

Extending Your Apps with SiriKit

Session 225

Vineet Khosla SiriKit Engineering
Diana Huang SiriKit Engineering
Scott Andrus SiriKit Engineering

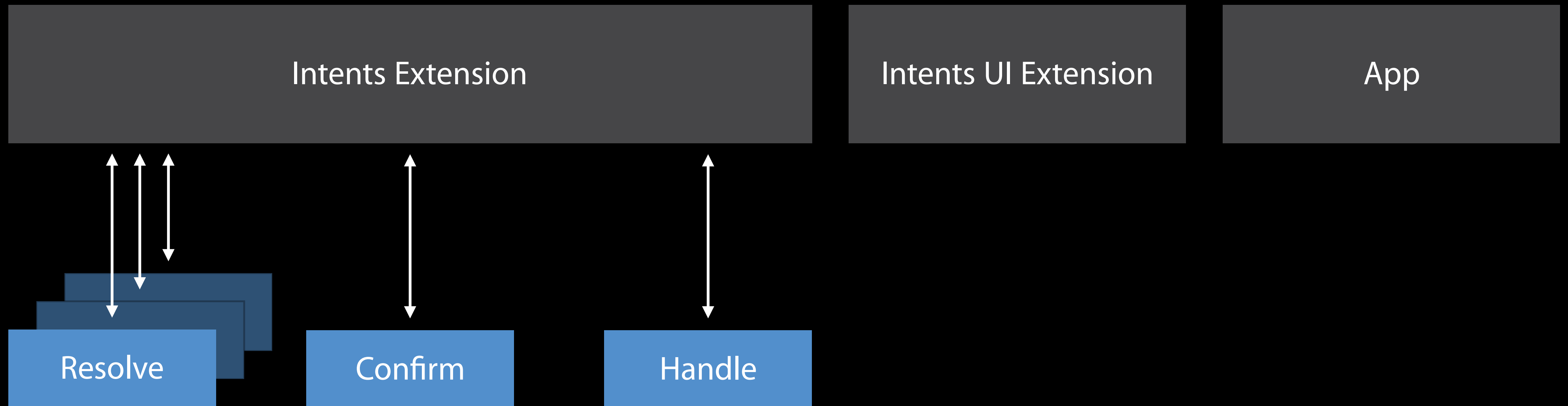
Adding SiriKit

Intents Extension

Intents UI Extension

App

Adding SiriKit



Agenda

Preparing to adopt SiriKit

Agenda

Preparing to adopt SiriKit

Adding your first Intents extension

Agenda

Preparing to adopt SiriKit

Adding your first Intents extension

Providing a user interface in Siri



UnicornChat

Preparing to Adopt SiriKit

Preparing to Adopt SiriKit

Preparing to Adopt SiriKit

Embedded frameworks

Preparing to Adopt SiriKit

Embedded frameworks

Unit tests

Preparing to Adopt SiriKit

Embedded frameworks

Unit tests

Architecting your extensions

Preparing to Adopt SiriKit

Embedded frameworks



Preparing to Adopt SiriKit

Embedded frameworks

Networking



Preparing to Adopt SiriKit

Embedded frameworks

Networking

Data model



Preparing to Adopt SiriKit

Embedded frameworks

Networking

Data model

Decision-making logic



Preparing to Adopt SiriKit

Embedded frameworks

Networking

Data model

Decision-making logic

User interfaces



Preparing to Adopt SiriKit

Embedded frameworks



Preparing to Adopt SiriKit

Embedded frameworks



Preparing to Adopt SiriKit

Unit tests

Preparing to Adopt SiriKit

Unit tests

Mock intents

Preparing to Adopt SiriKit

Unit tests

Mock intents

Make sure your app responds appropriately

Preparing to Adopt SiriKit

Architecting your extensions

SendMessageIntent

StartAudioCallIntent

StartVideoCallIntent

Preparing to Adopt SiriKit

Architecting your extensions

SendMessageIntent

StartAudioCallIntent

StartVideoCallIntent

Preparing to Adopt SiriKit

Architecting your extensions

SendMessageIntent

StartAudioCallIntent

StartVideoCallIntent

Preparing to Adopt SiriKit

Architecting your extensions

SendMessageIntent

StartAudioCallIntent

StartVideoCallIntent

Adding Your First Intents Extension

Diana Huang SiriKit Engineering

Getting Started

Getting Started

Add extension target

Getting Started

Add extension target

Configure Info.plist

Getting Started

Add extension target

Configure Info.plist

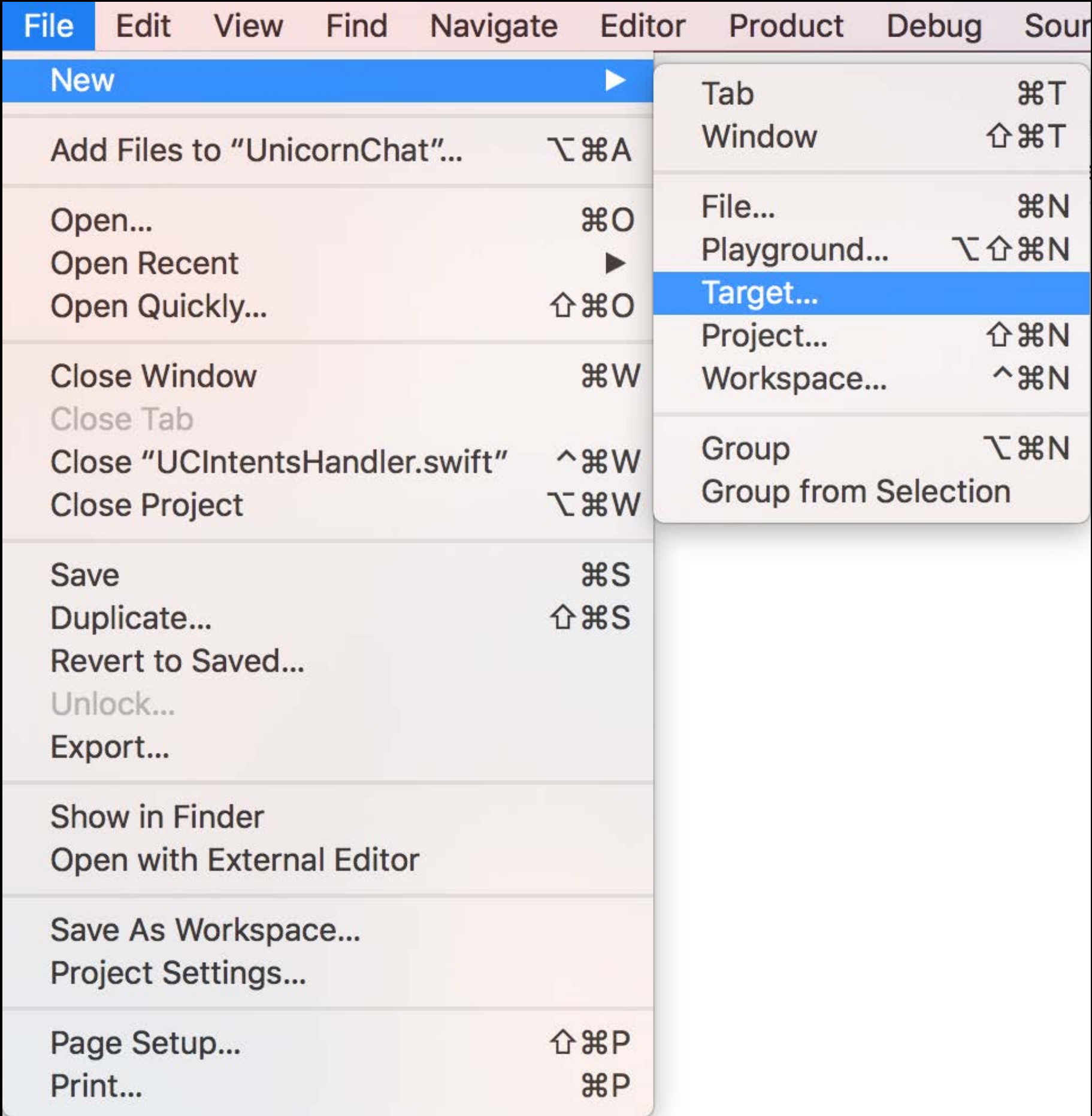
Modify principal class

Intents Extension

Adding an extension target

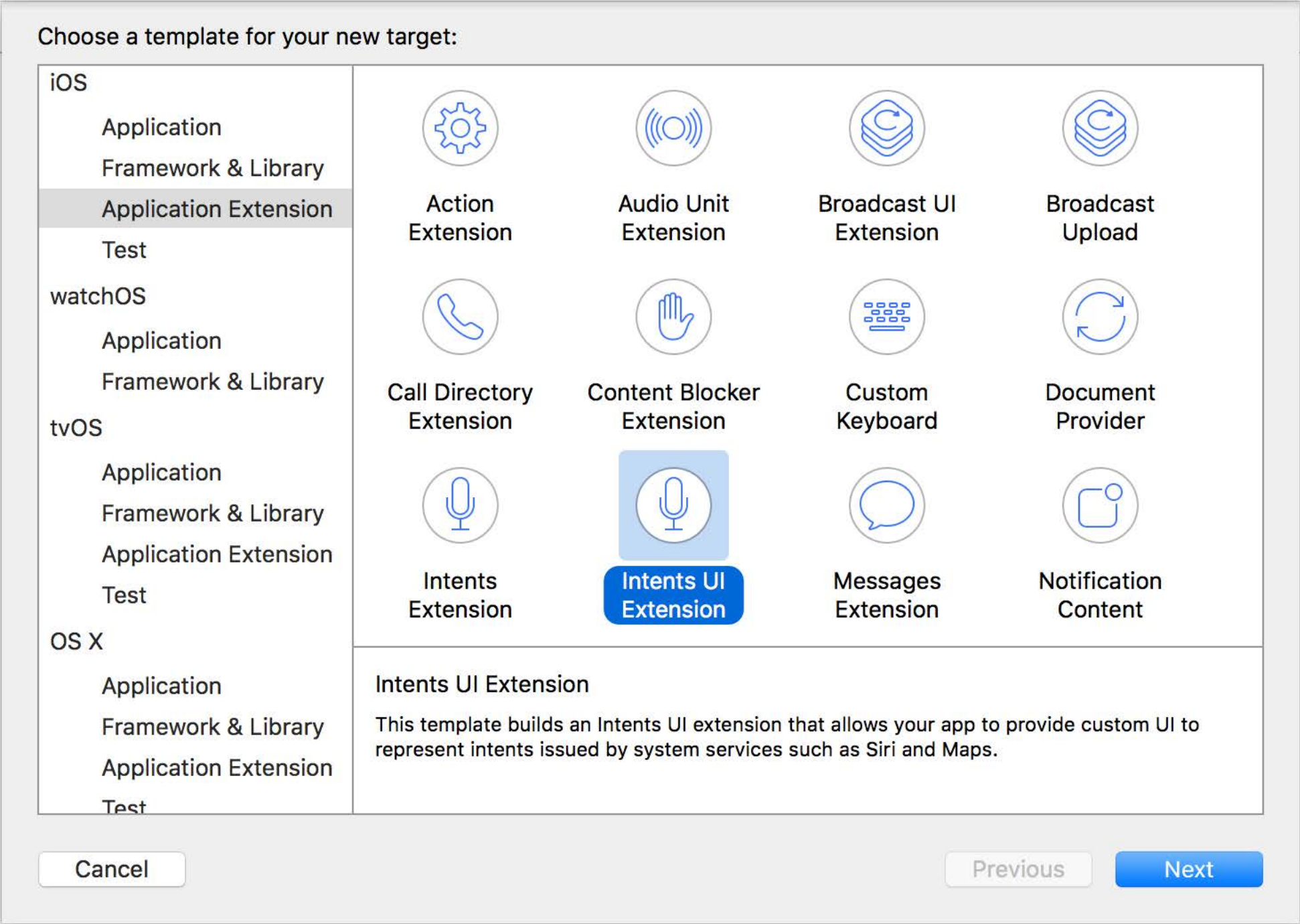
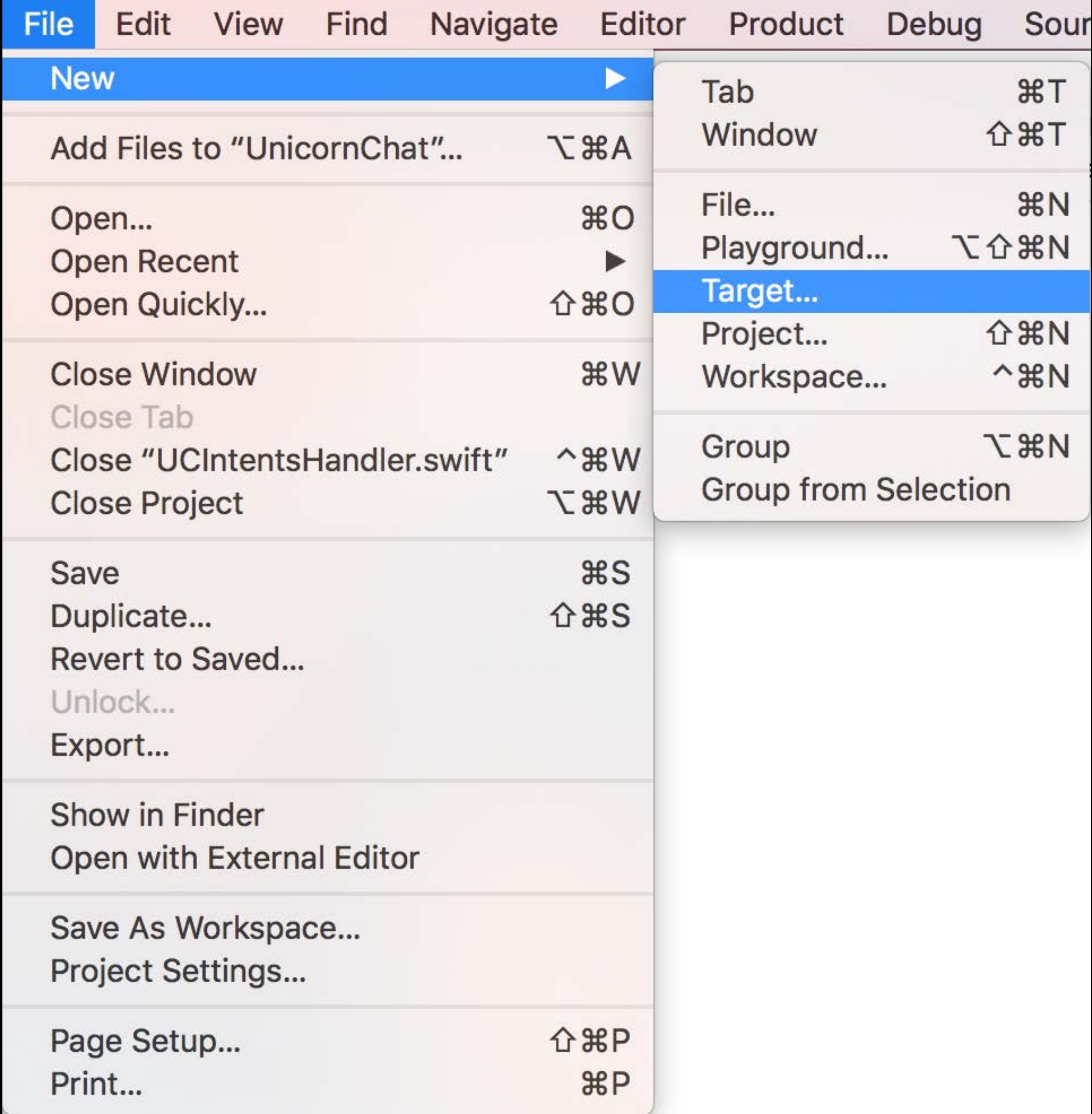
Intents Extension

