System Frameworks #WWDC16

Introduction to Notifications

Session 707

Kritarth Jain iOS Notifications Engineer
Julien Barlerin iOS Frameworks QA Engineer

Agenda

Agenda

Notifications Overview

Agenda

NEW

Notifications Overview

User Notifications Framework

Agenda

NEW

Notifications Overview

User Notifications Framework

- Registration
- Content
- Scheduling
- Management
- Actions

Agenda

NEW

Notifications Overview

User Notifications Framework

- Registration
- Content
- Scheduling
- Management
- Actions

Service Extensions

Agenda

There's a lot more…

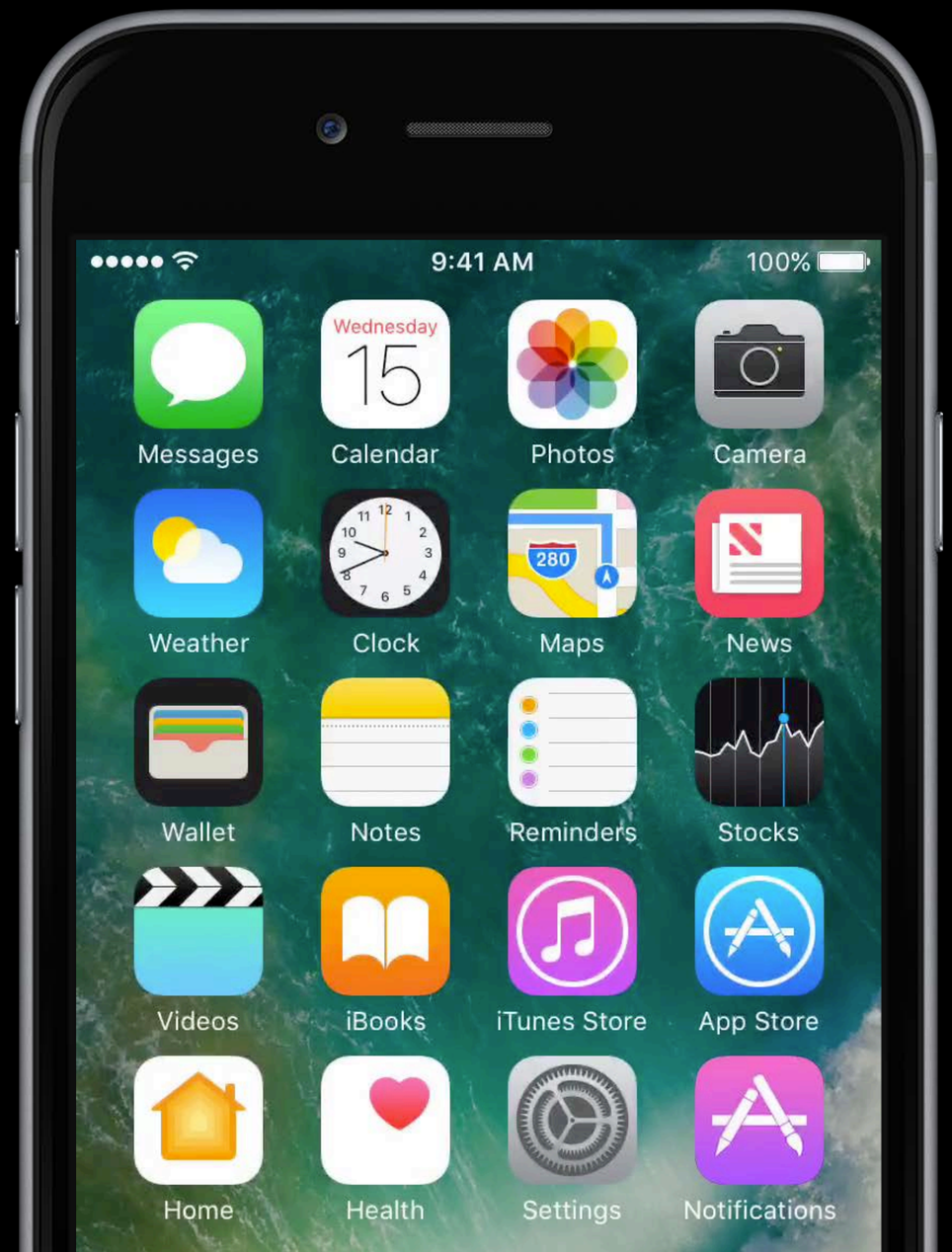
Advanced Notifications	Pacific Heights	Wednesday 10:00AM

