Adopting Handoff on iOS and OS X

Session 219
Michael Jurewitz
Engineering

Vince Spader
Cocoa Frameworks Engineer

Keith Stattenfield
CoreFrameworks Engineer
What You Will Learn

What is Handoff
Adopting Handoff in your app
In-depth Handoff adoption
What is Handoff?
Adopting Handoff

Decide which activities to support in your app
Create activities in specific parts of your app
Handle continuing incoming activities in your app
To: Vince Spader & I more...
Cc: Boc.
From: jwatts@hr.com
Subject: Got a minute?

Let's chat about tomorrow's meeting.

Sent from my iPhone
Got a minute?

To: Vince Spader & 1 more...
Cc: Bac, From: jwreta@hme.com
Subject: Got a minute?

Let's chat about tomorrow's meeting.

Sent from my iPhone
Got a minute?

Subject: Got a minute?
Let's chat about tomorrow's meeting.

Sent from my iPhone
Additional Handoff Support

Streams between applications in two devices
Handoff between native app and website you own
Agenda

AppKit and UIKit support for adopting Handoff
Working with NSUserActivity directly
Native app to website Handoff
Using continuation streams between apps
Adopting Handoff in Your App

Vince Spader
Cocoa Frameworks Engineer
Adopting Handoff in Your App

AppKit/UIKit support for Handoff

Creating
Updating
Continuing
Creating User Activities
Creating User Activities

What do users do in your app?
Creating User Activities

What do users do in your app?
Creating User Activities
What do users do in your app?
Creating User Activities

What do users do in your app?

Reading messages
Picking an item from a list
Editing a document
Creating User Activities

What do users do in your app?
Creating User Activities

What do users do in your app?
Creating User Activities
Documents and Responders
Creating User Activities
Documents and Responders

NSDocument, UIDocument, NSResponder and UIResponder now have:
@property (strong) NSUserActivity *userActivity;
Creating User Activities
Documents and Responders

NSDocument, UIDocument, NSResponder and UIResponder now have:
@property (strong) NSUserActivity *userActivity;

You can set it like this:
NSUserActivity *userActivity = [[NSUserActivity alloc]
initWithActivityType:@"com.company.viewing-message"];
userActivity.title = @"Viewing Message";
document.userActivity = userActivity;
Creating User Activities
Document-based apps
Creating User Activities

Document-based apps

Add `NSUbiquitousDocumentUserActivityType` to each `CFBundleDocumentTypes` entry
Creating User Activities

Document-based apps
Creating User Activities

Document-based apps

We set `userActivity` automatically when the document is in iCloud
Creating User Activities

Document-based apps

We set `userActivity` automatically when the document is in iCloud
• On OS X, you can KVO
Creating User Activities

Other apps
Creating User Activities

Other apps

**NSUserActivityTypes** in Info.plist

```
<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Property List</td>
<td>Dictionary</td>
<td>(15 items)</td>
</tr>
<tr>
<td>Localization native development region</td>
<td>String</td>
<td>en</td>
</tr>
<tr>
<td>Executable file</td>
<td>String</td>
<td>${EXECUTABLE_NAME}</td>
</tr>
<tr>
<td>Bundle identifier</td>
<td>String</td>
<td>com.company.${PRODUCT_NAME:rfc1034:identifier}</td>
</tr>
<tr>
<td>InfoDictionary version</td>
<td>String</td>
<td>6.0</td>
</tr>
<tr>
<td>NSUserActivityTypes</td>
<td>Array</td>
<td>(2 items)</td>
</tr>
<tr>
<td>Item 0</td>
<td>String</td>
<td>com.company.viewing-message</td>
</tr>
<tr>
<td>Item 1</td>
<td>String</td>
<td>com.company.composing-message</td>
</tr>
<tr>
<td>Bundle name</td>
<td>String</td>
<td>${PRODUCT_NAME}</td>
</tr>
</tbody>
</table>
```
Creating User Activities
Documents and Responders
Creating User Activities
Documents and Responders

We manage it for you
Creating User Activities
Documents and Responders

We manage it for you

• We call `becomeCurrent`
Creating User Activities

NSUserActivity

becomeCurrent
Creating User Activities

NSUserActivity

becomeCurrent
Creating User Activities

becomeCurrent on iOS
Creating User Activities
becomeCurrent on iOS

When the app is launched, comes into the foreground, or tabs are switched:
Creating User Activities

becomeCurrent on iOS

When the app is launched, comes into the foreground, or tabs are switched:

• UIKit walks the view controller hierarchy
Creating User Activities

becomeCurrent on iOS

When the app is launched, comes into the foreground, or tabs are switched:

• UIKit walks the view controller hierarchy
  - Including presented view controllers
Creating User Activities

**becomeCurrent on iOS**

When the app is launched, comes into the foreground, or tabs are switched:

- UIKit walks the view controller hierarchy
  - Including presented view controllers
  - The view controller’s view must be in the view hierarchy
Creating User Activities
becomeCurrent on iOS

When the app is launched, comes into the foreground, or tabs are switched:
• UIKit walks the view controller hierarchy
  - Including presented view controllers
  - The view controller’s view must be in the view hierarchy

When userActivity is set:
Creating User Activities

`becomeCurrent` on iOS

When the app is launched, comes into the foreground, or tabs are switched:

• UIKit walks the view controller hierarchy
  - Including presented view controllers
  - The view controller’s view must be in the view hierarchy

When `userActivity` is set:

• If the view controller is in a transition, we wait until after it’s done
Creating User Activities

becomeCurrent on iOS

When the app is launched, comes into the foreground, or tabs are switched:

- UIKit walks the view controller hierarchy
  - Including presented view controllers
  - The view controller’s view must be in the view hierarchy

When userActivity is set:

- If the view controller is in a transition, we wait until after it’s done
- If the view controller’s view is in the window hierarchy
Creating User Activities

becomeCurrent on iOS
Creating User Activities

becomeCurrent on iOS

UIDocument will not becomeCurrent automatically.
Creating User Activities

becomeCurrent on iOS

UIDocument will not becomeCurrent automatically.

Share the userActivity:

```
[document openWithCompletionHandler:^(BOOL success) {
    viewController.userActivity = document.userActivity;
    ...
}];
```
Creating User Activities

becomeCurrent on OS X
Creating User Activities

`becomeCurrent` on OS X

AppKit looks for a `userActivity`: 
Creating User Activities

becomeCurrent on OS X

AppKit looks for a `userActivity`:

- Main window’s responder chain
Creating User Activities

`becomeCurrent` on OS X

AppKit looks for a `userActivity`:
- Main window’s responder chain
- Main window controller’s document
Creating User Activities

**becomeCurrent** on OS X

AppKit looks for a `userActivity`:

- Main window’s responder chain
- Main window controller’s document

We’ll reevaluate when appropriate
Creating User Activities

Documents and Responders
Creating User Activities
Documents and Responders

We manage it for you

• We call `becomeCurrent`
Creating User Activities
Documents and Responders

We manage it for you
• We call `becomeCurrent`
• We call `invalidate`
Creating User Activities

NSUserActivity

invalidate
Creating User Activities

NSUserActivity

invalidate
Updating User Activities
Updating User Activities
Documents and Responders

NSUserActivity has a **userInfo** dictionary
Updating User Activities
Documents and Responders

NSUserActivity has a **userInfo** dictionary

Override:
- (void)updateUserActivityState:(NSUserActivity *)userActivity
Updating User Activities
Documents and Responders

NSUserActivity has a `userInfo` dictionary

Override:
- `(void) updateUserActivityState:(NSUserActivity *)userActivity`

The `userInfo` is emptied each time
Updating User Activities

Documents and Responders
Updating User Activities
Documents and Responders

Something like this:

```objc
- (void)updateUserActivityState:(NSUserActivity *)userActivity {
    [super updateUserActivityState:userActivity];

    [userActivity addUserInfoEntriesFromDictionary:@{
        @"messageID": self.messageID,
    }];
}
```
Updating User Activities
Documents and Responders

Something like this:

```objective-c
- (void)updateUserActivityState:(NSUserActivity *)userActivity {
    [super updateUserActivityState:userActivity];

    [userActivity addUserInfoEntriesFromDictionary:@{ @“messageID”: self.messageID, }];
}
```

When your info is stale:
```objective-c
userActivity.needsSave = YES;
```
Updating User Activities

What to include
Updating User Activities

What to include

Can store NSArray, NSData, NSDate, NSDictionary, NSNull, NSNumber, NSSet, NSString, NSUUID, or NSURL
Updating User Activities

What to include

Can store NSArray, NSDdata, NSDate, NSDictionary, NSNull, NSNumber, NSSet, NSString, NSUUID, or NSURL