Adding an extension target



Intents Extension

Adding an extension target

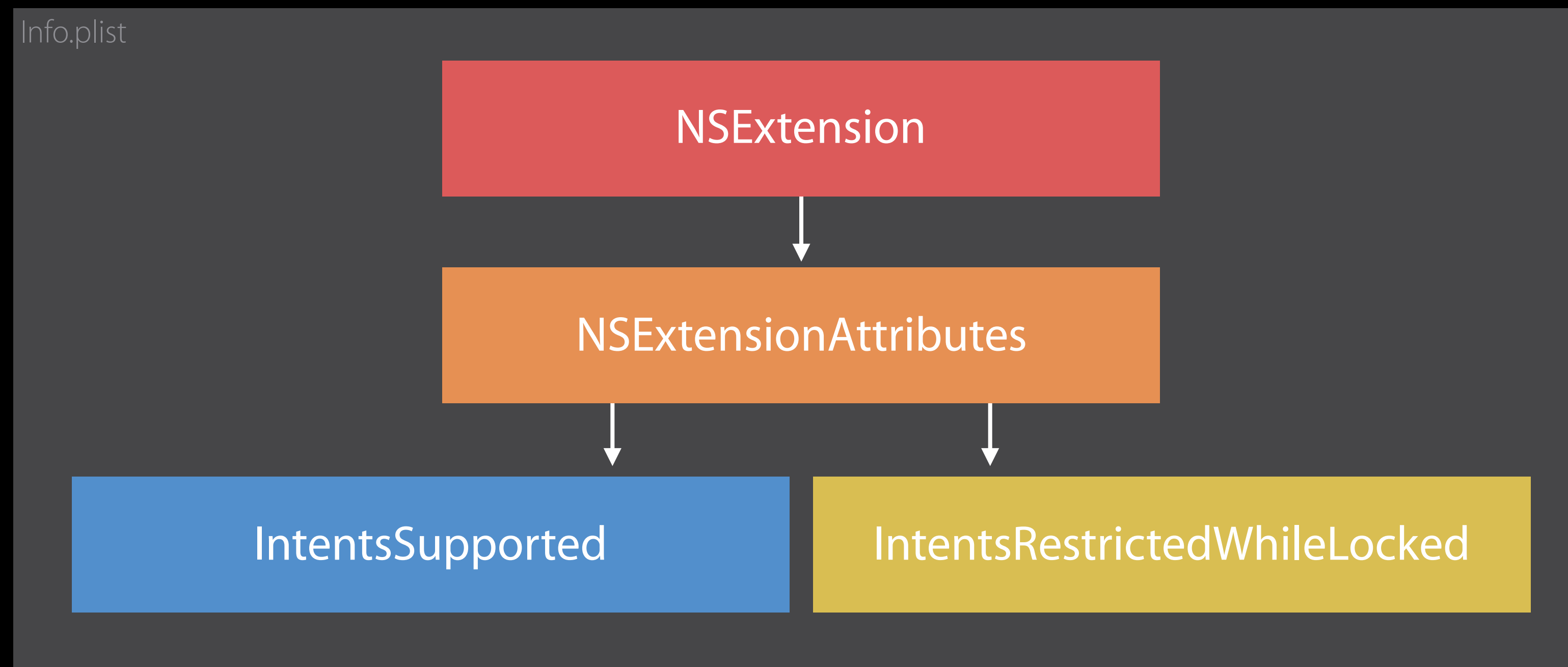


Intents Extension

Info.plist

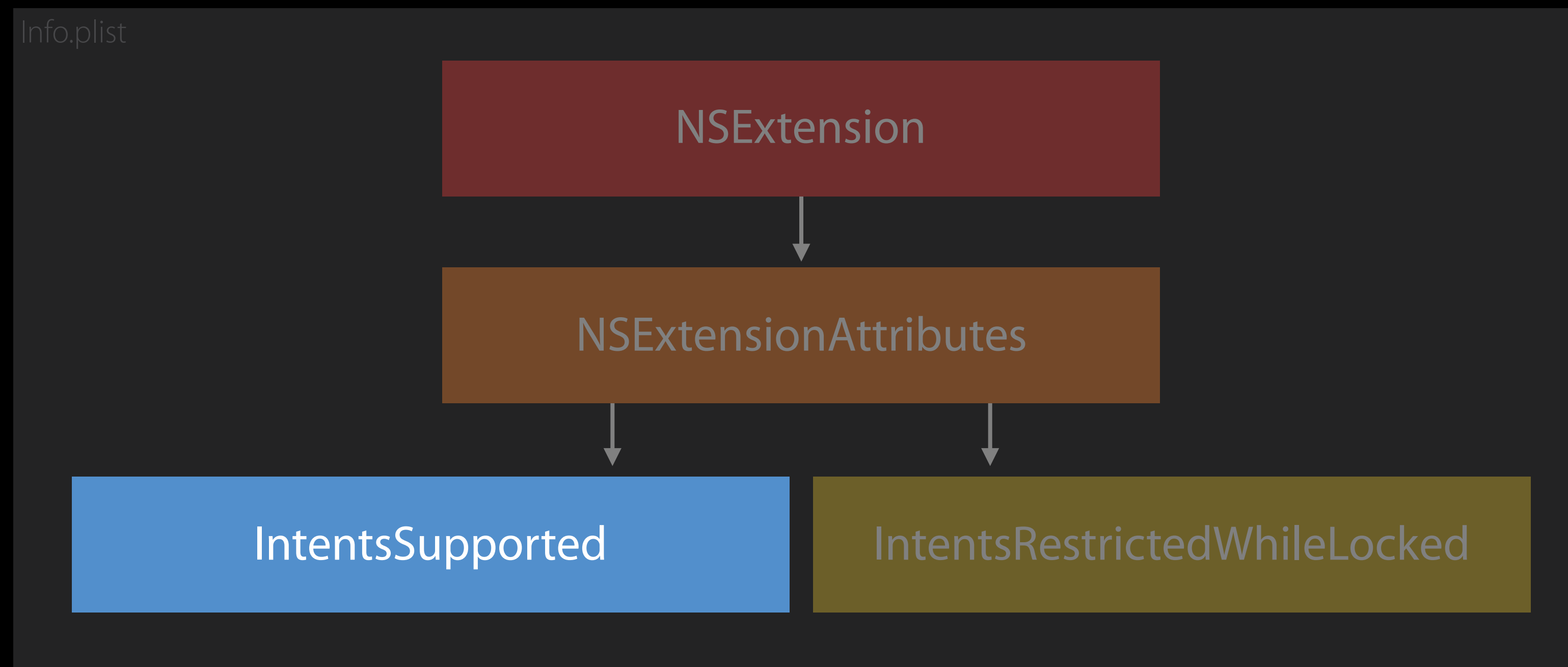
Intents Extension

Info.plist



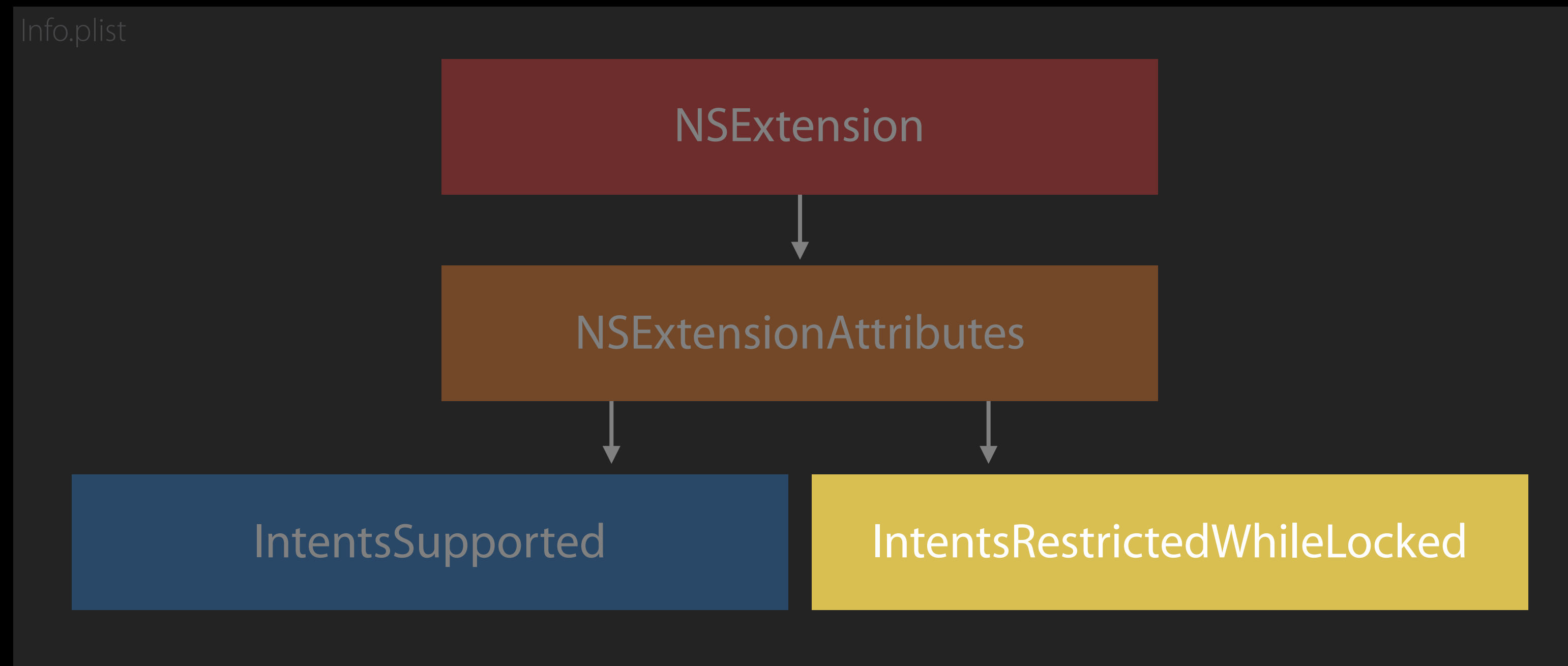
Intents Extension

Info.plist



Intents Extension

Info.plist



Intents Extension

Principal class

Intents Extension

Principal class

Subclass of INExtension

Intents Extension

Principal class

Subclass of INExtension

INIntentHandlerProviding

Intents Extension

Principal class

Subclass of INExtension

INIntentHandlerProviding

- handlerForIntent

Intents Extension

Principal class

Subclass of INExtension

INIntentHandlerProviding

- handlerForIntent
- handler class must conform to specific intent handling protocol

Demo

Creating my first Intents extension

Resolve, Confirm, Handle

Resolve

Resolve, Confirm, Handle

Resolve

Validate and clarify parameters

Resolve, Confirm, Handle

Resolve

Validate and clarify parameters

Implement it if you might need Siri to help ask users

Resolve, Confirm, Handle

Resolve

Resolve, Confirm, Handle

Resolve



recipients

Resolve, Confirm, Handle

Resolve

Contact search



recipients

The diagram consists of a large dark blue rectangle on the right side of the slide. Inside this rectangle, on the left side, is a smaller, lighter blue rectangle. The word 'recipients' is written in white text inside the lighter blue rectangle.

Resolve, Confirm, Handle

Resolve

Contact search

- Exactly one match



A diagram illustrating a search process. It consists of a large dark blue rectangle. Inside this rectangle, on the left side, is a smaller light blue rectangle labeled 'recipients'. To the right of the 'recipients' box is a large, empty dark blue rectangular area, suggesting a search space or a list of results.

recipients

Resolve, Confirm, Handle

Resolve

Contact search

- Exactly one match
- Two or more matches



recipients

Resolve, Confirm, Handle

Resolve

Contact search

- Exactly one match
- Two or more matches
- No match



A diagram illustrating a search process. It consists of a large, dark blue rectangular container. Inside this container, on the left side, is a smaller, light blue rectangular box labeled 'recipients'. The rest of the large container is empty, representing the search space or results area.

