

#WWDC18

Siri Shortcuts on the Siri Watch Face

Session 217

Paul Salzman, watchOS Engineer

Josh Ford, watchOS Engineer



WED 6



10:09

Lafayette



11AM

11:30AM

Meet with Ryan

Philz Coffee



WED 6



10:09

Lafayette



11AM

11:30AM

Meet with Ryan

Philz Coffee



WED 6



10:09



10% Happier



Your daily dose is here



Lose It!



Log carrots and hummus



WED 6



10:09



10% Happier



Your daily
dose is here



Lose It!



Log carrots
and hummus



Daily 10%

10:09



**Today's
Meditation**



JUNE 6

Lucy will help you
ease into the week by
guiding you toward



WED 6



10:09



10% Happier



Your daily
dose is here



Lose It!



Log carrots
and hummus



Done

Appearing on the Siri watch face

Appearing on the Siri watch face

Relevant Shortcuts

Appearing on the Siri watch face

Relevant Shortcuts

Relevant Shortcuts from iOS

Appearing on the Siri watch face

Relevant Shortcuts

Relevant Shortcuts from iOS

Predicting Shortcuts on watchOS

Appearing on the Siri watch face

Relevant Shortcuts

Relevant Shortcuts from iOS

Predicting Shortcuts on watchOS

Making a great experience

Appearing on the Siri Watch Face

Your Shortcuts on Siri Watch Face

Up Next

More Relevant

Least Relevant

Your Shortcuts on Siri Watch Face

Sorted by relevance

Up Next

More Relevant

Least Relevant

Your Shortcuts on Siri Watch Face

Sorted by relevance

Provide `INRelevantShortcut`

Up Next

More Relevant

Least Relevant

Your Shortcuts on Siri Watch Face

Sorted by relevance

Provide `INRelevantShortcut`

Optional relevance suggestions

Up Next

More Relevant

Least Relevant

Your Shortcuts on Siri Watch Face

Sorted by relevance

Provide `INRelevantShortcut`

Optional relevance suggestions

Disable/enable source in iOS face customization

Up Next

More Relevant

Least Relevant

Development Configuration

Development Configuration

Settings > Developer

SHORTCUTS TESTING

Display Recent Shortcuts



When enabled, Siri Suggestions in Search & Siri Watch Face show the most recently provided shortcuts rather than current predictions.

Display Donations on Lock Screen



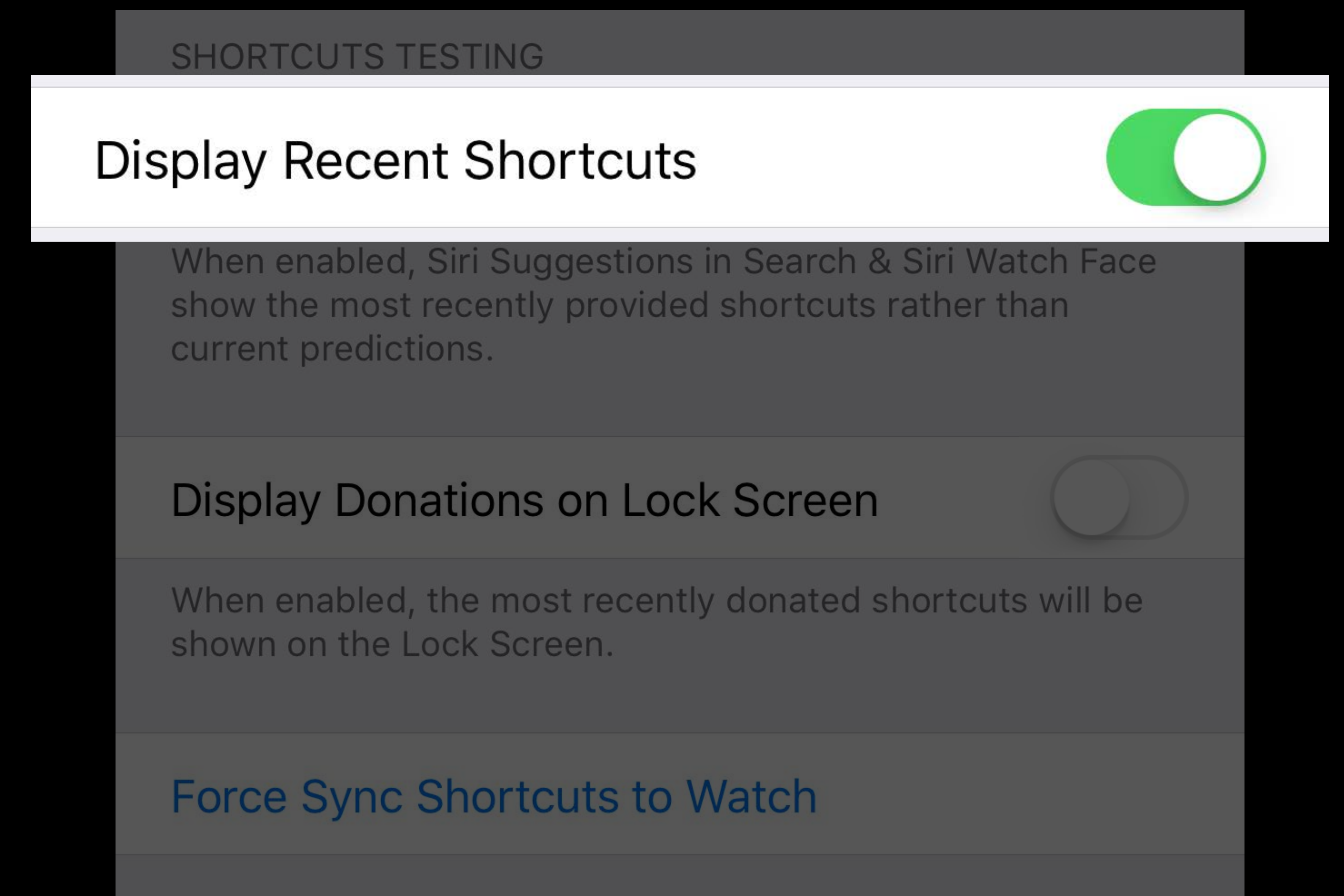
When enabled, the most recently donated shortcuts will be shown on the Lock Screen.

[Force Sync Shortcuts to Watch](#)

Development Configuration

Settings > Developer

Skip relevance calculations



Development Configuration

Settings > Developer

Skip relevance calculations

Bringing iOS shortcuts to watchOS now

SHORTCUTS TESTING

Display Recent Shortcuts



When enabled, Siri Suggestions in Search & Siri Watch Face show the most recently provided shortcuts rather than current predictions.

Display Donations on Lock Screen



When enabled, the most recently donated shortcuts will be shown on the Lock Screen.

[Force Sync Shortcuts to Watch](#)