User Notifications


100%
9:41
Wednesday, June 15

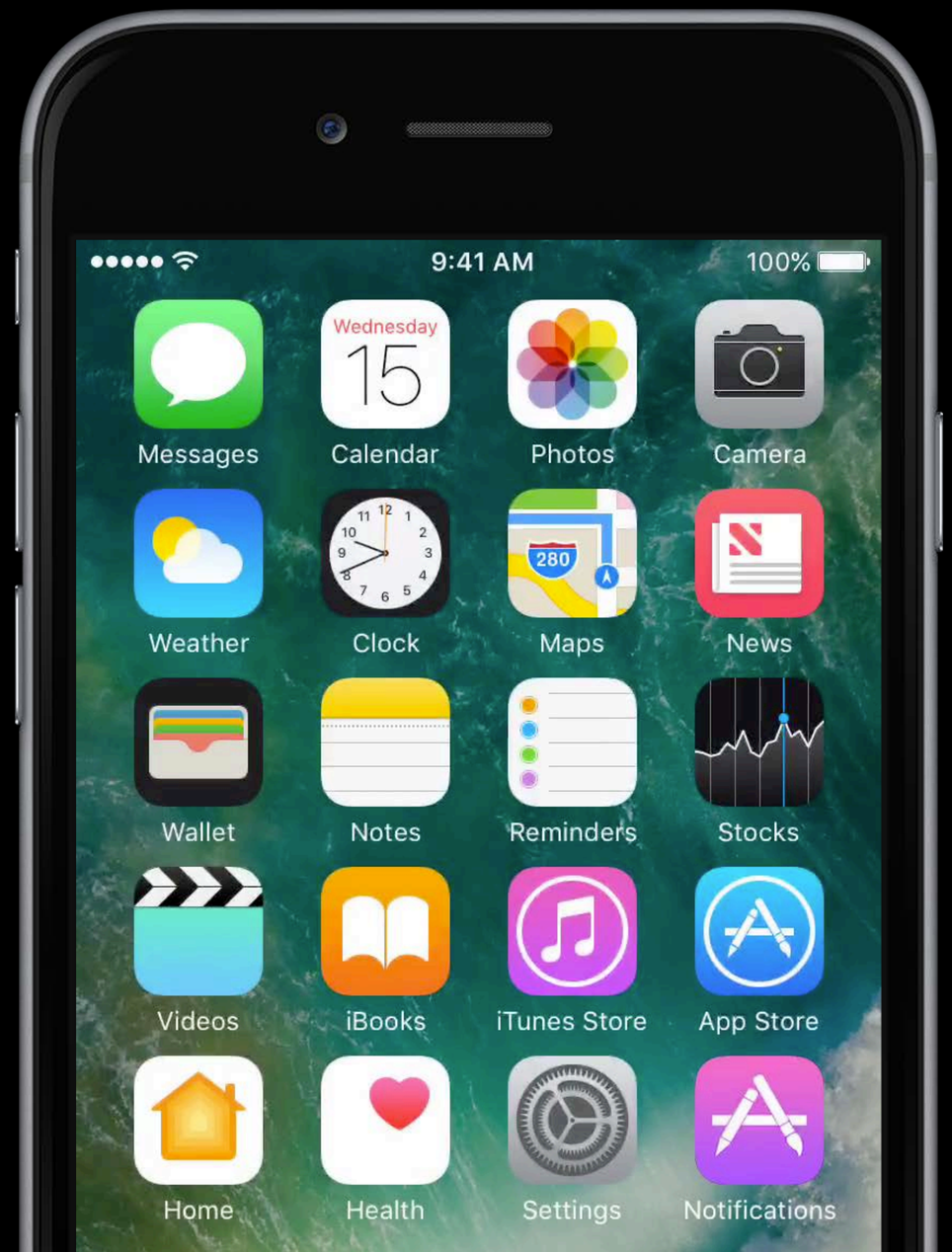

100%
9:41
Wednesday, June 15

User Notifications



User Notifications

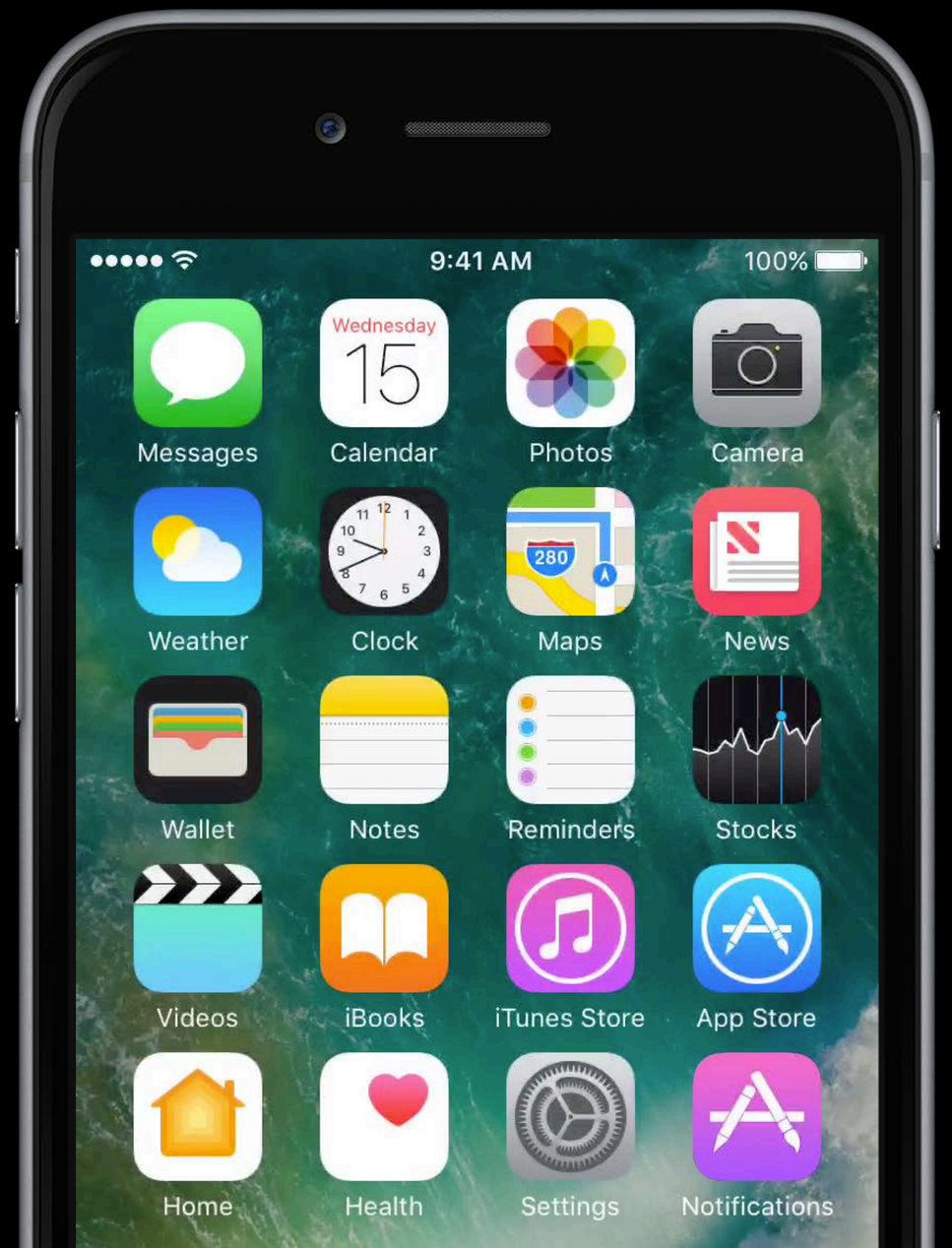
Visual alert



User Notifications

Visual alert

Sound and vibration

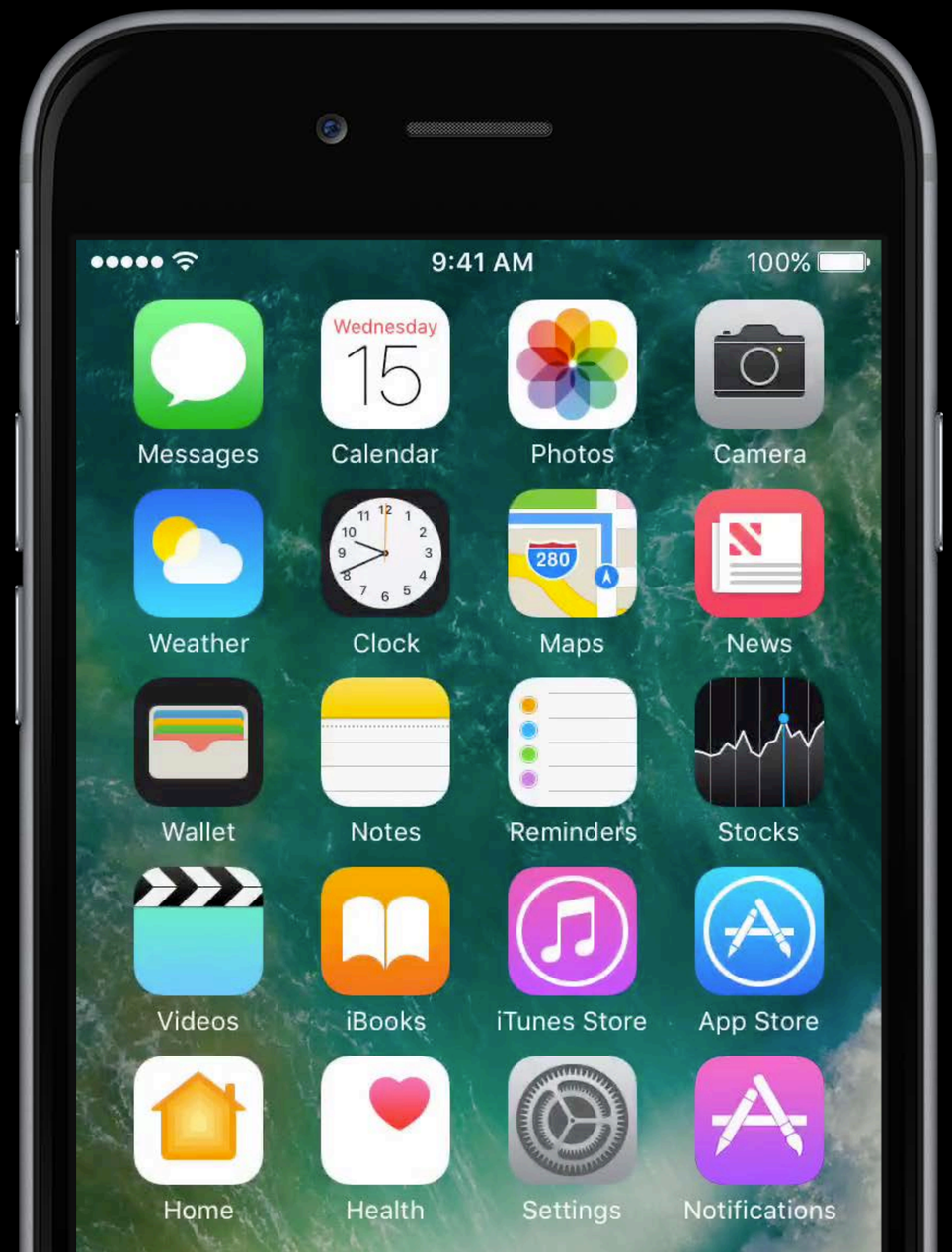


User Notifications

Visual alert

Sound and vibration

App icon badging



User Notifications

User Notifications

User Notifications

User Notifications

Types of Notifications

Types of Notifications

Local Notifications

Remote Notifications

Local Notifications

Local Notifications

Application on device

Local Notifications

Application on device

Local Notifications

Application on device

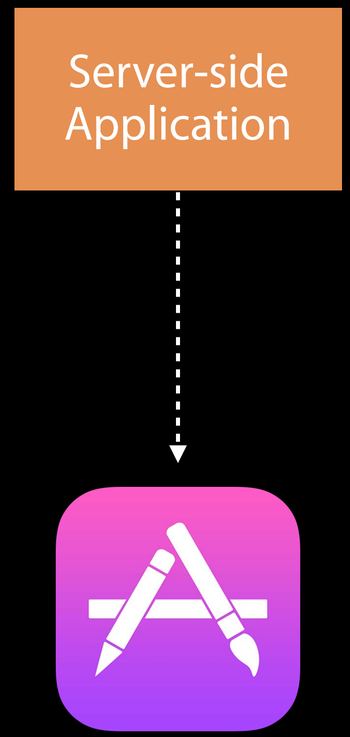
Examples

- Task reminder alerts
- Calendar alerts
- Location-based triggers

Remote Notifications

Remote Notifications

Server-side application component

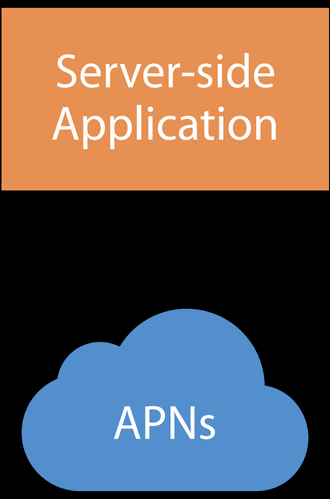


Remote Notifications

Server-side
Application

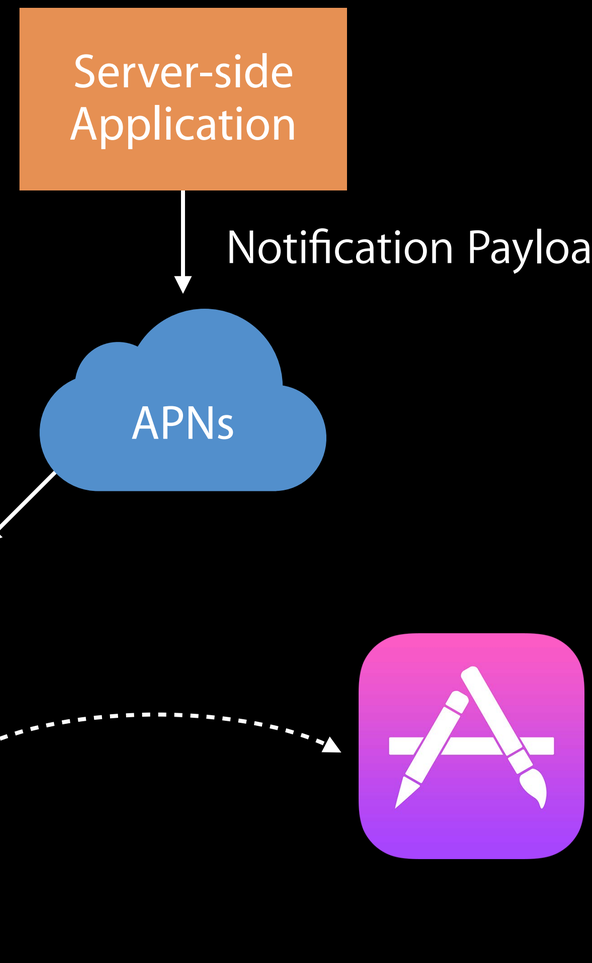
Remote Notifications

Apple Push Notification Service (APNs)



Remote Notifications

Apple Push Notification Service (APNs)

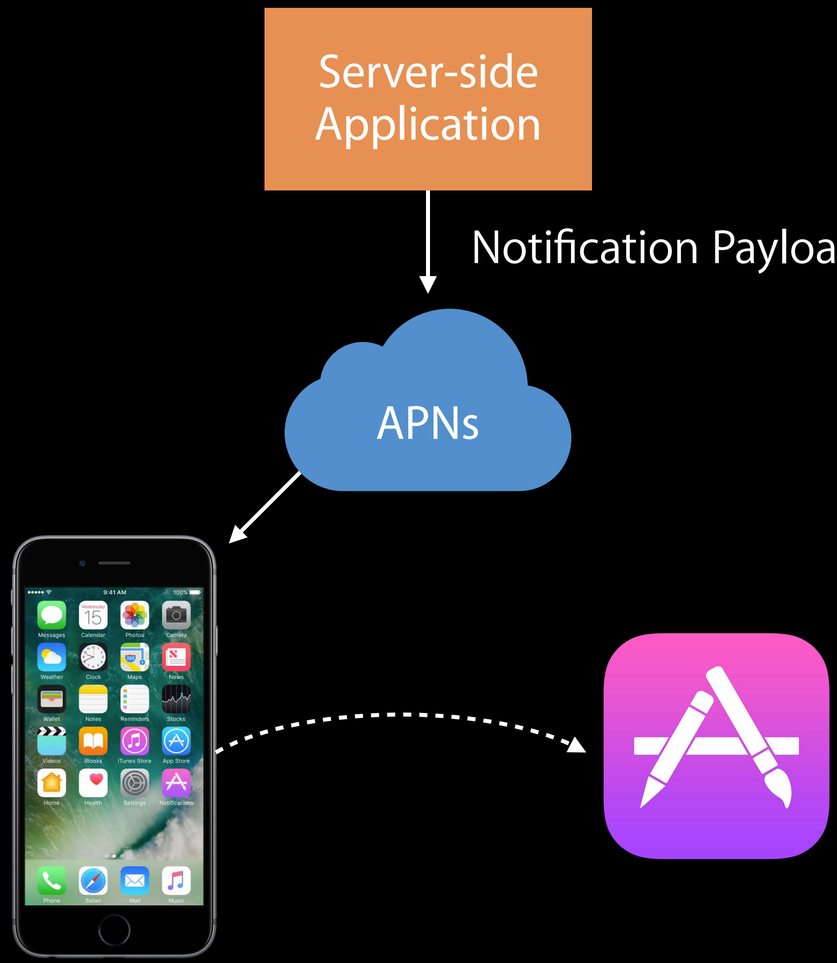


Remote Notifications

Apple Push Notification Service (APNs)

