#WWDC19

Introducing SiriKit Media Intents

Danny Mandel, SiriKit Frameworks Engineer

New SiriKit
Media Intents

Handling SiriKit
Media Requests

Best
Practices

SiriKit Media Intents

"Tell <MyApp> that I love pop music."

"Play Khalid on <MyApp>."

"Put Outer Peace on my <MyApp> road trip playlist."

"Find Billie Eilish on <MyApp>."

"Play the audiobook Becoming on <MyApp>."

"I don't like this song."

"Add this to my library."

"Put on the Stuff You Should Know podcast from <MyApp>."

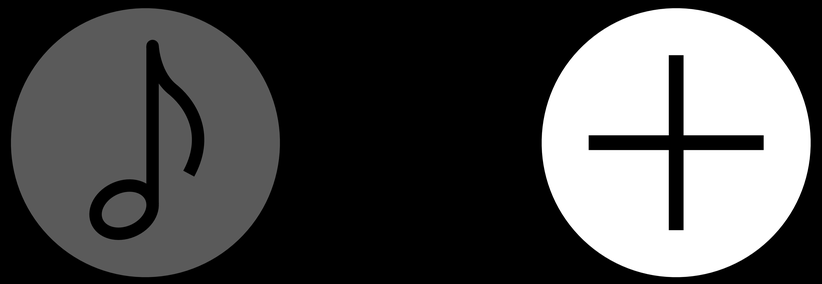
New SiriKit Media Intents

NEW

NEW

INPlayMediaIntent

NEW

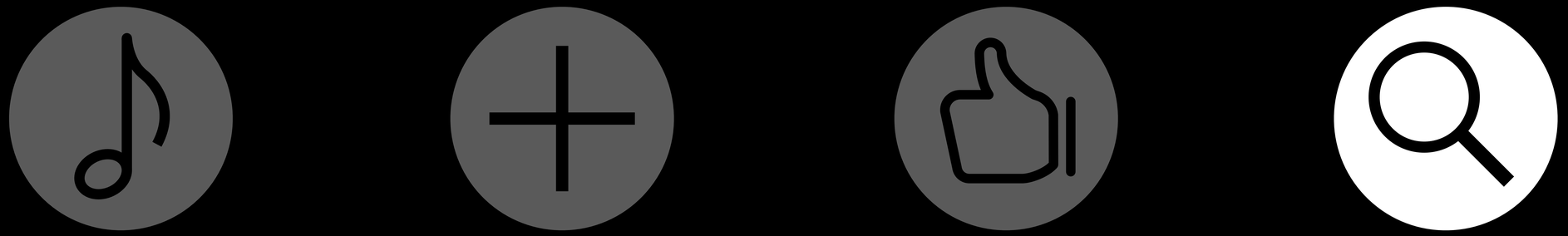


INAddMediaIntent

NEW

INUpdateMediaAffinityIntent

NEW



INSearchForMediaIntent

Music

Supported media types

“Play the song AwesomeSong in <MyApp>”

- Also — albums, artists, playlists, genres, and more!
- See `INMediaSearch` for full details

Playback controls

Podcasts

Supported media types

“Put on the Stuff You Should Know podcast from <MyApp>”

Playback

- Order
- Speed

Audiobooks

Supported media types

“Play the audiobook Becoming in <MyApp>”

Playback speed

Radio

Supported media types

“Play 89.1 FM in <MyApp>”

General

Supported media types

“Play <search term> in <MyApp>”

