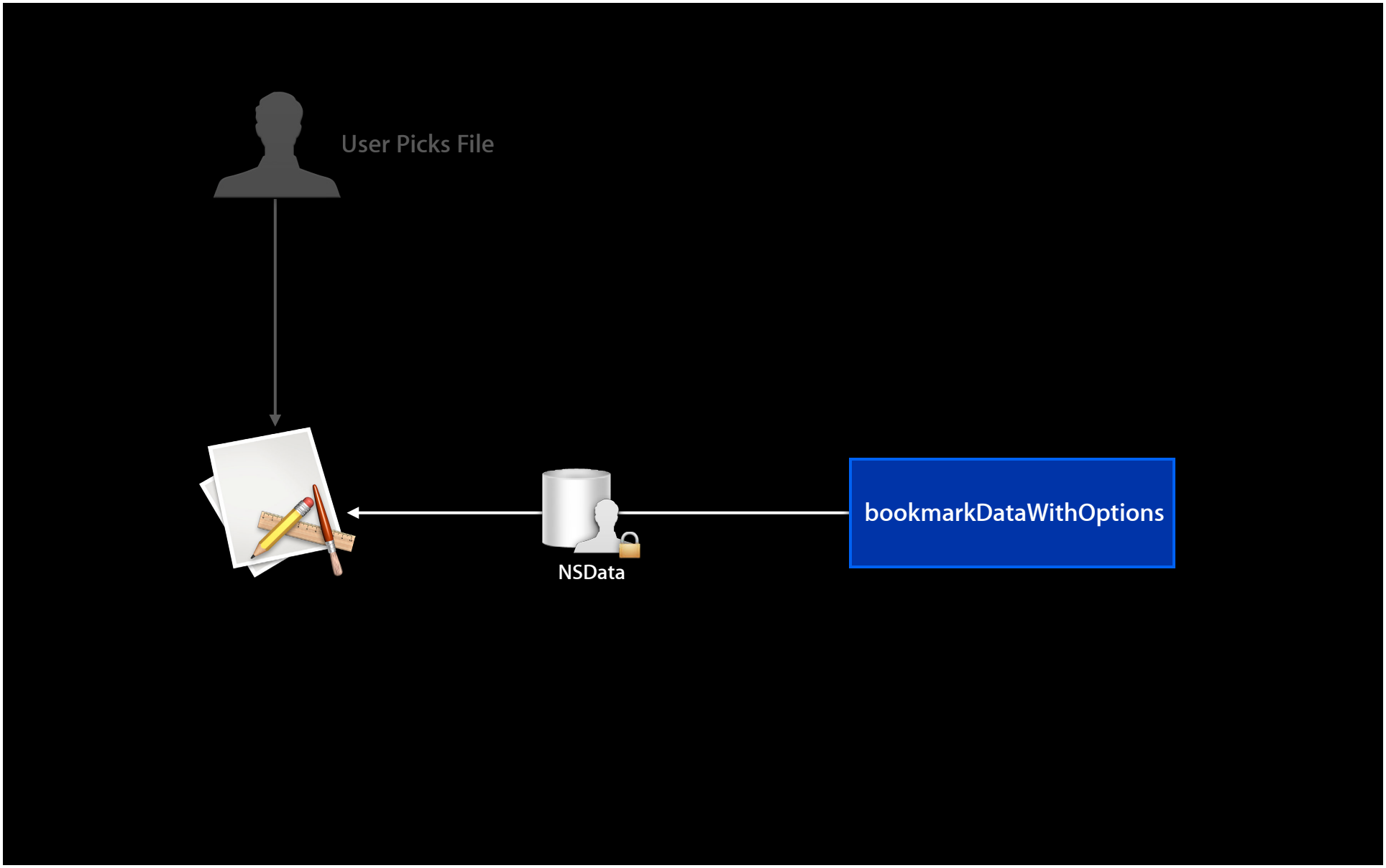
- App scope
  - `com.apple.security.files.user-selected.read-{write,only}`
  - Locked to the app and user that created them