Relevant Shortcuts

Shortcuts

Shortcuts

Quick access to key functionality

Shortcuts

Quick access to key functionality

Accessible via voice and UI

Shortcuts

Quick access to key functionality

Accessible via voice and UI

Introduction to Siri Shortcuts

WWDC 2018

Building for Voice with Siri Shortcuts

WWDC 2018

Shortcuts

Shortcuts

User activities

Shortcuts

User activities

Intents

Shortcuts

User activities

Intents

- Built-in

Shortcuts

User activities

Intents

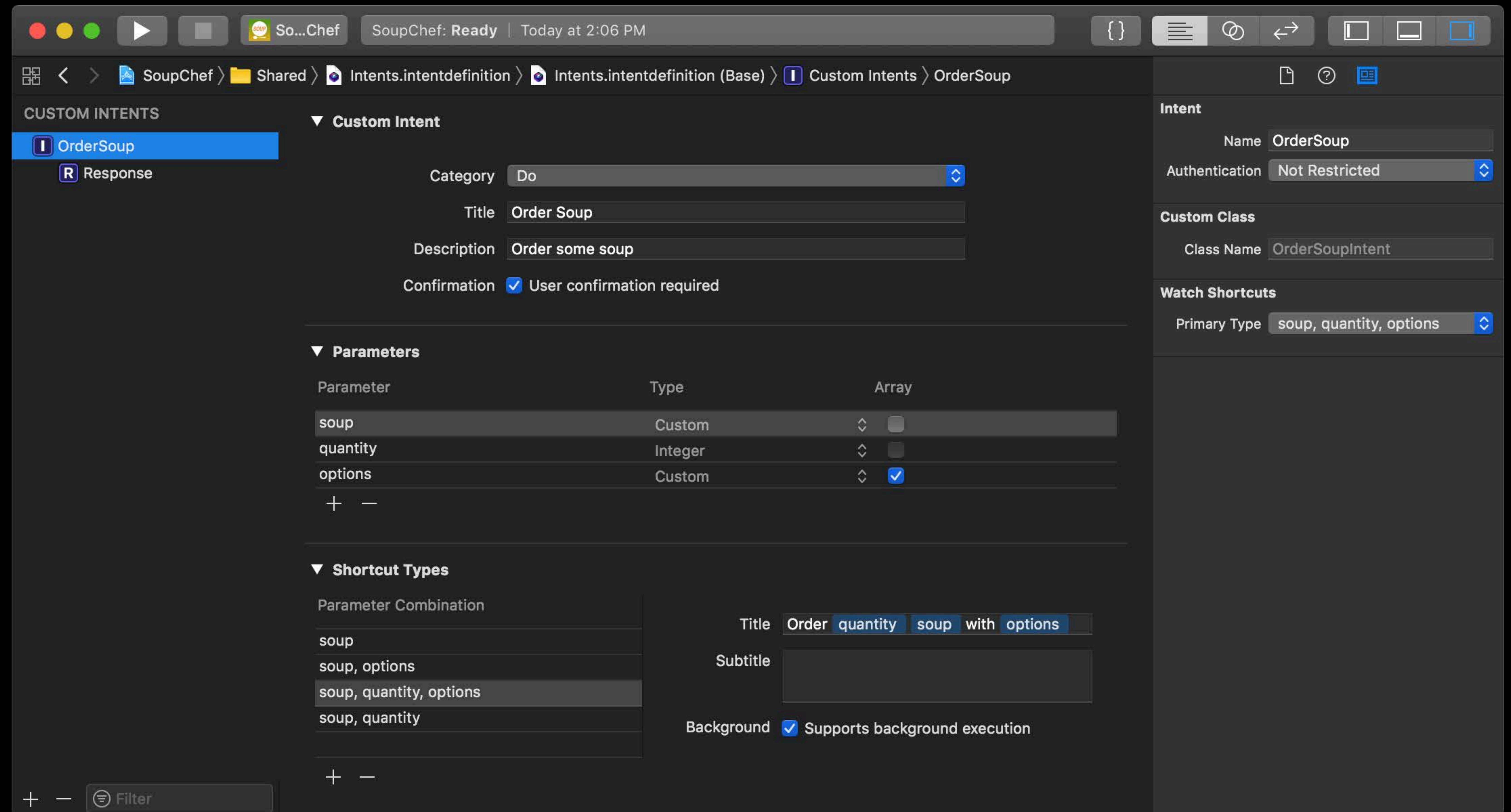
- Built-in
- Custom

Shortcuts

User activities

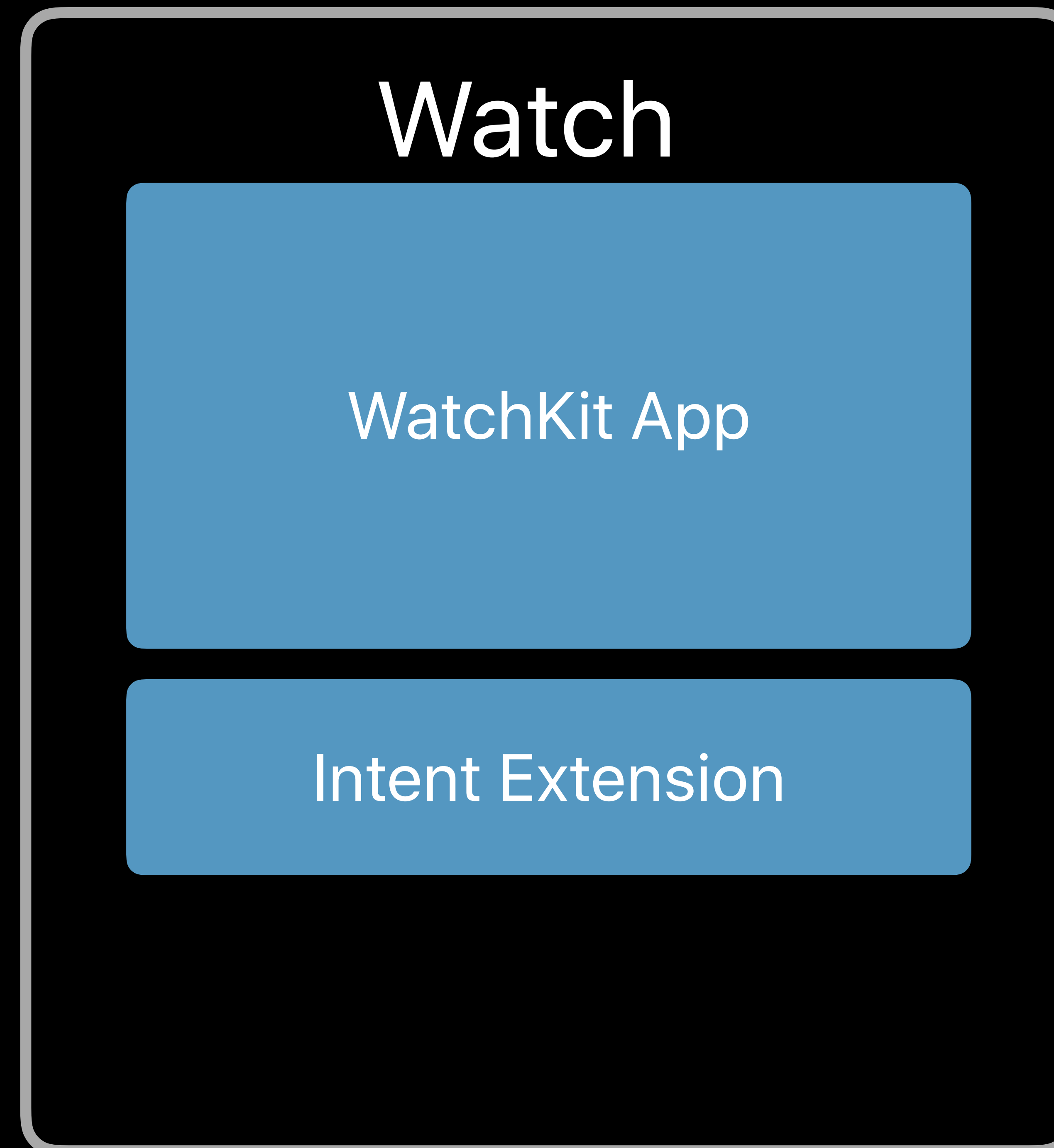
Intents

- Built-in
- Custom



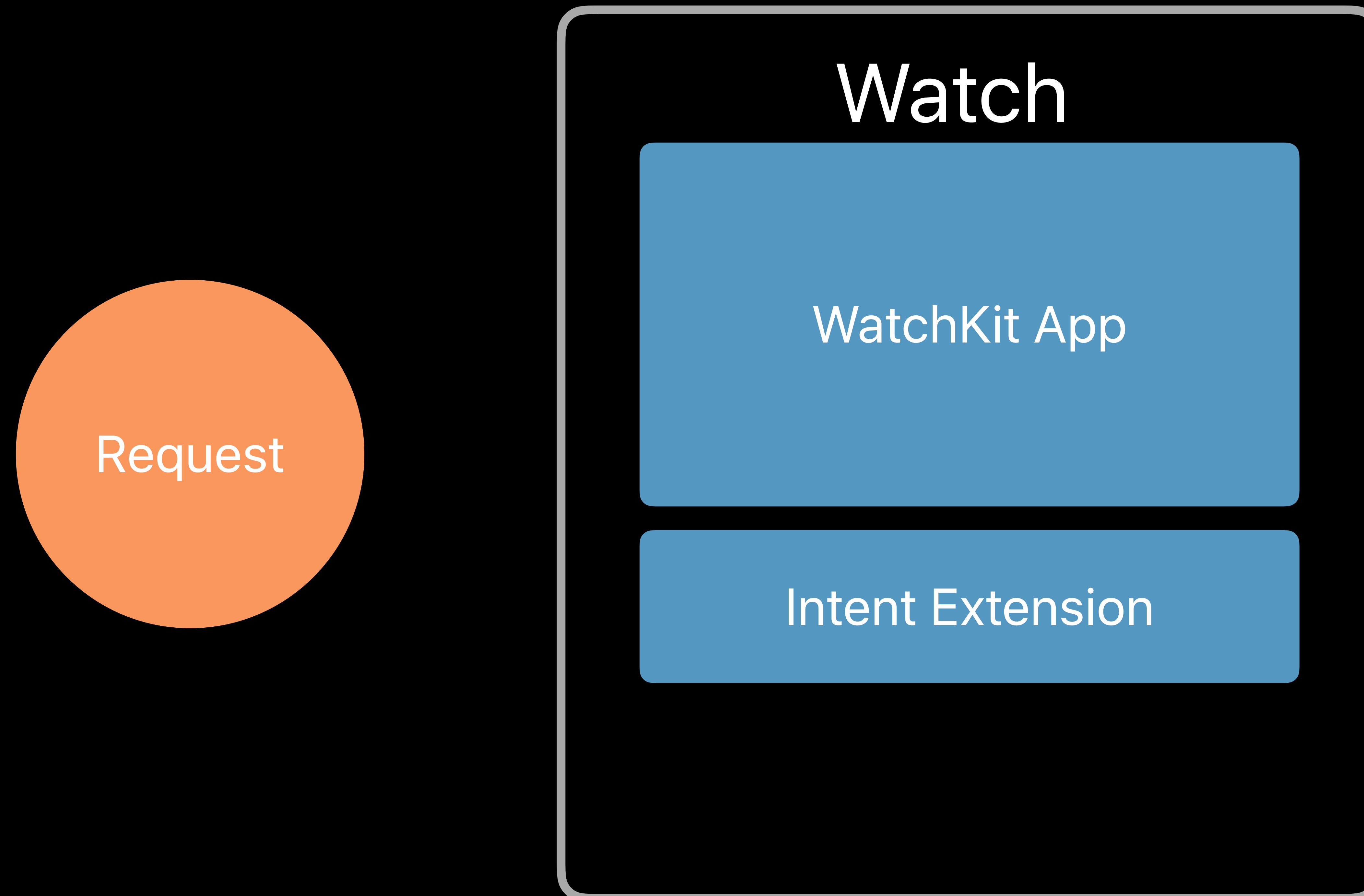
Running Shortcuts on Watch

Supported by watch app



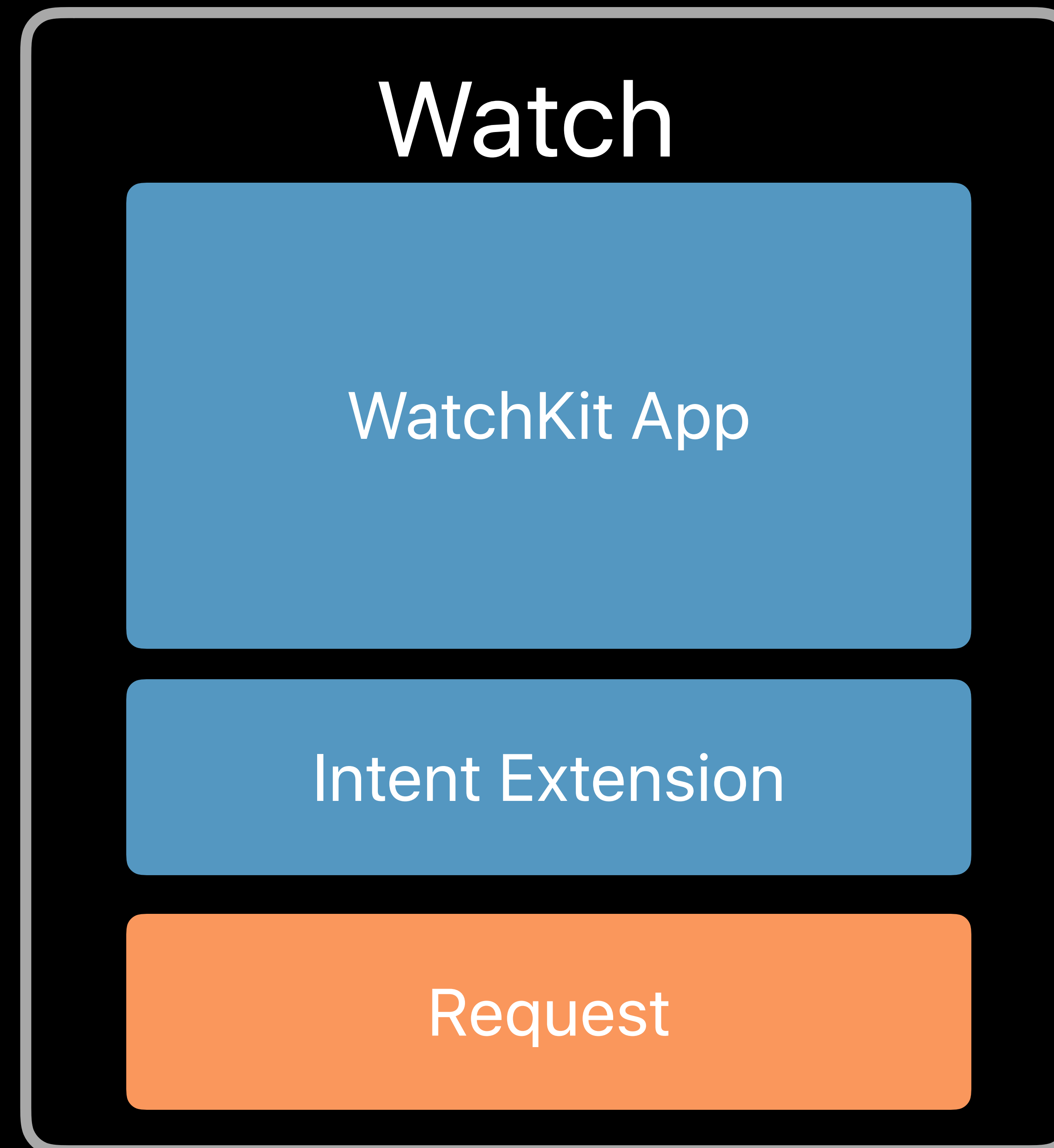
Running Shortcuts on Watch

Supported by watch app



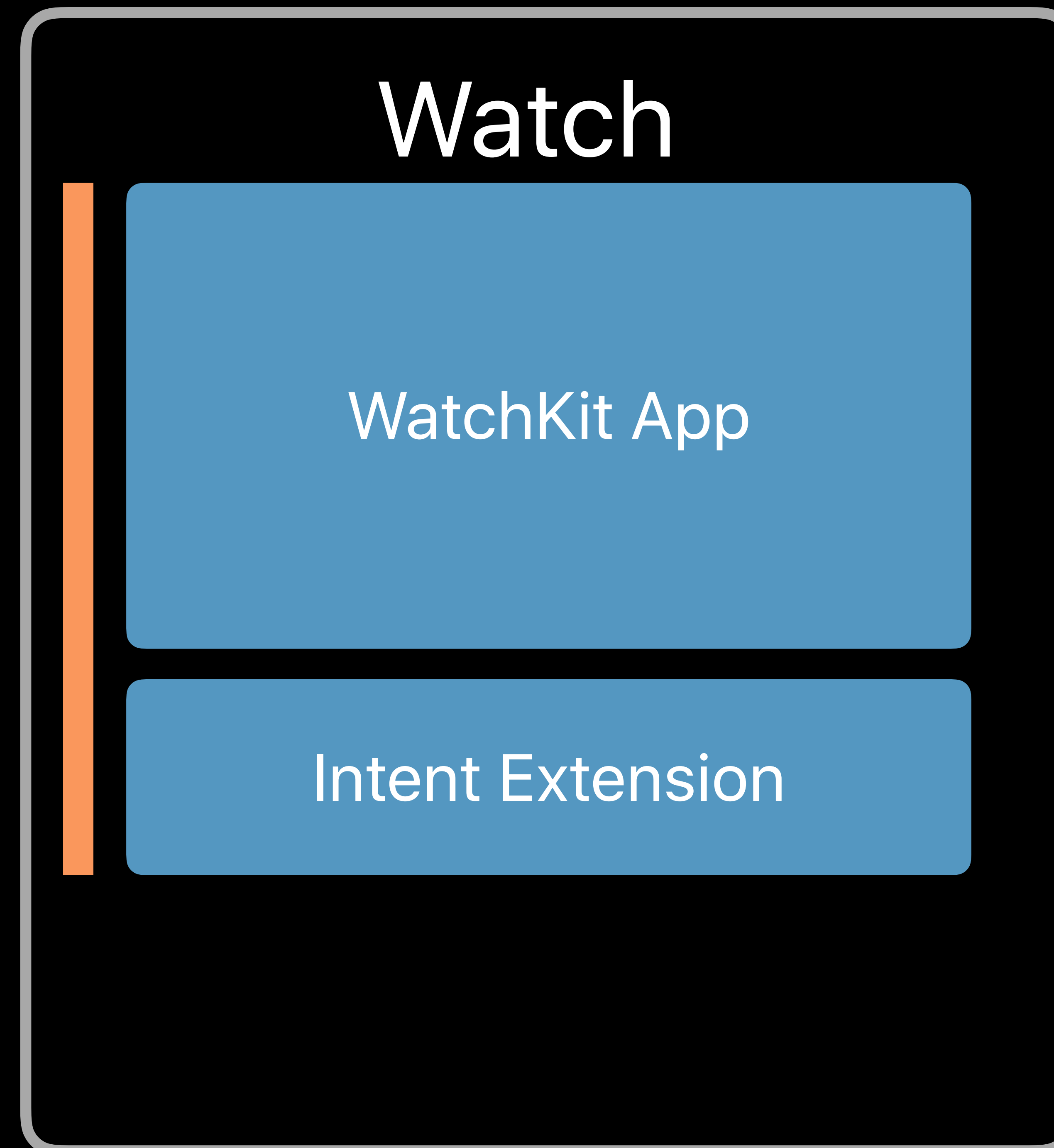
Running Shortcuts on Watch

Supported by watch app



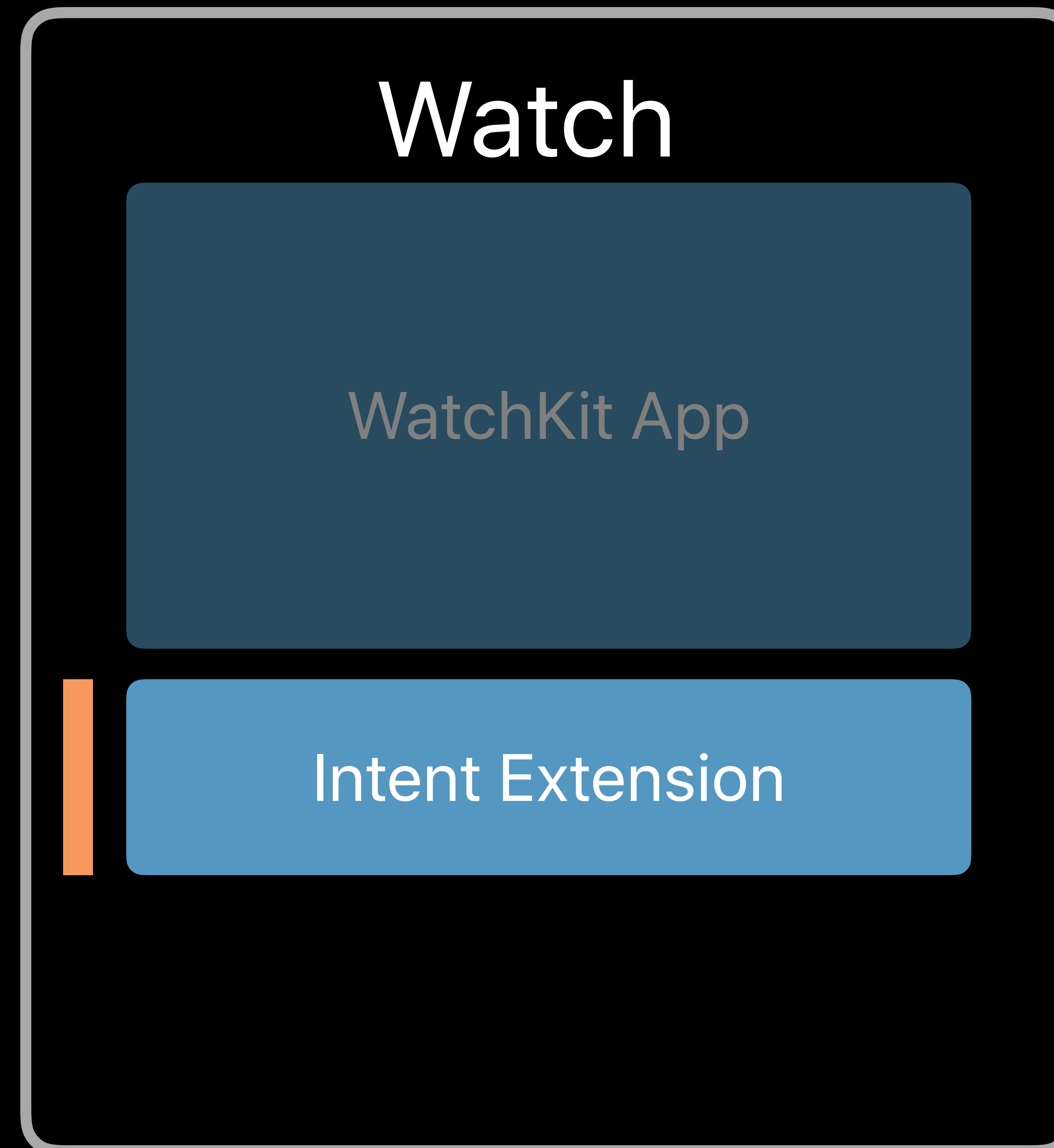
Running Shortcuts on Watch

Supported by watch app



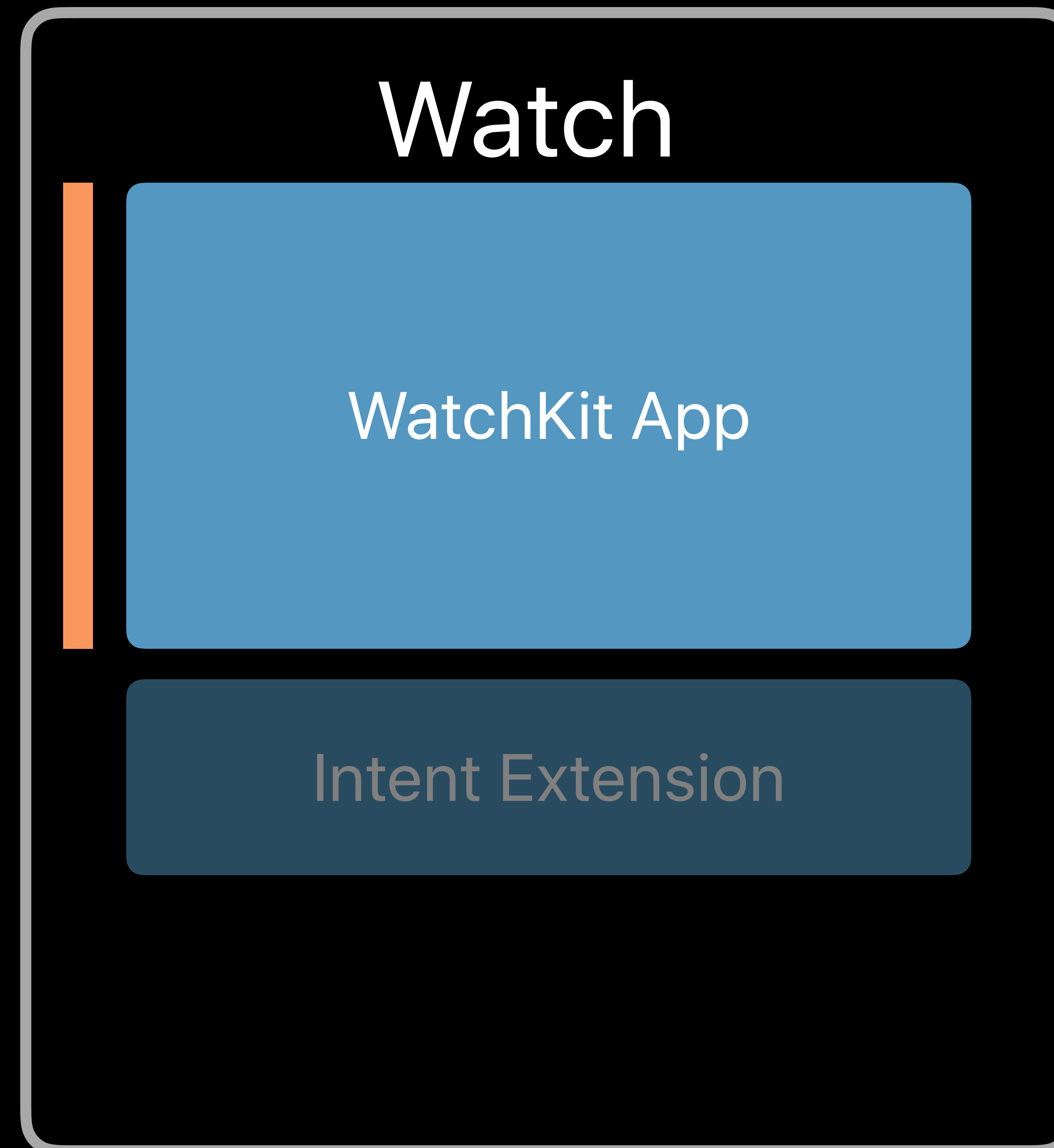
Running Shortcuts on Watch

Supported by watch app



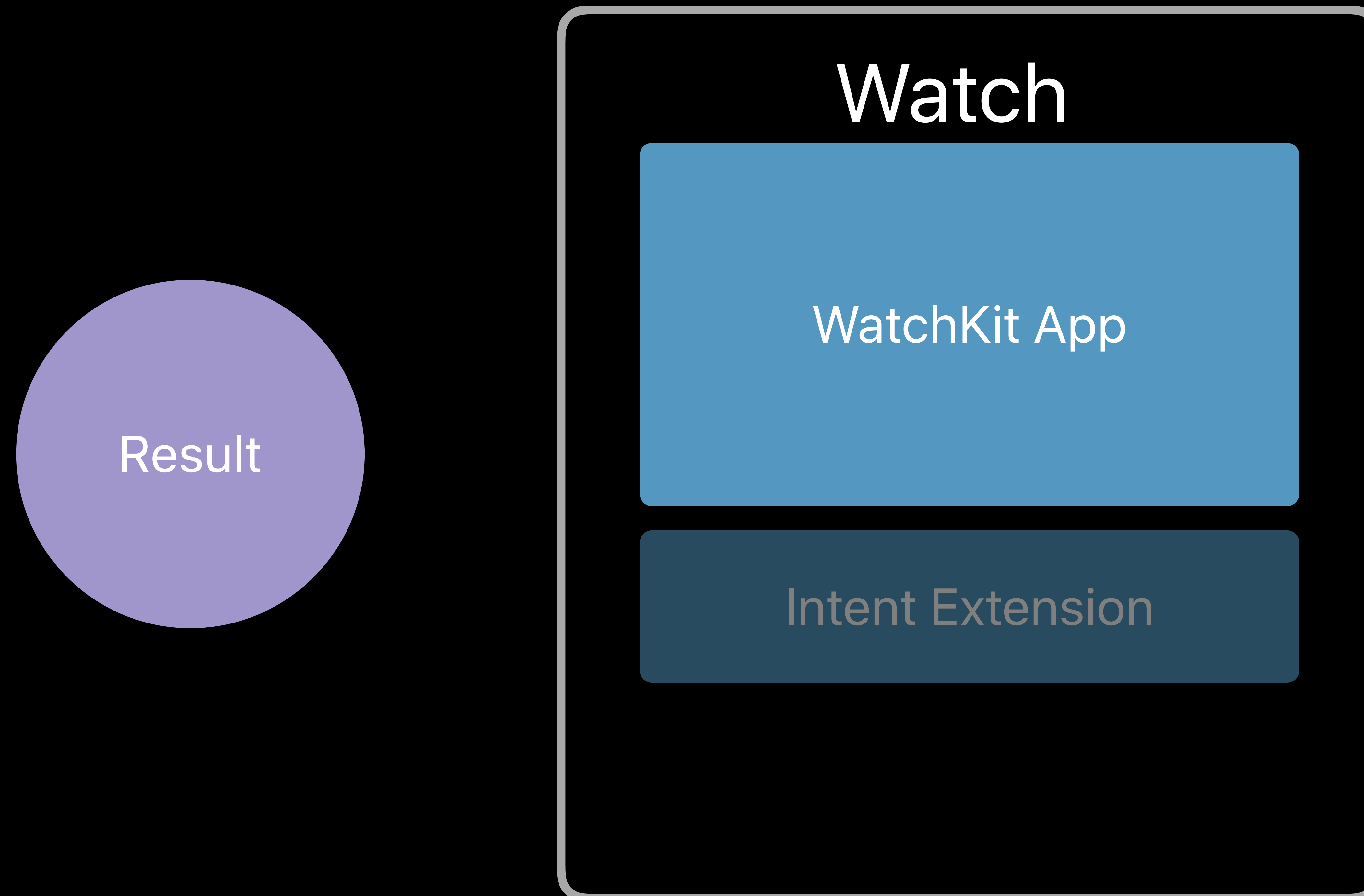
Running Shortcuts on Watch

Supported by watch app



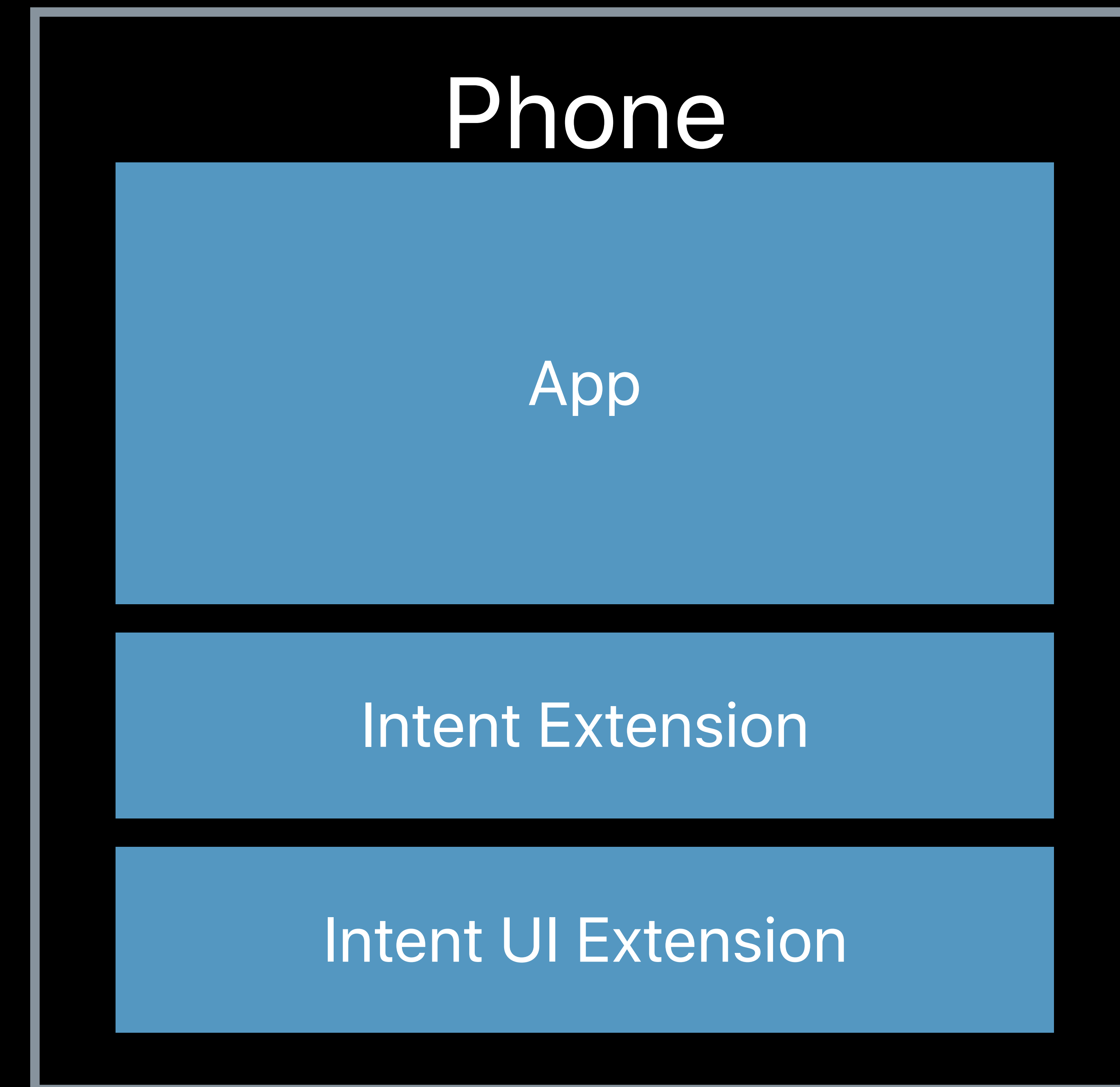
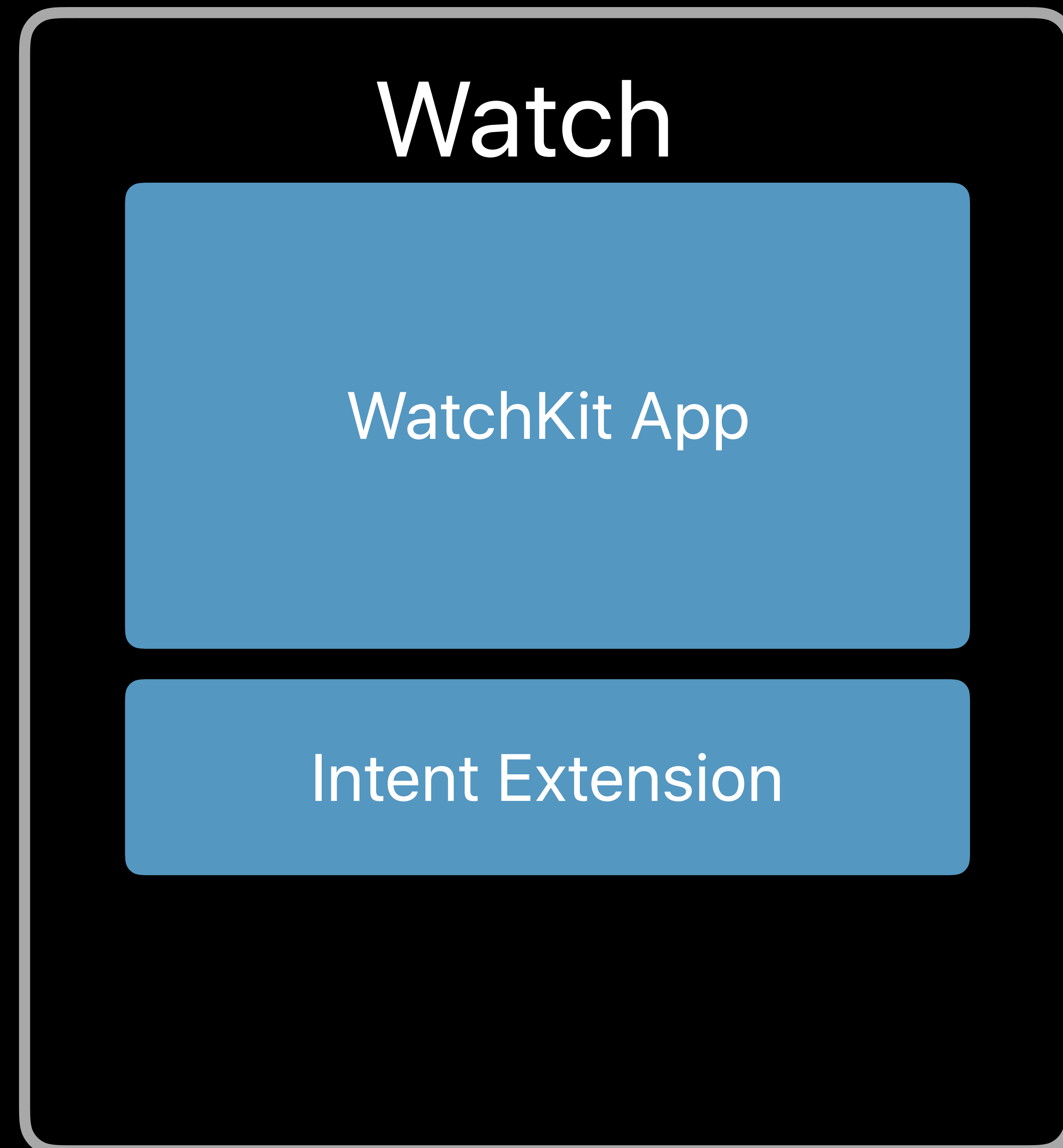
Running Shortcuts on Watch

Supported by watch app



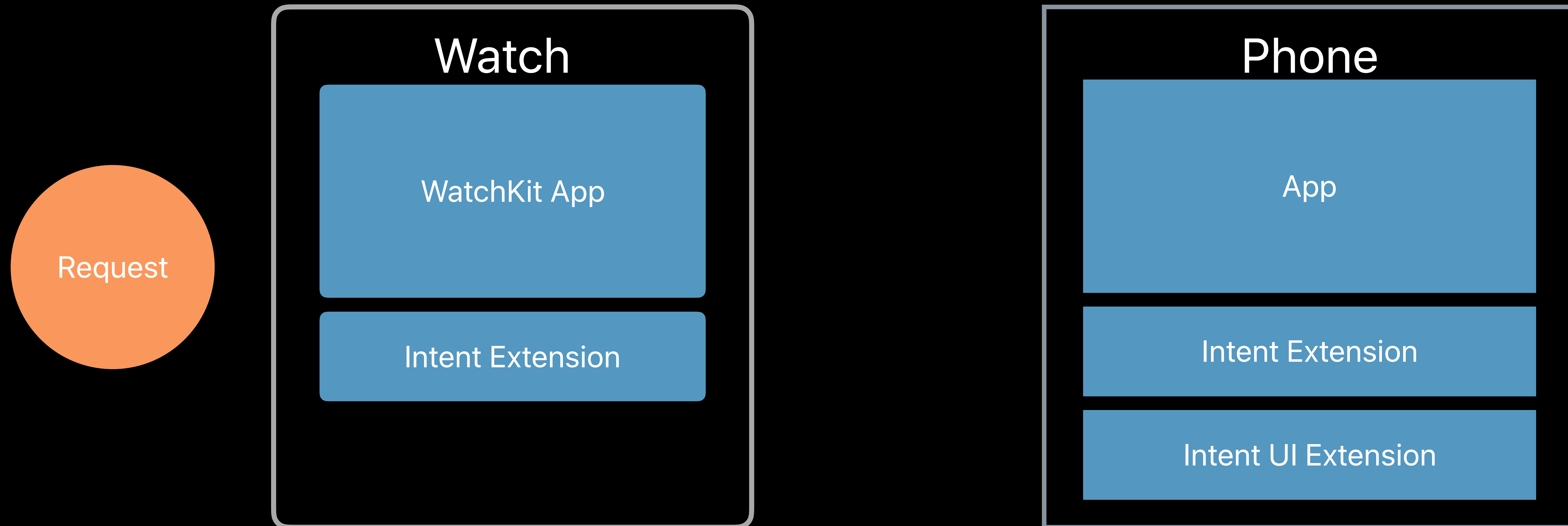
Running Shortcuts on Watch

Not supported by watch app



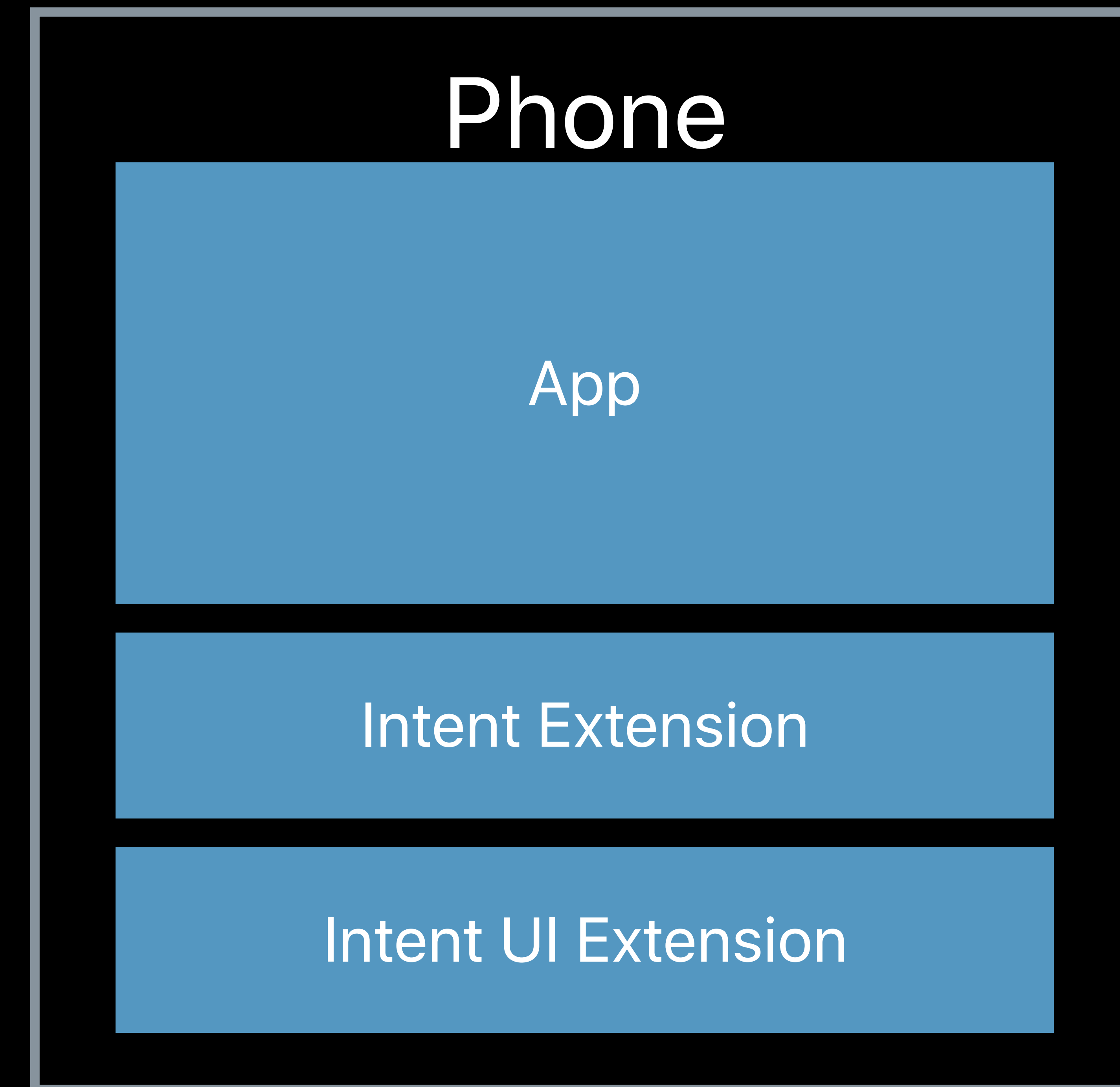
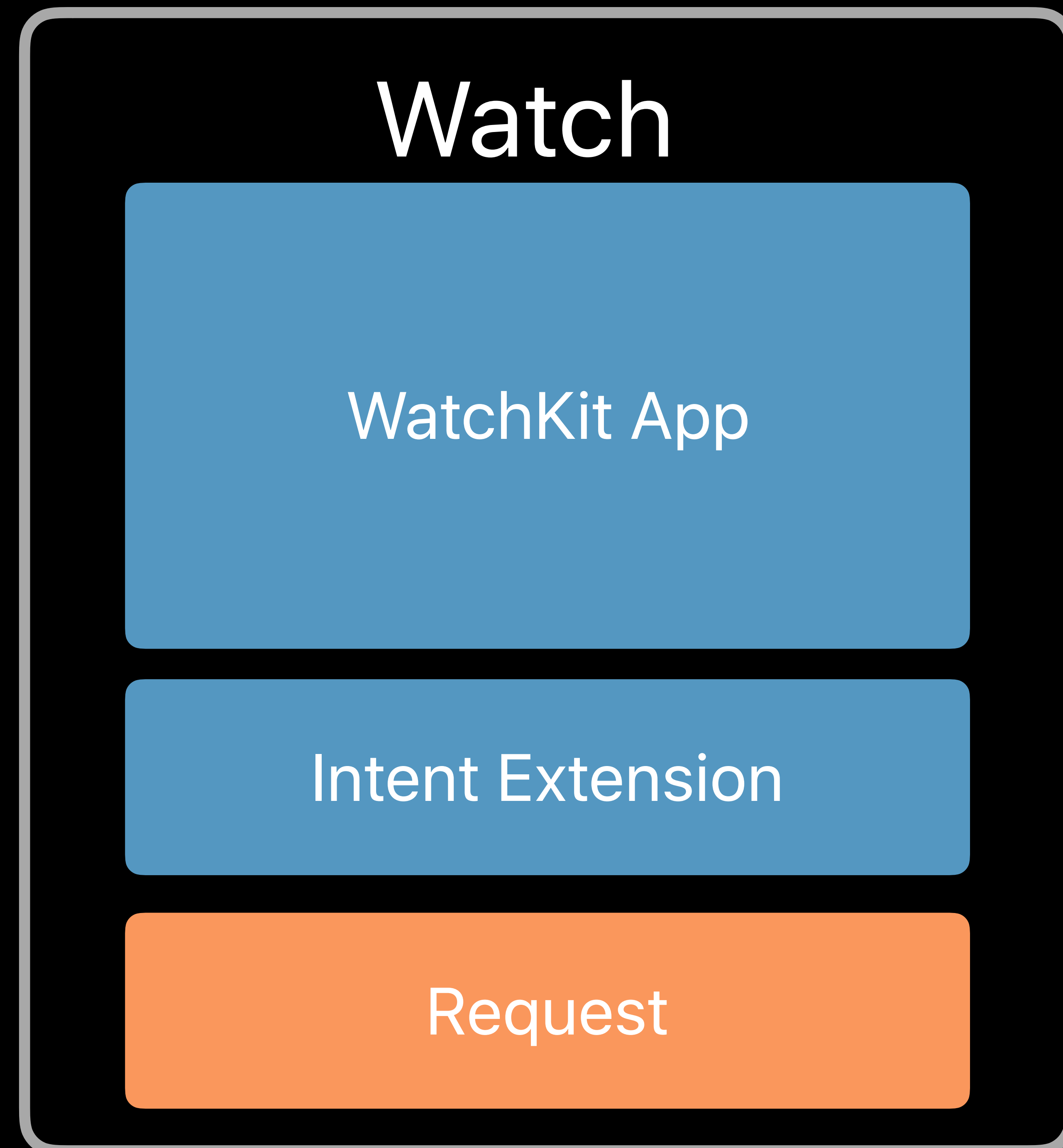
Running Shortcuts on Watch

Not supported by watch app



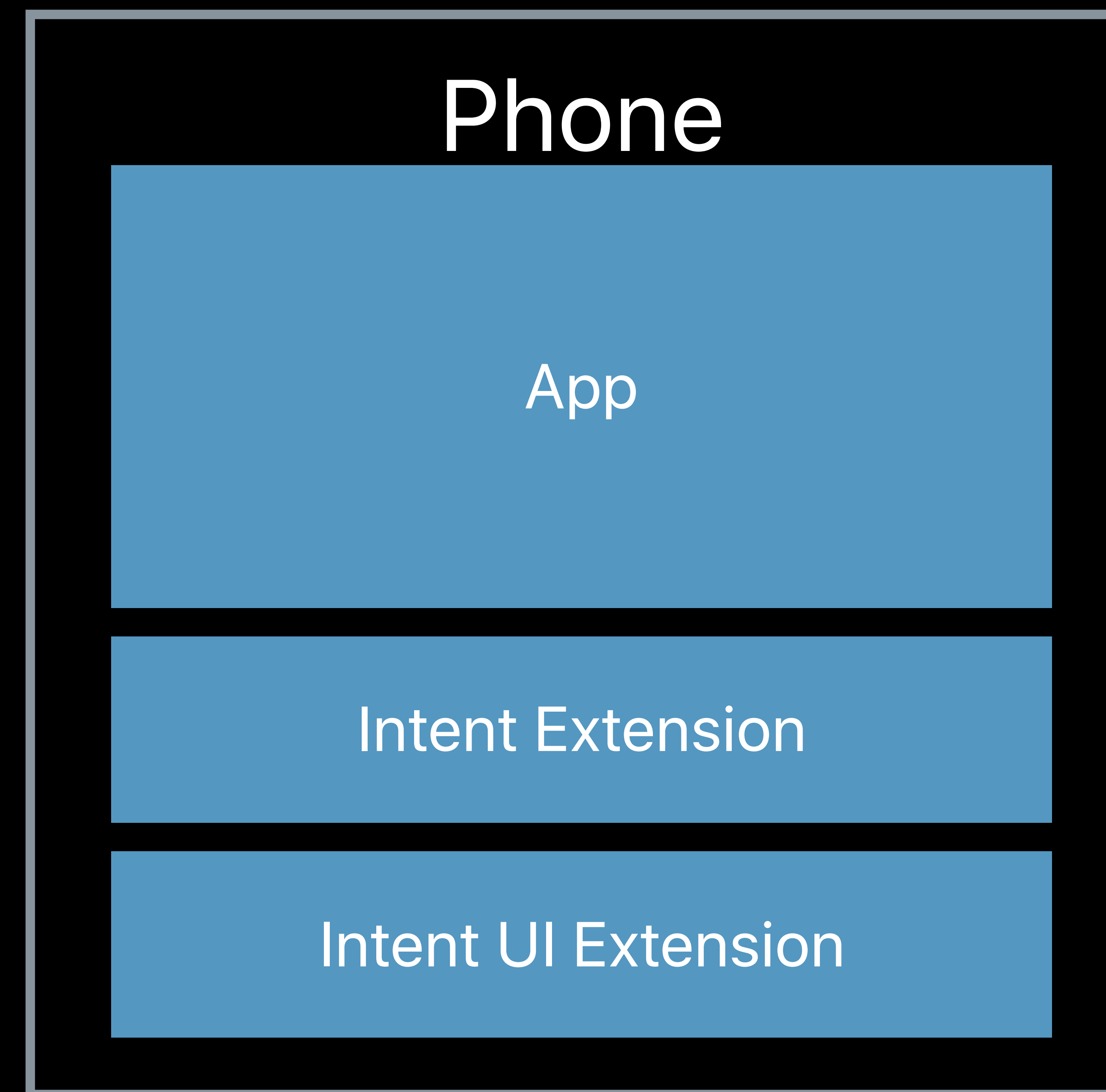
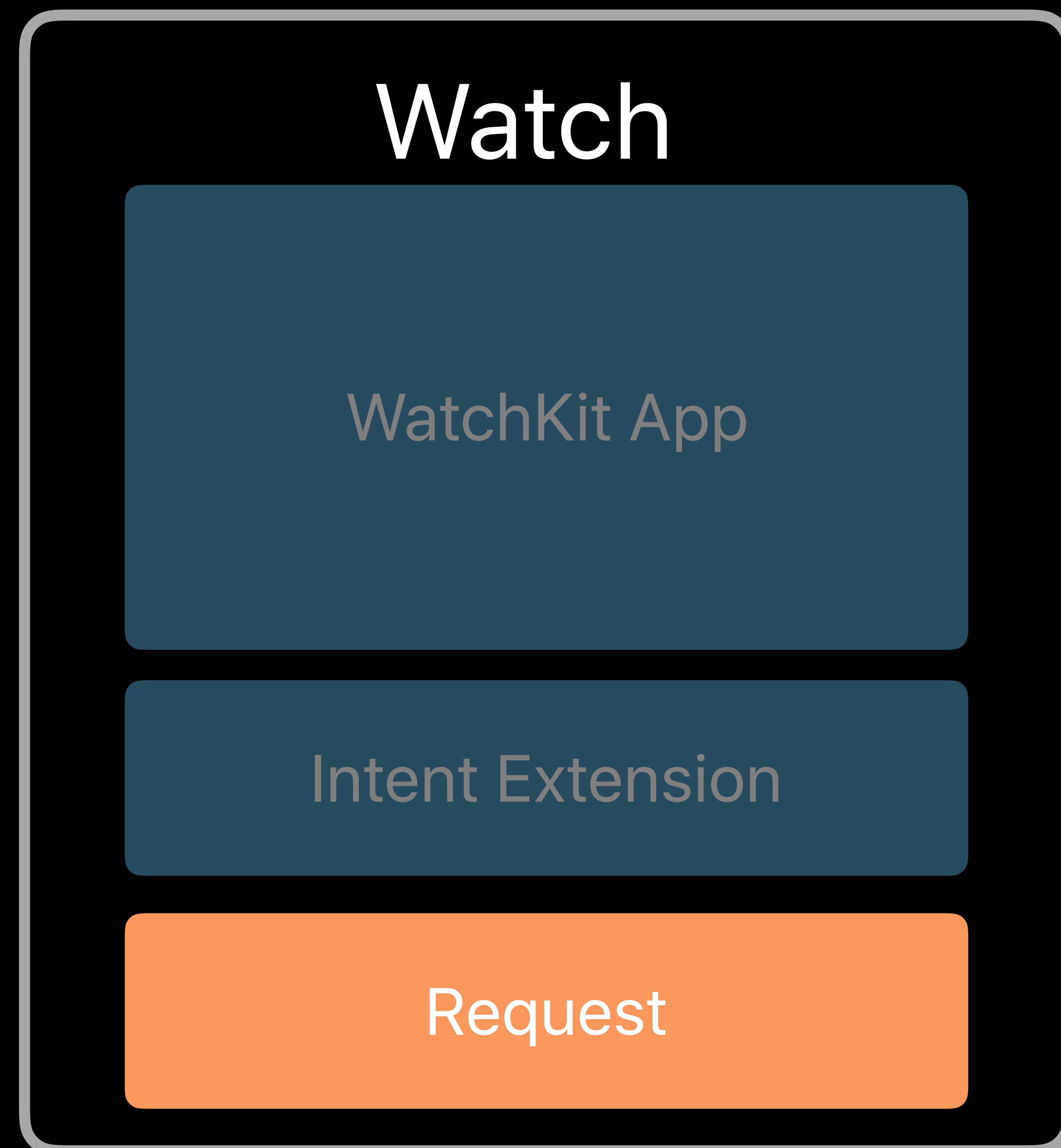
Running Shortcuts on Watch