Your app is still supported

Only caveat is that your search queries are untyped

- Siri won’t understand the nuances of your app content types

SiriKit Request Processing

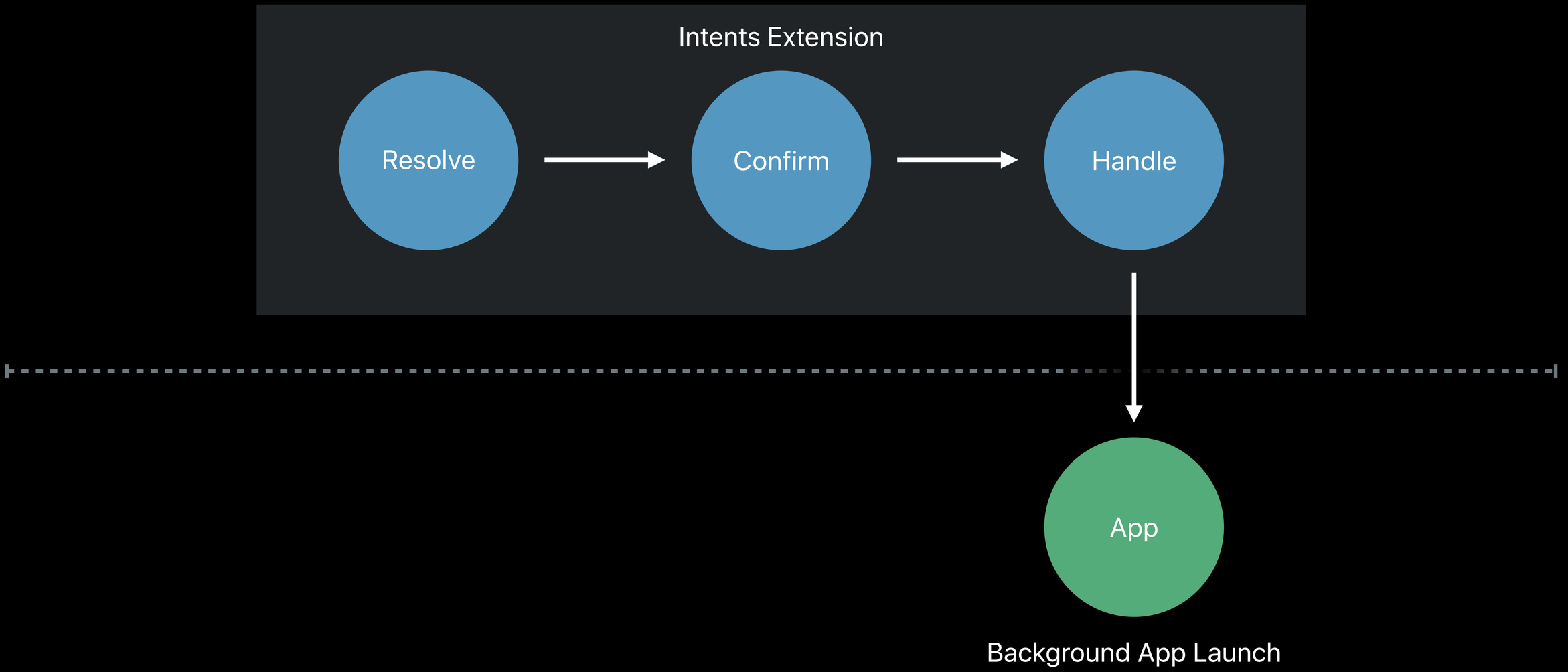
SiriKit Request Processing

SiriKit Media Intents behave like a regular SiriKit domain

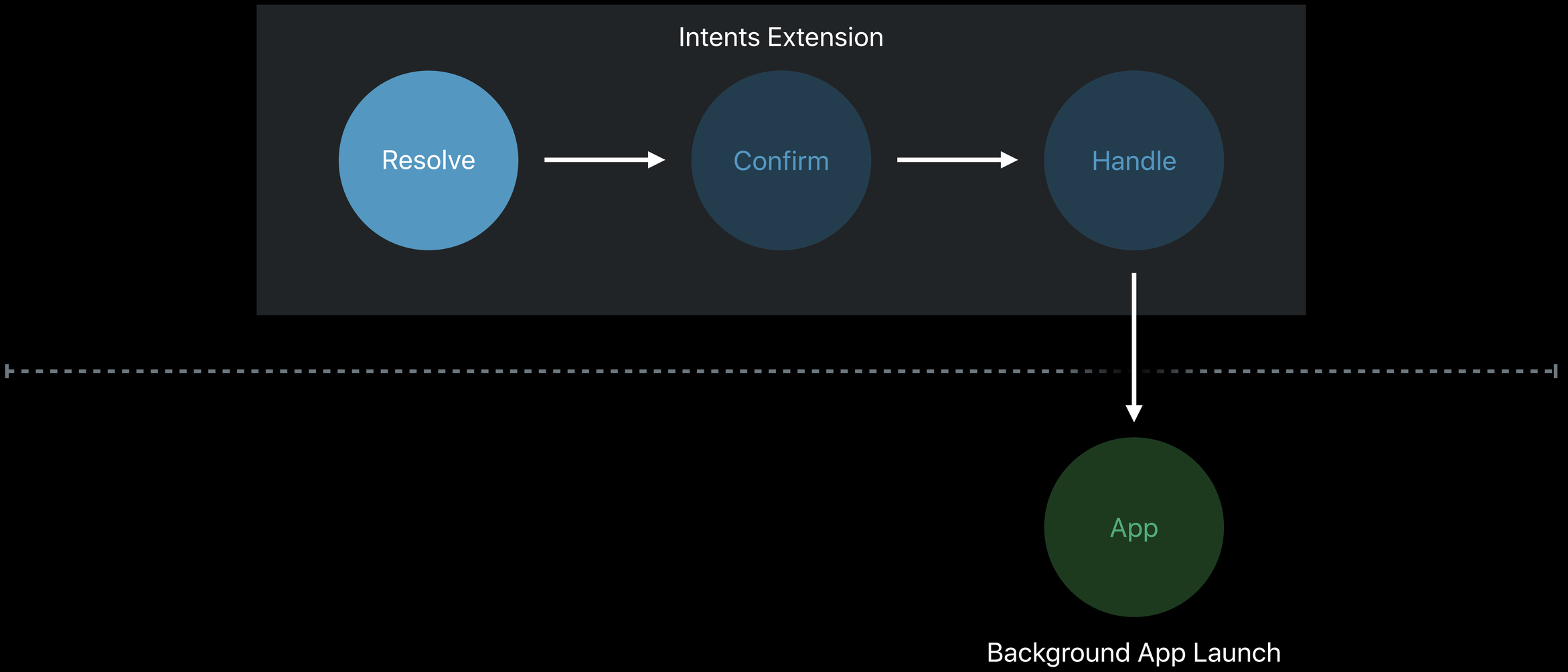
All request processing is done in your Intents app extension

