iOS 8 Share Extension

Extension

Why Extension?

iOS 8 Share Extensions

Rene Ritchie:

> With iOS 8 and Extensibility, gone are the days when Apple had to make a deal with social networks and laboriously integrate them one and a time into iOS. Now, any app you download from the App Store can hook into the Share Sheets and give you the option to share or upload your content with other members and to the service.
>
> [...]
>
> Because of custom sharing extensions, you also get the ability to customize your sharing options. Scroll all the way to the right on a Share Sheet and you'll see a special "More" icon. Tap it and you're taken to the Activities panel where you can toggle on or off all the sharing options (with the exception of Messages and Mail), and move all of them around into any order you like.
>
> [...]
>
> System defaults are both quick to implement and offer a lot of functionality, like image preview, text entry, audience picker, etc. "for free". They also help maintain continuity of experience. Custom sheets are more work but can leverage code from the existing app and better show off a service's branding. That can be useful in continuously, visually reminding someone which service they're sharing to throughout the process.
>
> [...]
>
> Both web URL and web pages are supported. The first is all about sharing a link. The second is all about pulling the data from the web page itself. Developers can, via JavaScript, determine which parts of a webpage their extension wants.

• Extensions • iOS • iOS 8

Why Extension?

- It's all about users experience
- Drive more app downloads
- More convenient to use the app
- Adopt iOS 8
- Increase user engagement, returns rate
- WatchKit!

Introduction

- Starting in iOS 8.0 and OS X v10.10
- App extension lets you *extend custom functionality and content*
- Different contexts.
- An app extension is different from an app.

[from App Extension Programming Guide]

Extension Point

- A **system area** that supports extensions is called the **extension point**.
- Each extension point defines **usage policies** and **provides APIs** that you **use** when you create an extension for that area.
- You choose an extension point to use based on the functionality you want to provide.

Extension Point

Table 1-1 Extension points in iOS and OS X

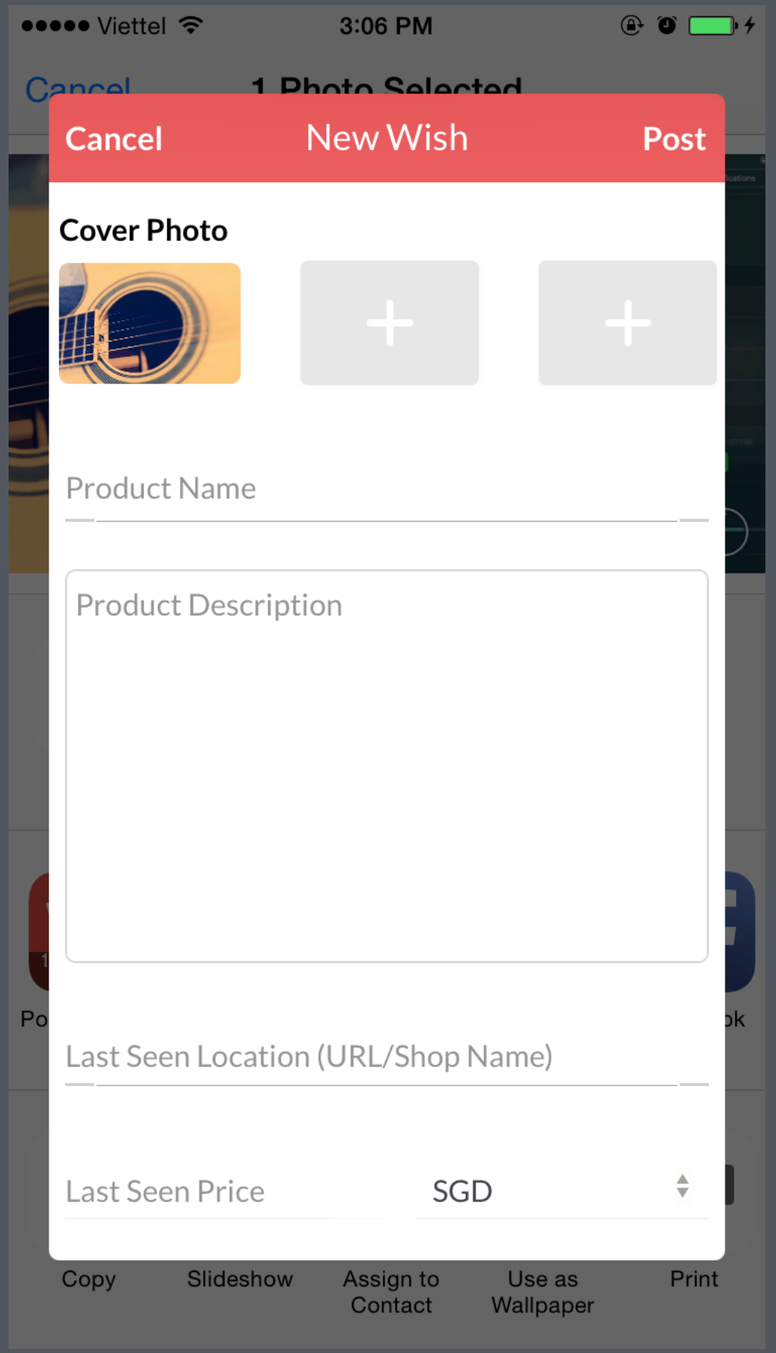
Extension point	Typical app extension functionality
Today (iOS and OS X)	Get a quick update or perform a quick task in the Today view of Notification Center (A Today extension is called a **widget**)
Share (iOS and OS X)	Post to a sharing website or share content with others
Action (iOS and OS X; UI and non-UI variants)	Manipulate or view content originating in a host app
Photo Editing (iOS)	Edit a photo or video within the Photos app
Finder Sync (OS X)	Present information about file sync state directly in Finder.
Document Provider (iOS; UI and non-UI variants)	Provide access to and manage a repository of files.
Custom Keyboard (iOS)	Replace the iOS system keyboard with a custom keyboard for use in all apps