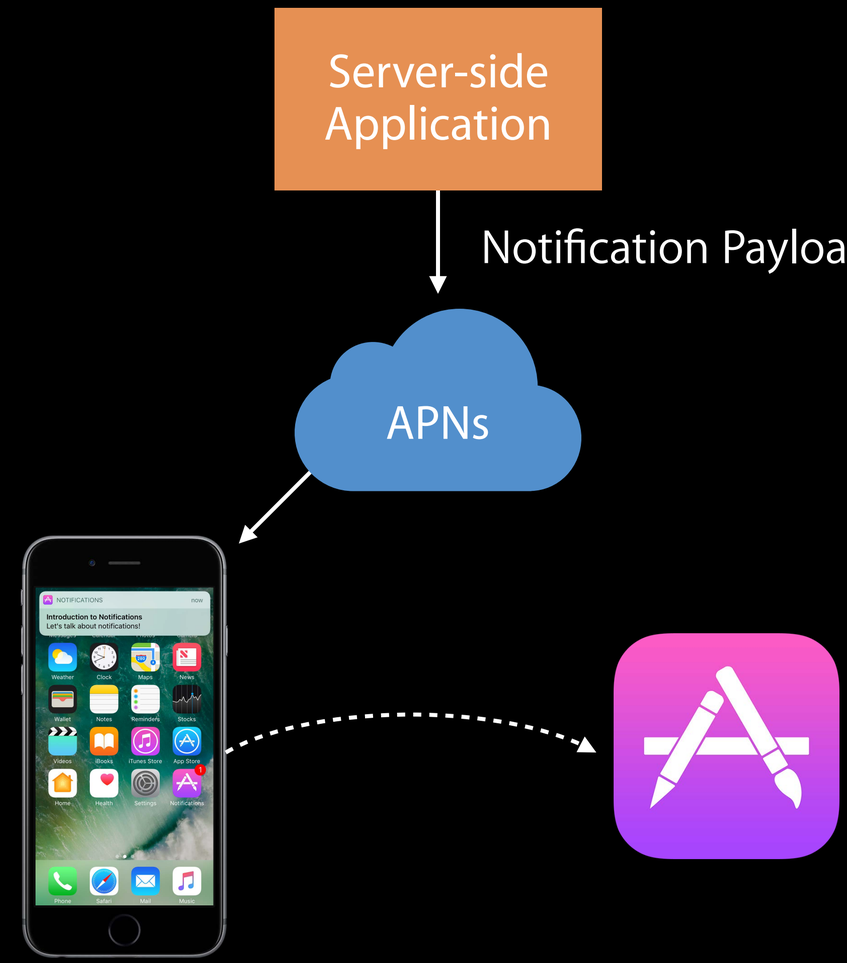
Examples

- News alerts
- Instant messaging
- Sports updates



Remote Notifications

Components

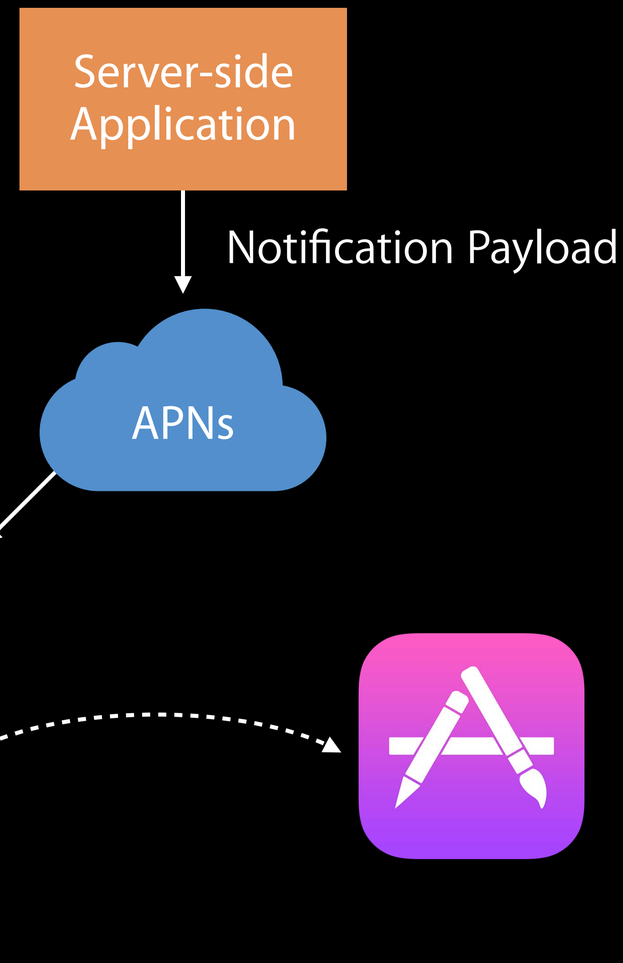


Remote Notifications

Components

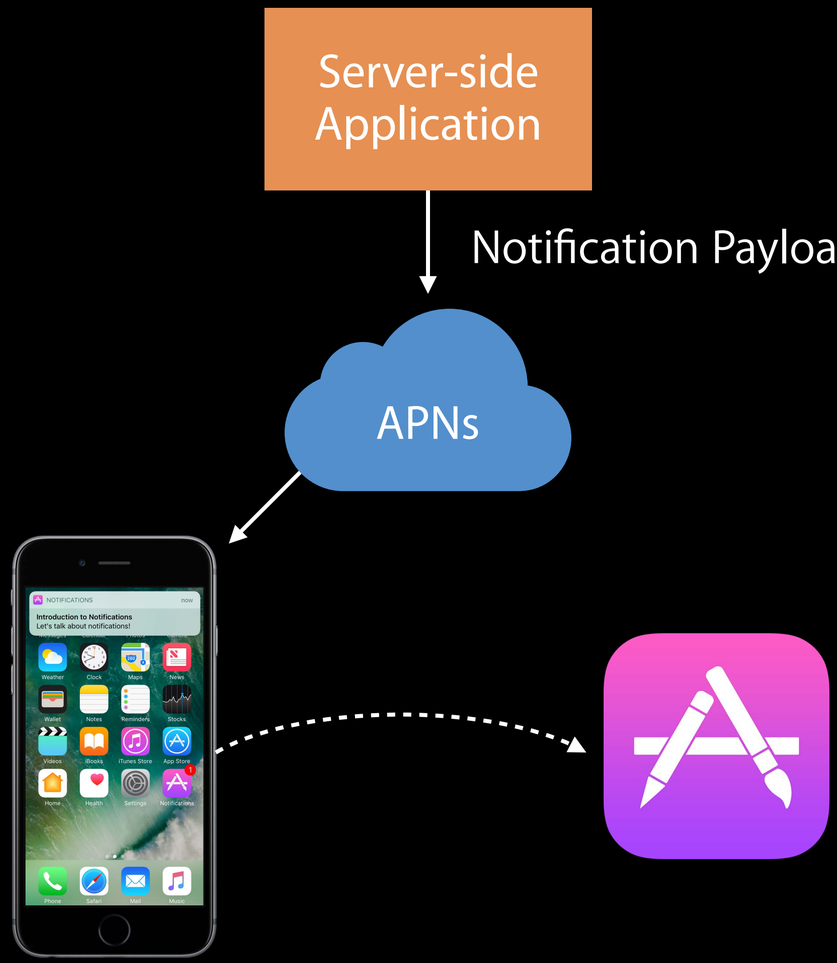
User-facing

Silent update



Remote Notifications

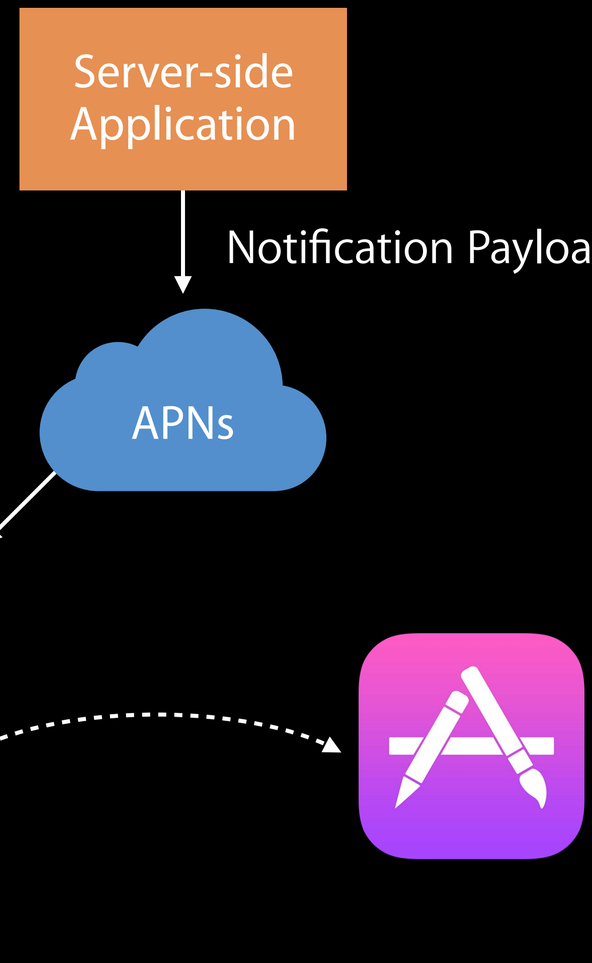
User-facing



Remote Notifications

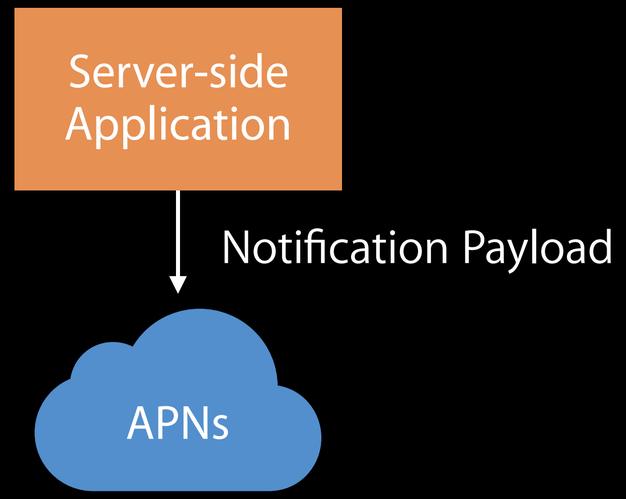
User-facing

Displayed to the user on device



Remote Notifications

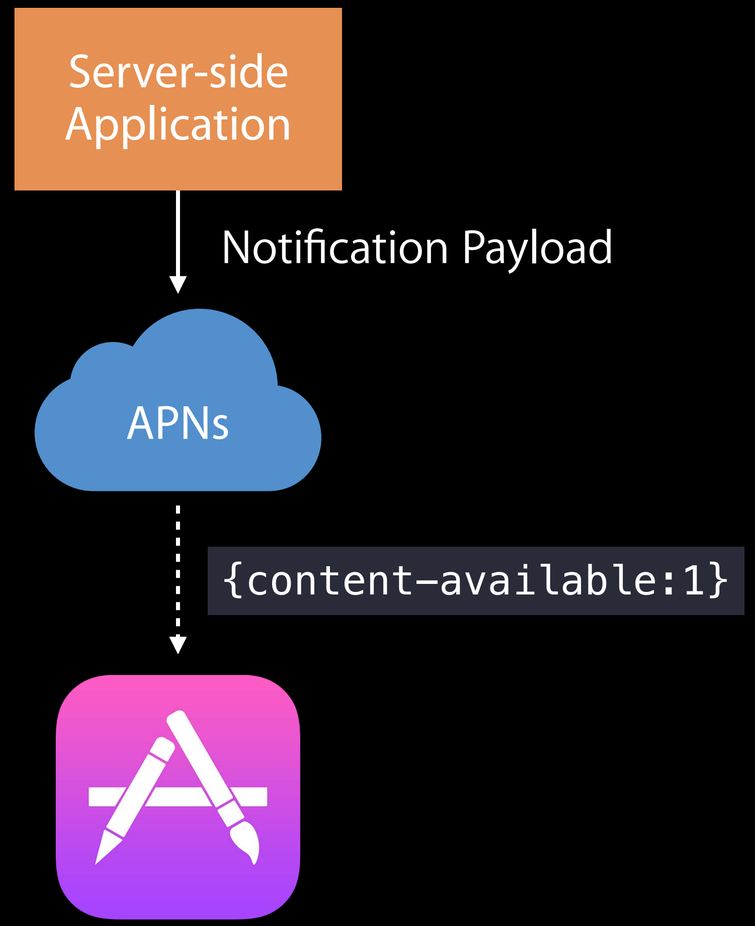
Silent update



Remote Notifications

Silent update

Background App Refresh



Existing API Overview

Existing API Overview

Existing API Overview

UIApplication

- Registration
- Scheduling

Existing API Overview

UIApplication

- Registration
- Scheduling

Different callbacks for local and remote notifications

Existing API Overview

UIApplication

- Registration
- Scheduling

Different callbacks for local and remote notifications

Limited control after notifications are scheduled

Existing API Overview

UIApplication

- Registration
- Scheduling

Different callbacks for local and remote notifications

Limited control after notifications are scheduled

Different support across multiple platforms

NEW

User Notifications Framework

User Notifications Framework

NEW

Overview

User Notifications Framework

NEW

Overview

Familiar API with feature parity

User Notifications Framework

NEW

Overview

Familiar API with feature parity

Expanded content

User Notifications Framework

NEW

Overview

Familiar API with feature parity

Expanded content

Same code path for local and remote notification handling

User Notifications Framework

NEW

Overview

Familiar API with feature parity

Expanded content

Same code path for local and remote notification handling

Simplified delegate methods

User Notifications Framework

NEW

Overview

Familiar API with feature parity

Expanded content

Same code path for local and remote notification handling

Simplified delegate methods

Better notification management

User Notifications Framework

NEW

Overview

Familiar API with feature parity

Expanded content

Same code path for local and remote notification handling

Simplified delegate methods

Better notification management

In-app presentation option

User Notifications Framework

NEW

Overview

Familiar API with feature parity

Expanded content

Same code path for local and remote notification handling

Simplified delegate methods

Better notification management

In-app presentation option

Schedule and handle notifications in extensions

User Notifications Framework

NEW

Overview

Familiar API with feature parity

Expanded content

Same code path for local and remote notification handling

Simplified delegate methods

Better notification management

In-app presentation option

Schedule and handle notifications in extensions

Notification Extensions!