Making Great SiriKit Experiences	WWDC 2017
Introducing SiriKit	WWDC 2016

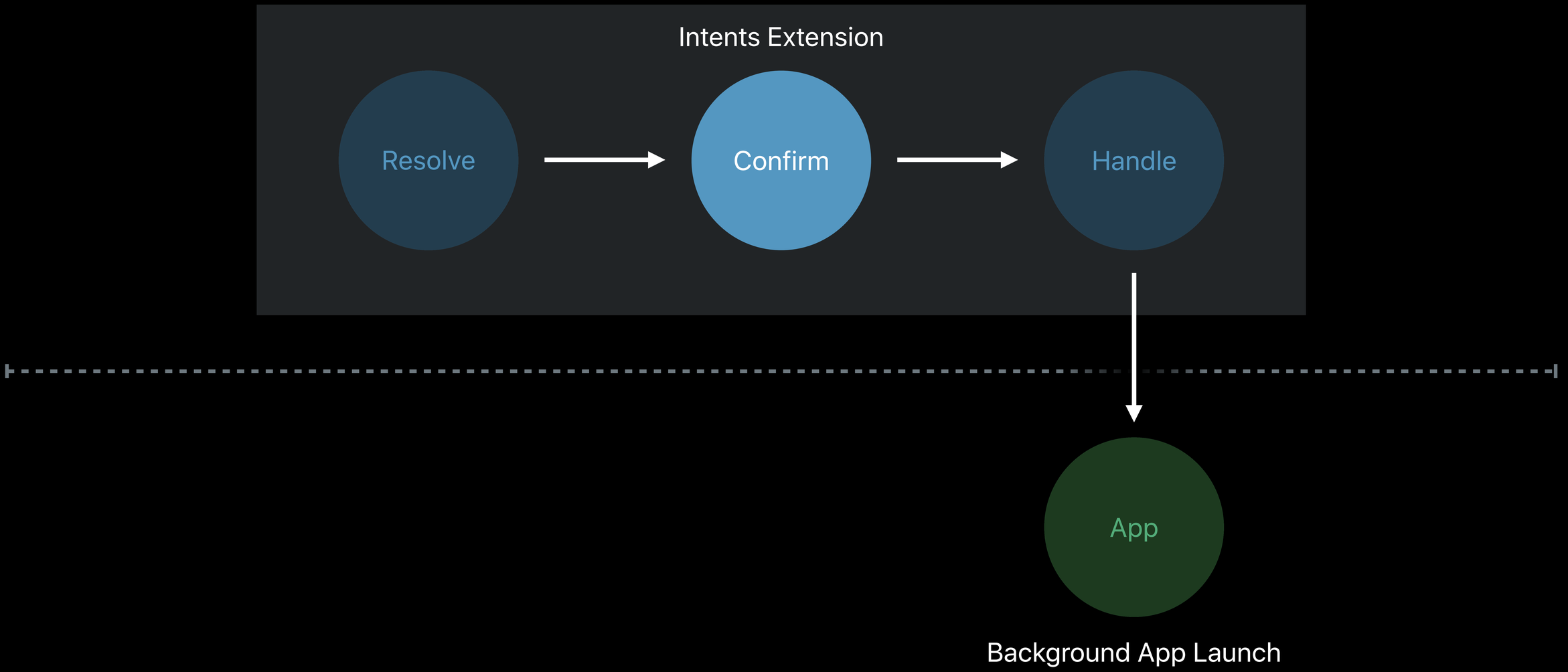
SiriKit Request Processing



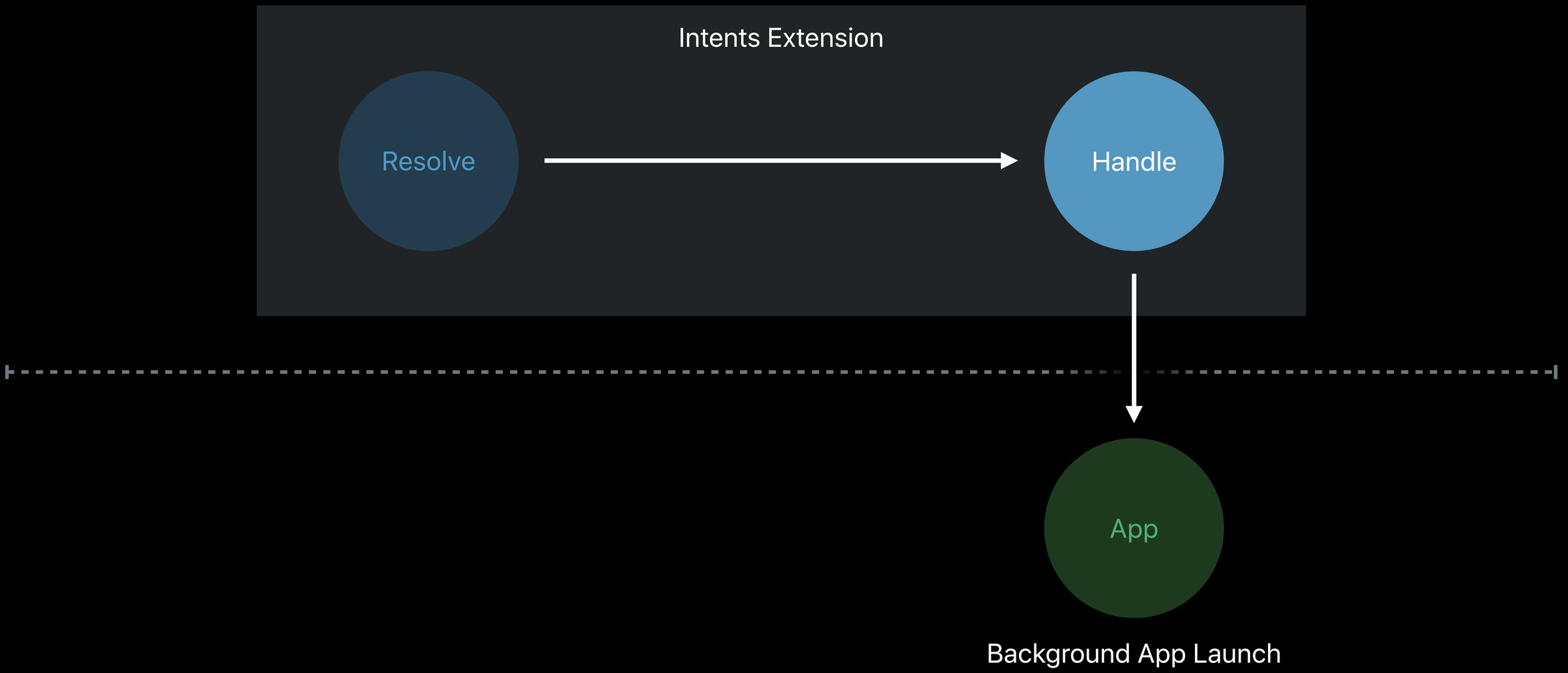
SiriKit Request Processing



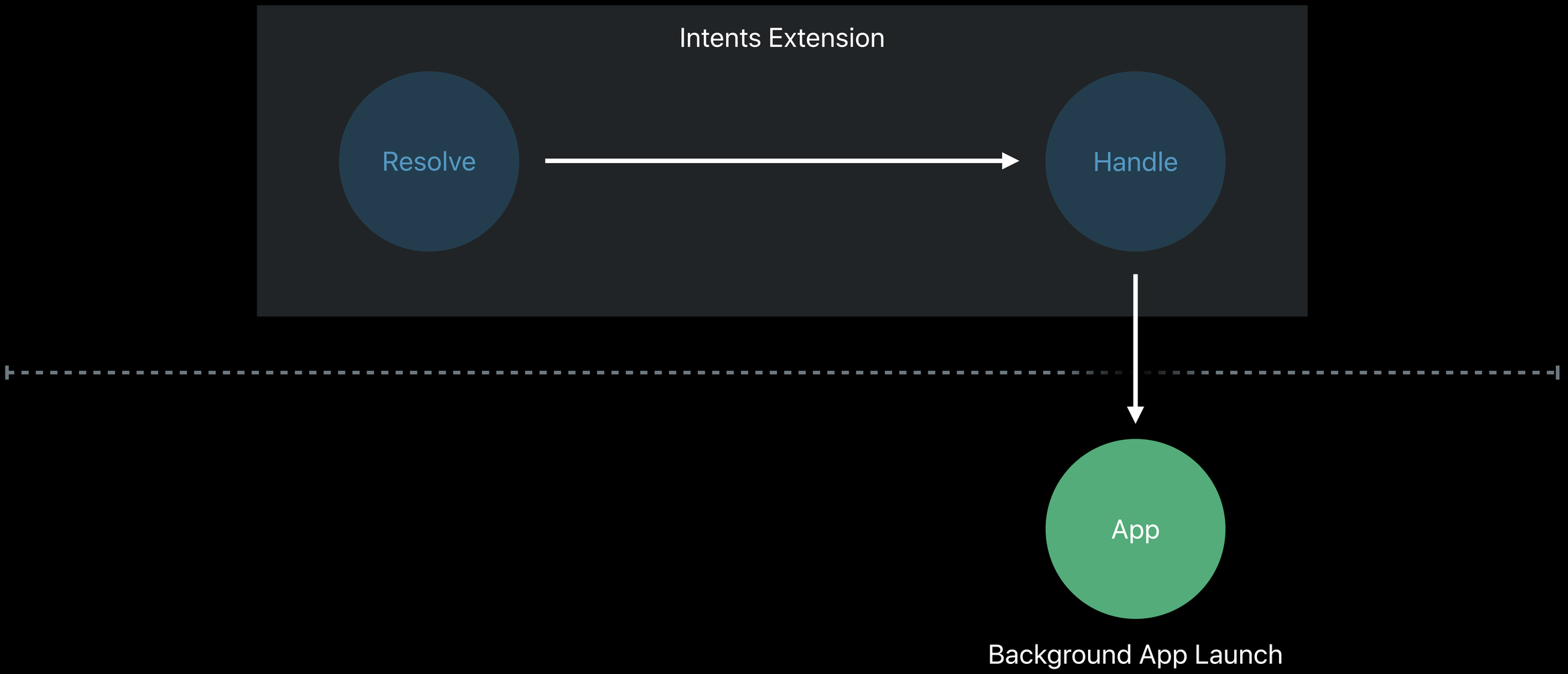
SiriKit Request Processing



SiriKit Request Processing



SiriKit Request Processing



```
func resolveMediaItems(for intent: INPlayMediaIntent, with completion: @escaping
    ([INPlayMediaMediaItemResolutionResult]) -> Void) {
    var result = INPlayMediaMediaItemResolutionResult.unsupported()
    if let mediaName = intent.mediaSearch?.mediaName {
        for item in mediaItemsFromMyAppCatalog(intent) where item.name == mediaName {
            let mediaItem = INMediaItem(identifier: item.id, title: item.name, type:
                item.type, artwork: nil, artist: item.artist)
            result = INPlayMediaMediaItemResolutionResult.success(with: mediaItem)
            break
        }
    }
    completion([result])
 }
```

```
func resolveMediaItems(for intent: INPlayMediaIntent, with completion: @escaping
    ([INPlayMediaMediaItemResolutionResult]) -> Void) {
    var result = INPlayMediaMediaItemResolutionResult.unsupported()
    if let mediaName = intent.mediaSearch?.mediaName {
        for item in mediaItemsFromMyAppCatalog(intent) where item.name == mediaName {
            let mediaItem = INMediaItem(identifier: item.id, title: item.name, type:
                item.type, artwork: nil, artist: item.artist)
            result = INPlayMediaMediaItemResolutionResult.success(with: mediaItem)
            break
        }
    }
    completion([result])
}
```

```
func resolveMediaItems(for intent: INPlayMediaIntent, with completion: @escaping
    ([INPlayMediaMediaItemResolutionResult]) -> Void) {
    var result = INPlayMediaMediaItemResolutionResult.unsupported()
    if let mediaName = intent.mediaSearch?.mediaName {
        for item in mediaItemsFromMyAppCatalog(intent) where item.name == mediaName {
            let mediaItem = INMediaItem(identifier: item.id, title: item.name, type:
                item.type, artwork: nil, artist: item.artist)
            result = INPlayMediaMediaItemResolutionResult.success(with: mediaItem)
            break
        }
    }
    completion([result])
}
```

```
func resolveMediaItems(for intent: INPlayMediaIntent, with completion: @escaping
    ([INPlayMediaMediaItemResolutionResult]) -> Void) {
    var result = INPlayMediaMediaItemResolutionResult.unsupported()
    if let mediaName = intent.mediaSearch?.mediaName {
        for item in mediaItemsFromMyAppCatalog(intent) where item.name == mediaName {
            let mediaItem = INMediaItem(identifier: item.id, title: item.name, type:
                item.type, artwork: nil, artist: item.artist)
            result = INPlayMediaMediaItemResolutionResult.success(with: mediaItem)