Share extension

Share extensions give users a convenient way to share content with other entities, such as social sharing websites or upload services.

— *Share extensions*

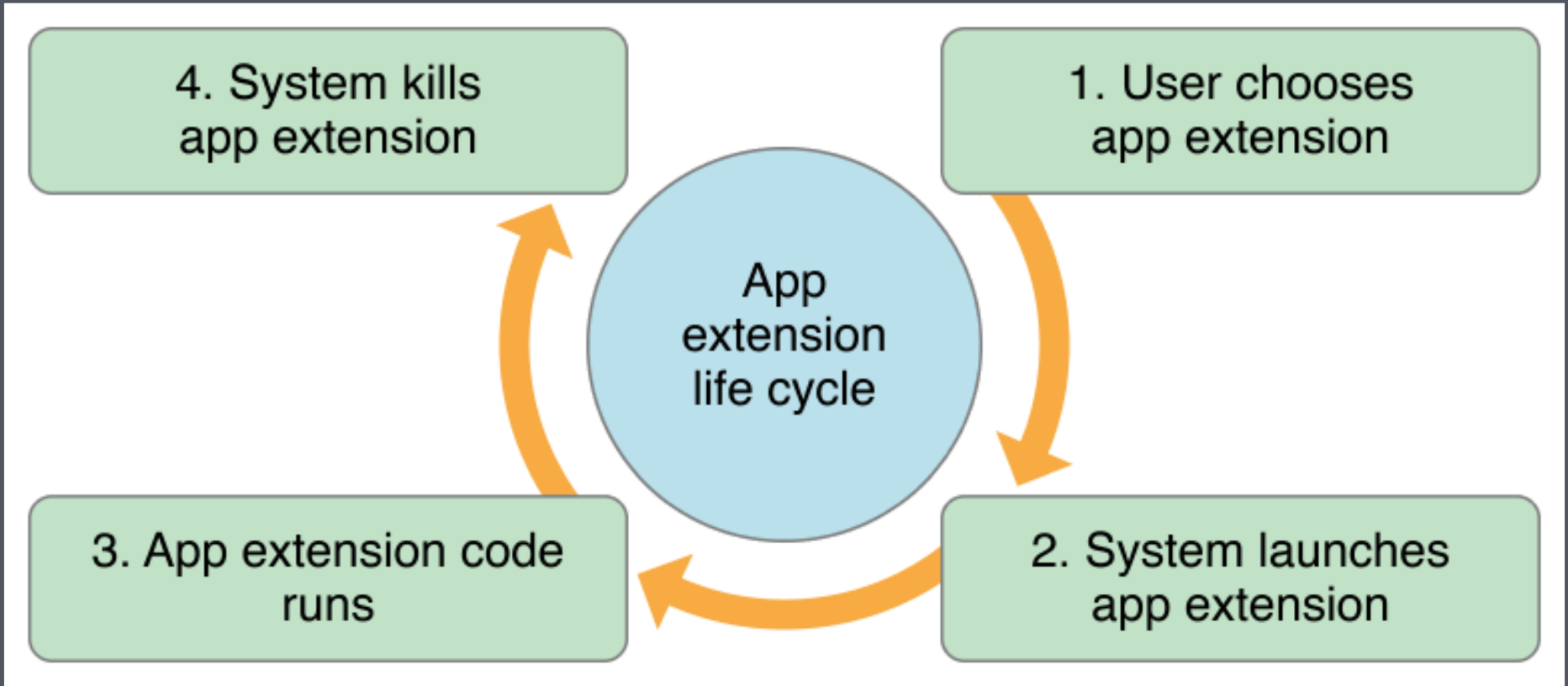
Available on both iOS and OS X



How It Works

An App Extension's Life Cycle

The basic life cycle of an app extension

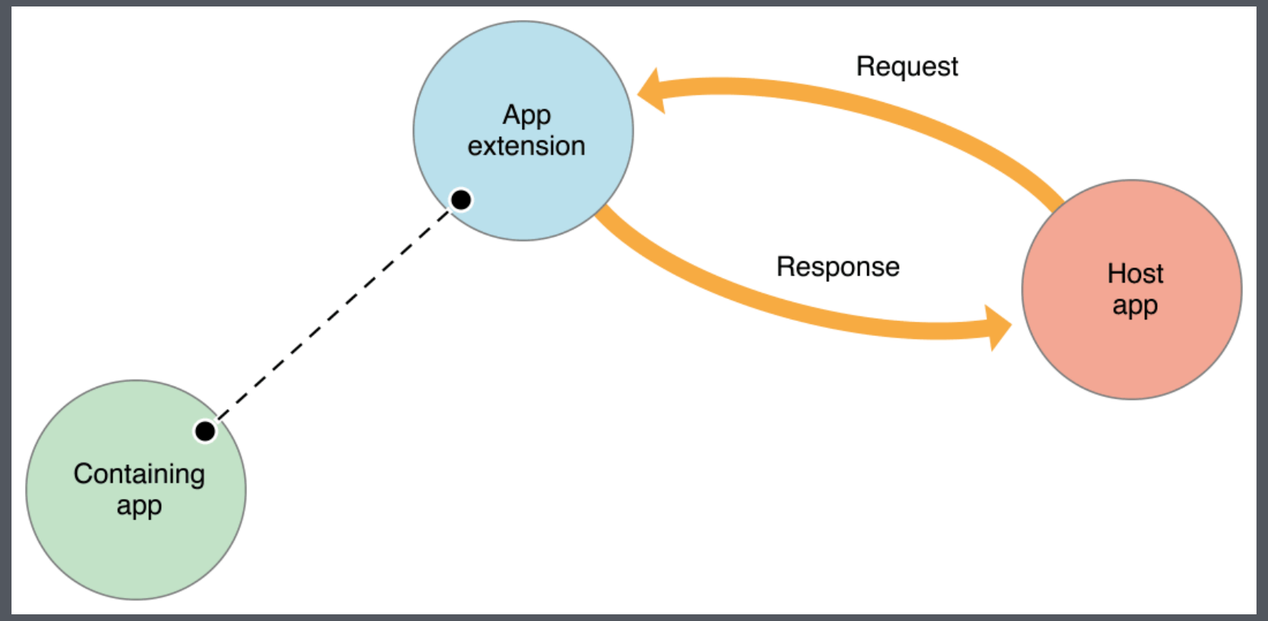


How an App Extension Communicates

An app extension communicates primarily with its host app, and does so in terms reminiscent of transaction processing: There is a request from the host and a response from the extension.

How an App Extension Communicates

Following figure shows a simplified view of the relationship between a running extension, the host app that launched it, and the containing app.



Differences

- Host app and Container app:
  - Host app is the app that runs your extensions
  - Container app is the app contains one of more of your extensions.
- Your container app, like the word phases out, contains and deliver your extension binary.
- The extension created by adding a **new target**. So as with any target, it specifies settings and files that combine to build a

Creating an extension

Xcode and the documents help you create and deliver App Extensions

Demo Creating an Extension from Xcode

Use Xcode's default template, it provides the extensions point-specific implementation files and settings, and produces a seperate binary that gets added to your containing app's bundle.

Understanding Share Extensions

On both platforms, a Share extension should:

- Make it easy for users to post content
- Let users preview, edit, annotate, and configure content, if appropriate
- Validate the user's content before sending it (use with or without system provided compose view controller)

Capability

If you want to share keychain access between the container app and extension. Check keychain Sharing option in capability tab.