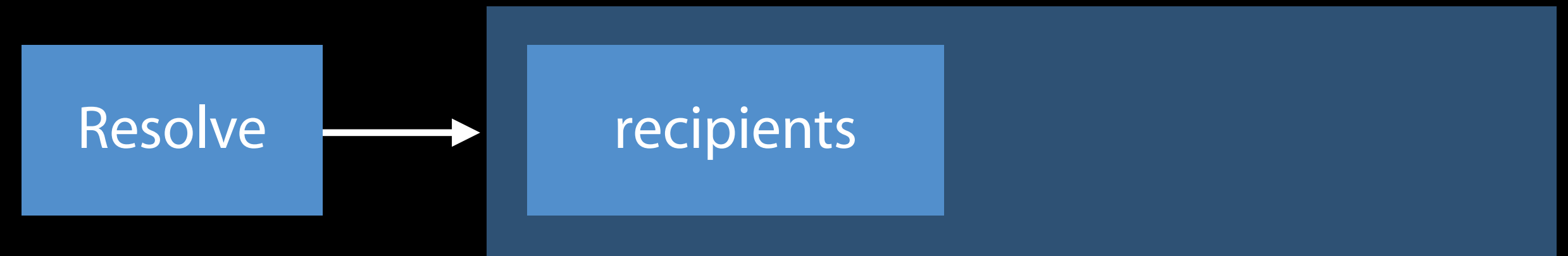
recipients

Resolve, Confirm, Handle

Resolve

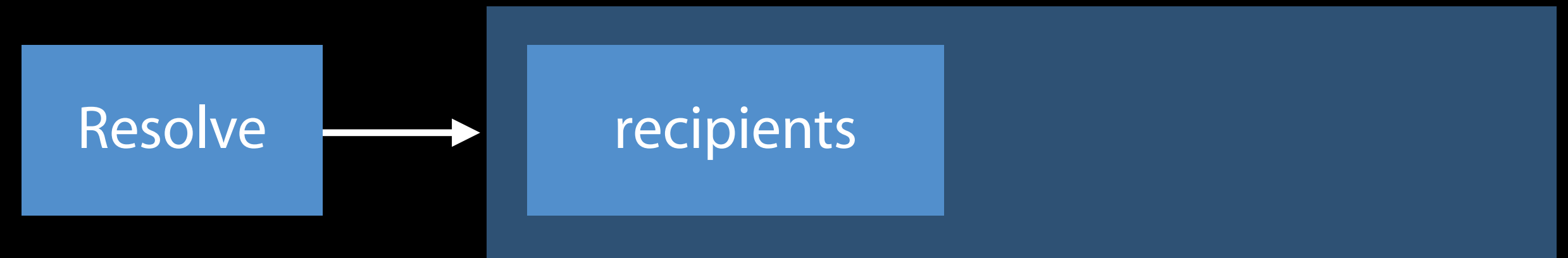
Contact search

- Exactly one match
- Two or more matches
- No match



Resolve, Confirm, Handle

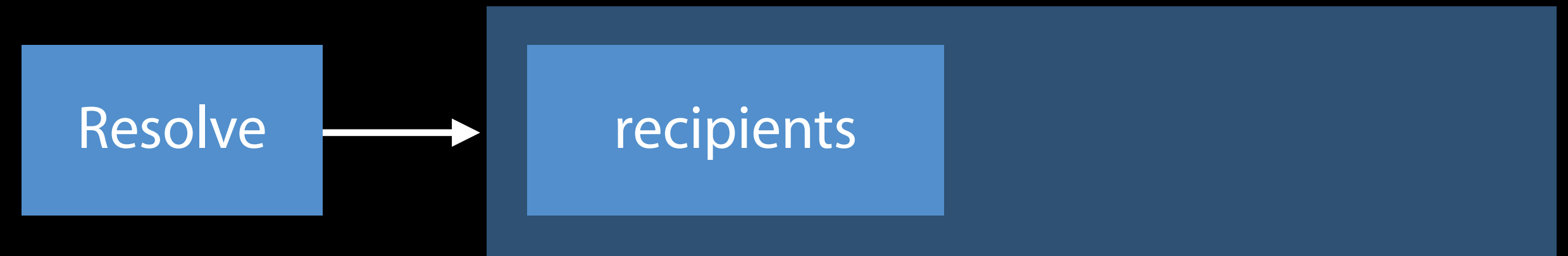
Resolve



Resolve, Confirm, Handle

Resolve

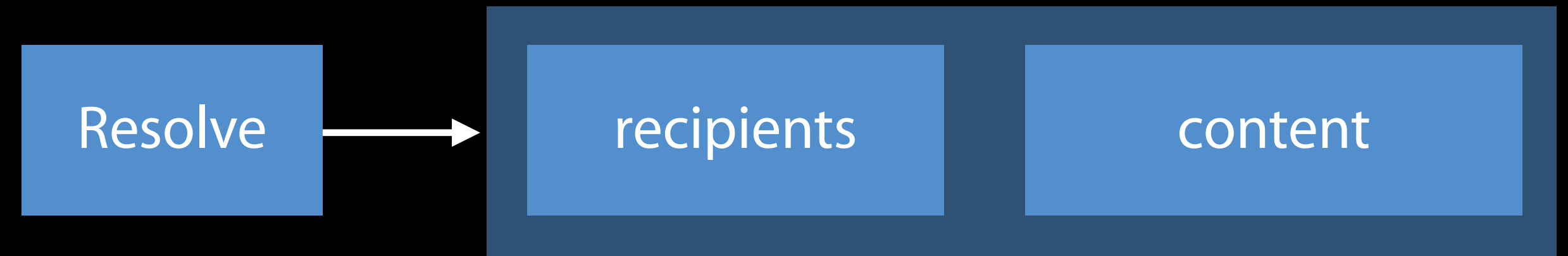
Contact search



Resolve, Confirm, Handle

Resolve

Contact search

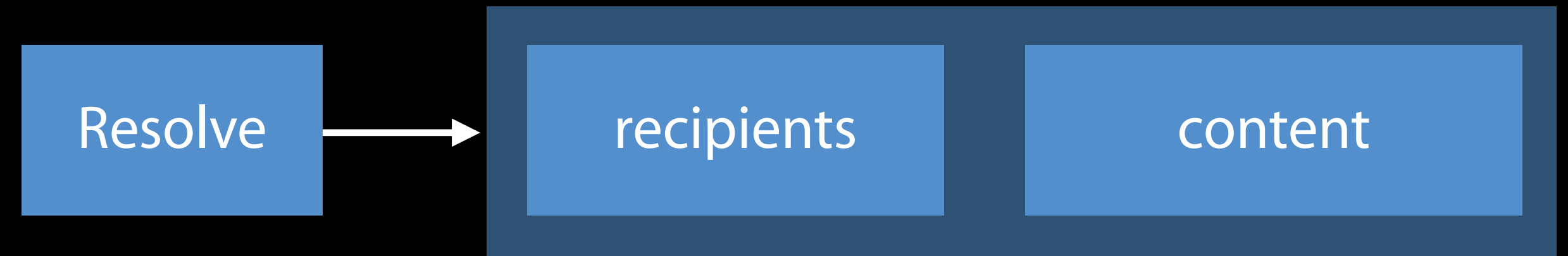


Resolve, Confirm, Handle

Resolve

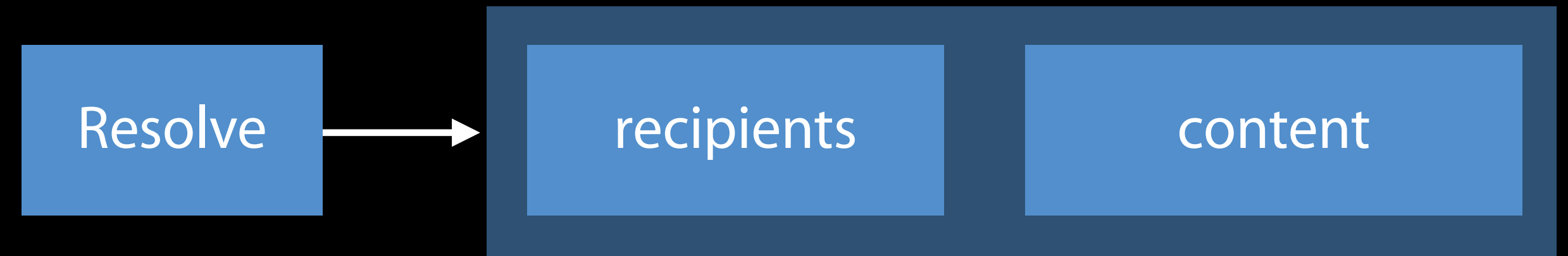
Contact search

Need a value to proceed



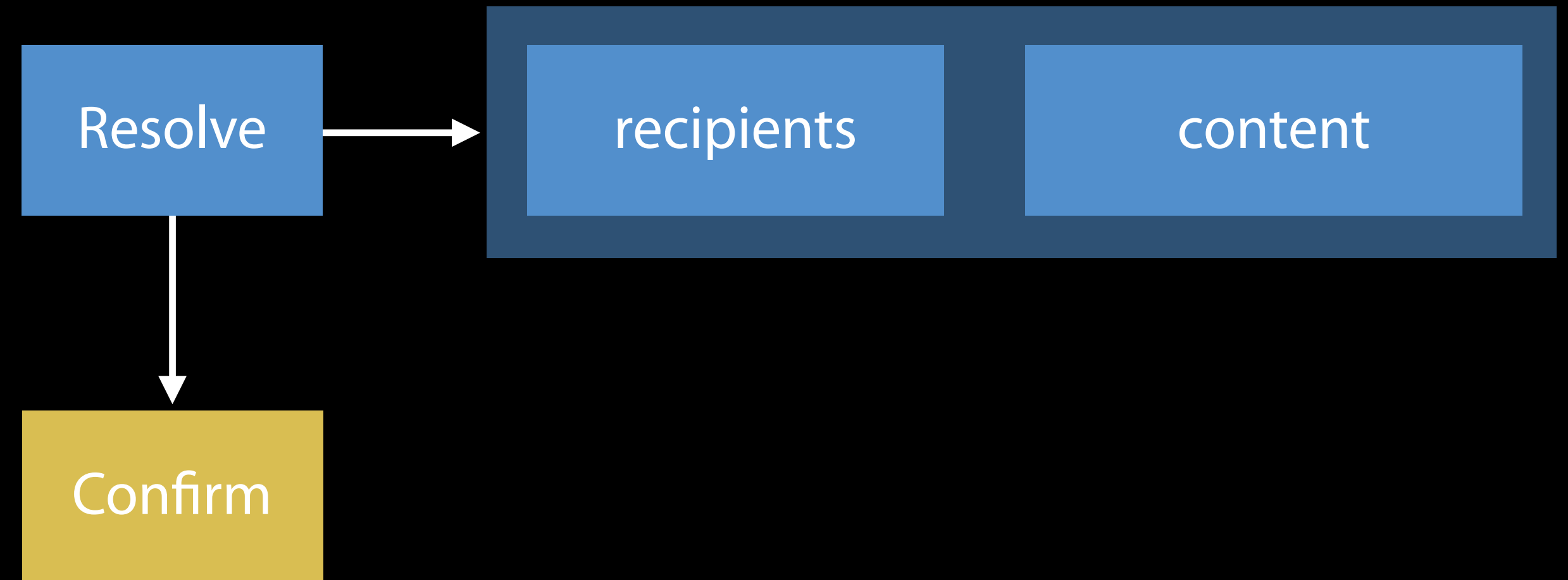
Resolve, Confirm, Handle

Confirm



Resolve, Confirm, Handle

Confirm

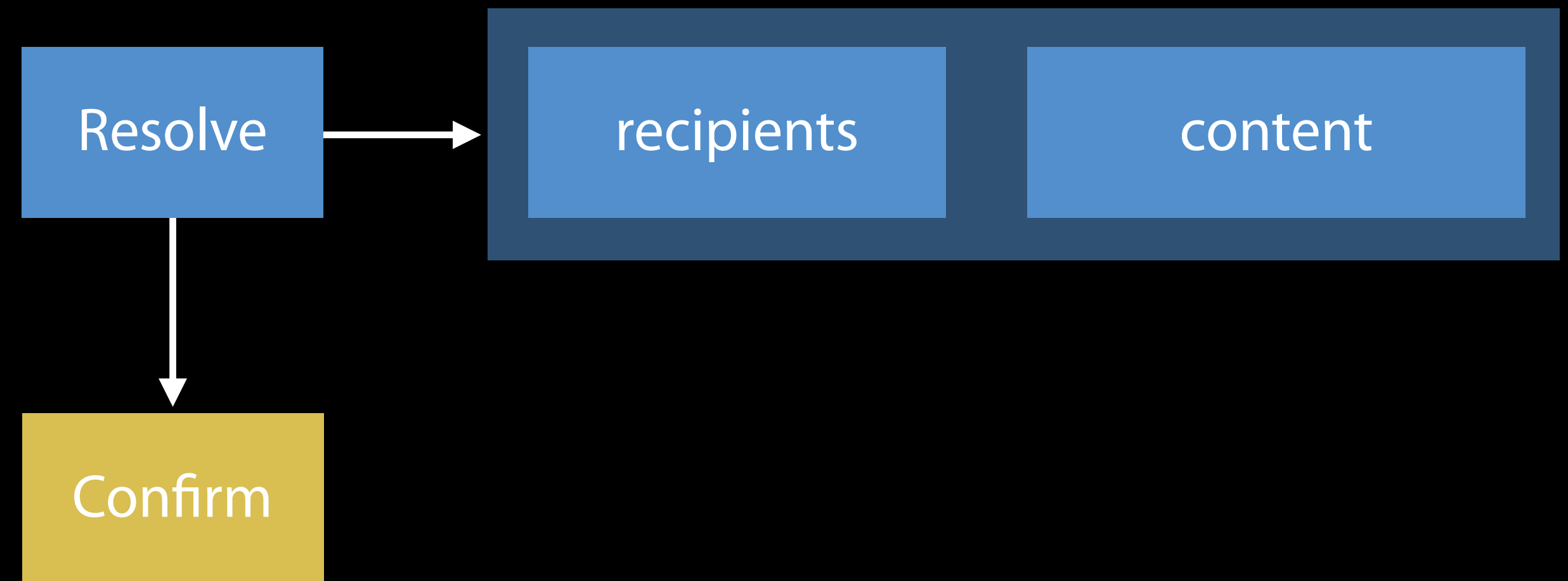


Resolve, Confirm, Handle

Confirm

Dry run

- Tell Siri how it went

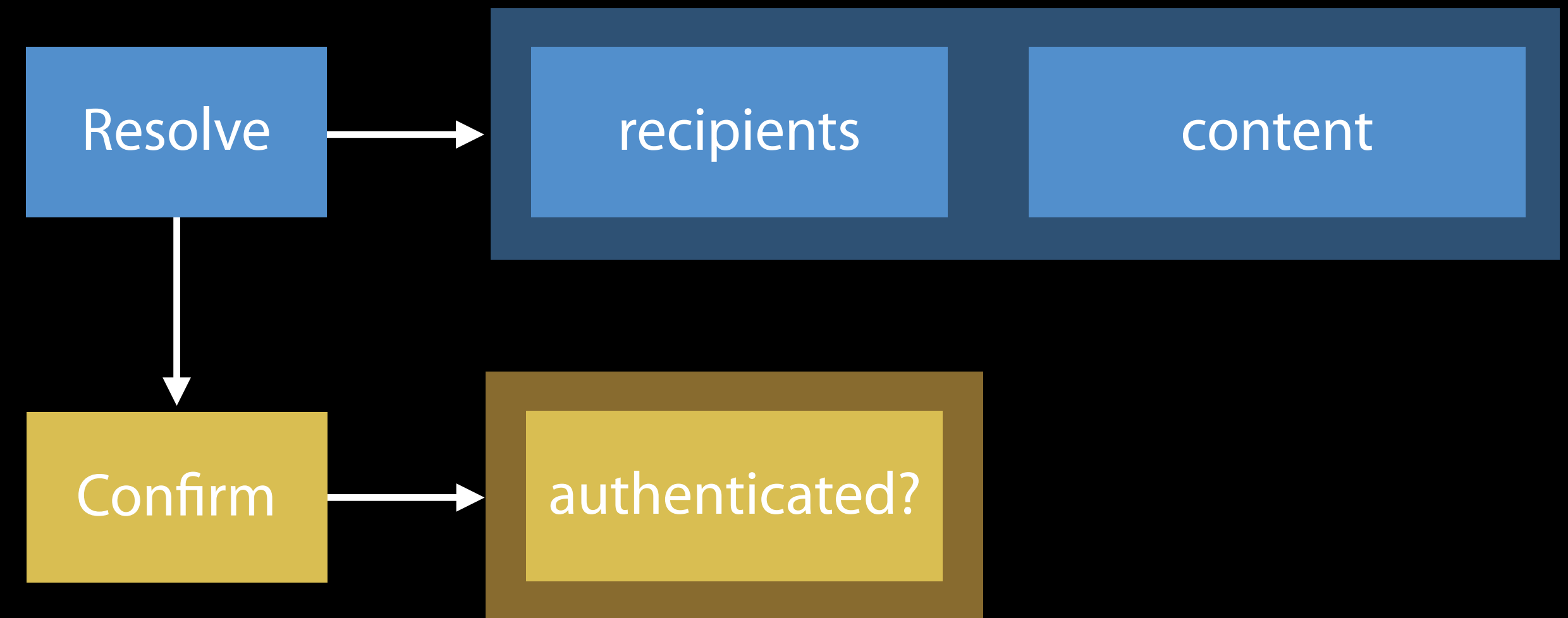


Resolve, Confirm, Handle

Confirm

Dry run

- Tell Siri how it went

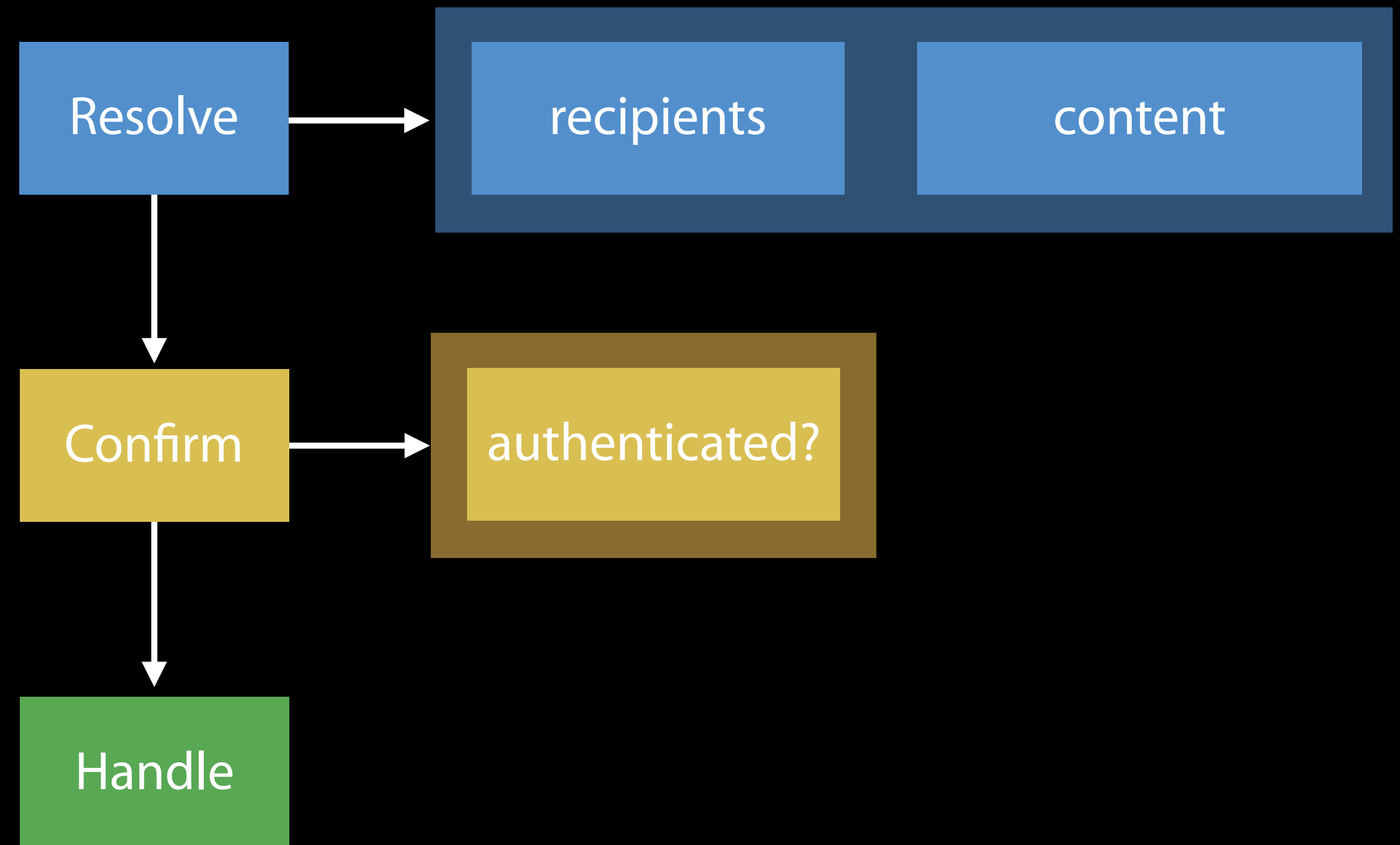


Resolve, Confirm, Handle

Handle

Just handle it!

- Again, tell Siri how it went



Demo

Filling in my app logic

NSUserActivity

Resume state in your app

NSUserActivity

Resume state in your app

Siri creates one by default

NSUserActivity

Resume state in your app

Siri creates one by default

- ActivityType is intent class name

NSUserActivity

Resume state in your app

Siri creates one by default

- ActivityType is intent class name

You can provide one to pass custom data

NSUserActivity

Resume state in your app

Siri creates one by default

- ActivityType is intent class name

You can provide one to pass custom data

INInteraction

Demo

Providing my own NSUserActivity

User-Specific Vocabulary

User-Specific Vocabulary

Phrases unique to your app and user

User-Specific Vocabulary

Phrases unique to your app and user

- e.g. contact names

User-Specific Vocabulary

Phrases unique to your app and user

- e.g. contact names

Help Siri understand what users mean

User-Specific Vocabulary

Phrases unique to your app and user

- e.g. contact names

Help Siri understand what users mean

INVocabulary API call from main app

User-Specific Vocabulary

Phrases unique to your app and user

- e.g. contact names

Help Siri understand what users mean

INVocabulary API call from main app

- NOT your extension


```
// User-Specific Vocabulary – UnicornChat
```

```
class UCAccessBookManager {
```

```
    // This method is called whenever there is a contact change
```

```
    func updateSiriKnowledgeOfContacts() {
```

```
        // provide the updated list of favorites' unicorn names to Siri
```

```
        var unicornNames = OrderedSet(array: self.sortedFavoriteUnicornNames)
```

```
        INVocabulary.shared().setVocabularyStrings(unicornNames, of:
```

```
            INVocabularyStringType.contactName)
```

```
    }
```

```
...
```

```
}
```

```
// User-Specific Vocabulary – UnicornChat
```

```
class UCAccessBookManager {
```