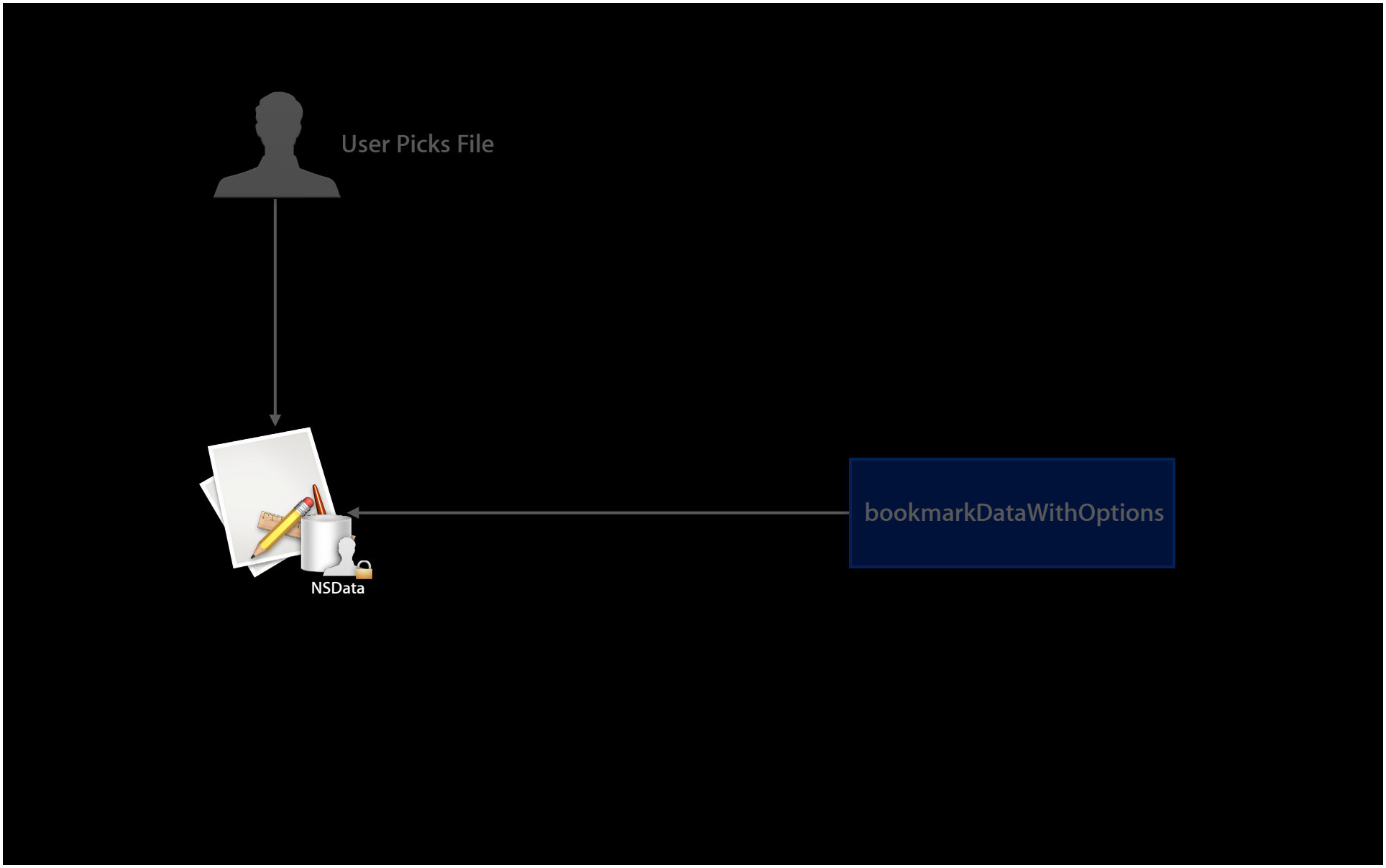


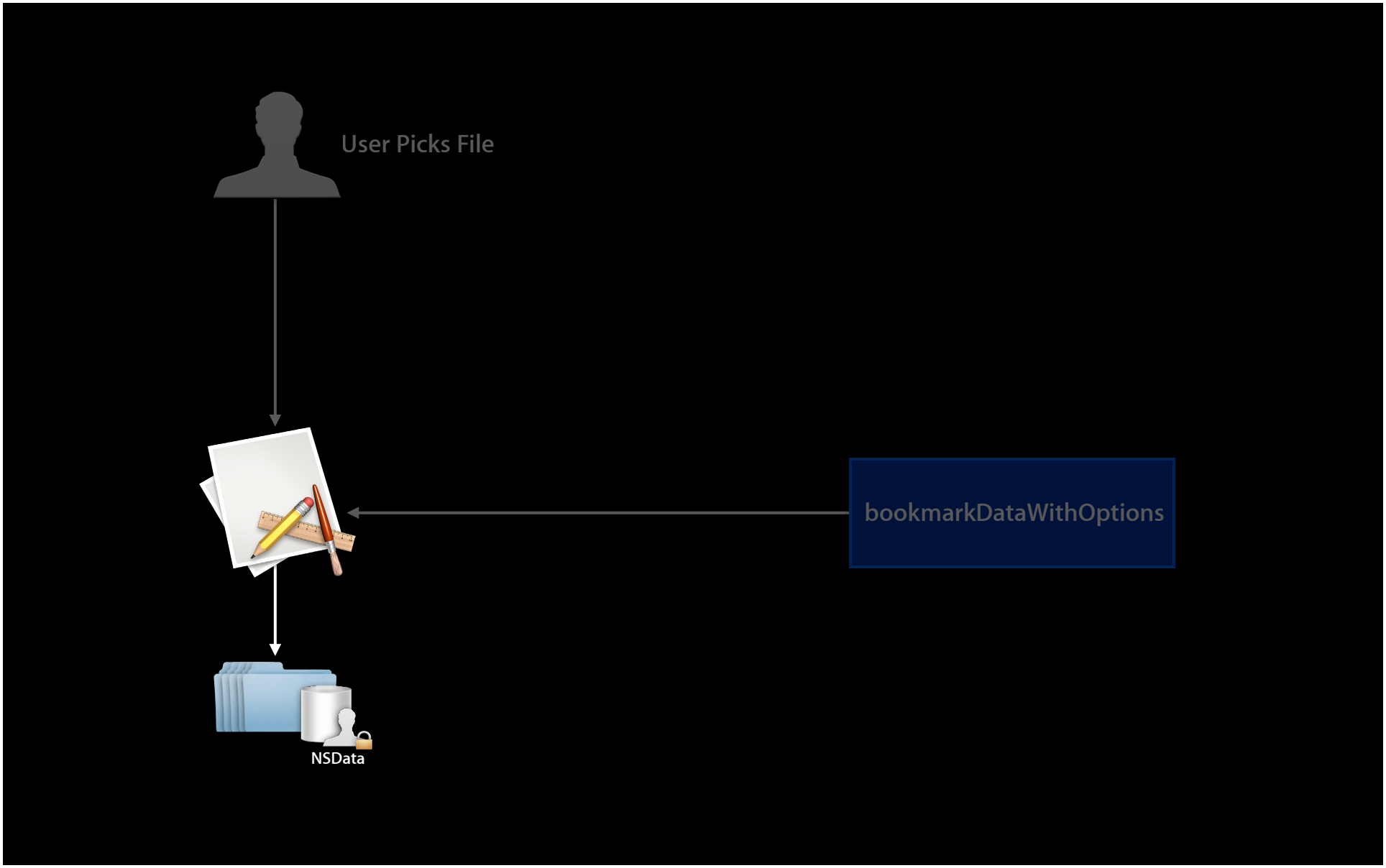


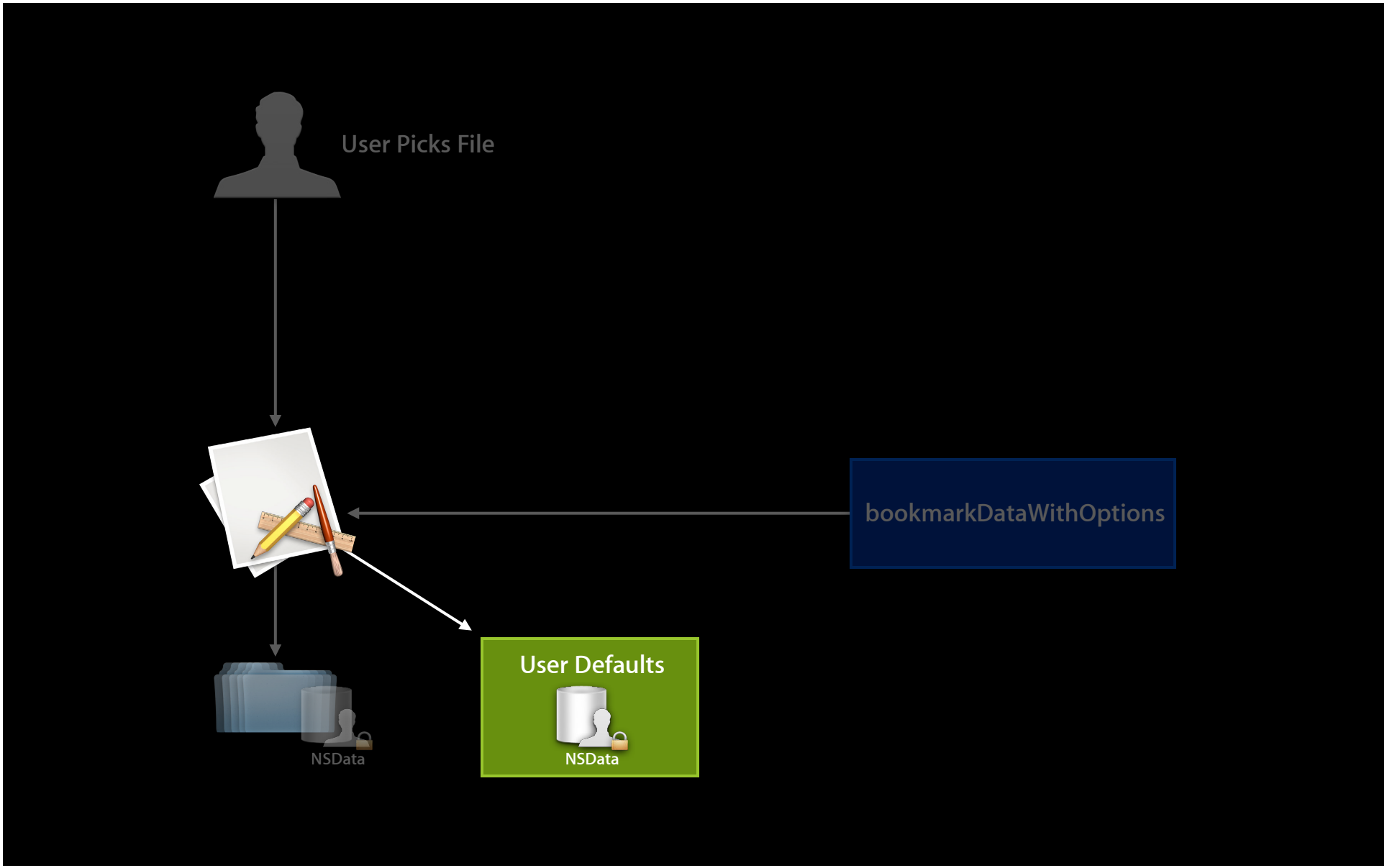

User Picks File

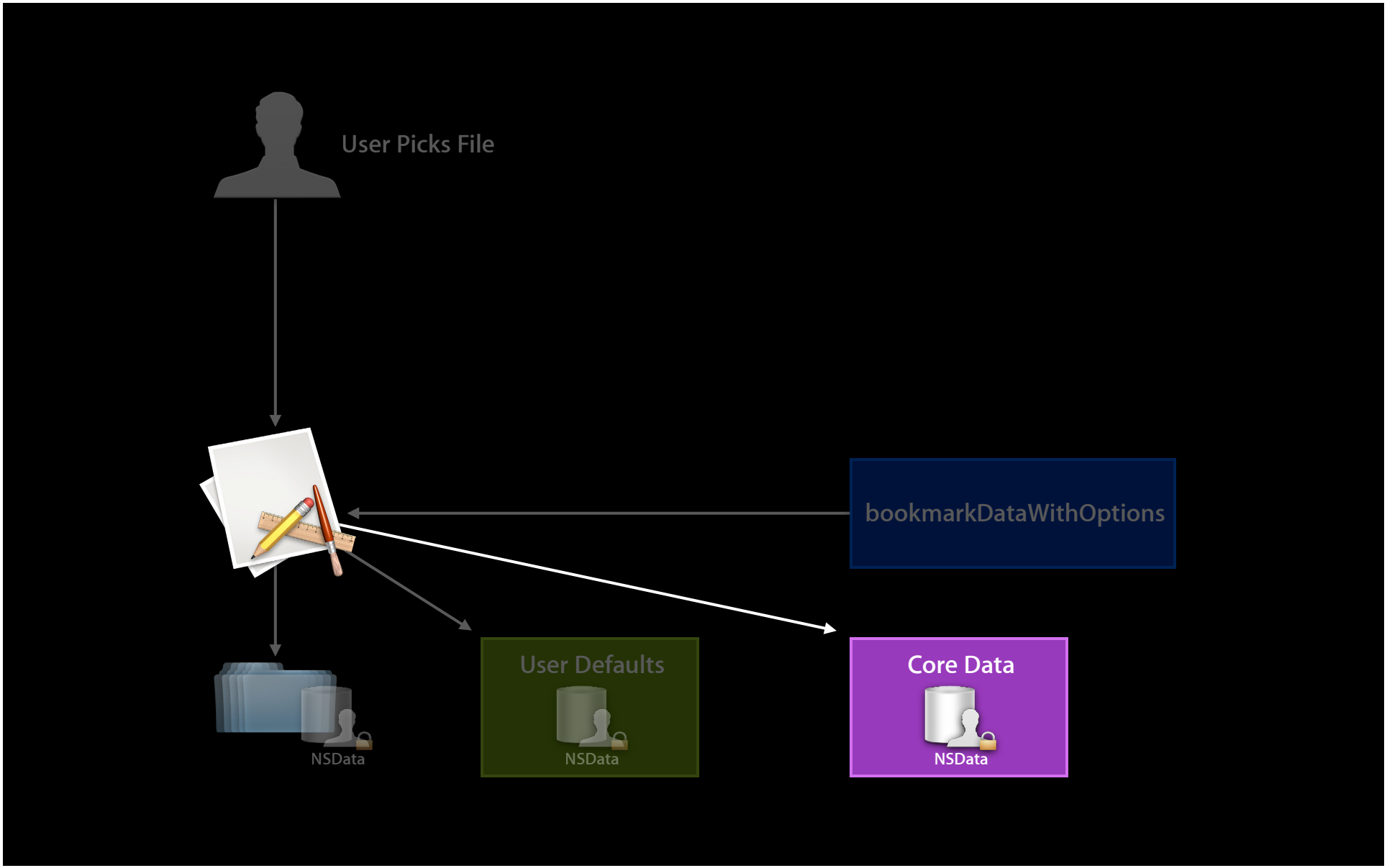

User Picks File
bookmarkDataWithOptions

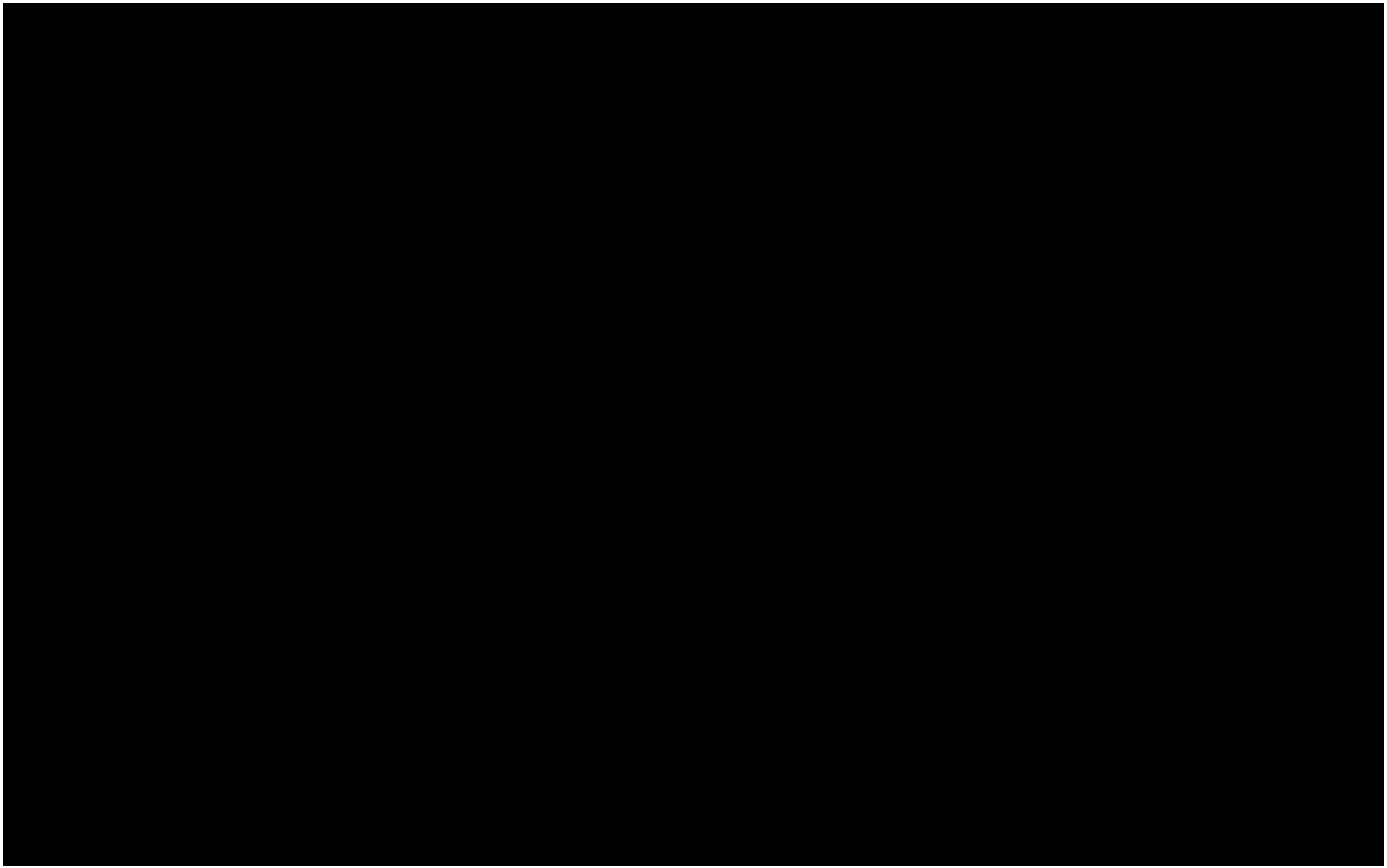

User Picks File
NSData
bookmarkDataWithOptions


User Picks File
NSData