Not supported by watch app



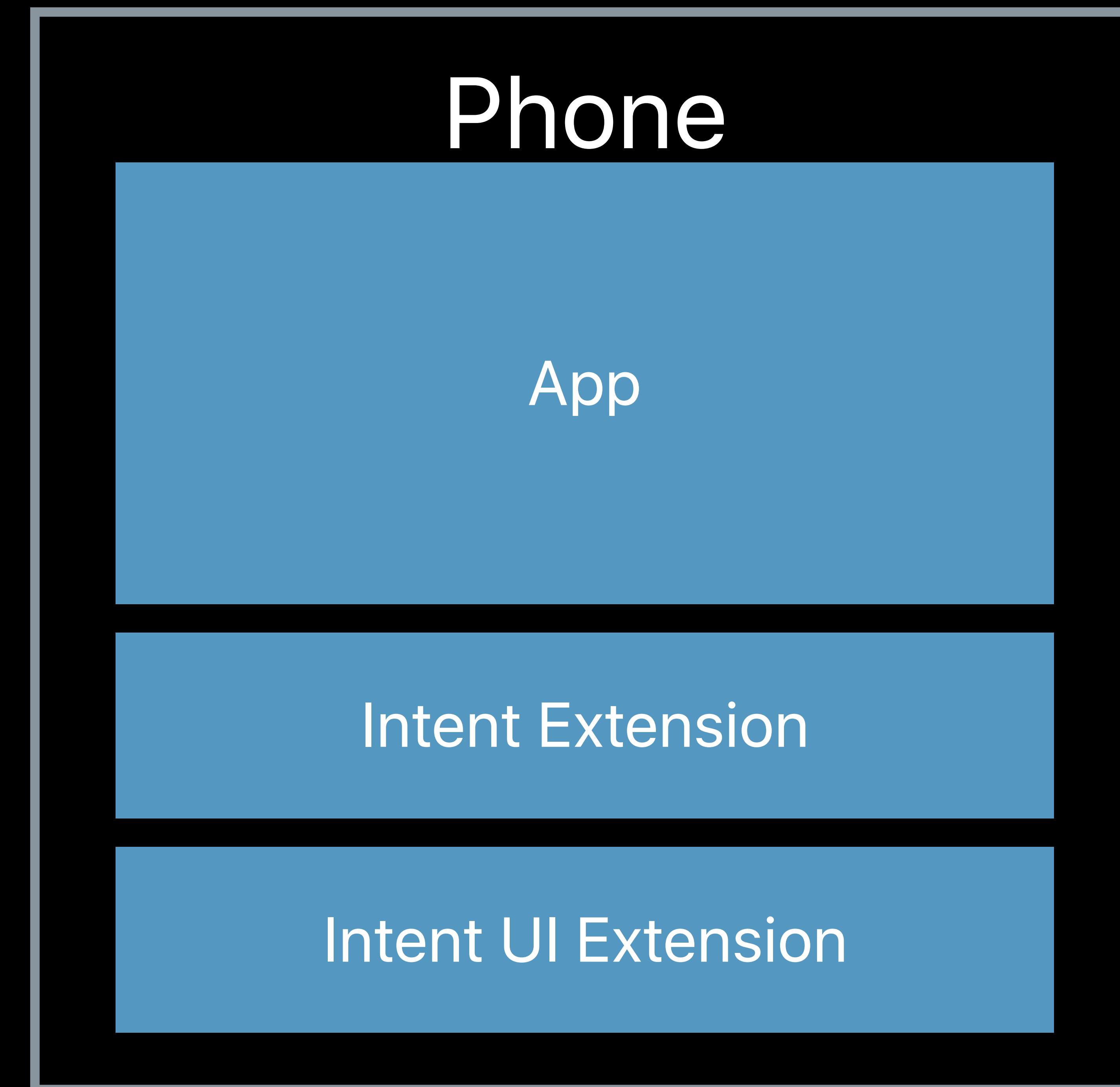
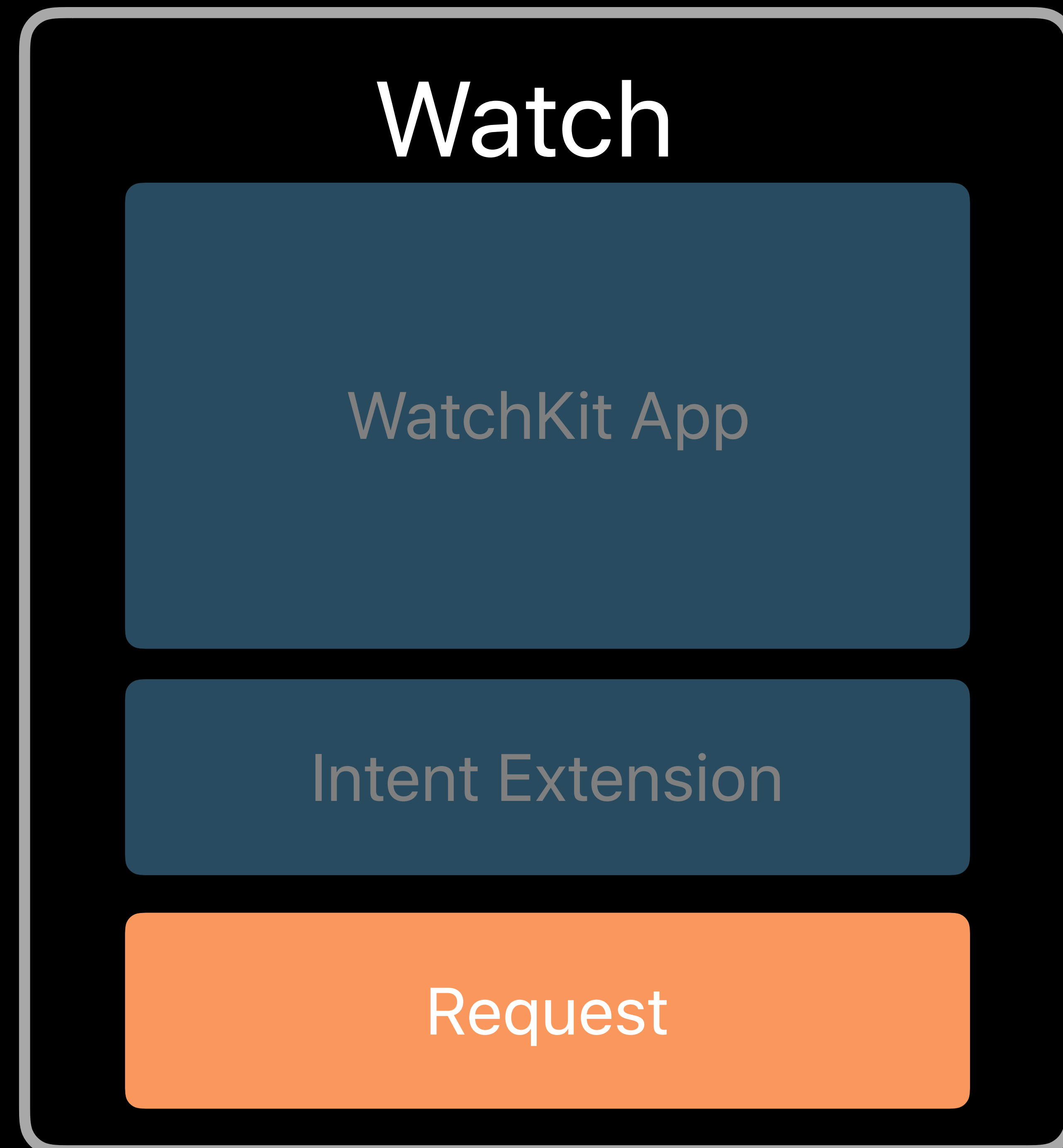
Running Shortcuts on Watch

Not supported by watch app



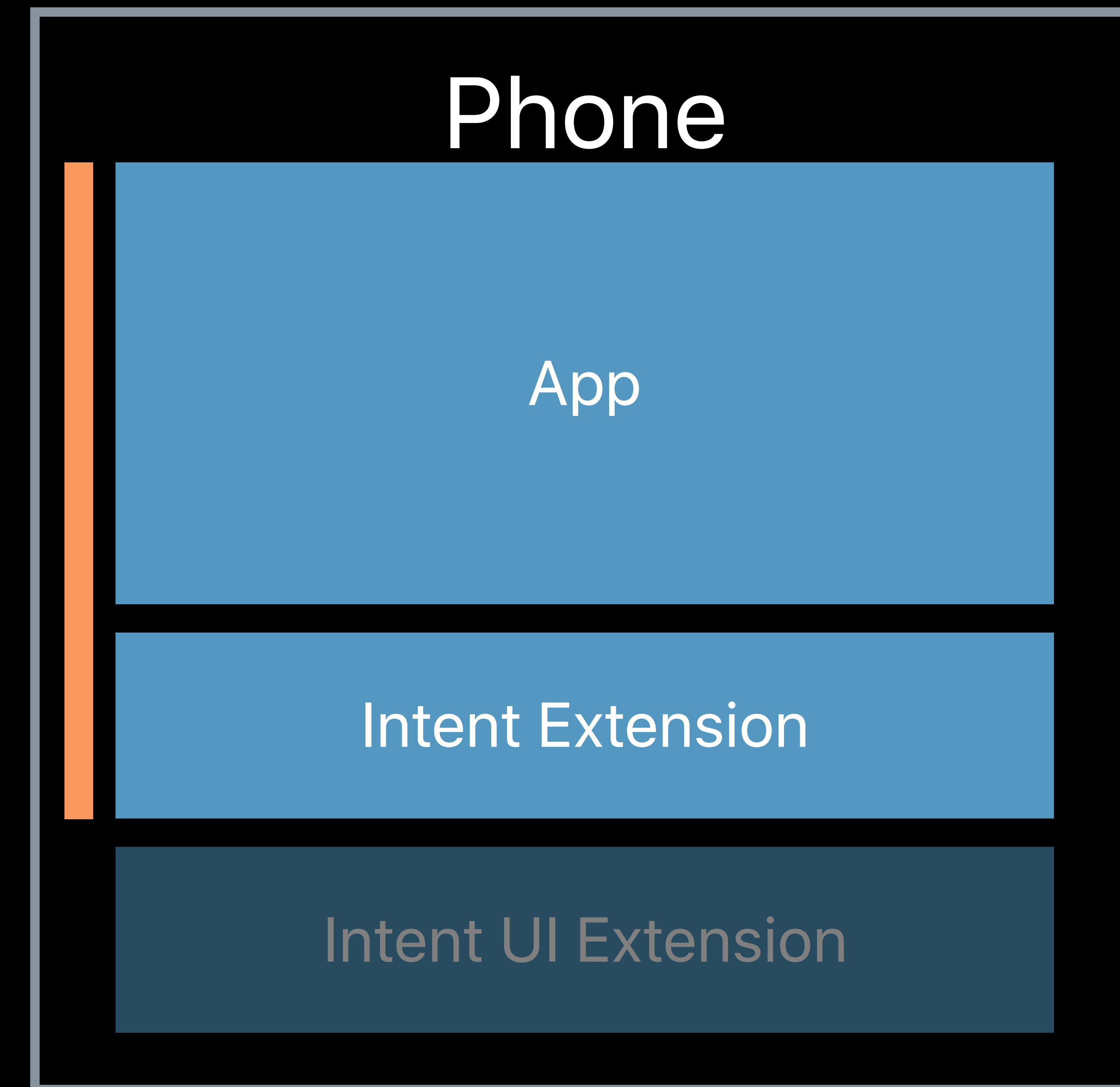
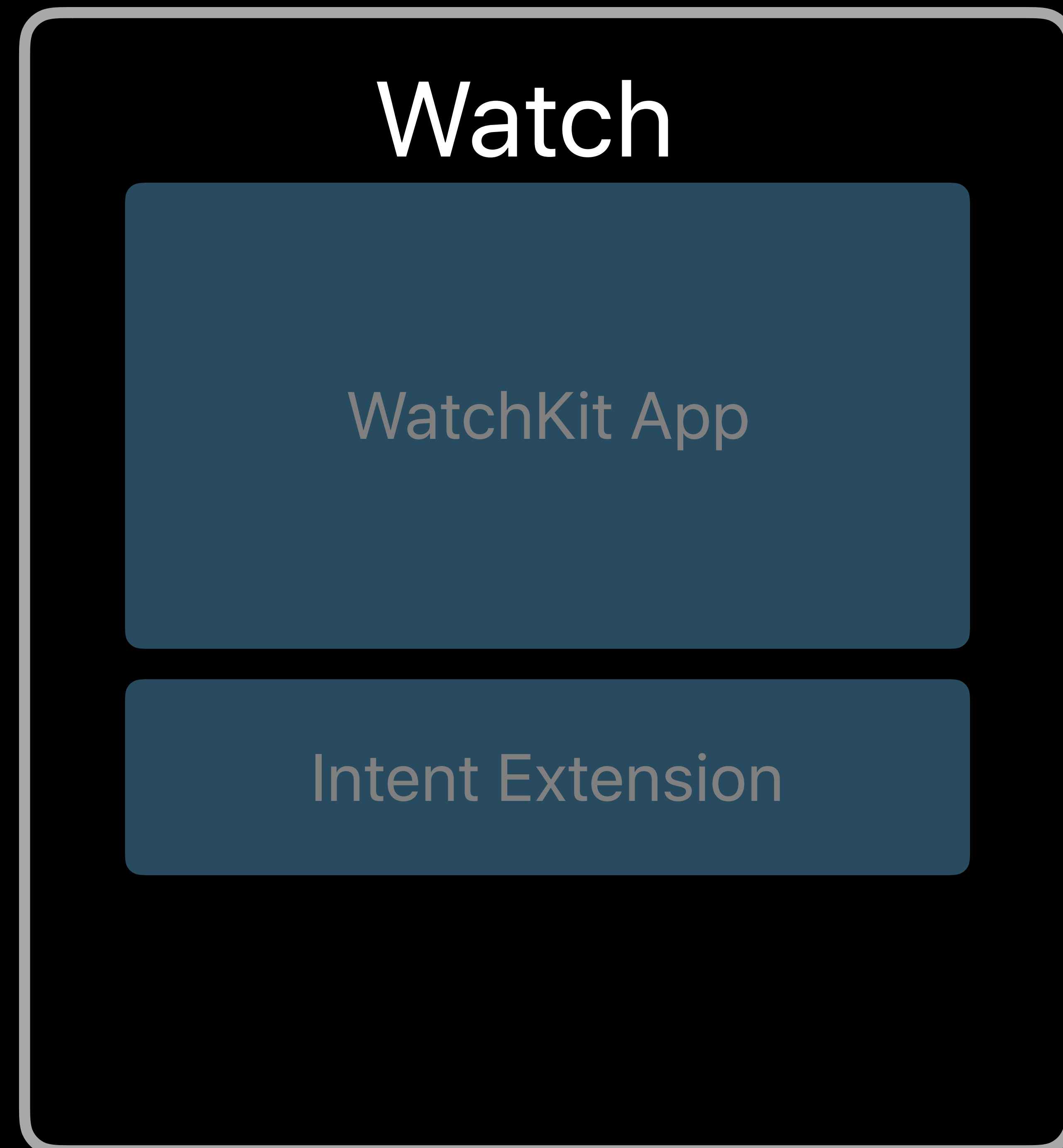
Running Shortcuts on Watch

Not supported by watch app



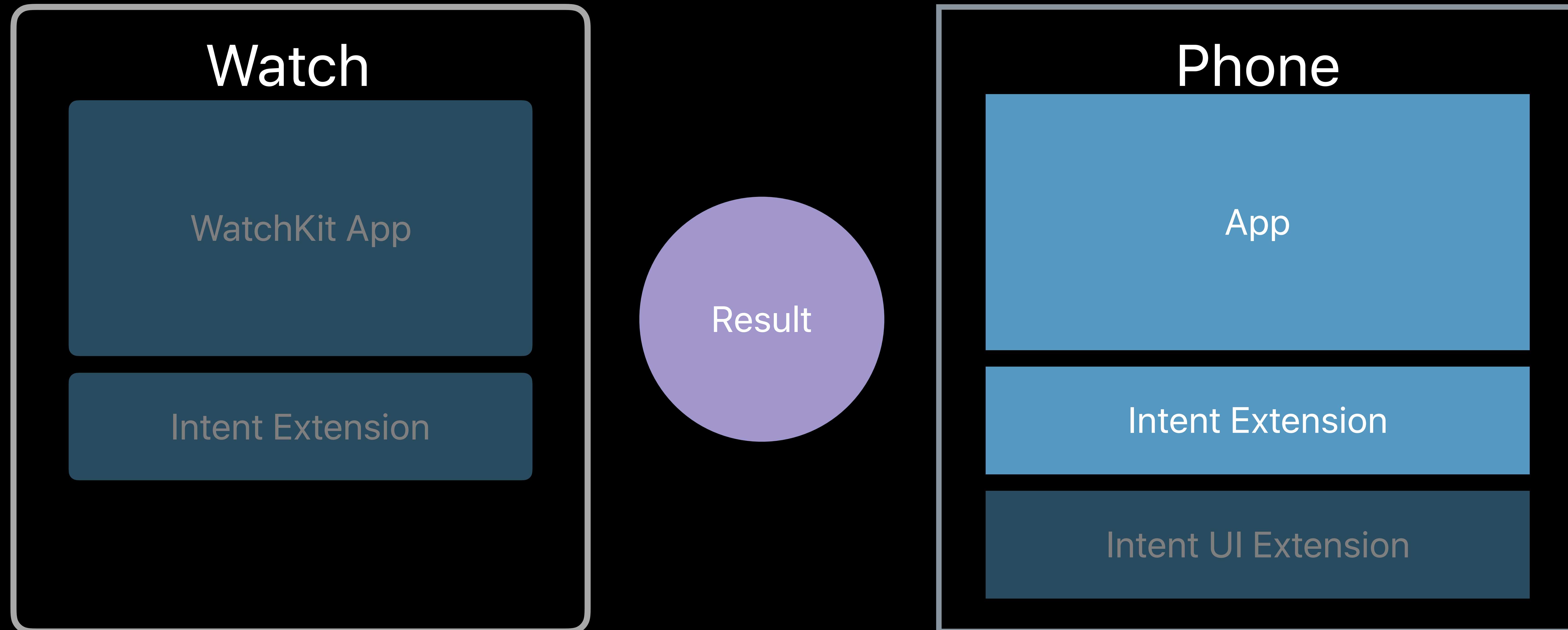
Running Shortcuts on Watch

Not supported by watch app



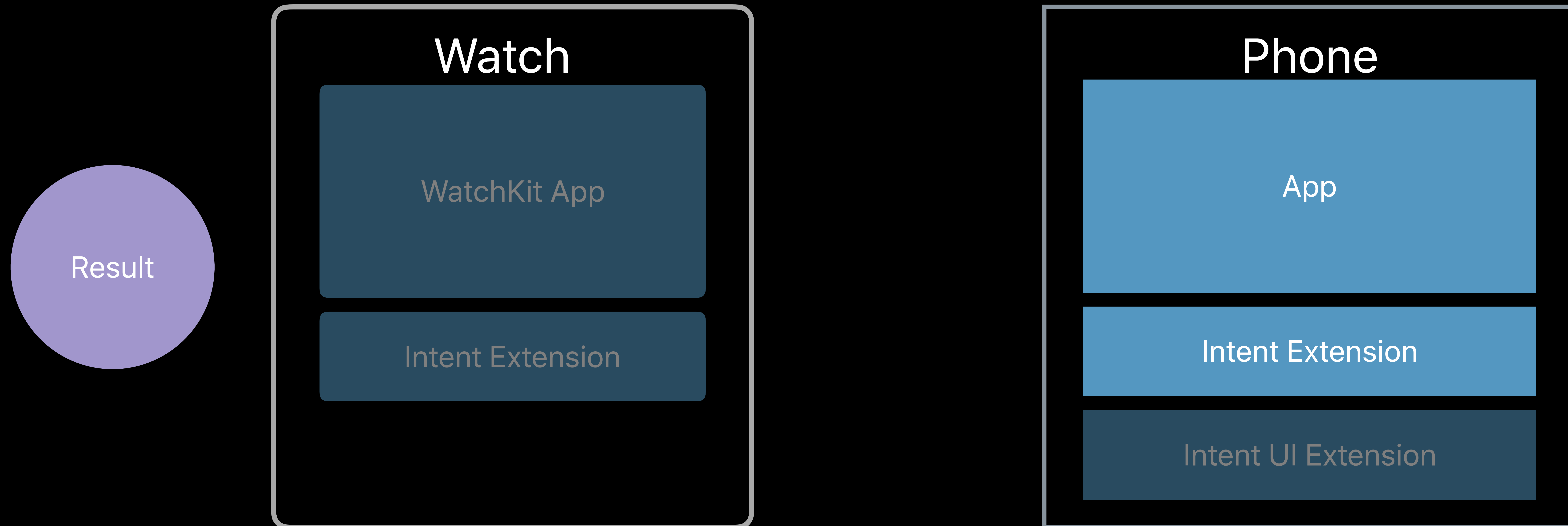
Running Shortcuts on Watch

Not supported by watch app



Running Shortcuts on Watch

Not supported by watch app



Relevant Shortcut



Relevant Shortcut

Displayed when most relevant



Relevant Shortcut

Displayed when most relevant

Customizable platter UI



Relevant Shortcut

Displayed when most relevant

Customizable platter UI

Tap activates shortcut



Relevant Shortcut

Displayed when most relevant

Customizable platter UI

Tap activates shortcut

- Launch for user activity



Relevant Shortcut

Displayed when most relevant

Customizable platter UI

Tap activates shortcut

- Launch for user activity



Relevant Shortcut

Displayed when most relevant

Customizable platter UI

Tap activates shortcut

- Launch for user activity



Relevant Shortcut

Displayed when most relevant

Customizable platter UI

Tap activates shortcut

- Launch for user activity
- Confirmation for intents



Relevant Shortcut

Displayed when most relevant

Customizable platter UI

Tap activates shortcut

- Launch for user activity
- Confirmation for intents



Relevant Shortcut

Displayed when most relevant

Customizable platter UI

Tap activates shortcut

- Launch for user activity
- Confirmation for intents




```
// Relevant Shortcut

open class INRelevantShortcut : NSObject, NSCopying, NSSecureCoding {
    @NSCopying open var shortcut: INShortcut { get }
    @NSCopying open var relevanceProviders: [INRelevanceProvider]
    @NSCopying open var watchTemplate: INDefaultCardTemplate?

    public init(shortcut: INShortcut)
}
}
```

```
// Relevant Shortcut
```

```
open class INRelevantShortcut : NSObject, NSCopying, NSSecureCoding {
```

```
    @NSCopying open var shortcut: INShortcut { get }
```

```
    @NSCopying open var relevanceProviders: [INRelevanceProvider]
```

```
    @NSCopying open var watchTemplate: INDefaultCardTemplate?
```

```
    public init(shortcut: INShortcut)
```

```
}
```

```
// Relevant Shortcut
```

```
open class INRelevantShortcut : NSObject, NSCopying, NSSecureCoding {
```

```
    @NSCopying open var shortcut: INShortcut { get }
```

```
    @NSCopying open var relevanceProviders: [INRelevanceProvider]
```

```
    @NSCopying open var watchTemplate: INDefaultCardTemplate?
```

```
    public init(shortcut: INShortcut)
```

```
}
```

```
// Relevant Shortcut
```

```
open class INRelevantShortcut : NSObject, NSCopying, NSSecureCoding {  
    @NSCopying open var shortcut: INShortcut { get }  
    @NSCopying open var relevanceProviders: [INRelevanceProvider]  
    @NSCopying open var watchTemplate: INDefaultCardTemplate?  
  
    public init(shortcut: INShortcut)  
}
```



```
// Relevant Shortcut Store
```

```
open class INRelevantShortcutStore : NSObject {  
    open class var `default`: INRelevantShortcutStore { get }  
    open func setRelevantShortcuts(_ shortcuts: [INRelevantShortcut],  
                                   completionHandler: ((Error?) -> Void)? = nil)  
}
```

```
// Relevant Shortcut Store
```

```
open class INRelevantShortcutStore : NSObject {
```

```
    open class var `default`: INRelevantShortcutStore { get }
```

```
    open func setRelevantShortcuts(_ shortcuts: [INRelevantShortcut],  
                                   completionHandler: ((Error?) -> Void)? = nil)
```

```
}
```

```
// Relevant Shortcut Store
```

```
open class INRelevantShortcutStore : NSObject {  
    open class var `default`: INRelevantShortcutStore { get }  
    open func setRelevantShortcuts(_ shortcuts: [INRelevantShortcut],  
        completionHandler: ((Error?) -> Void)? = nil)  
}
```