File URLs in iCloud or from a document provider are OK
Updating User Activities

What to include
Updating User Activities

What to include

Keep the minimal amount of information in the userInfo
Updating User Activities

What to include

Keep the minimal amount of information in the userInfo

• Just the state
Updating User Activities

What to include

Keep the minimal amount of information in the userInfo

- Just the state
- Avoid platform specifics
Updating User Activities

What to include

Keep the minimal amount of information in the userInfo

- Just the state
- Avoid platform specifics
- NS/UIDocument will add its `fileURL` with `NSUserActivityDocumentURLKey`
Updating User Activities

What to include
Updating User Activities

What to include

Think about versioning
Updating User Activities

What to include

Think about versioning

Maybe something like:

- (void)application:(NS/UIApplication *)application
didUpdateUserActivity:(NSUserActivity *)userActivity {
    [userActivity addUserInfoEntriesFromDictionary:@{
        @"handoffVersion": @"2.0",
    }];
}
Continuing User Activities
Continuing User Activity

App Delegate
Continuing User Activity

App Delegate

We start fetching it from the other device:

- `(BOOL)application:(NS/UIApplication *)application
  willContinueUserActivityWithTipo:(NSString *)activityType;`
Continuing User Activity

App Delegate

We start fetching it from the other device:

- `(BOOL)application:(NS/UIApplication *)application willContinueUserActivityWithOfType:(NSString *)activityType;`

Use this to show the user what’s being continued
Continuing User Activity

App Delegate
Continuing User Activity

App Delegate

- (BOOL)application:(NS/UIApplication *)application willContinueUserActivityWithType:(NSString *)activityType {
  if ([activityType isEqual:@"com.company.viewing-message"]) {
    id vc = [[MessageViewController alloc] init];
    vc.showLoadingIndicator = YES;
    [self showMessageViewController:vc];

    return YES;
  }

  return NO;
}
Continuing User Activity
App Delegate
Continuing User Activity

App Delegate

We got the activity:
- `(BOOL)application:(NS/UIApplication *)application continueUserActivity:(NSUserActivity *)userActivity restorationHandler: (void(^)(NSArray *restorableObjects))restorationHandler;`
Continuing User Activity

App Delegate

We got the activity:

– (BOOL)application:(NS/UIApplication *)application
    continueUserActivity:(NSUserActivity *)userActivity
    restorationHandler:
        (void(^)(NSArray *restorableObjects))restorationHandler;

Reconstruct the user’s activity
Continuing User Activity

App Delegate

We got the activity:

- `(BOOL)application:(NS/UIApplication *)application continueUserActivity:(NSUserActivity *)userActivity restorationHandler: (void(^)(NSArray *restorableObjects))restorationHandler;`

Reconstruct the user’s activity

Call the restorationHandler, passing it an array of documents or responders that present the user activity
Continuing User Activity
App Delegate
Continuing User Activity
App Delegate

Here's an example:

```objc
-(BOOL)application:continueUserActivity:restorationHandler: {
    NSString *activityType = activity.activityType;
    if ([activityType isEqual:@"com.company.viewing-message"]) {
        id vc = [[MessageViewController alloc] init];
        ...
        restorationHandler(@[vc]);
        return YES;
    }
    return NO;
}
```
Here’s an example:

```objective-c
-(BOOL)application:continueUserActivity:restorationHandler: {
    NSString *activityType = activity.activityType;
    if ([activityType isEqual:@"com.company.viewing-message"]) {
        id vc = [[MessageViewController alloc] init];
        ...
        restorationHandler(@[vc]);
    }
    return YES;
}
return NO;
```
Continuing User Activity
App Delegate
@implementation MessageViewController
...
-(void)restoreUserActivityState:(NSUserActivity *)activity {
    [super restoreUserActivityState:activity];
    [self setMessageID:activity.userInfo[@"messageID"]] ;
    ... 
    id cvc = [[ConversationViewController alloc] init];
    ... 
    [cvc restoreUserActivityState:activity];
}
...
@end
Continuing User Activity

App Delegate
@implementation ConversationViewController
...
-(void)restoreUserActivityState:(NSUserActivity *)activity {
    [super restoreUserActivityState:activity];

    NSString *version = activity.userInfo[@"handoffVersion"]
    BOOL isOldVersion = [self isOldVersion:version];

    NSString *recipientKey = isOldVersion ? @"to" : @"rcptID";
    self.recipient = activity.userInfo[recipientKey];
    [self updateRecipientImage];
}
...