            break
        }
    }
    completion([result])
}
```

```
func resolveMediaItems(for intent: INPlayMediaIntent, with completion: @escaping
    ([INPlayMediaMediaItemResolutionResult]) -> Void) {
    var result = INPlayMediaMediaItemResolutionResult.unsupported()
    if let mediaName = intent.mediaSearch?.mediaName {
        for item in mediaItemsFromMyAppCatalog(intent) where item.name == mediaName {
            let mediaItem = INMediaItem(identifier: item.id, title: item.name, type:
                item.type, artwork: nil, artist: item.artist)
            result = INPlayMediaMediaItemResolutionResult.success(with: mediaItem)
            break
        }
    }
    completion([result])
 }
```

```
func resolveMediaItems(for intent: INPlayMediaIntent, with completion: @escaping
    ([INPlayMediaMediaItemResolutionResult]) -> Void) {
    var result = INPlayMediaMediaItemResolutionResult.unsupported()
    if let mediaName = intent.mediaSearch?.mediaName {
        for item in mediaItemsFromMyAppCatalog(intent) where item.name == mediaName {
            let mediaItem = INMediaItem(identifier: item.id, title: item.name, type:
                item.type, artwork: nil, artist: item.artist)
            result = INPlayMediaMediaItemResolutionResult.success(with: mediaItem)
            break
        }
    }
    completion([result])
 }
```

```
func resolveMediaItems(for intent: INPlayMediaIntent, with completion: @escaping
    ([INPlayMediaMediaItemResolutionResult]) -> Void) {
    var result = INPlayMediaMediaItemResolutionResult.unsupported()
    if let mediaName = intent.mediaSearch?.mediaName {
        for item in mediaItemsFromMyAppCatalog(intent) where item.name == mediaName {
            let mediaItem = INMediaItem(identifier: item.id, title: item.name, type:
                item.type, artwork: nil, artist: item.artist)
            result = INPlayMediaMediaItemResolutionResult.success(with: mediaItem)
            break
        }
    }
    completion([result])
}
```

```
func resolveMediaItems(for intent: INPlayMediaIntent, with completion: @escaping
    ([INPlayMediaMediaItemResolutionResult]) -> Void) {
    var result = INPlayMediaMediaItemResolutionResult.unsupported()
    if let mediaName = intent.mediaSearch?.mediaName {
        for item in mediaItemsFromMyAppCatalog(intent) where item.name == mediaName {
            let mediaItem = INMediaItem(identifier: item.id, title: item.name, type:
                item.type, artwork: nil, artist: item.artist)
            result = INPlayMediaMediaItemResolutionResult.success(with: mediaItem)
            break
        }
    }
    completion([result])
}
```

```
func handle(intent: INPlayMediaIntent, completion: (INPlayMediaIntentResponse) -> Void) {
    completion(INPlayMediaIntentResponse(code: .handleInApp, userActivity: nil))
