#WWDC18

Building for Voice with Siri Shortcuts

Session 214

Amit Jain, Siri
Ayaka Nonaka, Siri

Adding shortcuts to Siri

Custom responses

Best practices for shortcuts

Bringing shortcuts into your app

Adding shortcuts to Siri

Custom responses

Best practices for shortcuts

Bringing shortcuts into your app

Adding shortcuts to Siri

Custom responses

Best practices for shortcuts

Bringing shortcuts into your app

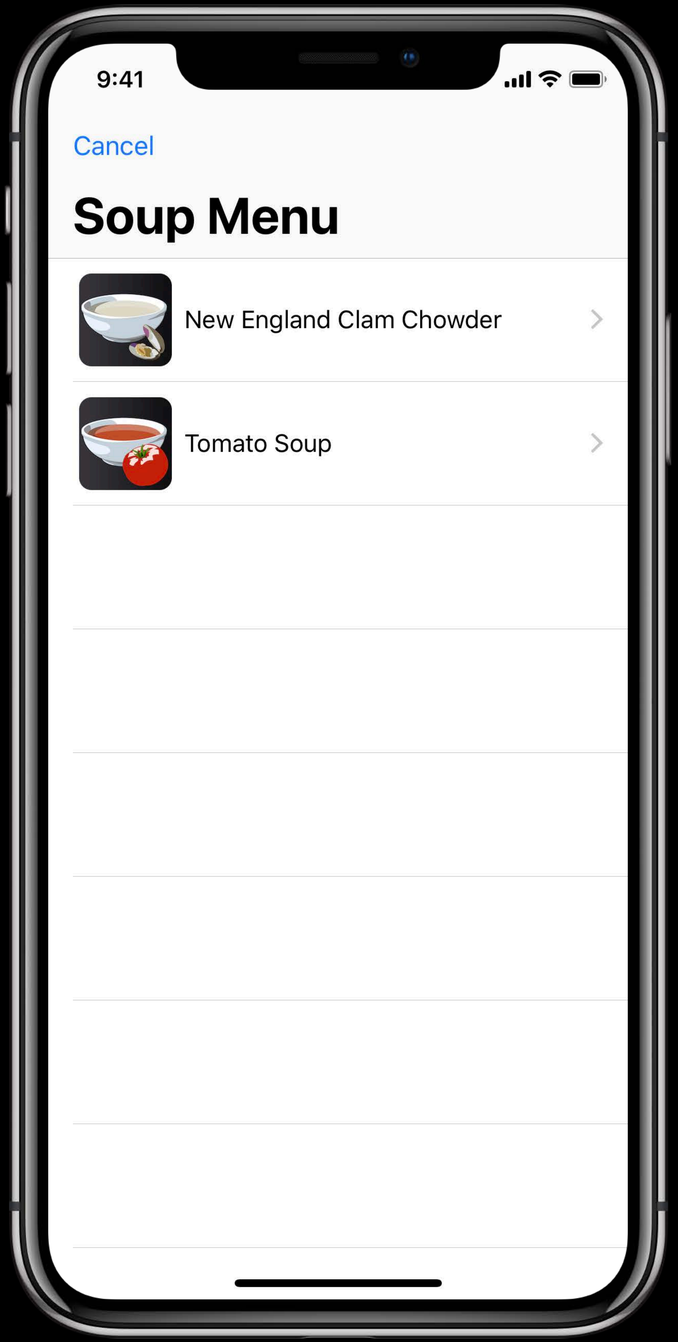
Adding shortcuts to Siri

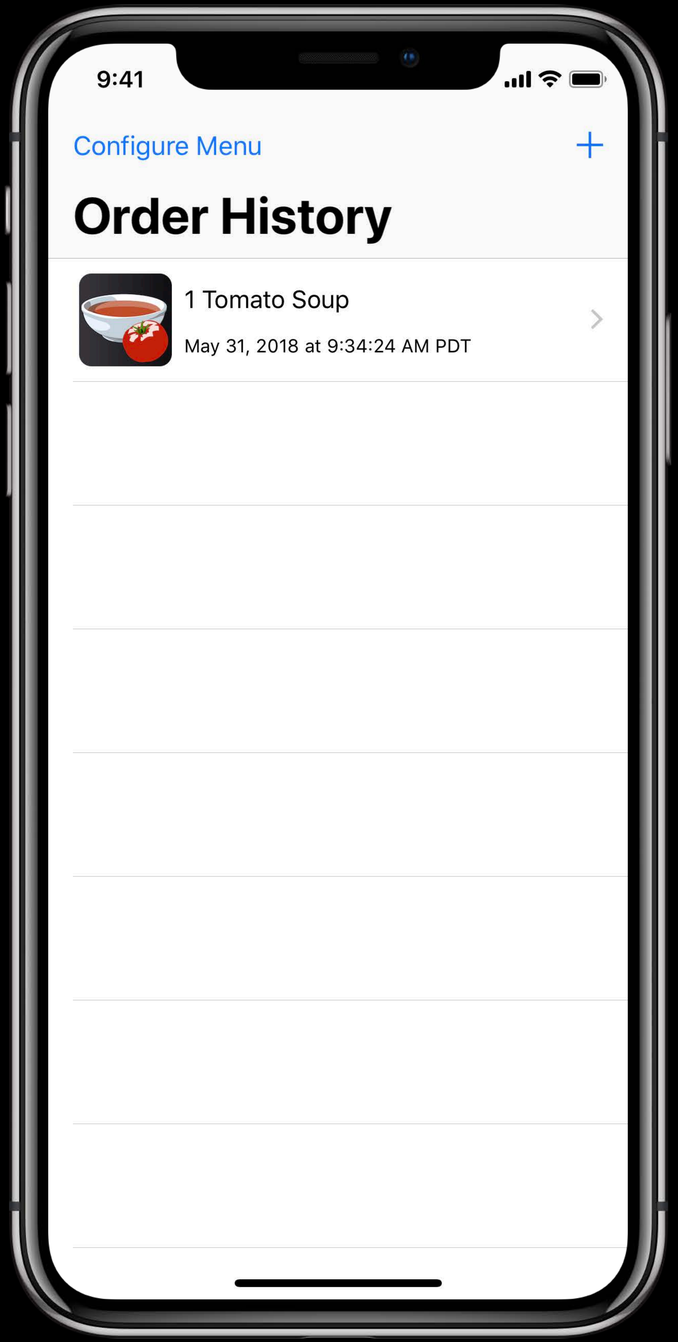
Custom responses

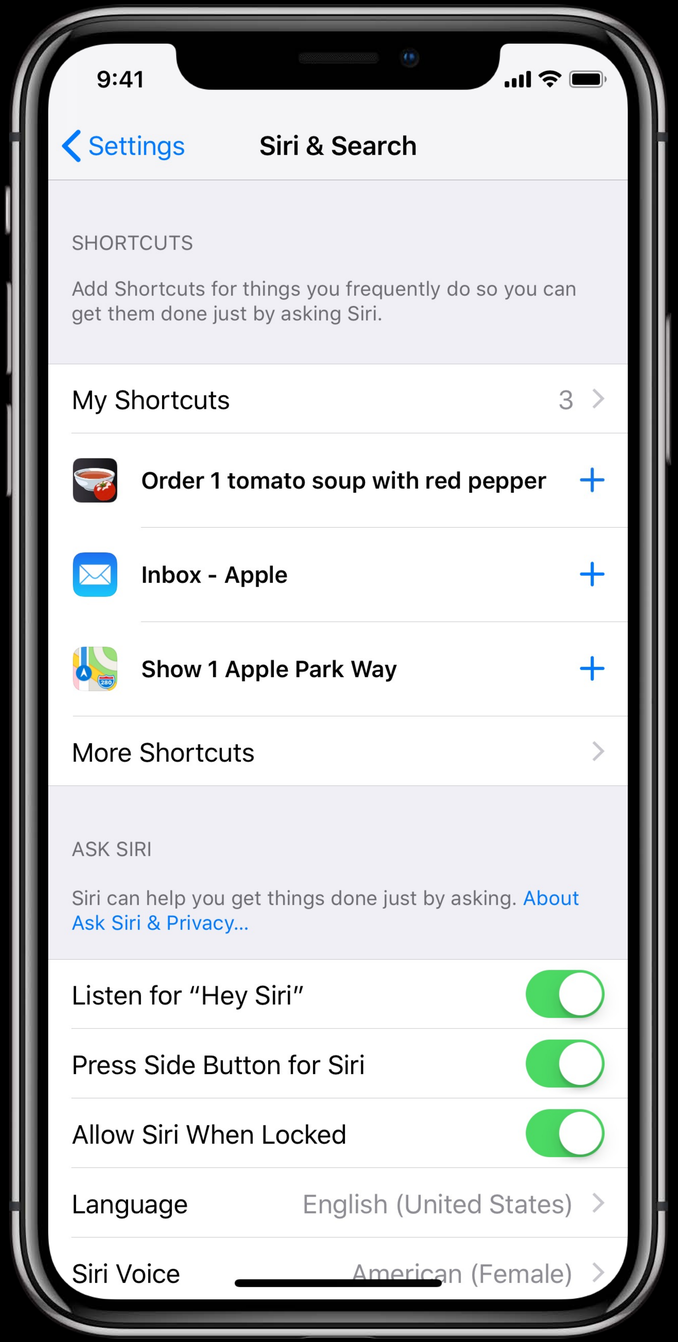
Best practices for shortcuts

Bringing shortcuts into your app

Adding Shortcuts to Siri


9:41
Cancel
Soup Menu
New England Clam Chowder
Tomato Soup

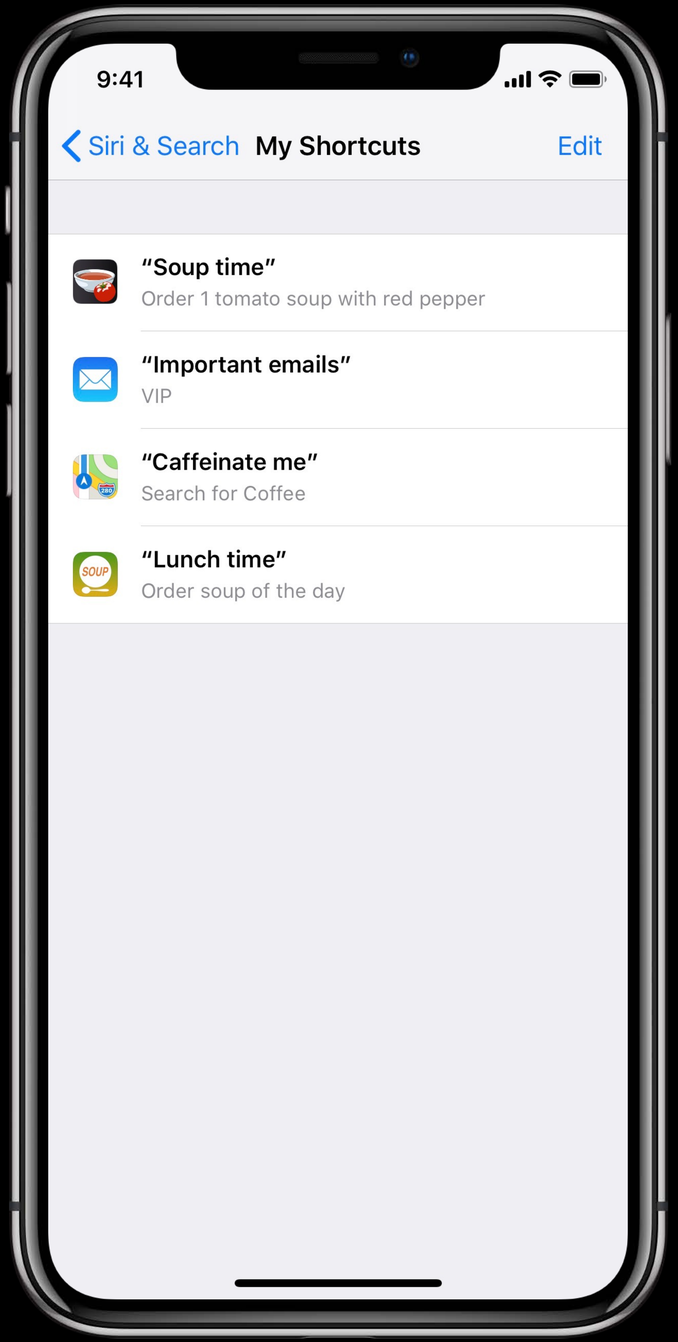

9:41
Configure Menu
Order History
1 Tomato Soup
May 31, 2018 at 9:34:24 AM PDT

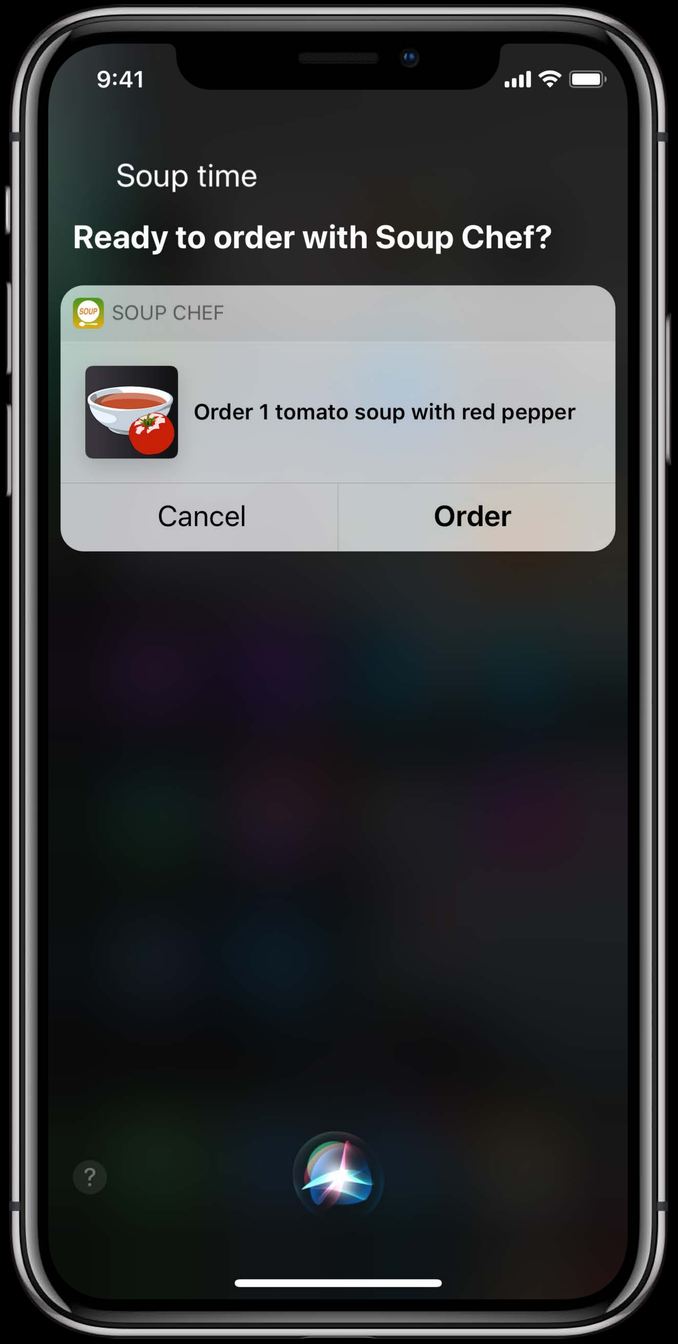

9:41
Settings
Siri & Search
SHORTCUTS
Add Shortcuts for things you frequently do so you can get them done just by asking Siri.
My Shortcuts
3
Order 1 tomato soup with red pepper
Inbox - Apple
Show 1 Apple Park Way
More Shortcuts
ASK SIRI
Siri can help you get things done just by asking. About Ask Siri & Privacy...
Listen for “Hey Siri”
Press Side Button for Siri
Allow Siri When Locked
Language
English (United States)
Siri Voice
American (Female)

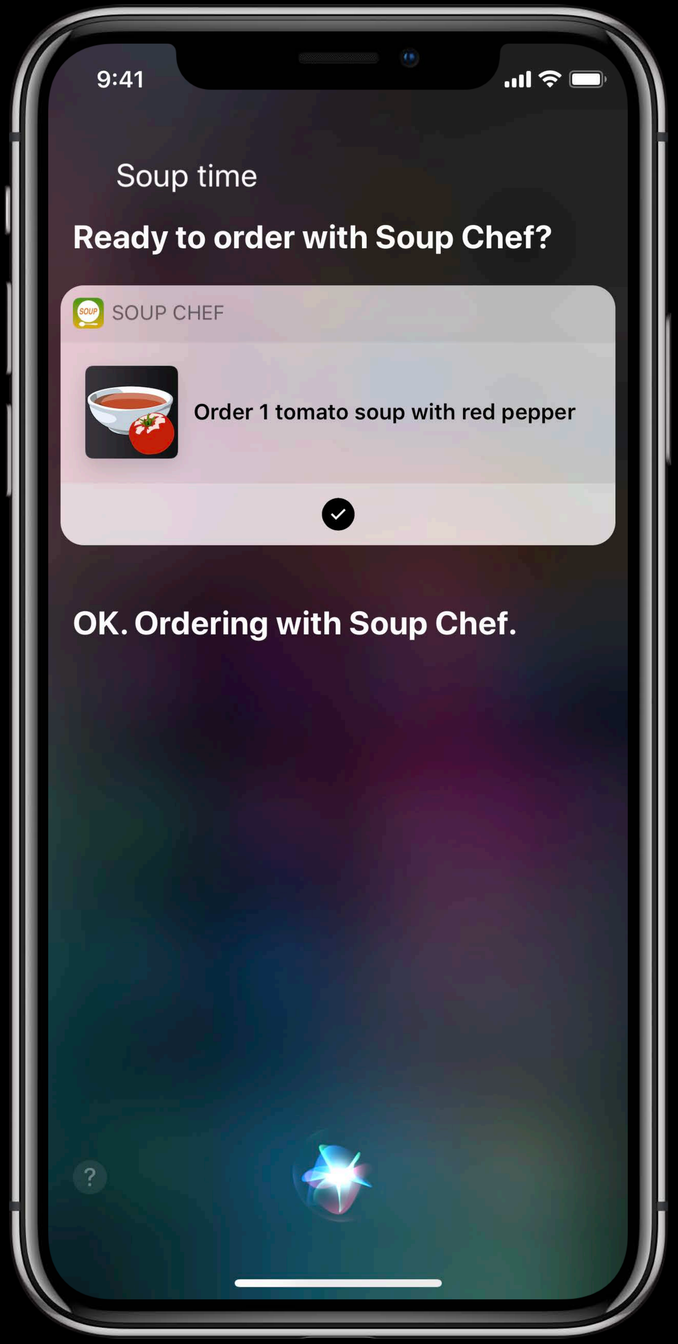

9:41
Cancel
Add to Siri
Order 1 tomato soup with red pepper
Soup Chef
You can say something like...
“Soup time”
After you record your personalized phrase, Siri can use it to tell “Soup Chef” to run this shortcut.


9:41
Cancel
“Soup time”


9:41
Cancel
Done
Order 1 tomato soup with red pepper
Soup Chef
“Soup time”
Edit


9:41
Siri & Search
My Shortcuts
Edit
“Soup time”
Order 1 tomato soup with red pepper
“Important emails”
VIP
“Caffeinate me”
Search for Coffee
“Lunch time”
Order soup of the day
SOUP


9:41
Soup time
Ready to order with Soup Chef?
SOUP CHEF
Order 1 tomato soup with red pepper
Cancel
Order


9:41
Soup time
Ready to order with Soup Chef?
SOUP CHEF
Order 1 tomato soup with red pepper
OK. Ordering with Soup Chef.

