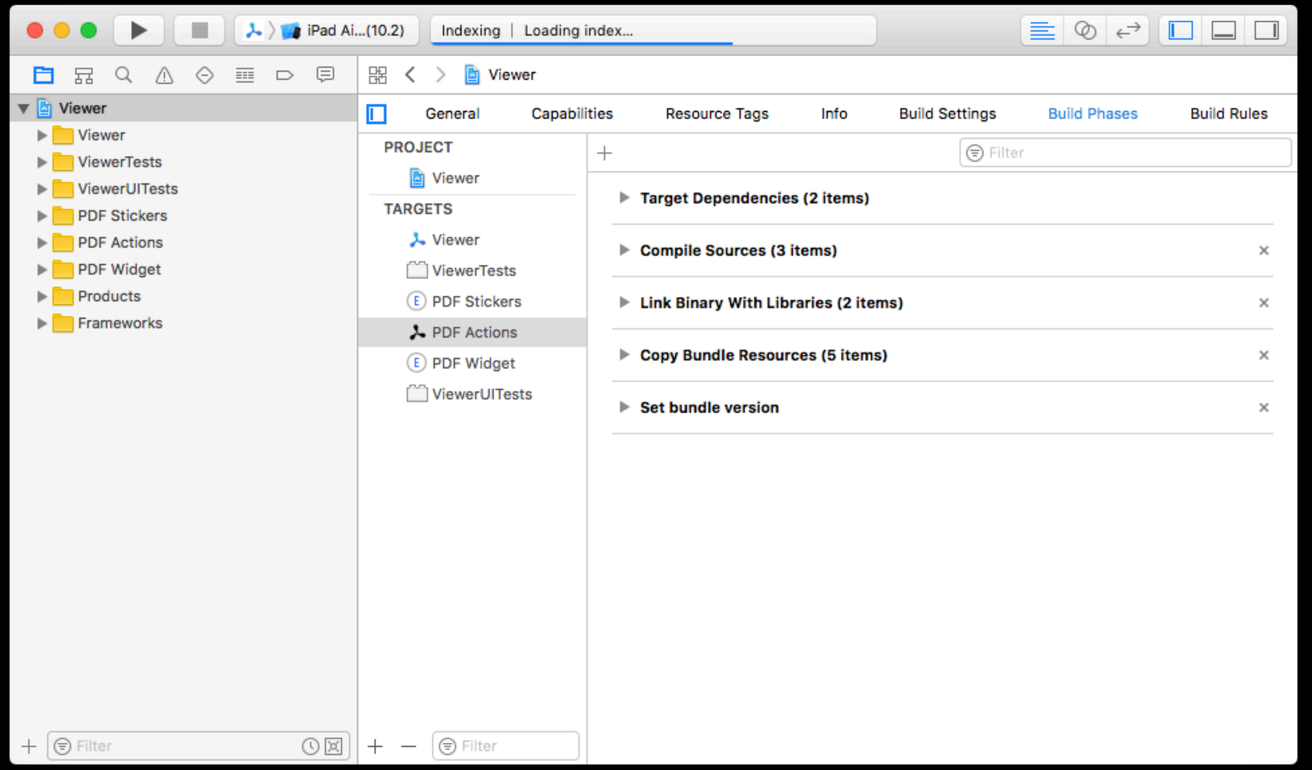
The unofficial guide to building Action Extensions

Extensions

- Set up as a separate target
- Out of process of the host app or the containing app
- Has its own sandbox
- Can share data with the containing app through App Groups
- Can share data with the host app only through XPC



Action Extensions

- Visible in the share sheet in the second row
- *“An Action extension helps users view or transform content originating in a host app.”* [1]
- On launch the extension gets the data to be shared from the host app through a `NSExtensionContext`