}
```

```
func handle(intent: INPlayMediaIntent, completion: (INPlayMediaIntentResponse) -> Void) {
    completion(INPlayMediaIntentResponse(code: .handleInApp, userActivity: nil))
}
```

```swift
func application(_ application: UIApplication, handle intent: INIntent, completionHandler:
@escaping (INIntentResponse) -> Void) {
    if let playMediaIntent = intent as? INPlayMediaIntent {
        if let mediaItems = playMediaIntent.mediaItems {
           let mediaItemToPlay = mediaItems.first
           // Do whatever your app normally does to begin playback
           beginPlayback(mediaItemToPlay)
           completionHandler(INPlayMediaIntentResponse(code: .success, userActivity: nil))
        }
    }
}
```

```swift
func application(_ application: UIApplication, handle intent: INIntent, completionHandler:
@escaping (INIntentResponse) -> Void) {
    if let playMediaIntent = intent as? INPlayMediaIntent {
        if let mediaItems = playMediaIntent.mediaItems {
           let mediaItemToPlay = mediaItems.first
           // Do whatever your app normally does to begin playback
           beginPlayback(mediaItemToPlay)
           completionHandler(INPlayMediaIntentResponse(code: .success, userActivity: nil))
        }
    }
}
```

```swift
func application(_ application: UIApplication, handle intent: INIntent, completionHandler:
@escaping (INIntentResponse) -> Void) {
    if let playMediaIntent = intent as? INPlayMediaIntent {
        if let mediaItems = playMediaIntent.mediaItems {
            let mediaItemToPlay = mediaItems.first
            // Do whatever your app normally does to begin playback
            beginPlayback(mediaItemToPlay)
            completionHandler(INPlayMediaIntentResponse(code: .success, userActivity: nil))
        }
    }
}
```

```
func application(_ application: UIApplication, handle intent: INIntent, completionHandler:
@escaping (INIntentResponse) -> Void) {
    if let playMediaIntent = intent as? INPlayMediaIntent {
        if let mediaItems = playMediaIntent.mediaItems {
            let mediaItemToPlay = mediaItems.first
            // Do whatever your app normally does to begin playback
            beginPlayback(mediaItemToPlay)
            completionHandler(INPlayMediaIntentResponse(code: .success, userActivity: nil))
        }
    }
}
```

```swift
func application(_ application: UIApplication, handle intent: INIntent, completionHandler:
@escaping (INIntentResponse) -> Void) {
    if let playMediaIntent = intent as? INPlayMediaIntent {
        if let mediaItems = playMediaIntent.mediaItems {
            let mediaItemToPlay = mediaItems.first
            // Do whatever your app normally does to begin playback
            beginPlayback(mediaItemToPlay)
            completionHandler(INPlayMediaIntentResponse(code: .success, userActivity: nil))
        }
    }
}
```