Relevant Shortcut Appearance

Platter

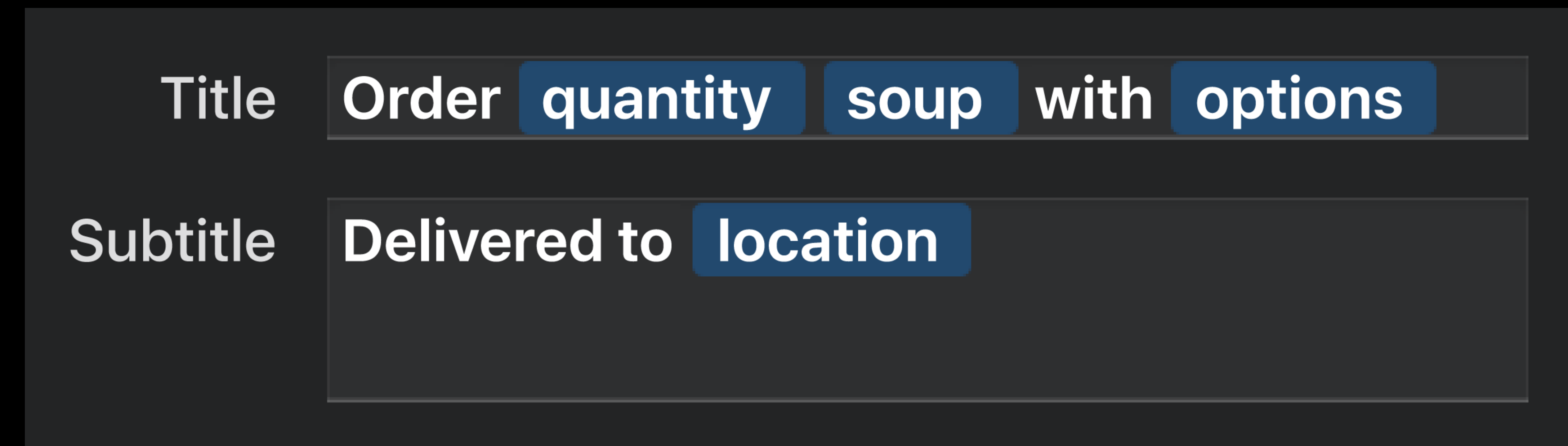


Relevant Shortcut Appearance

Platter



Custom Intent

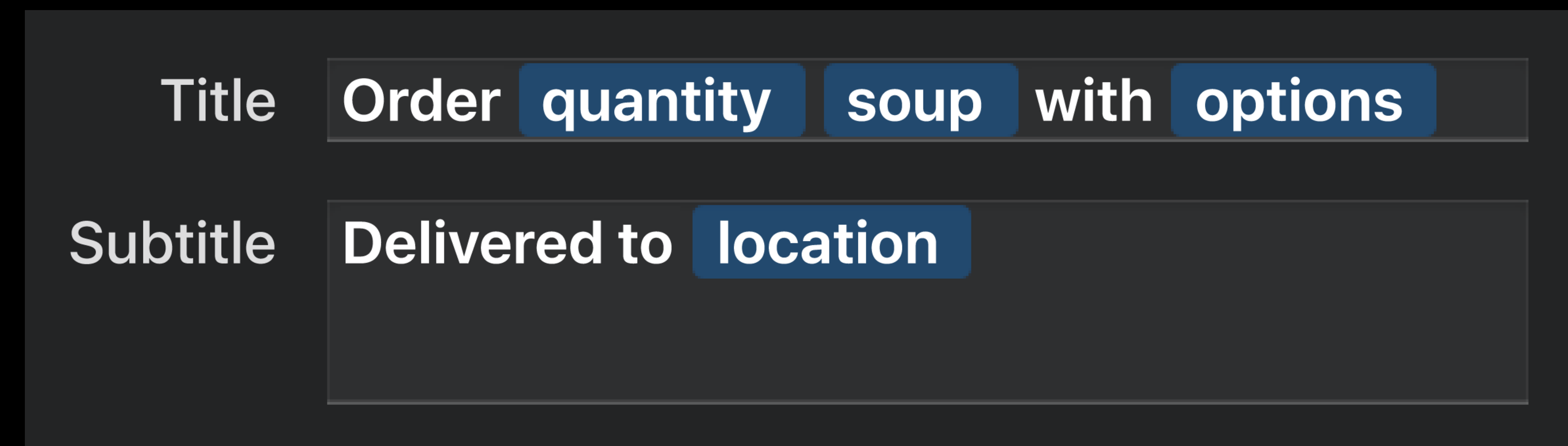


Relevant Shortcut Appearance

Platter



Custom Intent



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

```
// User Activity Default UI

let userActivity = NSUserActivity(activityType: "com.myapp.myactivity")

userActivity.title = "Title"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)

attributes.contentDescription = "Subtitle"

let image = UIImage(named: "CustomImage")
attributes.thumbnailData = UIImagePNGRepresentation(image)

userActivity.contentAttributeSet = attributes
```



```
// User Activity Default UI
```

```
let userActivity = NSUserActivity(activityType: "com.myapp.myactivity")
```

```
userActivity.title = "Title"
```

```
let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
```

```
attributes.contentDescription = "Subtitle"
```

```
let image = UIImage(named: "CustomImage")
```

```
attributes.thumbnailData = UIImagePNGRepresentation(image)
```

```
userActivity.contentAttributeSet = attributes
```



```
// User Activity Default UI
```

```
let userActivity = NSUserActivity(activityType: "com.myapp.myactivity")
```

```
userActivity.title = "Title"
```

```
let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
```

```
attributes.contentDescription = "Subtitle"
```

```
let image = UIImage(named: "CustomImage")
```

```
attributes.thumbnailData = UIImagePNGRepresentation(image)
```

```
userActivity.contentAttributeSet = attributes
```

```
// User Activity Default UI
```

```
let userActivity = NSUserActivity(activityType: "com.myapp.myactivity")
```

```
userActivity.title = "Title"
```

```
let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
```

```
attributes.contentDescription = "Subtitle"
```

```
let image = UIImage(named: "CustomImage")
```

```
attributes.thumbnailData = UIImagePNGRepresentation(image)
```

```
userActivity.contentAttributeSet = attributes
```

```
// User Activity Default UI

let userActivity = NSUserActivity(activityType: "com.myapp.myactivity")

userActivity.title = "Title"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)

attributes.contentDescription = "Subtitle"

let image = UIImage(named: "CustomImage")
attributes.thumbnailData = UIImagePNGRepresentation(image)

userActivity.contentAttributeSet = attributes
```

```
// User Activity Default UI

let userActivity = NSUserActivity(activityType: "com.myapp.myactivity")

userActivity.title = "Title"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)

attributes.contentDescription = "Subtitle"

let image = UIImage(named: "CustomImage")
attributes.thumbnailData = UIImagePNGRepresentation(image)

userActivity.contentAttributeSet = attributes
```



```
// User Activity Default UI

let userActivity = NSUserActivity(activityType: "com.myapp.myactivity")

userActivity.title = "Title"


let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)

attributes.contentDescription = "Subtitle"

let image = UIImage(named: "CustomImage")
attributes.thumbnailData = UIImagePNGRepresentation(image)

userActivity.contentAttributeSet = attributes
```


Default Card Template

■ **App Name**
 Title
Subtitle

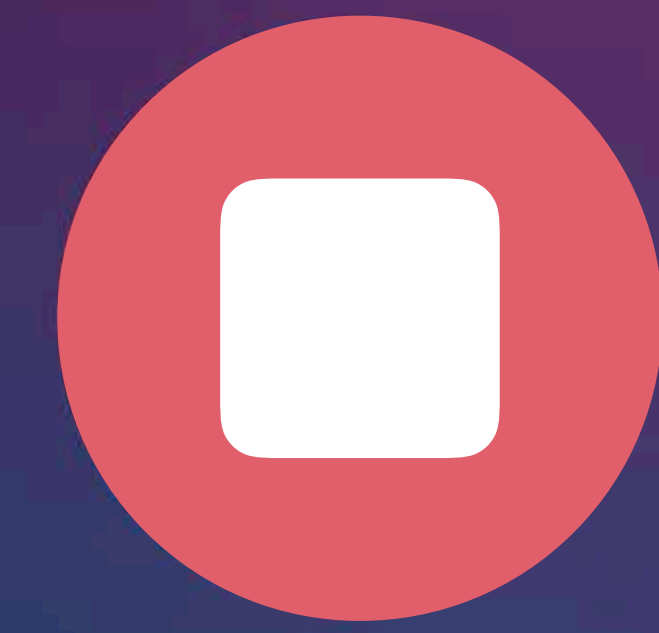
■ **App Name**
Title
Subtitle

■ **App Name**
 Title string
that wraps

■ **App Name**
Title text that will
wrap two lines

Relevant Shortcut Appearance

Intent Confirmation



App Name

Intent title

Intent subtitle

providing context

Action Verb

Dismiss

Relevant Shortcut Appearance

Intent Confirmation



App Name

Intent title

Intent subtitle
providing context

Action Verb

Dismiss

▼ Custom Intent	
Category	Generic <ul style="list-style-type: none">✓ DoRunGo
Title	
Description	Information <ul style="list-style-type: none">ViewOpen
Confirmation	
	Order <ul style="list-style-type: none">OrderBookBuy
	Start <ul style="list-style-type: none">StartNavigate
	Share <ul style="list-style-type: none">SharePostSend
	Create <ul style="list-style-type: none">CreateAdd
	Search <ul style="list-style-type: none">Search

Relevant Shortcut Appearance

Intent Confirmation



App Name

Intent title

Intent subtitle

providing context

Action Verb

Dismiss

Interface Builder Document

Opens in Latest Xcode (9.0)

Builds for Deployment Target (5.0)

Global Tint Custom

Page Direction Horizontal

Short Strings

Short Strings

Limited space

Short Strings

Limited space

`NSStringVariableWidthRuleType` in `stringsdict`

Key	Type	Value
▼ Strings Dictionary	Dictionary	(1 item)
▼ Start a <code>\${timerName}</code> timer	Dictionary	(1 item)
▼ <code>NSStringVariableWidthRuleType</code>	Dictionary	(2 items)
20	String	Start <code>\${timerName}</code>
24	String	Start a <code>\${timerName}</code> timer

Short Strings

Limited space

`NSStringVariableWidthRuleType` in `stringsdict`

Options for context

- 38mm: 20
- 42mm: 24

Key	Type	Value
▼ Strings Dictionary	Dictionary	(1 item)
▼ Start a <code>\${timerName}</code> timer	Dictionary	(1 item)
▼ <code>NSStringVariableWidthRuleType</code>	Dictionary	(2 items)
20	String	Start <code>\${timerName}</code>
24	String	Start a <code>\${timerName}</code> timer

Short Strings

Limited space

`NSStringVariableWidthRuleType` in stringsdict

Options for context

- 38mm: 20
- 42mm: 24

Key	Type	Value
▼ Strings Dictionary	Dictionary	(1 item)
▼ Start a <code>\${timerName}</code> timer	Dictionary	(1 item)
▼ NSStringVariableWidthRuleType	Dictionary	(2 items)
20	String	Start <code>\${timerName}</code>
24	String	Start a <code>\${timerName}</code> timer

Relevance Providers

Relevance Provider

Relevance Provider

Describe content relevance

Relevance Provider

Describe content relevance

Evaluated against time, location, and routine

Relevance Provider

Describe content relevance

Evaluated against time, location, and routine

Provide array to INRelevantShortcut

- Intersection of supplied relevance

```
//Relevance Providers
```

```
open class INDateRelevanceProvider : INRelevanceProvider {  
    open var startDate: Date { get }  
    open var endDate: Date? { get }  
    public init(start startDate: Date, end endDate: Date?)  
}
```

```
let startEvent = /* event start time */
```

```
let endEvent = /* event end time */
```

```
let relevanceProvider = INDateRelevanceProvider(start: startEvent, end: endEvent)
```



```
//Relevance Providers
```

```
open class INDateRelevanceProvider : INRelevanceProvider {  
    open var startDate: Date { get }  
    open var endDate: Date? { get }  
    public init(start startDate: Date, end endDate: Date?)  
}
```

```
let startEvent = /* event start time */
```

```
let endEvent = /* event end time */
```

```
let relevanceProvider = INDateRelevanceProvider(start: startEvent, end: endEvent)
```

```
//Relevance Providers
```

```
open class INDateRelevanceProvider : INRelevanceProvider {  
    open var startDate: Date { get }  
    open var endDate: Date? { get }  
    public init(start startDate: Date, end endDate: Date?)  
}
```

```
let startEvent = /* event start time */  
let endEvent = /* event end time */  
let relevanceProvider = INDateRelevanceProvider(start: startEvent, end: endEvent)
```

```
//Relevance Providers
```

```
open class INLocationRelevanceProvider : INRelevanceProvider {  
    @NSCopying open var region: CLRegion { get }  
    public init(region: CLRegion)  
}
```

```
let center = CLLocationCoordinate2D(latitude: 37.334728, longitude: -122.008883)  
let region = CLCircularRegion(center: center, radius: 2.0, identifier: "Apple Park")  
let relevanceProvider = INLocationRelevanceProvider(region: region)
```

```
//Relevance Providers
```

```
open class INLocationRelevanceProvider : INRelevanceProvider {  
    @NSCopying open var region: CLRegion { get }  
    public init(region: CLRegion)  
}
```

```
let center = CLLocationCoordinate2D(latitude: 37.334728, longitude: -122.008883)  
let region = CLCircularRegion(center: center, radius: 2.0, identifier: "Apple Park")  
let relevanceProvider = INLocationRelevanceProvider(region: region)
```



```
//Relevance Providers
```

```
open class INLocationRelevanceProvider : INRelevanceProvider {  
    @NSCopying open var region: CLRegion { get }  
    public init(region: CLRegion)  
}
```

```
let center = CLLocationCoordinate2D(latitude: 37.334728, longitude: -122.008883)  
let region = CLCircularRegion(center: center, radius: 2.0, identifier: "Apple Park")  
let relevanceProvider = INLocationRelevanceProvider(region: region)
```

```
//Relevance Providers
```

```
open class INDailyRoutineRelevanceProvider : INRelevanceProvider {  
    public enum Situation : Int {  
        case morning  
        case evening  
        case home  
        case work  
        case school  
        case gym  
    }  
    open var situation: Situation { get }  
    public init(situation: Situation)  
}
```

```
let relevanceProvider = INDailyRoutineRelevanceProvider(situation: .gym)
```

```
//Relevance Providers
```

```
open class INDailyRoutineRelevanceProvider : INRelevanceProvider {  
    public enum Situation : Int {  
        case morning  
        case evening  
        case home  
        case work  
        case school  
        case gym  
    }  
    open var situation: Situation { get }  
    public init(situation: Situation)  
}
```

```
let relevanceProvider = INDailyRoutineRelevanceProvider(situation: .gym)
```

```
//Relevance Providers
```

```
open class INDailyRoutineRelevanceProvider : INRelevanceProvider {  
    public enum Situation : Int {  
        case morning  
        case evening  
        case home  
        case work  
        case school  
        case gym  
    }  
    open var situation: Situation { get }  
    public init(situation: Situation)  
}
```

```
let relevanceProvider = INDailyRoutineRelevanceProvider(situation: .gym)
```



```
// Setting INRelevantShortcuts
let userActivity = NSUserActivity(activityType: "com.myapp.LogMeal")
userActivity.userInfo["meal"] = "Dinner"
let shortcut = INShortcut(userActivity: userActivity)
let relevantShortcut = INRelevantShortcut(shortcut: shortcut!)

let cardTemplate = INDefaultCardTemplate(title: "Log Dinner")
cardTemplate.subtitle = "Beat the veggie challenge!"
cardTemplate.image = INImage(named: "MenacingBroccoli")
relevantShortcut.watchTemplate = cardTemplate

let relevanceProvider = INDailyRoutineRelevanceProvider(situation: .evening)
relevantShortcut.relevanceProviders = [relevanceProvider]

INRelevantShortcutStore.default.setRelevantShortcuts([relevantShortcut]) { error in
    if let error = error { /* */ }
}
```

```
// Setting INRelevantShortcuts
let userActivity = NSUserActivity(activityType: "com.myapp.LogMeal")
userActivity.userInfo["meal"] = "Dinner"
let shortcut = INShortcut(userActivity: userActivity)
let relevantShortcut = INRelevantShortcut(shortcut: shortcut!)

let cardTemplate = INDefaultCardTemplate(title: "Log Dinner")
cardTemplate.subtitle = "Beat the veggie challenge!"
cardTemplate.image = INImage(named: "MenacingBroccoli")
relevantShortcut.watchTemplate = cardTemplate

let relevanceProvider = INDailyRoutineRelevanceProvider(situation: .evening)
relevantShortcut.relevanceProviders = [relevanceProvider]

INRelevantShortcutStore.default.setRelevantShortcuts([relevantShortcut]) { error in
    if let error = error { /* */ }
}
```

```
// Setting INRelevantShortcuts
let userActivity = NSUserActivity(activityType: "com.myapp.LogMeal")
userActivity.userInfo["meal"] = "Dinner"
let shortcut = INShortcut(userActivity: userActivity)
let relevantShortcut = INRelevantShortcut(shortcut: shortcut!)

let cardTemplate = INDefaultCardTemplate(title: "Log Dinner")
cardTemplate.subtitle = "Beat the veggie challenge!"
cardTemplate.image = INImage(named: "MenacingBroccoli")
relevantShortcut.watchTemplate = cardTemplate

let relevanceProvider = INDailyRoutineRelevanceProvider(situation: .evening)
relevantShortcut.relevanceProviders = [relevanceProvider]

INRelevantShortcutStore.default.setRelevantShortcuts([relevantShortcut]) { error in
    if let error = error { /* */ }
}
```



```
// Setting INRelevantShortcuts
let userActivity = NSUserActivity(activityType: "com.myapp.LogMeal")
userActivity.userInfo["meal"] = "Dinner"
let shortcut = INShortcut(userActivity: userActivity)
let relevantShortcut = INRelevantShortcut(shortcut: shortcut!)
```

```
let cardTemplate = INDefaultCardTemplate(title: "Log Dinner")
cardTemplate.subtitle = "Beat the veggie challenge!"
cardTemplate.image = INImage(named: "MenacingBroccoli")
relevantShortcut.watchTemplate = cardTemplate
```

```
let relevanceProvider = INDailyRoutineRelevanceProvider(situation: .evening)
relevantShortcut.relevanceProviders = [relevanceProvider]
```

```
INRelevantShortcutStore.default.setRelevantShortcuts([relevantShortcut]) { error in
    if let error = error { /* */ }
}
```



```
// Setting INRelevantShortcuts
let userActivity = NSUserActivity(activityType: "com.myapp.LogMeal")
userActivity.userInfo["meal"] = "Dinner"
let shortcut = INShortcut(userActivity: userActivity)
let relevantShortcut = INRelevantShortcut(shortcut: shortcut!)

let cardTemplate = INDefaultCardTemplate(title: "Log Dinner")
cardTemplate.subtitle = "Beat the veggie challenge!"
cardTemplate.image = INImage(named: "MenacingBroccoli")
relevantShortcut.watchTemplate = cardTemplate

let relevanceProvider = INDailyRoutineRelevanceProvider(situation: .evening)
relevantShortcut.relevanceProviders = [relevanceProvider]

INRelevantShortcutStore.default.setRelevantShortcuts([relevantShortcut]) { error in
    if let error = error { /* */ }
}
```

```
// Setting INRelevantShortcuts
let userActivity = NSUserActivity(activityType: "com.myapp.LogMeal")
userActivity.userInfo["meal"] = "Dinner"
let shortcut = INShortcut(userActivity: userActivity)
let relevantShortcut = INRelevantShortcut(shortcut: shortcut!)

let cardTemplate = INDefaultCardTemplate(title: "Log Dinner")
cardTemplate.subtitle = "Beat the veggie challenge!"
cardTemplate.image = INImage(named: "MenacingBroccoli")
relevantShortcut.watchTemplate = cardTemplate

let relevanceProvider = INDailyRoutineRelevanceProvider(situation: .evening)
relevantShortcut.relevanceProviders = [relevanceProvider]
```

```
INRelevantShortcutStore.default.setRelevantShortcuts([relevantShortcut]) { error in
    if let error = error { /* */ }
}
```

```
// Setting INRelevantShortcuts

let intent = LogFavoriteIntent()
intent.favorite = "Cookies"
intent.setImage(INImage(named: "Cookies"), forParameterNamed: "favorite")

let shortcut = INShortcut(intent: intent)
let relevantShortcut = INRelevantShortcut(shortcut: shortcut!)

INRelevantShortcutStore.default.setRelevantShortcuts([relevantShortcut]) { error in
    if let error = error { /* */ }
}
```

```
// Setting INRelevantShortcuts
```

```
let intent = LogFavoriteIntent()
intent.favorite = "Cookies"
intent.setImage(INImage(named: "Cookies"), forParameterNamed: "favorite")
```

```
let shortcut = INShortcut(intent: intent)
```

```
let relevantShortcut = INRelevantShortcut(shortcut: shortcut!)
```

```
INRelevantShortcutStore.default.setRelevantShortcuts([relevantShortcut]) { error in
    if let error = error { /* */ }
}
```



```
// Setting INRelevantShortcuts
```

```
let intent = LogFavoriteIntent()
```

```
intent.favorite = "Cookies"
```

```
intent.setImage(INImage(named: "Cookies"), forParameterNamed: "favorite")
```

```
let shortcut = INShortcut(intent: intent)
```

```
let relevantShortcut = INRelevantShortcut(shortcut: shortcut!)
```

```
INRelevantShortcutStore.default.setRelevantShortcuts([relevantShortcut]) { error in
```

```
    if let error = error { /* */ }
```

```
}
```

```
// Setting INRelevantShortcuts
```

```
let intent = LogFavoriteIntent()
```

```
intent.favorite = "Cookies"
```

```
intent.setImage(INImage(named: "Cookies"), forParameterNamed: "favorite")
```

```
let shortcut = INShortcut(intent: intent)
```

```
let relevantShortcut = INRelevantShortcut(shortcut: shortcut!)
```

```
INRelevantShortcutStore.default.setRelevantShortcuts([relevantShortcut]) { error in  
    if let error = error { /* */ }  
}
```



NEW

```
// Handle restoring from NSUserActivity

class ExtensionDelegate: NSObject, WKExtensionDelegate {
    func handle(_ userActivity: NSUserActivity) {
        if userActivity.activityType == "com.myapp.LogMeal" {
            WKExtension.shared().rootInterfaceController?.popToRootController()
            WKExtension.shared().rootInterfaceController?.pushController(withName:
                "LogMealInterfaceController", context: userActivity.userInfo!["meal"])
        }
        else if userActivity.activityType == "LogFavoriteIntent" {
            let interaction = userActivity!.interaction
            let intent = interaction.intent
            // restore
        }
    }
}
```

NEW

```
// Handle restoring from NSUserActivity

class ExtensionDelegate: NSObject, WKExtensionDelegate {
    func handle(_ userActivity: NSUserActivity) {
        if userActivity.activityType == "com.myapp.LogMeal" {
            WKExtension.shared().rootInterfaceController?.popToRootController()
            WKExtension.shared().rootInterfaceController?.pushController(withName:
                "LogMealInterfaceController", context: userActivity.userInfo!["meal"])
        }
        else if userActivity.activityType == "LogFavoriteIntent" {
            let interaction = userActivity!.interaction
            let intent = interaction.intent
            // restore
        }
    }
}
```


NEW

```
// Handle restoring from NSUserActivity

class ExtensionDelegate: NSObject, WKExtensionDelegate {
    func handle(_ userActivity: NSUserActivity) {
        if userActivity.activityType == "com.myapp.LogMeal" {
            WKExtension.shared().rootInterfaceController?.popToRootController()
            WKExtension.shared().rootInterfaceController?.pushController(withName:
                "LogMealInterfaceController", context: userActivity.userInfo!["meal"])
        }
        else if userActivity.activityType == "LogFavoriteIntent" {
            let interaction = userActivity!.interaction
            let intent = interaction.intent
            // restore
        }
    }
}
```

NEW

```
// Handle restoring from NSUserActivity

class ExtensionDelegate: NSObject, WKExtensionDelegate {
    func handle(_ userActivity: NSUserActivity) {
        if userActivity.activityType == "com.myapp.LogMeal" {
            WKExtension.shared().rootInterfaceController?.popToRootController()
            WKExtension.shared().rootInterfaceController?.pushController(withName:
                "LogMealInterfaceController", context: userActivity.userInfo!["meal"])
        }
        else if userActivity.activityType == "LogFavoriteIntent" {
            let interaction = userActivity!.interaction
            let intent = interaction.intent
            // restore
        }
    }
}
```

NEW

```
// Handle restoring from NSUserActivity

class ExtensionDelegate: NSObject, WKExtensionDelegate {
    func handle(_ userActivity: NSUserActivity) {
        if userActivity.activityType == "com.myapp.LogMeal" {
            WKExtension.shared().rootInterfaceController?.popToRootController()
            WKExtension.shared().rootInterfaceController?.pushController(withName:
                "LogMealInterfaceController", context: userActivity.userInfo!["meal"])
        }
        else if userActivity.activityType == "LogFavoriteIntent" {
            let interaction = userActivity!.interaction
            let intent = interaction.intent
            // restore
        }
    }
}
```




NEW

```
// Handle restoring from NSUserActivity

class ExtensionDelegate: NSObject, WKExtensionDelegate {
    func handle(_ userActivity: NSUserActivity) {
        if userActivity.activityType == "com.myapp.LogMeal" {
            WKExtension.shared().rootInterfaceController?.popToRootController()
            WKExtension.shared().rootInterfaceController?.pushController(withName:
                "LogMealInterfaceController", context: userActivity.userInfo!["meal"])
        }
        else if userActivity.activityType == "LogFavoriteIntent" {
            let interaction = userActivity!.interaction
            let intent = interaction.intent
            // restore
        }
    }
}
```


Refresh Background Task

Provide relevant shortcuts

Refresh Background Task

Provide relevant shortcuts

Apps in platters not running

Refresh Background Task

Provide relevant shortcuts

Apps in platters not running

```
WKRelevantShortcutRefreshBackgroundTask
```

Refresh Background Task

Provide relevant shortcuts

Apps in platters not running

```
WKRelevantShortcutRefreshBackgroundTask
```

- Scheduled by user engagement

Refresh Background Task

Provide relevant shortcuts

Apps in platters not running

```
WKRelevantShortcutRefreshBackgroundTask
```

- Scheduled by user engagement
- Refresh data and relevant shortcuts

Refresh Background Task

Update after running intent

Refresh Background Task

Update after running intent

Intents extension separate process

Refresh Background Task

Update after running intent

Intents extension separate process

`WKIntentDidRunRefreshBackgroundTask` after intent execution

Refresh Background Task

Update after running intent

Intents extension separate process

`WKIntentDidRunRefreshBackgroundTask` after intent execution

Update appropriate UI

Refresh Background Task

Update after running intent

Intents extension separate process

`WKIntentDidRunRefreshBackgroundTask` after intent execution

Update appropriate UI

- Snapshot

Refresh Background Task

Update after running intent

Intents extension separate process

`WKIntentDidRunRefreshBackgroundTask` after intent execution

Update appropriate UI

- Snapshot
- Complication

Relevant Shortcuts from iOS

iOS Relevant Shortcuts on Siri Watch Face

iOS Relevant Shortcuts on Siri Watch Face

iOS apps use same API

iOS Relevant Shortcuts on Siri Watch Face

iOS apps use same API

Periodic sync

iOS Relevant Shortcuts on Siri Watch Face

iOS apps use same API

Periodic sync

Same UI options with iOS app icon

iOS Relevant Shortcuts on Siri Watch Face

iOS Relevant Shortcuts on Siri Watch Face

WatchKit app support —> execute on watch

iOS Relevant Shortcuts on Siri Watch Face

WatchKit app support —> execute on watch

- `NSUserActivityTypes` in Info.plist

iOS Relevant Shortcuts on Siri Watch Face

WatchKit app support —> execute on watch

- `NSUserActivityTypes` in Info.plist
- `IntentsSupported` in Intent Info.plist

iOS Relevant Shortcuts on Siri Watch Face

WatchKit app support —> execute on watch

- `NSUserActivityTypes` in Info.plist
- `IntentsSupported` in Intent Info.plist

No WatchKit app support —> execute on phone over Internet

iOS Relevant Shortcuts on Siri Watch Face

iOS Relevant Shortcuts on Siri Watch Face

Surfacing/executing iOS only shortcuts requires

iOS Relevant Shortcuts on Siri Watch Face

Surfacing/executing iOS only shortcuts requires

- Intent based

iOS Relevant Shortcuts on Siri Watch Face

Surfacing/executing iOS only shortcuts requires

- Intent based
- Background execution

iOS Relevant Shortcuts on Siri Watch Face

Surfacing/executing iOS only shortcuts requires

- Intent based
- Background execution
- No protected data needed

iOS Relevant Shortcuts on Siri Watch Face

The screenshot shows the Xcode interface for configuring a custom intent named "OrderSoup". The interface is divided into several sections:

- CUSTOM INTENTS:** A sidebar on the left shows "OrderSoup" selected under "CUSTOM INTENTS".
- Custom Intent:** The main configuration area includes:
 - Category:** Do
 - Title:** Order Soup
 - Description:** Order some soup
 - Confirmation:** User confirmation required
- Parameters:** A table defining the parameters for the intent:

Parameter	Type	Array
soup	Custom	<input type="checkbox"/>
quantity	Integer	<input type="checkbox"/>
options	Custom	<input checked="" type="checkbox"/>
- Shortcut Types:** A list of parameter combinations for the shortcut:
 - soup
 - soup, options (highlighted)
 - soup, quantity, options
 - soup, quantity
- Title and Subtitle:** The title is "Order soup with options".
- Background:** Supports background execution

The right-hand pane shows additional settings for the intent:

- Intent:** Name: OrderSoup, Authentication: Not Restricted
- Custom Class:** Class Name: OrderSoupIntent
- Watch Shortcuts:** Primary Type: None

iOS Relevant Shortcuts on Siri Watch Face

The screenshot shows the Xcode interface for configuring a custom intent named "OrderSoup". The interface is divided into several sections:

- CUSTOM INTENTS:** A sidebar on the left shows "OrderSoup" selected under "CUSTOM INTENTS".
- Custom Intent:** The main configuration area includes:
 - Category: Do
 - Title: Order Soup
 - Description: Order some soup
 - Confirmation: User confirmation required
- Parameters:** A table defining the parameters for the intent:

Parameter	Type	Array
soup	Custom	<input type="checkbox"/>
quantity	Integer	<input type="checkbox"/>
options	Custom	<input checked="" type="checkbox"/>
- Shortcut Types:** A list of parameter combinations with their corresponding titles and subtitles:

Parameter Combination	Title	Subtitle
soup	Order soup	
soup, options	Order soup with options	
soup, quantity, options		
soup, quantity		
- Intent:** A panel on the right showing:
 - Name: OrderSoup
 - Authentication: Not Restricted
- Custom Class:** A panel showing Class Name: OrderSoupIntent
- Watch Shortcuts:** A panel showing Primary Type: None

A callout box at the bottom right of the "Shortcut Types" section contains the text: **Background** Supports background execution.

iOS Relevant Shortcuts on Siri Watch Face

The screenshot displays the Xcode interface for configuring a custom intent. The main window shows the 'OrderSoup' intent configuration with the following details:

- Category:** Do
- Title:** Order Soup
- Description:** Order some soup
- Confirmation:** User confirmation required

The **Parameters** section contains a table with the following entries:

Parameter	Type	Array
soup	Custom	<input type="checkbox"/>
quantity	Integer	<input type="checkbox"/>
options	Custom	<input checked="" type="checkbox"/>

The **Shortcut Types** section shows a list of parameter combinations with the following titles and subtitles:

Parameter Combination	Title	Subtitle
soup	Order soup with options	
soup, options		
soup, quantity, options		
soup, quantity		

An overlay window titled **Intent** is visible, showing the following configuration:

- Name:** OrderSoup
- Authentication:** Not Restricted
- Class Name:** OrderSoupIntent
- Watch Shortcuts Primary Type:** None

A callout box at the bottom indicates: **Background** Supports background execution.

iOS Relevant Shortcuts on Siri Watch Face

The screenshot shows the Xcode interface for configuring a custom intent named "OrderSoup". The interface is divided into several sections:

- Intent Configuration:**
 - Name:** OrderSoup
 - Authentication:** Not Restricted (checked), Restricted While Locked, Restricted While Locked or Protected Data Unavailable
 - Watch Shortcuts:** Primary Type: None
- Parameters:**

Parameter	Type	Array
soup	Custom	<input type="checkbox"/>
quantity	Integer	<input type="checkbox"/>
options	Custom	<input checked="" type="checkbox"/>
- Shortcut Types:**

Parameter Combination	Title	Subtitle
soup	Order soup with options	
soup, options		
soup, quantity, options		
soup, quantity		

Additional settings include:

- Category:** Do
- Title:** Order Soup
- Description:** Order some soup
- Confirmation:** User confirmation required
- Background:** Supports background execution