Continuing User Activity
App Delegate
Continuing User Activity

App Delegate

If there was an error:

- (void)application:(NS/UIApplication *)application
didFailToContinueUserActivityWith.TypeString:activityType
error:(NSError *)error;
Continuing User Activity

App Delegate

If there was an error:
– (void)application:(NS/UIApplication *)application
didFailToContinueUserActivityWithTypeDef:activityType
error:(NSError *)error;

Can be **NSUserCancelledError**!
Continuing User Activity
Document-based app
Continuing User Activity
Document-based app

On iOS, you continue the user activity:
– (BOOL)application:continueUserActivity:restorationHandler: {
    ...
    NSURL *url = activity.userInfo[NSUserActivityDocumentURLKey];
    restorationHandler(@[doc]);
    return YES;
    ...
}
Continuing User Activity

Document-based app
Continuing User Activity

Document-based app

On OS X, AppKit can use NSDocumentController `restoreUserActivityState`: 
Continuing User Activity
Continuing User Activity

application:willContinueUserActivityWithTypet:
Continuing User Activity

application:willContinueUserActivityWithTypet:
Continuing User Activity

application:willContinueUserActivityWithTypet:
Continuing User Activity
Continuing User Activity

application:continueUserActivity:restorationHandler:
Continuing User Activity
Continuing User Activity

application:continueUserActivity:restorationHandler:
Continuing User Activity

application:continueUserActivity:restorationHandler:

restorationHandler([window, viewController])
Continuing User Activity

application:continueUserActivity:restorationHandler:

restorationHandler(@[window, viewController])

Window
restoreUserActivityState:
Continuing User Activity

application:continueUserActivity:restorationHandler:

restorationHandler([window, viewController])

Window
restoreUserActivityState:

View Controller
restoreUserActivityState:
Continuing User Activity
Continuing User Activity
NSUserActivity In-depth

Keith Stattenfield
CoreFrameworks Engineer
Non AppKit/UIKit uses

NSUserActivity
Non AppKit/UIKit uses

NSUserActivity

Your application creates an activity with an activity type string

```objective-c
[[NSUserActivity alloc] initWithActivityType:@"com.company.edit.foo"];
```
Activity Type Strings
NSUserActivity
Activity Type Strings

NSUserActivity

Applications which want to receive activities claim them in their Info.plist
Either in NSUserActivityTypes or in CFBundleDocumentTypes

<table>
<thead>
<tr>
<th>Bundle version</th>
<th>String</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ NSUserActivityTypes</td>
<td>Array</td>
<td>(3 items)</td>
</tr>
<tr>
<td>Item 0</td>
<td>String</td>
<td>com.company.edit.foo</td>
</tr>
<tr>
<td>Item 1</td>
<td>String</td>
<td>com.company.viewing.foo</td>
</tr>
<tr>
<td>Item 2</td>
<td>String</td>
<td>com.company.viewing.bar</td>
</tr>
<tr>
<td>Application requires iPhone environ...</td>
<td>Boolean</td>
<td>YES</td>
</tr>
</tbody>
</table>

| ▼ Document types | Array | (13 items) |
| ▼ Item 0 (NSRTFPboardType) | Dictionary | (8 items) |
| Icon File Name | String | rtf.icns |
| Document Type Name | String | NSRTFPboardType |
| Document Content Type UTIs | Array | (1 item) |
Activity Type Strings

NSUserActivity
Activity Type Strings

NSUserActivity

All applications from the same developer can exchange activities
Activity Type Strings

NSUserActivity

All applications from the same developer can exchange activities

Applications don’t have to claim the same activity types they create

Applications don’t have to claim any activity types, but can still create them
All applications from the same developer can exchange activities

Applications don’t have to claim the same activity types they create
Applications don’t have to claim any activity types, but can still create them

OS X
MP3  M4V
ePub
PNG
Activity Type Strings
NSUserActivity

All applications from the same developer can exchange activities

Applications don’t have to claim the same activity types they create
Applications don’t have to claim any activity types, but can still create them
Advanced NSUserActivity

Setting the activity information
Advanced NSUserActivity

Setting the activity information

activity.title = @“ … “
activity.userInfo = @{ … }
        [activity addUserInfoEntriesFromDictionary: @{ … }]
[activity becomeCurrent]
[activity invalidate]
Advanced NSUserActivity
NSUserActivityDelegate
Advanced NSUserActivity

NSUserActivityDelegate

activity.delegate = self;
...

activity.needsSave = YES;

Then, when the system needs information from your activity
- (void)userActivityWillSave:(NSUserActivity *)userActivity
Advanced NSUserActivity