Demo

Ryan Klems, SiriKit Frameworks Engineer

Demo Summary

Add your intents extension to your app

Specify supported intents and media types

Implement resolve, handle for `INPlayMediaIntent` and `INAddMediaIntent`

Best Practices

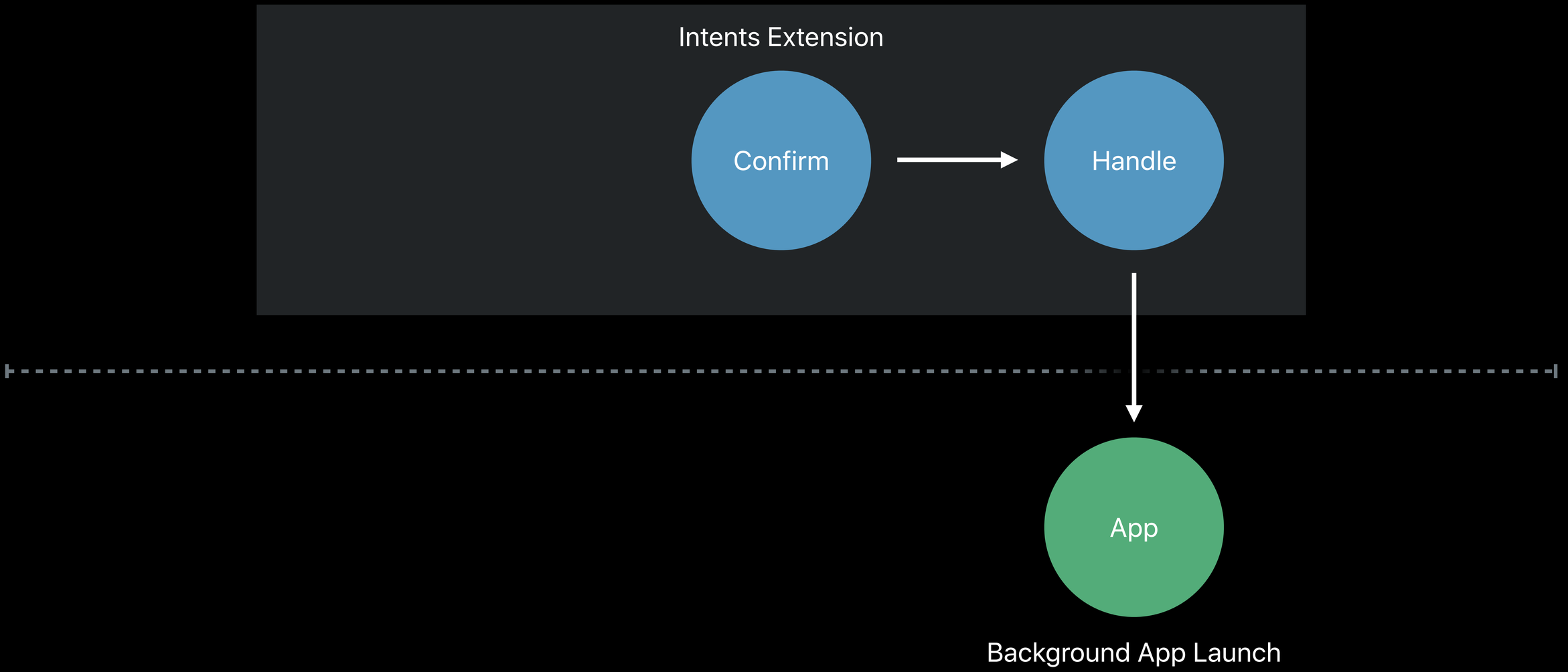
Adding SiriKit Support to Shortcuts

Handle and background app launch are the same

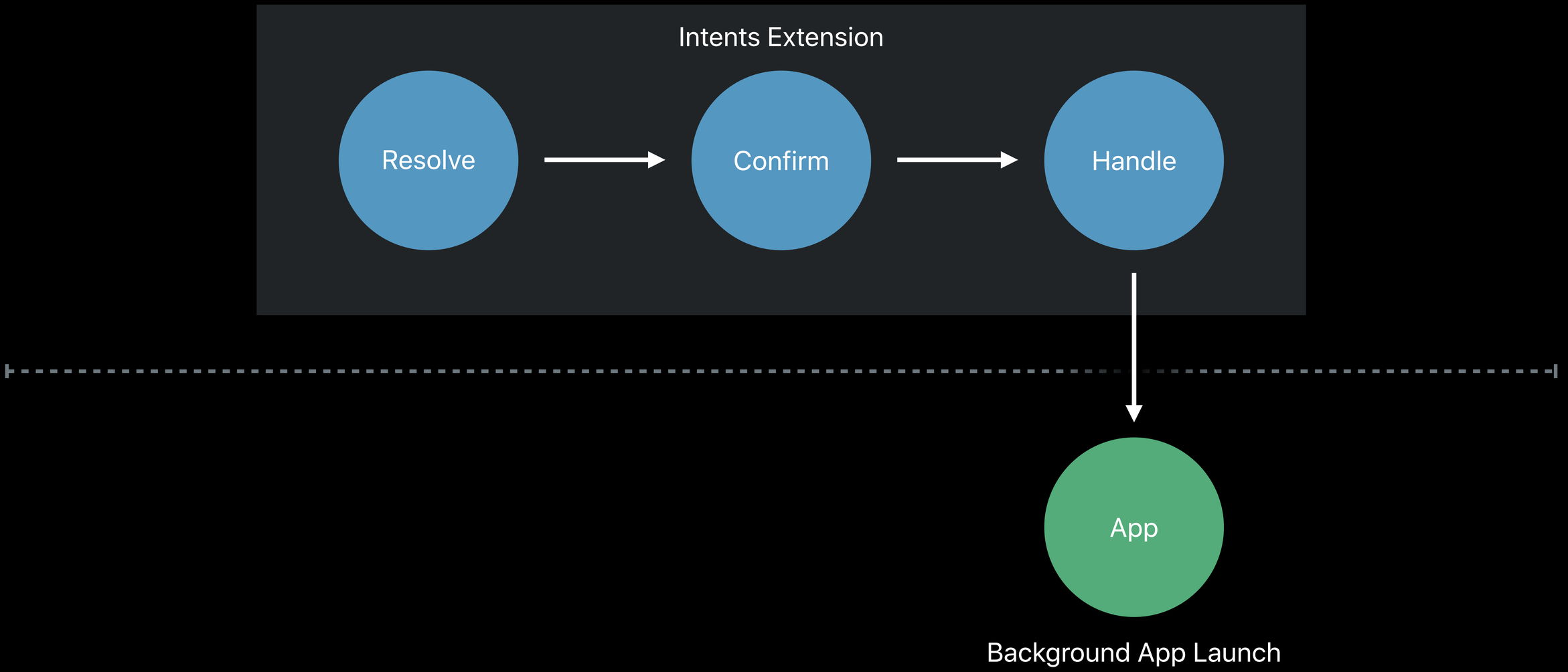
Need to add resolve methods

Update intents extension with supported media types

SiriKit Request Processing



SiriKit Request Processing



Best Practices for Apple Watch

Best Practices for Apple Watch

Foreground app launch via `INPlayMediaIntentResponseCode.continueInApp`

Best Practices for Apple Watch

Foreground app launch via `INPlayMediaIntentResponseCode.continueInApp`

Intent is handled by your `WKExtensionDelegate`

Best Practices for Apple Watch

Foreground app launch via `INPlayMediaIntentResponseCode.continueInApp`

Intent is handled by your `WKExtensionDelegate`

```
func handle(_ userActivity: NSUserActivity) {
    if let intent = userActivity.interaction?.intent {
        let playMediaIntent = intent as? INPlayMediaIntent
        let mediaItemToPlay = playMediaIntent?.mediaItems?.first
        beginPlayback(mediaItemToPlay)
    }
}
```

Best Practices for Apple Watch

Foreground app launch via `INPlayMediaIntentResponseCode.continueInApp`

Intent is handled by your `WKExtensionDelegate`

```
func handle(_ userActivity: NSUserActivity) {
    if let intent = userActivity.interaction?.intent {
        let playMediaIntent = intent as? INPlayMediaIntent
        let mediaItemToPlay = playMediaIntent?.mediaItems?.first
        beginPlayback(mediaItemToPlay)
    }
}
```

Prefer on-device cache in your resolve method

Writing an Effective Search Method in Resolve

“Play AwesomeSong in <MyApp>”

We know we will compare against the `IN*MediaIntent.mediaSearch.mediaName`

What are some common edge cases we need to consider?

Writing an Effective Search Method in Resolve

Things to ignore

- Case, Punctuation
- These should resolve the same
  - **“Play hello in <MyApp>”**
  - **“Play HELLO in <MyApp>”**
  - **“Play HELLO! in <MyApp>”**

Writing an Effective Search Method in Resolve