```
    // This method is called whenever there is a contact change
```

```
    func updateSiriKnowledgeOfContacts() {
```

```
        // provide the updated list of favorites' unicorn names to Siri
```

```
        var unicornNames = OrderedSet(array: self.sortedFavoriteUnicornNames)
```

```
        INVocabulary.shared().setVocabularyStrings(unicornNames, of:
```

```
            INVocabularyStringType.contactName)
```

```
    }
```

```
...
```

```
}
```



```
// User-Specific Vocabulary – UnicornChat
```

```
class UCAccessBookManager {
```

```
    // This method is called whenever there is a contact change
```

```
    func updateSiriKnowledgeOfContacts() {
```

```
        // provide the updated list of favorites' unicorn names to Siri
```

```
        var unicornNames = OrderedSet(array: self.sortedFavoriteUnicornNames)
```

```
        INVocabulary.shared().setVocabularyStrings(unicornNames, of:
```

```
            INVocabularyStringType.contactName)
```

```
    }
```

```
...
```

```
}
```

```
// User-Specific Vocabulary – UnicornChat
```

```
class UCAAddressBookManager {
```

```
    // This method is called whenever there is a contact change
```

```
    func updateSiriKnowledgeOfContacts() {
```

```
        // provide the updated list of favorites' unicorn names to Siri
```

```
        DispatchQueue(label: "UCSiriVocabulary").asynchronously(execute: { () -> Void in
```

```
            var unicornNames = OrderedSet(array: self.sortedFavoriteUnicornNames)
```

```
            INVocabulary.shared().setVocabularyStrings(unicornNames, of:
```

```
                INVocabularyStringType.contactName)
```

```
        })
```

```
    }
```

```
...
```

```
}
```



```
// User-Specific Vocabulary – UnicornChat
```

```
class UCAccessBookManager {
```

```
    // This method is called whenever there is a contact change
```

```
    func updateSiriKnowledgeOfContacts() {
```

```
        // provide the updated list of favorites' unicorn names to Siri
```

```
        DispatchQueue(label: "UCSiriVocabulary").asynchronously(execute: { () -> Void in
```

```
            var unicornNames = OrderedSet(array: self.sortedFavoriteUnicornNames)
```

```
            INVocabulary.shared().setVocabularyStrings(unicornNames, of:
```

```
                INVocabularyStringType.contactName)
```

```
        })
```

```
    }
```

```
...
```

```
}
```

```

// User-Specific Vocabulary - UnicornChat
AddressBook

c7 This method is called whenever there is a contact change
func updateSiriKnowledge() {
    // provide the updated list of favorites' unicorn names to Siri
    DispatchQueue(label: "CSiriVocabulary").asynchronously(execute: { () -> Void in
        var unicornNames = OrderedSet(array: self.sortedFavoriteUnicornNames)
        self.sorted().setVocabularyStrings(unicornNames, of:
            INVocabularyStringType.contactName)
    })
}

...
}

```




9:41 AM

100% 

It's sent.