Info.plist

Info.plist file define extension point and other target configuration

```
<key>NSExtension</key>
    <dict>
        <key>NSExtensionMainStoryboard</key>
        <string>MainInterface</string>
        <key>NSExtensionPointIdentifier</key>
        <string>com.apple.share-services</string>
</dict>
```

Declaring Supported Data Types for a Share or Action Extension

Context

Extension Context

Starting from iOS 8, in every view controller, there is a extensionContext property to get the `NSExtensionContext` object that contains the user's initial text and any attachments for a post, such as links, images, or videos.

NSExtensionContext

```
//
//  UIViewController.h
//  UIKit
//
//  Copyright (c) 2007-2014 Apple Inc. All rights reserved.
//

@class NSExtensionContext;

@interface UIViewController(NSExtensionAdditions) <NSExtensionRequestHandling>

// Returns the extension context. Also acts as a convenience method
// for a view controller to check if it participating in an extension request.
@property (nonatomic,readonly,retain) NSExtensionContext *extensionContext NS_AVAILABLE_IOS(8_0);

@end
```

Extension Context

The extension context object also contains information about the status of the posting operation.

(To learn more about how an extension can interact with its context, see Respond to the Host App's Request)

Controller

Xcode template default provide the SLComposeServiceViewController object includes a text view that displays the user's editable text content.

Controller

When a user choose Post, a Share extension validates the text view's content (in addition to attachments, if any) and calls the `completeRequestReturningItems:expirationHandler:completion:` method of `NSExtensionContext`, using code like the following:

```
NSExtensionItem *outputItem = [[NSExtensionItem alloc] init];
// Set the appropriate value in outputItem
NSArray *outputItems = @[outputItem];
[self.extensionContext completeRequestReturningItems:outputItems expirationHandler:nil completion:nil];
```

Posting Content

The primary purpose of a Share extension is to help users post content.

```
- (void)didSelectPost {
    // Perform the post operation.
    // When the operation is complete (probably asynchronously), the Share extension
    // should notify the success or failure, as well as the items that were actually shared.

    NSExtensionItem *inputItem = self.extensionContext.inputItems.firstObject;

    NSExtensionItem *outputItem = [inputItem copy];
    outputItem.attributedContentText = [[NSAttributedString alloc] initWithString:self.contentText attributes:nil];
    // Complete this implementation by setting the appropriate value on the output item.

    NSArray *outputItems = @[outputItem];

    [self.extensionContext completeRequestReturningItems:outputItems expirationHandler:nil completion:nil];
    // Or call [super didSelectPost] to use the superclass's default completion behavior.
}
```

Demo from  Wusic app

Somethings to Note

Some APIs Are Unavailable to App Extensions

Because of its focused role in the system, an app extension is ineligible to participate in certain activities. An app extension cannot:

- Access a sharedApplication object, and so cannot use any of the methods on that object
- Use any API marked in header files with the NS*EXTENSION*UNAVAILABLE macro, or similar unavailability macro, or any API in an unavailable framework

Some APIs Are Unavailable to App Extensions

- For example, in iOS 8.0, the HealthKit framework and EventKit UI framework are unavailable to app extensions.
- Access the camera or microphone on an iOS device
- Perform long-running background tasks

Some APIs Are Unavailable to App Extensions

- The specifics of this limitation vary by platform, as described in the extension point chapters in this document. (An app extension can initiate uploads or downloads using an NSURLSession object, with results of those operations reported to the containing app.)
- Receive data using AirDrop
  (An app extension can send data using AirDrop in the same way an app does: by employing the UIActivityViewController class.)

Optimize Efficiency and Performance

- App extensions should feel nimble and lightweight to users.
- Memory limits for running app extensions are significantly lower than the memory limits imposed on a foreground app.
- Your app extension doesn’t own the main run loop, so it’s crucial that you follow the established rules for good behavior in main run loops
- Keep in mind that the GPU is a shared resource in the system.

Summary

- Creating extension although not an easy task, but fun and you will learn a lot
- APIs still new and (unfortunately) buggy
- Beware library architecture
- UI and auto layout
- Keychain access group (if want to share keychain between container app and the extension)

Resources

Official

+ App Extension Programming Guide *****
+ iOS Human Interface Guidelines

Other engineering blogs, screencast:

+ Tumblr's What we learned building the Tumblr iOS share extension
+ NSSreencast part 1, part 2
+ WWDC session 205, session 217
+ Shinobi's iOS 8 day by day

You can view this slide again on Speaker Deck slide

 Thanks 😄!