NEW

Custom Responses

Types of custom responses

Confirmation

Success

Informational

Error

Types of custom responses

Confirmation

Bean says: “Your coffee will be $4.95.” Ready to order?

Types of custom responses

Success

Bean says: “You can pick up your latte in 10 minutes.”

Types of custom responses

Information

Commute says: “35 Eureka will be at the Castro and 17th stop in 4 minutes.”

Types of custom responses

Error

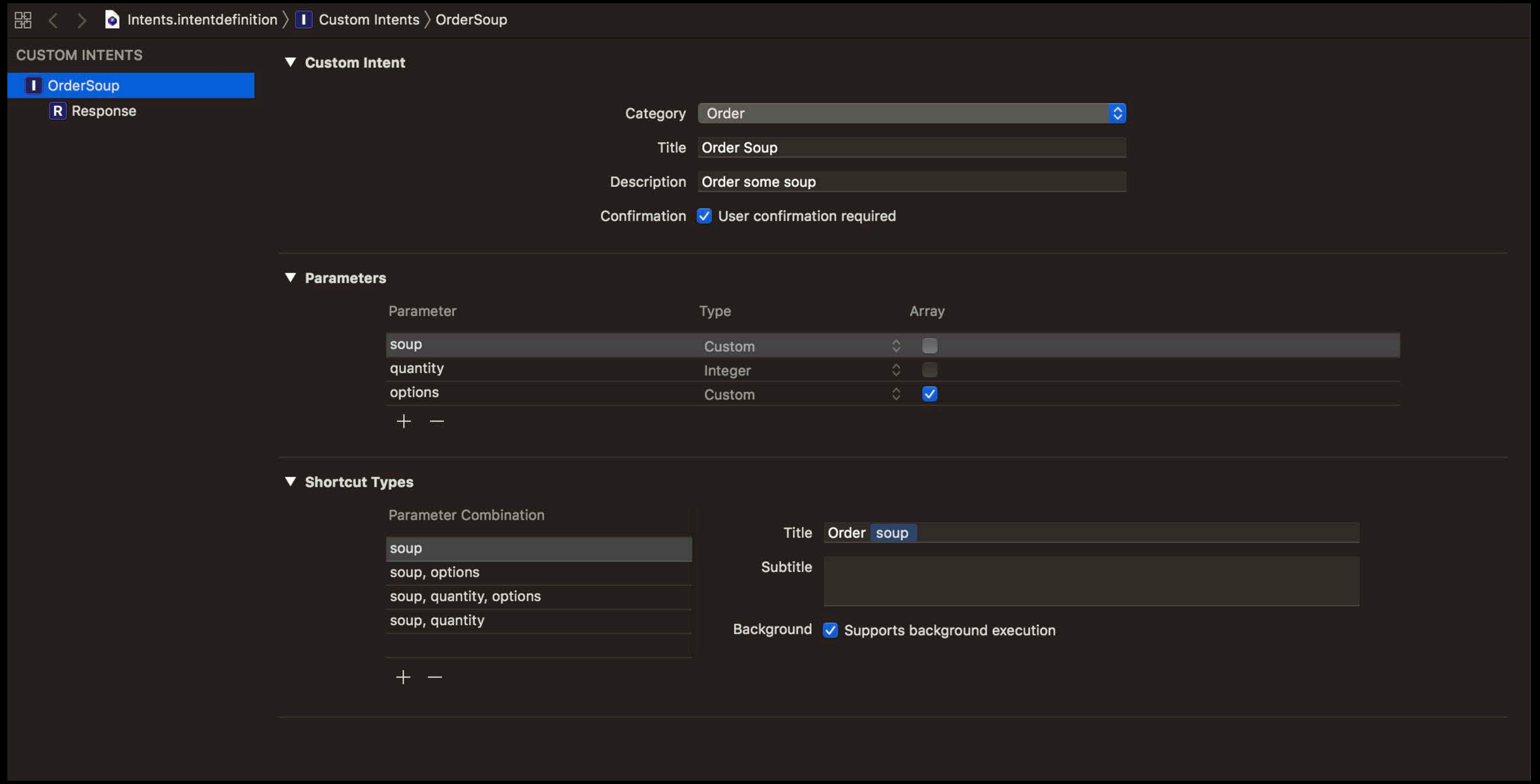
Something went wrong.
Bean says, “The Cupertino store is currently closed.”
You can continue in the app.

Adopting Custom Responses

Adopting Custom Responses

Defining a custom response

Define a custom intent


Intents.intentdefinition
Custom Intents
OrderSoup
CUSTOM INTENTS
OrderSoup
Response
Custom Intent
Category
Order
Title
Order Soup
Description
Order some soup
Confirmation
User confirmation required
Parameters
Parameter
Type
Array
soup
Custom
quantity
Integer
options
Custom
Shortcut Types
Parameter Combination
soup
soup, options
soup, quantity, options
soup, quantity
Title
Order
soup
Subtitle
Background
Supports background execution

Adopting Custom Responses

Defining a custom response

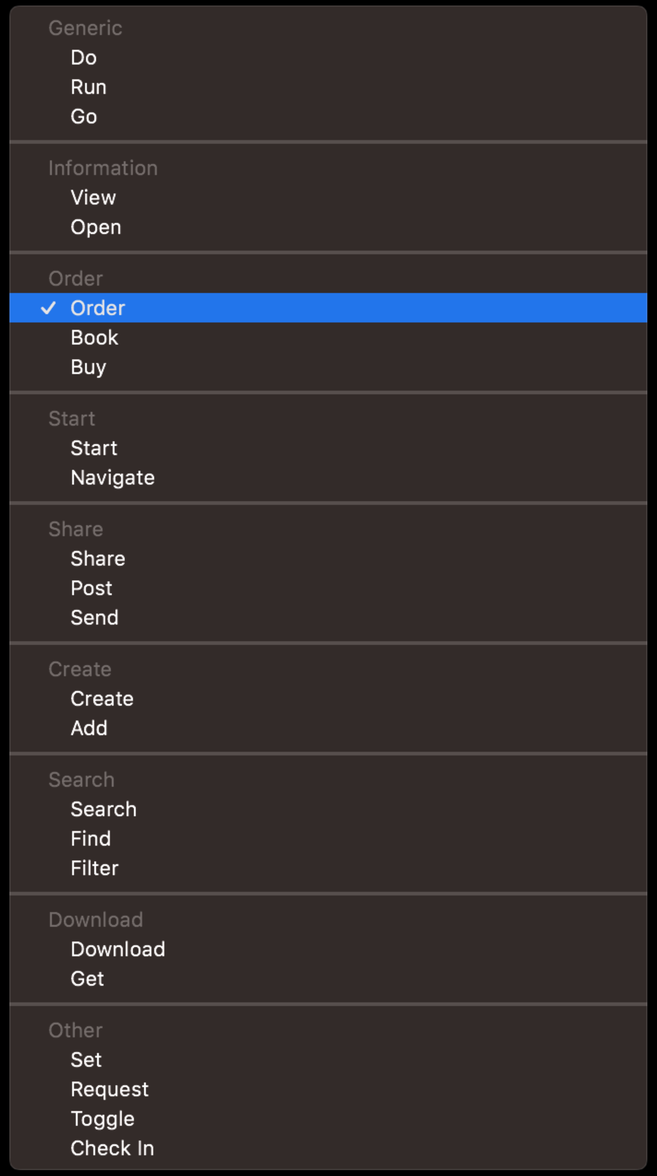
Define a custom intent

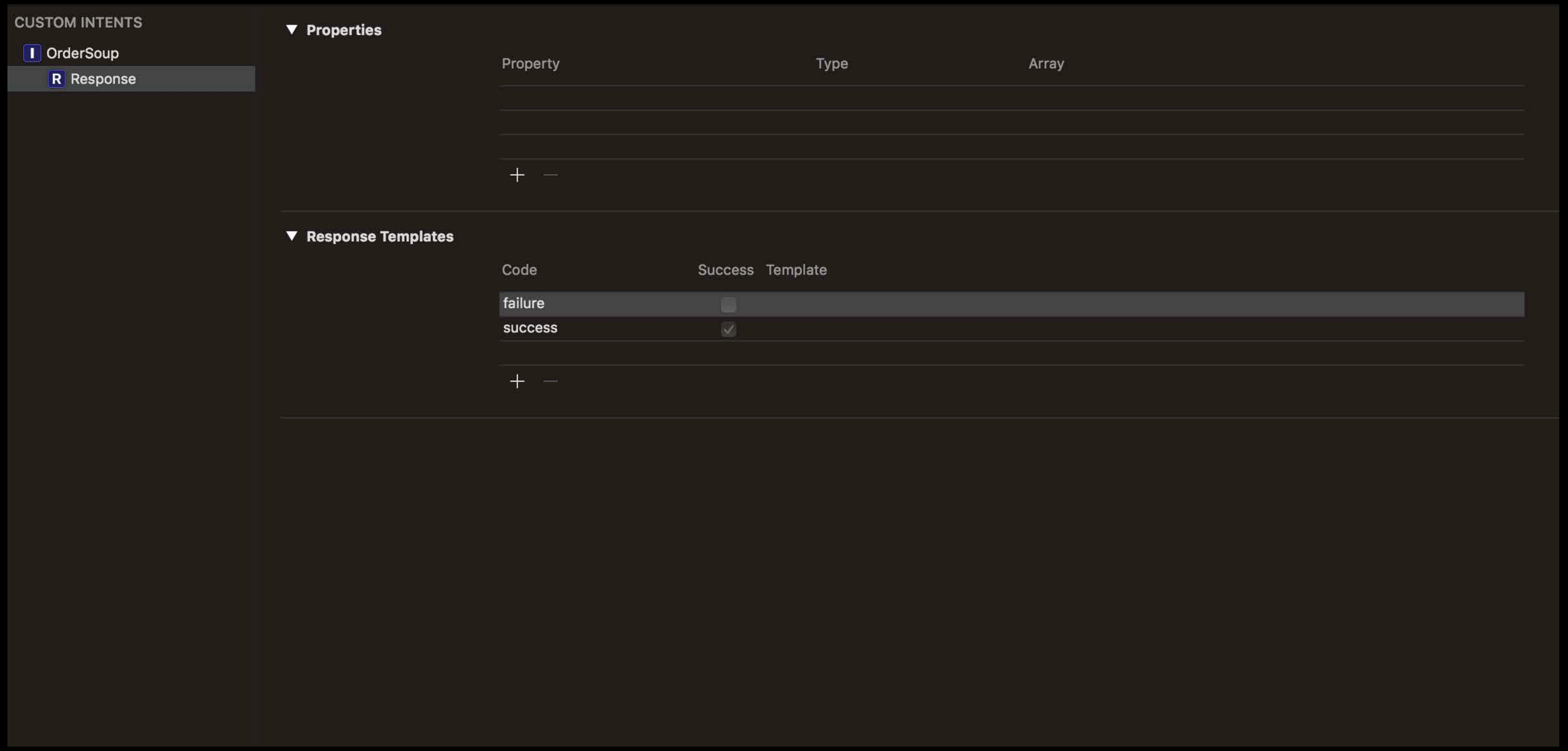
Adopting Custom Responses

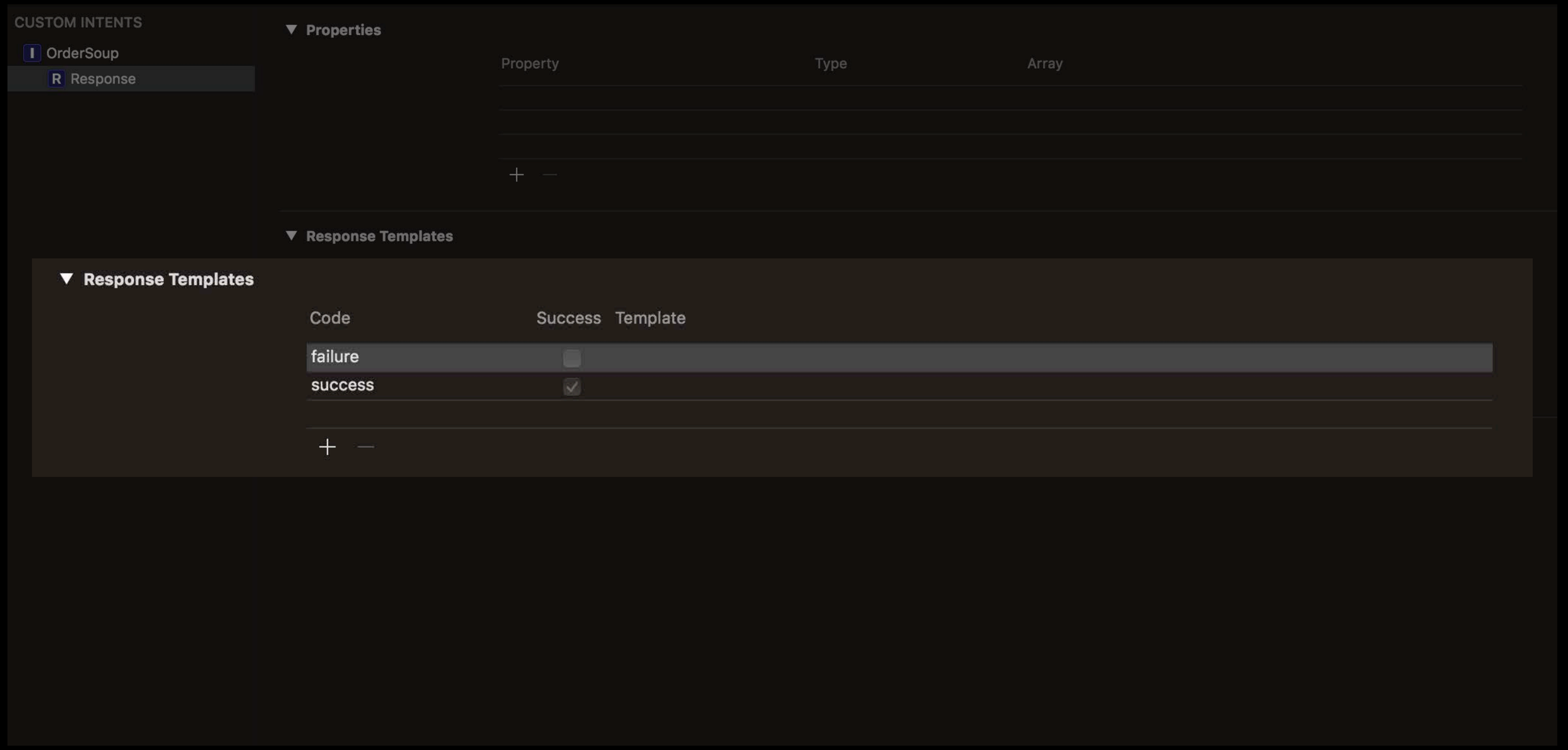
Defining a custom response

Define a custom intent

Select a category for your intent




CUSTOM INTENTS
I OrderSoup
R Response
Properties
Property
Type
Array
Response Templates
Code
Success
Template
failure
success


CUSTOM INTENTS
I OrderSoup
R Response
Properties
Property
Type
Array
Response Templates
Response Templates
Code
Success
Template
failure
success