Predicting Shortcuts on watchOS

Josh Ford, watchOS Engineer

Predicting Shortcuts

Content is ordered by relevance

Up Next

More Relevant

Least Relevant

Predicting Shortcuts

Content is ordered by relevance

User interaction instructs ordering

Up Next

More Relevant

Least Relevant

Predicting Shortcuts

Content is ordered by relevance

User interaction instructs ordering

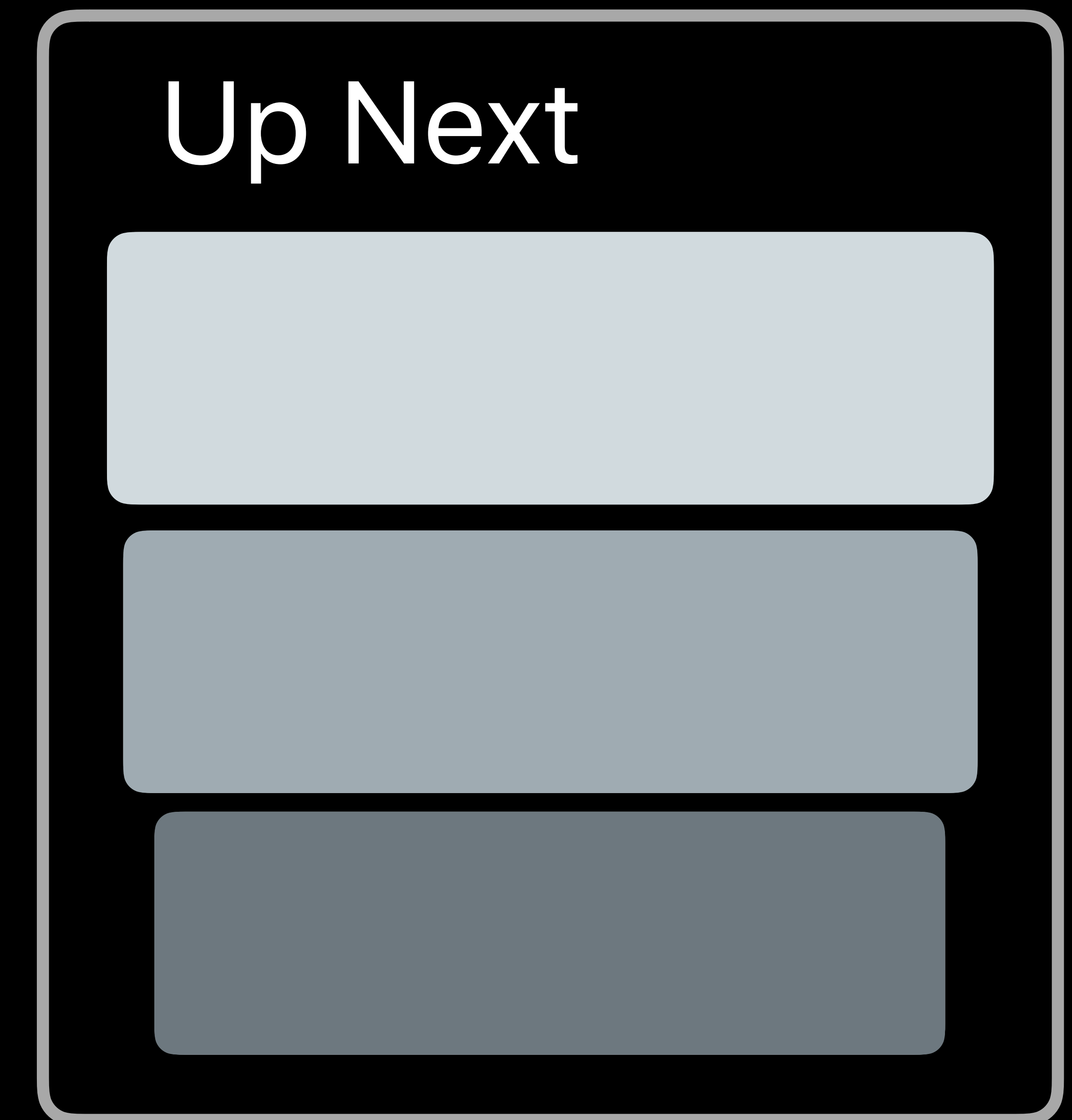
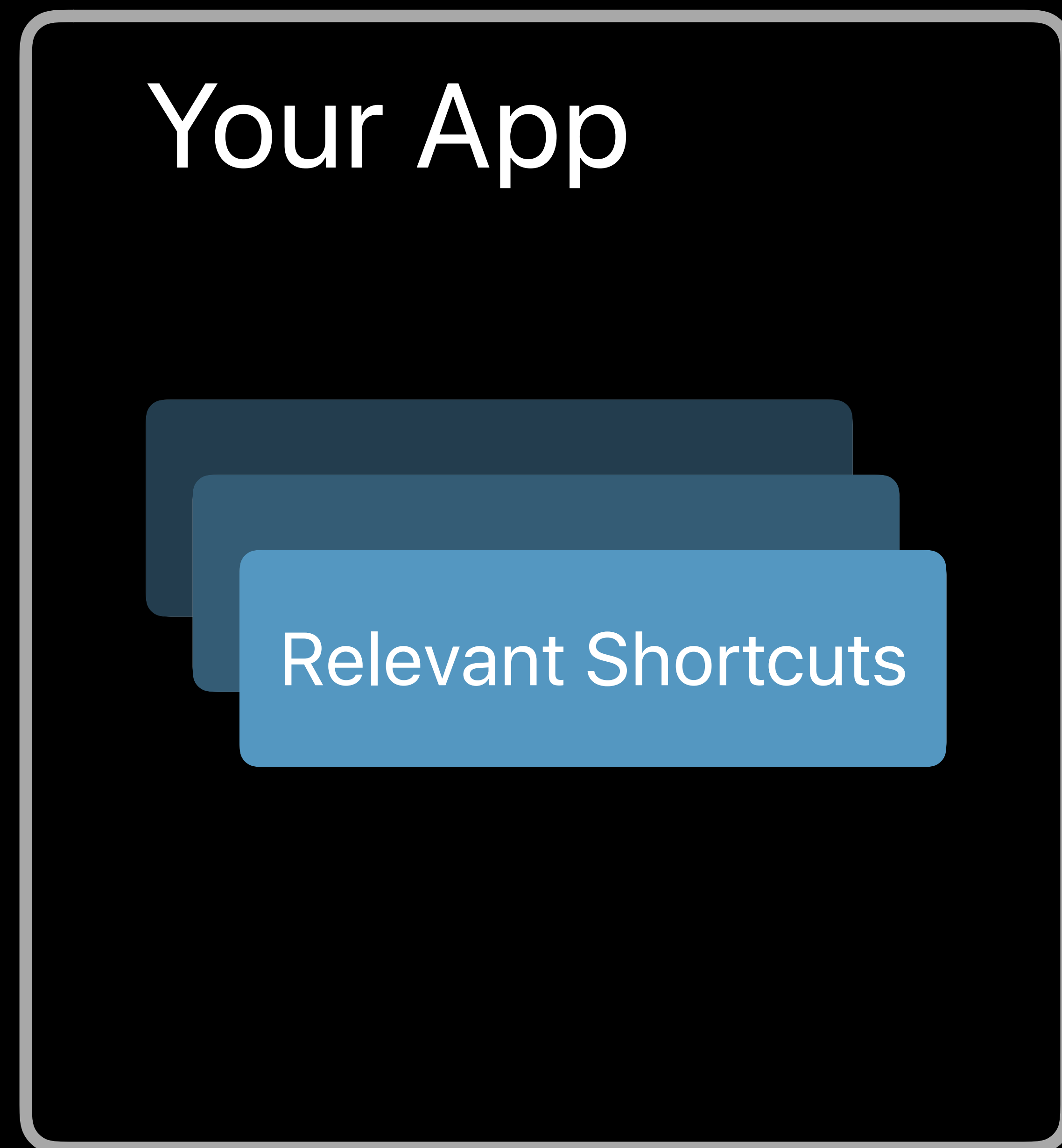
Provide engaging and relevant content

Up Next

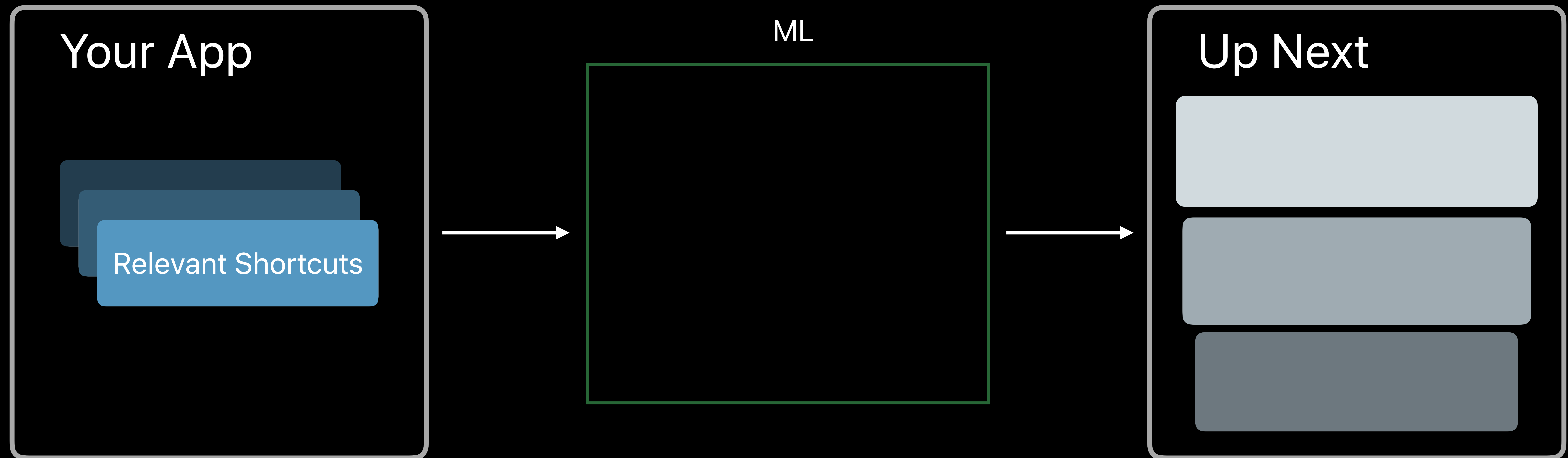
More Relevant

Least Relevant

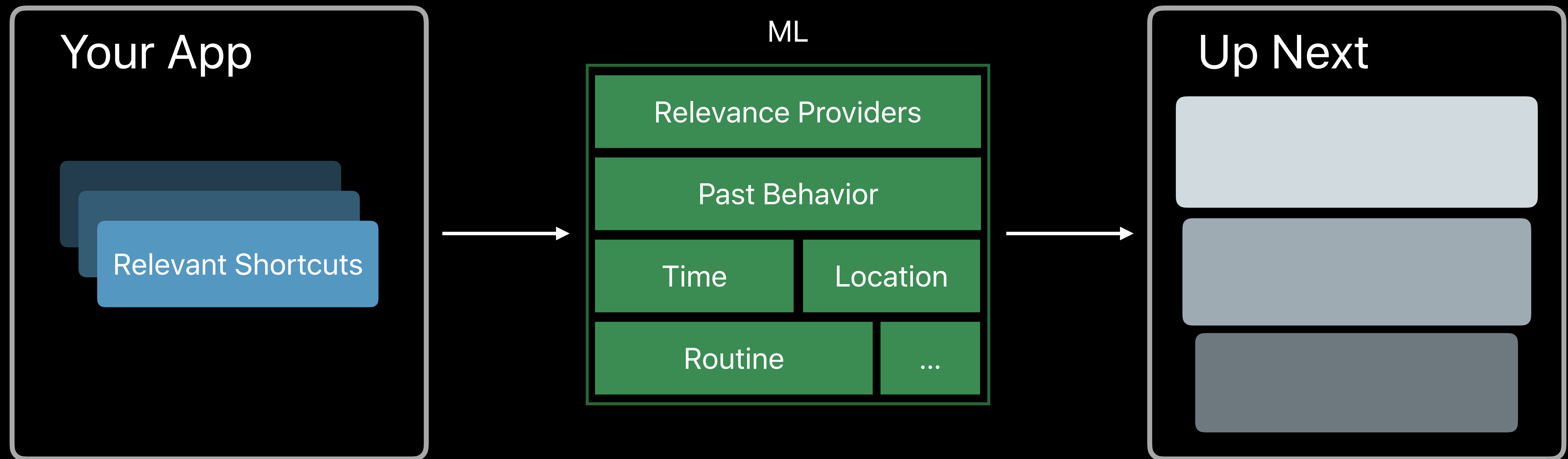
Predicting Shortcuts



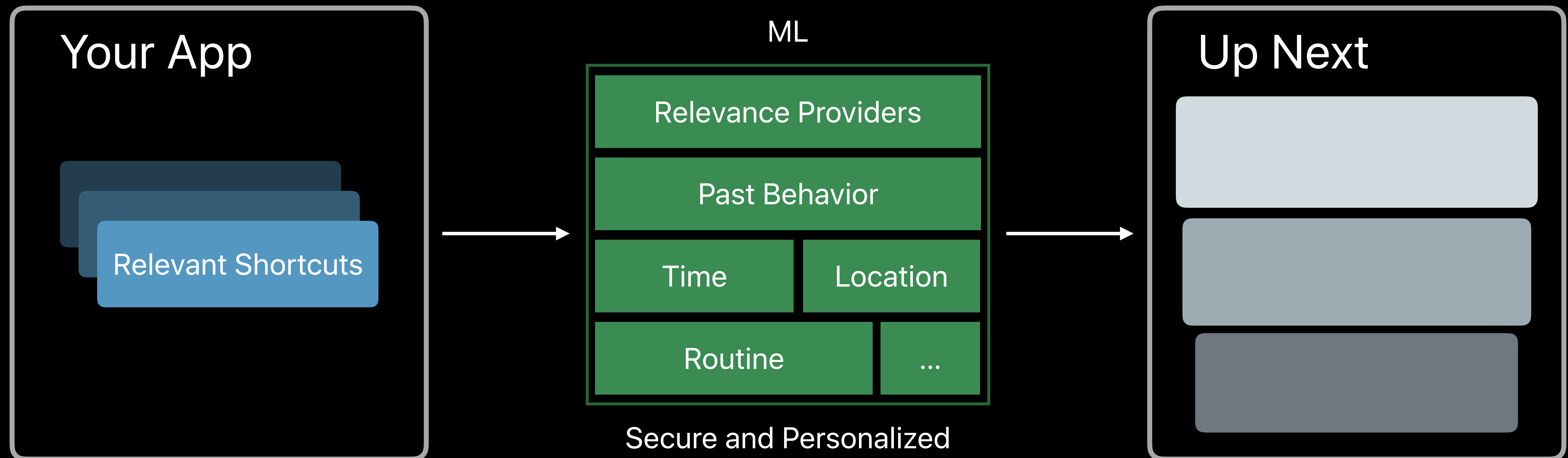
Predicting Shortcuts



Predicting Shortcuts



Predicting Shortcuts



Predicting Shortcuts

Your App

ML

Relevance Providers

Past Behavior

Time

Location

Routine

...

Up Next

Relevant Shortcuts



Predicting Shortcuts

Your App

ML

Relevance Providers

Past Behavior

Time

Location

Routine

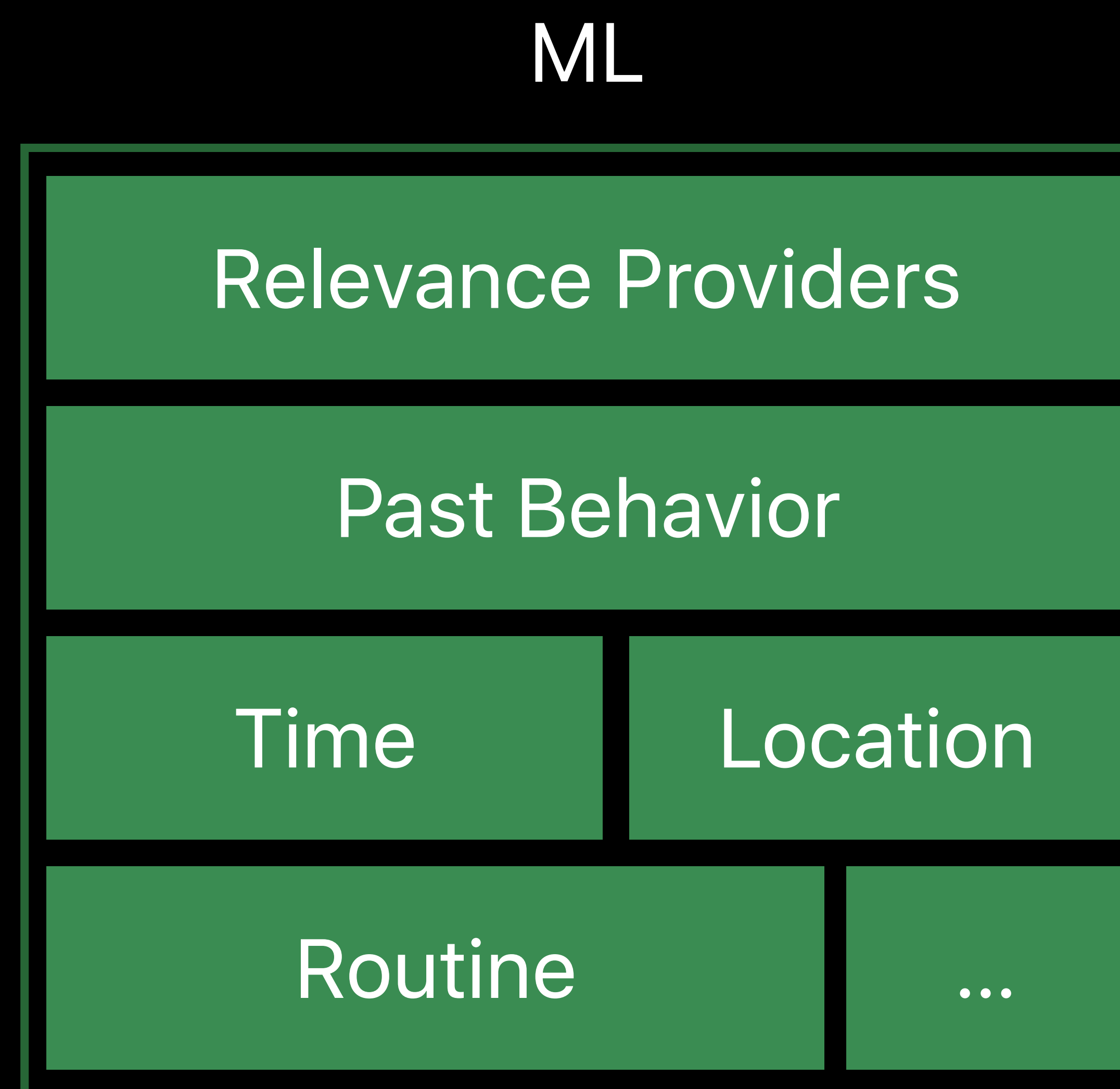
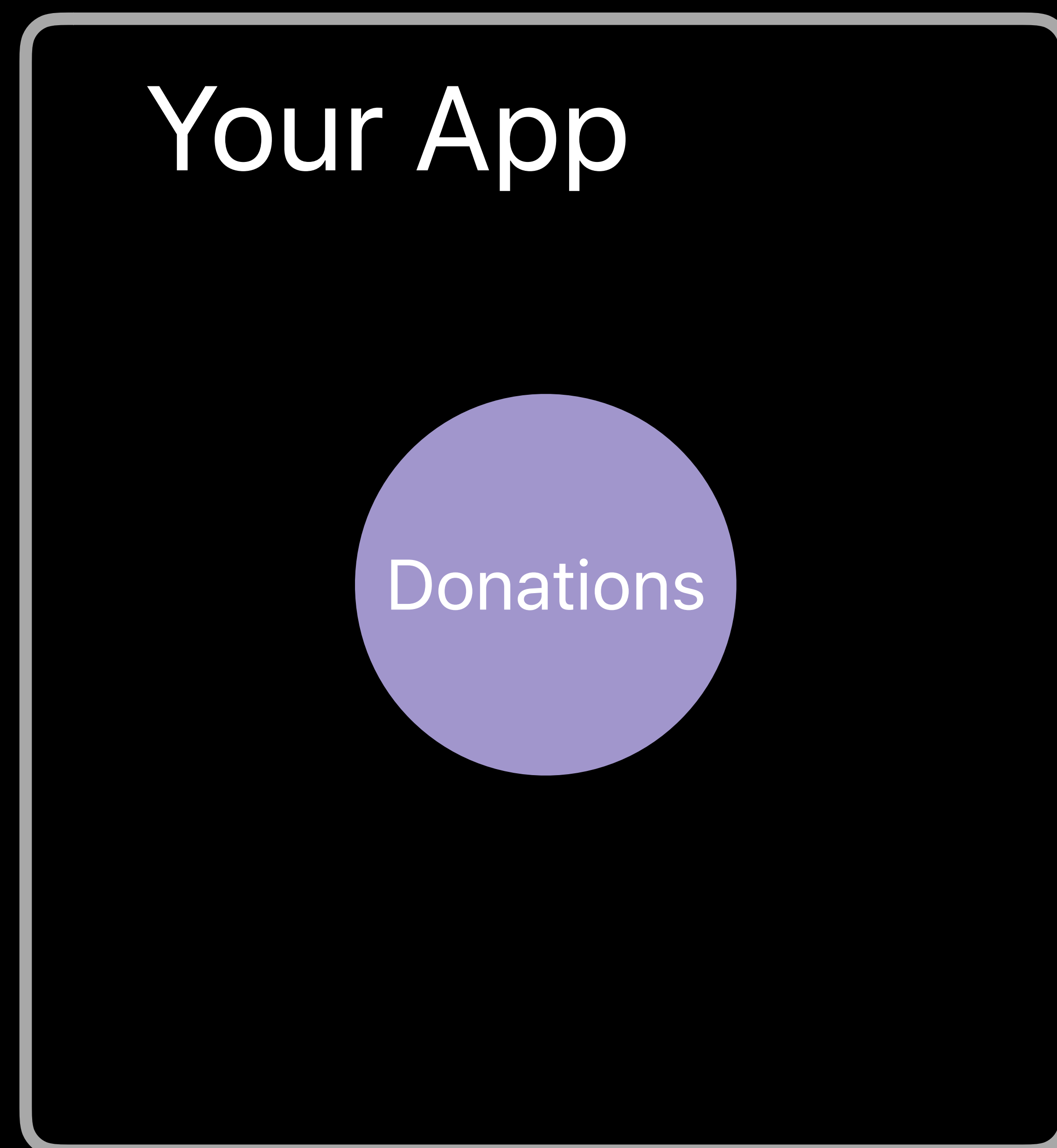
...

Up Next

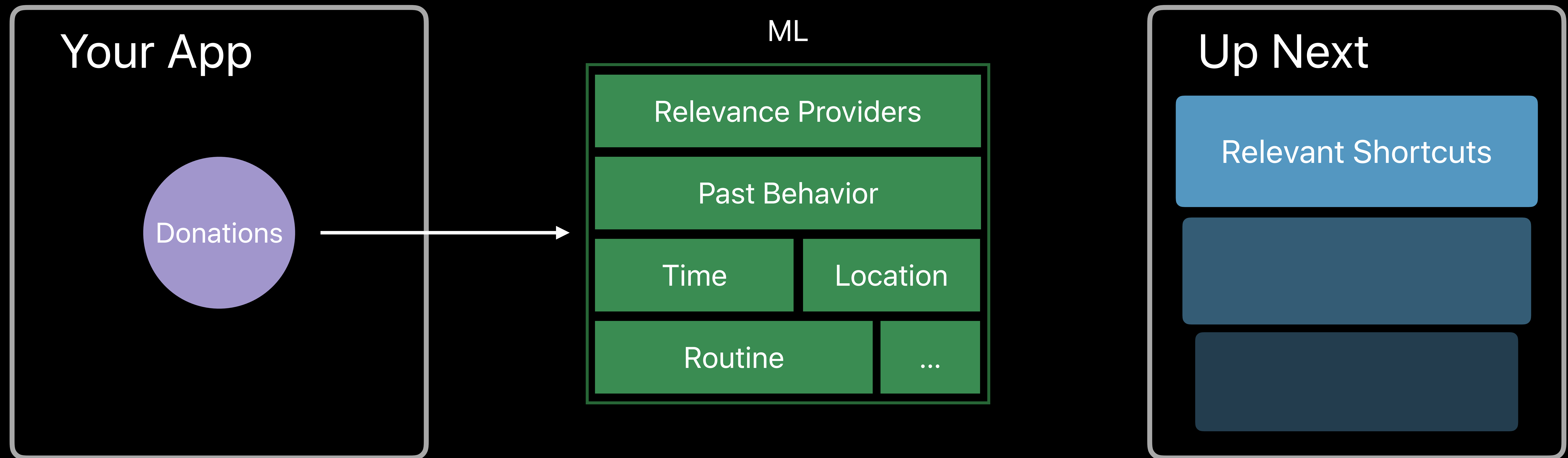
Relevant Shortcuts



Predicting Shortcuts



Predicting Shortcuts



Donations

Indicate important tasks within app

Donations

Indicate important tasks within app

Inform predictions

Donations

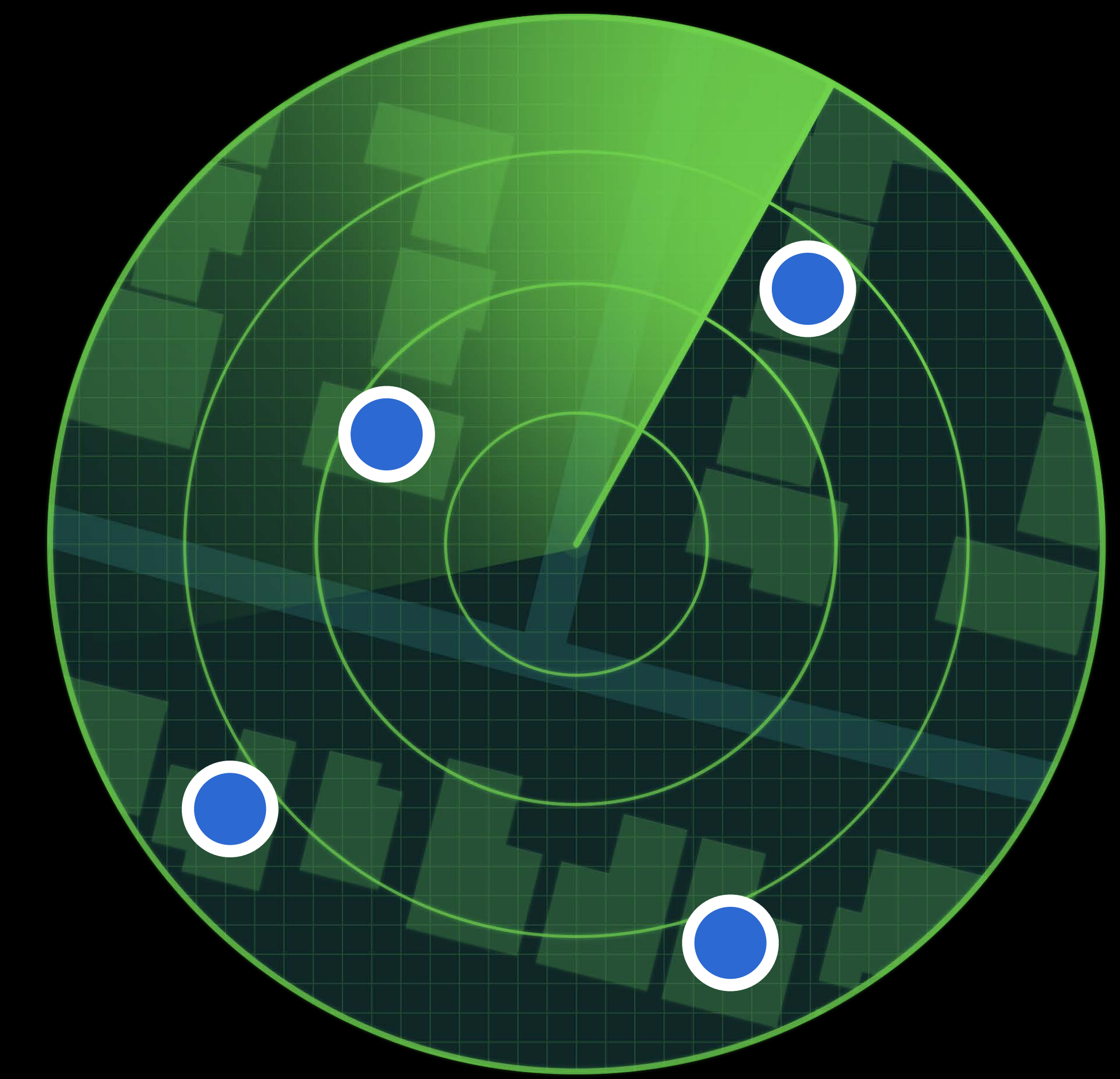
Indicate important tasks within app

Inform predictions

Donations

Indicate important tasks within app

Inform predictions



NSUserActivity Donations

Set `eligibleForPrediction` and `eligibleForSearch`

NSUserActivity Donations

Set `eligibleForPrediction` and `eligibleForSearch`

Supported activity types in Info.plist

NSUserActivity Donations

Set `eligibleForPrediction` and `eligibleForSearch`

Supported activity types in Info.plist

Donate when visible

NSUserActivity Donations

Set `eligibleForPrediction` and `eligibleForSearch`

Supported activity types in Info.plist

Donate when visible

```
let userActivity = NSUserActivity(activityType: "MyActivityType")
userActivity.becomeCurrent()
```

NSUserActivity Donations

NEW

Set `eligibleForPrediction` and `eligibleForSearch`

Supported activity types in Info.plist

Donate when visible

```
let userActivity = NSUserActivity(activityType: "MyActivityType")
userActivity.becomeCurrent()
```

```
let userActivity = NSUserActivity(activityType: "MyActivityType")
interfaceController.updateUserActivity(activityType)
```

NSUserActivity Donations

NEW

Set `eligibleForPrediction` and `eligibleForSearch`

Supported activity types in Info.plist

Donate when visible

```
let userActivity = NSUserActivity(activityType: "MyActivityType")
userActivity.becomeCurrent()
```



```
let userActivity = NSUserActivity(activityType: "MyActivityType")
interfaceController.updateUserActivity(activityType)
```



INIntent Donations

Donate using `INInteraction`

INIntent Donations

Donate using `INInteraction`

```
let intent = /* Create intent and set parameters */  
let interaction = INInteraction(intent: intent, response: nil)  
interaction.donate(completion: nil)
```

INIntent Donations

Donate using `INInteraction`

```
let intent = /* Create intent and set parameters */  
let interaction = INInteraction(intent: intent, response: nil)  
interaction.donate(completion: nil)
```

Indicate primary shortcut type

Primary Shortcut Type

The screenshot displays the configuration for a Custom Intent in an Android application. The breadcrumb path is: SoupChef > Shared > CheckIN.intentdefinition > Custom Intents > Intent.