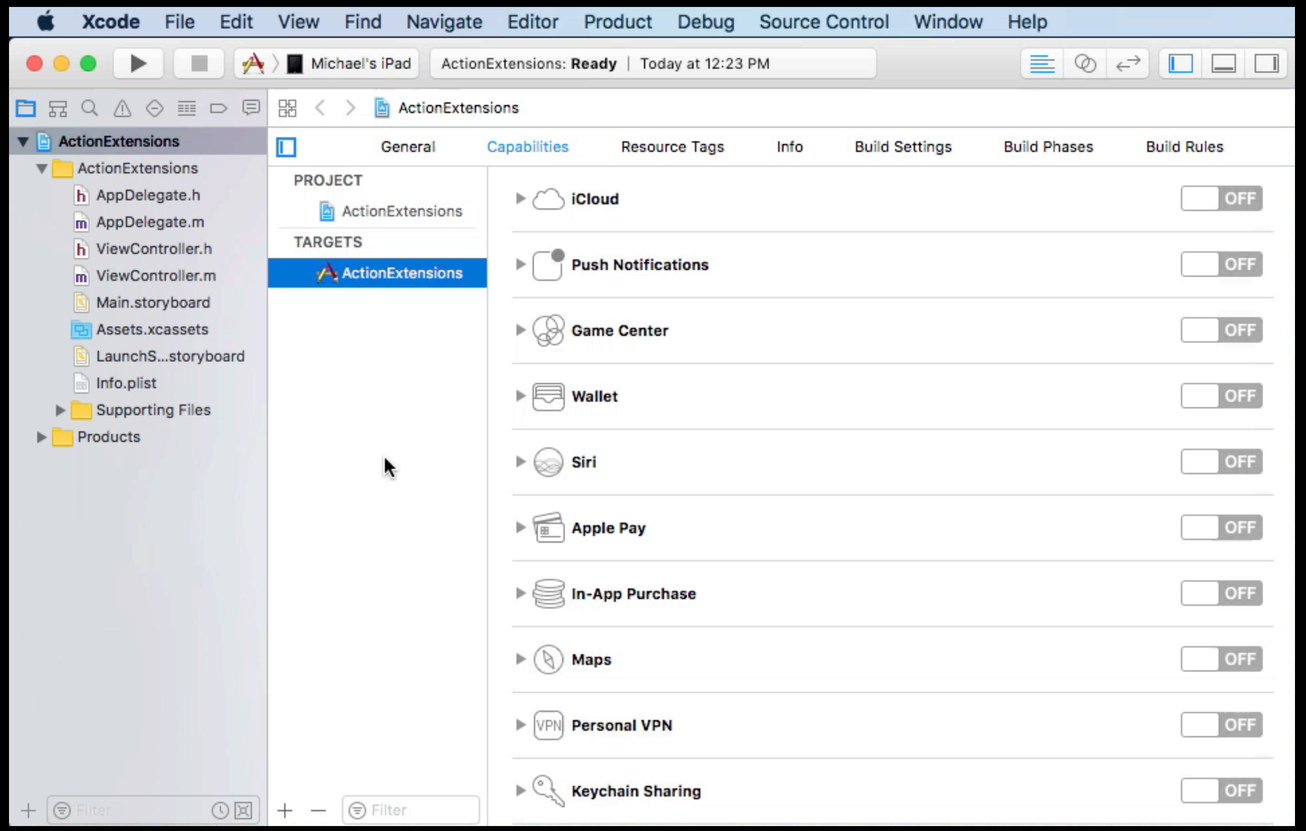
[1]: https://developer.apple.com/library/content/documentation/General/Conceptual/ExtensibilityPG/Action.html

Today's Talk

- How to implement an action extension in theory
- How to make that implementation work in practice

- I will not cover the host app's integration

The Theory


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Michael's iPad
ActionExtensions: Ready | Today at 12:23 PM
ActionExtensions
ActionExtensions
ActionExtensions
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchS...storyboard
Info.plist
Supporting Files
Products
General
Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
ActionExtensions
TARGETS
ActionExtensions
iCloud
OFF
Push Notifications
OFF
Game Center
OFF
Wallet
OFF
Siri
OFF
Apple Pay
OFF
In-App Purchase
OFF
Maps
OFF
Personal VPN
OFF
Keychain Sharing
OFF

The Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(UIImage *image, NSError *error) {
        if(image) {
          [[NSOperationQueue mainQueue] addOperationWithBlock:^{
            [imageView setImage:image];
          }];
        }
      }];
    }
  }
}
```

The Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(UIImage *image, NSError *error) {
        if(image) {
          [[NSOperationQueue mainQueue] addOperationWithBlock:^{
            [imageView setImage:image];
          }];
        }
      }];
    }
  }
}
```

The Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(UIImage *image, NSError *error) {
        if(image) {
          [[NSOperationQueue mainQueue] addOperationWithBlock:^{
            [imageView setImage:image];
          }];
        }
      }];
    }
  }
}
```

The Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(UIImage *image, NSError *error) {
        if(image) {
          [[NSOperationQueue mainQueue] addOperationWithBlock:^{
            [imageView setImage:image];
          }];
        }
      }];
    }
  }
}
```

The Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(UIImage *image, NSError *error) {
        if(image) {
          [[NSOperationQueue mainQueue] addOperationWithBlock:^{
            [imageView setImage:image];
          }];
        }
      }];
    }
  }
}
```

The Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(UIImage *image, NSError *error) {
        if(image) {
          [[NSOperationQueue mainQueue] addOperationWithBlock:^{
            [imageView setImage:image];
          }];
        }
      }];
    }
  }
}
```

The Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(UIImage *image, NSError *error) {
        if(image) {
          [[NSOperationQueue mainQueue] addOperationWithBlock:^{
            [imageView setImage:image];
          }];
        }
      }];
    }
  }
}
```

The Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(UIImage *image, NSError *error) {
        if(image) {
          [[NSOperationQueue mainQueue] addOperationWithBlock:^{
            [imageView setImage:image];
          }];
        }
      }];
    }
  }
}
```

Modifying the Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(NSData *data, NSError *error) {
        NSURL* targetURL = …;
        [data writeToURL:targetURL atomically:YES];
      }];
    }
  }
}
```

Modifying the Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(NSData *data, NSError *error) {
        NSURL* targetURL = …;
        [data writeToURL:targetURL atomically:YES];
      }];
    }
  }
}
```

Modifying the Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(NSData *data, NSError *error) {
        NSURL* targetURL = …;
        [data writeToURL:targetURL atomically:YES];
      }];
    }
  }
}
```

Modifying the Loop

```
for (NSExtensionItem *item in self.extensionContext.inputItems) {
  for (NSItemProvider *prov in item.attachments) {
    if ([prov hasItemConformingToTypeIdentifier:(NSString *)kUTTypeImage]){
      __weak UIImageView *imageView = self.imageView;
      [prov loadItemForTypeIdentifier:(NSString *)kUTTypeImage
                              options:nil
                    completionHandler:^(NSData *data, NSError *error) {
        NSURL* targetURL = …;
        [data writeToURL:targetURL atomically:YES];
      }];
    }
  }
}
```

Demo

Pitfalls

- Data might come in various different formats even for the same type identifier
- Some fileURLs seem to not be accessible from the sandbox
- Specify the loaded item as `id<NSSecureCoding>` and look at the data you get

Error Handling

- What can go wrong, will go wrong
- If you use Swift: Don’t do any force unwrapping or `fatalError()` calls
- If you use Objective-C: Be super careful with everything that is declared `nullable`

Error Handling

- Sometimes your extension might even get launched but does not receive any items at all
- Handle absolutely everything that could happen, from an API point of view, not from a documentation point of view
- You are dealing with random apps that could do random things

Radar Time

- Safari shares images as Data, other apps as URL (rdar://29924023)
- Message.app shares image file url not accessible from inside an action extension (rdar://29918507)
- Photos.app advertises live photo to action extensions which then can't be accessed (rdar://29924331)
- Messages.app advertises live photos to action extensions as public.image and is sharing a jpeg (rdar://29924679)
- Host apps interfere with the tint color of extensions (rdar://30141950)
- There needs to be a document describing the differences of NSExtensionItem, NSItemProvider and the items from an NSItemProvider (rdar://30184485)

Radar Time

- P.s.: Please file radars or dupe the ones you want to get fixed!

Radar Time

- P.s.: Please file radars or dupe the ones you want to get fixed!

Radar Time

- P.s.: Please **file radars** or dupe the ones you want to get fixed!

“Sometimes they say, well it’s a really obvious problem! I’m sure you have 12 copies of the bug […] should I still file a bug report? Yes, you should still file a bug report. Better have 5 copies of a bug than none at all. At Apple, if an issue is not tracked using a bug report, it essentially does not exist for us.”

–Tanya Gupta, Engineering Manager at Apple

Radar Time

- Sample projects help you to identify the problem
- Use OpenRadar if you can

Radar Time

- Safari shares images as Data, other apps as URL (rdar://29924023)
- Message.app shares image file url not accessible from inside an action extension (rdar://29918507)
- Photos.app advertises live photo to action extensions which then can't be accessed (rdar://29924331)
- Messages.app advertises live photos to action extensions as public.image and is sharing a jpeg (rdar://29924679)
- Host apps interfere with the tint color of extensions (rdar://30141950)
- There needs to be a document describing the differences of NSExtensionItem, NSItemProvider and the items from an NSItemProvider (rdar://30184485)

Radar Time

- Safari shares images as Data, other apps as URL (rdar://29924023)
- Message.app shares image file url not accessible from inside an action extension (rdar://29918507)
- Photos.app advertises live photo to action extensions which then can't be accessed (rdar://29924331)
- Messages.app advertises live photos to action extensions as public.image and is sharing a jpeg (rdar://29924679)
- Host apps interfere with the tint color of extensions (rdar://30141950)
- There needs to be a document describing the differences of NSExtensionItem, NSItemProvider and the items from an NSItemProvider (rdar://30184485)

Demo

UIAppearance

- Either ensure your extension can deal with every possible appearance
- Or set `NSExtensionOverridesHostUIAppearance` to `YES` in the `NSExtension` dictionary in the extension's `Info.plist`

UIAppearance

- Either ensure your extension can deal with every possible appearance
- Or set `NSExtensionOverridesHostUIAppearance` to `YES` in the `NSExtension` dictionary in the extension's `Info.plist`

Getting ready to ship

- Set the activation rule in you extension’s Info.plist
- `TRUEPREDICATE` is great for debugging

Getting ready to ship

- Set the activation rule in you extension’s Info.plist
- `TRUEPREDICATE` is great for debugging

Getting ready to ship