bookmarkDataWithOptions

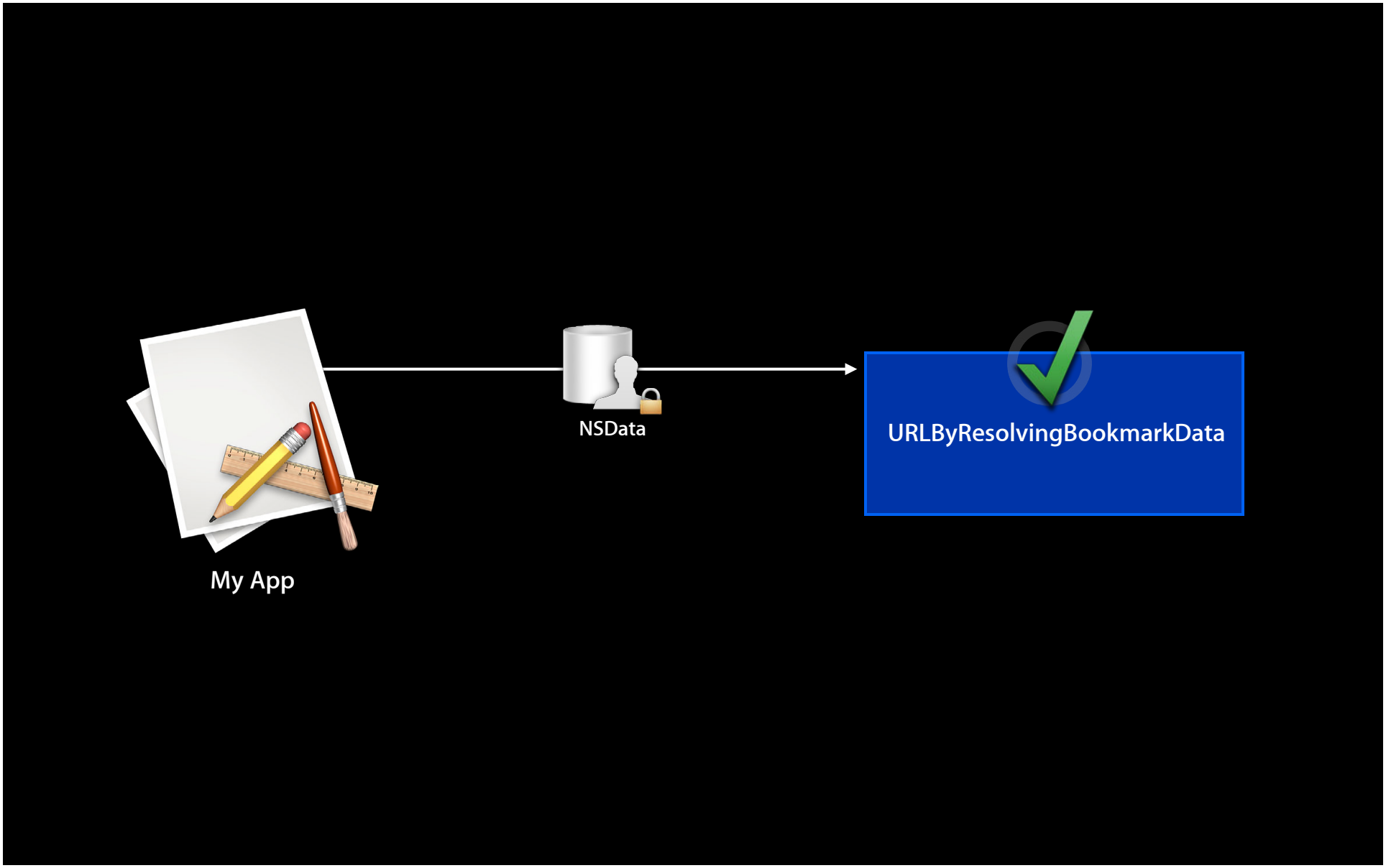

User Picks File
bookmarkDataWithOptions
NSData

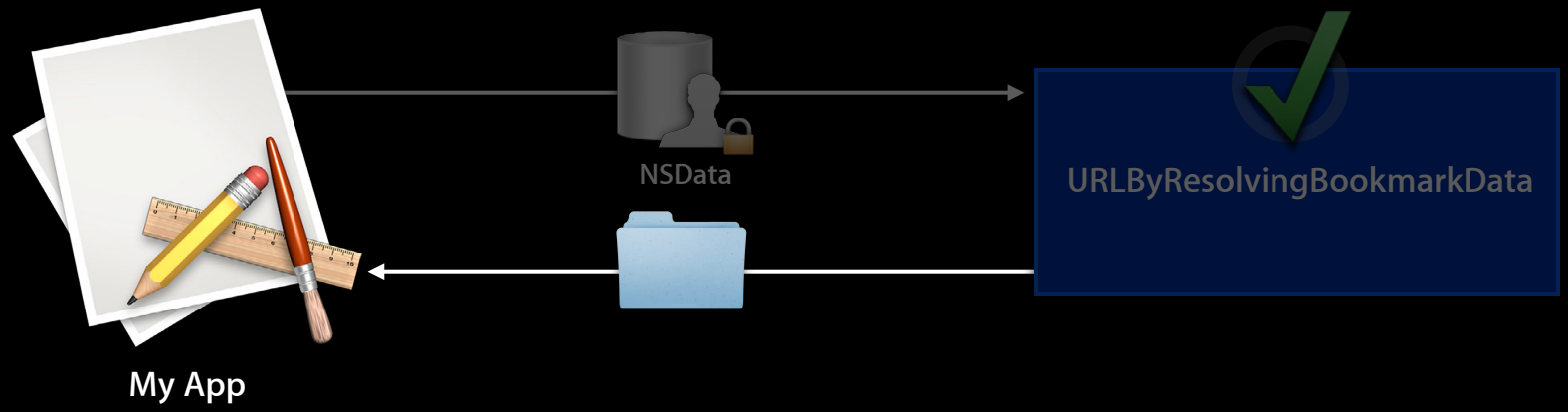

User Picks File
bookmarkDataWithOptions
User Defaults
NSData
NSData

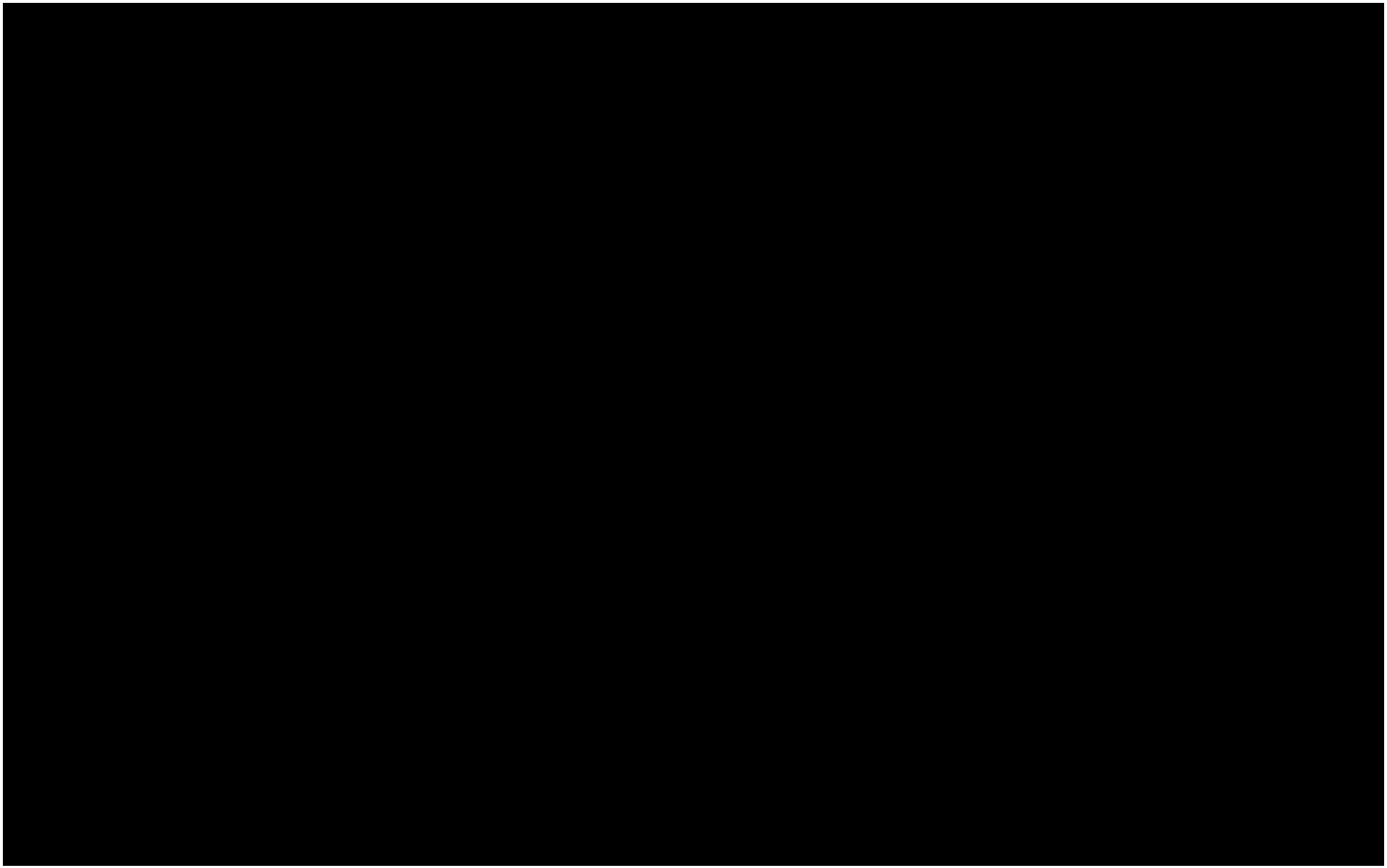

User Picks File
bookmarkDataWithOptions
NSData
User Defaults
NSData
Core Data
NSData