Adopting Custom Responses

Defining a custom response

Define a custom intent

Select a category for your intent

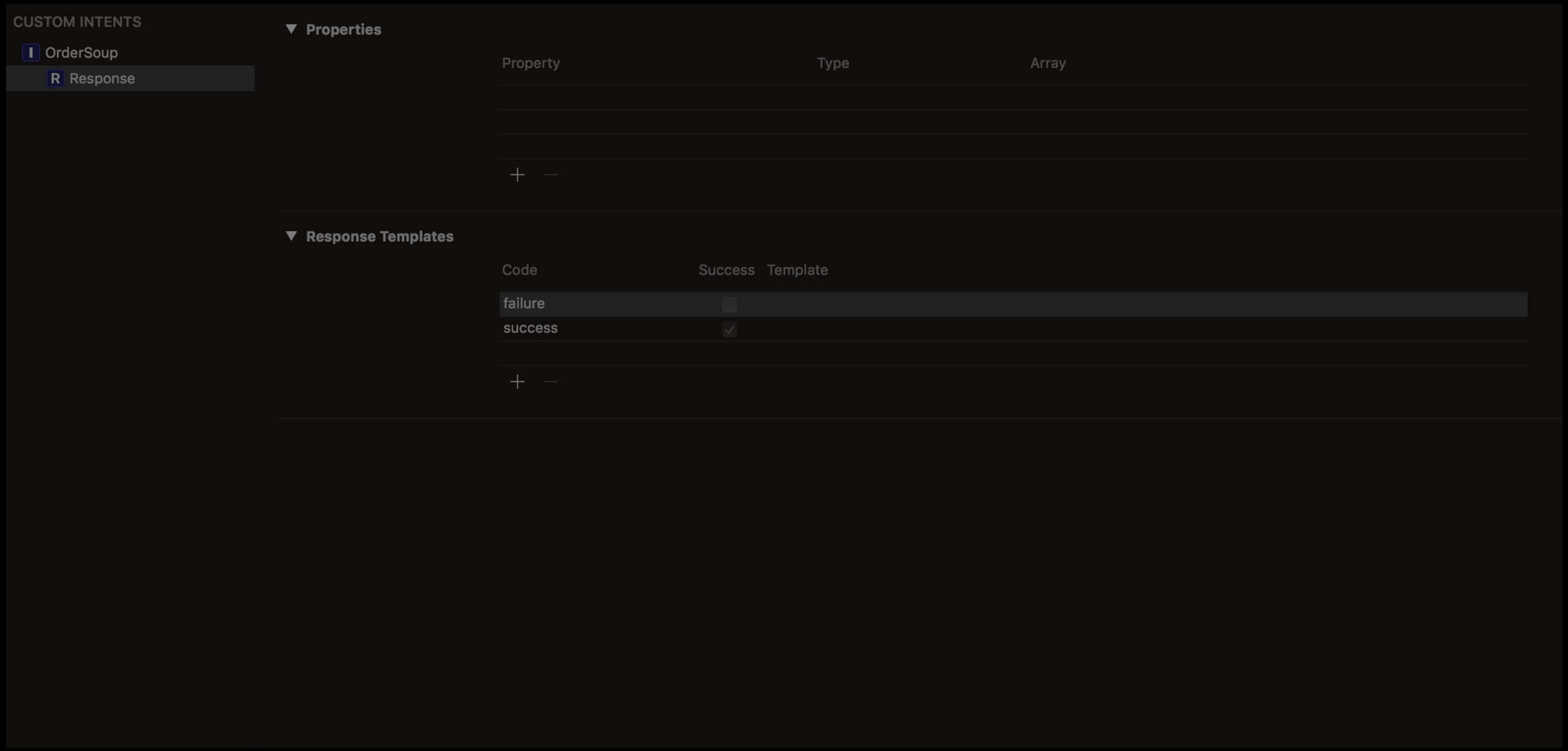
Adopting Custom Responses

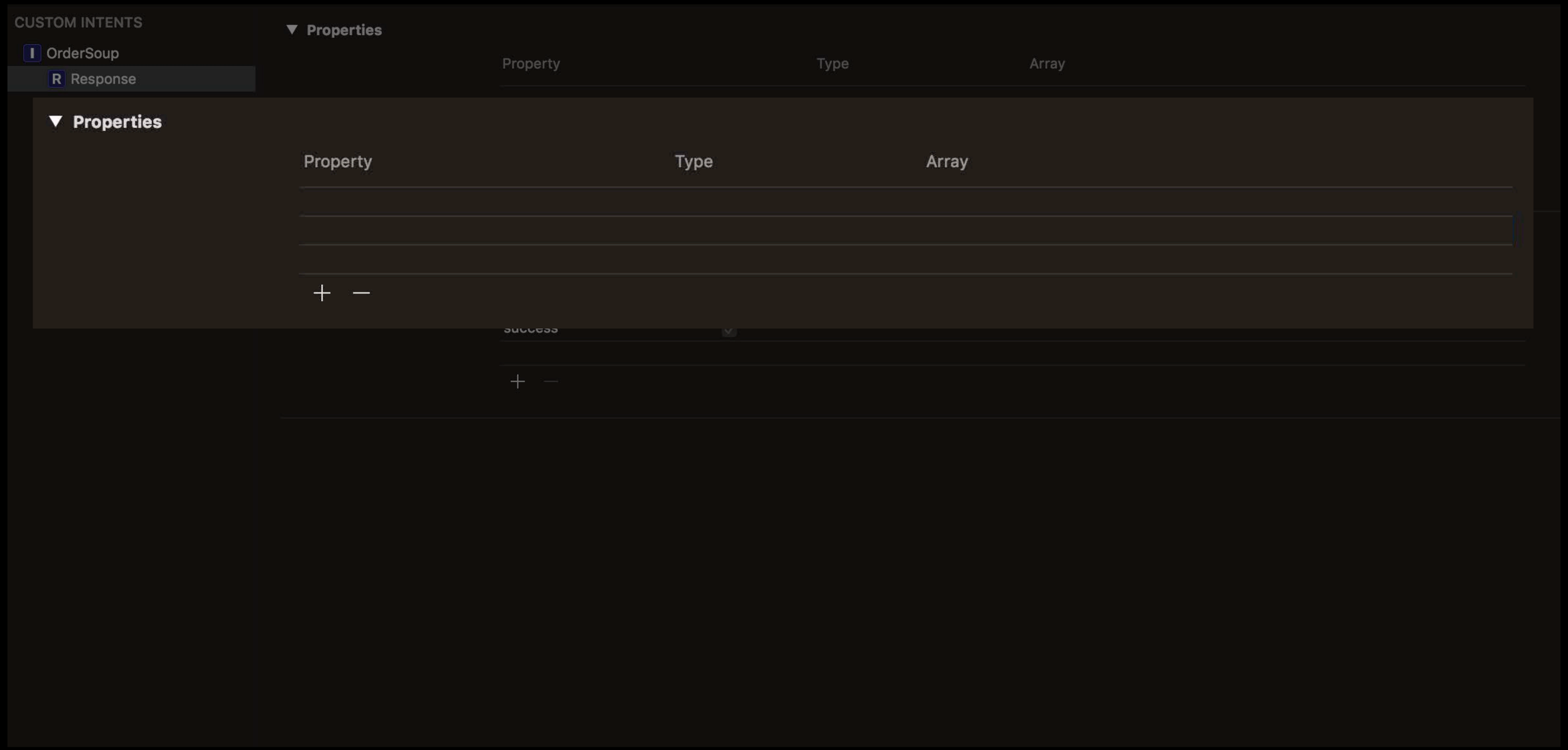
Defining a custom response

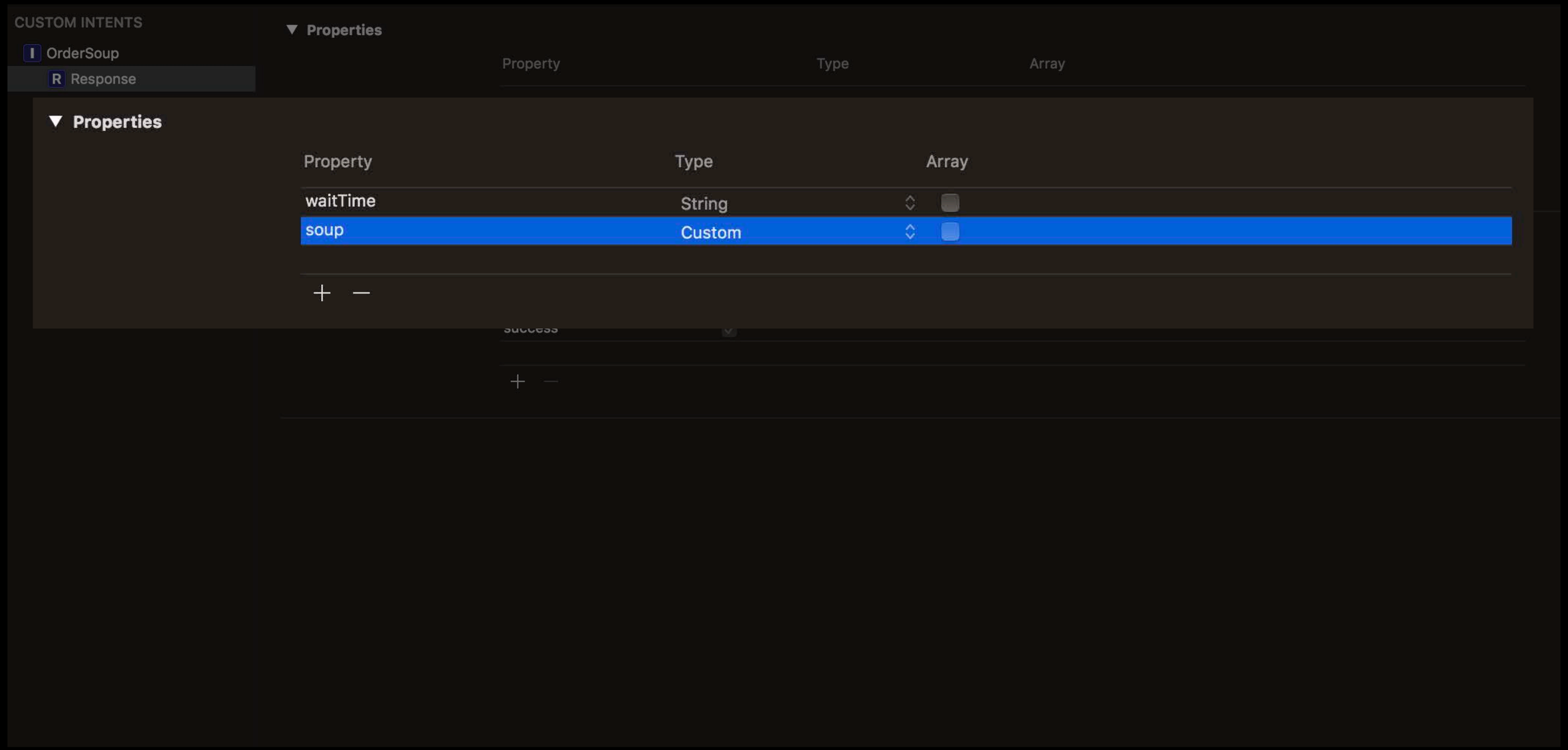
Define a custom intent

Select a category for your intent

Declare response properties


CUSTOM INTENTS
I OrderSoup
R Response
Properties
Property
Type
Array
Response Templates
Code
Success
Template
failure
success


CUSTOM INTENTS
I OrderSoup
R Response
Properties
Property
Type
Array
Properties
Property
Type
Array


CUSTOM INTENTS
OrderSoup
Response
Properties
Property
Type
Array
Properties
Property
Type
Array
waitTime
String
soup
Custom

Adopting Custom Responses

Defining a custom response

Define a custom intent

Select a category for your intent

Declare response properties

Adopting Custom Responses

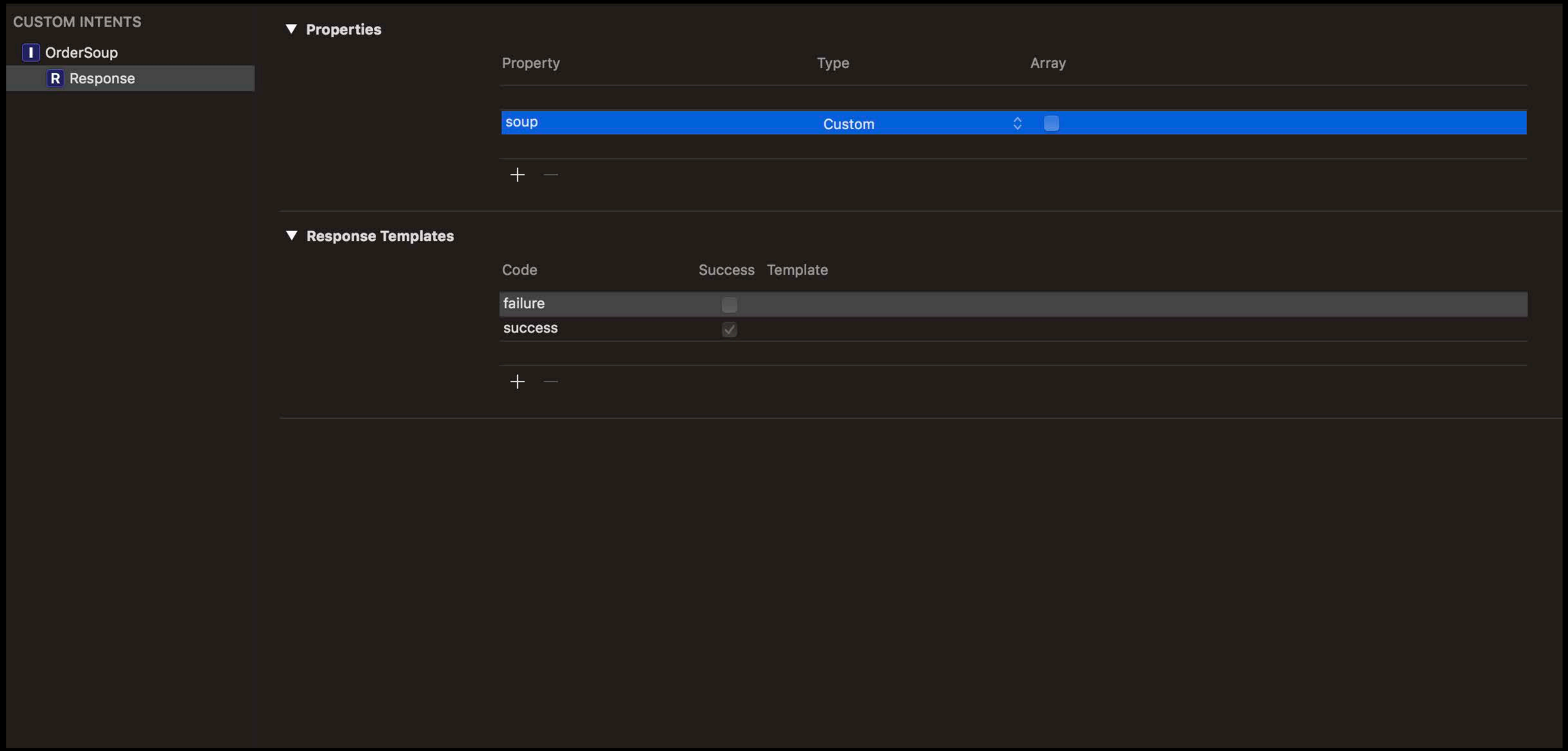
Defining a custom response

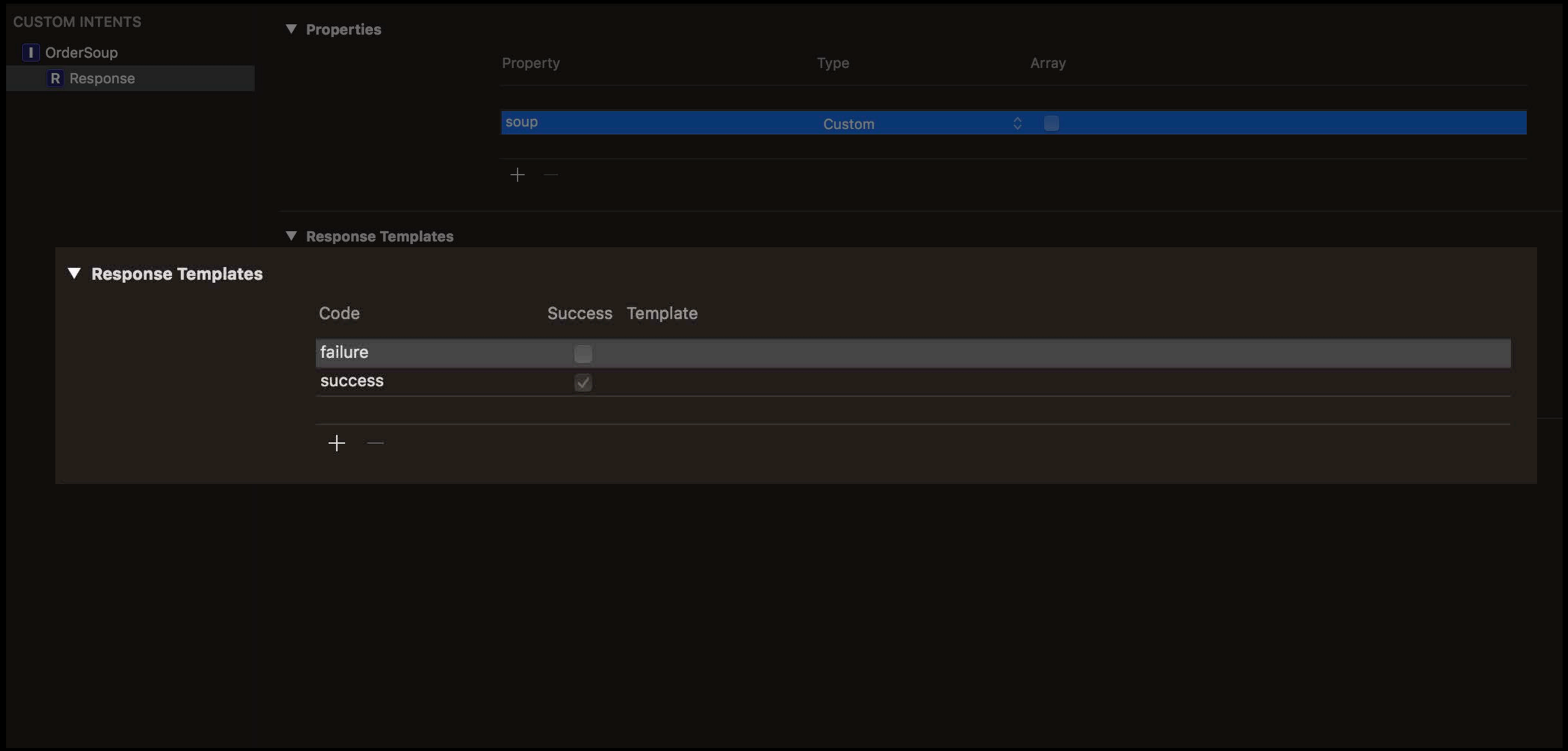
Define a custom intent

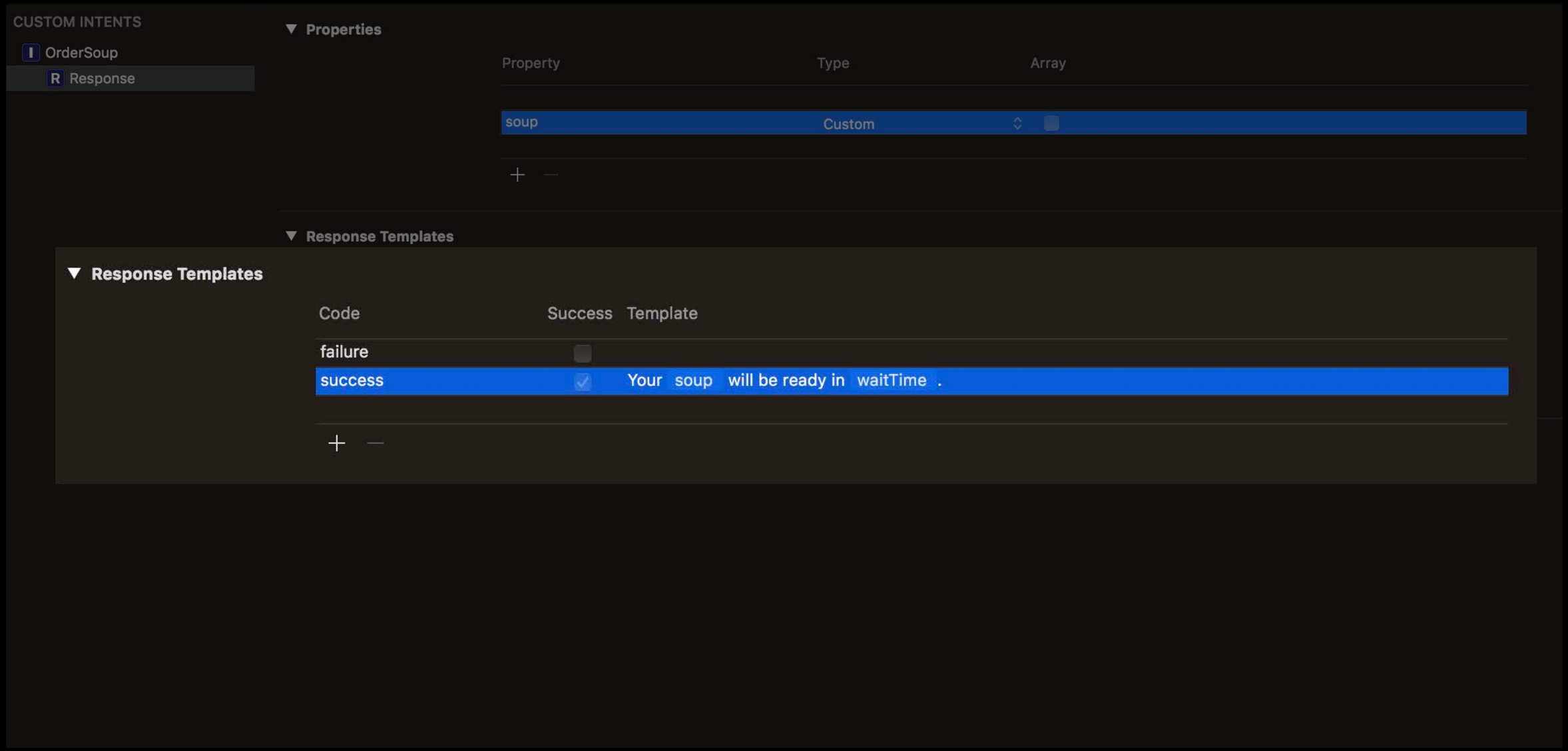
Select a category for your intent

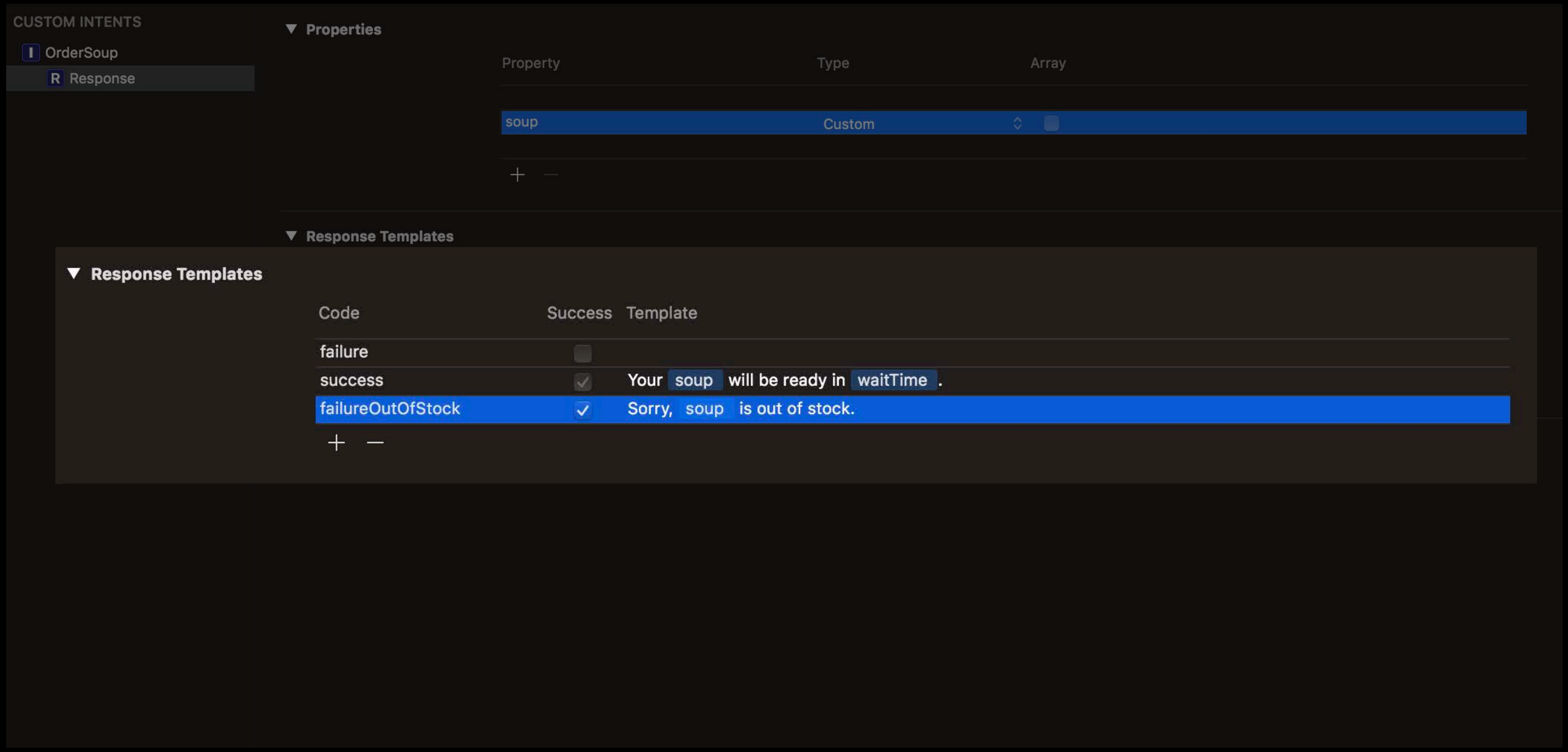
Declare response properties

Provide a response template string


CUSTOM INTENTS
I OrderSoup
R Response
Properties
Property
Type
Array
soup
Custom
Response Templates
Code
Success
Template
failure
success