```
SUBQUERY (
 extensionItems,
 $extensionItem,
 SUBQUERY (
  $extensionItem.attachments,
  $attachment,
  (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "com.adobe.pdf" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.file-url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.jpeg" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.png" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.image") AND
  NOT (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO
                                                 "com.pspdfkit.viewer.blocker")
 ).@count == $extensionItem.attachments.@count
).@count >= 1
```

Getting ready to ship

```
SUBQUERY (
 extensionItems,
 $extensionItem,
 SUBQUERY (
  $extensionItem.attachments,
  $attachment,
  (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "com.adobe.pdf" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.file-url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.jpeg" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.png" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.image") AND
  NOT (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO
                                                 "com.pspdfkit.viewer.blocker")
 ).@count == $extensionItem.attachments.@count
).@count >= 1
```

Getting ready to ship

```
SUBQUERY (
 extensionItems,
 $extensionItem,
 SUBQUERY (
  $extensionItem.attachments,
  $attachment,
  (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "com.adobe.pdf" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.file-url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.jpeg" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.png" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.image") AND
  NOT (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO
                                                 "com.pspdfkit.viewer.blocker")
 ).@count == $extensionItem.attachments.@count
).@count >= 1
```

Getting ready to ship

```
SUBQUERY (
 extensionItems,
 $extensionItem,
 SUBQUERY (
  $extensionItem.attachments,
  $attachment,
  (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "com.adobe.pdf" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.file-url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.jpeg" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.png" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.image") AND
  NOT (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO
                                                 "com.pspdfkit.viewer.blocker")
 ).@count == $extensionItem.attachments.@count
).@count >= 1
```

Getting ready to ship

```
SUBQUERY (
 extensionItems,
 $extensionItem,
 SUBQUERY (
  $extensionItem.attachments,
  $attachment,
  (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "com.adobe.pdf" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.file-url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.jpeg" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.png" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.image") AND
  NOT (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO
                                                 "com.pspdfkit.viewer.blocker")
 ).@count == $extensionItem.attachments.@count
).@count >= 1
```

Getting ready to ship

```
SUBQUERY (
 extensionItems,
 $extensionItem,
 SUBQUERY (
  $extensionItem.attachments,
  $attachment,
  (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "com.adobe.pdf" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.file-url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.jpeg" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.png" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.image") AND
  NOT (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO
                                                 "com.pspdfkit.viewer.blocker")
 ).@count == $extensionItem.attachments.@count
).@count >= 1
```

Getting ready to ship

```
SUBQUERY (
 extensionItems,
 $extensionItem,
 SUBQUERY (
  $extensionItem.attachments,
  $attachment,
  (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "com.adobe.pdf" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.file-url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.url" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.jpeg" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.png" OR
   ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO "public.image") AND
  NOT (ANY $attachment.registeredTypeIdentifiers UTI-CONFORMS-TO
                                                 "com.pspdfkit.viewer.blocker")
 ).@count == $extensionItem.attachments.@count
).@count >= 1
```

Reading Material

- The Struggle with Action Extensions
  https://pspdfkit.com/blog/2017/action-extension/
- Hiding Your Action and Share Extensions In Your Own Apps
  https://pspdfkit.com/blog/2016/hiding-action-share-extensions-in-your-own-apps/
- Writing good bug reports
  https://pspdfkit.com/blog/2016/writing-good-bug-reports/

Thank you

Michael Ochs
@_mochs
https://pspdfkit.com/blog/