UNICORNCHAT

To: Scott

Are you ready for your presentation



Providing a User Interface with SiriKit

Scott Andrus SiriKit Engineering

UI Extensions Increase Your App's Impact



UI Extensions Increase Your App's Impact

Your view alongside Siri



UI Extensions Increase Your App's Impact

Your view alongside Siri

Experiences unique to your application



UI Extensions Increase Your App's Impact

Your view alongside Siri

Experiences unique to your application

User-specific customization



UI Extensions Increase Your App's Impact

Your view alongside Siri

Experiences unique to your application

User-specific customization

Information Siri might not otherwise show



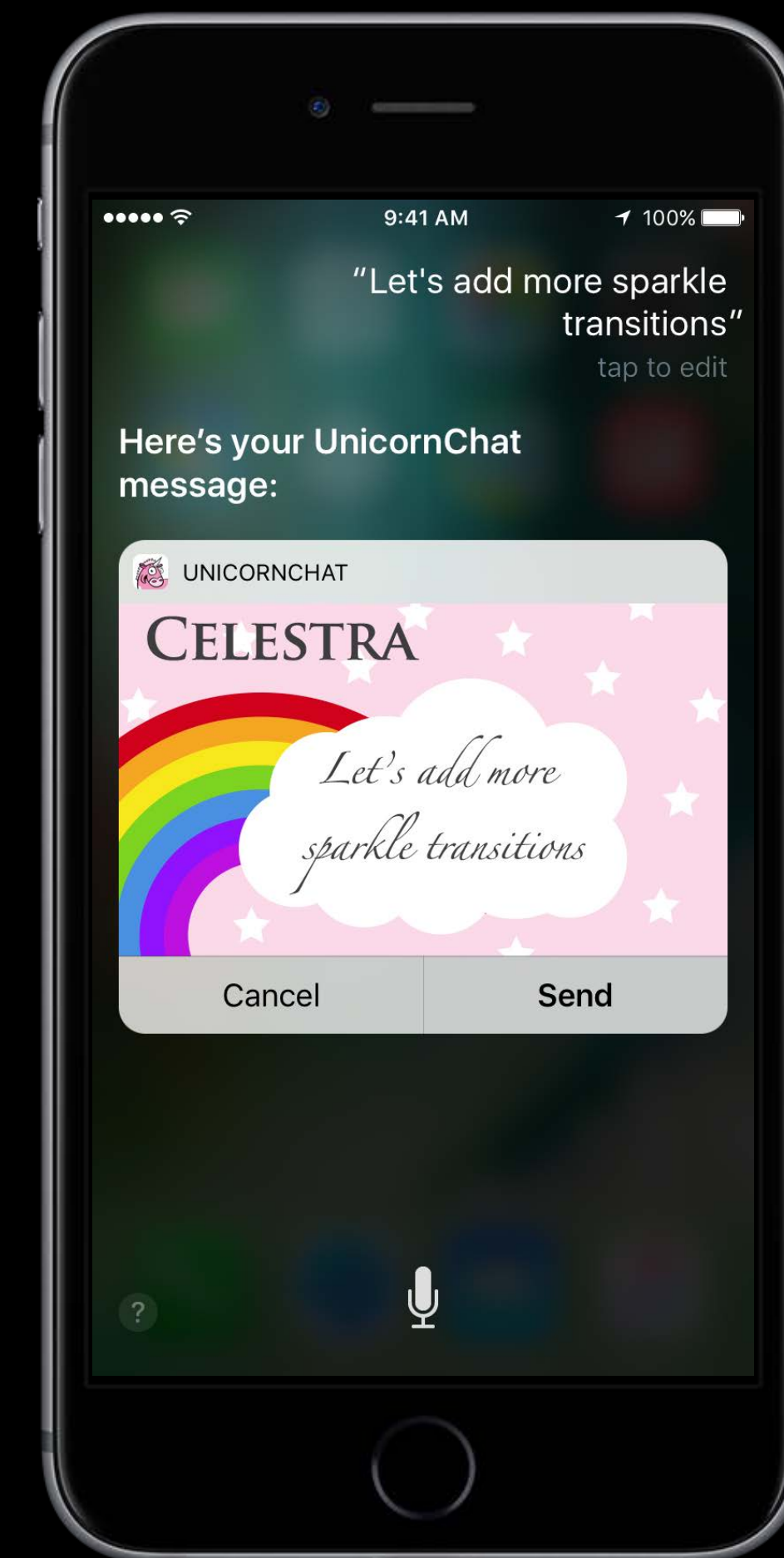
UI Extensions Increase Your App's Impact