CUSTOM INTENTS

- Intent (selected)
- Response

Custom Intent

- Category: Check In
- Title: Check In
- Description: Check into a nearby location
- Confirmation: User confirmation required

Parameters

Parameter	Type	Array
location	Location	<input type="checkbox"/>
comment	String	<input type="checkbox"/>

Shortcut Types

- Parameter Combination
 - location
 - location, comment
- Title: [Text Field]
- Subtitle: [Text Field]
- Background: Supports background execution

Intent

- Name: Intent
- Authentication: Not Restricted

Custom Class

- Class Name: IntentIntent

Watch Shortcuts

- Primary Type: location

Bottom bar: + - Filter

Primary Shortcut Type

CUSTOM INTENTS

Custom Intent

Category: Check In

Title: Check In

Description: Check into a nearby location

Confirmation: User confirmation required

Parameters

Parameter	Type	Array
location	Location	<input type="checkbox"/>
comment	String	<input type="checkbox"/>

Shortcut Types

Parameter Combination

- location
- location, comment

Title:

Subtitle:

Background: Supports background execution

Intent

Name: Intent

Authentication: Not Restricted

Custom Class

Class Name: IntentIntent

Watch Shortcuts

Primary Type: location

Primary Shortcut Type

The screenshot shows the Android Studio interface for configuring a custom intent. The breadcrumb path is: SoupChef > Shared > CheckIN.intentdefinition > Custom Intents > Intent. The main editor displays the 'Custom Intent' configuration for an intent named 'Intent' with category 'Check In' and title 'Check In'. The description is 'Check into a nearby location'. Under 'Parameters', there are two entries: 'location' (Type: Location, Array: unchecked) and 'comment' (Type: String, Array: unchecked). Under 'Shortcut Types', there are two entries: 'location' and 'location, comment'. The 'Background' checkbox is checked, indicating 'Supports background execution'. A 'Watch Shortcuts' dialog is overlaid on the right, showing the 'Primary Type' dropdown set to 'location'.

Custom Intent Configuration:

- Name: Intent
- Authentication: Not Restricted
- Category: Check In
- Title: Check In
- Description: Check into a nearby location
- Confirmation: User confirmation required

Parameters:

Parameter	Type	Array
location	Location	<input type="checkbox"/>
comment	String	<input type="checkbox"/>

Shortcut Types:

Parameter Combination	Title	Subtitle
location		
location, comment		

Watch Shortcuts Dialog:

- Primary Type: location

Primary Shortcut Type

Indicate primary use case

Primary Shortcut Type

Indicate primary use case

One per intent

Primary Shortcut Type

Indicate primary use case

One per intent

Subset of Relevant Shortcut parameters

Primary Shortcut Type

Indicate primary use case

One per intent

Subset of Relevant Shortcut parameters

Quickly identify patterns

Primary Shortcut Type



Primary Shortcut Type



Recipient, Message

Recipient

Primary Shortcut Type



Recipient, Message

Recipient

Primary Shortcut Type



Recipient, Message



Recipient

Primary Shortcut Type



Primary Shortcut Type



Coffee, Condiment, Location

Coffee, Condiment

Coffee

Primary Shortcut Type

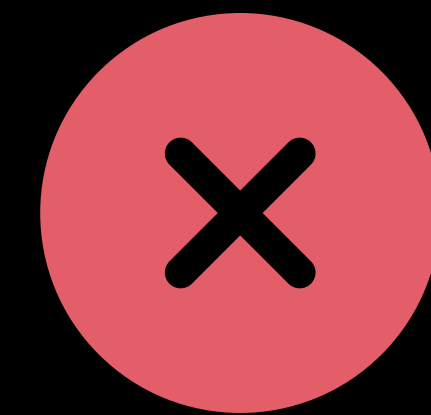


Coffee, Condiment, Location

Coffee, Condiment

Coffee

Primary Shortcut Type



Coffee, Condiment, Location



Coffee, Condiment

Coffee

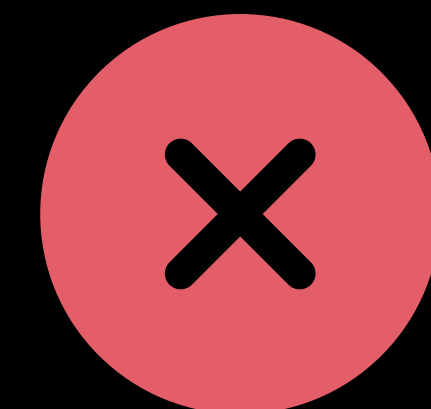
Primary Shortcut Type



Coffee, Condiment, Location



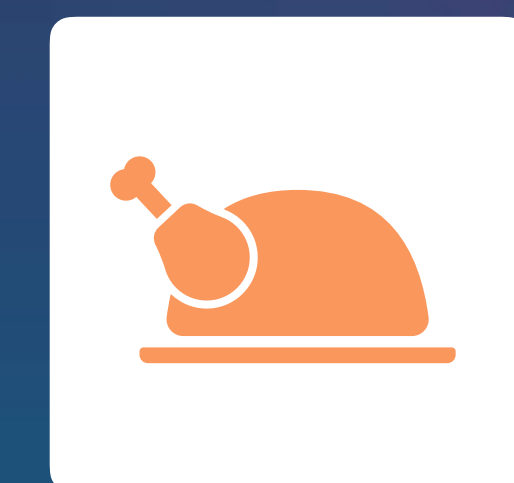
Coffee, Condiment



Coffee

Predicting Relevant Shortcuts

Recipes



Chicken
Parmesan

Fitness Trainer

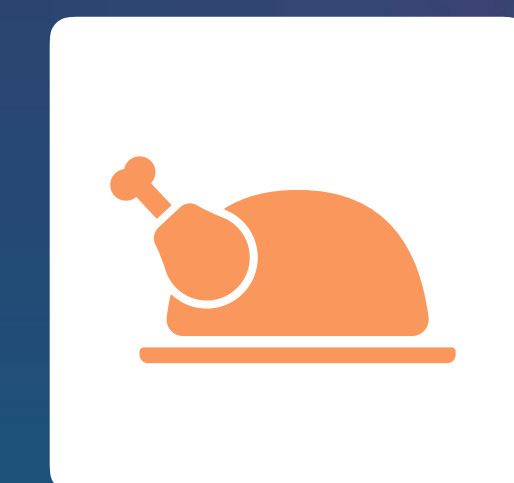
Evening Run
8:00 PM

Travel Guide

Nearby Landmarks
Golden Gate Park

Predicting Relevant Shortcuts

Recipes



Chicken
Parmesan

Fitness Trainer

Evening Run
8:00 PM

Travel Guide

Nearby Landmarks
Golden Gate Park

Predicting Relevant Shortcuts

Recipes



Chicken
Parmesan

Fitness Trainer

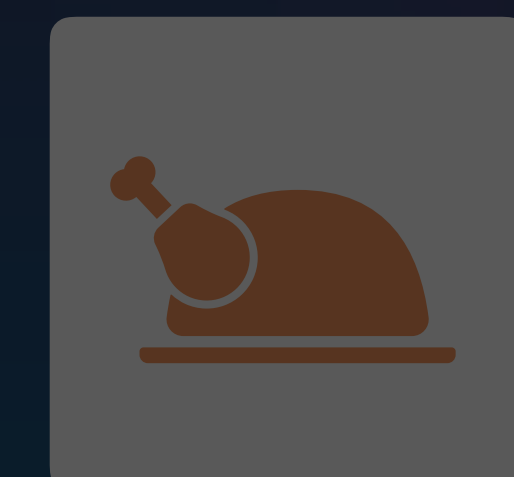
Evening Run
8:00 PM

Travel Guide

Nearby Landmarks
Golden Gate Park

Predicting Relevant Shortcuts

Recipes



Chicken
Parmesan

Fitness Trainer

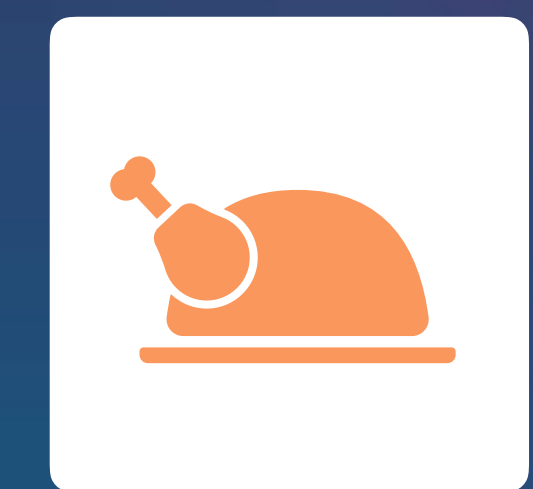
Evening Run
8:00 PM

Travel Guide

Nearby Landmarks
Golden Gate Park

Predicting Relevant Shortcuts

 **Recipes**



Chicken
Parmesan

Downtime

 **Fitness Trainer**

Evening Run
8:00 PM

Time

 **Travel Guide**

Nearby Landmarks
Golden Gate Park

Location

Relevance Providers

Hint ideal time to surface

Relevance Providers

Hint ideal time to surface

Surface new content

Relevance Providers

Hint ideal time to surface

Surface new content

User engagement paramount

Relevance Providers

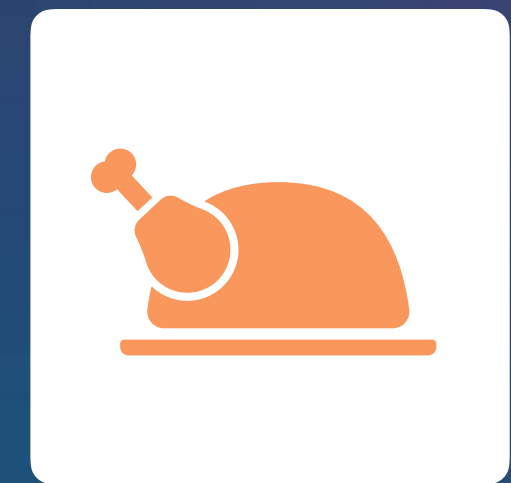
 **Recipes**



Chicken
Parmesan

Relevance Providers

Recipes



Chicken
Parmesan

```
relevantShortcut.relevanceProviders = []
```

Relevance Providers

Recipes



Chicken
Parmesan

```
relevantShortcut.relevanceProviders = []
```

Based on user engagement

INDateRelevanceProvider

 **Fitness Trainer**

Evening Run

8:00 PM

INDateRelevanceProvider


 **Fitness Trainer**

Evening Run

8:00 PM

```
let dateRelevanceProvider = INDateRelevanceProvider(start: eventDate, end: nil)
relevantShortcut.relevanceProviders = [dateRelevanceProvider]
```


INDateRelevanceProvider

 **Fitness Trainer**
Evening Run
8:00 PM

```
let dateRelevanceProvider = INDateRelevanceProvider(start: eventDate, end: nil)
relevantShortcut.relevanceProviders = [dateRelevanceProvider]
```

Surface at specific time

INDateRelevanceProvider

 **Fitness Trainer**
Evening Run
8:00 PM



```
let dateRelevanceProvider = INDateRelevanceProvider(start: eventDate, end: nil)
relevantShortcut.relevanceProviders = [dateRelevanceProvider]
```


Surface at specific time

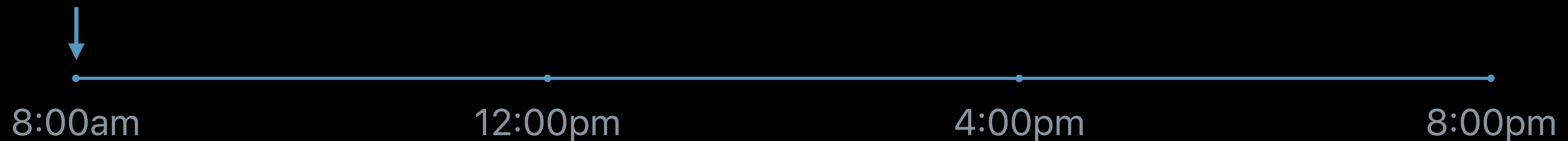
More relevant closer to event

INDateRelevanceProvider

10:00–11:00 AM
WWDC Prep
Apple Park



 **Space News**
 New Planets
Found


 **Fitness Trainer**
Evening Run
8:00 PM



INDateRelevanceProvider

10:00–11:00AM
WWDC Prep
Apple Park

 **Space News**
 New Planets
Found

 **Fitness Trainer**
Evening Run
8:00 PM



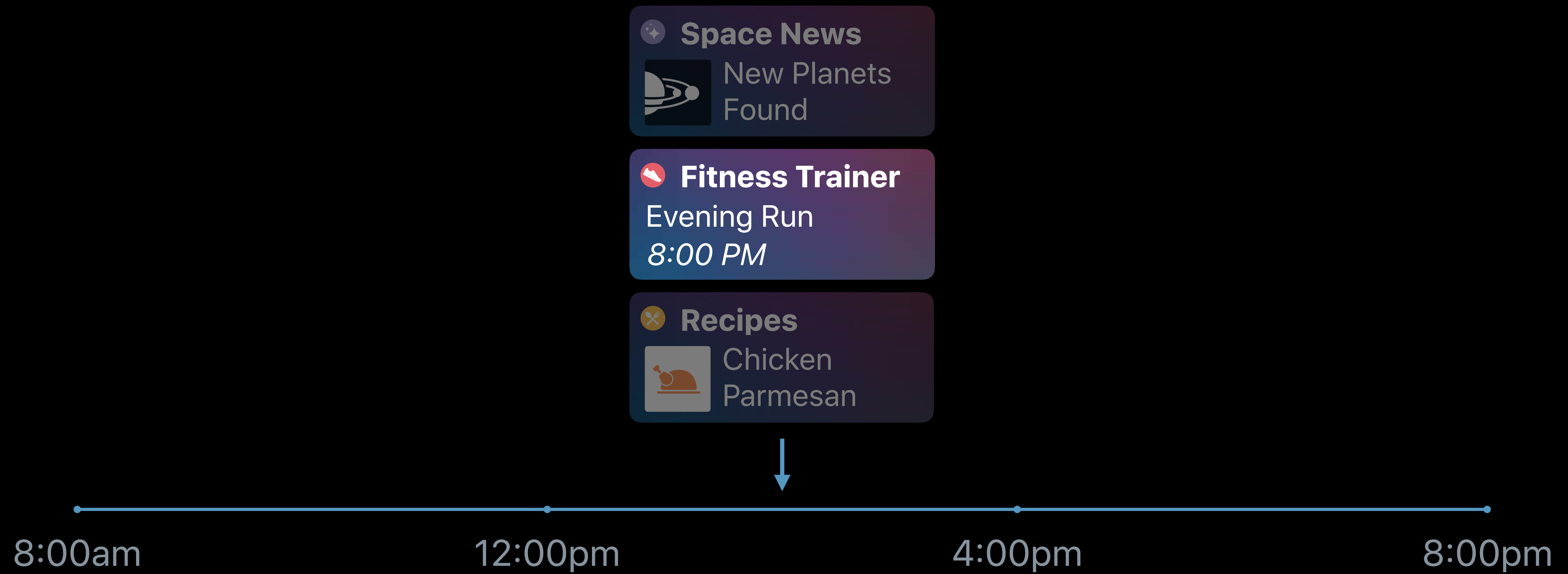
8:00am

12:00pm

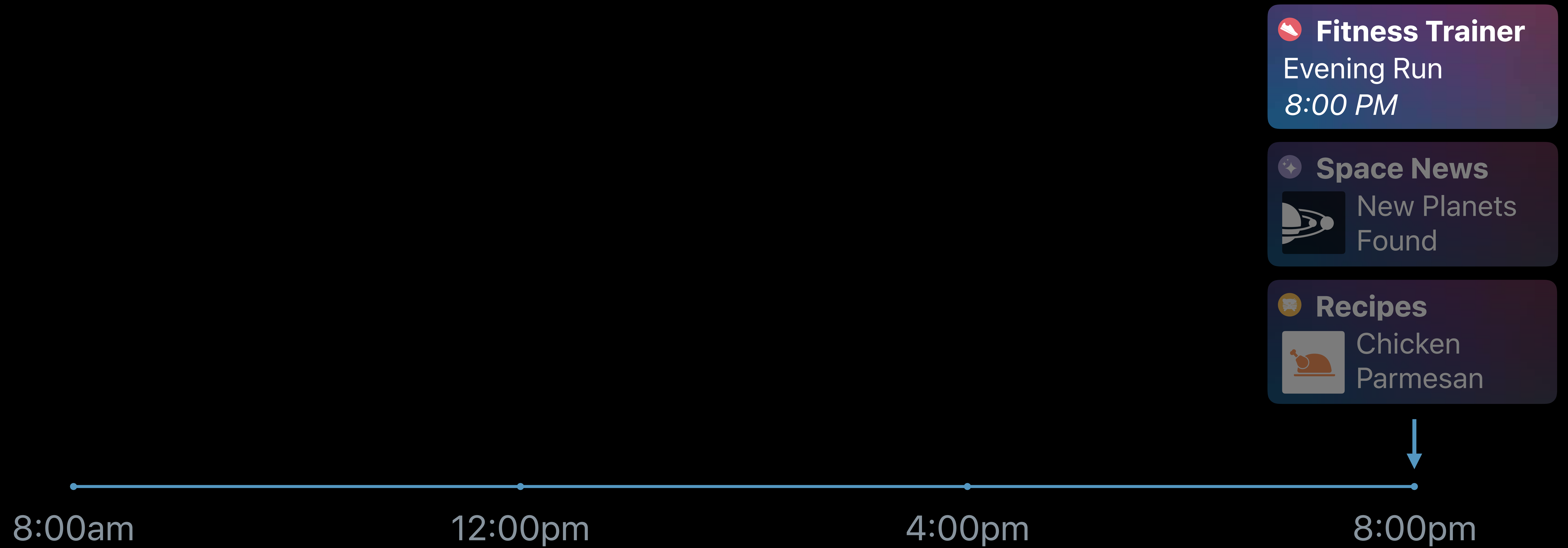
4:00pm

8:00pm

INDateRelevanceProvider



INDateRelevanceProvider



INLocationRelevanceProvider

Travel Guide

Nearby Landmarks

Golden Gate Park

INLocationRelevanceProvider

Travel Guide

Nearby Landmarks

Golden Gate Park

```
let locationRelevanceProvider = INLocationRelevanceProvider(region: region)
relevantShortcut.relevanceProviders = [locationRelevanceProvider]
```

INLocationRelevanceProvider

Travel Guide

Nearby Landmarks

Golden Gate Park

```
let locationRelevanceProvider = INLocationRelevanceProvider(region: region)
relevantShortcut.relevanceProviders = [locationRelevanceProvider]
```

Surface around location

INLocationRelevanceProvider

Travel Guide

Nearby Landmarks

Golden Gate Park


```
let locationRelevanceProvider = INLocationRelevanceProvider(region: region)
relevantShortcut.relevanceProviders = [locationRelevanceProvider]
```