CUSTOM INTENTS
I OrderSoup
R Response
Properties
Property
Type
Array
soup
Custom
Response Templates
Response Templates
Code
Success
Template
failure
success


CUSTOM INTENTS
I OrderSoup
R Response
Properties
Property
Type
Array
soup
Custom
Response Templates
Response Templates
Code
Success
Template
failure
success
Your soup will be ready in waitTime .


CUSTOM INTENTS
I OrderSoup
R Response
Properties
Property
Type
Array
soup
Custom
Response Templates
Response Templates
Code
Success
Template
failure
success
Your soup will be ready in waitTime .
failureOutOfStock
Sorry, soup is out of stock.

```
// Custom response object
public class OrderSoupIntentResponse : INIntentResponse {
    public var waitTime: String? { get set }
    public var soup: INObject? { get set }

    public convenience init(code: OrderSoupIntentResponseCode, userActivity: NSUserActivity?)

    public static func success(soup: INObject, waitTime: String) -> OrderSoupIntentResponse

    public static func failureOutOfStock(soup: INObject) -> OrderSoupIntentResponse
}
```

```
// Custom response object
public class OrderSoupIntentResponse : INIntentResponse {
    public var waitTime: String? { get set }
    public var soup: INObject? { get set }

    public convenience init(code: OrderSoupIntentResponseCode, userActivity: NSUserActivity?)

    public static func success(soup: INObject, waitTime: String) -> OrderSoupIntentResponse

    public static func failureOutOfStock(soup: INObject) -> OrderSoupIntentResponse
}
```

```
// Custom response object
public class OrderSoupIntentResponse : INIntentResponse {
    public var waitTime: String? { get set }
    public var soup: INObject? { get set }

    public convenience init(code: OrderSoupIntentResponseCode, userActivity: NSUserActivity?)

    public static func success(soup: INObject, waitTime: String) -> OrderSoupIntentResponse

    public static func failureOutOfStock(soup: INObject) -> OrderSoupIntentResponse
}
```

Adopting Custom Responses

Defining a custom response

Define a custom intent

Select a category for your intent

Declare response properties

Provide a response template string

Adopting Custom Responses

Defining a custom response

Define a custom intent

Select a category for your intent

Declare response properties

Provide a response template string

Provide custom responses in your intent handler

```swift
// Confirm that the intent can be run
class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }

}
```

```swift
// Confirm that the intent can be run
class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }

}
```

```swift
// Confirm that the intent can be run
class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }

}
```

```swift
// Confirm that the intent can be run
class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        if menuItem.isAvailable == false {
            completion(OrderSoupIntentResponse.failureOutOfStock(soup: soup))
            return
        }

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }

}
```

```swift
// Confirm that the intent can be run
class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        if menuItem.isAvailable == false {
            completion(OrderSoupIntentResponse.failureOutOfStock(soup: soup))
            return
        }

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }

}
```

```swift
// Confirm that the intent can be run
class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        if menuItem.isAvailable == false {
            completion(OrderSoupIntentResponse.failureOutOfStock(soup: soup))
            return
        }

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }

}
```

```swift
// Confirm that the intent can be run
class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        if menuItem.isAvailable == false {
            completion(OrderSoupIntentResponse.failureOutOfStock(soup: soup))
            return
        }

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }

}
```

```swift
// Confirm that the intent can be run
class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        if menuItem.isAvailable == false {
            completion(OrderSoupIntentResponse.failureOutOfStock(soup: soup))
            return
        }

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }

}
```

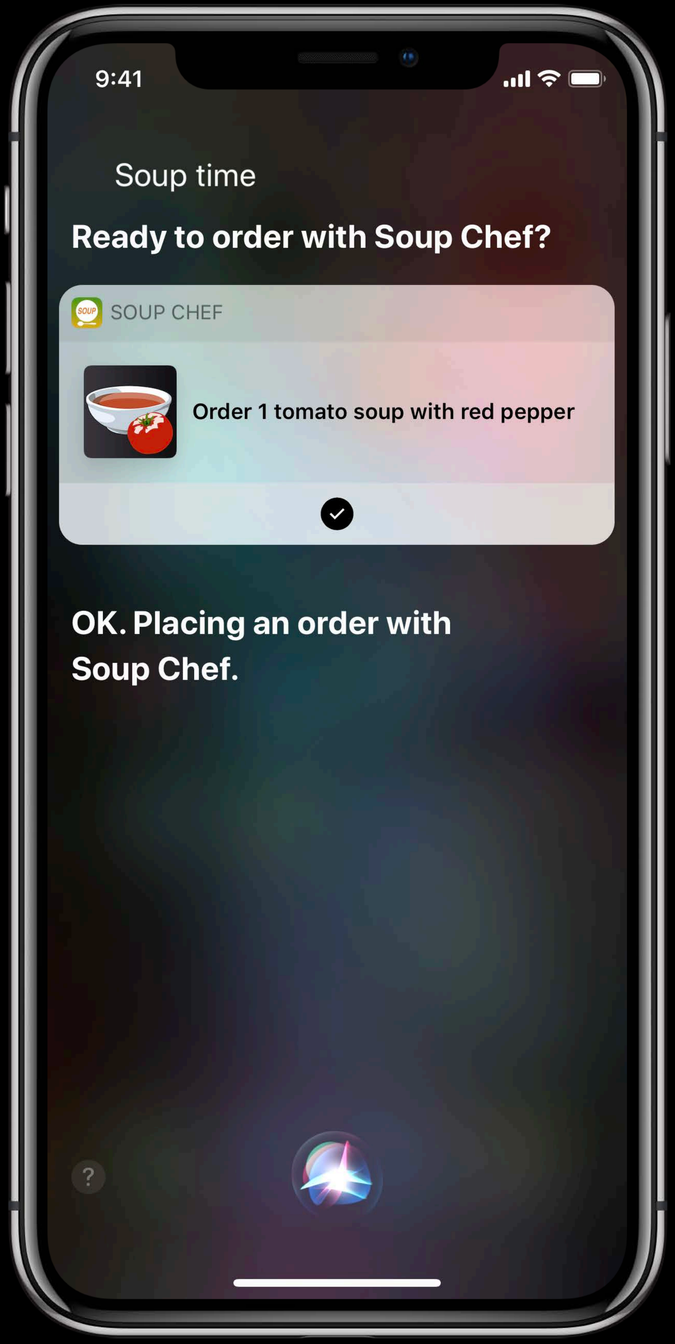
```swift
// Handle the intent
class IntentHandler: INExtension, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        ...
    }

    func handle(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        guard
            let soup = intent.soup,
            let order = Order(from: intent)
        else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let orderManager = SoupOrderDataManager()
        orderManager.placeOrder(order: order)

        completion(OrderSoupIntentResponse(code: .success, userActivity: nil))
    }
}
```

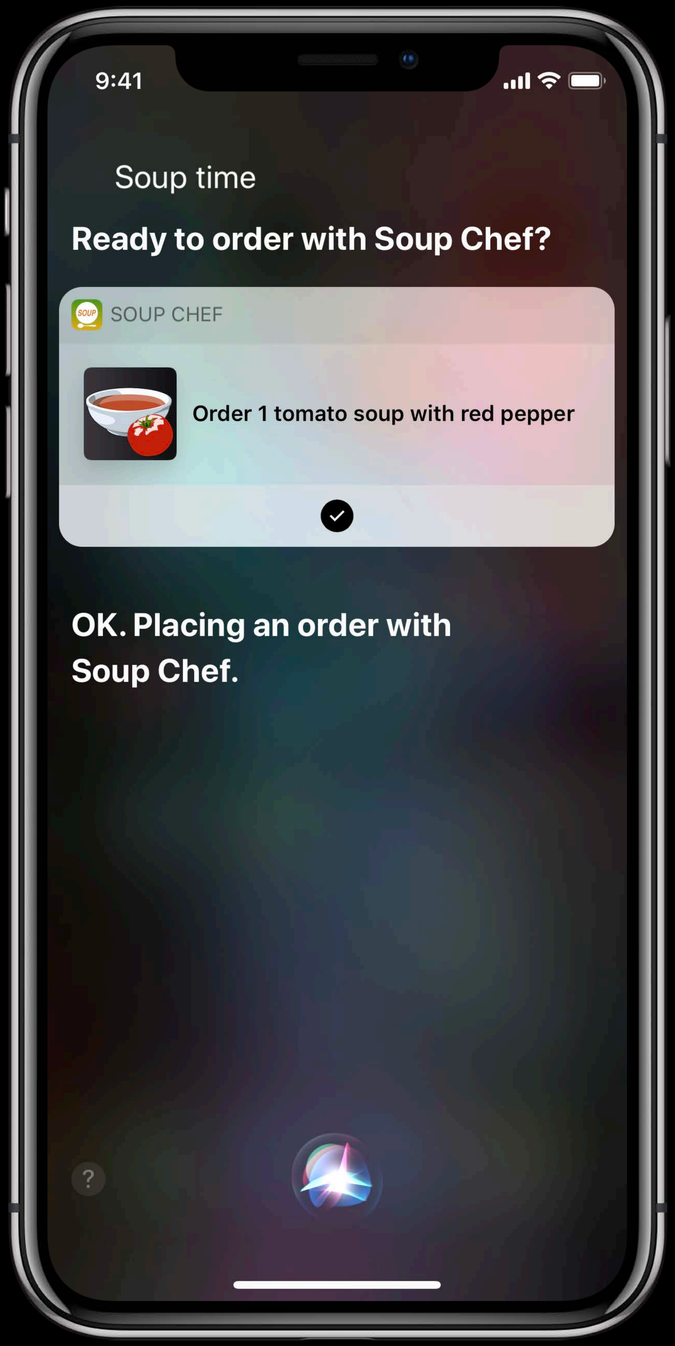
```swift
// Handle the intent
class IntentHandler: INExtension, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        ...
    }

    func handle(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        guard
            let soup = intent.soup,
            let order = Order(from: intent)
        else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let orderManager = SoupOrderDataManager()
        orderManager.placeOrder(order: order)

        completion(OrderSoupIntentResponse(code: .success, userActivity: nil))
    }
}
```

```swift
// Handle the intent
class IntentHandler: INExtension, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        ...
    }

    func handle(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        guard
            let soup = intent.soup,
            let order = Order(from: intent)
        else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let orderManager = SoupOrderDataManager()
        orderManager.placeOrder(order: order)

        completion(OrderSoupIntentResponse(code: .success, userActivity: nil))
    }
}
```

```swift
// Handle the intent
class IntentHandler: INExtension, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        ...
    }

    func handle(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        guard
            let soup = intent.soup,
            let order = Order(from: intent)
        else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let orderManager = SoupOrderDataManager()
        orderManager.placeOrder(order: order)

        completion(OrderSoupIntentResponse(code: .success, userActivity: nil))
    }
}
```

```swift
// Handle the intent
class IntentHandler: INExtension, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        ...
    }

    func handle(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        guard
            let soup = intent.soup,
            let order = Order(from: intent)
        else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let orderManager = SoupOrderDataManager()
        orderManager.placeOrder(order: order)

        completion(OrderSoupIntentResponse.success(soup: soup, waitTime: order.waitTime))
    }
}
```

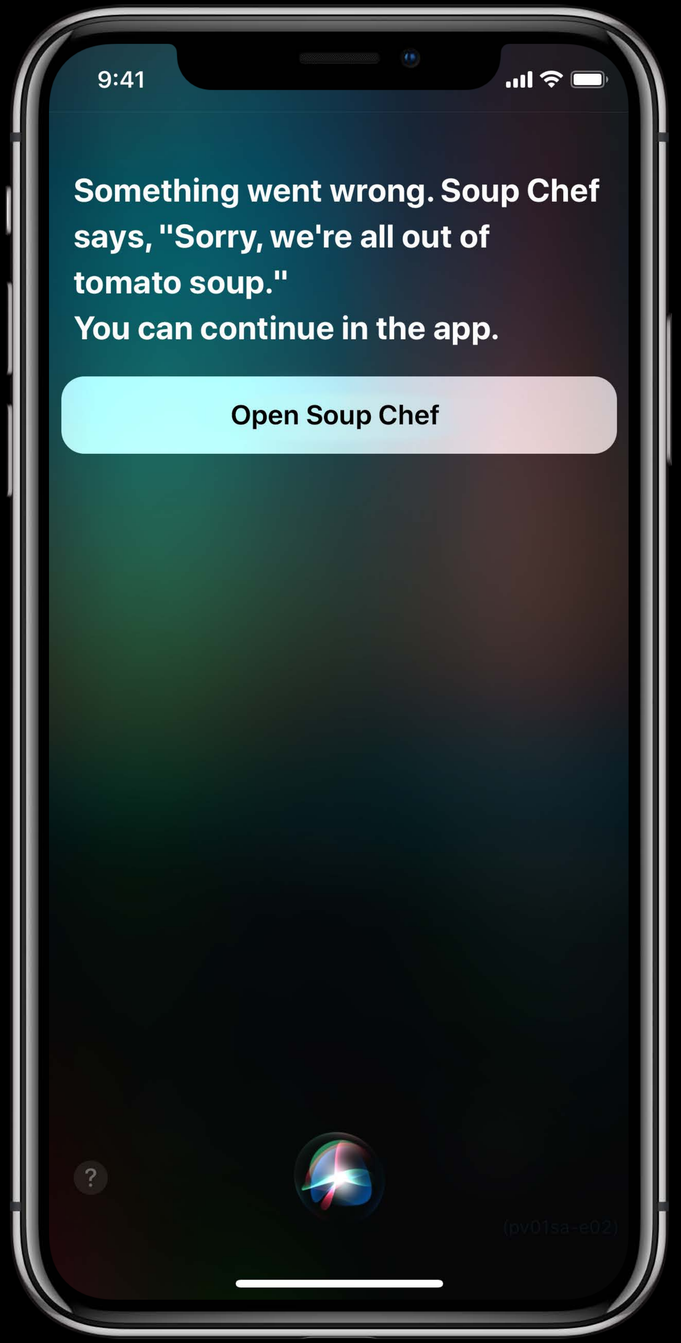

9:41
Soup time
Ready to order with Soup Chef?
SOUP CHEF
Order 1 tomato soup with red pepper
OK. Placing an order with Soup Chef.


9:41
Soup time
Ready to order with Soup Chef?
SOUP CHEF
Order 1 tomato soup with red pepper
OK. Placing an order with Soup Chef.


9:41
Soup time
Ready to order with Soup Chef?
SOUP CHEF
Order 1 tomato soup with red pepper
OK. Ordering. Soup Chef says: "Your tomato soup will be ready in 10 minutes."


9:41
Sorry, there was a problem with the app.
?


9:41
Sorry, there was a problem with the app.