User Notifications Framework

NEW

Multi-Platform Support

User Notifications Framework

Multi-Platform Support

NEW

Single Notifications API across platforms

iOS watchOS tvOS

User Notifications Framework

iOS

NEW

iOS

User Notifications Framework

iOS

NEW

Full support to schedule and manage notifications

iOS

User Notifications Framework

watchOS

User Notifications Framework

watchOS

Existing support for forwarded notifications

watchOS

User Notifications Framework

watchOS

NEW

Existing support for forwarded notifications

Local Notifications on the watch

watchOS

User Notifications Framework

watchOS

NEW

Existing support for forwarded notifications

Local Notifications on the watch

Examples

- “Workout goal met”
- “Timers on the watch”

watchOS

User Notifications Framework

watchOS

NEW

Existing support for forwarded notifications

Local Notifications on the watch

Examples

- “Workout goal met”
- “Timers on the watch”

watchOS

Quick Interaction Techniques for watchOS	Presidio	Wednesday 11:00AM

User Notifications Framework

tvOS

NEW

tvOS

User Notifications Framework

tvOS

NEW

Support to badge app icons

tvOS

User Notifications Framework

tvOS

NEW

Support to badge app icons

- Examples
- “3 unwatched episodes”
- “Pending user turn in game”

tvOS

Notification Delivery

Notification Delivery

Notification Delivery

Notification Delivery

Notification Delivery

Registration

Registration

Registration

Registration

User Authorization

- Banners
- Sound alerts
- Badging

Registration

User Authorization

- Banners
- Sound alerts
- Badging

Needed for local and remote notifications

Registration

User Authorization

- Banners
- Sound alerts
- Badging

Needed for local and remote notifications

```
UNUserNotificationCenter.current().requestAuthorization([.alert, .sound, .badge])
    { (granted, error) in // ... }
```

Registration

Notification settings

Registration

Notification settings

Configurable in Settings per app

Registration

Notification settings

NEW

Configurable in Settings per app

Access to user-defined settings

```
UNUserNotificationCenter.current().getNotificationSettings { (settings) in // ... }
```

Token Registration

Token Registration

Remote Notifications

Token Registration

Remote Notifications

Existing API

```
UIApplication.shared().registerForRemoteNotifications()
```

Server-side Application

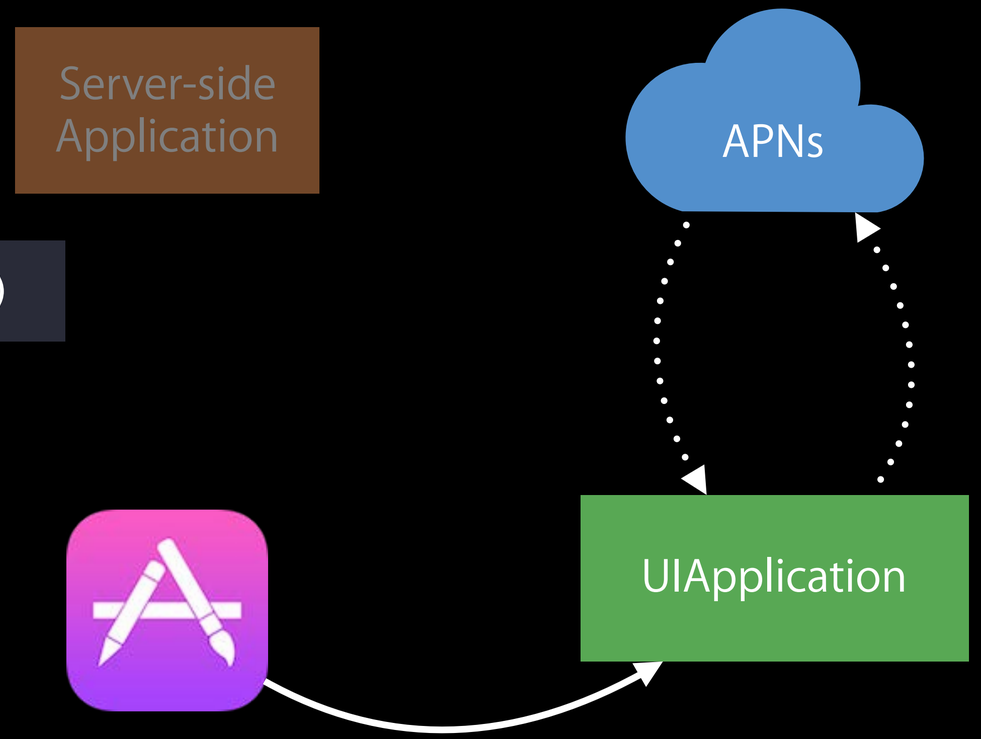
Token Registration

Remote Notifications

Existing API

```
UIApplication.shared().registerForRemoteNotifications()
```

Need network connection to talk to APNs



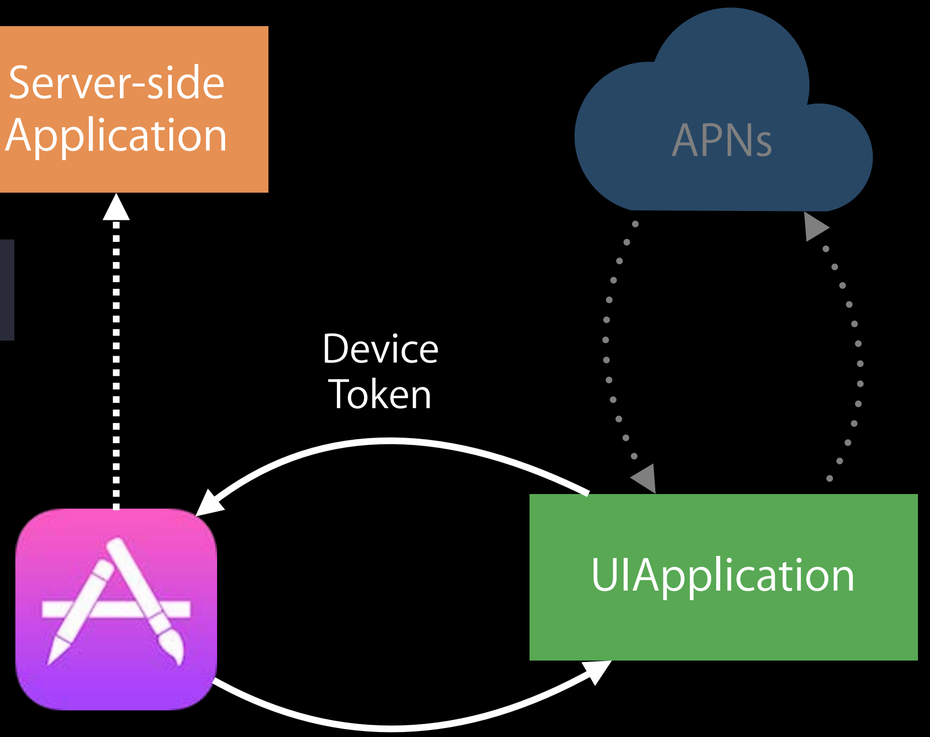
Token Registration

Remote Notifications

Existing API

```
UIApplication.shared().registerForRemoteNotifications()
```

Need network connection to talk to APNs



Token Registration

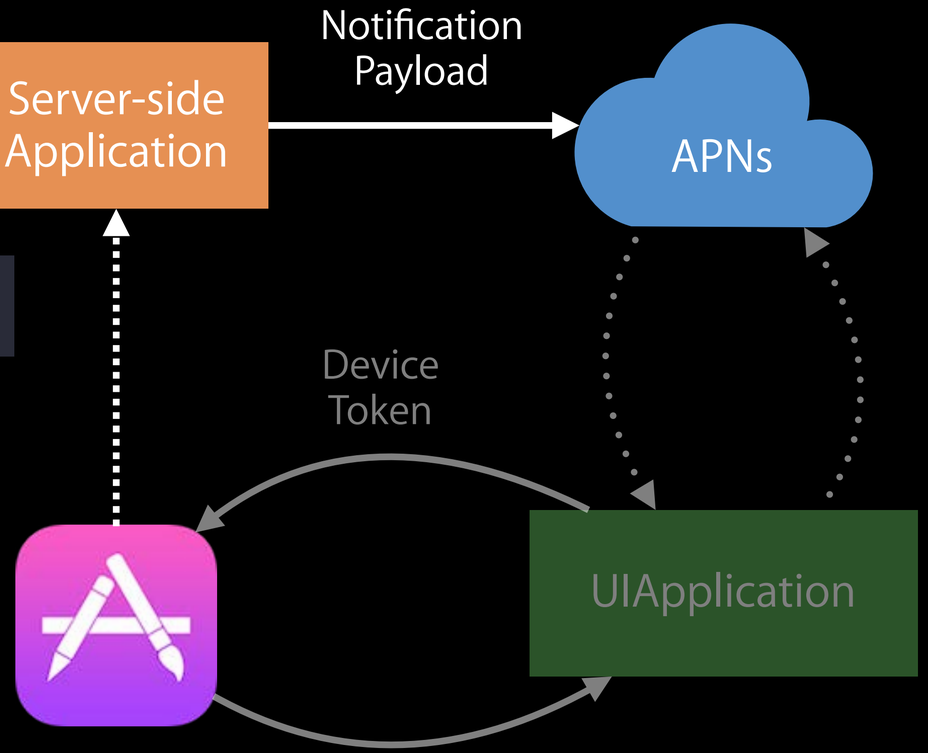
Remote Notifications

Existing API

```
UIApplication.shared().registerForRemoteNotifications()
```

Need network connection to talk to APNs

Token must be included in remote payload

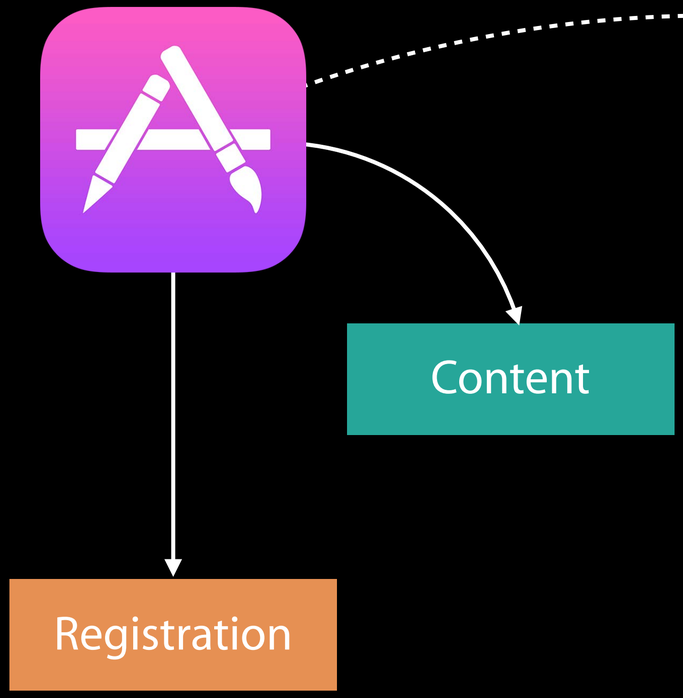


Notification Delivery

Registration

Notification Delivery

Content



Content


9:41 AM
100%
Messages
Wednesday
15
Calendar
Photos
Camera
Weather
Clock
280
Maps
News
Wallet
Notes
Reminders
Stocks