Surface around location

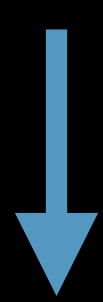
More relevant closer to region

INLocationRelevanceProvider

2:00–3:00PM
Coffee with Paul

🌠 **Space News**
 New Planets Found

📍 **Travel Guide**
Nearby Landmarks
Golden Gate Park

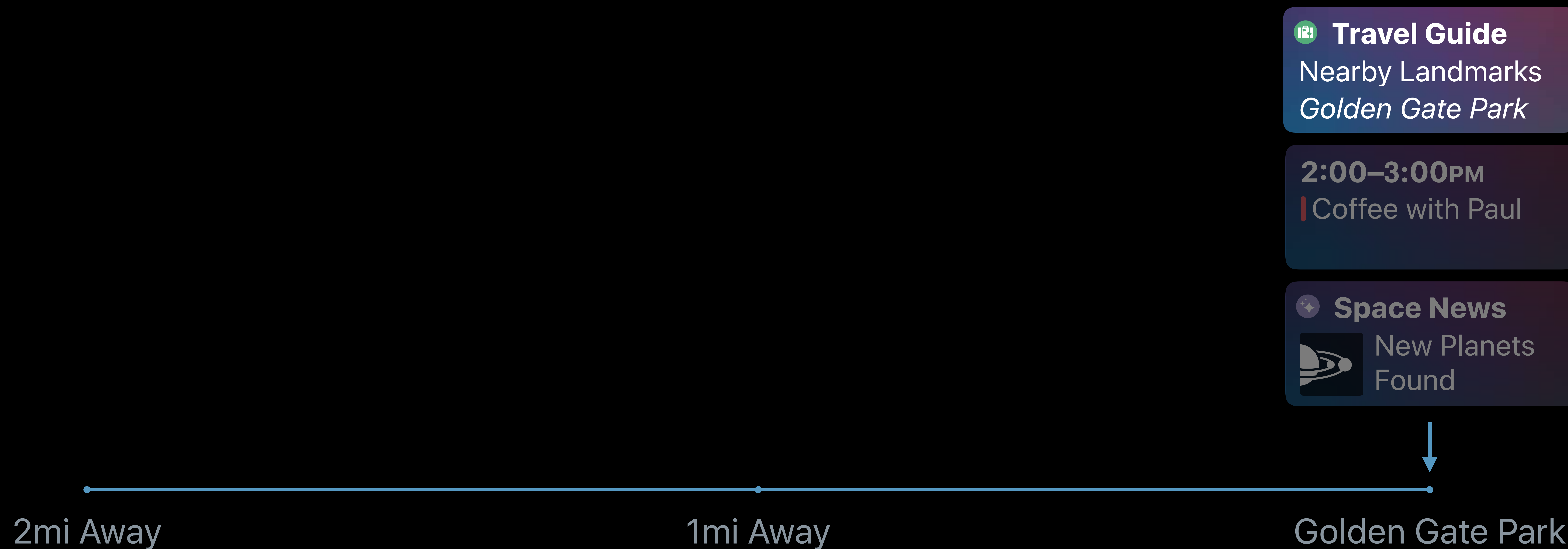


2mi Away

1mi Away

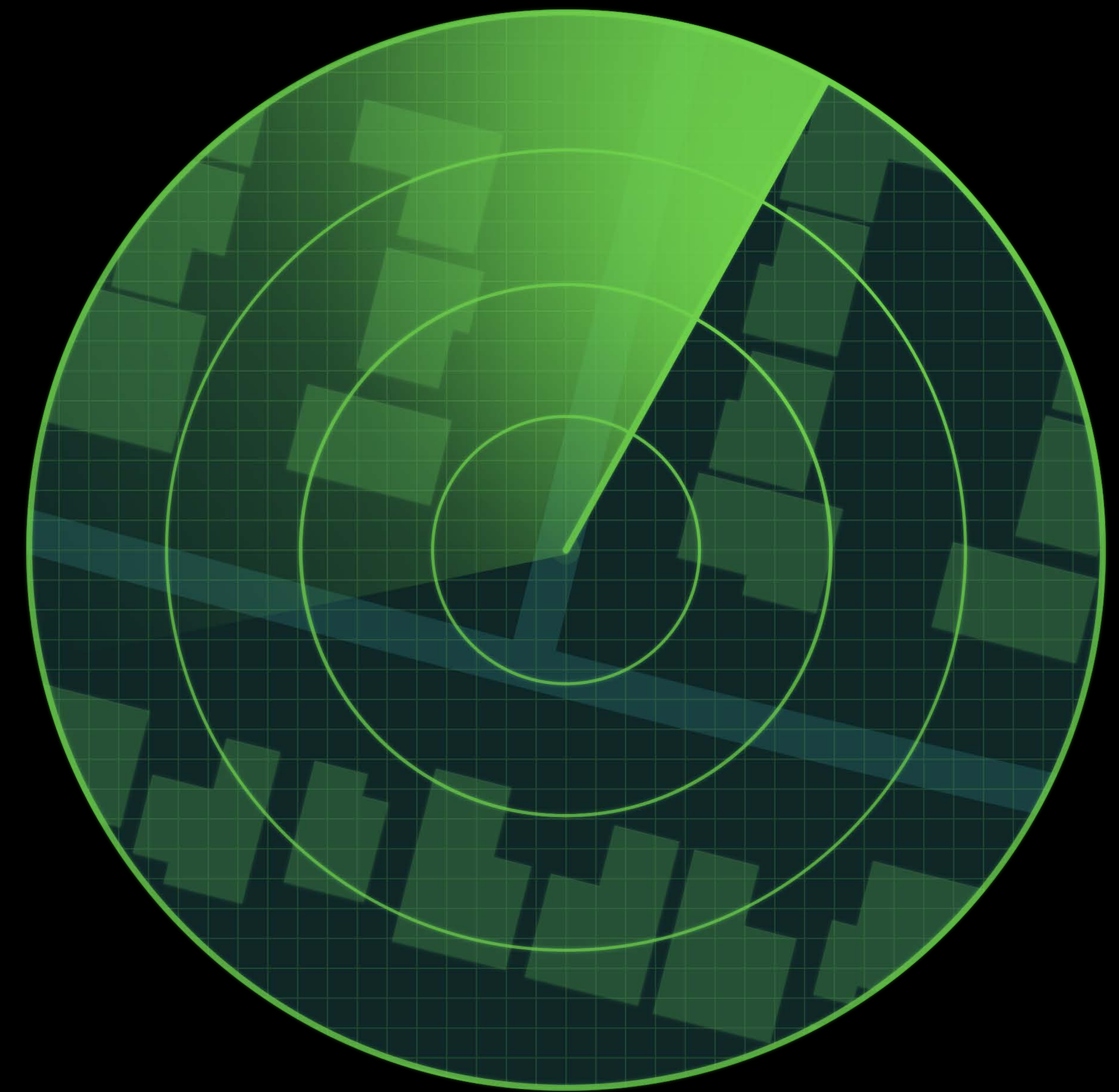
Golden Gate Park

INLocationRelevanceProvider



INLocationRelevanceProvider

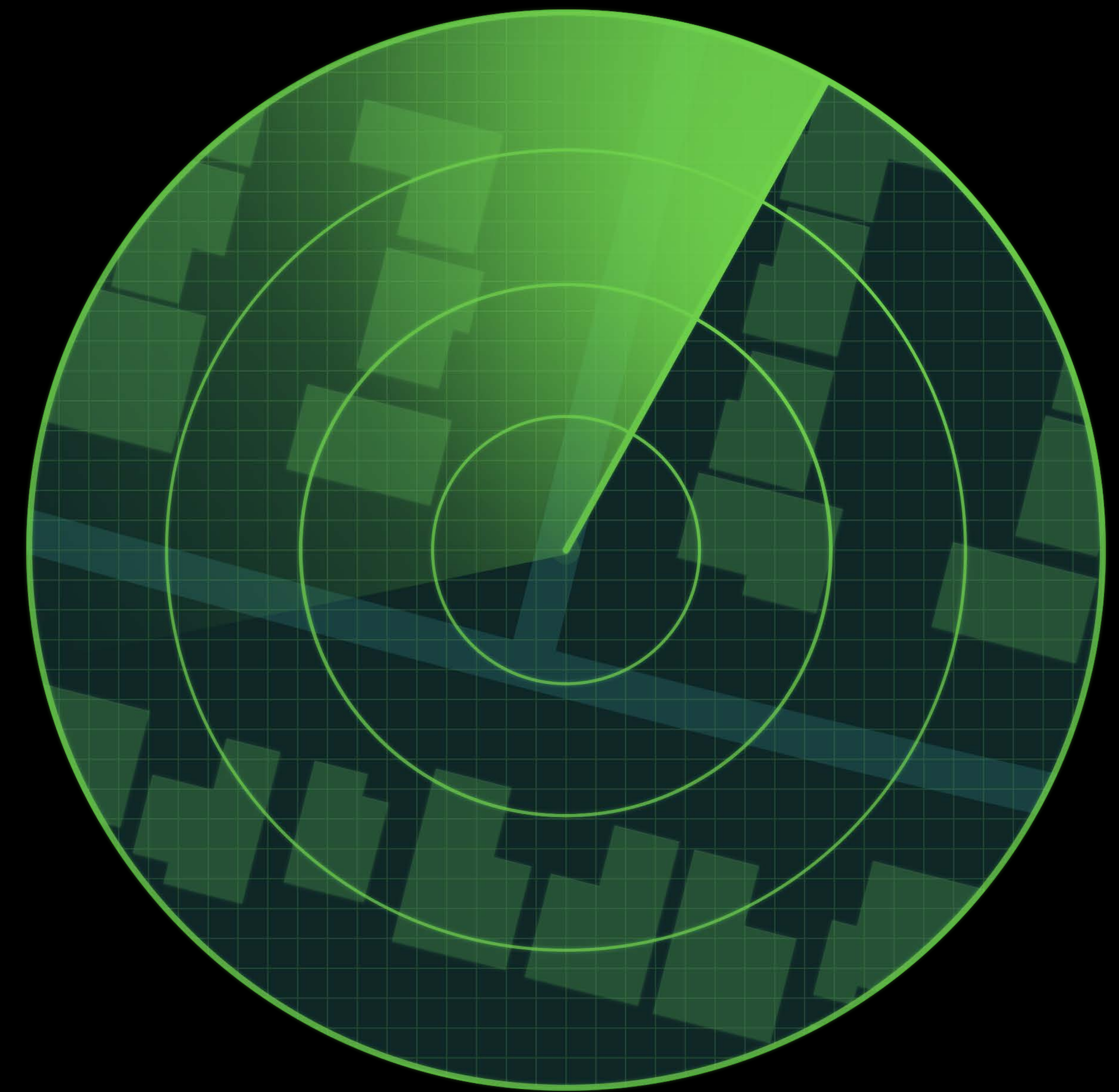
Provide a `CLRegion`



INLocationRelevanceProvider

Provide a `CLRegion`

Change region interpretation

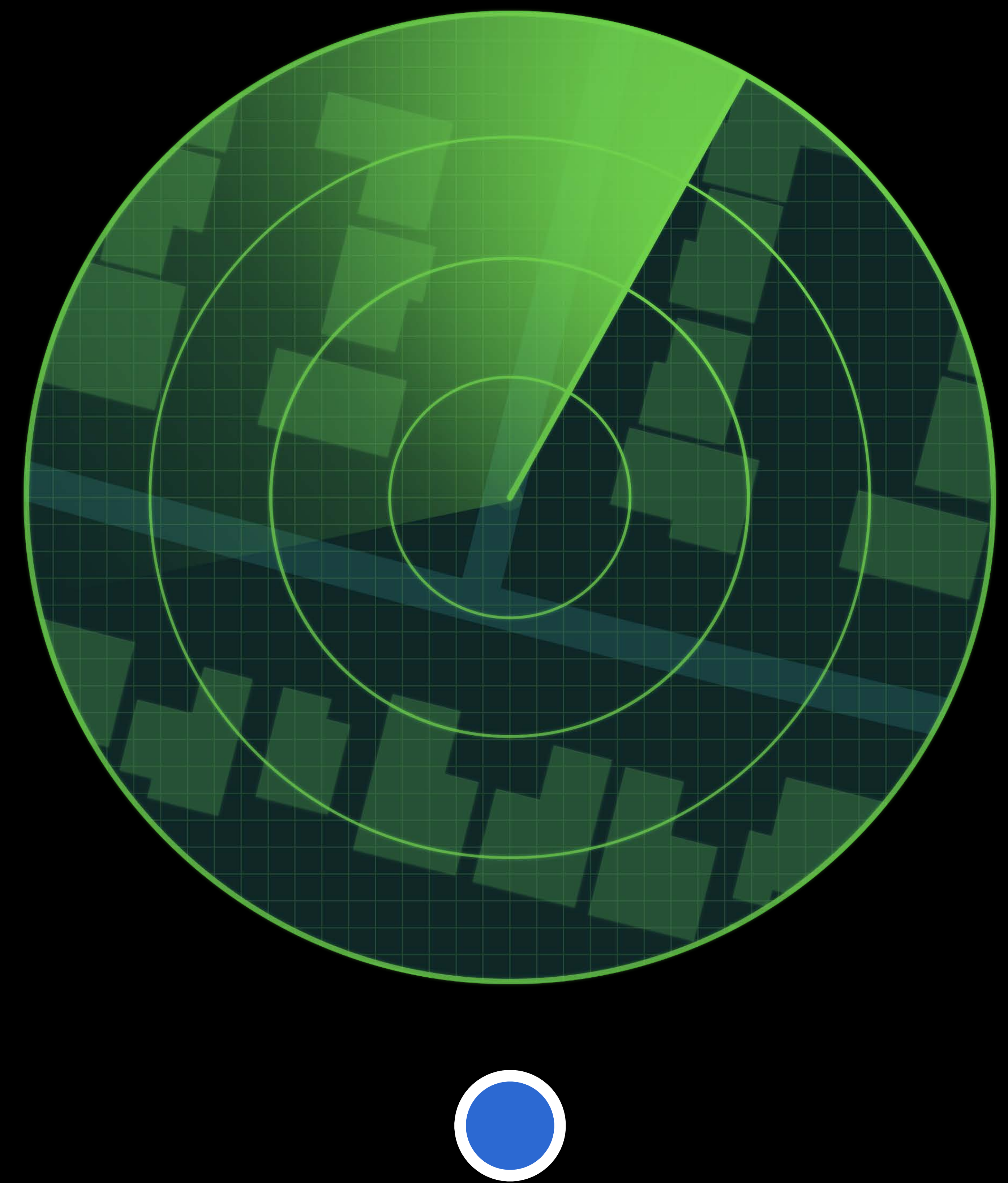


INLocationRelevanceProvider

Provide a `CLRegion`

Change region interpretation

- `notifyOnEntry`

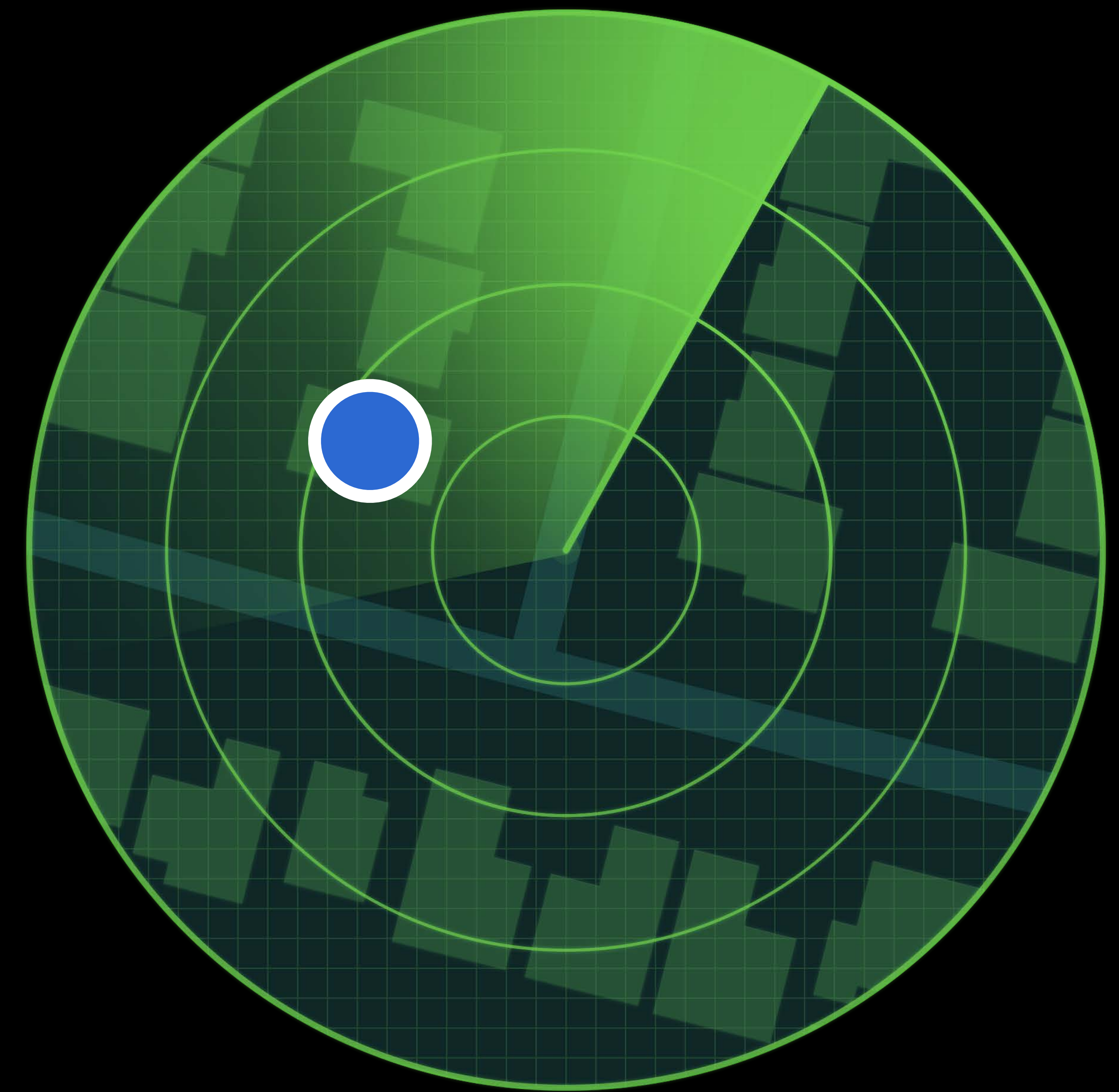


INLocationRelevanceProvider

Provide a `CLRegion`

Change region interpretation

- `notifyOnEntry`

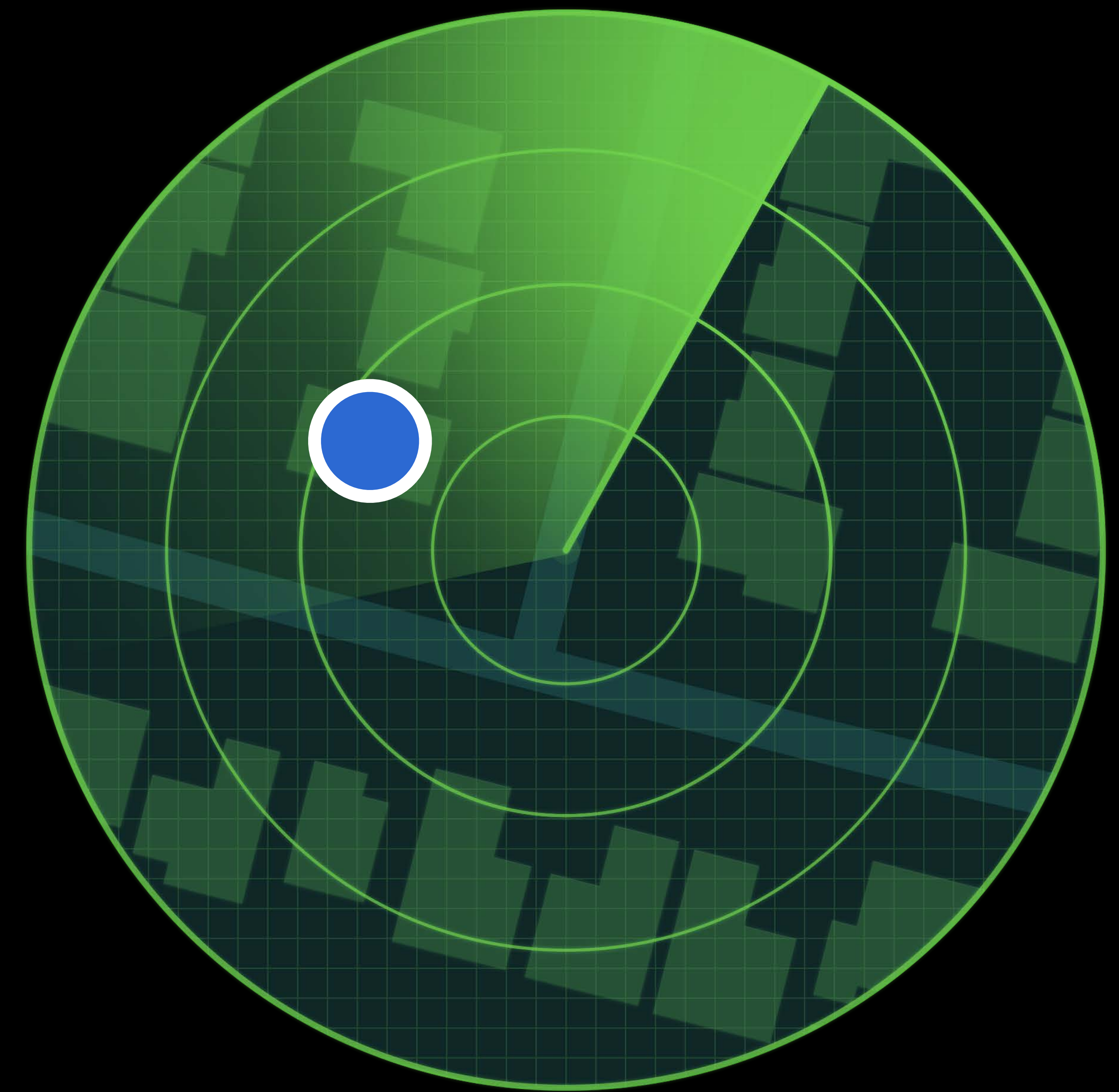


INLocationRelevanceProvider

Provide a `CLRegion`

Change region interpretation

- `notifyOnEntry`
- `notifyOnExit`

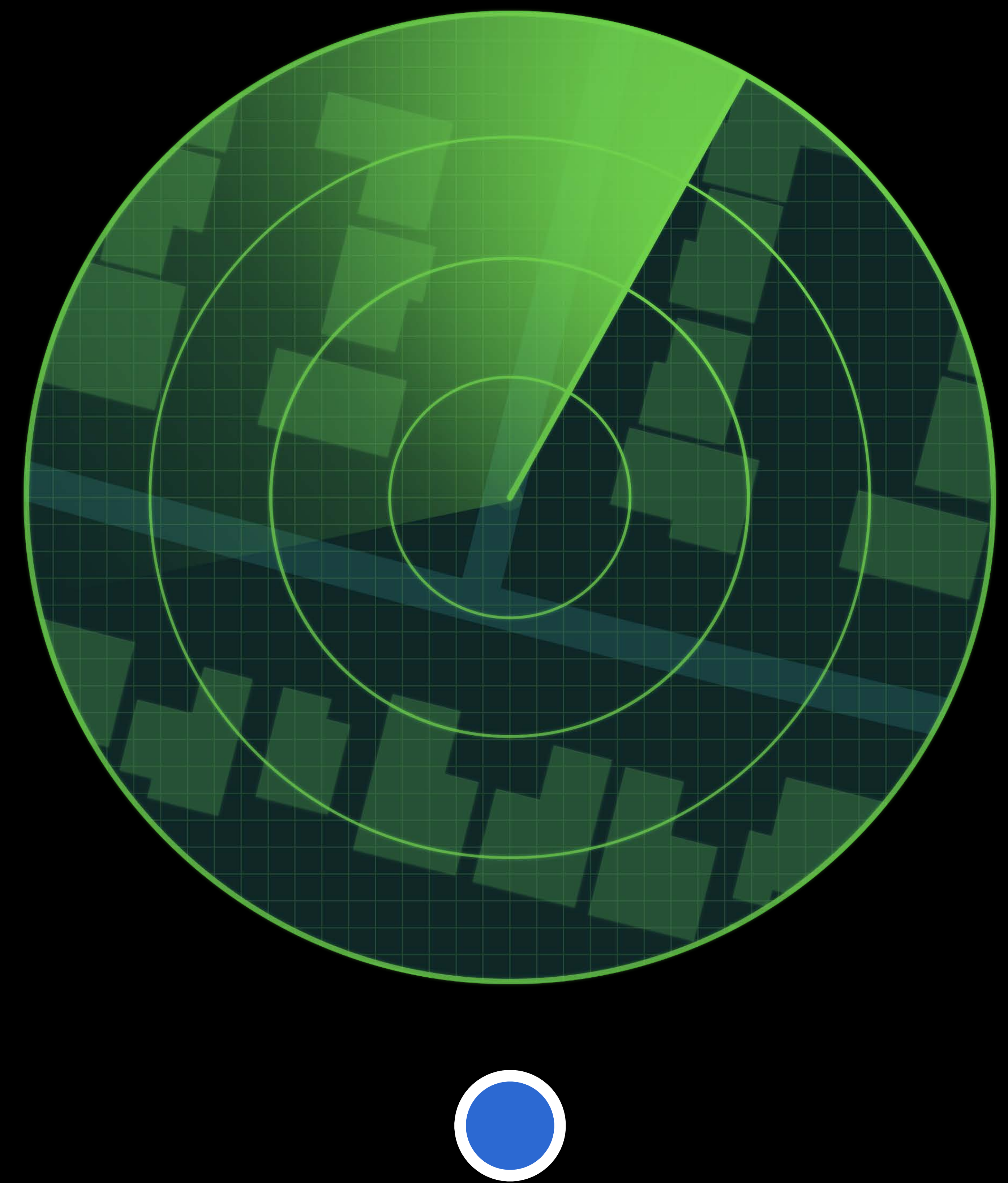


INLocationRelevanceProvider

Provide a `CLRegion`

Change region interpretation

- `notifyOnEntry`
- `notifyOnExit`



```
// Creating a Region
```

```
let center = CLLocationCoordinate2D(latitude: 37.334728, longitude: -122.008883)
```

```
let region = CLCircularRegion(center: center, radius: 2.0, identifier: "Apple Park")
```

```
region.notifyOnEntry = true
```

```
region.notifyOnExit = false
```

```
let relevanceProvider = INLocationRelevanceProvider(region: region)
```

```
// Creating a Region
```

```
let center = CLLocationCoordinate2D(latitude: 37.334728, longitude: -122.008883)
```

```
let region = CLCircularRegion(center: center, radius: 2.0, identifier: "Apple Park")
```

```
region.notifyOnEntry = true
```

```
region.notifyOnExit = false
```

```
let relevanceProvider = INLocationRelevanceProvider(region: region)
```



```
// Creating a Region
```

```
let center = CLLocationCoordinate2D(latitude: 37.334728, longitude: -122.008883)
let region = CLCircularRegion(center: center, radius: 2.0, identifier: "Apple Park")
```

```
region.notifyOnEntry = true
region.notifyOnExit = false
```

```
let relevanceProvider = INLocationRelevanceProvider(region: region)
```



```
// Creating a Region
```

```
let center = CLLocationCoordinate2D(latitude: 37.334728, longitude: -122.008883)
```

```
let region = CLCircularRegion(center: center, radius: 2.0, identifier: "Apple Park")
```

```
region.notifyOnEntry = true
```

```
region.notifyOnExit = false
```

```
let relevanceProvider = INLocationRelevanceProvider(region: region)
```

```
// Creating a Region
```

```
let center = CLLocationCoordinate2D(latitude: 37.334728, longitude: -122.008883)
```

```
let region = CLCircularRegion(center: center, radius: 2.0, identifier: "Apple Park")
```

```
region.notifyOnEntry = true
```

```
region.notifyOnExit = false
```

```
let relevanceProvider = INLocationRelevanceProvider(region: region)
```

```
// Creating a Region
```

```
let center = CLLocationCoordinate2D(latitude: 37.334728, longitude: -122.008883)
```

```
let region = CLCircularRegion(center: center, radius: 2.0, identifier: "Apple Park")
```

```
region.notifyOnEntry = true
```

```
region.notifyOnExit = false
```

```
let relevanceProvider = INLocationRelevanceProvider(region: region)
```

```
// Creating a Region
```

```
let center = CLLocationCoordinate2D(latitude: 37.334728, longitude: -122.008883)
```

```
let region = CLCircularRegion(center: center, radius: 2.0, identifier: "Apple Park")
```

```
region.notifyOnEntry = true
```

```
region.notifyOnExit = false
```

```
let relevanceProvider = INLocationRelevanceProvider(region: region)
```

INLocationRelevanceProvider

Requires location authorization

INLocationRelevanceProvider

Requires location authorization

Location updates are limited

Providing Personalization

Use `INDailyRoutineRelevanceProvider`

Providing Personalization

Use `INDailyRoutineRelevanceProvider`

Meaningful times or locations to the user

Providing Personalization

Use `INDailyRoutineRelevanceProvider`

Meaningful times or locations to the user

Personalized to each user

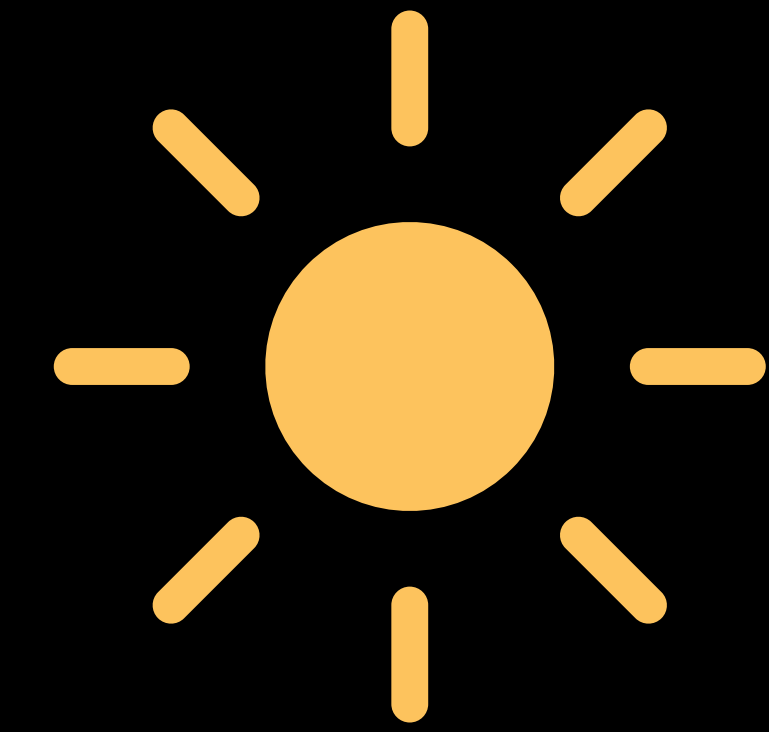
Providing Personalization

Morning

Providing Personalization

Morning

- After user wakes up

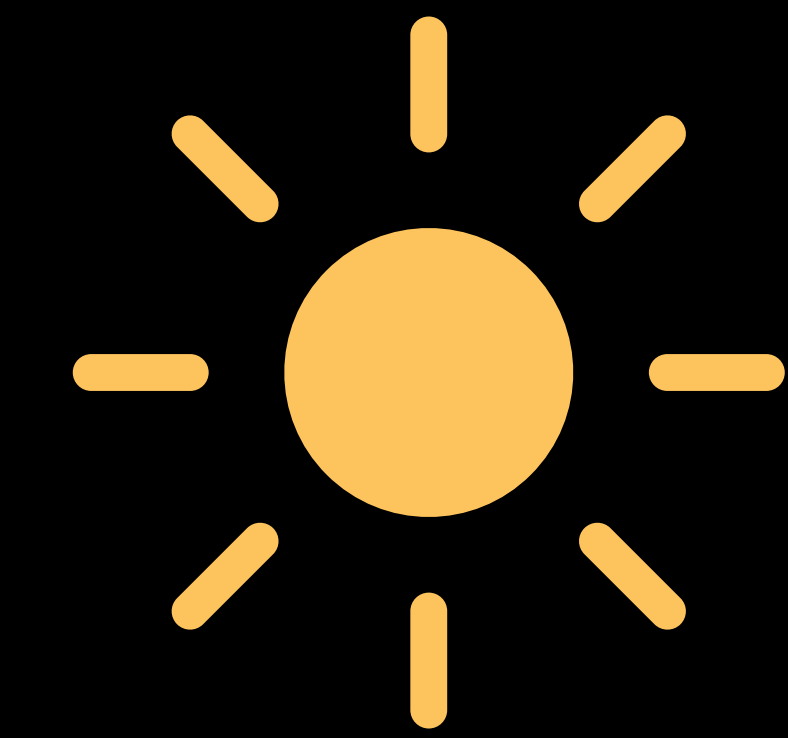


Providing Personalization

Morning

- After user wakes up

Evening



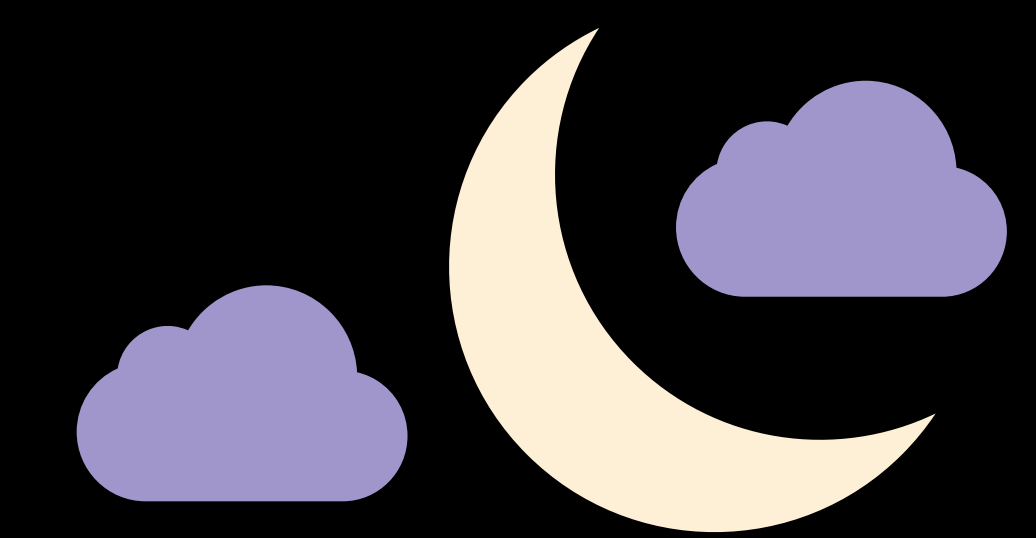
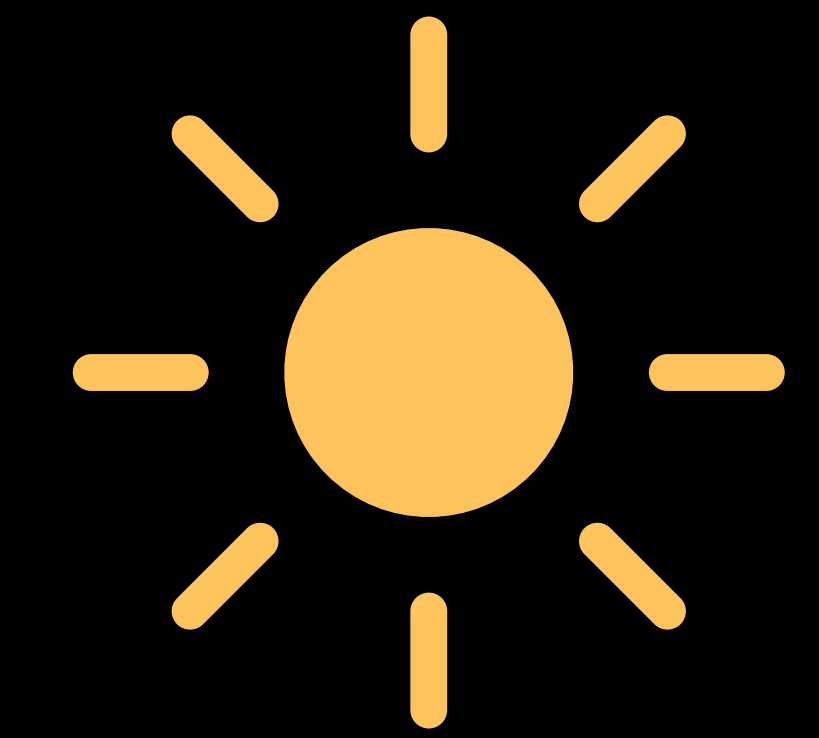
Providing Personalization

Morning

- After user wakes up

Evening

- Before going to bed



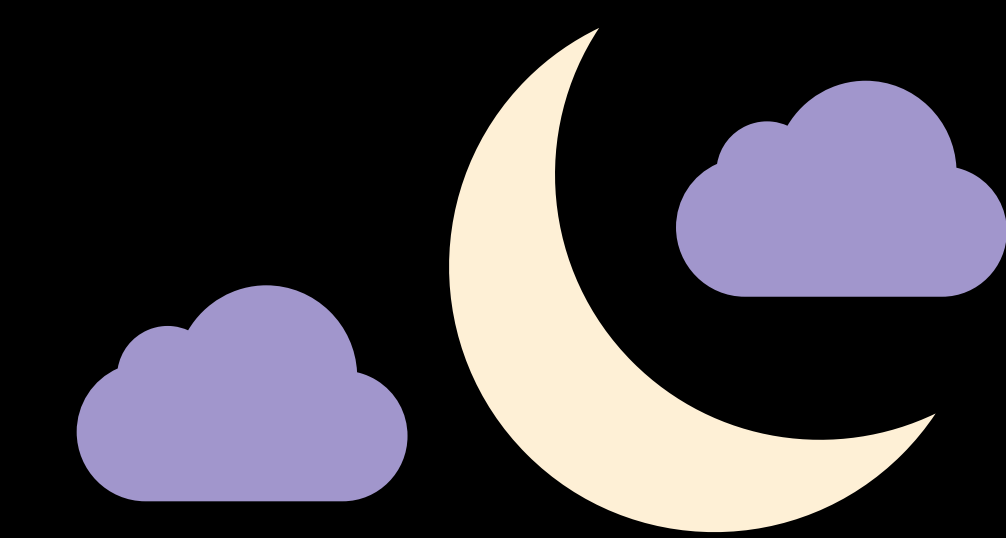
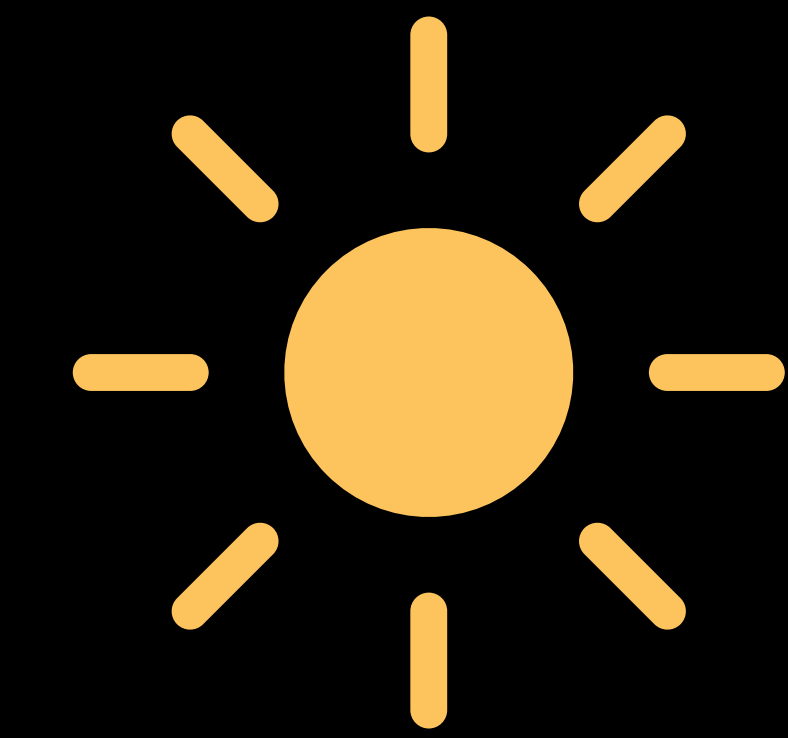
Providing Personalization

Morning

- After user wakes up

Evening

- Before going to bed



```
let morningRelevanceProvider = INDailyRoutineRelevanceProvider(situation: .morning)
```