9:41
Something went wrong. Soup Chef says, "Sorry, we're all out of tomato soup."
You can continue in the app.
Open Soup Chef

Adopting Custom Responses

Custom UI Extension

Provide a custom UI for your shortcut

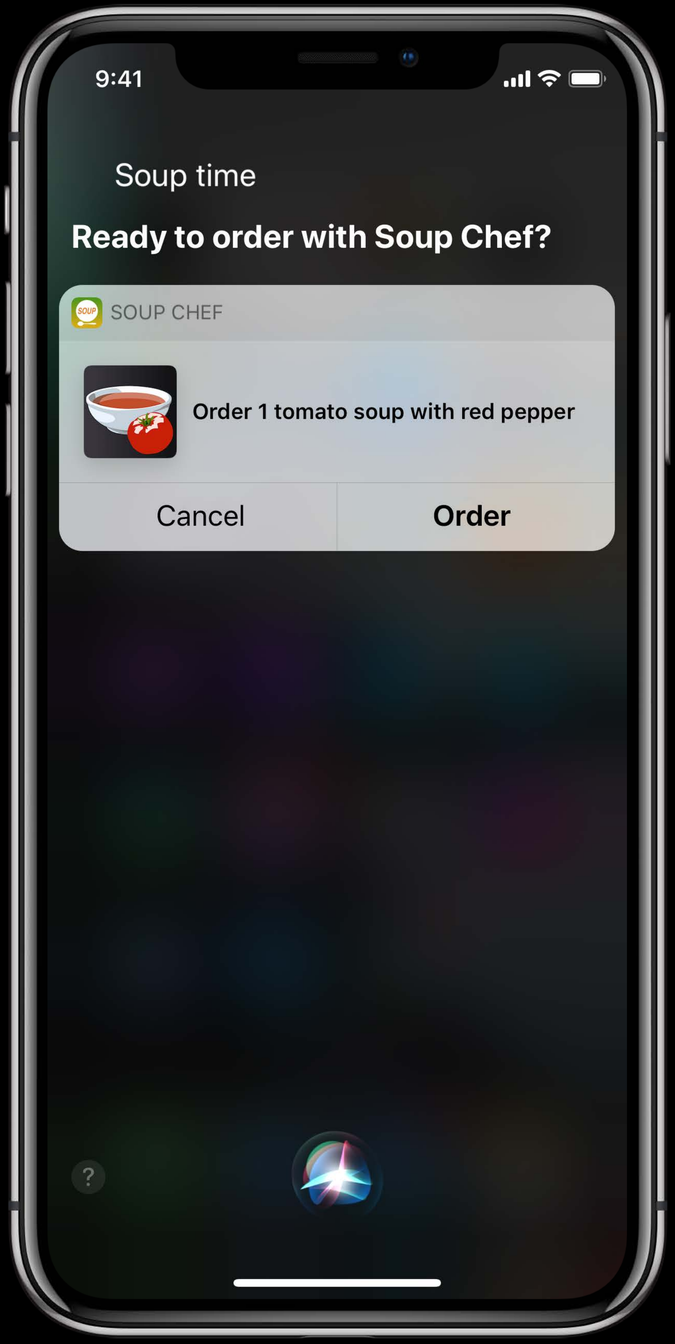
Shows in Siri, on Lock Screen, and in Search

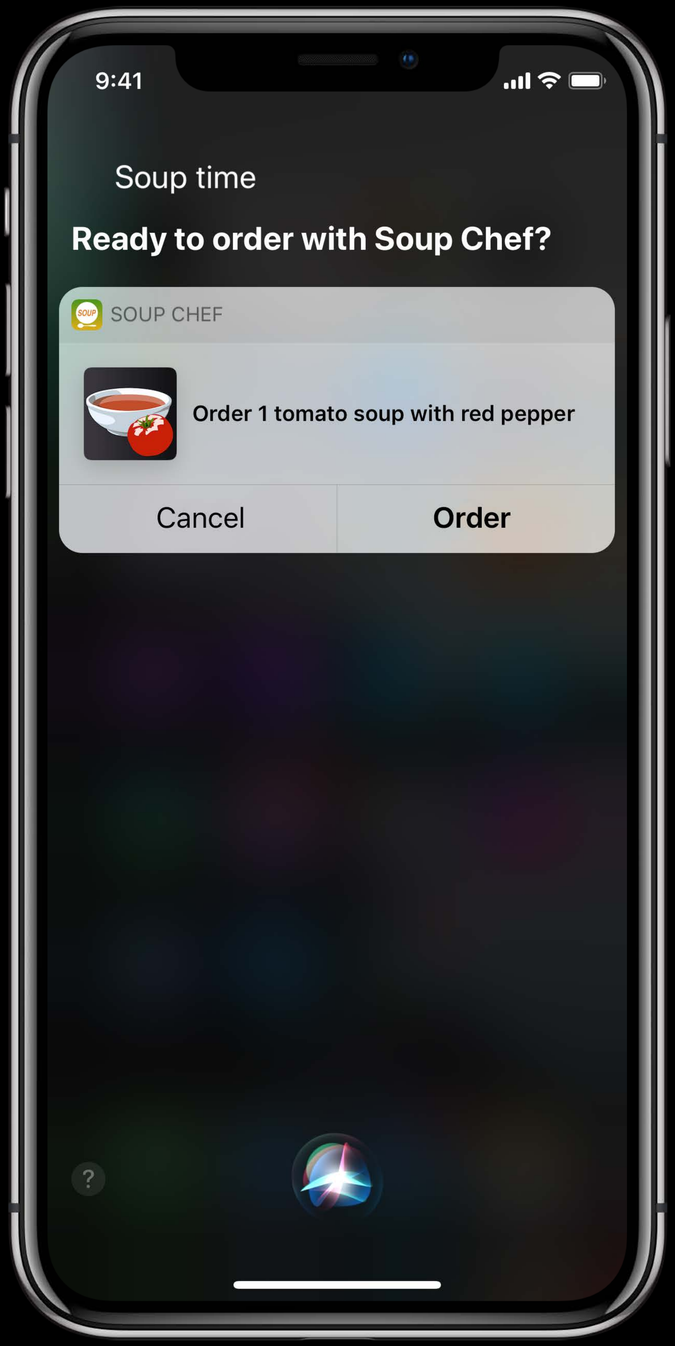
Adopting Custom Responses

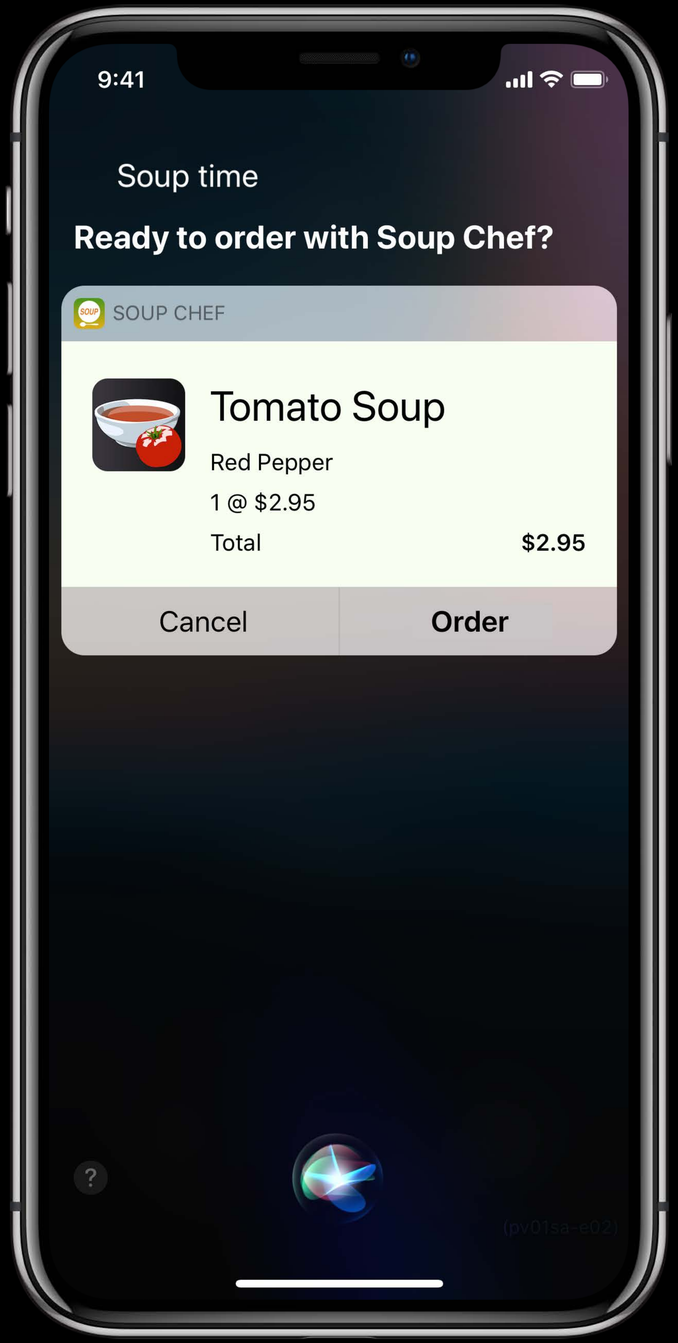
Custom UI Extension

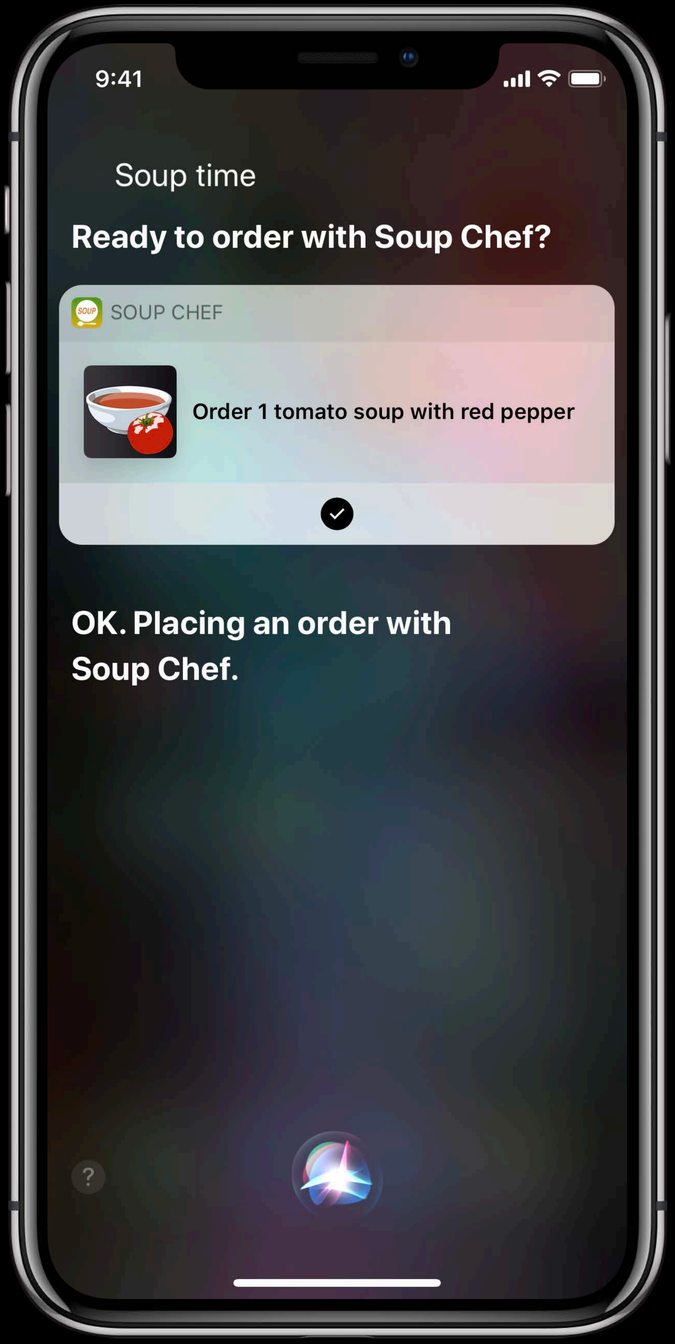
Provide a custom UI for your shortcut

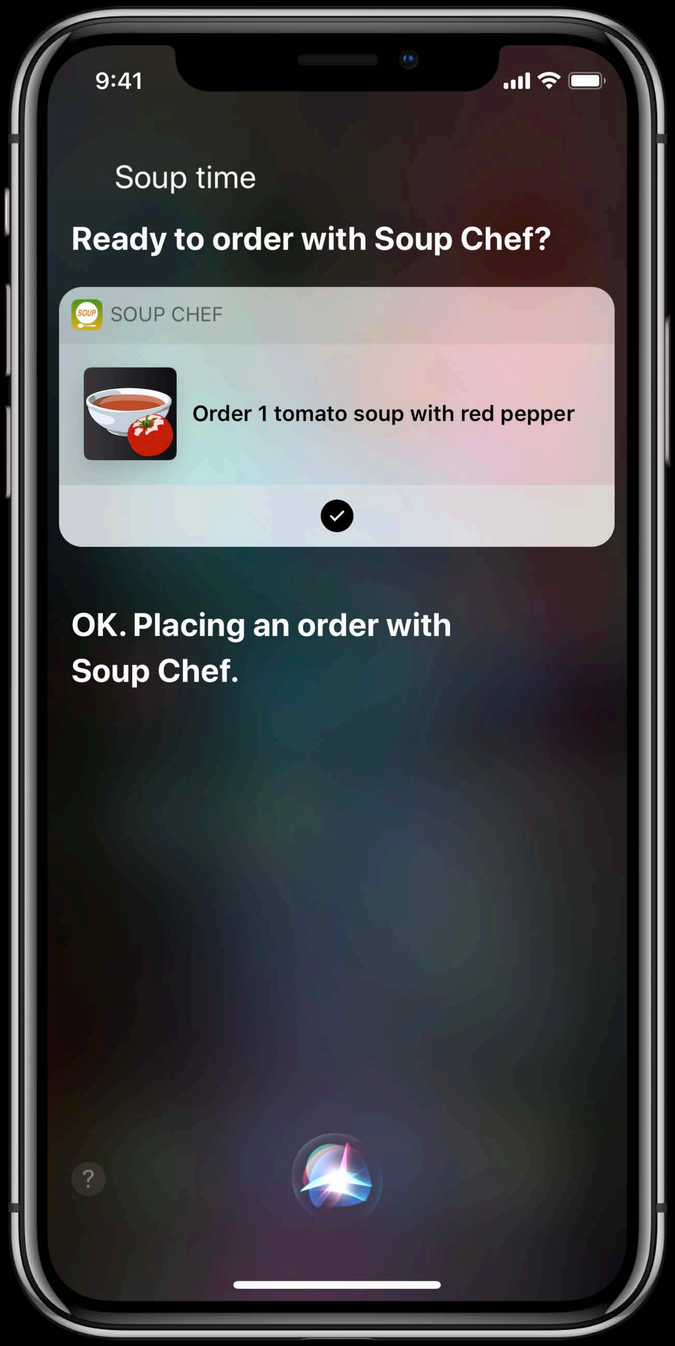
Shows in Siri, on Lock Screen, and in Search


9:41
Soup time
Ready to order with Soup Chef?
SOUP CHEF
Order 1 tomato soup with red pepper
Cancel
Order


9:41
Soup time
Ready to order with Soup Chef?
SOUP CHEF
Order 1 tomato soup with red pepper
Cancel
Order


9:41
Soup time
Ready to order with Soup Chef?
SOUP CHEF
Tomato Soup
Red Pepper
1 @ $2.95
Total
$2.95
Cancel
Order


9:41
Soup time
Ready to order with Soup Chef?
SOUP CHEF
Order 1 tomato soup with red pepper
OK. Placing an order with Soup Chef.


9:41
Soup time
Ready to order with Soup Chef?
SOUP CHEF
Order 1 tomato soup with red pepper
OK. Placing an order with Soup Chef.


9:41
Soup time
Ready to order with Soup Chef?
SOUP CHEF
Tomato Soup
Order Confirmed
Wait time 10 Minutes
Total $2.95
OK. Placing an order. Soup Chef says: "Your tomato soup will be ready in 10 minutes."