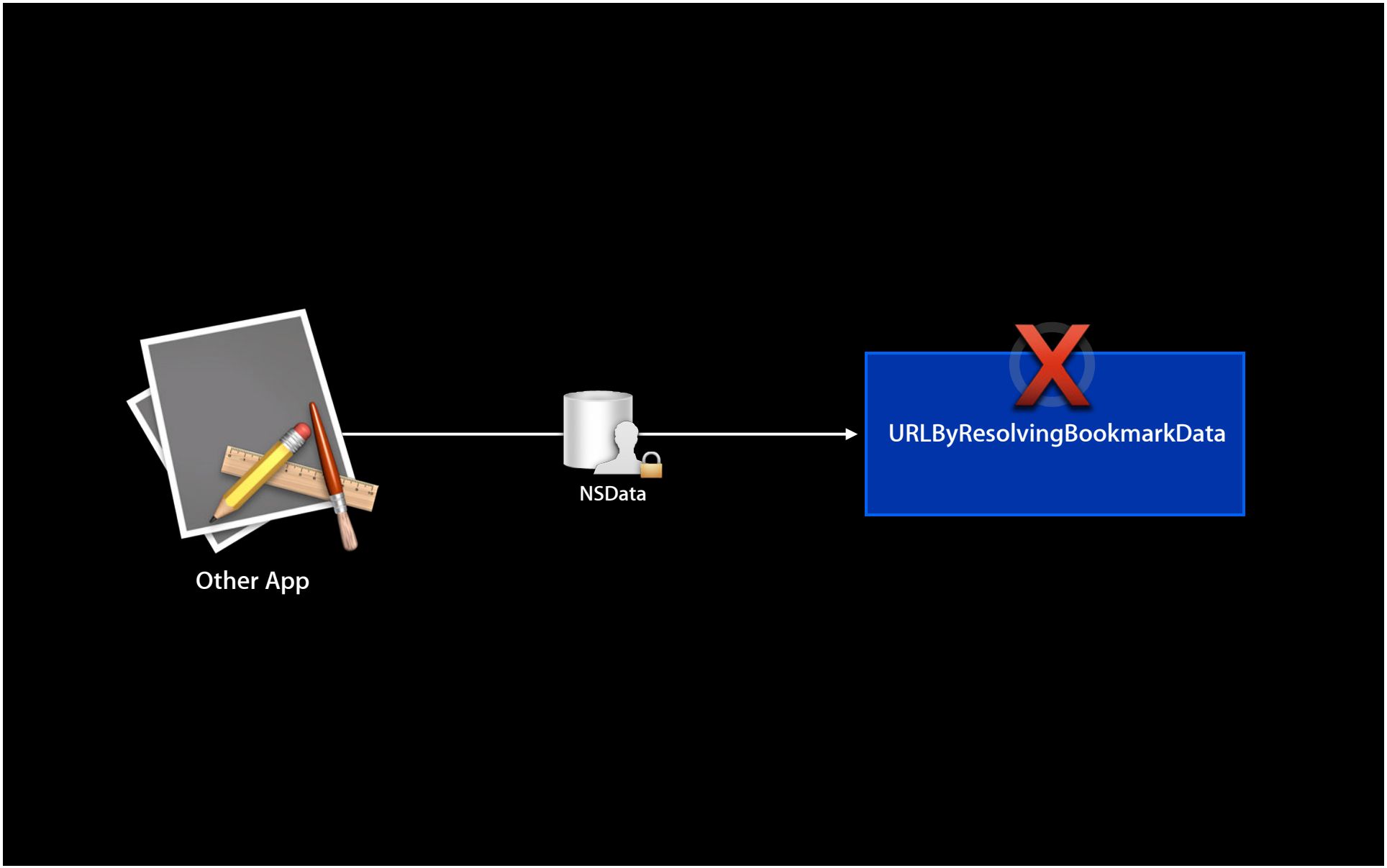
My App


My App
NSData
URLByResolvingBookmarkData


NSData
URLByResolvingBookmarkData
My App

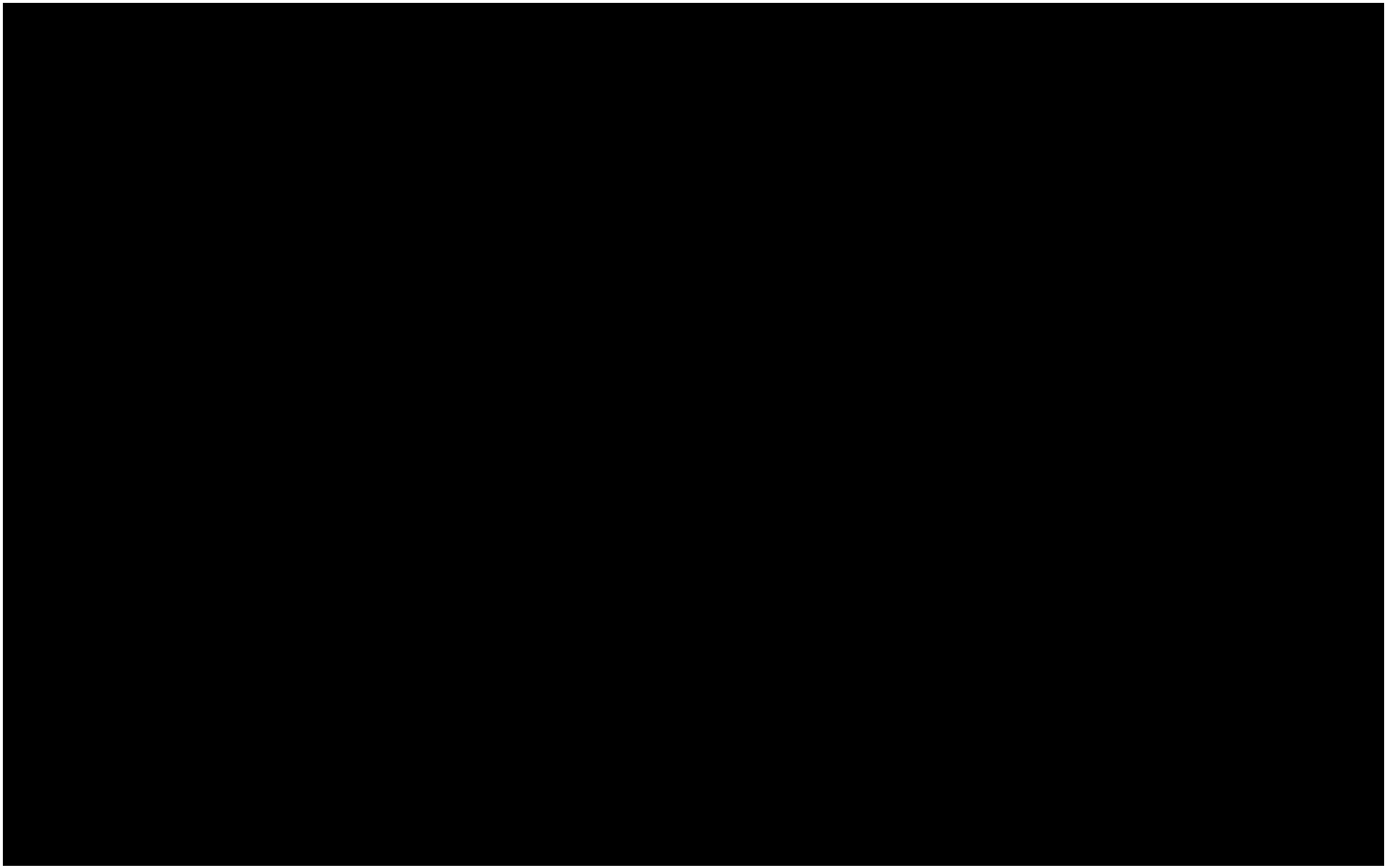


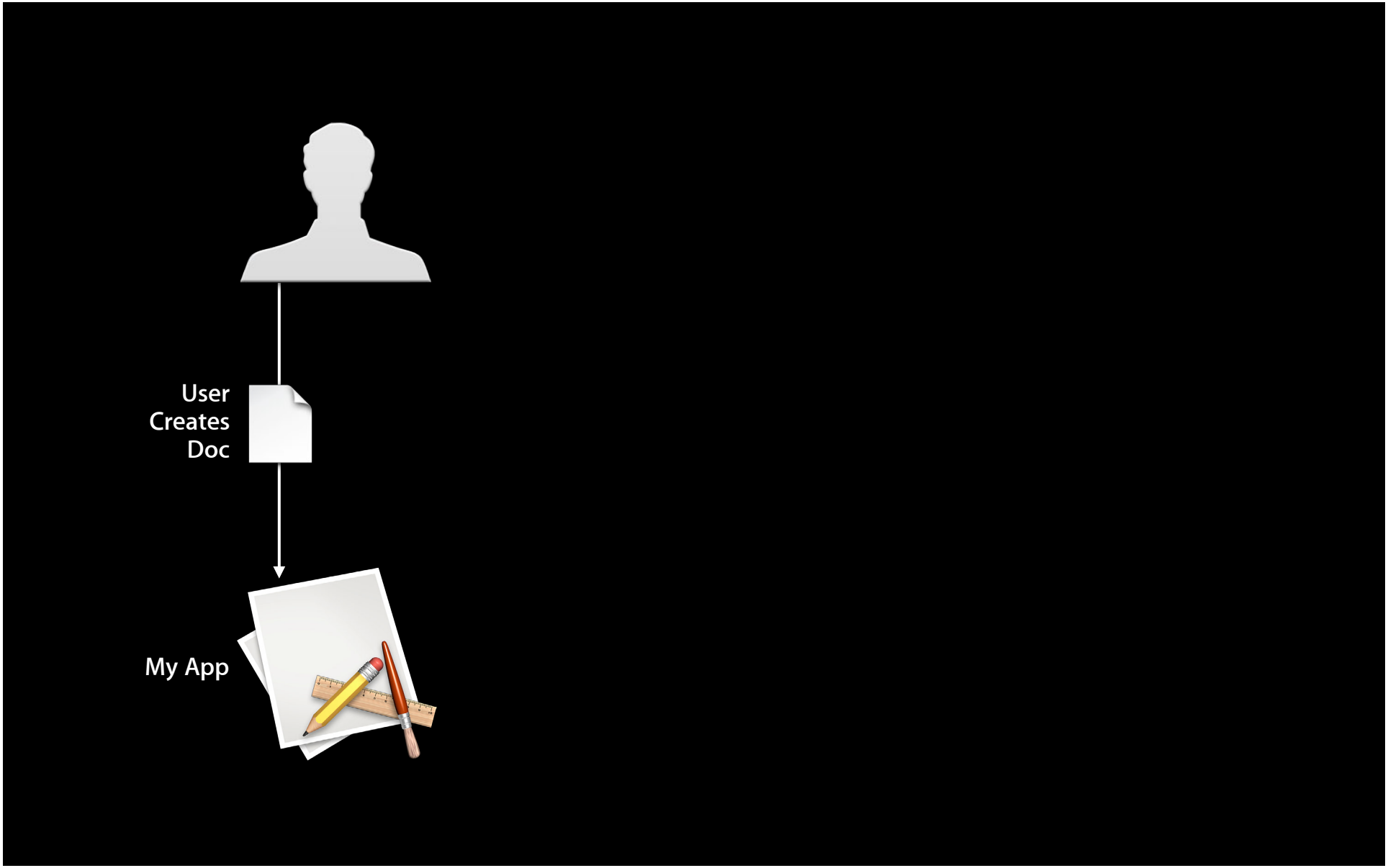
Other App

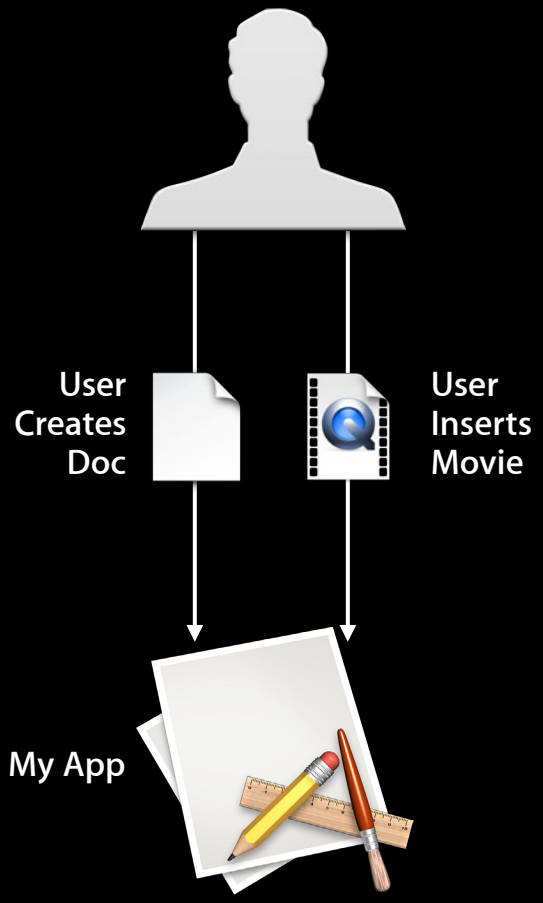

Other App
NSData
URLByResolvingBookmarkData

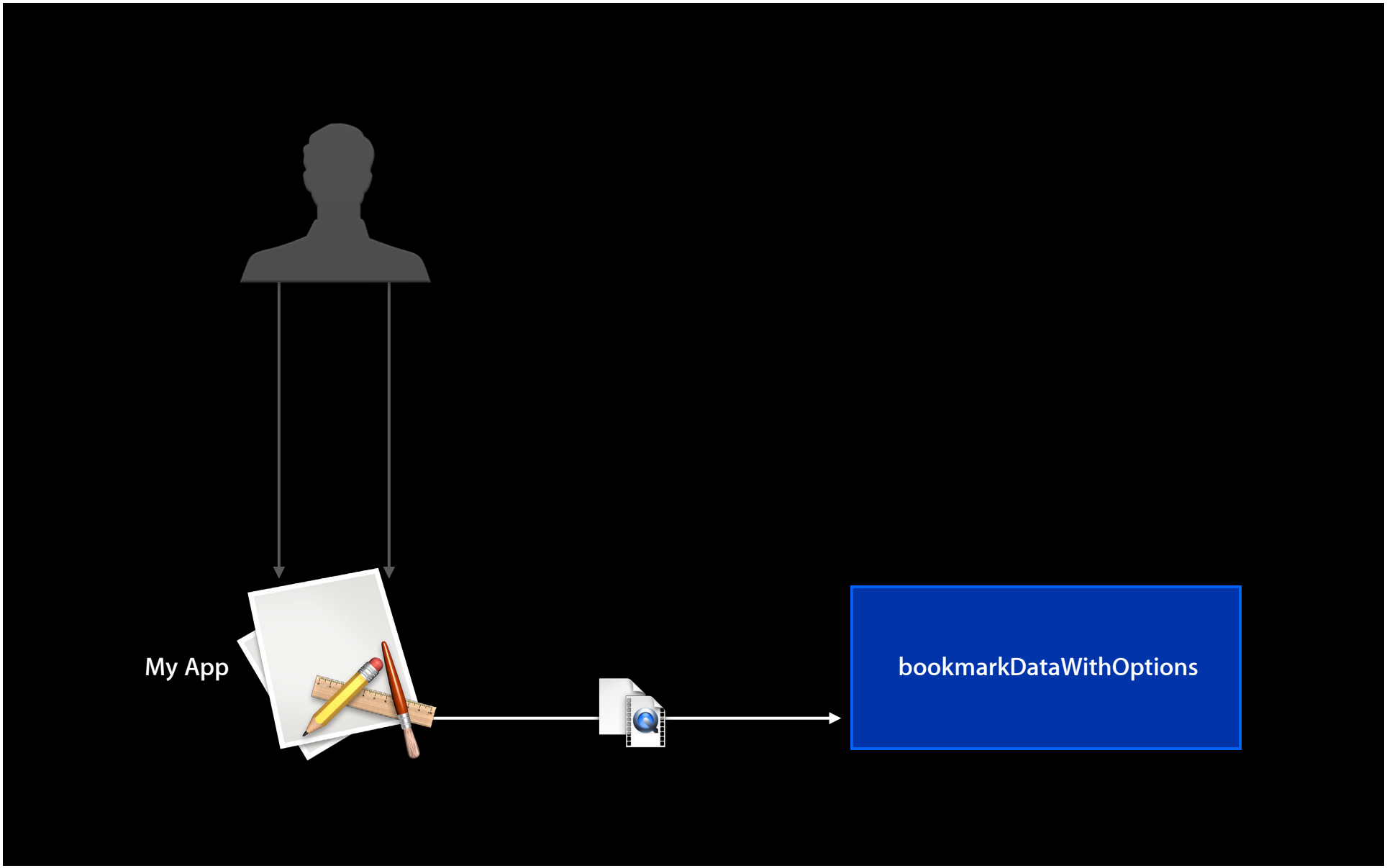
Security-Scoped Bookmarks

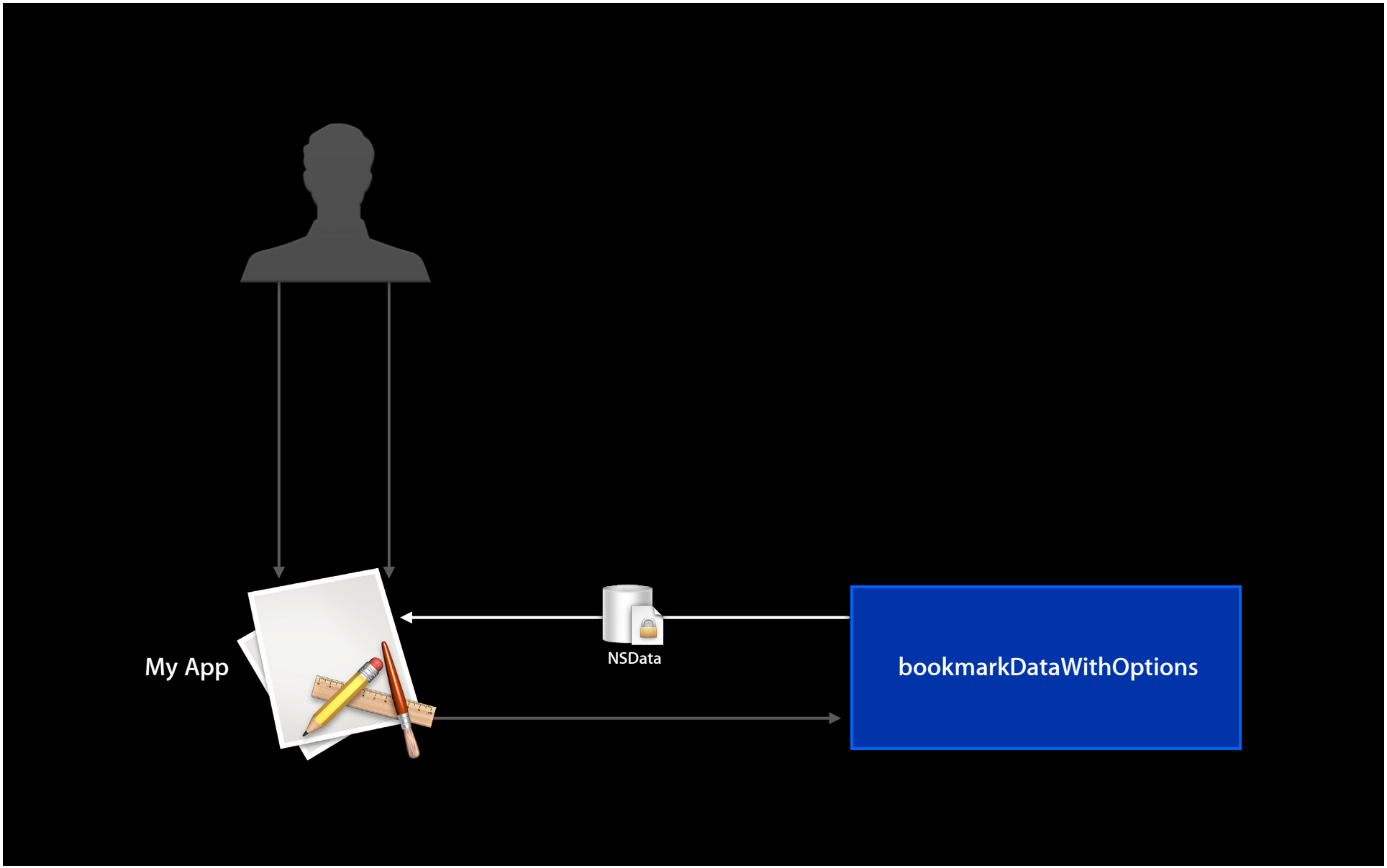
- Document scope
  - `com.apple.security.files.bookmarks.document-scope`
  - Allows a document format to contain references to files (but not folders) that travel with it
  - Bookmark must be stored in the document file/bundle itself
  - Cannot point to system or hidden locations (~/Library)