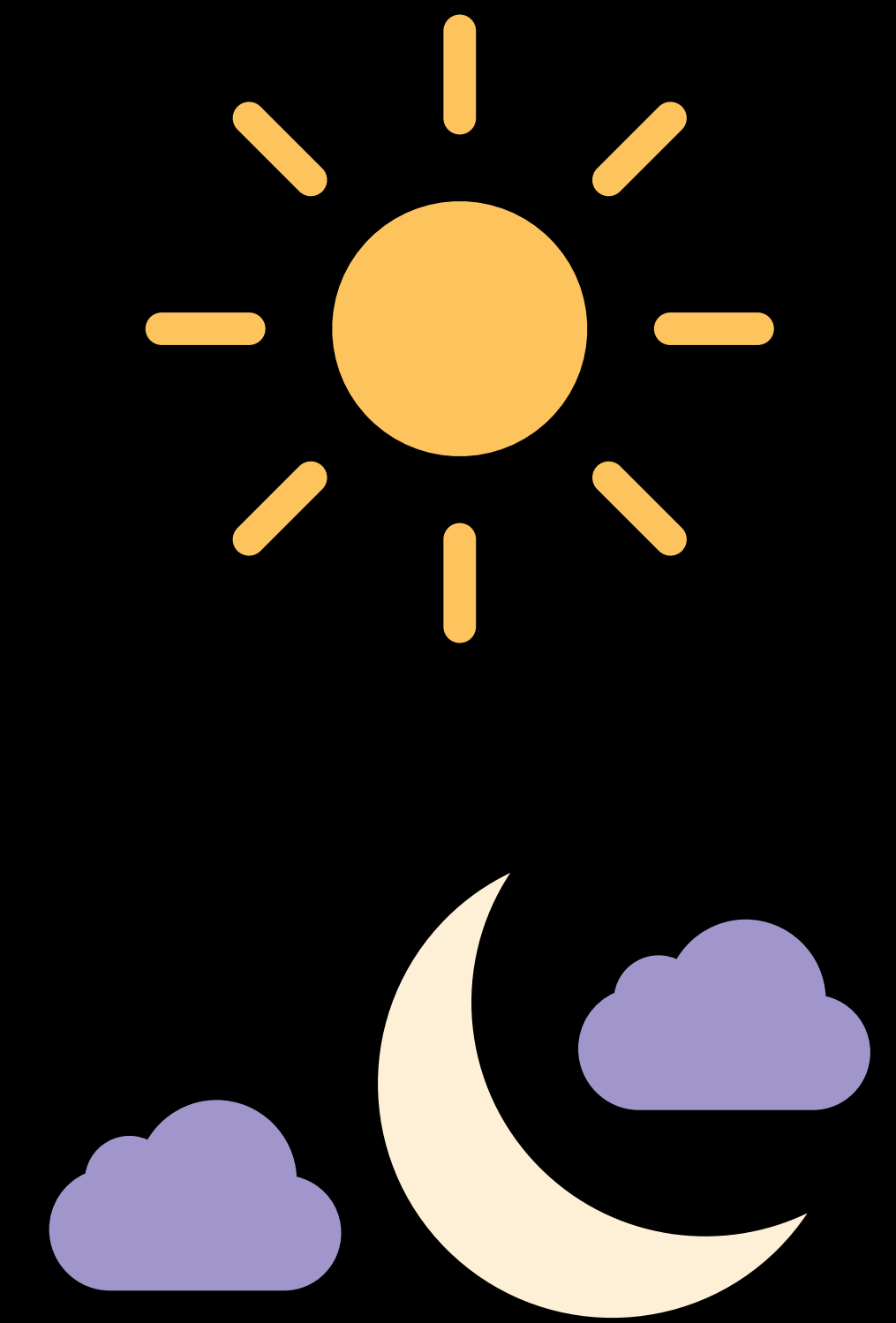
Providing Personalization

Morning

- After user wakes up

Evening

- Before going to bed



```
let morningRelevanceProvider = INDailyRoutineRelevanceProvider(situation: .morning)
```

Similar to `INDateRelevanceProvider`

Providing Personalization

Important locations to user

Providing Personalization

Important locations to user

- Home 

Providing Personalization

Important locations to user

- Home 
- Work 

Providing Personalization

Important locations to user

- Home 
- Work 
- School 



Providing Personalization

Important locations to user

- Home 
- Work 
- School 
- Gym 

Providing Personalization


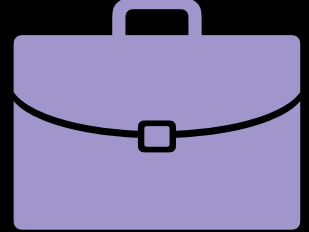
Important locations to user

- Home 
- Work 
- School 
- Gym 

```
let homeRelevanceProvider = INDailyRoutineRelevanceProvider(situation: .home)
```

Providing Personalization

Important locations to user

- Home 
- Work 
- School 
- Gym 

```
let homeRelevanceProvider = INDailyRoutineRelevanceProvider(situation: .home)
```

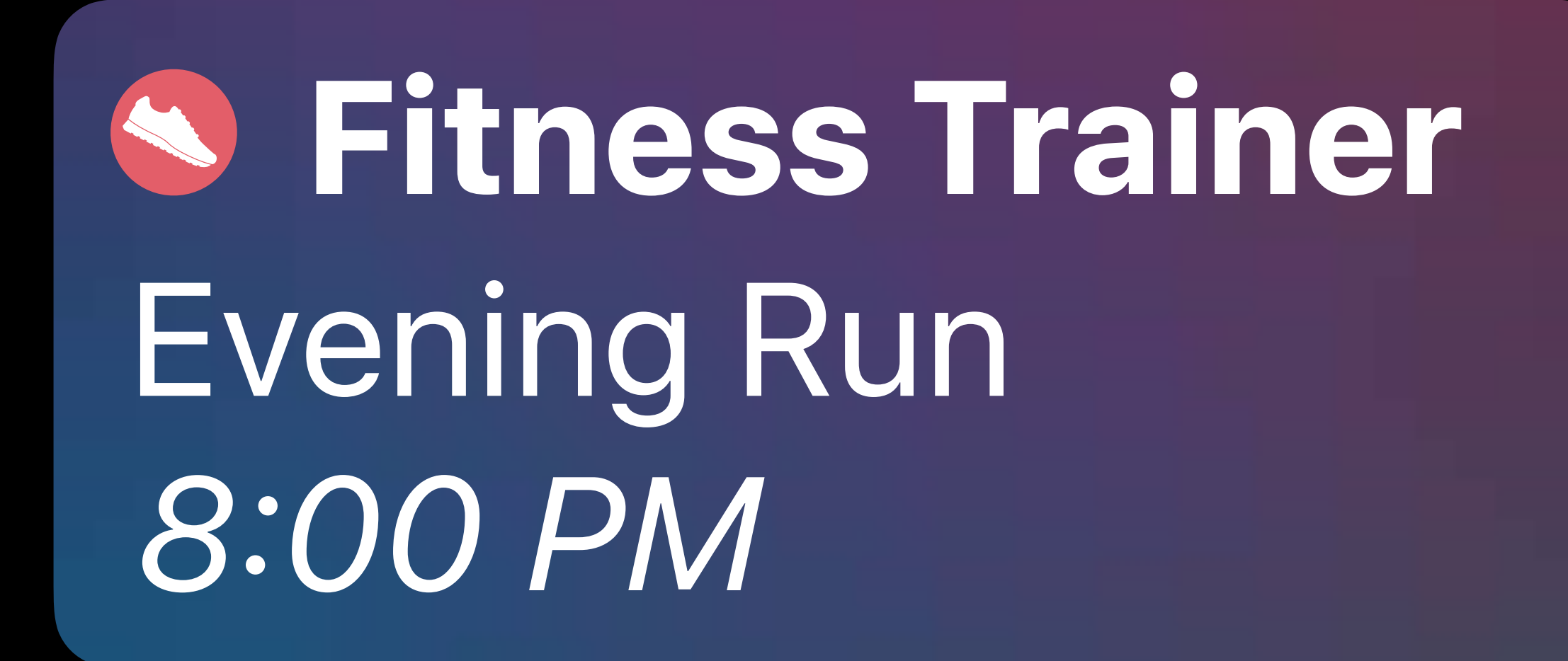
Similar to `INLocationRelevanceProvider`

Making a Great Experience

Making a Great Experience



Glanceable Information



Tappable Actions



WED 6

10:09



 Recipes



Chicken
Parmesan



Space News



New Planets
Found



WED 6



10:09



Recipes



Chicken
Parmesan



Space News



New Planets
Found



Recipes

10:09



Chicken
Parmesan

Ingredients

2 eggs

1 cup bread
crumbs



Recipes

10:09



Chicken
Parmesan

Ingredients

2 eggs

1 cup bread
crumbs



Recipes

10:09



Chicken
Parmesan

Ingredients

2 eggs

1 cup bread
crumbs

Making a Great Experience

Provide most critical information



Glanceable Information

Making a Great Experience

Provide most critical information

Open to location in app



Glanceable Information

Making a Great Experience

Provide most critical information

Open to location in app

Take advantage of background runtime



Glanceable Information

Making a Great Experience

Provide most critical information

Open to location in app

Take advantage of background runtime

Update relevant shortcuts with data



Glanceable Information

Making a Great Experience

Provide most critical information

Open to location in app

Take advantage of background runtime

Update relevant shortcuts with data

Relevance providers for timely information

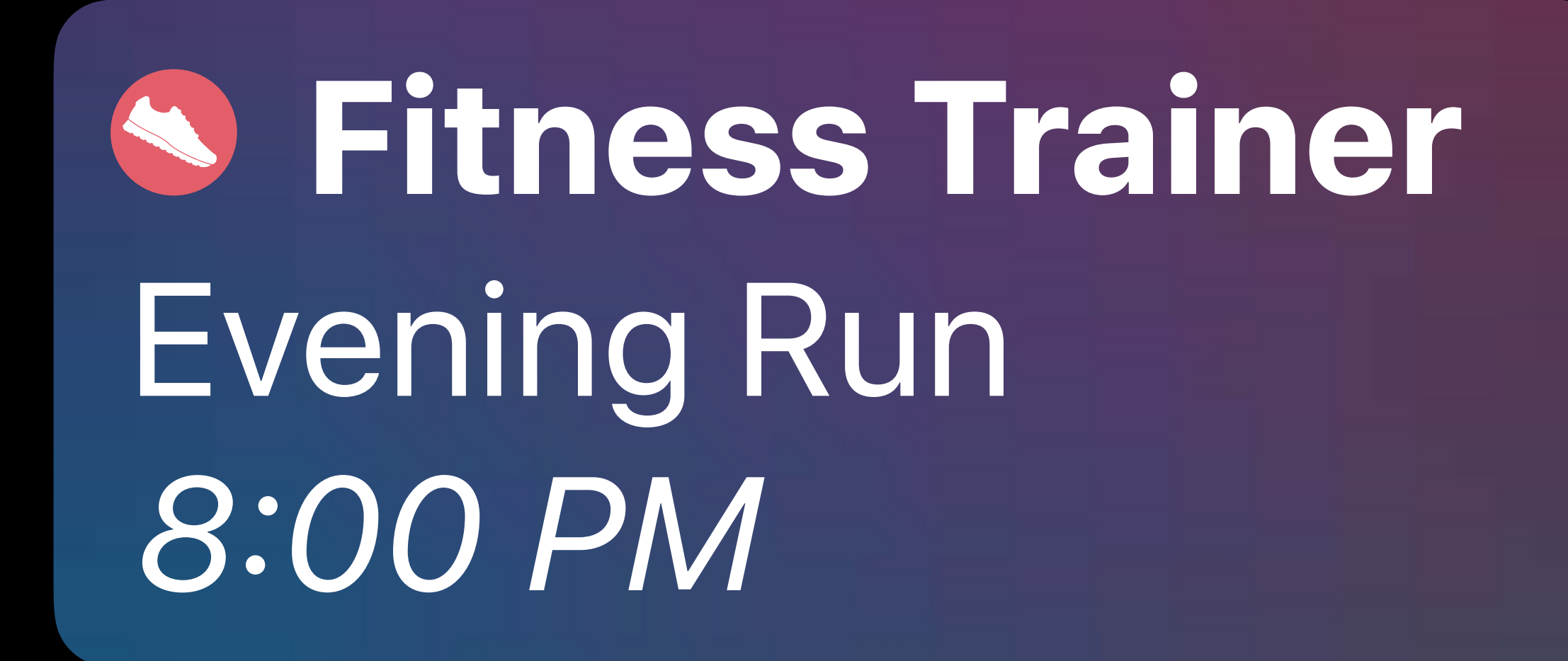


Glanceable Information

Making a Great Experience



Glanceable Information



Tappable Actions



WED 6



10:09



Fitness Trainer

Evening Run

8:00 PM



Space News



New Planets
Found



WED 6



10:09



Fitness Trainer

Evening Run

8:00 PM



Space News



New Planets

Found



WED 6



10:09



Fitness Trainer

Evening Run

8:00 PM



Space News



New Planets

Found



Fitness Trai...

Start Evening Run

Take a 5 mile run
around Golden
Gate Park

Confirm



Fitness Trai...

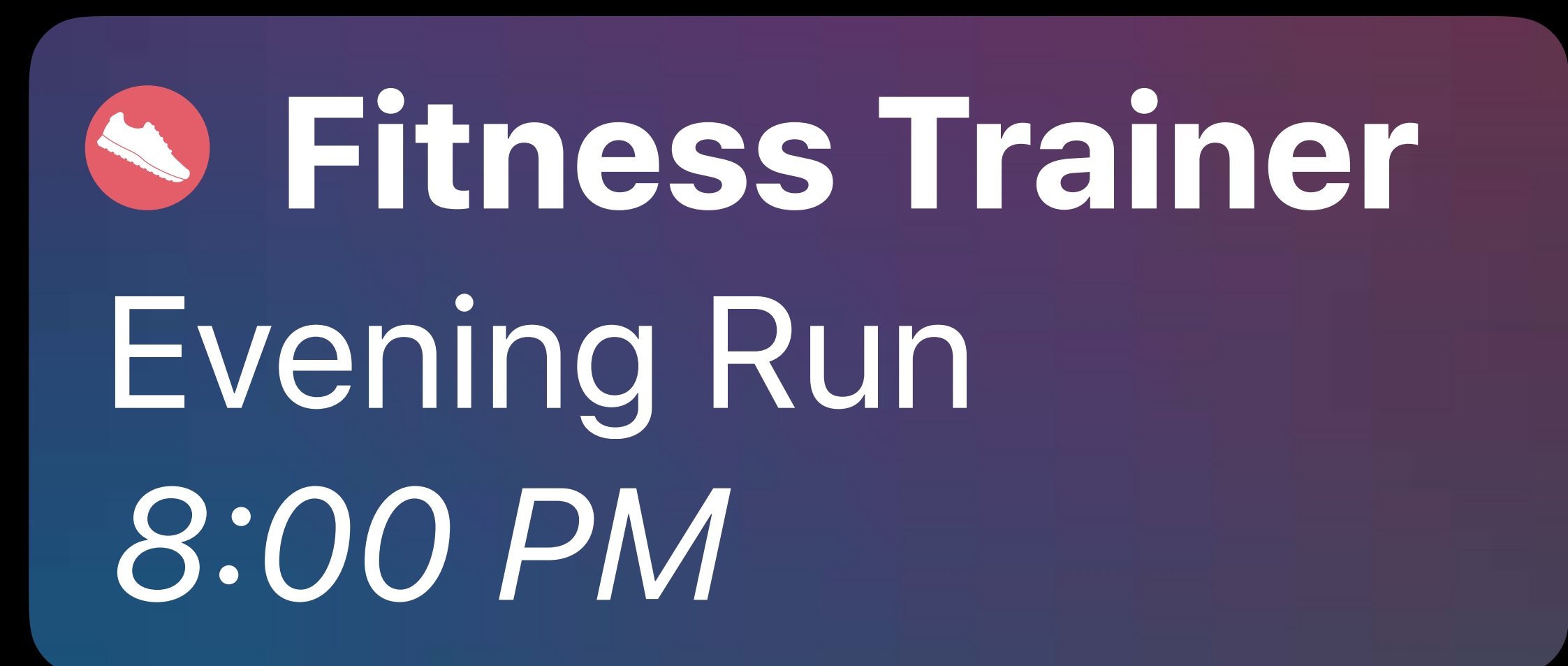
Start Evening Run

Take a 5 mile run
around Golden
Gate Park

Confirm

Making a Great Experience

Run action in Intents extension

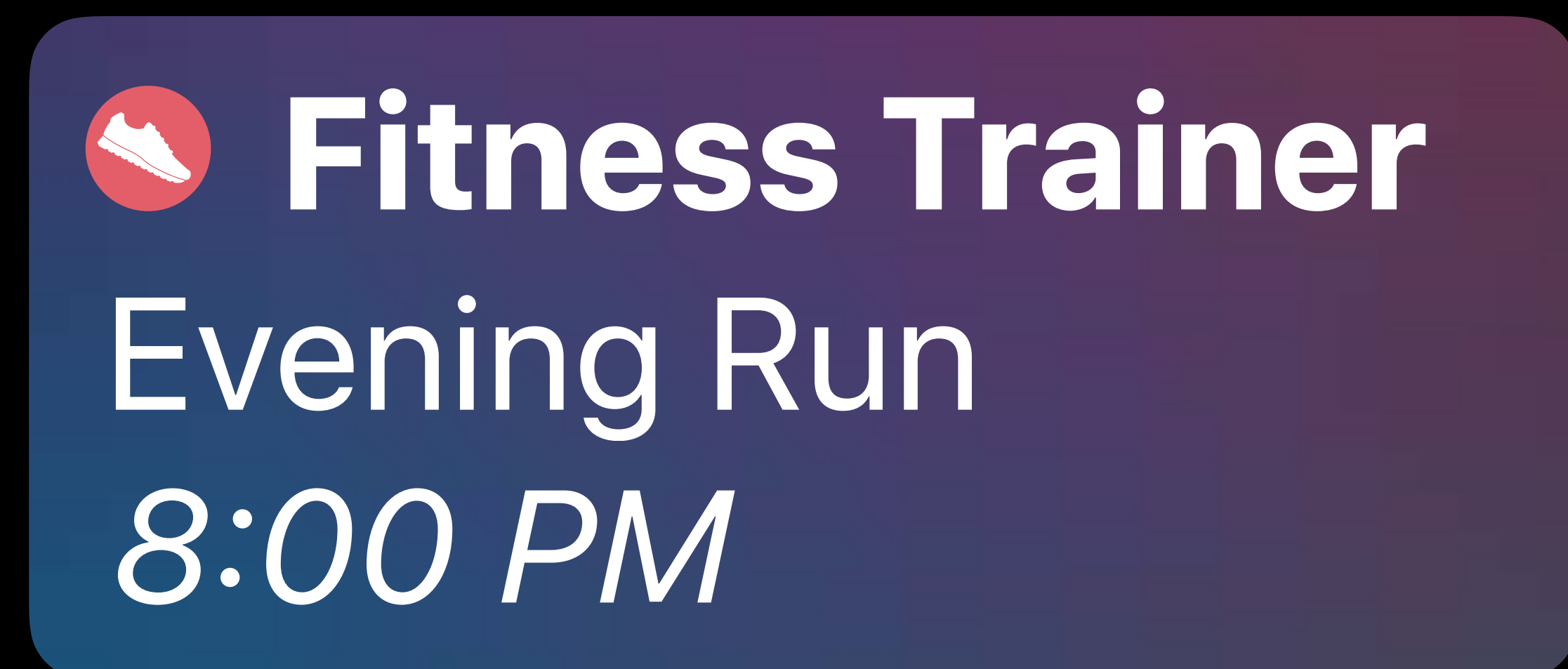


Tappable Actions

Making a Great Experience

Run action in Intents extension

Standard system confirmation UI



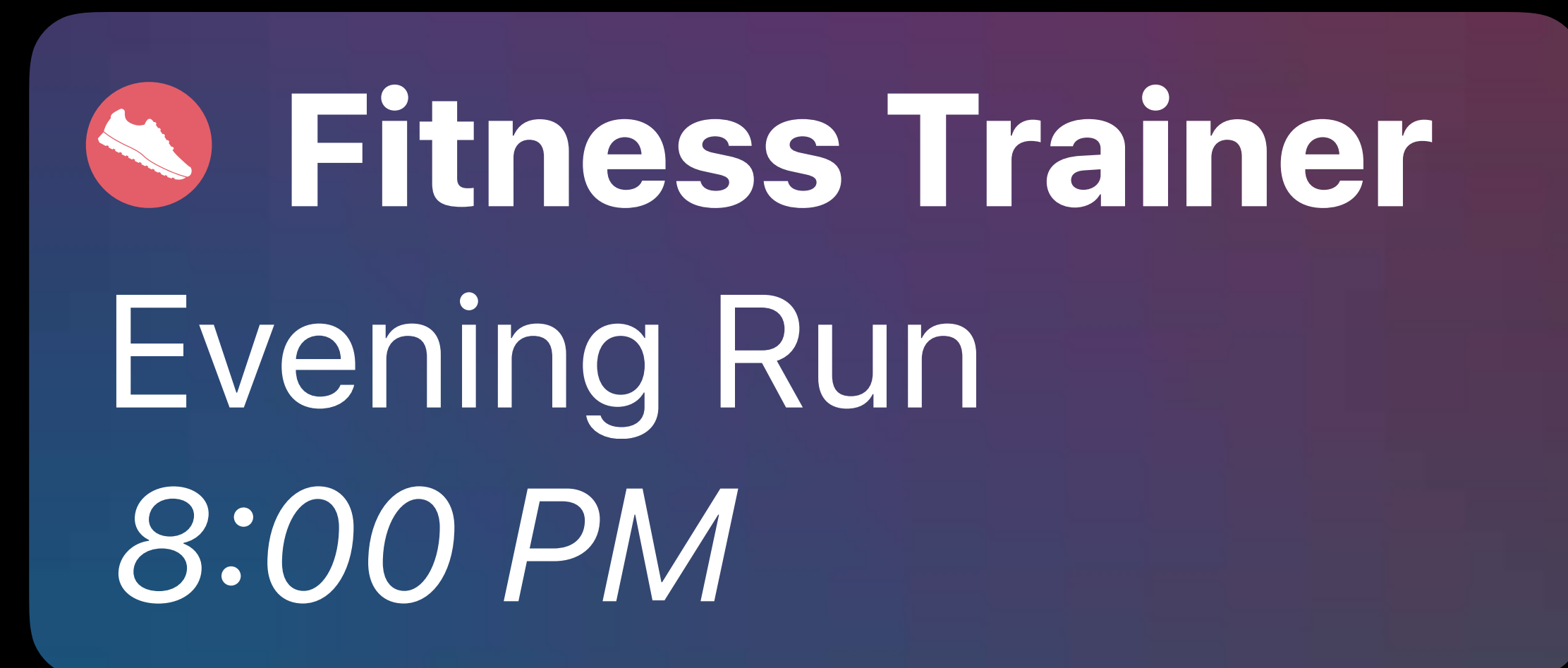
Tappable Actions

Making a Great Experience

Run action in Intents extension

Standard system confirmation UI

Fully specified Intents



Tappable Actions

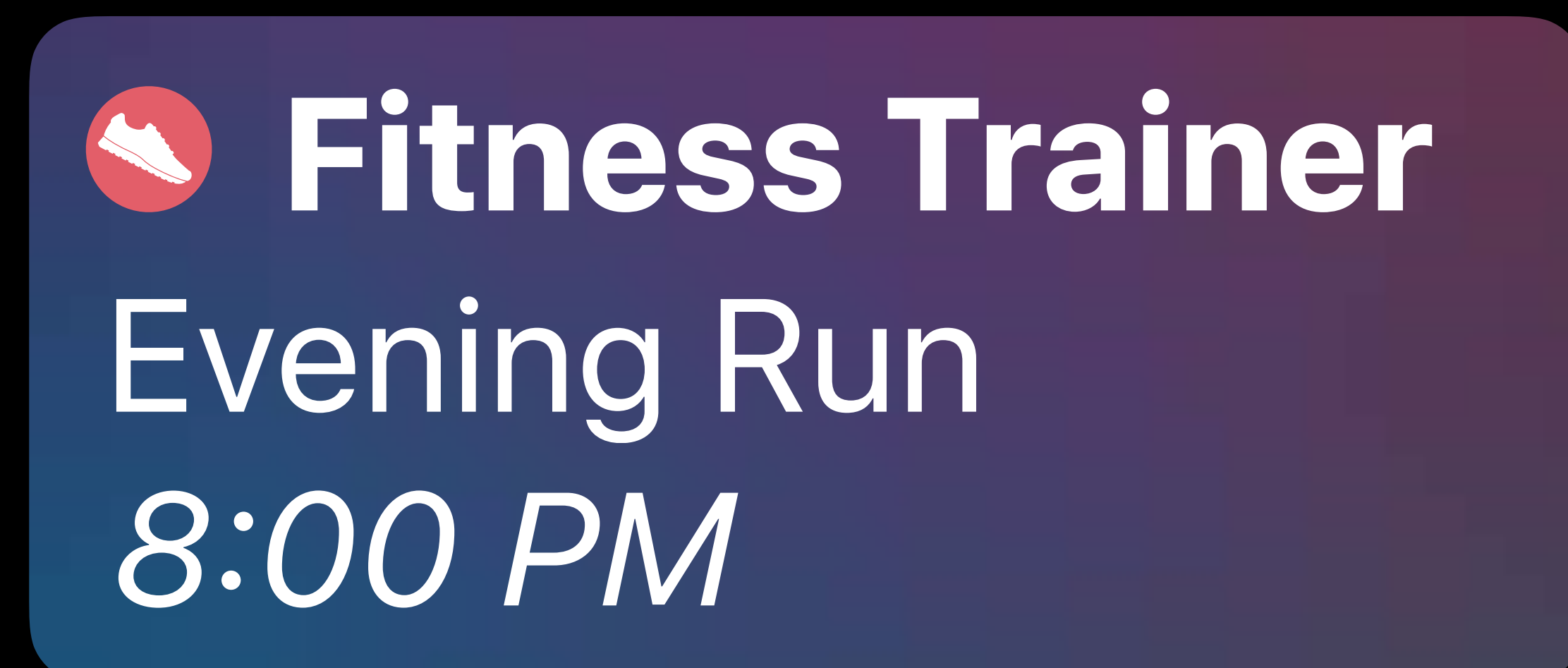
Making a Great Experience

Run action in Intents extension

Standard system confirmation UI

Fully specified Intents

Provide actions commonly performed



Tappable Actions

Making a Great Experience

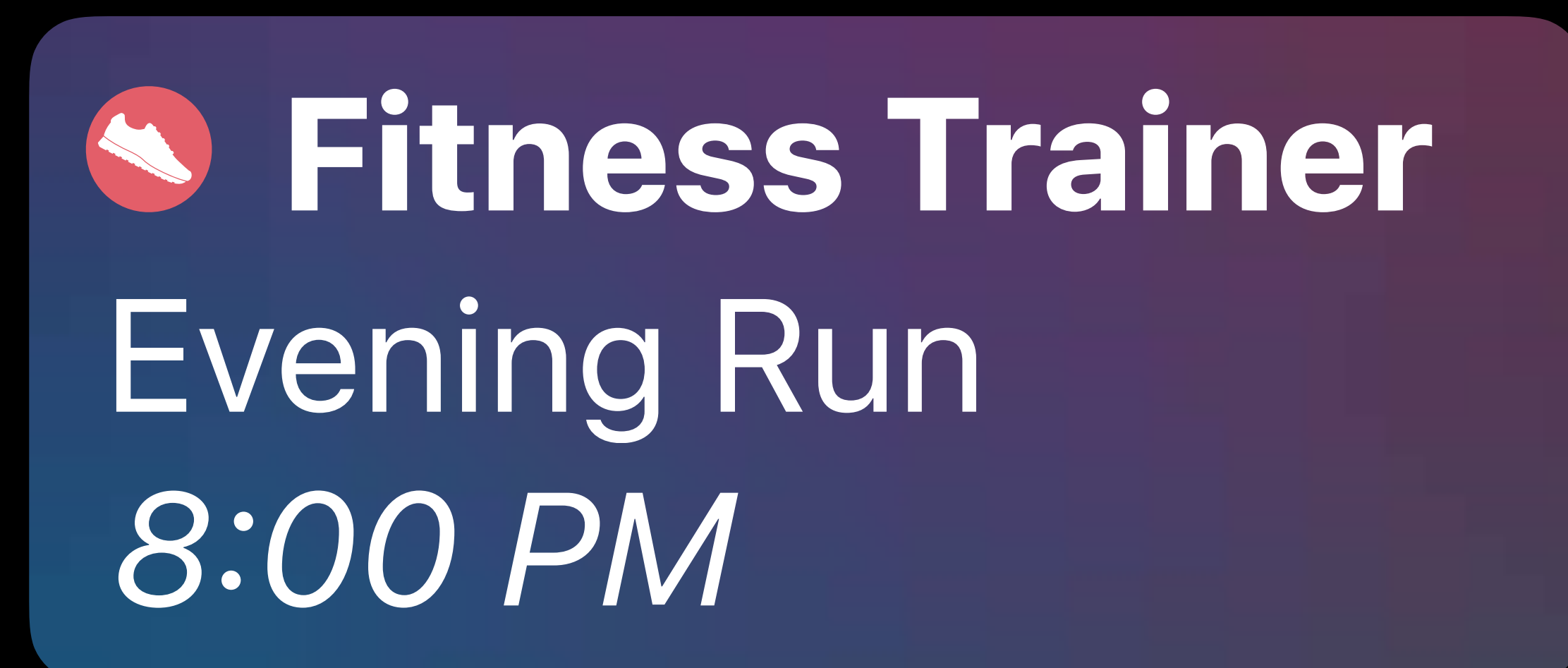
Run action in Intents extension

Standard system confirmation UI

Fully specified Intents

Provide actions commonly performed

Create relevant shortcuts often



Tappable Actions

Summary

Relevant shortcuts on Siri watch face

Summary

Relevant shortcuts on Siri watch face

Provide great content

Summary

Relevant shortcuts on Siri watch face

Provide great content

Richest experience with watchOS app

More Information

<https://developer.apple.com/wwdc18/217>

WatchKit, ClockKit, and Siri Shortcuts Lab

Technology Lab 7

Wednesday 3:00PM

 **WWDC18**