Content


9:41 AM
100%
Messages
Wednesday
15
Calendar
Photos
Camera
Weather
Clock
280
Maps
News
Wallet
Notes
Reminders
Stocks

Content

NEW

Title

Content

NEW

Title

Subtitle

Content

NEW

Subtitle

Title

Body

Content

Local Notification

```
let content = UNMutableNotificationContent()
content.title = "Introduction to Notifications"
content.subtitle = "Session 707"
content.body = "Woah! These new notifications look amazing! Don’t you agree?"
content.badge = 1
```

Content

Remote Notification

```
{
    "aps" : {
        "alert" : {
            "title" : "Introduction to Notifications",
            "subtitle" : "Session 707",
            "body" : "Woah! These new notifications look amazing! Don’t you agree?"
        },
        "badge" : 1
    },
}
```

Content


9:41 AM
100%
Wednesday
15
Messages
Calendar
Photos
Camera
Weather
Clock
Maps
News
Wallet
Notes
Reminders
Stocks
Videos
iBooks
iTunes Store
App Store

Content

Media Attachments

NEW

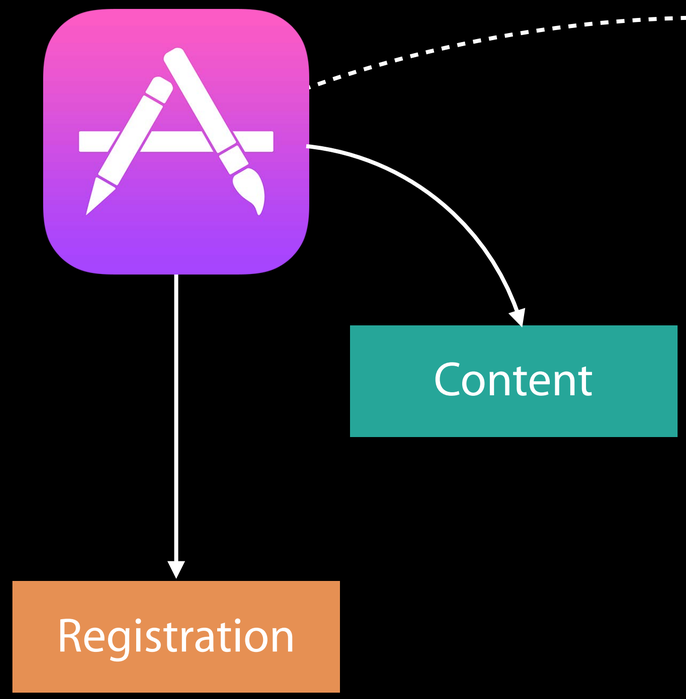
Media Attachment

Content

Media Attachments

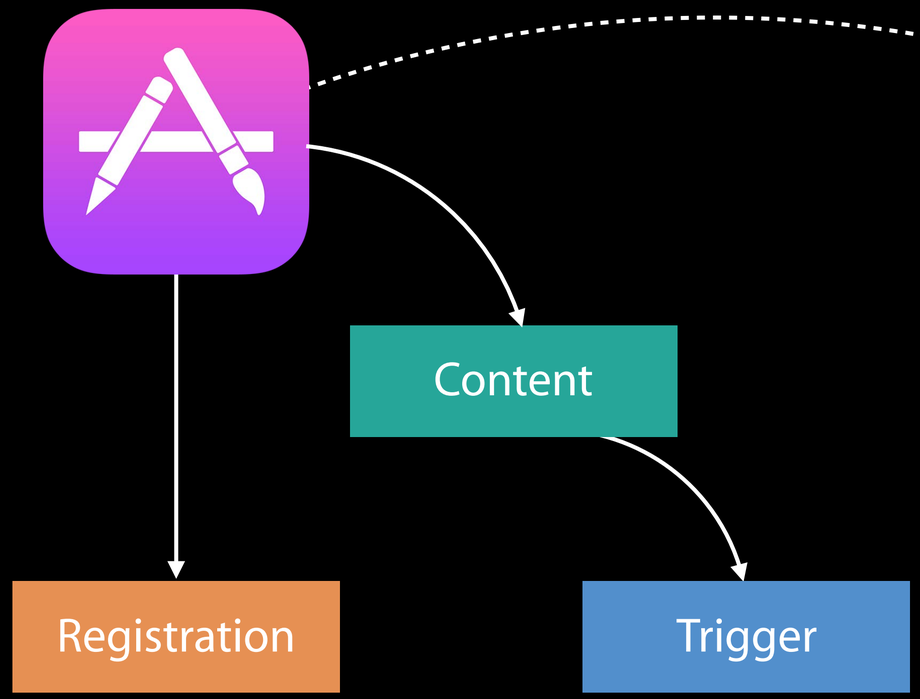
More on Media Attachments

Notification Delivery



Notification Delivery

Trigger



Triggers

Triggers

Push

Time Interval

Calendar

Location

Triggers

Push

Push

Triggers

Push

Remote Notifications

Push

Triggers

Time Interval

Time Interval

Triggers

Time Interval

Examples

- “In 2 minutes from now”

```
UNTimeIntervalNotificationTrigger(timeInterval: 120,
                                  repeats: false)
```

Time Interval

Triggers

Time Interval

Examples

- "In 2 minutes from now"

```
UNTimeIntervalNotificationTrigger(timeInterval: 120,
                                  repeats: false)
```

- "Repeat every hour starting now"

```
UNTimeIntervalNotificationTrigger(timeInterval: 3600,
                                  repeats: true)
```

Time Interval

Triggers

Calendar

Calendar

Triggers

Calendar

Examples

- “8:00am tomorrow morning”
- “Repeat every Monday at 6:00pm”

```
let dateComponents = DateComponents()
// Configure dateComponents
UNCalendarNotificationTrigger(dateMatching: dateComponents,
                              repeats: false)
```

Calendar

Triggers

Location

Location

Triggers

Location

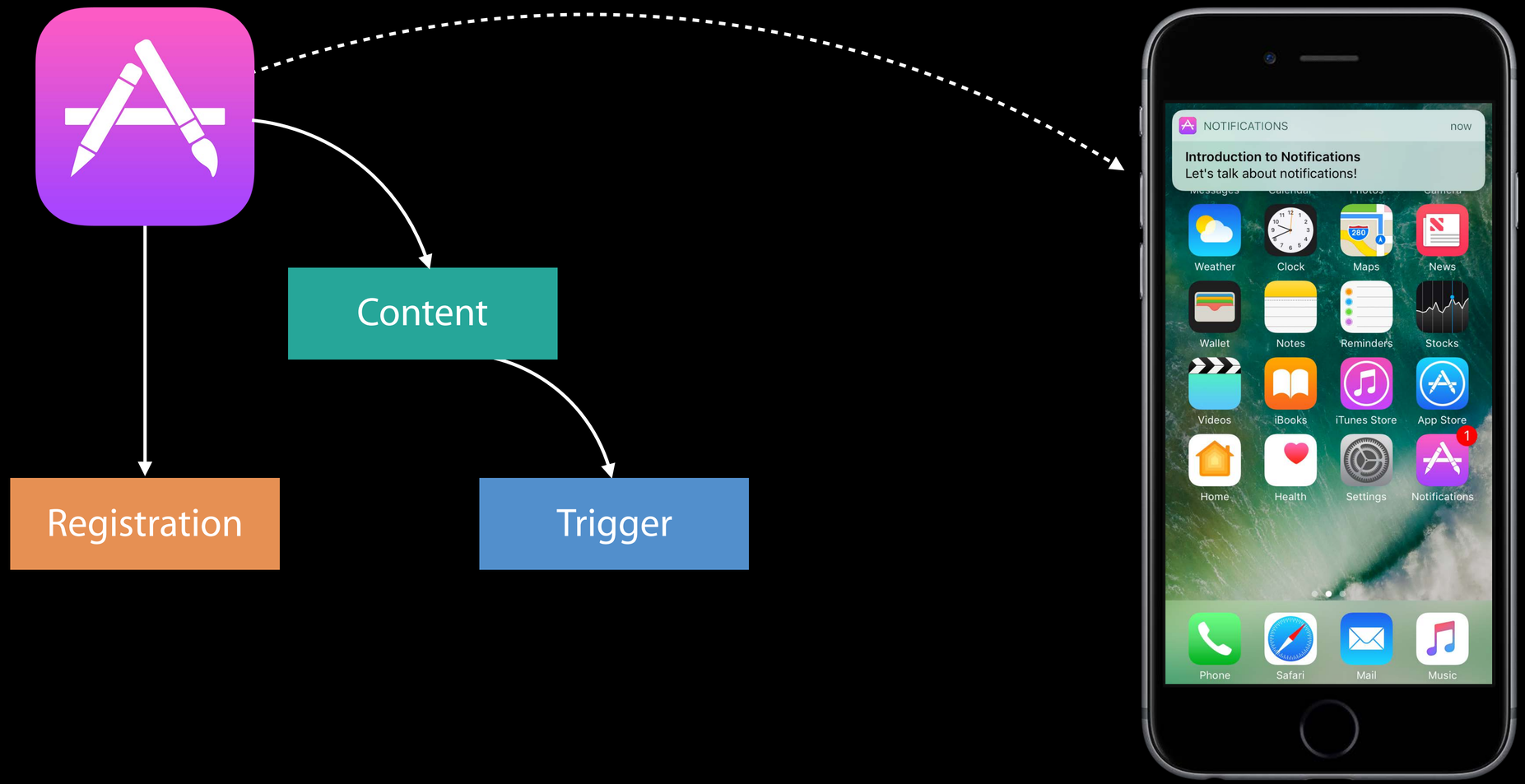
Examples

- “When leaving home”
- “When arriving in proximity of grocery store”

```
let region = CLRegion()
// Configure region
UNLocationNotificationTrigger(region: region,
                              repeats: false);
```