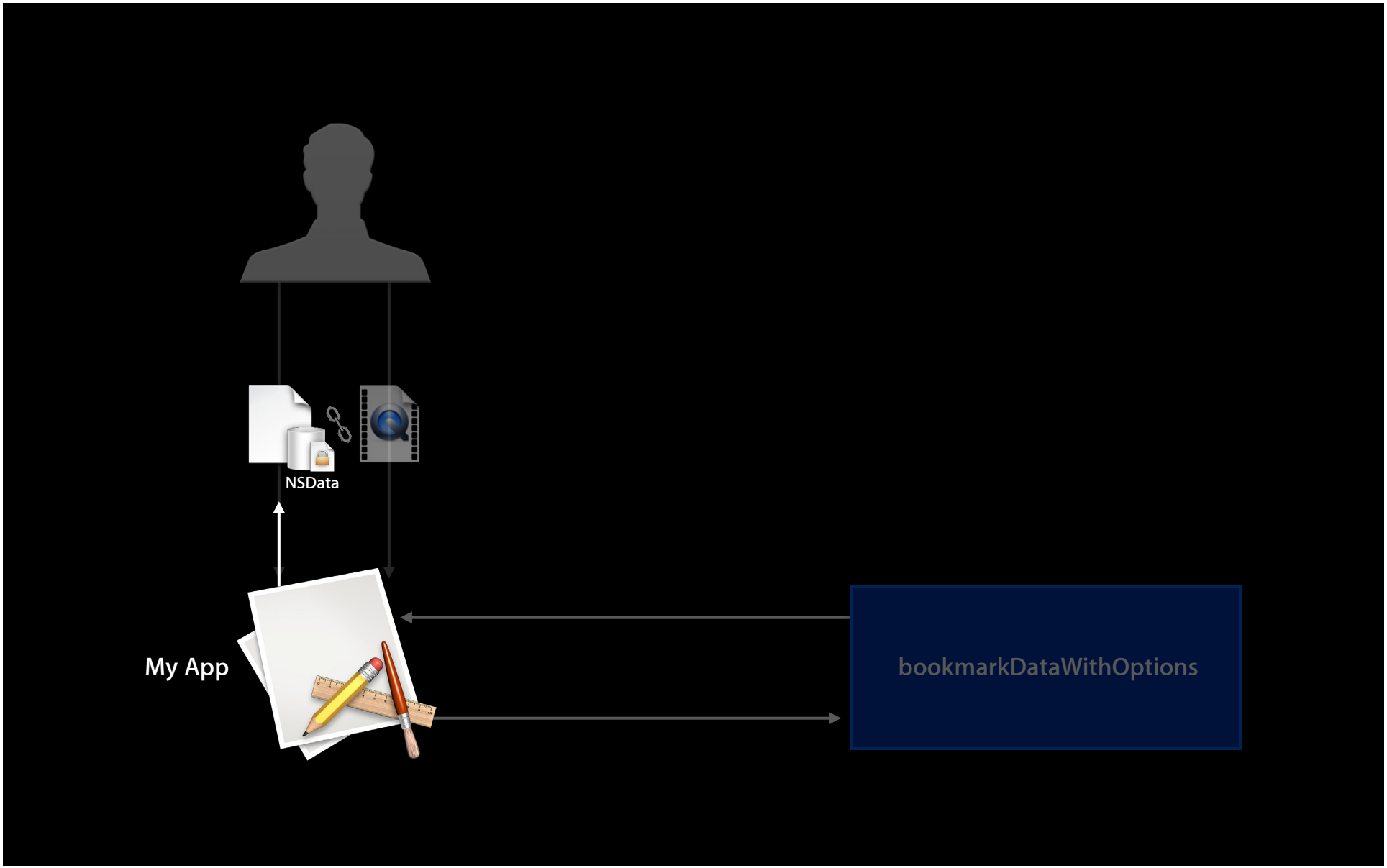


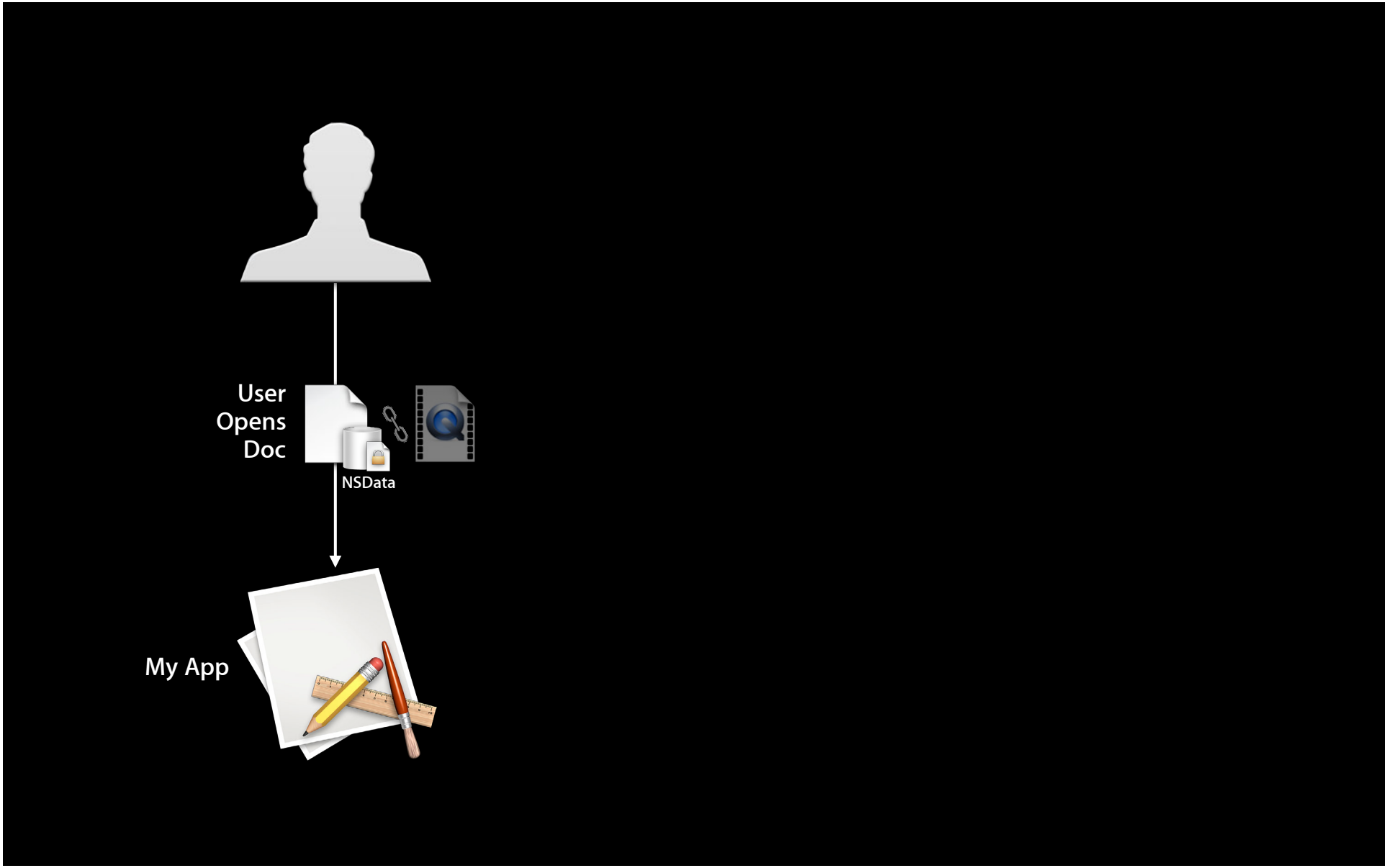

User
Creates
Doc
My App

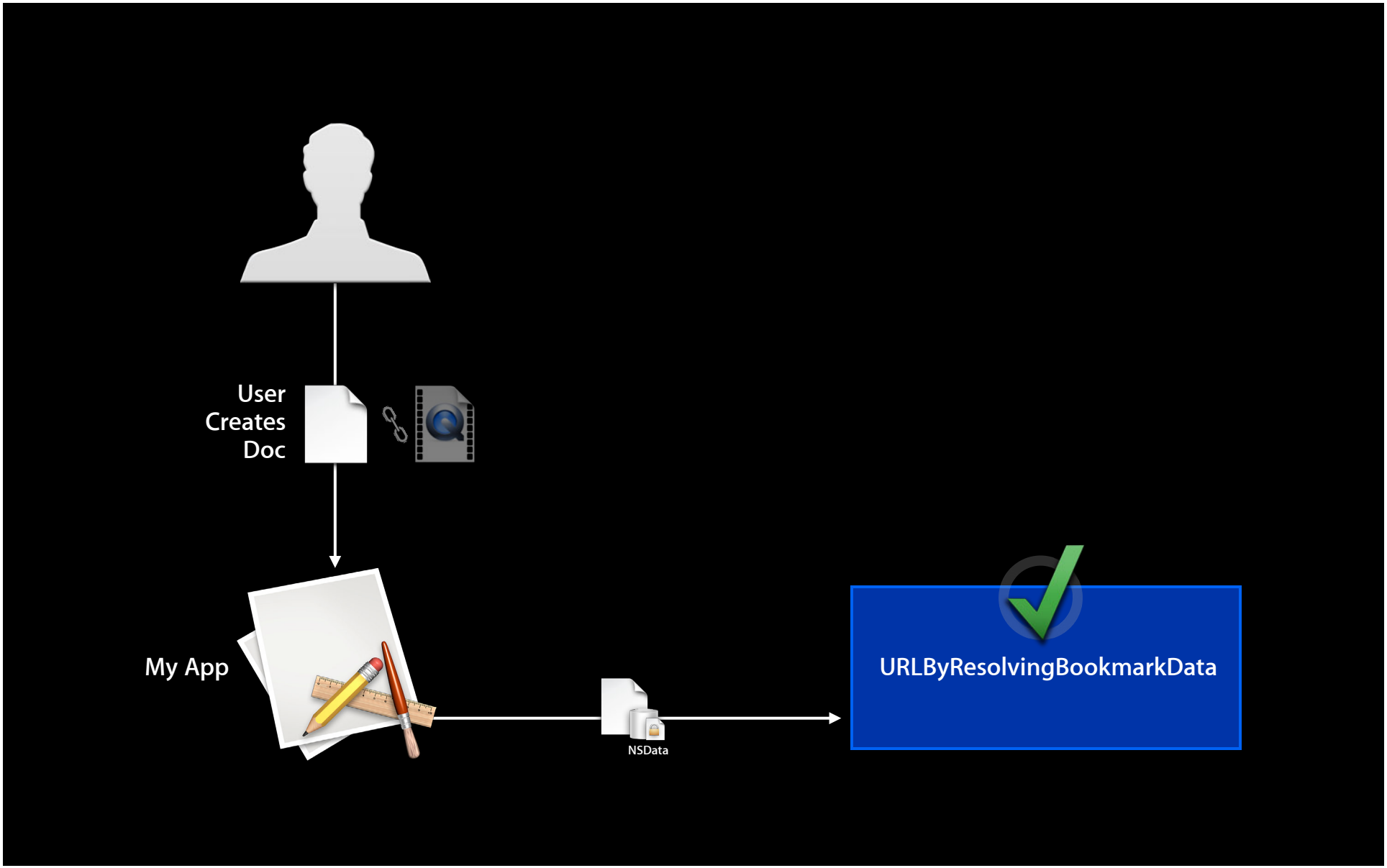

User
Creates
Doc
User
Inserts
Movie
My App

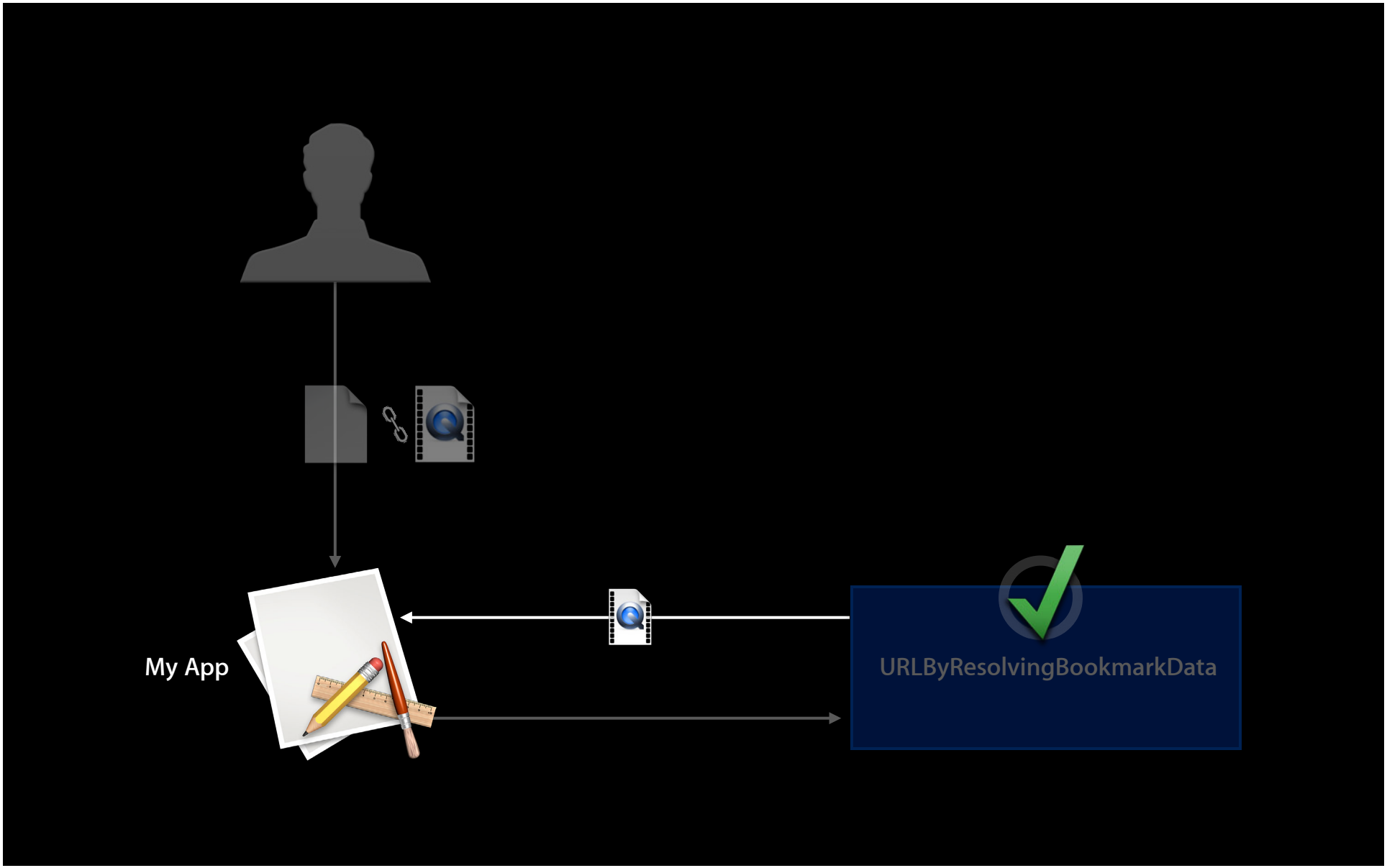

My App
bookmarkDataWithOptions

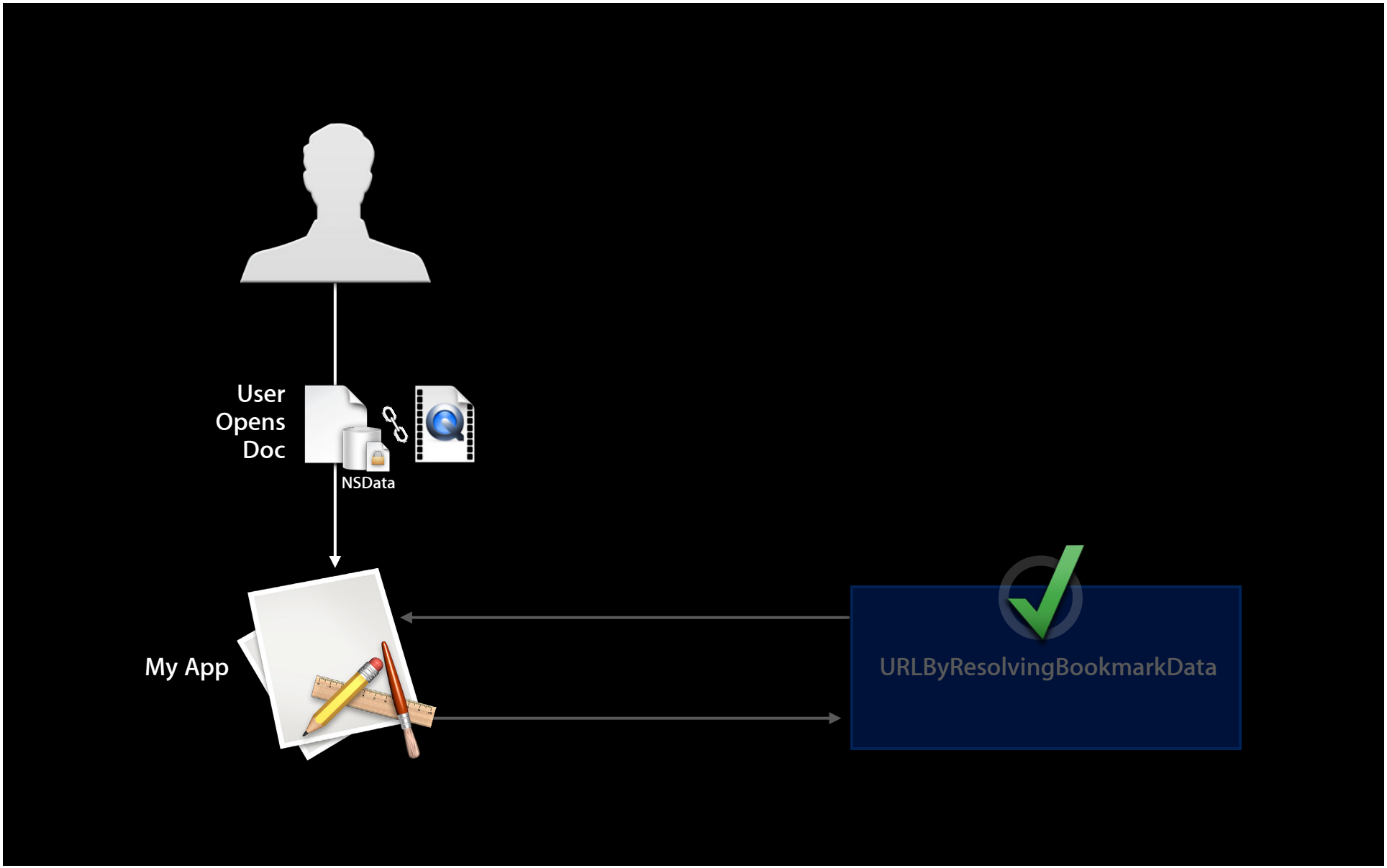

My App
NSData
bookmarkDataWithOptions

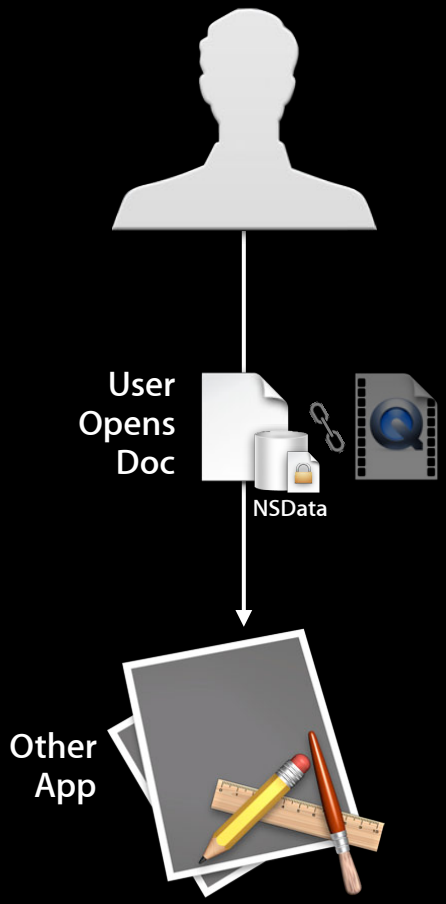

NSData
My App
bookmarkDataWithOptions

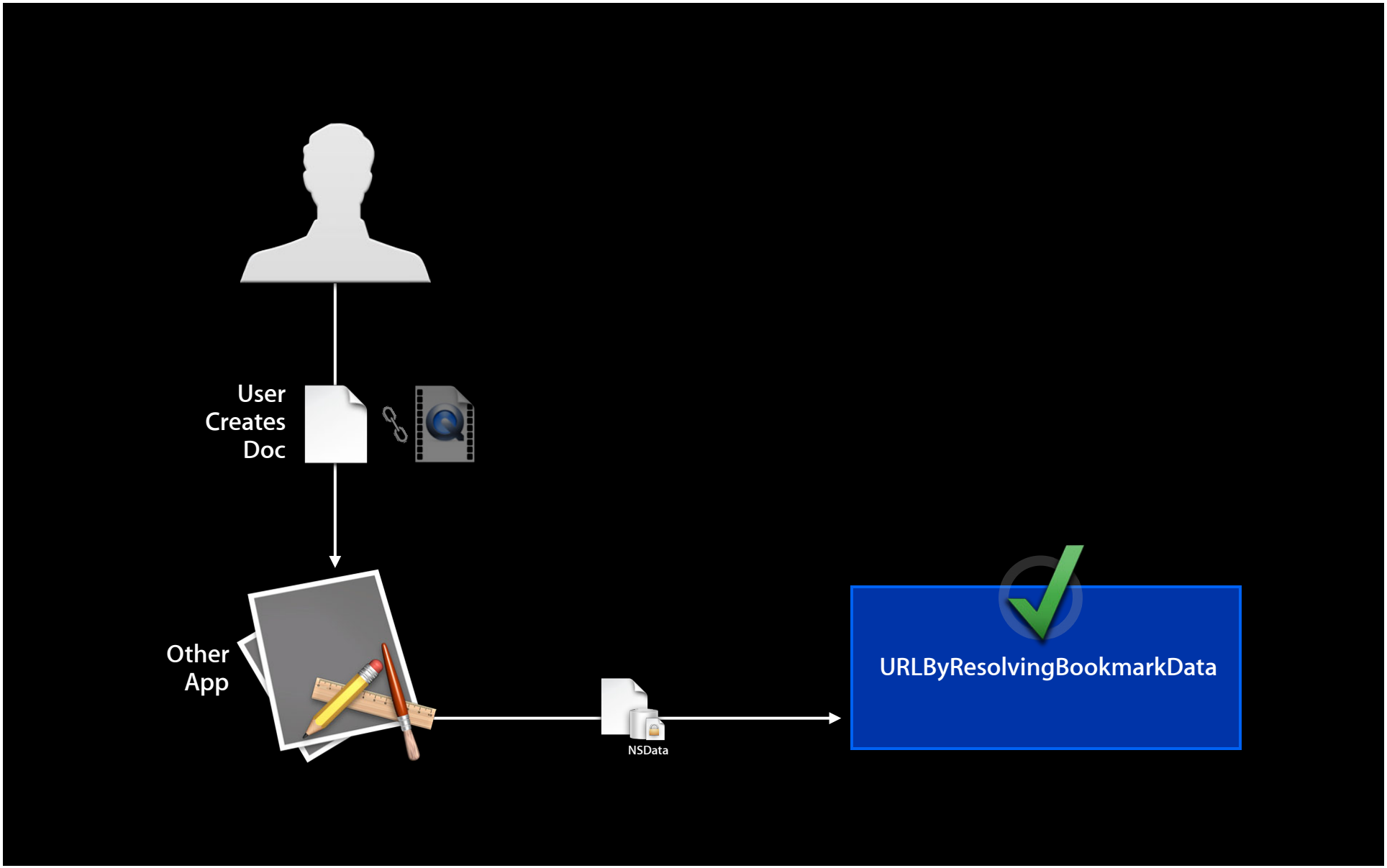

User
Opens
Doc
NSData
My App

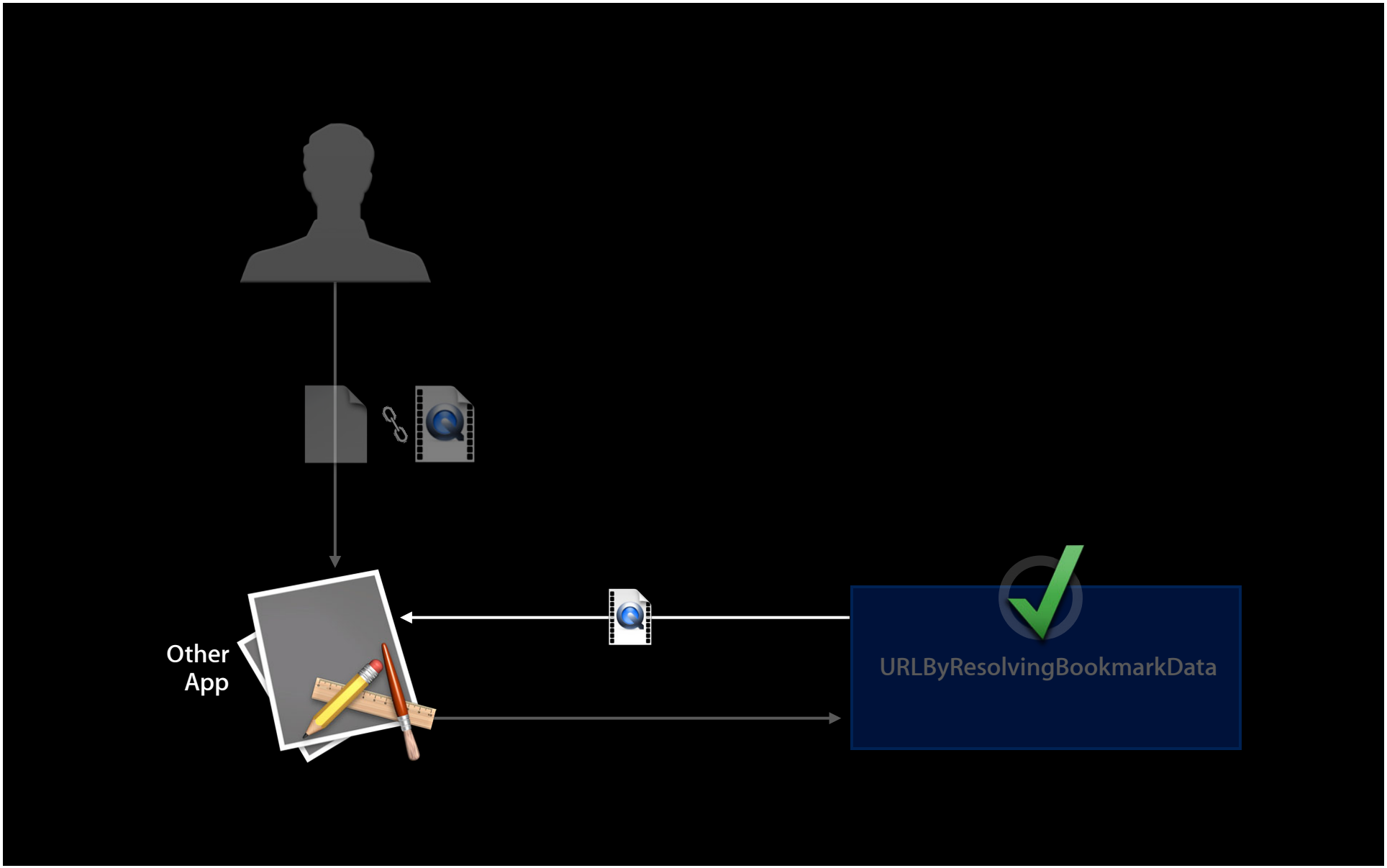

User
Creates
Doc
My App
NSData
URLByResolvingBookmarkData

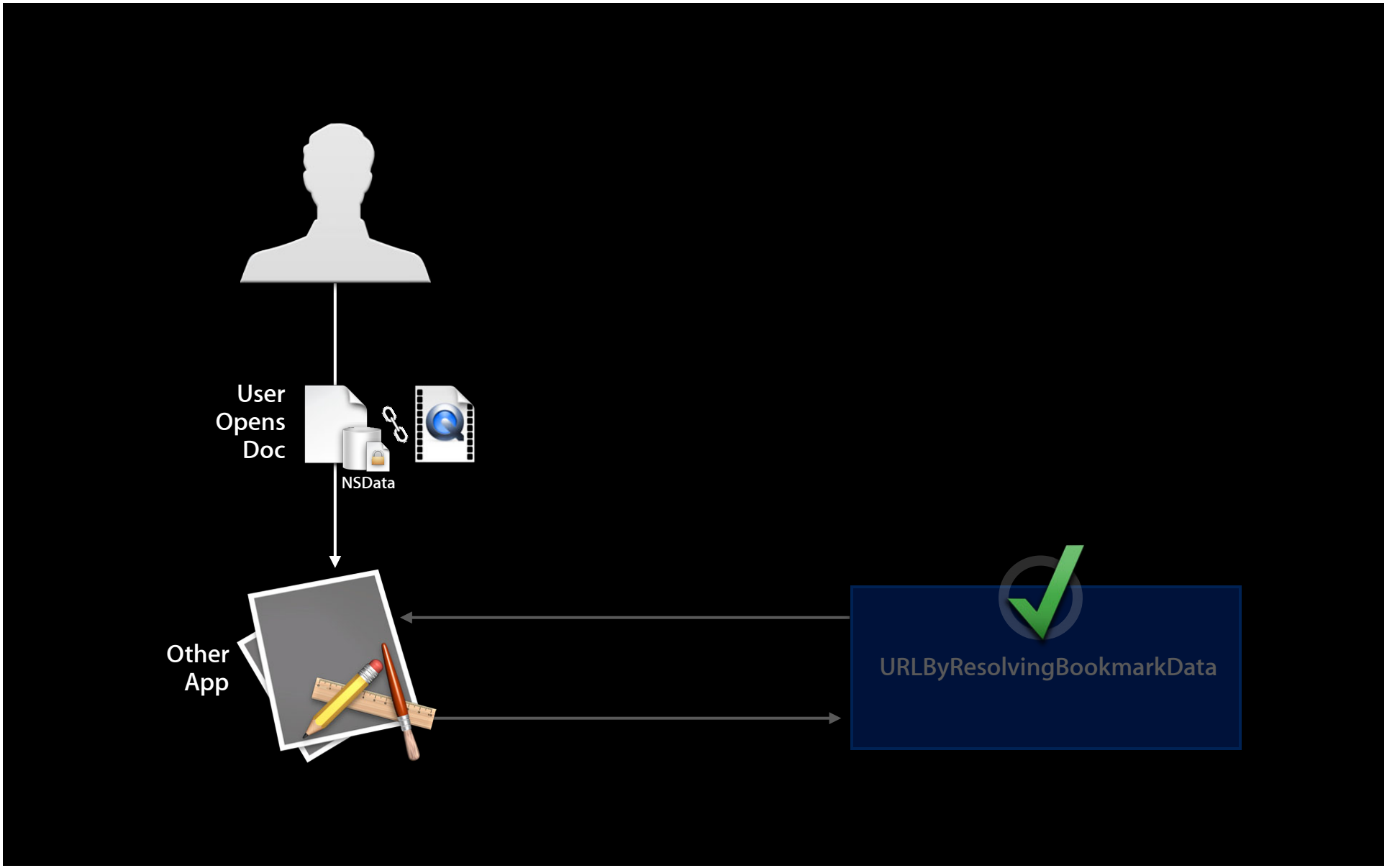

My App
URLByResolvingBookmarkData


User
Opens
Doc
NSData
My App
URLByResolvingBookmarkData


User
Opens
Doc
NSData
Other
App


User
Creates
Doc
Other
App
NSData
URLByResolvingBookmarkData


Other
App
URLByResolvingBookmarkData


User
Opens
Doc
NSData
Other
App
URLByResolvingBookmarkData

Security-Scoped Bookmarks

- No new API, just a flag on existing NSURL methods
  - `+ URLByResolvingBookmarkData:options:relativeToURL:bookmarkDataIsStale:error:`
  - `– bookmarkDataWithOptions:includingResourceValuesForKeys:relativeToURL:error:`
- Big difference: Resolution returns a *security-scoped NSURL*
  - Must call `{start,stop}AccessingSecurityScopedResource` to gain and discontinue access to resource
  - Failure to do so will leak kernel resources and will eventually suspend app's ability to use user-selected files until relaunch