Writing an Effective Search Method in Resolve

“Play the album Outer Peace (Deluxe Edition) in <MyApp>”

“Play the album Outer Peace in <MyApp>”

Writing an Effective Search Method in Resolve

~~“Play the album Outer Peace (Deluxe Edition) in <MyApp>”~~

“Play the album Outer Peace in <MyApp>”

Writing an Effective Search Method in Resolve

~~“Play the album Outer Peace (Deluxe Edition) in <MyApp>”~~

“Play the album Outer Peace in <MyApp>”

“Play the album Rocketman (Music from the Motion Picture) in <MyApp>”

“Play the Rocketman soundtrack in <MyApp>”

Writing an Effective Search Method in Resolve

~~“Play the album Outer Peace (Deluxe Edition) in <MyApp>”~~

“Play the album Outer Peace in <MyApp>”

~~“Play the album Rocketman (Music from the Motion Picture) in <MyApp>”~~

“Play the Rocketman soundtrack in <MyApp>”

Writing an Effective Search Method in Resolve

~~“Play the album Outer Peace (Deluxe Edition) in <MyApp>”~~

“Play the album Outer Peace in <MyApp>”

~~“Play the album Rocketman (Music from the Motion Picture) in <MyApp>”~~

“Play the Rocketman soundtrack in <MyApp>”

“Play the song Mile High (Feat. Travis Scott & Metro Boomin) on <MyApp>”

“Play the song Mile High on <MyApp>”

Writing an Effective Search Method in Resolve

~~“Play the album Outer Peace (Deluxe Edition) in <MyApp>”~~

“Play the album Outer Peace in <MyApp>”

~~“Play the album Rocketman (Music from the Motion Picture) in <MyApp>”~~

“Play the Rocketman soundtrack in <MyApp>”

~~“Play the song Mile High (Feat. Travis Scott & Metro Boomin) on <MyApp>”~~

“Play the song Mile High on <MyApp>”

Writing an Effective Search Method in Resolve

“Play The Stuff You Should Know Podcast in <MyApp>”

```
{
  "mediaName" : "Stuff You Should Know",
  "mediaType" : INMediaItemTypePodcast
}
```

Some podcasts also have “audio“ or “video” in the title

Writing an Effective Search Method in Resolve

Word formatting variations

- **“Play the song eighty-first in <MyApp>”**
- **“Play the song 81st in <MyApp>”**

Writing an Effective Search Method in Resolve

Word formatting variations

- **“Play the song eighty-first in <MyApp>”**
- **“Play the song 81st in <MyApp>”**
- **“Play the song I love you sun in <MyApp>”**
- **“Play the song I love you son in <MyApp>”**