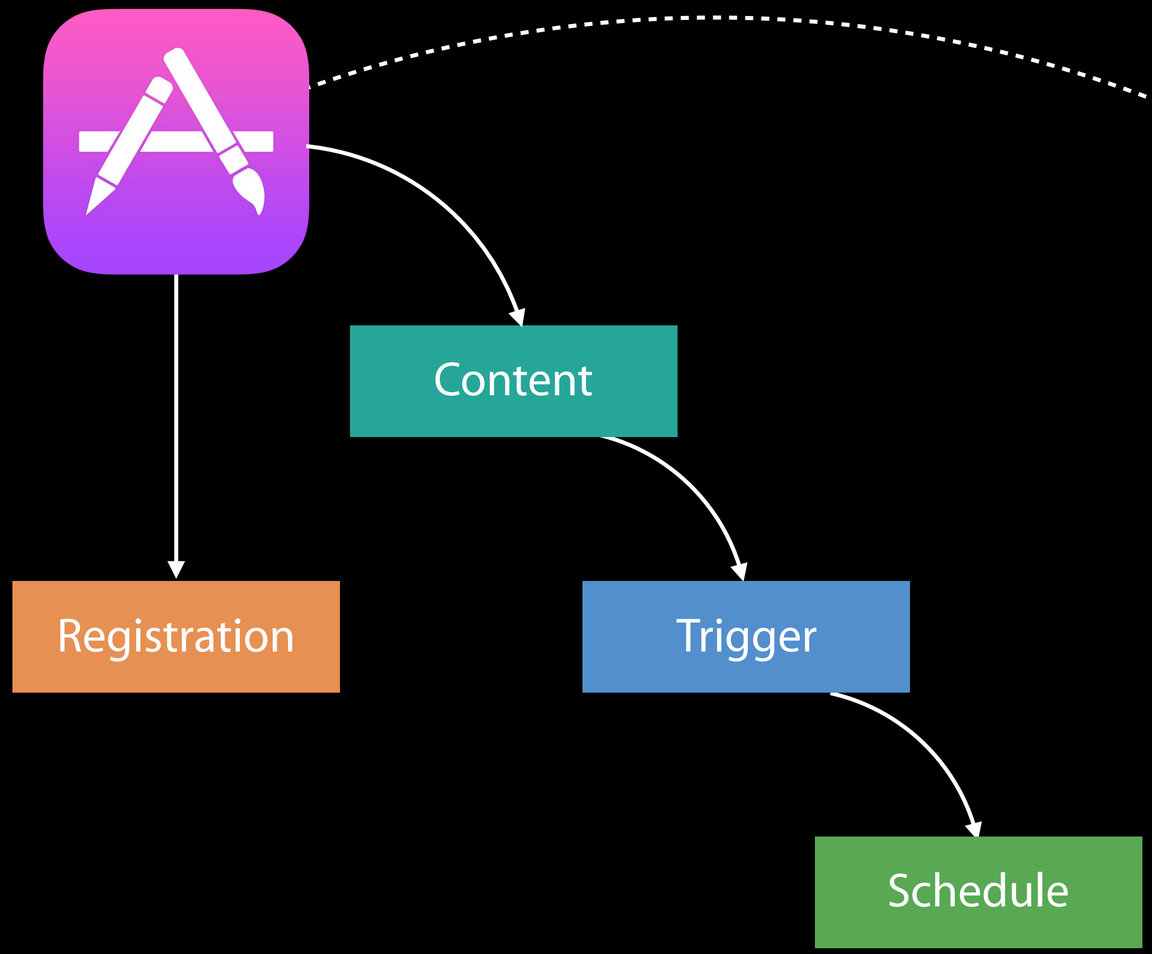
Location

Notification Delivery



Notification Delivery

Schedule



Schedule

Local Notifications

Schedule

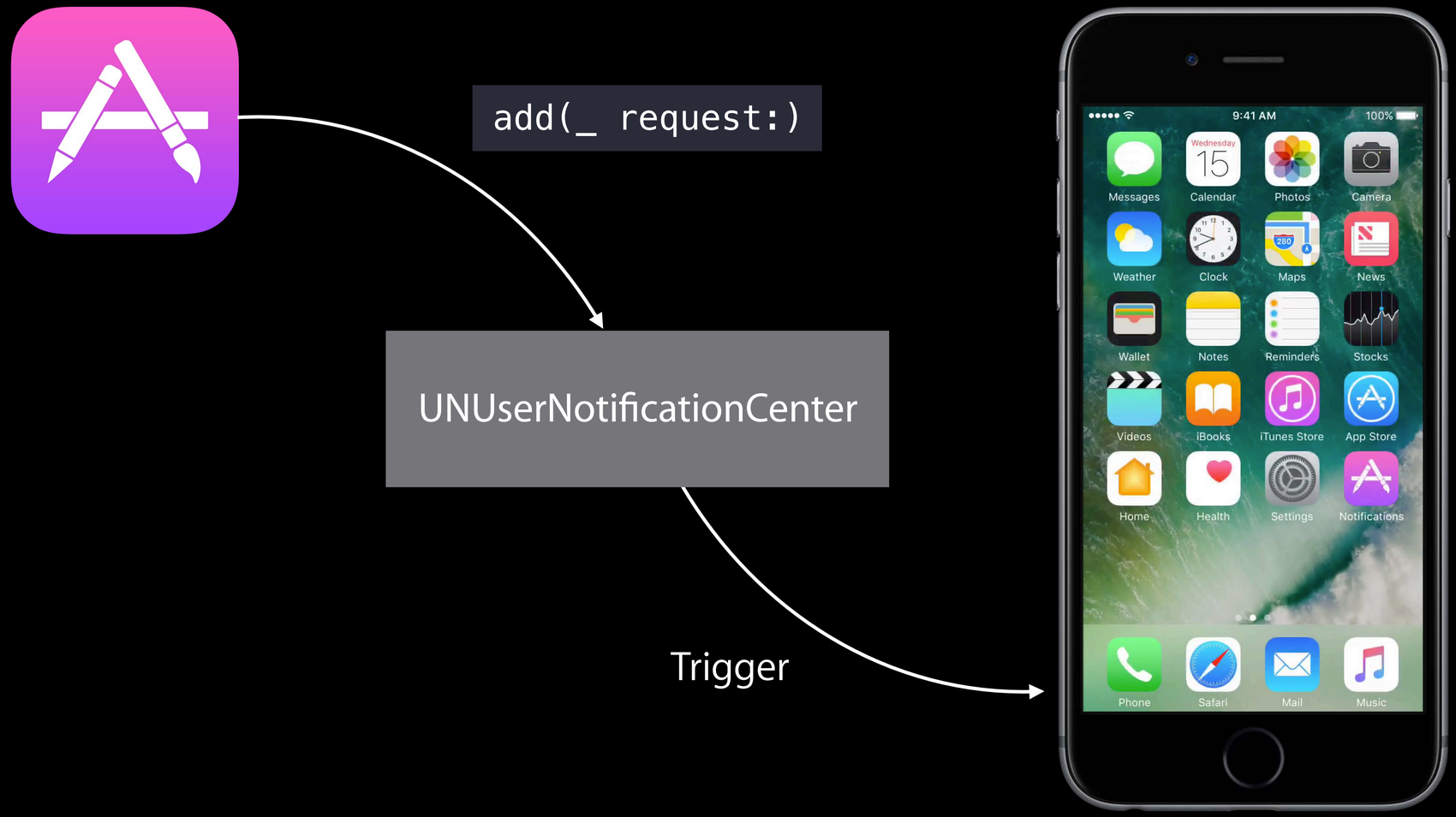
Local Notifications

```
add(_ request:)
```