Your view alongside Siri

Experiences unique to your application

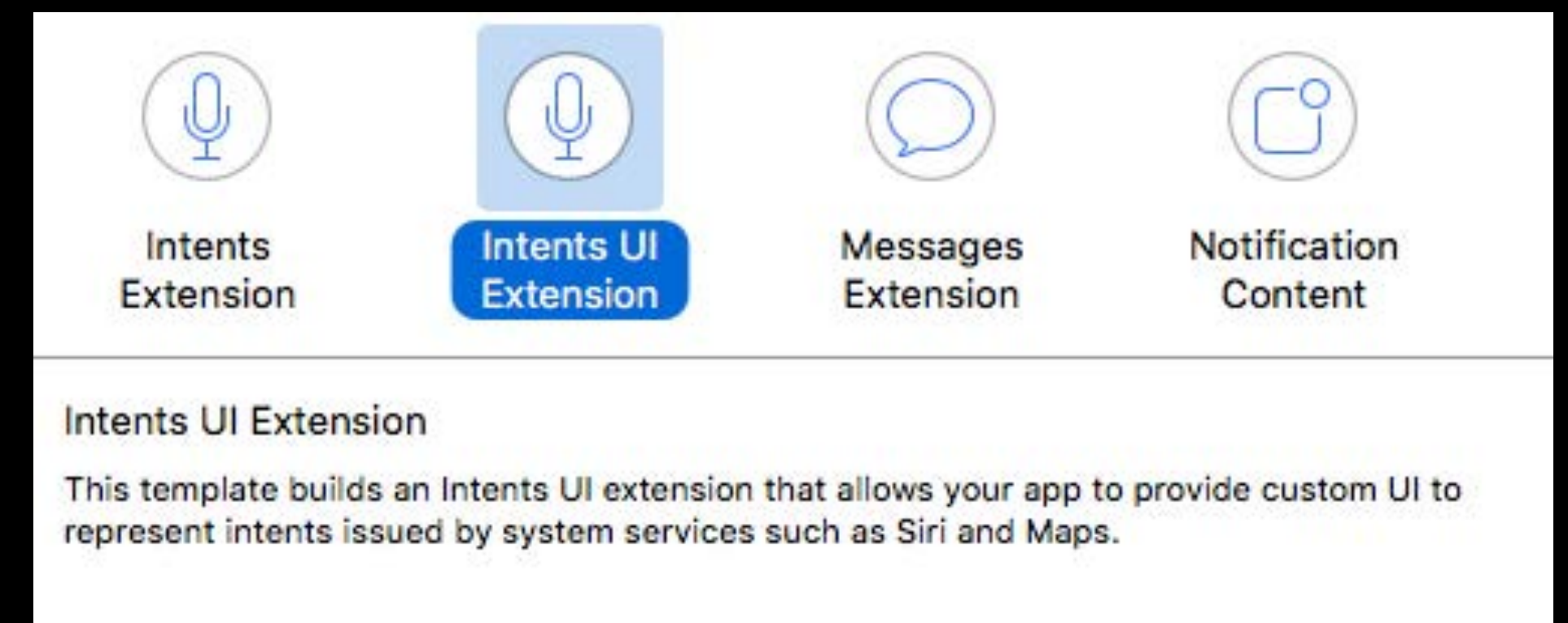
User-specific customization

Information Siri might not otherwise show



Add UI Extension Target to Xcode

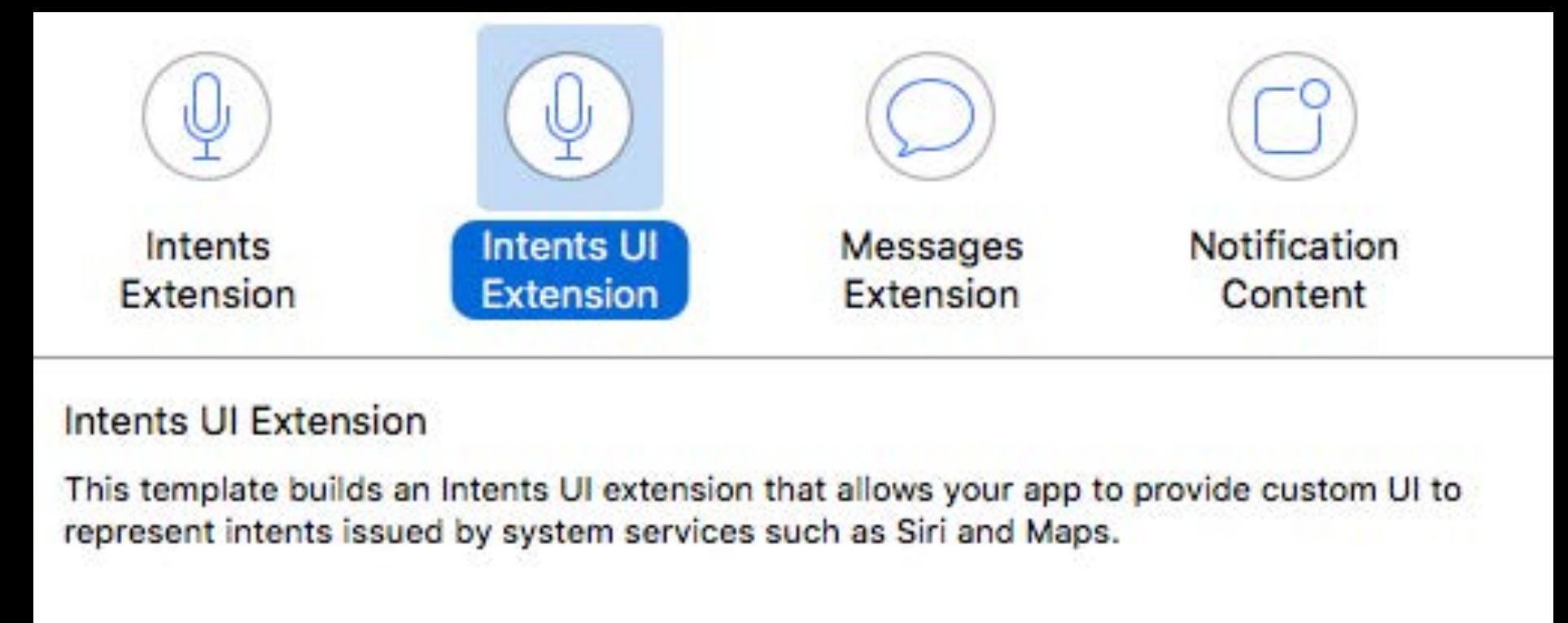
How to get started



Add UI Extension Target to Xcode

How to get started

Add Intents UI Extension target

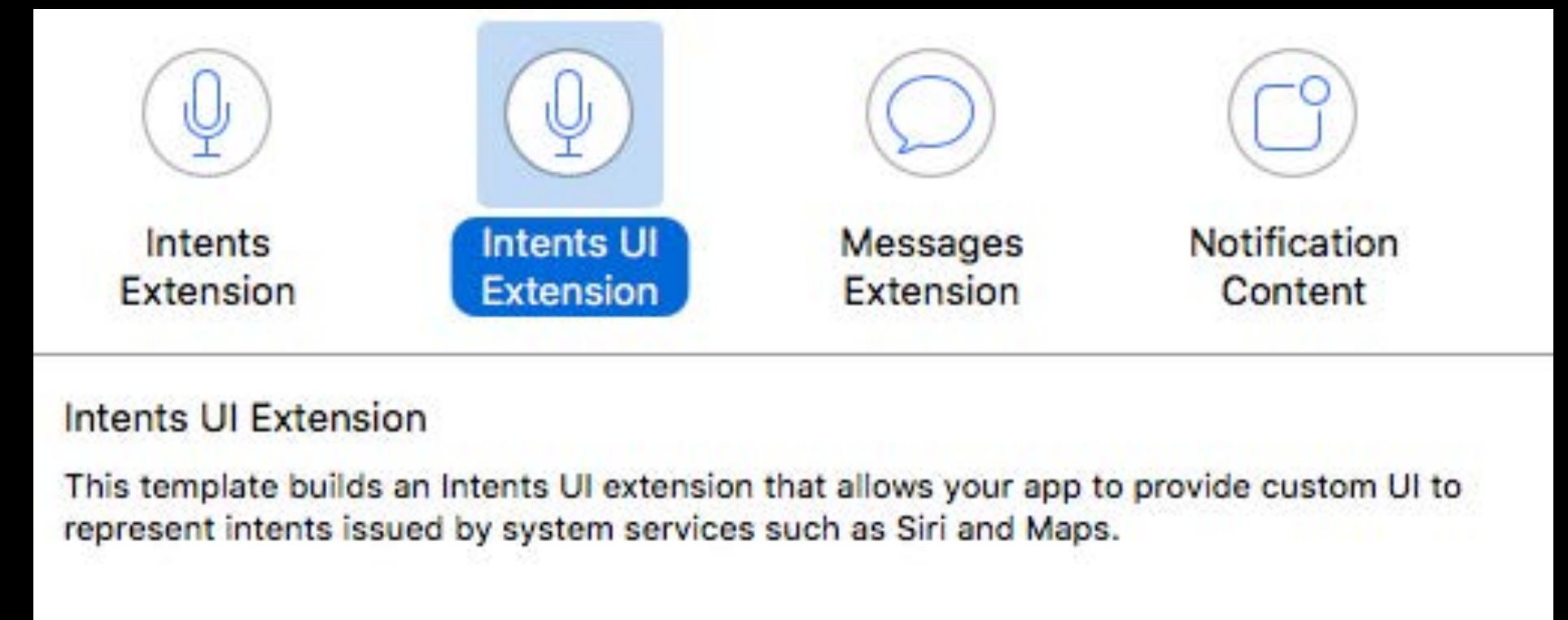


Add UI Extension Target to Xcode

How to get started

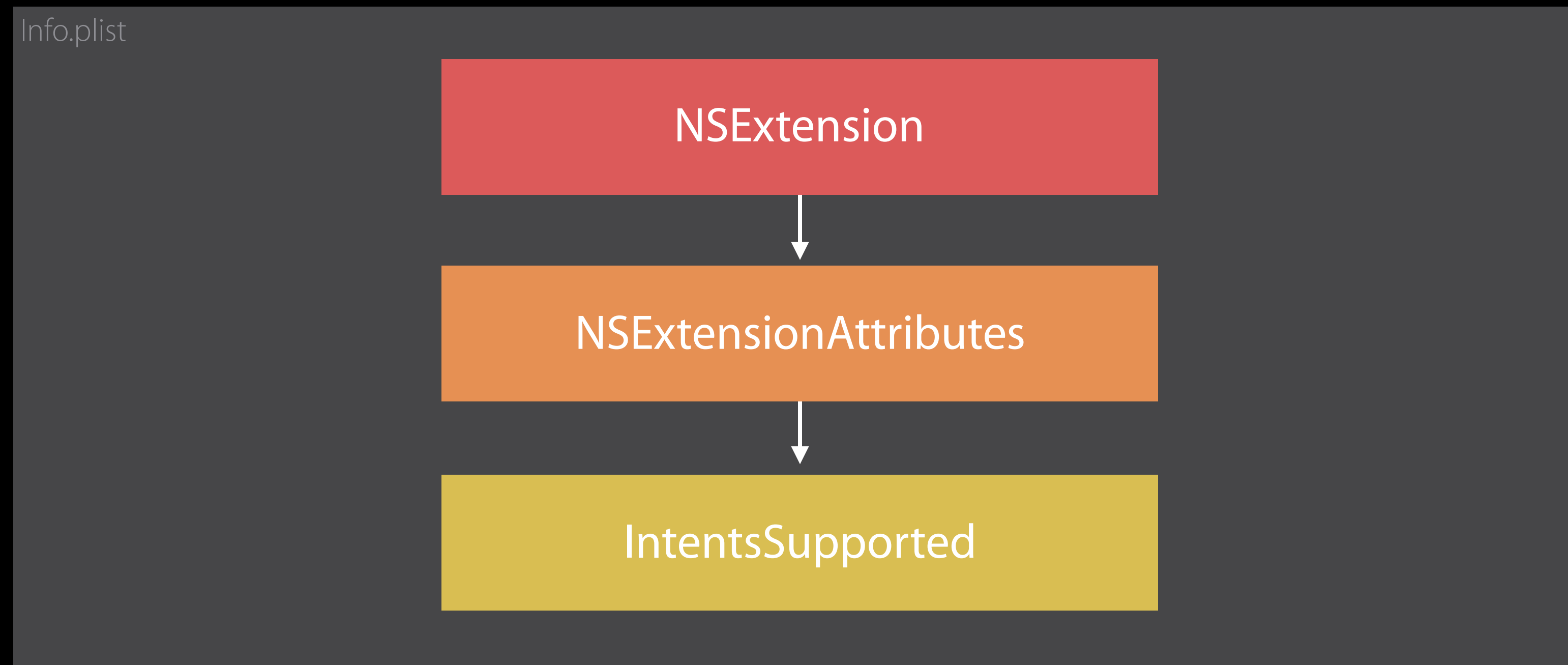
Add Intents UI Extension target

Embed in application



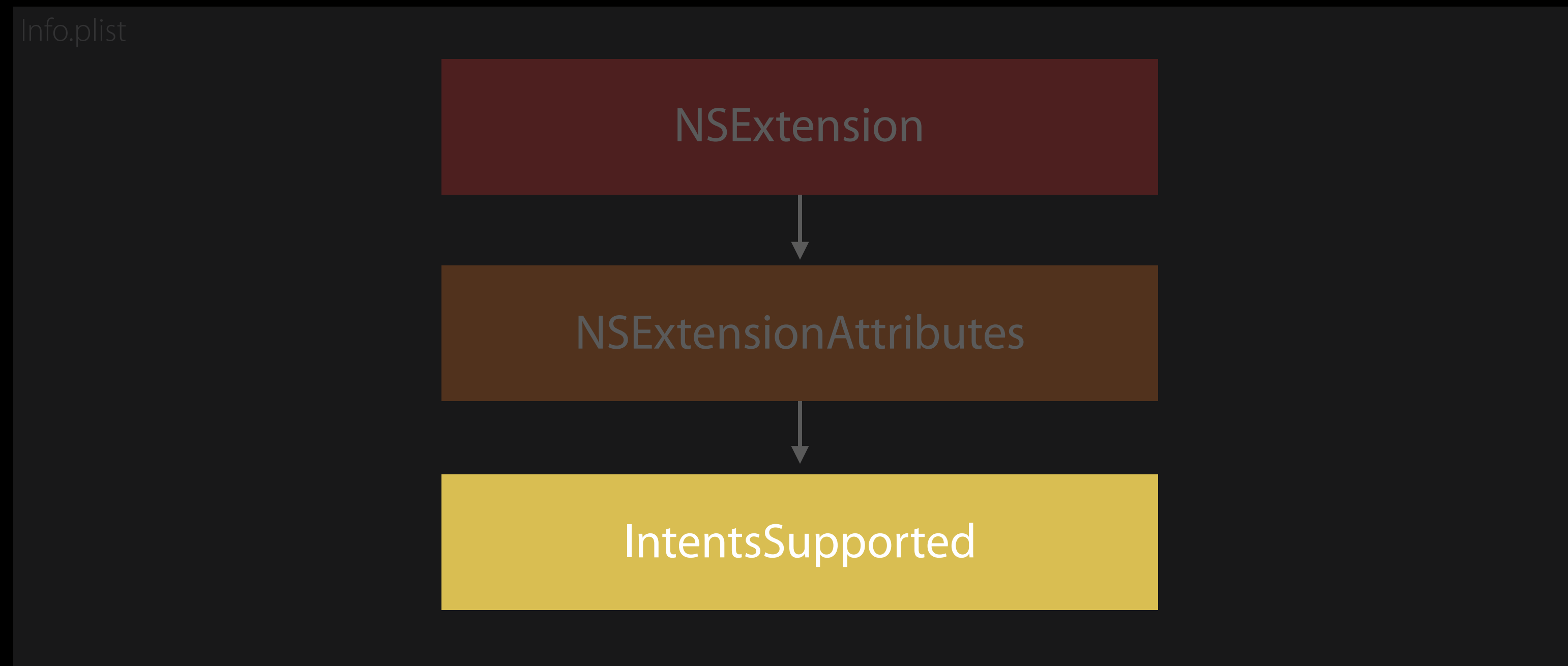
Intents UI Extension

Info.plist



Intents UI Extension

Info.plist



Siri



Siri



→
configure(with interaction:...)

Siri



→
configure(with interaction:...)

UI extension

Siri



→
configure(with interaction:...)

UI extension

UIViewController :
INUIHostedViewControlling

Siri



INInteraction

→
configure(with interaction:...)