Writing an Effective Search Method in Resolve

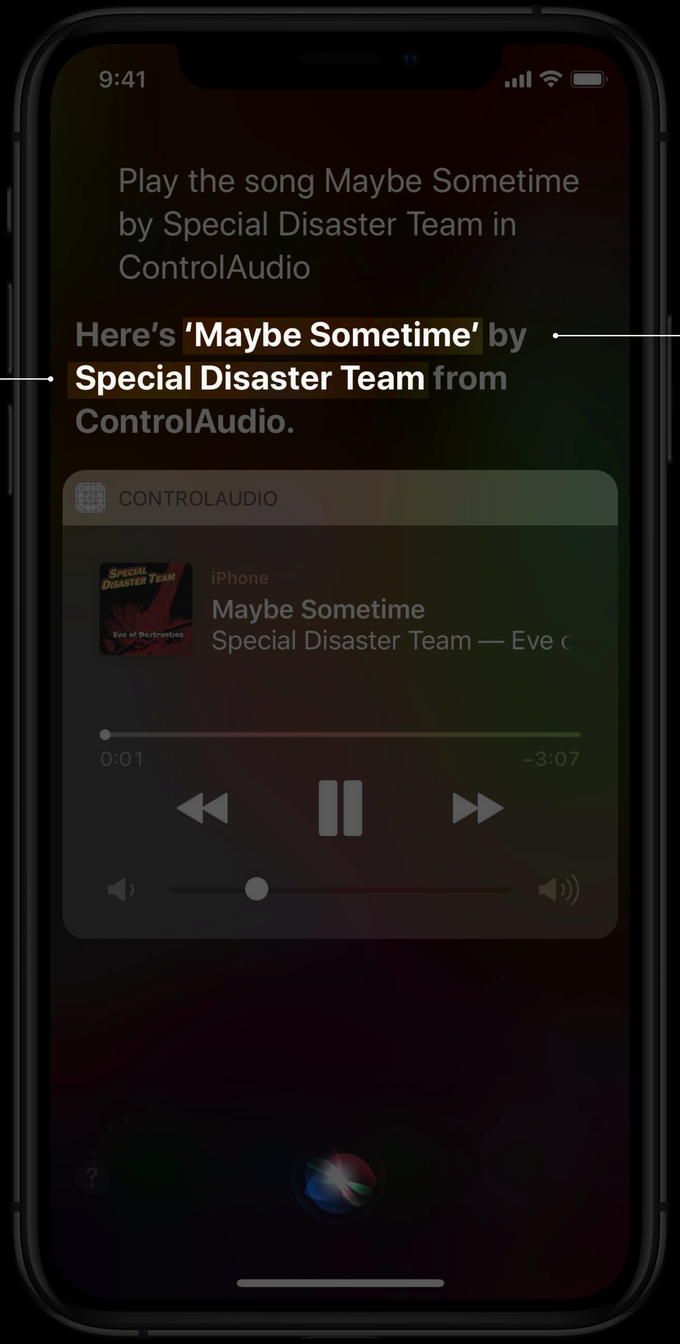
Word formatting variations

- **“Play the song eighty-first in <MyApp>”**
- **“Play the song 81st in <MyApp>”**
- **“Play the song I love you sun in <MyApp>”**
- **“Play the song I love you son in <MyApp>”**

Siri will attempt to match the entity name, but search flexibility is best


9:41
Play the song Maybe Sometime by Special Disaster Team in ControlAudio
Here's 'Maybe Sometime' by Special Disaster Team from ControlAudio.
CONTROLAUDIO
iPhone
Maybe Sometime
Special Disaster Team — Eve
0:01
-3:07

INMediaItem.artist



INMediaItem.title

Errors

Handling errors gracefully provides a nicer experience

The most common error case is not found

```
INPlayMediaMediaItemResolutionResult(INMediaItemResolutionResult.unsupported())
```

Many additional error cases trigger different dialogs

Full list in `INPlayMediaMediaItemUnsupportedReason`

Things People Can Say to Siri

Embracing Uncertainty

Unqualified playback

“Play <MyApp>”

Choose something interesting to play

An interesting playlist, recommendation

Resume the queue

Embracing Uncertainty

Unqualified playback

“Play <MyApp>”

Choose something interesting to play

An interesting playlist, recommendation

Resume the queue

Ask what they want to play

Playback Options

Repeat

- **“Play <SongName> on repeat in <MyApp>”**

Shuffle

- **“Play <PlaylistName> on shuffle in <MyApp>”**

Resume

- **“Resume <PodcastTitle> at double speed in <MyApp>”**

Playback queue location

- **“Play <ArtistName> in <MyApp> next|later”**

Search Options

Sorted playback `(INMediaSortOrder)`

- **“Play the new <PodcastTitle> podcast in <MyApp>”**
- **“Play the best <ArtistName> album in <MyApp>”**
- **“Play me something good in <MyApp>”**

Search Options

Searching by currently playing `(INMediaReferenceCurrentlyPlaying)`

- **“Add this song to my library”**
- **“I love this song”**

May also contain `MPNowPlayingInfoPropertyExternalContentIdentifier` from `MPNowPlayingInfoCenter`

Telling Siri About Your Customer

Vocabulary

User vocabulary helps Siri recognize important named entities

Only populate entities specific to your customer

- Not your entire catalog

Different types available

- PlaylistTitle, MusicArtistName, AudiobookTitle, AudiobookAuthor, ShowTitle

Global vocabulary is appropriate for global app terms

Summary

New SiriKit media intents for audio

Build the best Siri experience

Help Siri to know your user

More Information

developer.apple.com/wwdc19/207

SiriKit, Shortcuts, and Siri Event Suggestions Lab	Thursday, 12:00
SiriKit and Shortcuts Lab	Friday, 1:00


WWDC19