Demo

Adopting Custom Responses

Define a custom intent

Select a category for your intent

Declare response properties

Provide a response template string

Provide the response in your intent handler

Siri Shortcuts Best Practices

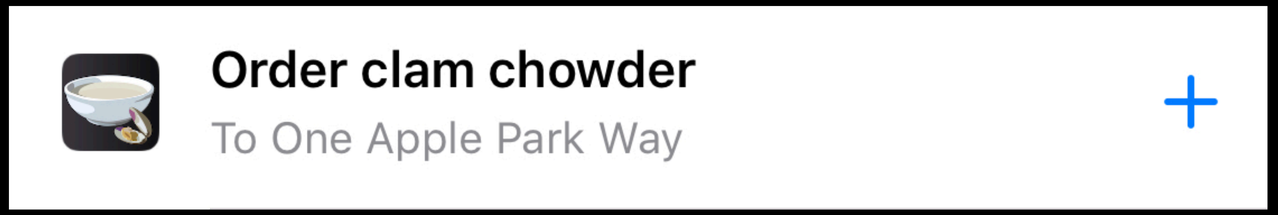
Ayaka Nonaka, Siri

NSUserActivity

NSUserActivity

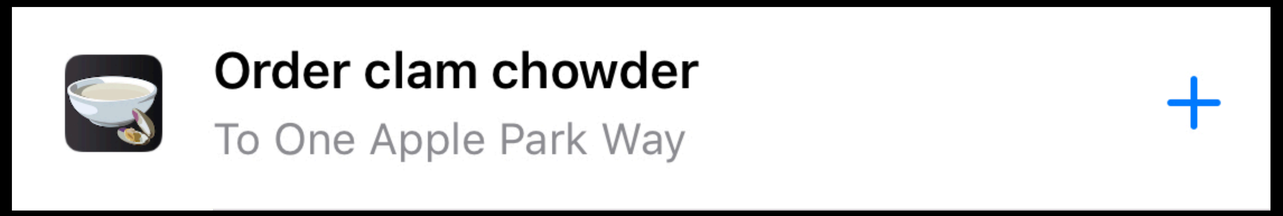
Intents

Customizing Donations



Customizing Donations

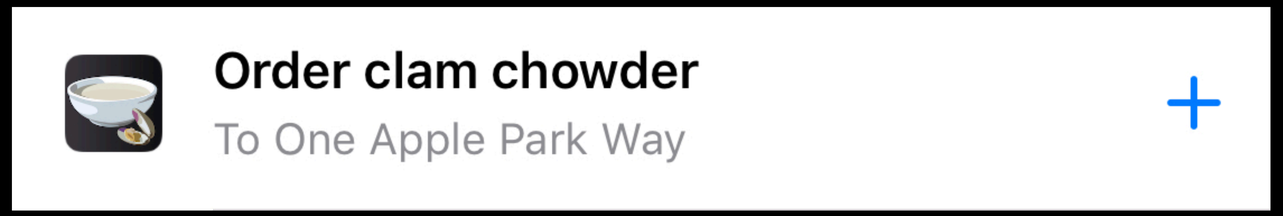
Title



Customizing Donations

Title

Subtitle

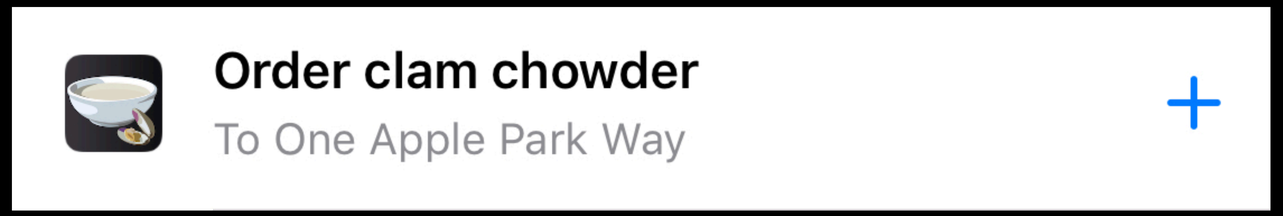


Customizing Donations

Title

Subtitle

Image



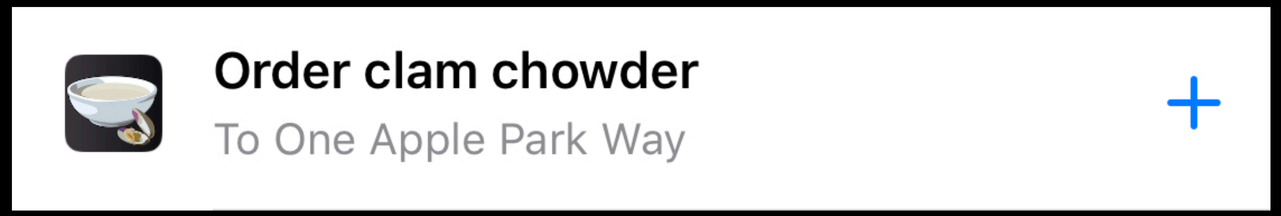
Customizing Donations

Title

Subtitle

Image

Suggested invocation phrase



Choosing a Good Title and Subtitle

Choosing a Good Title and Subtitle

The title should represent what happens when you run the shortcut

Choosing a Good Title and Subtitle

The title should represent what happens when you run the shortcut

The subtitle should provide supplementary information if needed

Basic Guidelines for Title and Subtitle

Basic Guidelines for Title and Subtitle

Use sentence case

Basic Guidelines for Title and Subtitle

Use sentence case

Keep the title concise

Basic Guidelines for Title and Subtitle

Use sentence case

Keep the title concise

Include critical information

Basic Guidelines for Title and Subtitle

Use sentence case

Keep the title concise

Include critical information

Include a verb for Intents

Basic Guidelines for Title and Subtitle

Use sentence case

Keep the title concise

Include critical information

Include a verb for Intents

Localize

Things to Avoid in Title and Subtitle

Things to Avoid in Title and Subtitle

The name of your app

Things to Avoid in Title and Subtitle

The name of your app

Duplicate information in the title and subtitle

Things to Avoid in Title and Subtitle

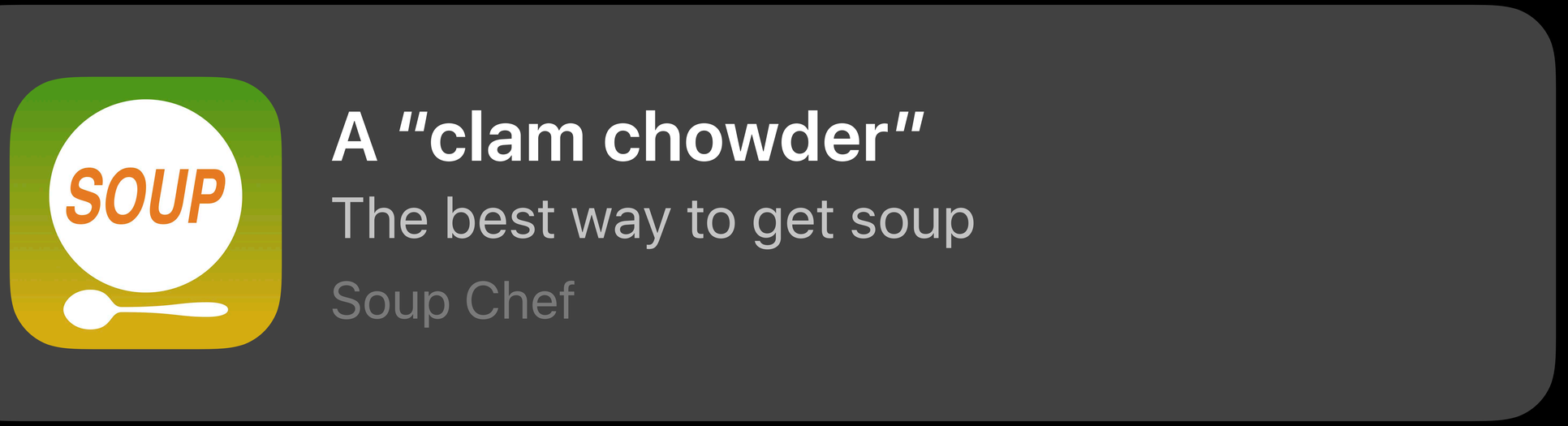
The name of your app

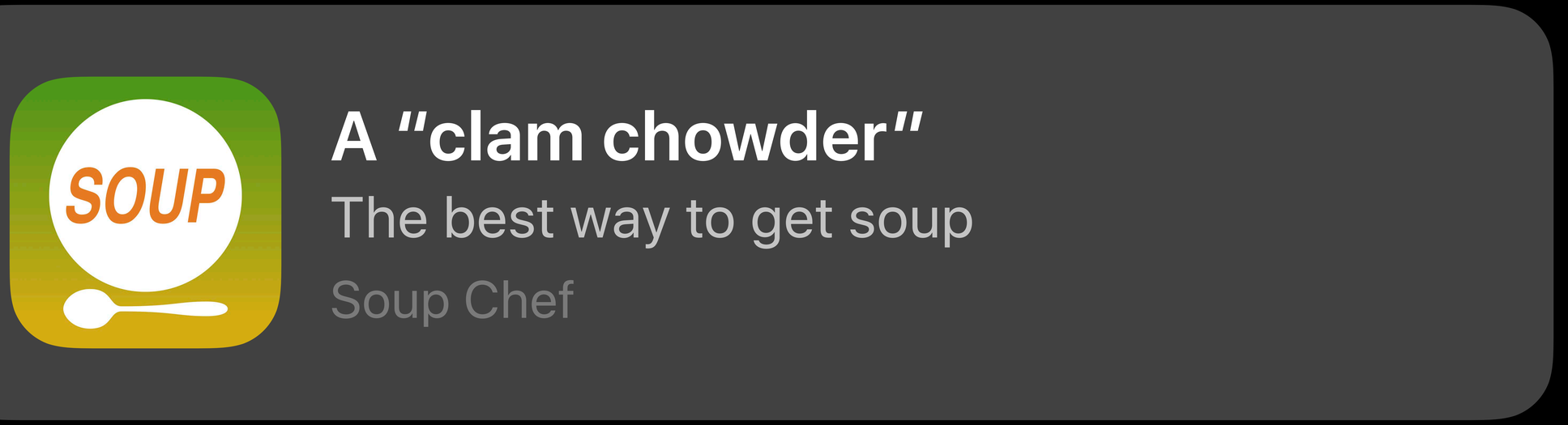
Duplicate information in the title and subtitle

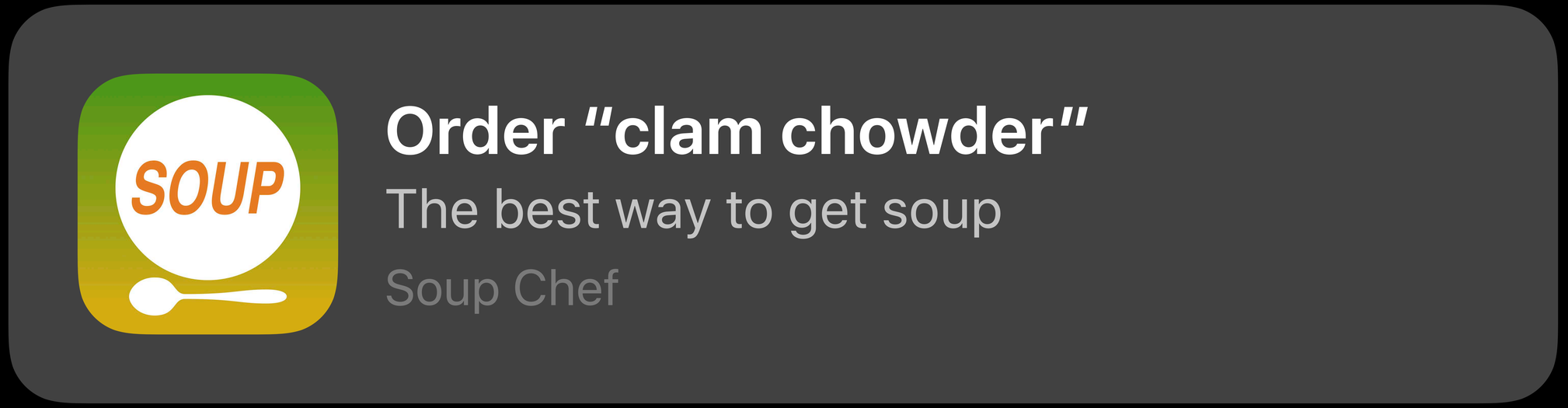
“Quotation marks” except when referring to strings that will be used verbatim in the shortcut

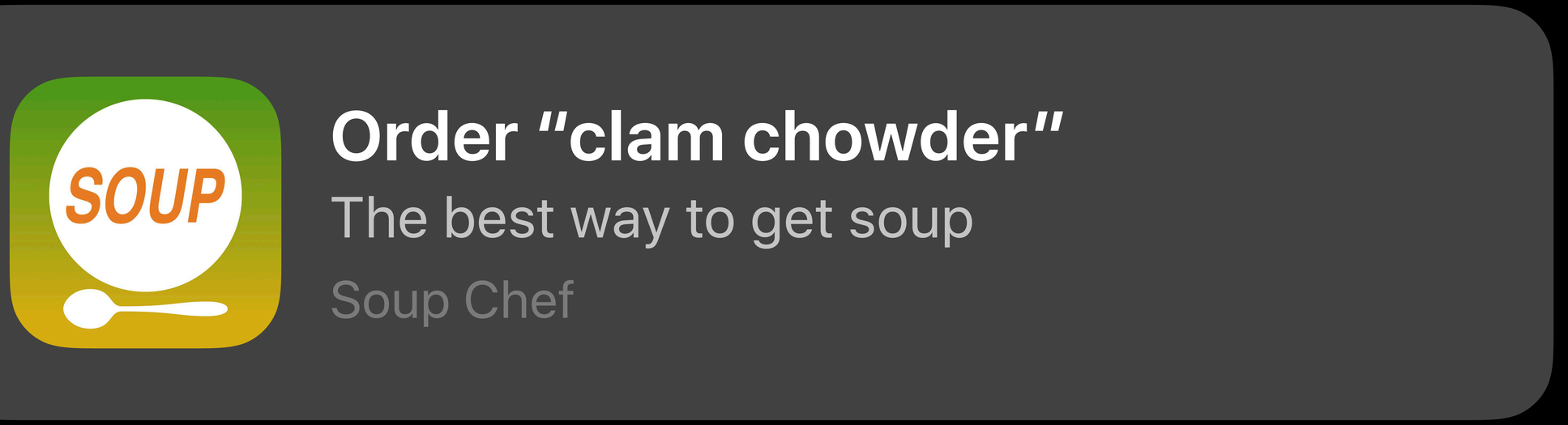

SOUP
A “clam chowder” from Soup Chef
The best way to get soup
Soup Chef


SOUP
A “clam chowder” from Soup Chef
The best way to get soup
Soup Chef


SOUP
A “clam chowder”
The best way to get soup
Soup Chef

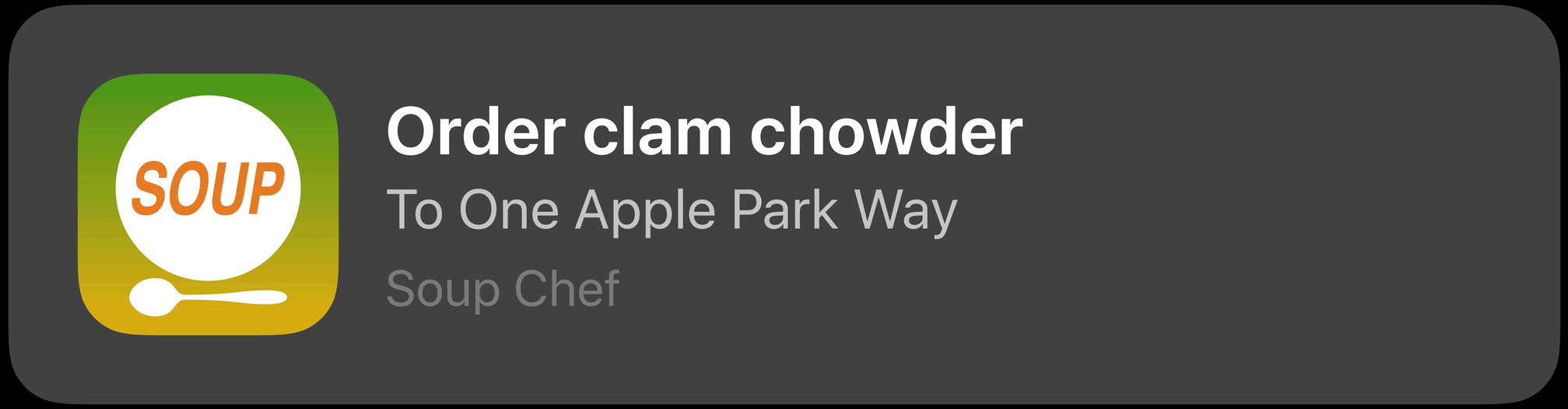

SOUP
A “clam chowder”
The best way to get soup
Soup Chef

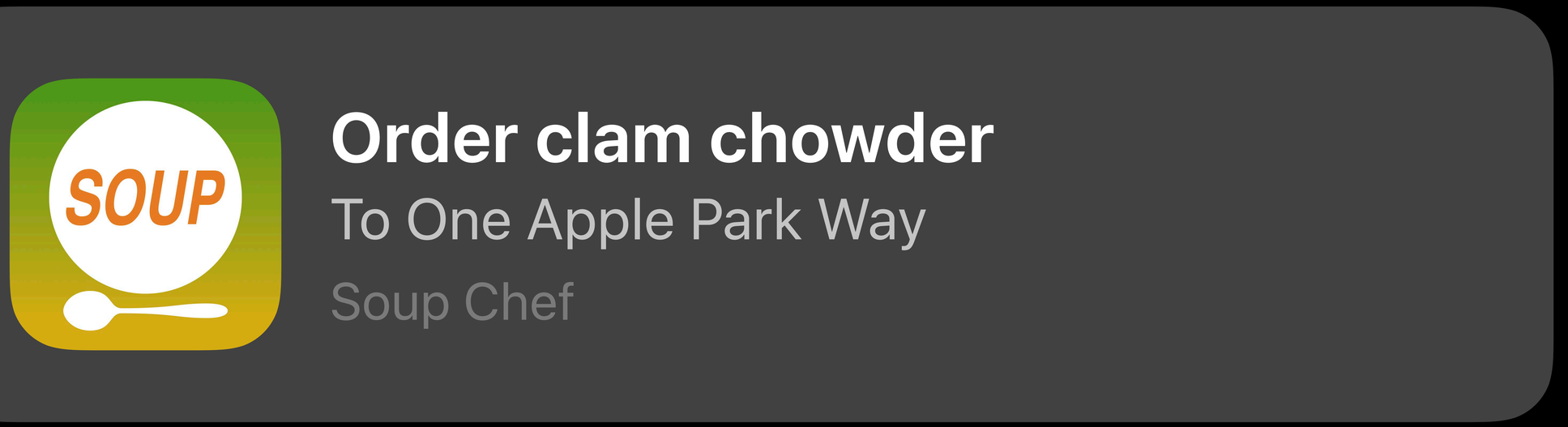

SOUP
Order “clam chowder”
The best way to get soup
Soup Chef

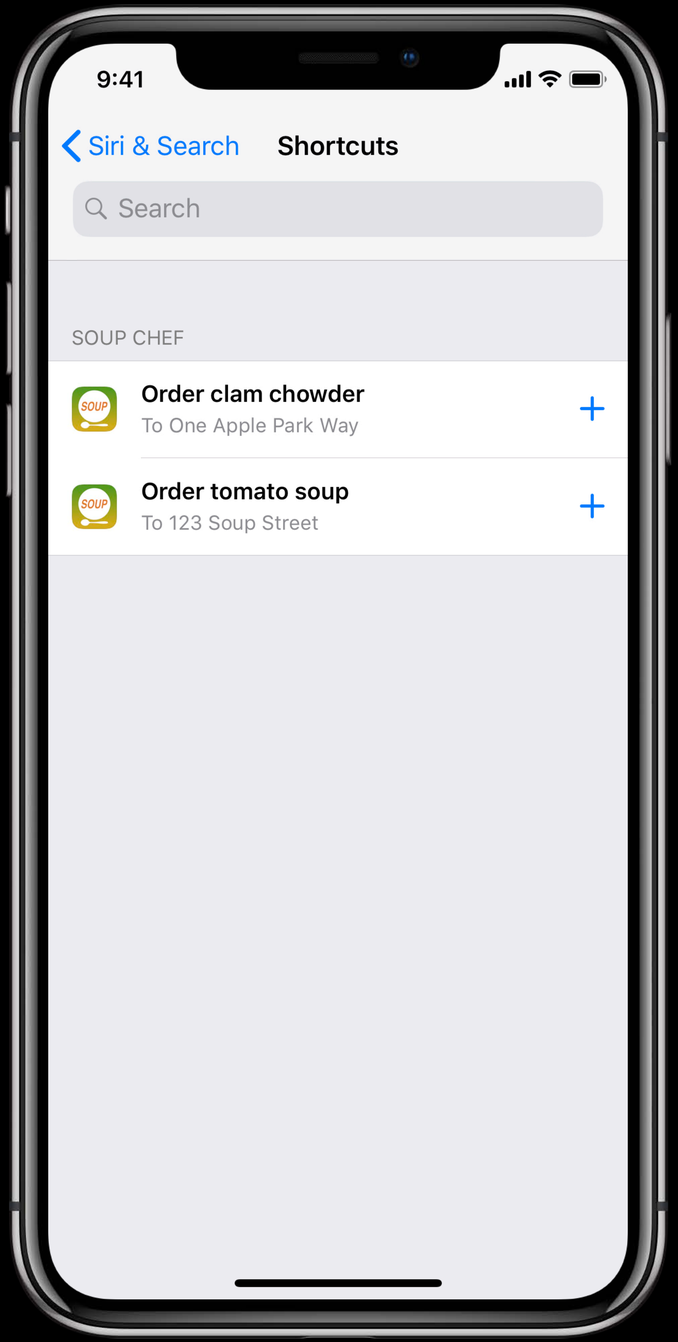

SOUP
Order “clam chowder”
The best way to get soup
Soup Chef

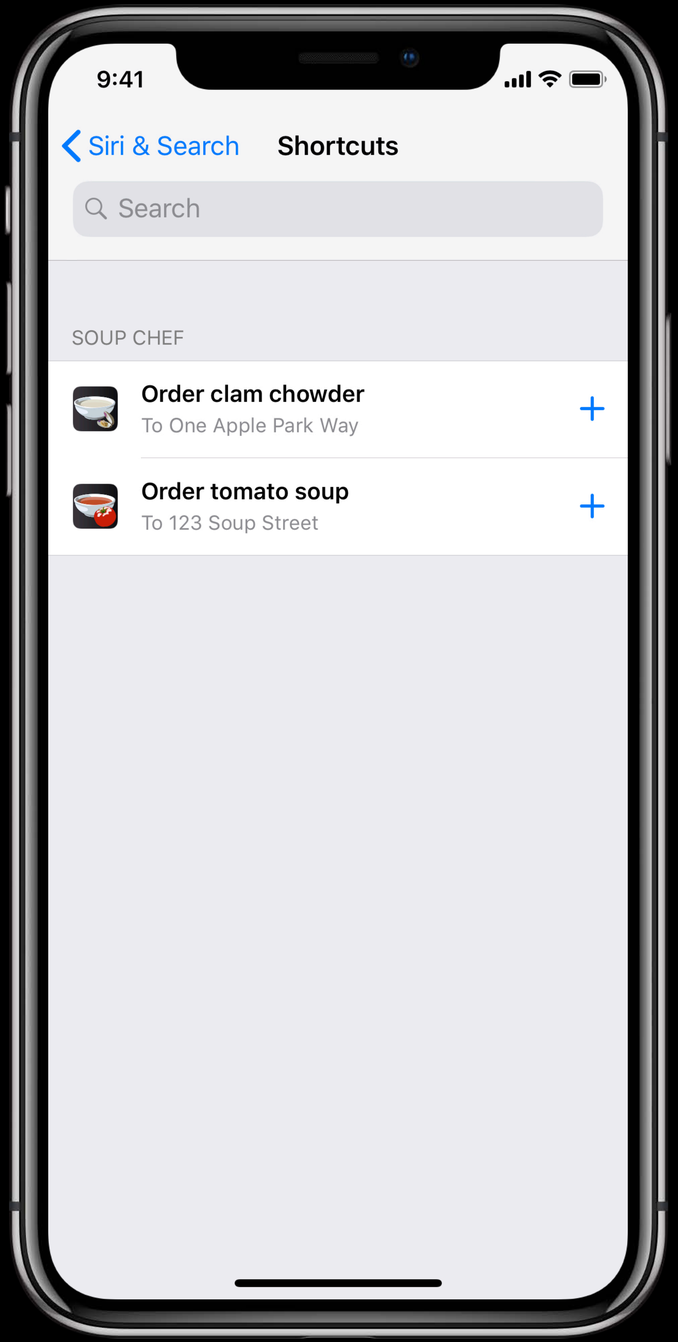

SOUP
Order clam chowder
The best way to get soup
Soup Chef


SOUP
Order clam chowder
The best way to get soup
Soup Chef


SOUP
Order clam chowder
To One Apple Park Way
Soup Chef


SOUP
Order clam chowder
To One Apple Park Way
Soup Chef


9:41
Siri & Search
Shortcuts
Search
SOUP CHEF
Order clam chowder
To One Apple Park Way
Order tomato soup
To 123 Soup Street