UI extension

UIViewController :
INUIHostedViewControlling

INInteraction

Breaking it down

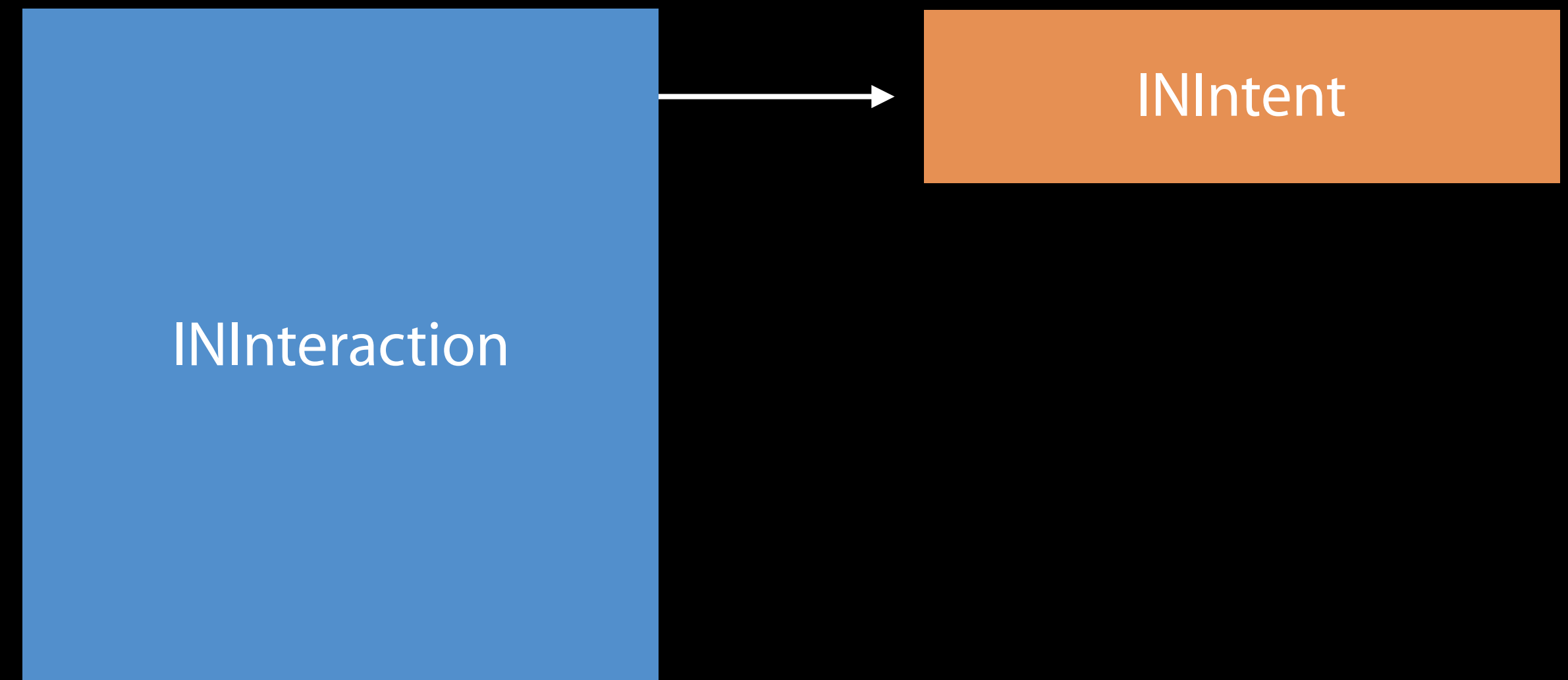


INInteraction

INInteraction

Breaking it down

The handled (or to-be-handled) INIntent

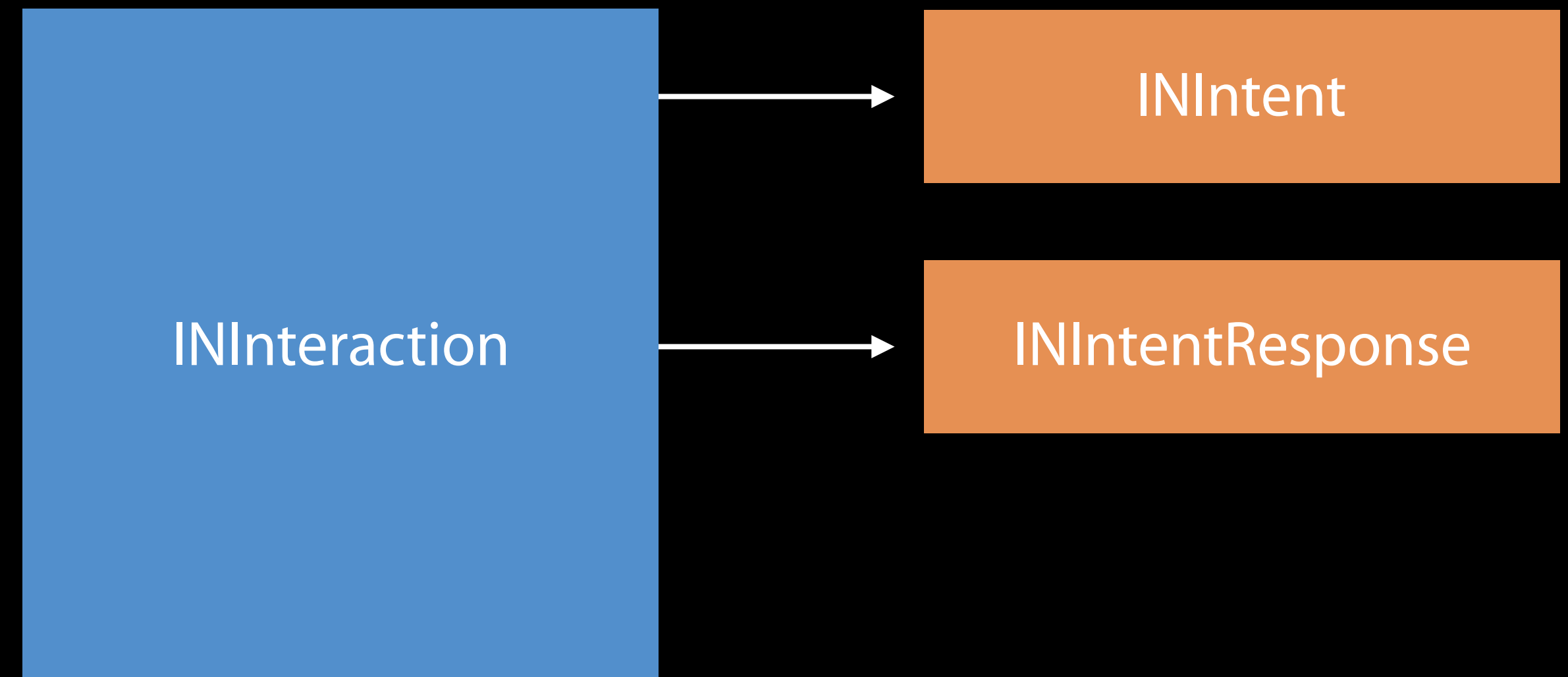


INInteraction

Breaking it down

The handled (or to-be-handled) INIntent

The provided INIntentResponse



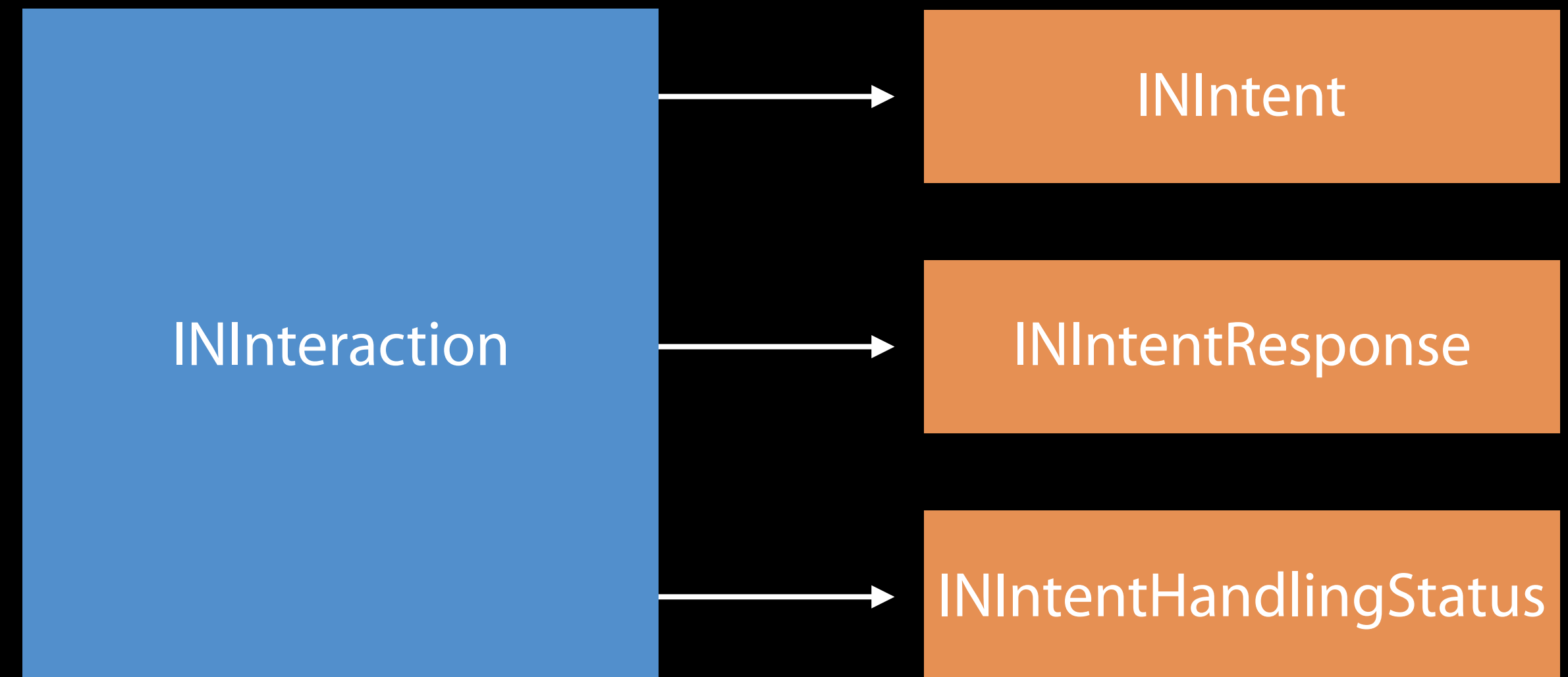
INInteraction

Breaking it down

The handled (or to-be-handled) INIntent

The provided INIntentResponse

INIntentHandlingStatus



Implementing Your View Controller

Implementing Your View Controller

Principal class

Implementing Your View Controller

Principal class

Subclass of UIViewController

Implementing Your View Controller

Principal class

Subclass of UIViewController

Configure with interaction

Implementing Your View Controller

Principal class

Subclass of UIViewController

Configure with interaction

Provided view context

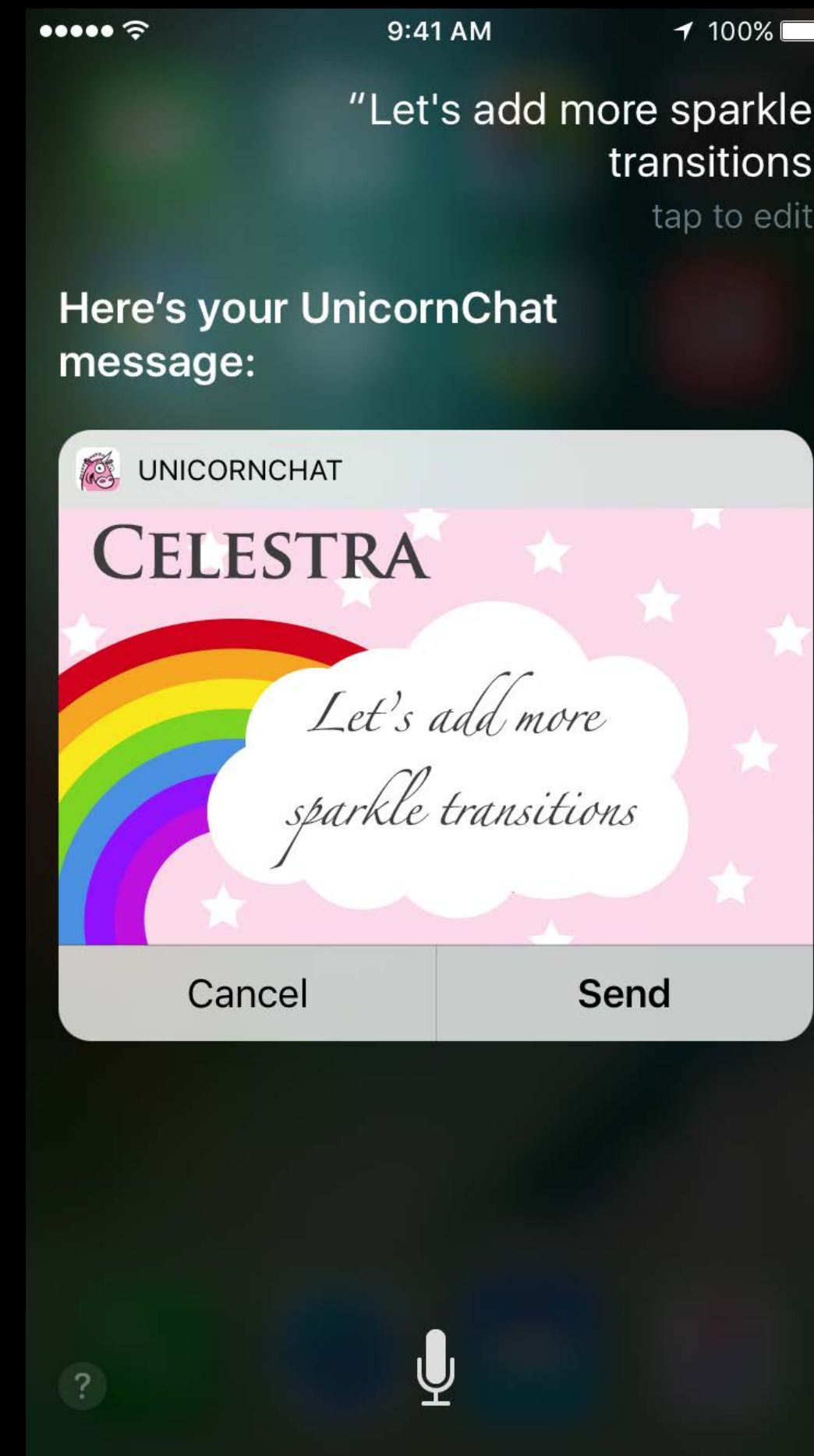
Implementing Your View Controller

Principal class

Subclass of UIViewController

Configure with interaction

Provided view context



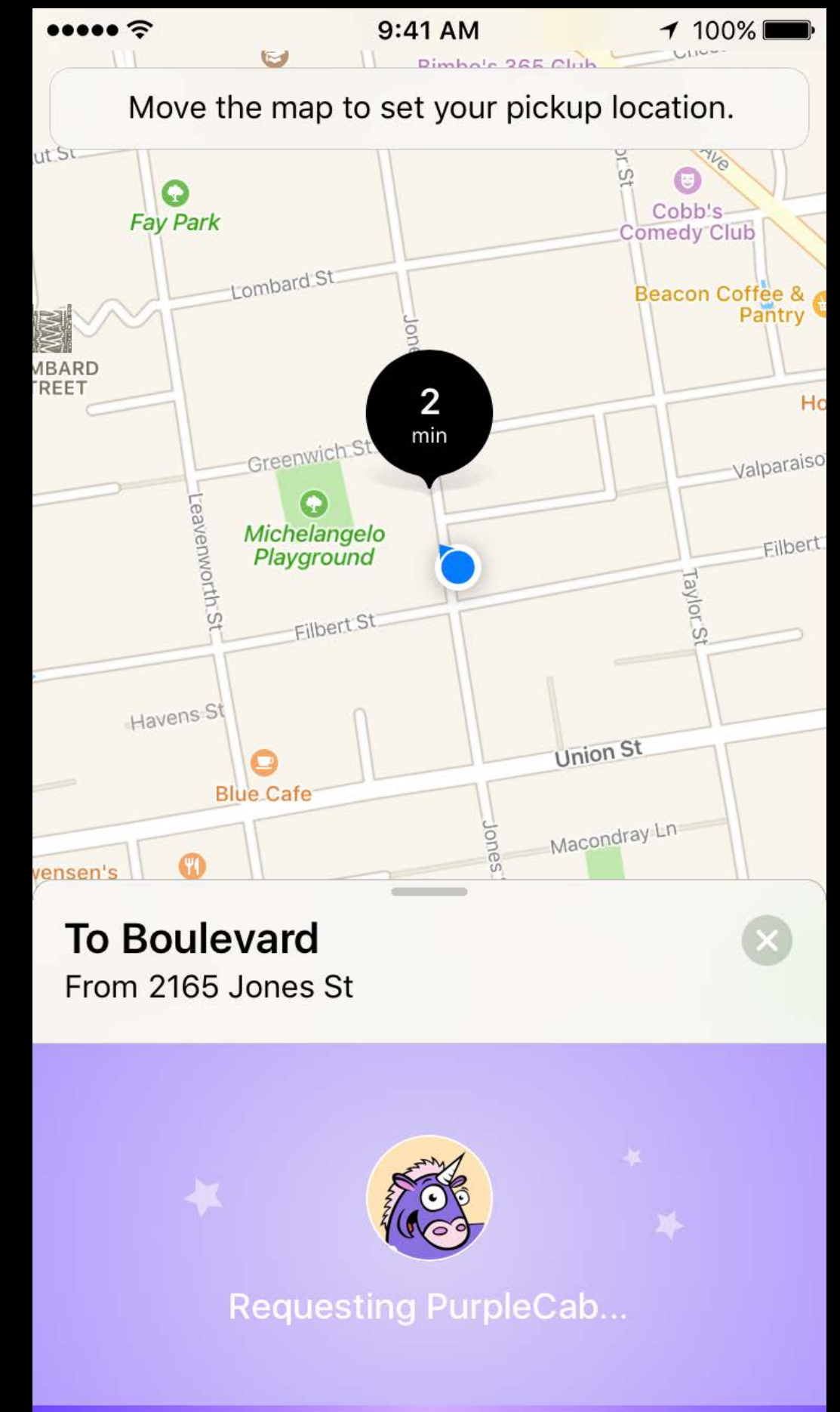
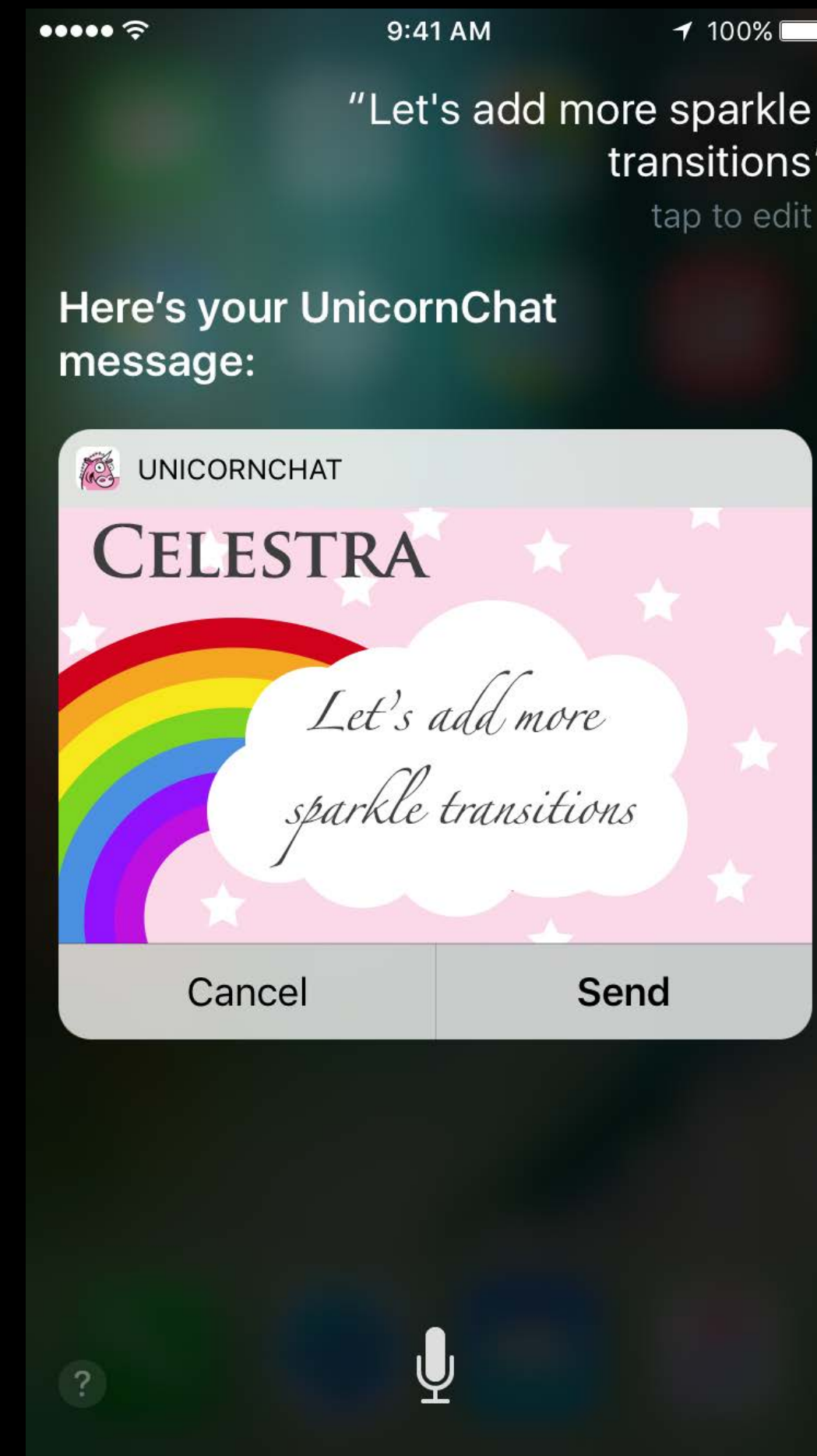
Implementing Your View Controller