New Since Lion

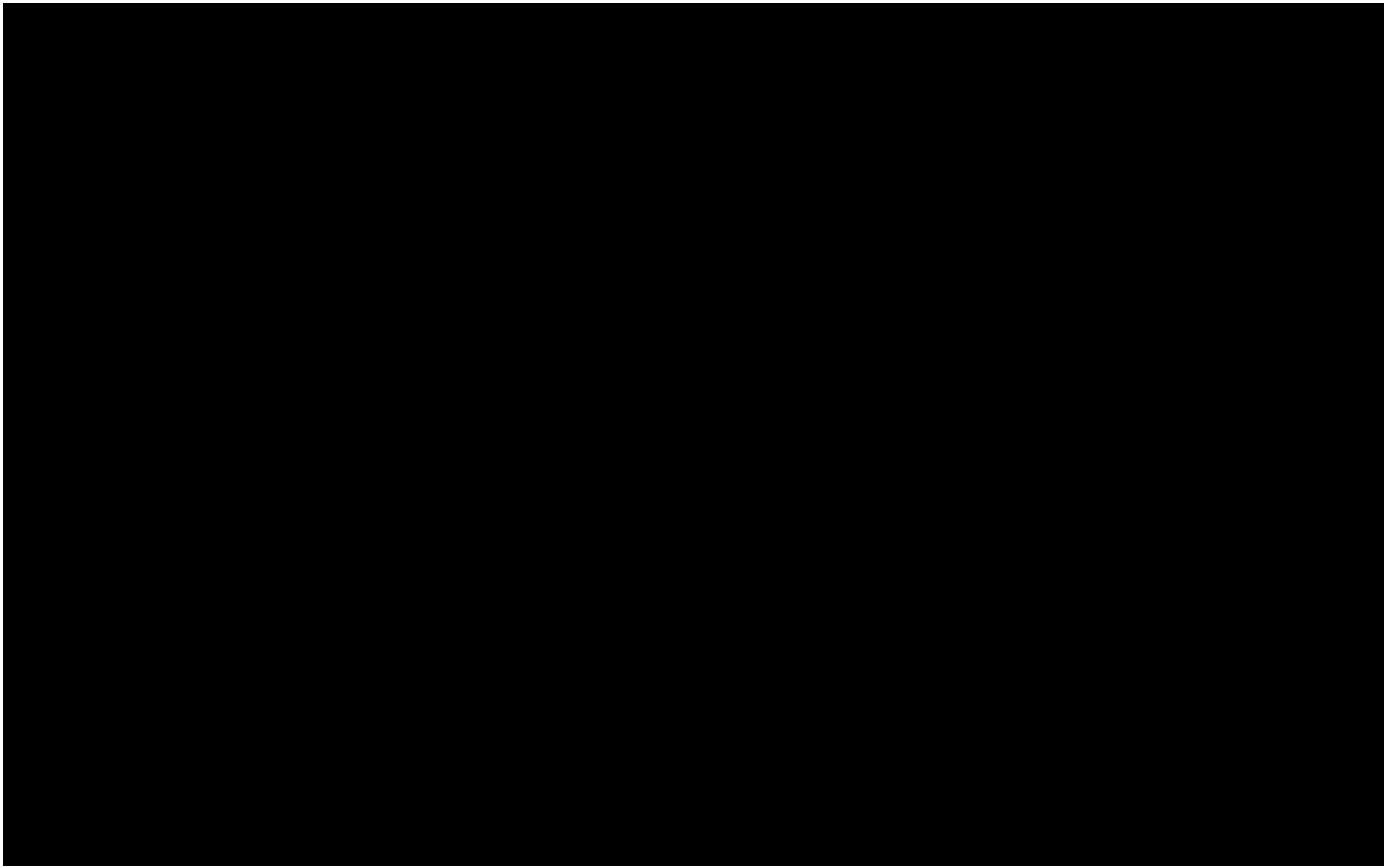
Application groups

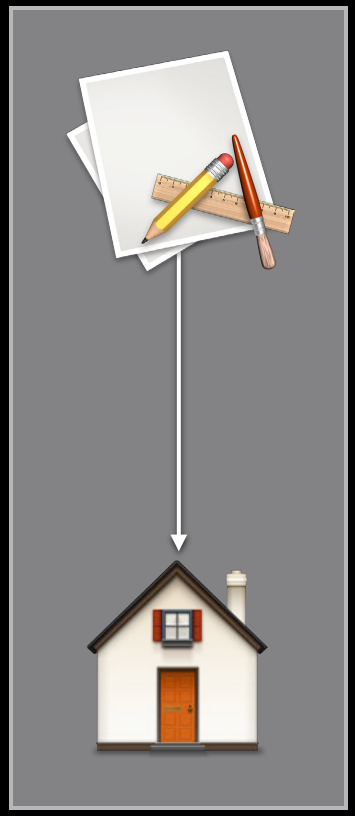
Application Groups

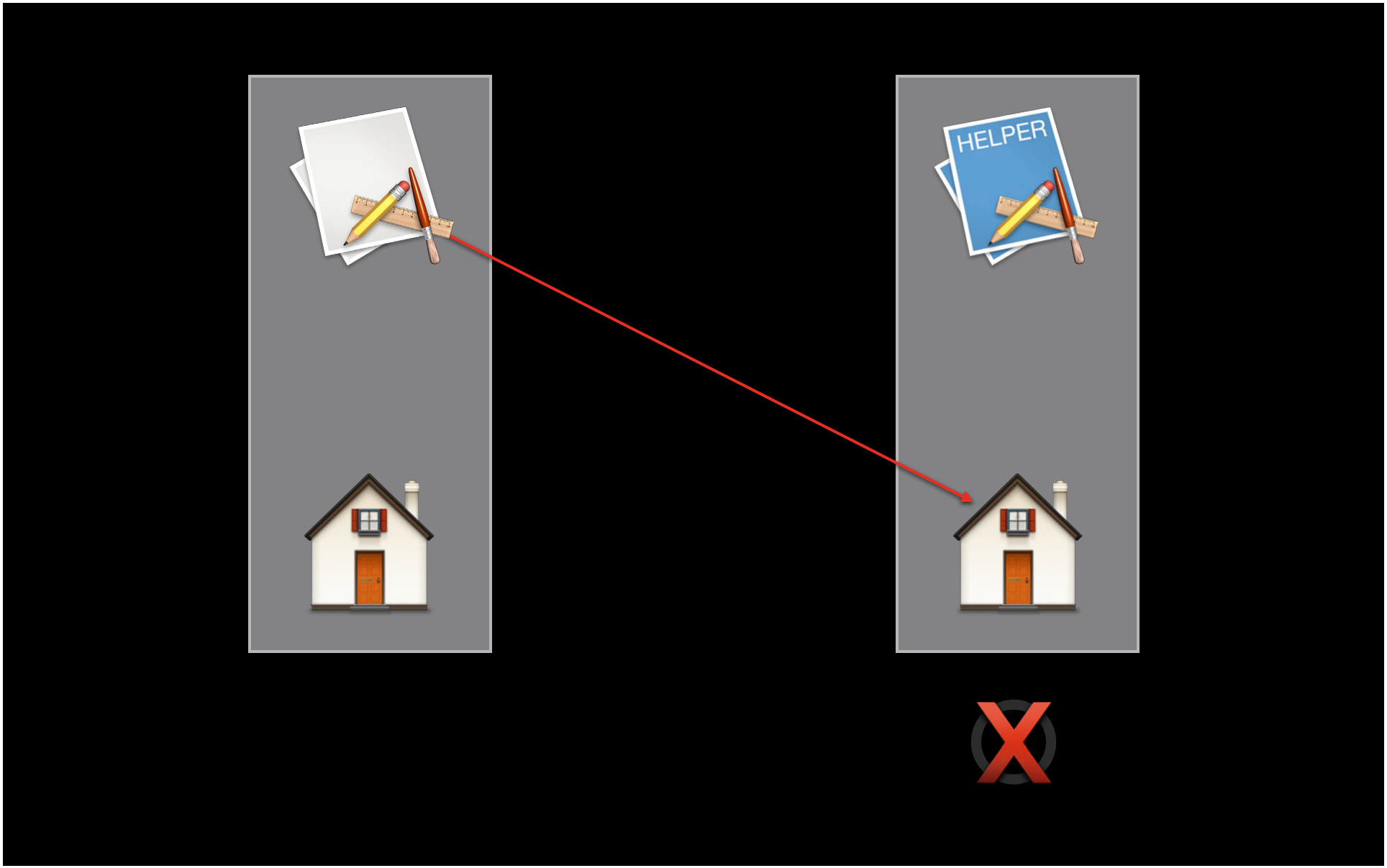
- Groups of apps from the same developer sometimes need more direct communication
- IPC, file sharing
- App Sandbox now offers a special affordance for this scenario

Application Groups

- `com.apple.security.application-groups`
- Each group name must begin with Apple-assigned Team ID
- Useful for suites of different apps, or a single app and its helper(s)
- Direct IPC permitted: XPC, POSIX
- Each group is assigned a shared file system location






HELPER


HELPER


Certificate
Entitlement
HELPER
com.apple.security.application-groups
8314ABCD.myapp


Certificate
Entitlement
HELPER


Certificate


HELPER

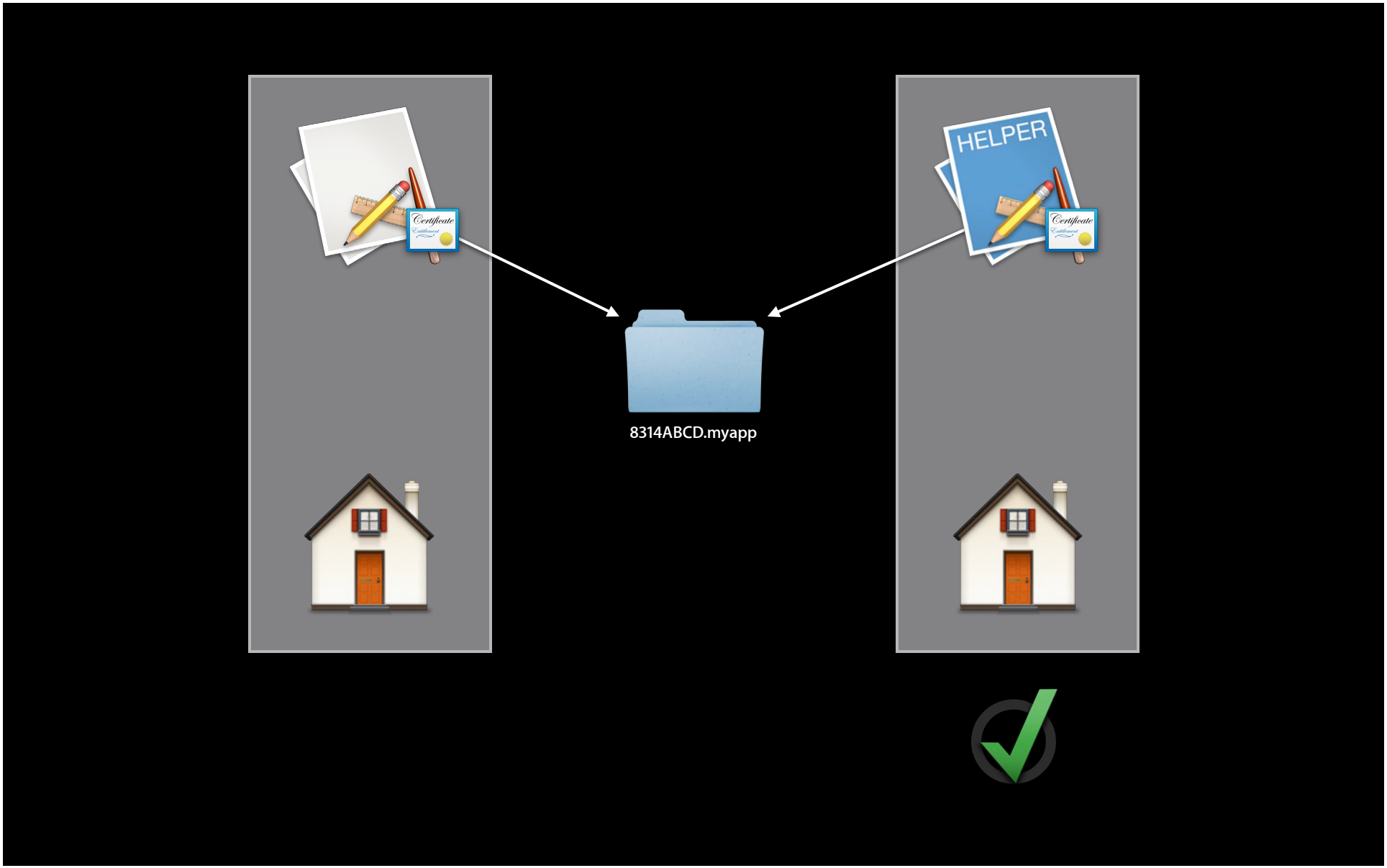

Certificate


HELPER
Certificate


Certificate

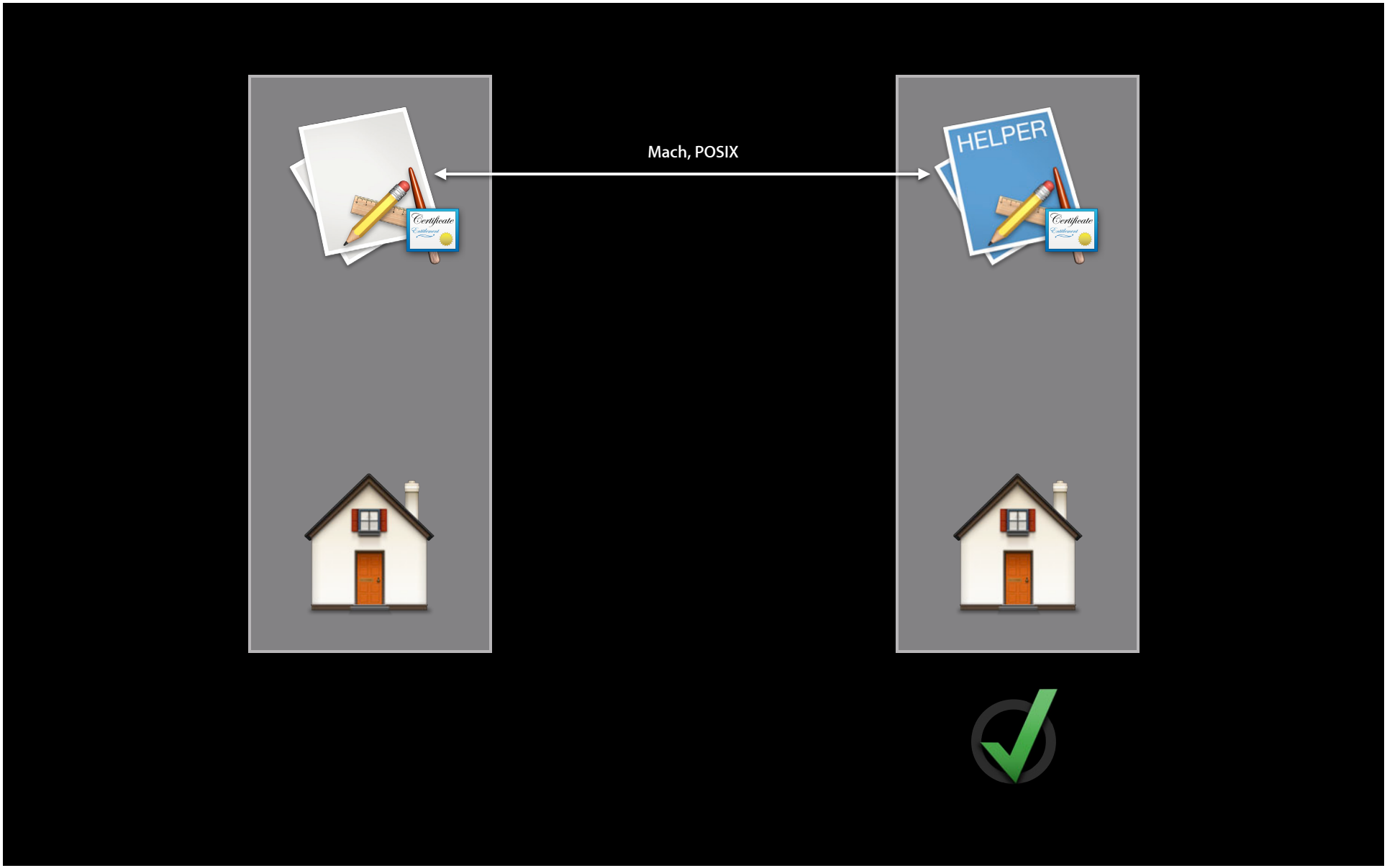

HELPER
Certificate


HELPER
Certificate
8314ABCD.myapp

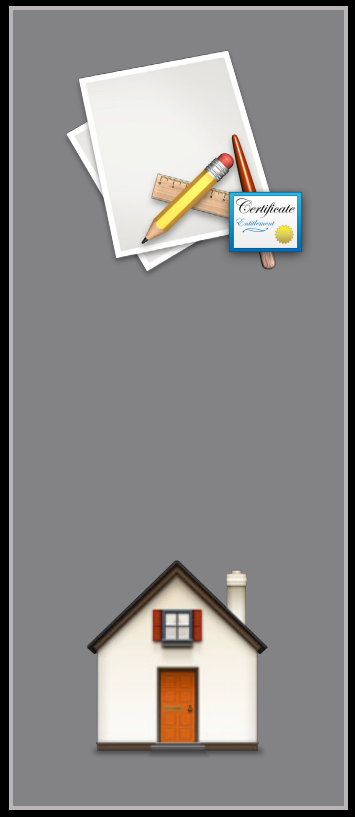

HELPER
Certificate
8314ABCD.myapp

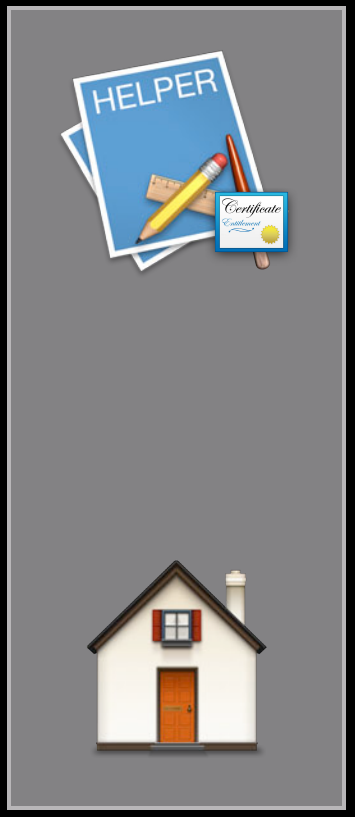

Certificate

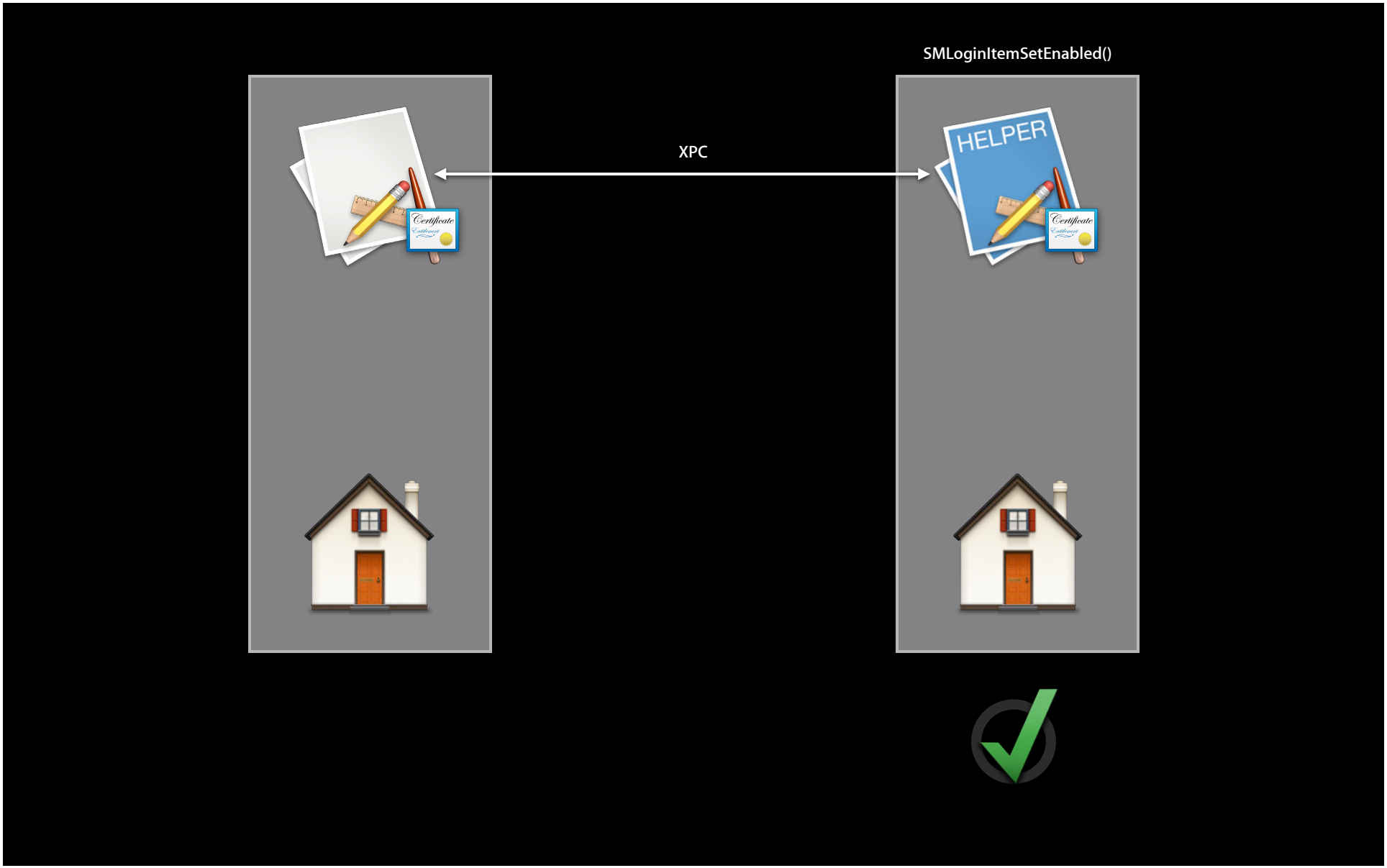

HELPER
Certificate


Mach, POSIX
HELPER
Certificate
Certificate

SMLoginItemSetEnabled()