NSUserActivityDelegate
Advanced NSUserActivity

NSUserActivityDelegate

When continued from another device:

- (void)userActivityWasContinued:(NSUserActivity *)userActivity

Called when this activity was successfully continued on another device
Most applications won’t need this at all
Website Handoff
Native application to web browser
Website Handoff
Native application to web browser
Website Handoff
Native application to web browser
Website Handoff
Native application to web browser

NSUserActivity* activity = [[NSUserActivity alloc] initWithActivityType:…];

activity.userInfo = @{ … }
activity.webpageURL = [NSURL URLWithString: …];
Website Handoff

Web browser to native application
Website Handoff
Web browser to native application
Website Handoff
Web browser to native application
Website Handoff
Web browser to native application

<table>
<thead>
<tr>
<th>▼ Root</th>
<th>Dictionary</th>
<th>(1 item)</th>
</tr>
</thead>
<tbody>
<tr>
<td>▼ com.apple.developer.associated-domains</td>
<td>Array</td>
<td>(2 items)</td>
</tr>
<tr>
<td>Item 0</td>
<td>String</td>
<td>AVeryLongSampleDomainName.com</td>
</tr>
<tr>
<td>Item 1</td>
<td>String</td>
<td>AVeryLongSampleDomainName.co.uk</td>
</tr>
</tbody>
</table>
Website Handoff

Web browser to native application

```objective-c
application:continueUserActivity:(NSUserActivity*)userActivity
restorationHandler:(void(^)(NSArray*
*restorableObjects))restorationHandler {
    if ([userActivity.activityType
        isEqual:NSUserActivityTypeContinuingFromWebBrowser]) {
        /* resume an activity based on the webpageURL */
        ...
    } else if ([userActivity isEqual:@“com.company.type12”]) {
        ...
    }
}
```
Continuation Streams

Need more than a one-way, one time exchange of data from creator to receiver

Establishes a bidirectional stream for some kind of interactive purposes
Continuation Streams

Need more than a one-way, one time exchange of data from creator to receiver.

Establishes a bidirectional stream for some kind of interactive purposes.
Continuation Streams

Need more than a one-way, one time exchange of data from creator to receiver

Establishes a bidirectional stream for some kind of interactive purposes
Continuation Streams

Need more than a one-way, one time exchange of data from creator to receiver

Establishes a bidirectional stream for some kind of interactive purposes
Continuation Streams

NSUserActivity

NSUserActivity* activity = [[NSUserActivity alloc] initWithActivityType:
@"com.company.interact"];

activity.userInfo = @{ … }

activity.delegate = self;

activity.supportsContinuationStreams = YES;

[activity becomeCurrent ];
Continuation Streams
NSUserActivity, on the receiving device

- application:(NS/UIApplication*) continueUserActivity:
  (NSUserActivity*)activity restorationHandler:...
{
    if (activity.supportsContinuationStreams) {
      [activity getContinuationStreamsWithCompletionHandler:
        ^(NSInputStream* inputstream, NSOutputStream*
          outputstream, NSError* error) {
          if (!error) {
            /* You can send and receive over these streams! */
          }
      }"}
ConnectBack

NSUserActivity, back on the initiating device

Lastly, this delegate method is called with the streams

```c
-(void) userActivity:(NSUserActivity *)userActivity
didReceiveInputStream:(NSInputStream *)inputStream outputStream:
(NSOutputStream *)outputStream {
    ...
}
```
So, you've learned

AppKit/UIKit support
NS/UIDocument support
Continuation streams
Website interoperability
More Information

Jake Behrens
Frameworks Evangelist
behrens@apple.com

Documentation
Handoff Programming Guide
http://apple.com

Apple Developer Forums
http://devforums.apple.com
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Documents</td>
<td>Marina</td>
<td>Thursday 11:30AM</td>
</tr>
<tr>
<td>Your App, Your Website, and Safari</td>
<td>Nob Hill</td>
<td>Tuesday 4:30PM</td>
</tr>
</tbody>
</table>
## Labs

<table>
<thead>
<tr>
<th>Lab</th>
<th>Frameworks Lab</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handoff Lab</td>
<td>B</td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td>Cocoa Touch Lab</td>
<td>A</td>
<td>Thursday 2:00PM</td>
</tr>
<tr>
<td>Cocoa Lab</td>
<td>B</td>
<td>Thursday 4:30PM</td>
</tr>
</tbody>
</table>