UNUserNotificationCenter

Schedule

Local Notifications

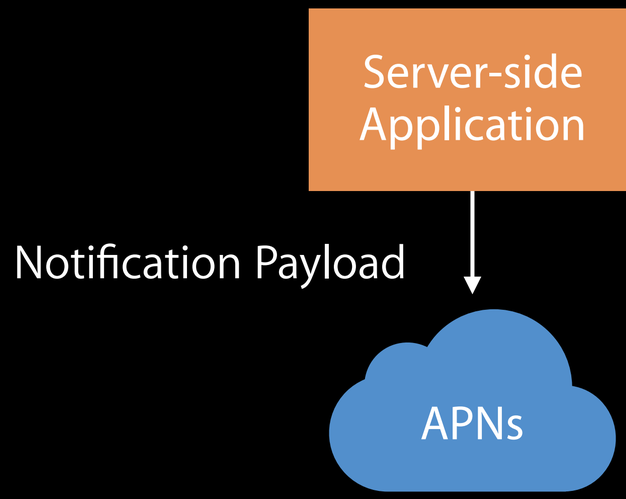


Schedule

Remote Notifications

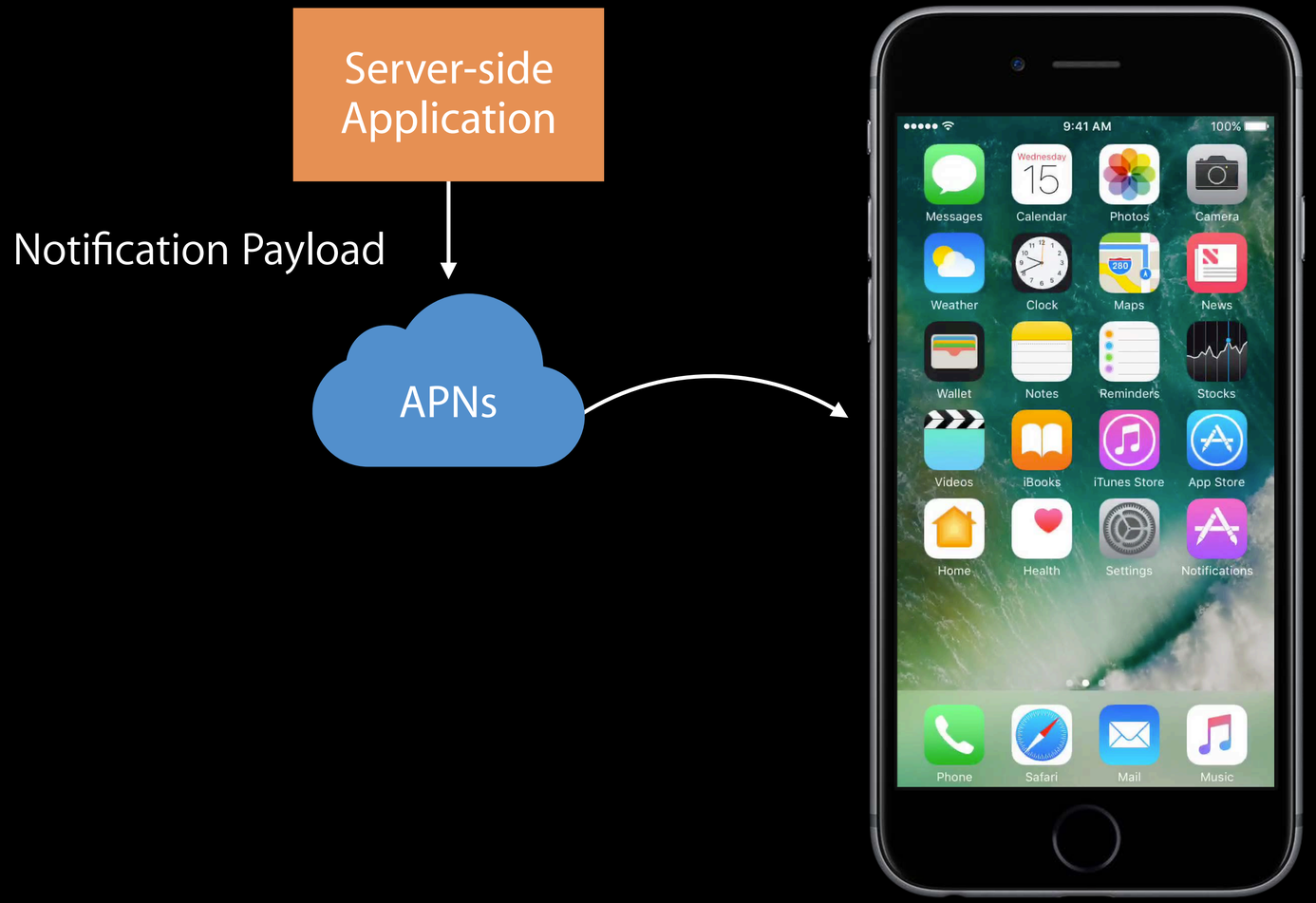
Schedule

Remote Notifications



Schedule

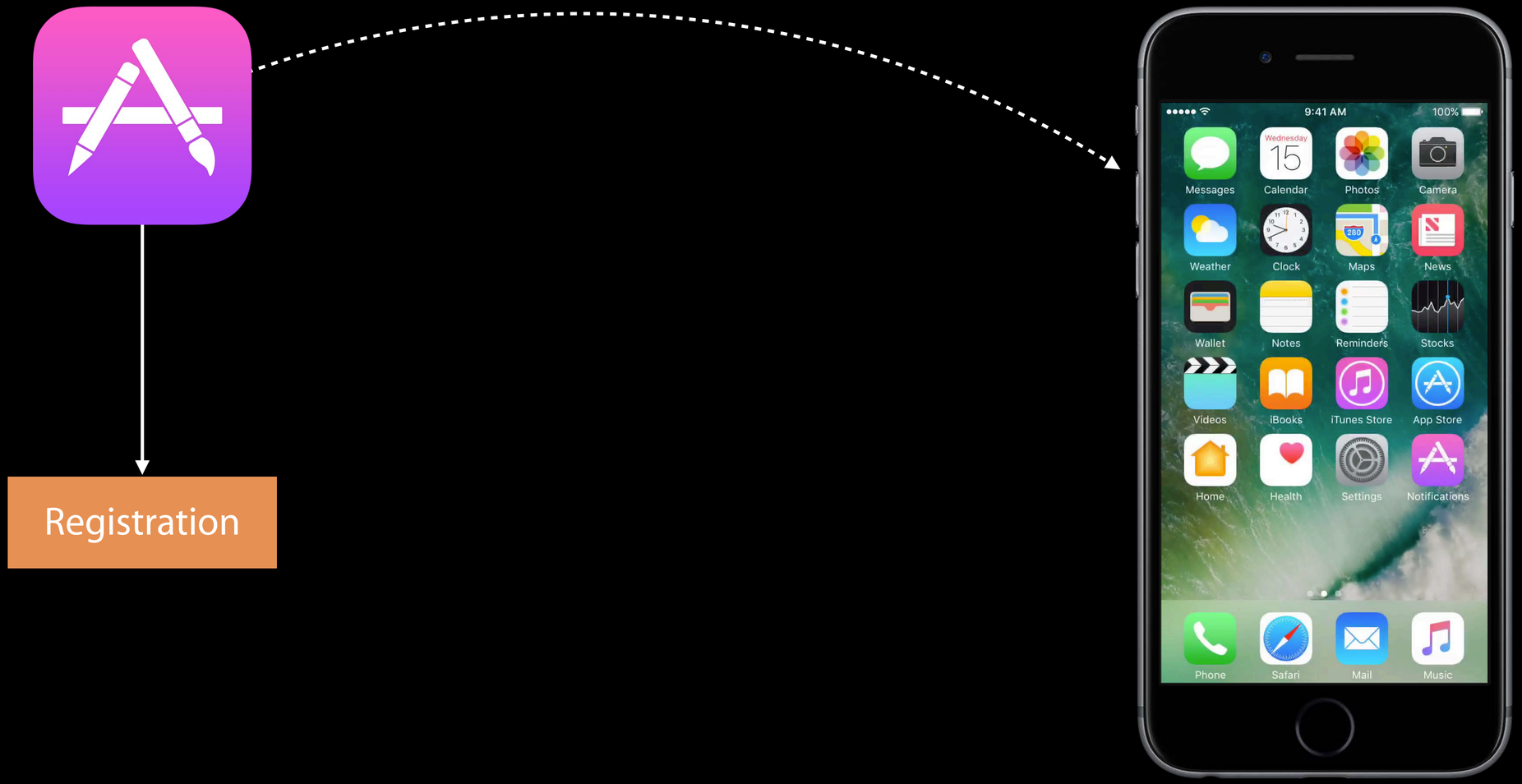
Remote Notifications



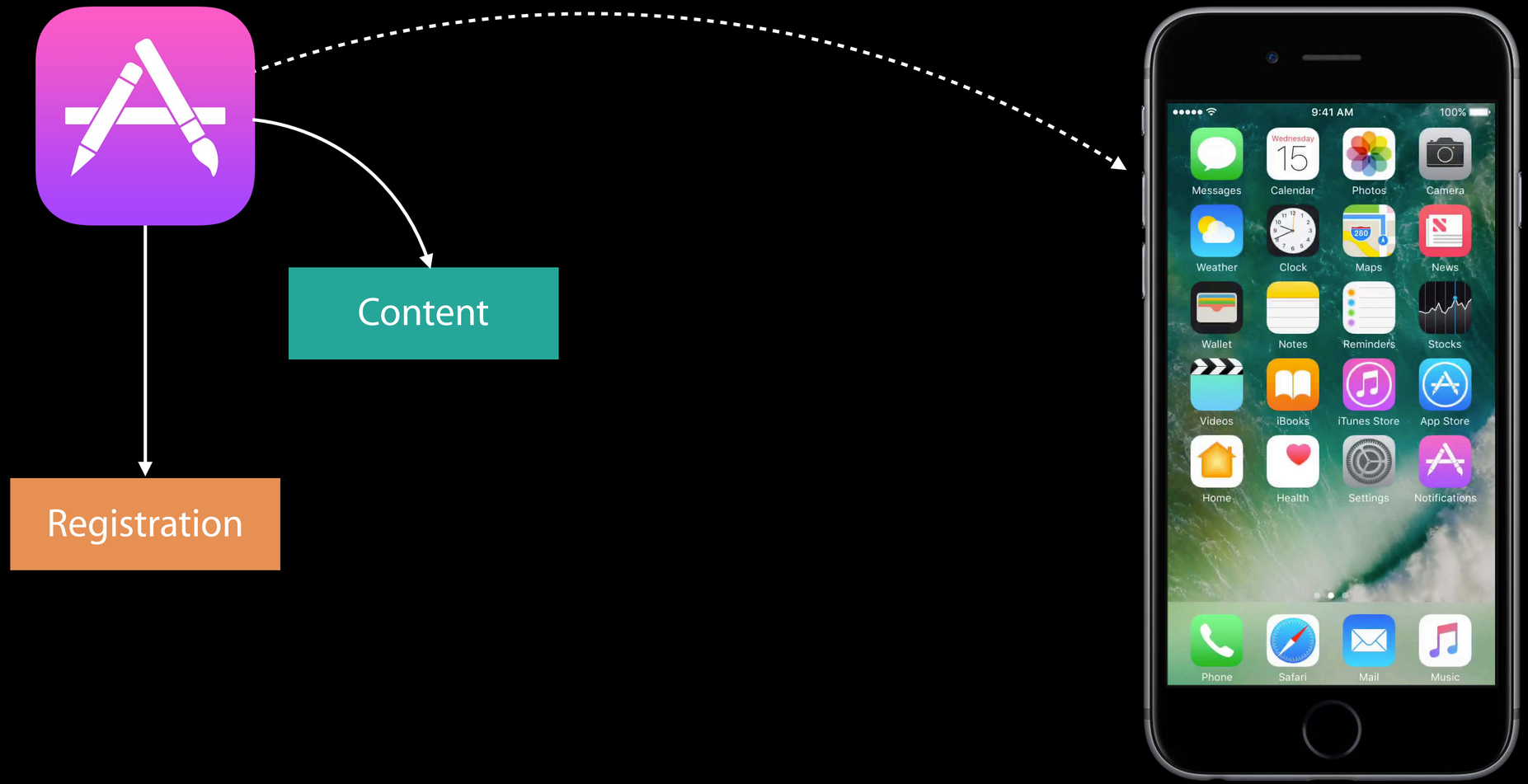
Delivery Summary

Delivery Summary

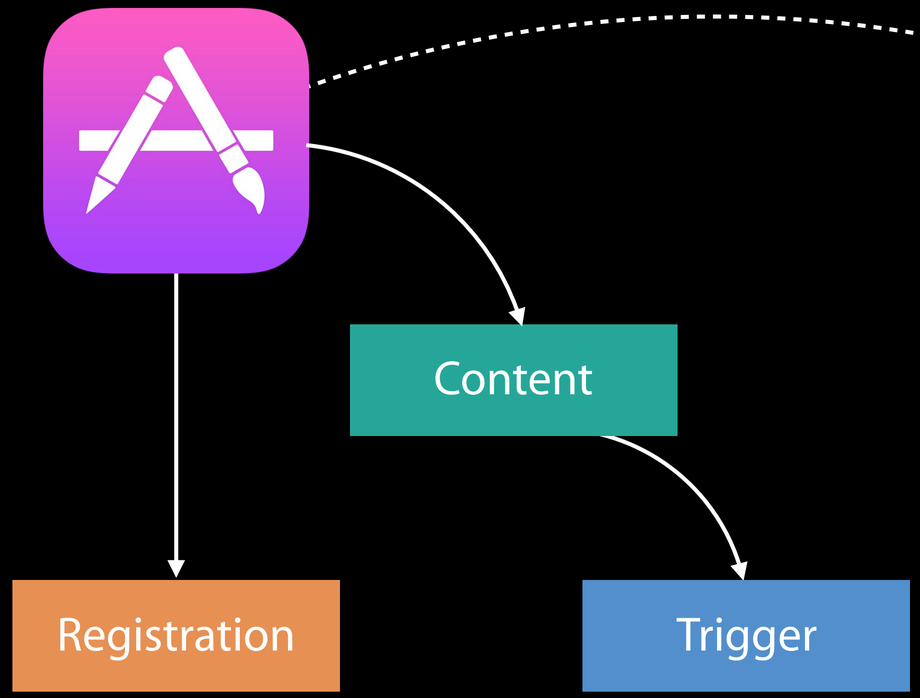
Delivery Summary



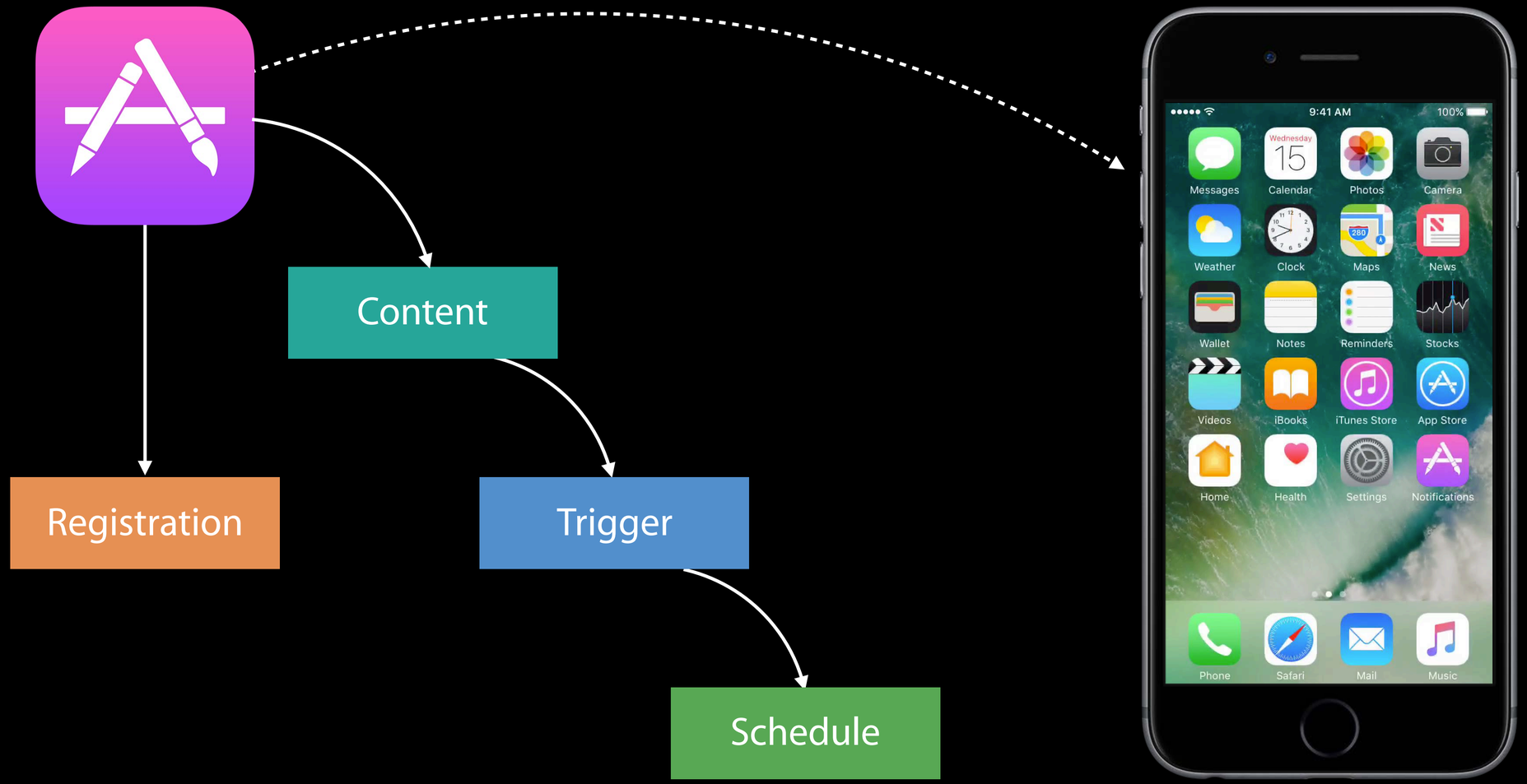
Delivery Summary



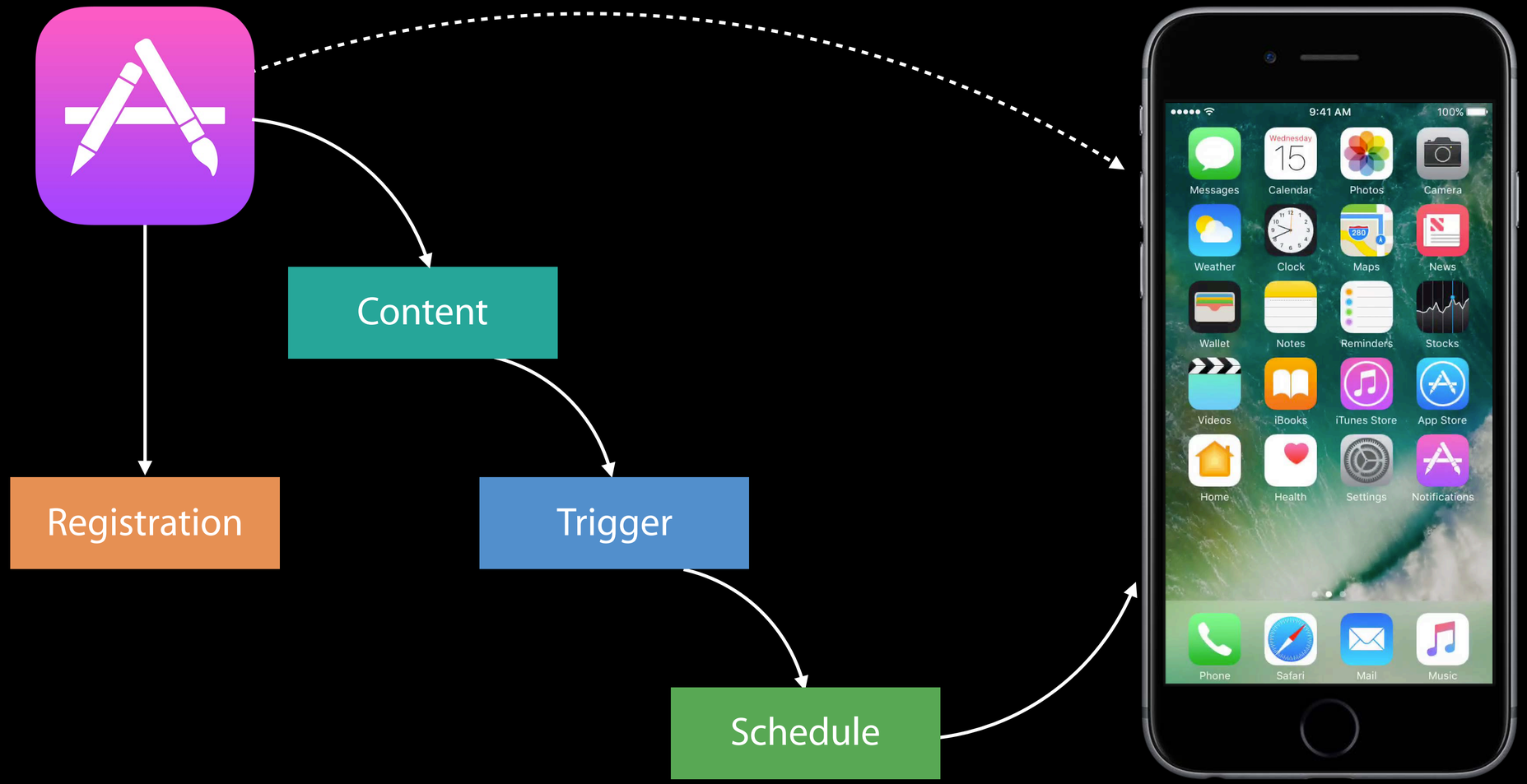
Delivery Summary



Delivery Summary



Delivery Summary



```
// Notification Delivery Summary
import UserNotifications

UNUserNotificationCenter.current().requestAuthorization([.alert, .sound, .badge])
    { (granted, error) in // ... }

let content = UNMutableNotificationContent()
content.title = "Introduction to Notifications"
content.body = "Let's talk about notifications!"

let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 5, repeats: false)

let requestIdentifier = "sampleRequest"
let request = UNNotificationRequest(identifier: requestIdentifier,
                                    content: content,
                                    trigger: trigger)

UNUserNotificationCenter.current().add(request) { (error) in // ... }
```

```swift
// Notification Delivery Summary
import UserNotifications

UNUserNotificationCenter.current().requestAuthorization([.alert, .sound, .badge])
    { (granted, error) in // ... }

let content = UNMutableNotificationContent()
content.title = "Introduction to Notifications"
content.body = "Let's talk about notifications!"

let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 5, repeats: false)

let requestIdentifier = "sampleRequest"
let request = UNNotificationRequest(identifier: requestIdentifier,
                                    content: content,
                                    trigger: trigger)

UNUserNotificationCenter.current().add(request) { (error) in // ... }
```

```
// Notification Delivery Summary
import UserNotifications

UNUserNotificationCenter.current().requestAuthorization([.alert, .sound, .badge])
    { (granted, error) in // ... }

let content = UNMutableNotificationContent()
content.title = "Introduction to Notifications"
content.body = "Let's talk about notifications!"

let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 5, repeats: false)

let requestIdentifier = "sampleRequest"
let request = UNNotificationRequest(identifier: requestIdentifier,
                                    content: content,
                                    trigger: trigger)

UNUserNotificationCenter.current().add(request) { (error) in // ... }
```

```swift
// Notification Delivery Summary
import UserNotifications

UNUserNotificationCenter.current().requestAuthorization([.alert, .sound, .badge])
    { (granted, error) in // ... }

let content = UNMutableNotificationContent()
content.title = "Introduction to Notifications"
content.body = "Let's talk about notifications!"

let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 5, repeats: false)

let requestIdentifier = "sampleRequest"
let request = UNNotificationRequest(identifier: requestIdentifier,
                                    content: content,
                                    trigger: trigger)

UNUserNotificationCenter.current().add(request) { (error) in // ... }
```

```
// Notification Delivery Summary
import UserNotifications

UNUserNotificationCenter.current().requestAuthorization([.alert, .sound, .badge])
    { (granted, error) in // ... }

let content = UNMutableNotificationContent()
content.title = "Introduction to Notifications"
content.body = "Let's talk about notifications!"

let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 5, repeats: false)

let requestIdentifier = "sampleRequest"
let request = UNNotificationRequest(identifier: requestIdentifier,
                                    content: content,
                                    trigger: trigger)

UNUserNotificationCenter.current().add(request) { (error) in // ... }
```

```
// Notification Delivery Summary
import UserNotifications

UNUserNotificationCenter.current().requestAuthorization([.alert, .sound, .badge])
    { (granted, error) in // ... }

let content = UNMutableNotificationContent()