9:41
Siri & Search
Shortcuts
Search
SOUP CHEF
Order clam chowder
To One Apple Park Way
Order tomato soup
To 123 Soup Street

```
// Setting an image on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons” // Subtitle

let image = UIImage(named: “Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
```

```
// Setting an image on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons” // Subtitle

let image = UIImage(named: “Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
```

```
// Setting an image on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons” // Subtitle

let image = UIImage(named: “Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
```

```
// Setting an image on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons” // Subtitle

let image = UIImage(named: “Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
```

```
// Setting an image on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons” // Subtitle

let image = UIImage(named: “Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
```

```
// Setting an image on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons” // Subtitle

let image = UIImage(named: “Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
```

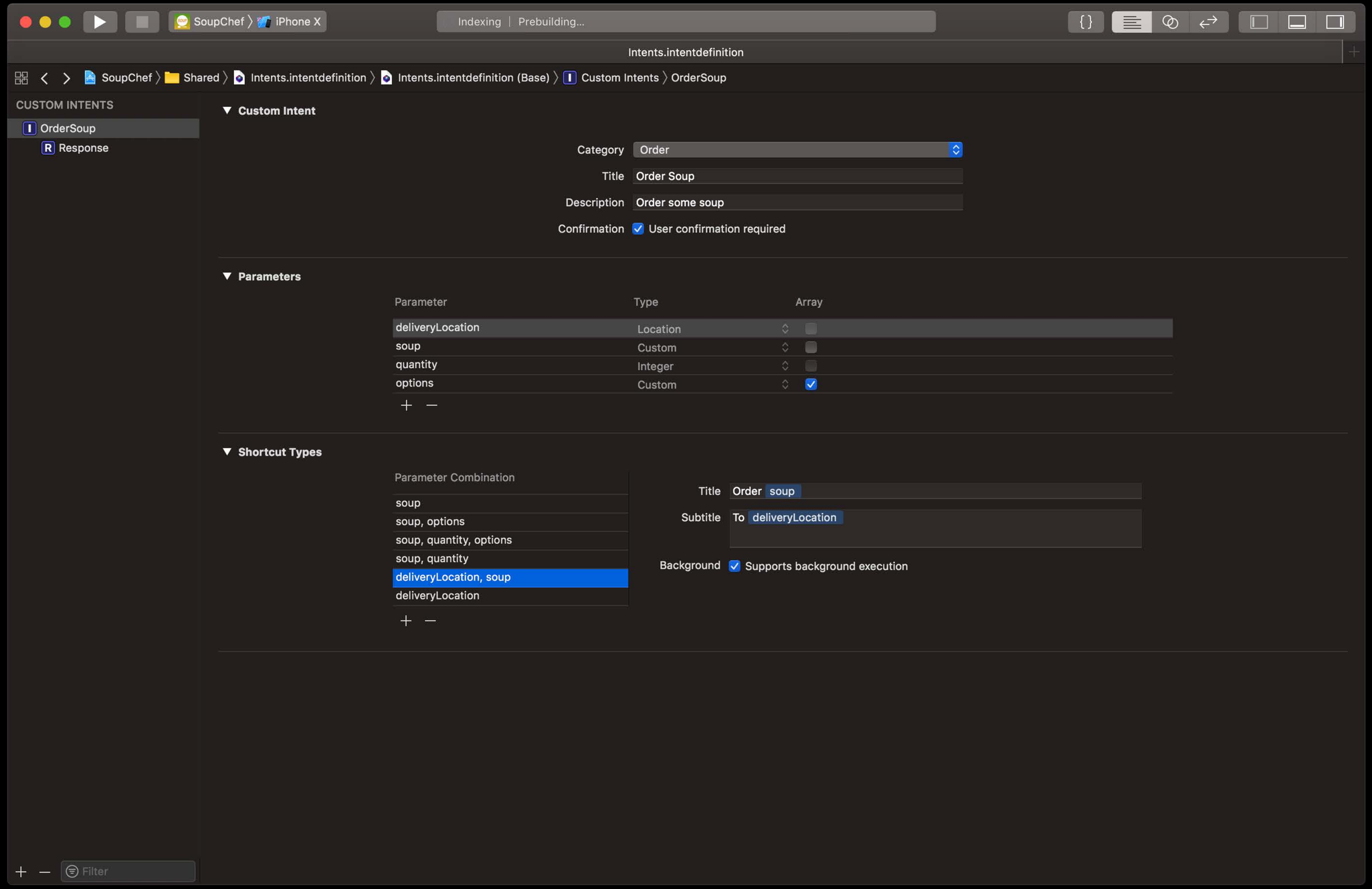
```
// Setting an image on an INIntent

import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
     // Handle error
}
```

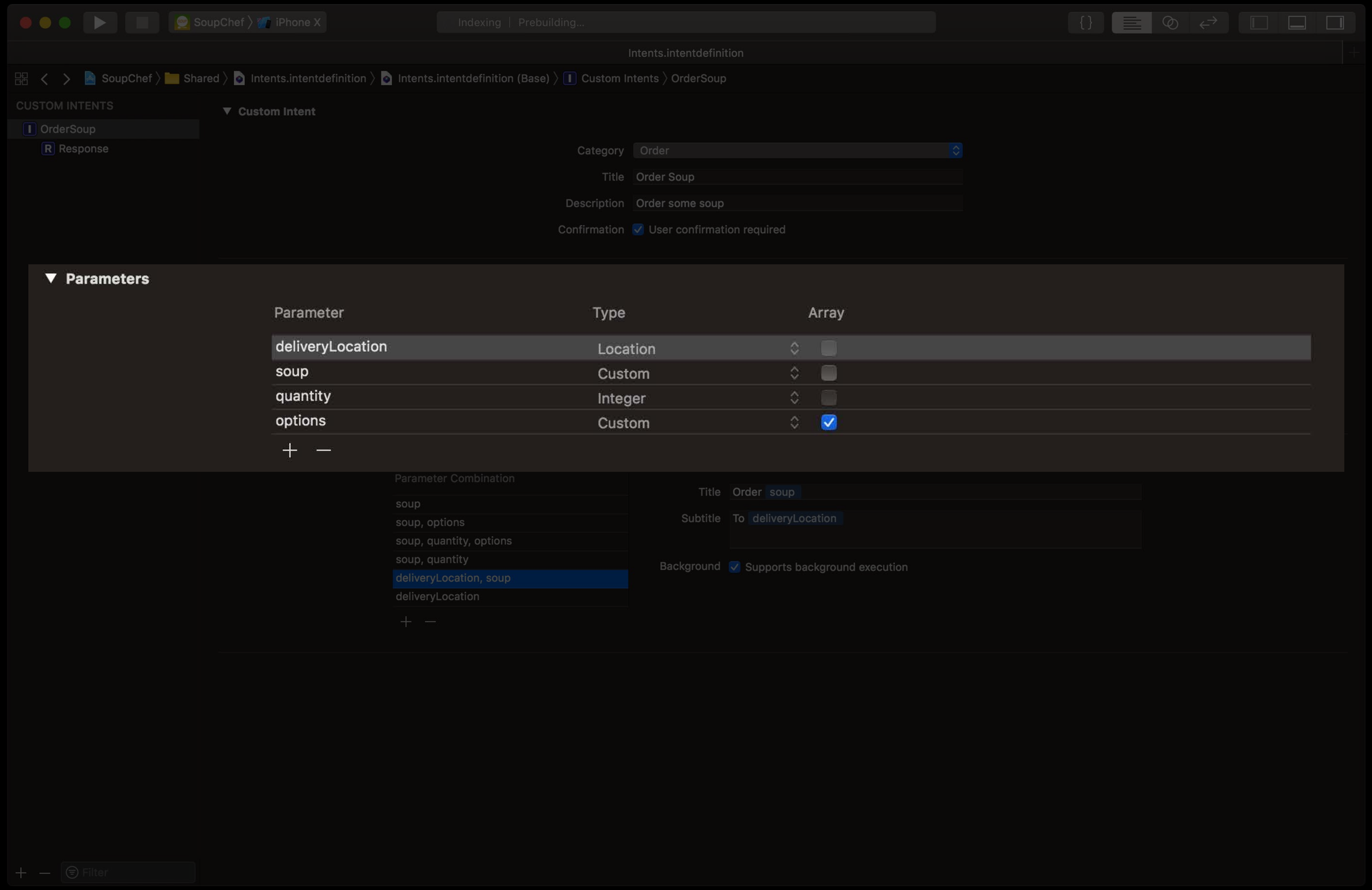
```
// Setting an image on an INIntent

import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
     // Handle error
}
```

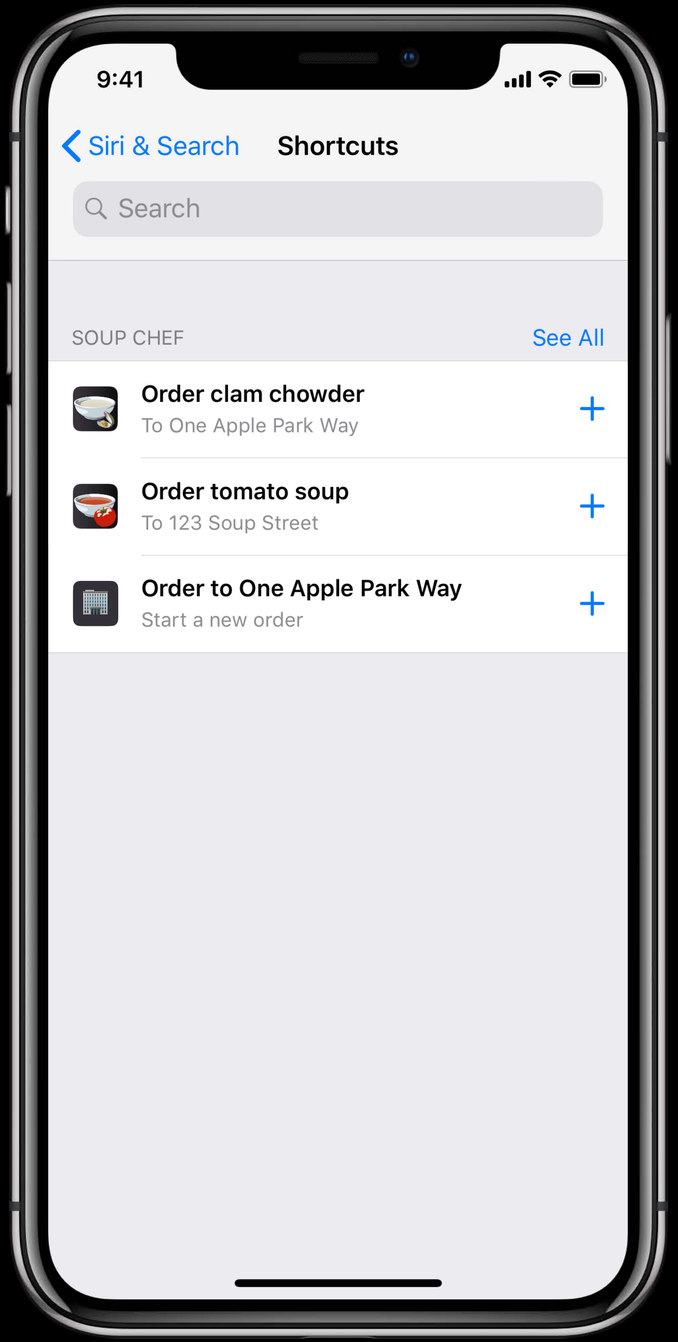
```
// Setting an image on an INIntent

import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
     // Handle error
}
```

```
// Setting an image on an INIntent

import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
     // Handle error
}
```

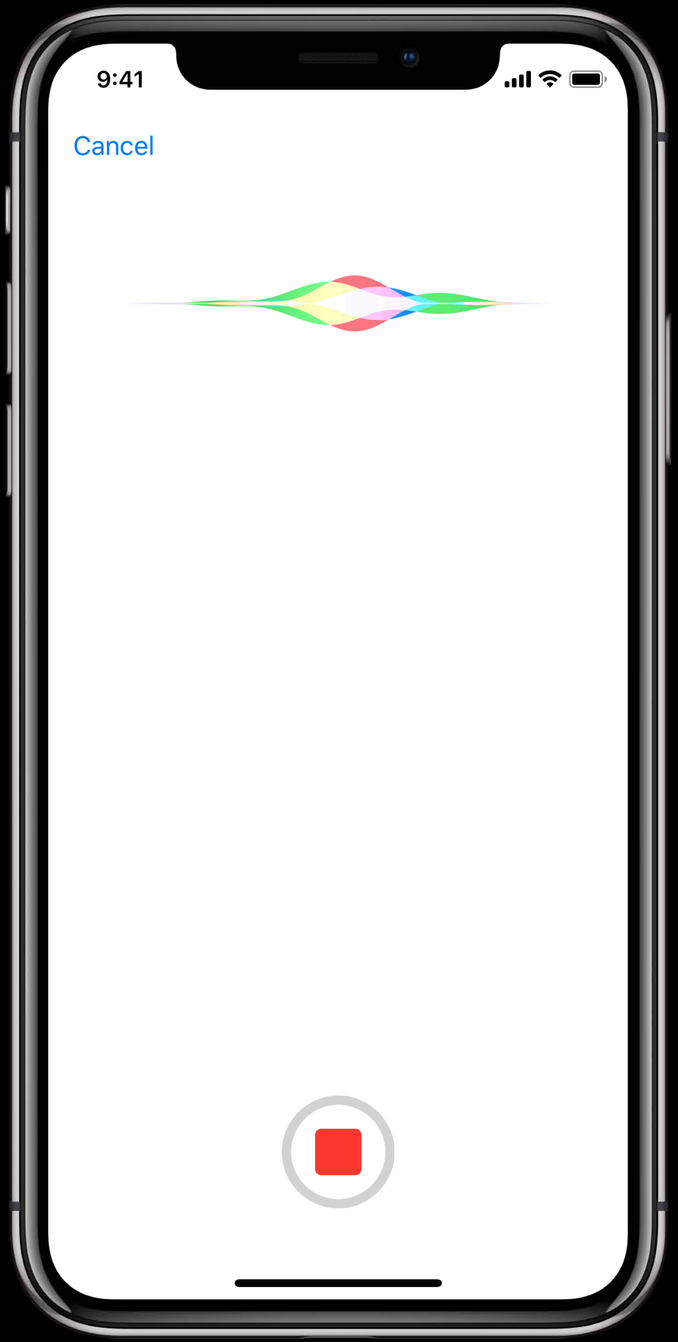
```
// Setting an image on an INIntent

import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
     // Handle error
}
```


SoupChef 〉 iPhone X
Indexing | Prebuilding...
Intents.intentdefinition
SoupChef 〉 Shared 〉 Intents.intentdefinition 〉 Intents.intentdefinition (Base) 〉 Custom Intents 〉 OrderSoup
CUSTOM INTENTS
OrderSoup
Response
Filter
Custom Intent
Category
Order
Title
Order Soup
Description
Order some soup
Confirmation
User confirmation required
Parameters
Parameter
Type
Array
deliveryLocation
Location
soup
Custom
quantity
Integer
options
Custom
Shortcut Types
Parameter Combination
soup
soup, options
soup, quantity, options
soup, quantity
deliveryLocation, soup
deliveryLocation
Title
Order soup
Subtitle
To deliveryLocation
Background
Supports background execution


SoupChef 〉 iPhone X
Indexing | Prebuilding...
Intents.intentdefinition
SoupChef 〉 Shared 〉 Intents.intentdefinition 〉 Intents.intentdefinition (Base) 〉 Custom Intents 〉 OrderSoup
CUSTOM INTENTS
OrderSoup
Response
Custom Intent
Category Order
Title Order Soup
Description Order some soup
Confirmation User confirmation required
Parameters
Parameter
Type
Array
deliveryLocation Location
soup Custom
quantity Integer
options Custom
Parameter Combination
soup
soup, options
soup, quantity, options
soup, quantity
deliveryLocation, soup
deliveryLocation
Title Order soup
Subtitle To deliveryLocation
Background Supports background execution
Filter


9:41
Siri & Search
Shortcuts
Search
SOUP CHEF
See All
Order clam chowder
To One Apple Park Way
Order tomato soup
To 123 Soup Street
Order to One Apple Park Way
Start a new order


9:41
Cancel
Add to Siri
Order clam chowder
To One Apple Park Way
Soup Chef
After you record your personalized phrase, Siri can use it to tell “Soup Chef” to run this shortcut.


9:41
Cancel


9:41
Cancel
You can say something like...
“Chowder time”

```
// Suggesting an invocation phrase on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons" // Subtitle

let image = UIImage(named: "Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

userActivity.suggestedInvocationPhrase = "Chowder time"

viewController.userActivity = userActivity
```

```
// Suggesting an invocation phrase on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons" // Subtitle

let image = UIImage(named: "Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

userActivity.suggestedInvocationPhrase = "Chowder time"

viewController.userActivity = userActivity
```

```
// Suggesting an invocation phrase on an INIntent

import Intents



let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: “soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

intent.suggestedInvocationPhrase = "Chowder time"

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
   // Handle error
}
```

```
// Suggesting an invocation phrase on an INIntent

import Intents



let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: “soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

intent.suggestedInvocationPhrase = "Chowder time"

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
    // Handle error
}
```

Choosing a Good Suggested Invocation Phrase

Choosing a Good Suggested Invocation Phrase

Short and memorable

Choosing a Good Suggested Invocation Phrase

Short and memorable

Don’t include “Hey Siri”


9:41
Cancel
You can say something like...
“Hey Siri, please place an order, thank you”


9:41
Cancel
You can say something like...
“Hey Siri, please place an order, thank you”


9:41
Cancel
You can say something like...
"Order a clam chowder to my office"


9:41
Cancel
You can say something like...
"Order a clam chowder to my office"


9:41
Cancel
You can say something like...
“Chowder time”


9:41
Cancel
You can say something like...
“Chowder time”


9:41
キャンセル
例えばこんな風に言うことができます...
「チャウダー食べたい」

Localization

New Localization Workflows in Xcode 10	Hall 3	WWDC 2018
Internationalization Lab	Technology Lab 9	Wednesday 11:00AM

Localization

Localize both your code and the strings in your intent definition files

New Localization Workflows in Xcode 10	Hall 3	WWDC 2018
Internationalization Lab	Technology Lab 9	Wednesday 11:00AM