SMLoginItemSetEnabled()
XPC
HELPER

New Since Lion

Related items

Related Items

- Access to files/folders with same name, but different file extension
  - Movie player opening a subtitle file for a movie
  - TextEdit upgrading a .rtf document to a .rtfd for attachments
- NSFilePresenter's `primaryPresentedItemURL` for the former, `itemAtURL:willMoveToURL:` for the latter
- Requires a declaration of allowed patterns in the app's Info.plist

New Since Lion

Automation

Automation

- Rich history of automation on OS X
- App Sandbox does not impose restrictions on how your apps can be scripted
- But your apps were very limited in how they can script other apps
- Two new mechanisms in Mountain Lion aim to safely support most common scripting scenarios

Background

- Apple events can escape App Sandbox
  - Use Finder to escape file system restrictions
  - Use Safari to escape network restrictions
  - Use Terminal to escape everything!
- Therefore, no Apple event sending by default
- Had to use a temporary exception entitlement

Apple Events Entitlement

Lion: Sending events to Mail

```
<key>com.apple.security.temporary-exception.apple-events<key>
<array>
    <string>com.apple.mail<string>
</array>
```

Apple Events Entitlement

Lion: Sending events to Mail

```
<key>com.apple.security.temporary-exception.apple-events<key>
<array>
    <string>com.apple.mail<string>
</array>
```

Apple Events Entitlement

Lion: Sending events to Mail

```
<key>com.apple.security.temporary-exception.apple-events<key>
<array>
    <string>com.apple.mail<string>
</array>
```

Apple Event Access Groups

New in Mountain Lion

- *Access groups* define groups of scriptable operations
  - Commands, classes, properties
  - Part of the application's scripting interface (sdef)
  - man 5 sdef
- Already in Mountain Lion applications
  - Mail: com.apple.Mail.compose
  - iTunes: com.apple.iTunes.playback, com.apple.iTunes.library.read, com.apple.iTunes.library.read-write

Defining an Access Group

Compose Mail message

```
<class-extension name="application">
    <element name="outgoing message"/>
</class>



<class name="outgoing message">
    ...
<class>


<command name="send">
    <direct-parameter type="outgoing message"/>
</command>
```

Defining an Access Group

Compose Mail message

```
<class-extension name="application">
    <element name="outgoing message"/>
        <access-group identifier="com.apple.Mail.compose" access="rw"/>
    </element>
</class>

<class name="outgoing message">
    ...
<class>


<command name="send">
    <direct-parameter type="outgoing message"/>
</command>
```

Defining an Access Group

Compose Mail message

```
<class-extension name="application">
    <element name="outgoing message"/>
        <access-group identifier="com.apple.Mail.compose" access="rw"/>
    </element>
</class>

<class name="outgoing message">
    <access-group identifier="com.apple.Mail.compose" access="rw"/>
    ...
<class>

<command name="send">
    <direct-parameter type="outgoing message"/>
</command>
```

Defining an Access Group

Compose Mail message

```
<class-extension name="application">
    <element name="outgoing message"/>
        <access-group identifier="com.apple.Mail.compose" access="rw"/>
    </element>
</class>

<class name="outgoing message">
    <access-group identifier="com.apple.Mail.compose" access="rw"/>
    ...
<class>

<command name="send">
    <!-- Not part of any access group.  No sending for you! -->
    <direct-parameter type="outgoing message"/>
</command>
```

Using an Access Group

- New entitlement
  - com.apple.security.scripting-targets
- Value is a dictionary
  - Keys are application code signing identifiers
  - Values are access group identifiers

Using an Access Group

Compose Mail message

```
<key>com.apple.security.scripting-targets<key>
<dict>
    <key>com.apple.Mail</key>
    <array>
        <string>com.apple.Mail.compose<string>
    </array>
</dict>
```

Using an Access Group

Compose Mail message

```
<key>com.apple.security.scripting-targets<key>
<dict>
    <key>com.apple.Mail</key>
    <array>
        <string>com.apple.Mail.compose<string>
    </array>
</dict>
```

Using an Access Group

Compose Mail message

```
<key>com.apple.security.scripting-targets<key>
<dict>
    <key>com.apple.Mail</key>
    <array>
        <string>com.apple.Mail.compose<string>
    </array>
</dict>
```

Using an Access Group

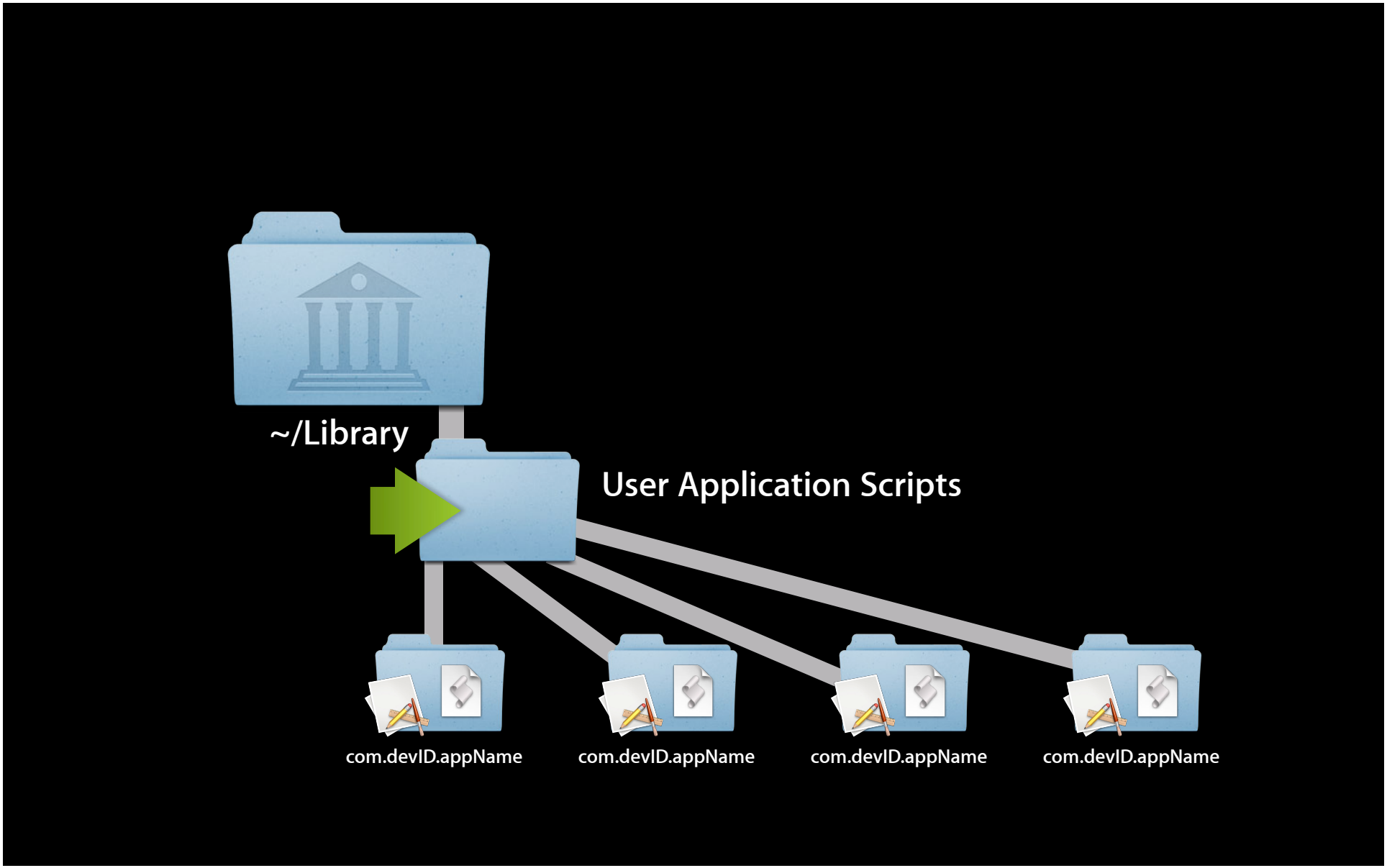
Compose Mail message

```
<key>com.apple.security.scripting-targets<key>
<dict>
    <key>com.apple.Mail</key>
    <array>
        <string>com.apple.Mail.compose<string>
    </array>
</dict>
```

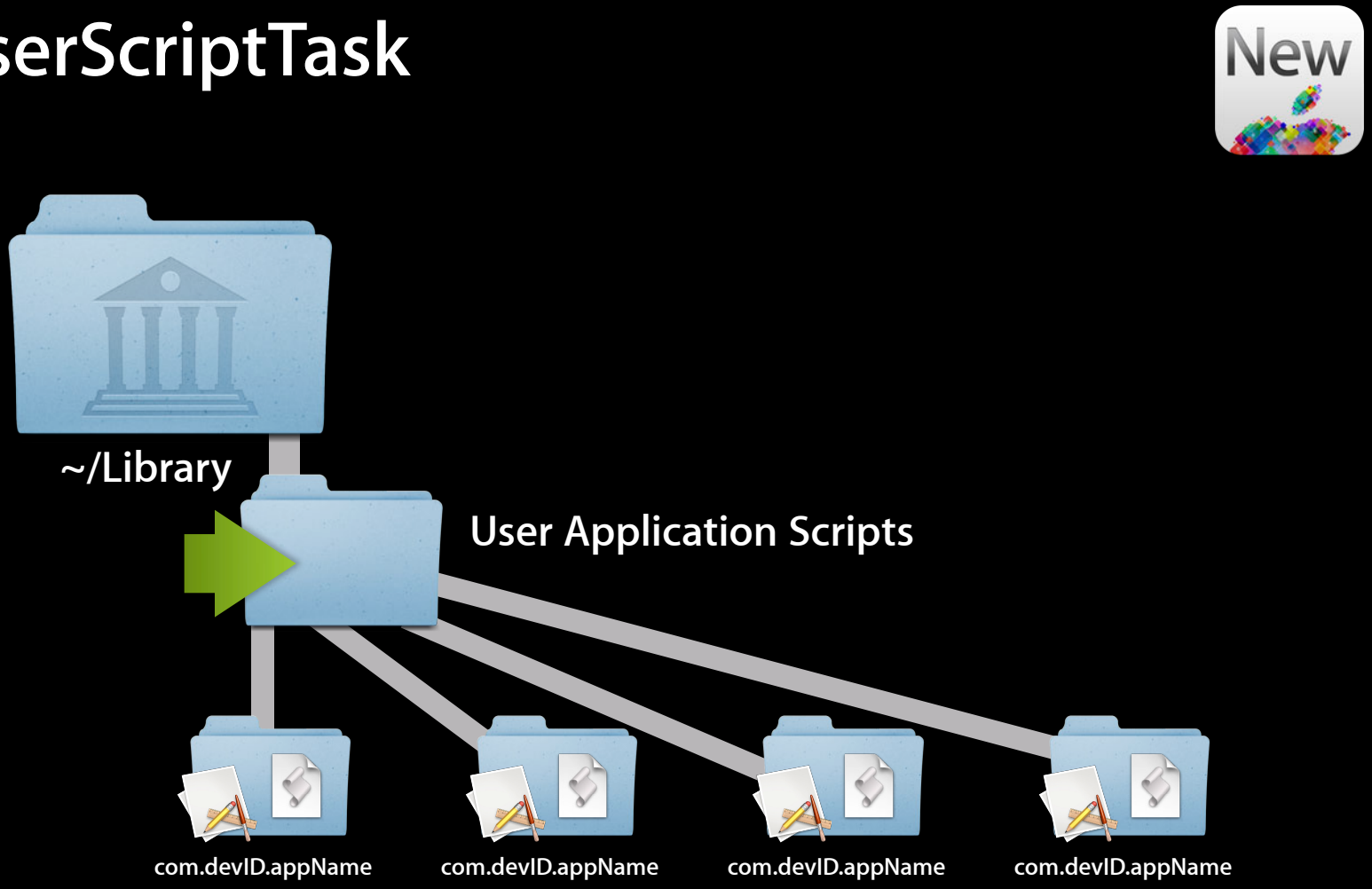
Application-Run User Scripts

New in Mountain Lion

- Application Script Menu
- Event Handlers
  - Mail Rule
  - Aperture Import Action
  - Messages Events
- Scripts executed by the application
- Inherit application's permissions


~/Library
User Application Scripts
com.devID.appName
com.devID.appName
com.devID.appName
com.devID.appName

NSUserScriptTask



NSUserScriptTask

Running attached user scripts

- Part of Foundation.framework
- NSUserScriptTask for generic scripts
  - Supports AppleScript, Automator, and UNIX scripts
- Subclasses for specific control
  - NSUserAppleScriptTask, NSUserAutomatorTask, NSUserUnixTask
- Script runs outside the sandbox
- No entitlement required

Summary

App Sandbox

- Strong barrier against exploitation and coding errors
- Drives policy by user intent
- Complementary to Gatekeeper
- See the *App Sandbox Design Guide*
- Sample code available

Summary

- iOS Sandbox: 30 billion app downloads with confidence
- Delight users with carefree apps on OS X

Related Sessions

Gatekeeper and Developer ID	Nob Hill Tuesday 11:30AM
Secure Automation Techniques in OS X	Russian Hill Tuesday 3:15PM
Asynchronous Design Patterns with Blocks, GCD, and XPC	Pacific Heights Friday 9:00AM

Related Labs

Security Lab	Core OS Lab B Tuesday 3:15PM
Sandboxing Audio Lab	Graphics, Media & Games Lab D Wednesday 2:00PM
Open and Save Panels Within an App Sandbox Q&A Lab	Essentials Lab B Wednesday 4:30PM
Security Lab	Core OS Lab B Thursday 9:00AM
Cocoa and XPC Lab	Essentials Lab A Friday 10:15AM
Core OS Open Hours	Core OS Lab A, B Friday 2:00PM


WWDC2012