Principal class

Subclass of UIViewController

Configure with interaction

Provided view context



Implementing Your View Controller

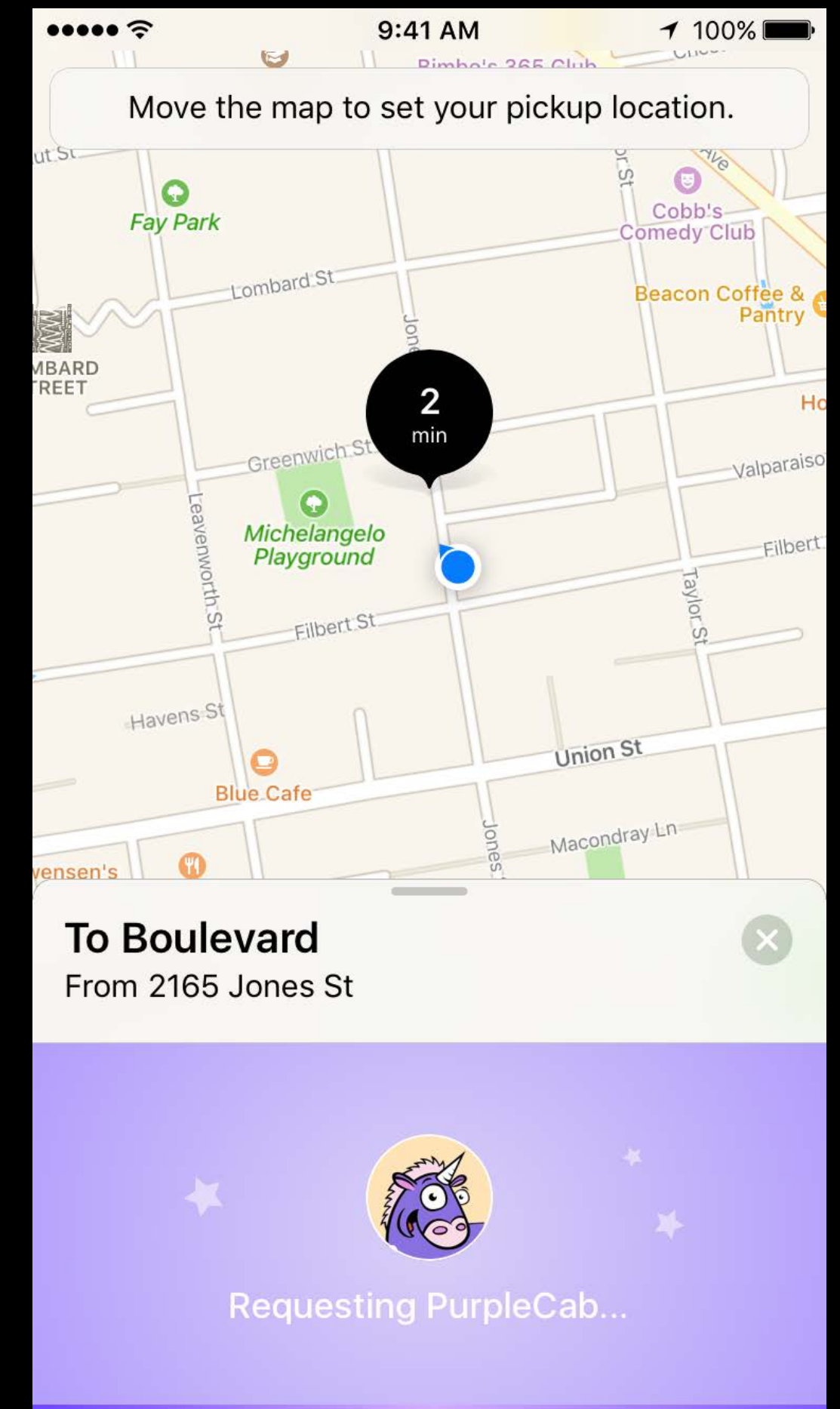
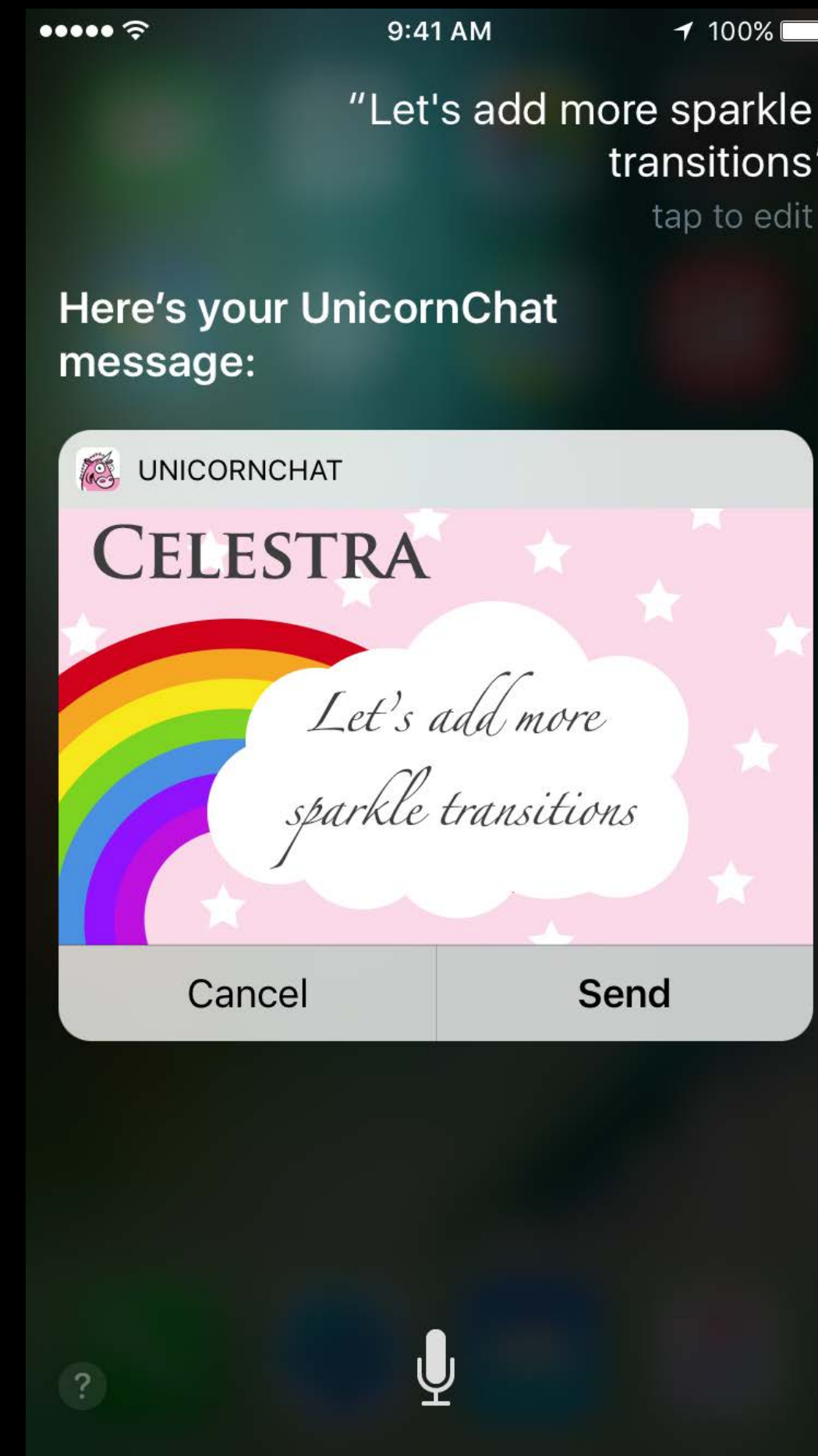
Principal class

Subclass of UIViewController

Configure with interaction

Provided view context

Desired size in the completion



Demo

Building a SiriKit UI extension



9:41 AM

100%

Here's your UnicornChat
message:



UNICORNCAT

Diana

Great job on your
presentation today

To: Diana

Great job on your presentation today

Cancel

Send

Siri Gives You Override Control

Siri Gives You Override Control

Implement `INUIHostedViewSiriProviding`

Siri Gives You Override Control

Implement `INUIHostedViewSiriProviding`

Properties for displaying messages and maps

Siri Gives You Override Control

Implement `INUIHostedViewSiriProviding`

Properties for displaying messages and maps

Opt-in to displaying a particular interface

Siri Gives You Override Control

Implement `INUIHostedViewSiriProviding`

Properties for displaying messages and maps

Opt-in to displaying a particular interface

Siri will accommodate your view's content

```
class IntentViewController: UIViewController, INUIHostedViewControlling,
    func configure(with interaction: INInteraction!, context: INUIHostedViewContext,
completion: ((CGSize) -> Void)!) {
    // Configure your view
    completion(self.extensionContext!.hostedViewMaximumAllowedSize)
}
}
```



```
class IntentViewController: UIViewController, INUIHostedViewControlling,
INUIHostedViewSiriProviding {
    func configure(with interaction: INInteraction!, context: INUIHostedViewContext,
completion: ((CGSize) -> Void)!) {
        // Configure your view
        completion(self.extensionContext!.hostedViewMaximumAllowedSize)
    }
    var displaysMessage: Bool {
        return true
    }
}
```

```
class IntentViewController: UIViewController, INUIHostedViewControlling,  
INUIHostedViewSiriProviding {  
    func configure(with interaction: INInteraction!, context: INUIHostedViewContext,  
completion: ((CGSize) -> Void)!) {  
        // Configure your view  
        completion(self.extensionContext!.hostedViewMaximumAllowedSize)  
    }  
    var displaysMessage: Bool {  
        return true  
    }  
}
```



```
class IntentViewController: UIViewController, INUIHostedViewControlling,  
INUIHostedViewSiriProviding {  
    func configure(with interaction: INInteraction!, context: INUIHostedViewContext,  
completion: ((CGSize) -> Void)!) {  
        // Configure your view  
        completion(self.extensionContext!.hostedViewMaximumAllowedSize)  
    }  
    var displaysMessage: Bool {  
        return true  
    }  
}
```

Demo

Telling Siri about our interface



9:41 AM

100%

"Send a message to Diana using UnicornChat that says it's pretty tough to type demo code with unicorn hands"

tap to edit

Here's your UnicornChat message:



UNICORNCHAT

Diana

It's pretty tough to
type demo code
with unicorn hands

Cancel

Send





9:41 AM

100%

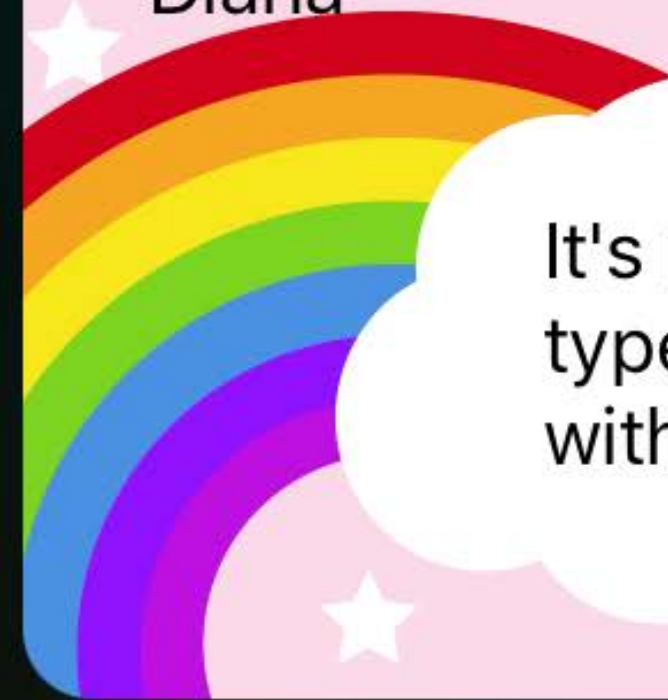
"Yeah"
tap to edit

It's sent.



UNICORNCHAT

Diana



It's pretty tough to
type demo code
with unicorn hands



Considerations

Considerations

Be memory conscious

Considerations

Be memory conscious

Minimum and maximum view sizes

Considerations

Be memory conscious

Minimum and maximum view sizes

Flexible and adaptive layout patterns

Summary

Summary

Prepare to adopt SiriKit

Summary

Prepare to adopt SiriKit

Add your first Intents extension

Summary

Prepare to adopt SiriKit

Add your first Intents extension

Provide a user interface in Siri

More Information

<https://developer.apple.com/wwdc16/225>

Related Sessions

Introducing SiriKit

Presidio

Wednesday 5:00PM

App Extension Best Practices

WWDC 2015

Labs

SiriKit Lab	Frameworks Lab C	Thursday 3:00PM
SiriKit Lab	Frameworks Lab B	Friday 9:00AM