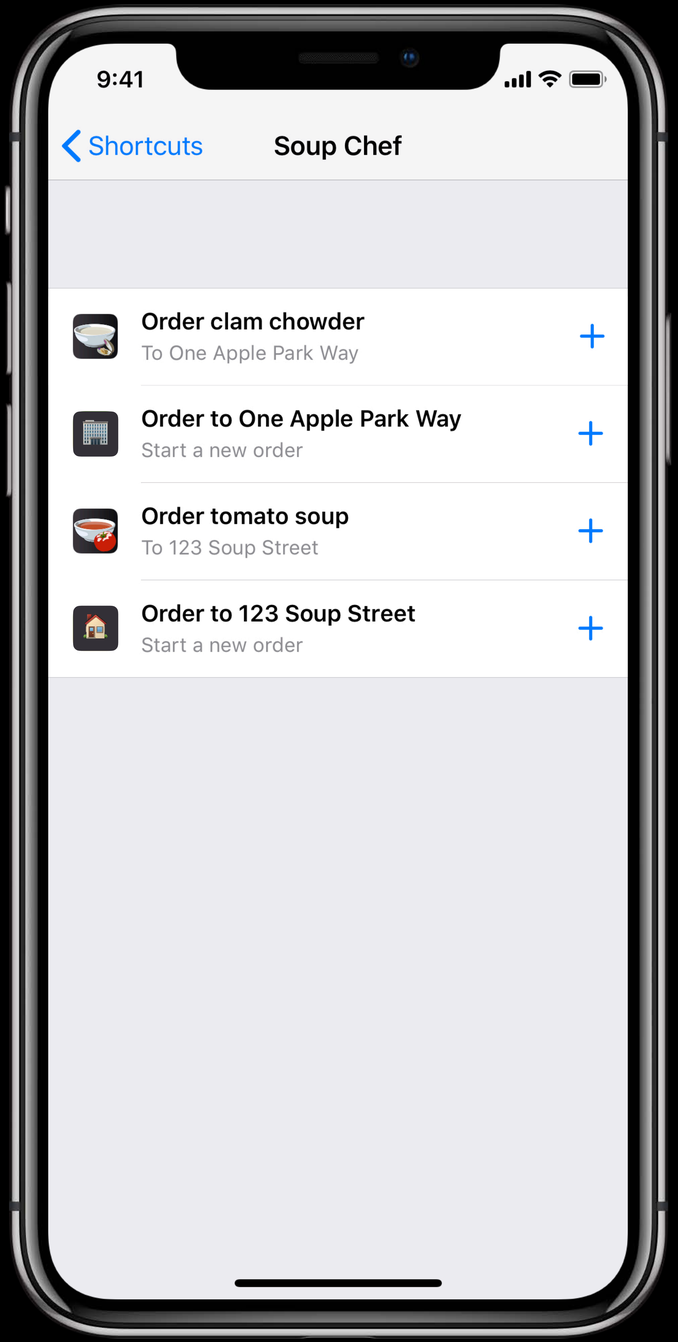
Localization

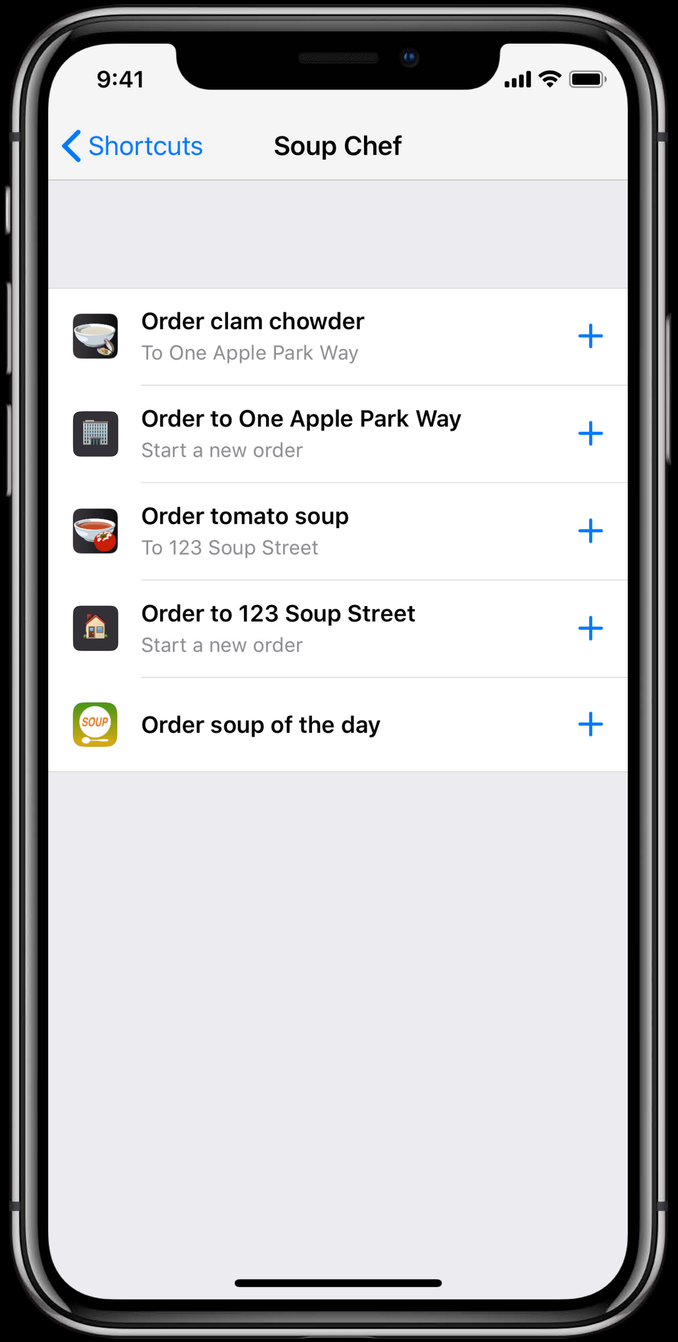
Localize both your code and the strings in your intent definition files

Pluralization

New Localization Workflows in Xcode 10	Hall 3	WWDC 2018
Internationalization Lab	Technology Lab 9	Wednesday 11:00AM


9:41
Cancel
You can say something like...
“Chowder time”


9:41
Shortcuts
Soup Chef
Order clam chowder
To One Apple Park Way
Order to One Apple Park Way
Start a new order
Order tomato soup
To 123 Soup Street
Order to 123 Soup Street
Start a new order


9:41
Shortcuts
Soup Chef
Order clam chowder
To One Apple Park Way
Order to One Apple Park Way
Start a new order
Order tomato soup
To 123 Soup Street
Order to 123 Soup Street
Start a new order
SOUP
Order soup of the day

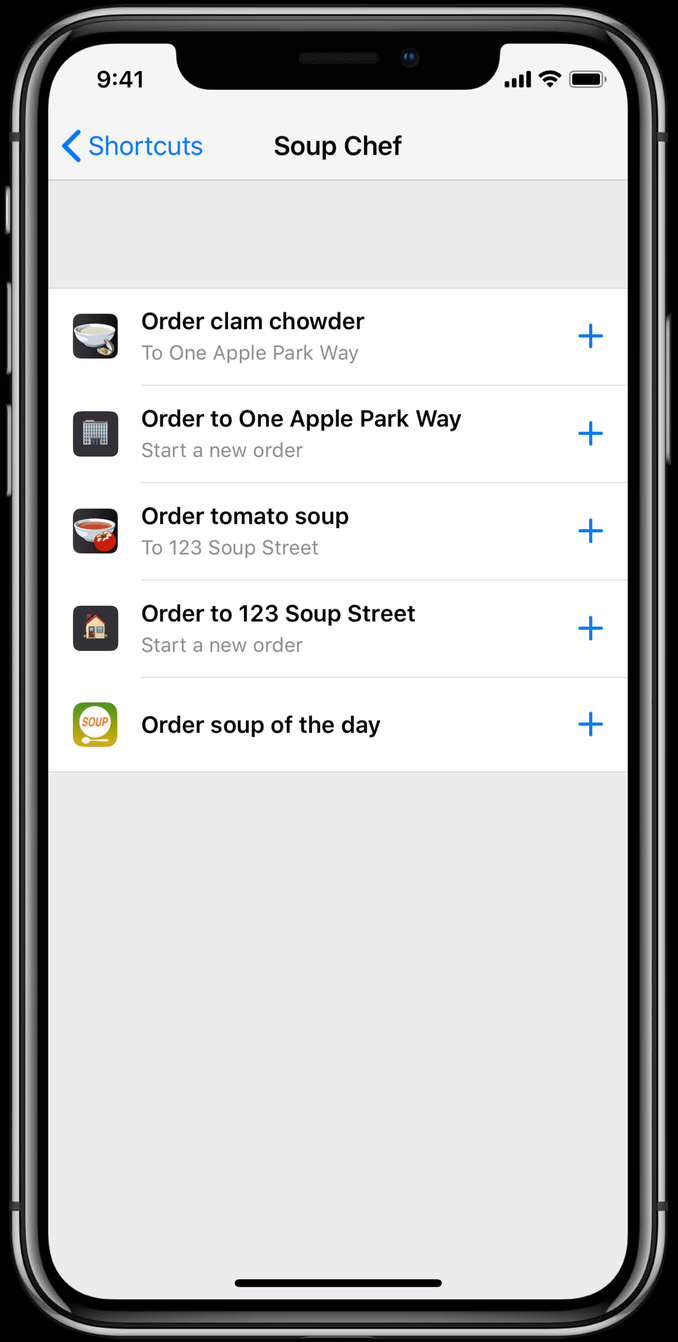
```swift
// How to suggest a shortcut for something a user hasn't done yet

import Intents

let suggestions = [
    INShortcut(intent: orderFavoriteSoupIntent),
    INShortcut(userActivity: orderFavoriteBeverageUserActivity),
]


INVoiceShortcutCenter.shared.setShortcutSuggestions(suggestions)
```

```
// How to suggest a shortcut for something a user hasn't done yet

import Intents

let suggestions = [
    INShortcut(intent: orderFavoriteSoupIntent),
    INShortcut(userActivity: orderFavoriteBeverageUserActivity),
]


INVoiceShortcutCenter.shared.setShortcutSuggestions(suggestions)
```

```
// How to suggest a shortcut for something a user hasn't done yet

import Intents

let suggestions = [
    INShortcut(intent: orderFavoriteSoupIntent),
    INShortcut(userActivity: orderFavoriteBeverageUserActivity),
]


INVoiceShortcutCenter.shared.setShortcutSuggestions(suggestions)
```

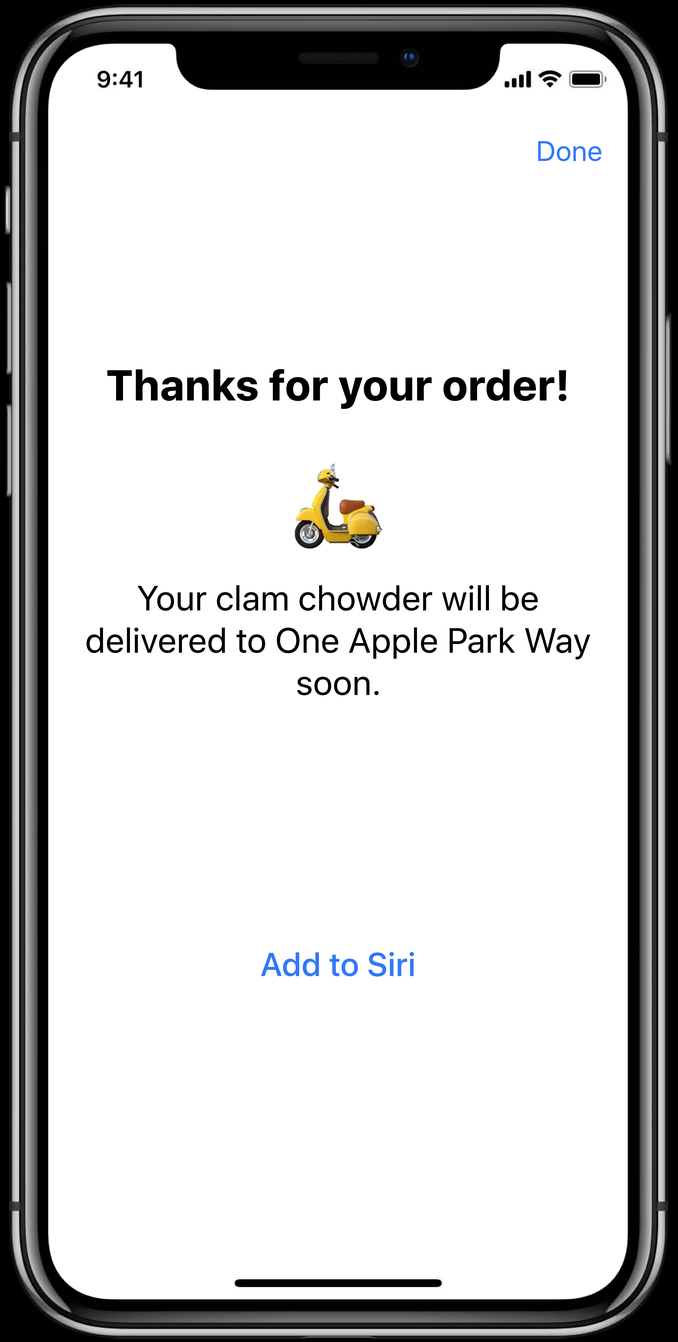
```swift
// How to suggest a shortcut for something a user hasn't done yet

import Intents

let suggestions = [
    INShortcut(intent: orderFavoriteSoupIntent),
    INShortcut(userActivity: orderFavoriteBeverageUserActivity),
]


INVoiceShortcutCenter.shared.setShortcutSuggestions(suggestions)
```

```
// How to suggest a shortcut for something a user hasn't done yet

import Intents

let suggestions = [
    INShortcut(intent: orderFavoriteSoupIntent),
    INShortcut(userActivity: orderFavoriteBeverageUserActivity),
]


INVoiceShortcutCenter.shared.setShortcutSuggestions(suggestions)
```


9:41
Shortcuts
Soup Chef
Order clam chowder
To One Apple Park Way
Order to One Apple Park Way
Start a new order
Order tomato soup
To 123 Soup Street
Order to 123 Soup Street
Start a new order
SOUP
Order soup of the day


9:41
Lunch Time
Done
Order soup of the day
Text
It’s soup time!
Send Message
Recipients
Amit Jain,
Soup Chef
Order clam chowder
To One Apple Park Way
Order to One Apple Park Way
Start a new order
Order tomato soup
To 123 Soup Street
Order to 123 Soup Street
Start a new order

Bringing Shortcuts Into Your App


9:41
Done
Thanks for your order!
Your clam chowder will be delivered to One Apple Park Way soon.
Add to Siri


9:41
Cancel
Add to Siri
Order clam chowder
To One Apple Park Way
Soup Chef
You can say something like...
“Chowder time”
After you record your personalized phrase, Siri can use it to tell “Soup Chef” to run this shortcut.

```swift
// Adding a shortcut from an NSUserActivity

import IntentsUI

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")

// Configure your NSUserActivity

let shortcut = INShortcut(userActivity: userActivity)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
```

```swift
// Adding a shortcut from an NSUserActivity

import IntentsUI

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")

// Configure your NSUserActivity

let shortcut = INShortcut(userActivity: userActivity)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
```

```
// Adding a shortcut from an NSUserActivity

import IntentsUI

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")

// Configure your NSUserActivity

let shortcut = INShortcut(userActivity: userActivity)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
```

```swift
// Adding a shortcut from an NSUserActivity

import IntentsUI

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")

// Configure your NSUserActivity

let shortcut = INShortcut(userActivity: userActivity)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
```

```swift
// Adding a shortcut from an NSUserActivity

import IntentsUI

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")

// Configure your NSUserActivity

let shortcut = INShortcut(userActivity: userActivity)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
```

```swift
// Adding a shortcut from an INIntent

import IntentsUI

let intent = PlaceOrderIntent()

// Configure your INIntent

let shortcut = INShortcut(intent: intent)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
```

```swift
// Adding a shortcut from an INIntent

import IntentsUI

let intent = PlaceOrderIntent()

// Configure your INIntent

let shortcut = INShortcut(intent: intent)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
```

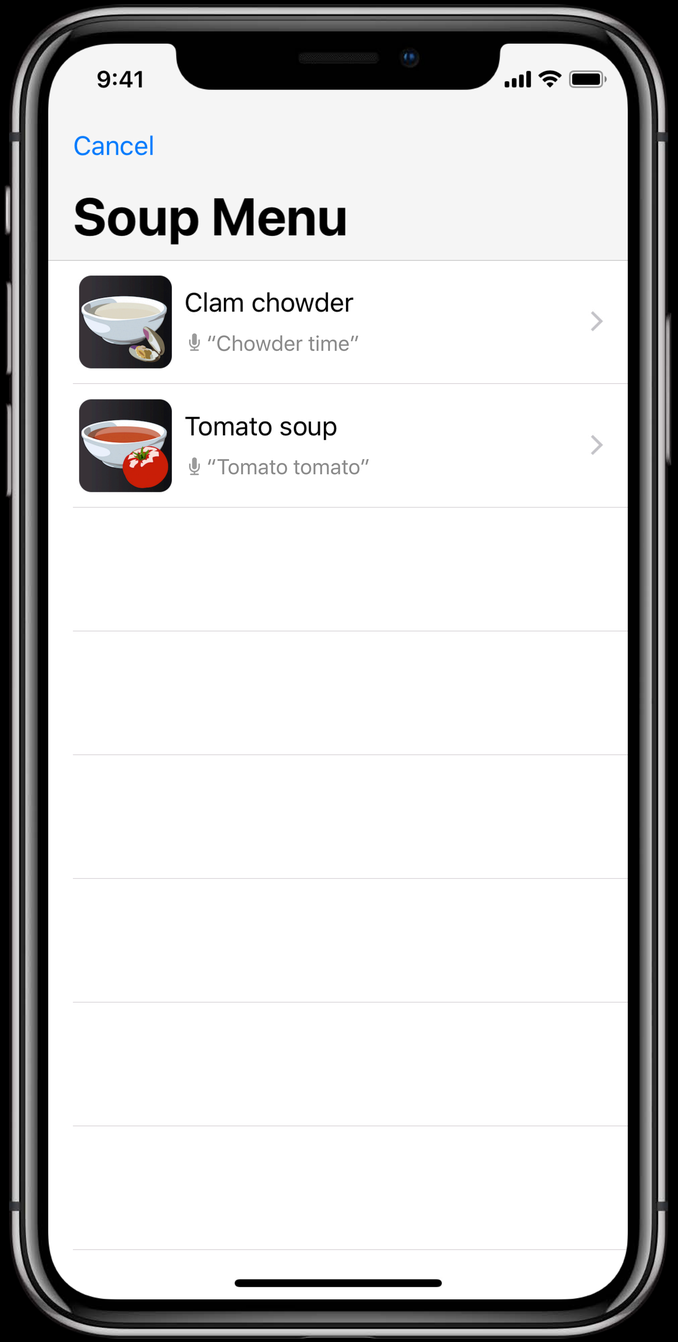
```swift
// Adding a shortcut from an INIntent

import IntentsUI

let intent = PlaceOrderIntent()

// Configure your INIntent

let shortcut = INShortcut(intent: intent)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
```

```swift
// Adding a shortcut from an INIntent

import IntentsUI

let intent = PlaceOrderIntent()

// Configure your INIntent

let shortcut = INShortcut(intent: intent)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
```

```swift
// Adding a shortcut from an INIntent

import IntentsUI

let intent = PlaceOrderIntent()

// Configure your INIntent

let shortcut = INShortcut(intent: intent)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
```


9:41
Cancel
Add to Siri
Order clam chowder
To One Apple Park Way
Soup Chef
You can say something like...
“Chowder time”
After you record your personalized phrase, Siri can use it to tell “Soup Chef” to run this shortcut.


9:41
Cancel
Order clam chowder
To One Apple Park Way
Soup Chef
“Chowder time”
Edit
Delete Shortcut


9:41
Cancel
Soup Menu
Clam chowder
“Chowder time”
Tomato soup
“Tomato tomato”

Summary

Summary

Custom responses

Summary

Custom responses

Create rich intents UI

Summary

Custom responses

Create rich intents UI

Details matter

Summary

Custom responses

Create rich intents UI

Details matter

Bring shortcuts into your app

More Information

https://developer.apple.com/wwdc2018/214

Siri Shortcuts on the Siri Watch Face	Hall 3	Wednesday 11:00AM
Shortcuts Lab	Technology Lab 7	Wednesday 12:00PM
Shortcuts Lab	Technology Lab 7	Thursday 11:00 AM


WWDC18