content.title = "Introduction to Notifications"
content.body = "Let's talk about notifications!"

let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 5, repeats: false)

let requestIdentifier = "sampleRequest"
let request = UNNotificationRequest(identifier: requestIdentifier,
                                    content: content,
                                    trigger: trigger)

UNUserNotificationCenter.current().add(request) { (error) in // ... }
```

Notification Handling

Notification Handling

Notification Handling

Application in foreground

Notification Handling

Application in foreground

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol

func userNotificationCenter(_ center: UNUserNotificationCenter,
                            willPresent notification: UNNotification,
                            withCompletionHandler completionHandler:
                                (UNNotificationPresentationOptions) -> Void)
```

Notification Handling

Application in foreground

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol
```

```
func userNotificationCenter(_ center: UNUserNotificationCenter,
                            willPresent notification: UNNotification,
                            withCompletionHandler completionHandler:
                                (UNNotificationPresentationOptions) -> Void)
```

Notification Handling

Application in foreground

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol

func userNotificationCenter(_ center: UNUserNotificationCenter,
                            willPresent notification: UNNotification,
                            withCompletionHandler completionHandler:
                                (UNNotificationPresentationOptions) -> Void)
```

Notification Handling

NEW

Application in foreground

In-app presentation

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol

func userNotificationCenter(_ center: UNUserNotificationCenter,
                            willPresent notification: UNNotification,
                            withCompletionHandler completionHandler:
                                (UNNotificationPresentationOptions) -> Void)
```

Notification Handling

NEW

Application in foreground

In-app presentation

Notification Handling

NEW

Application in foreground

In-app presentation

```
func userNotificationCenter(_ center: UNUserNotificationCenter,
                            willPresent notification: UNNotification,
                            withCompletionHandler handlerBlock:
                                (UNNotificationPresentationOptions) -> Void) {
    // Roll banner and sound alert
    handlerBlock([.alert, .sound])
}
```

Notification Handling

NEW

Application in foreground

In-app presentation

```
func userNotificationCenter(_ center: UNUserNotificationCenter,
                            willPresent notification: UNNotification,
                            withCompletionHandler handlerBlock:
                                (UNNotificationPresentationOptions) -> Void) {
    // Roll banner and sound alert
    handlerBlock([.alert, .sound])
}
```

NEW

Notification Management

Notification Management

NEW

Overview

Notification Management

NEW

Overview

Access

- Pending Notifications
- Delivered Notifications

Notification Management

NEW

Overview

Access

- Pending Notifications
- Delivered Notifications

Remove Notifications

Notification Management

NEW

Overview

Access

- Pending Notifications
- Delivered Notifications

Remove Notifications

Update and promote Notifications

Notification Management

Request Identifier

Notification Management

Request Identifier

Local Notifications

- Set on Notification Request

Notification Management

Request Identifier

Local Notifications

- Set on Notification Request

Remote Notifications

- New field on the HTTP/2 request header: `apns-collapse-id`

Notification Management

Example

Notification Management

Example

Notifications scheduled for a sports game

- Start of game

```
// Pending Notification Removal

let gameStartIdentifier = "game1.start.identifier"
let gameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                             content: content,
                                             trigger: startTrigger)

UNUserNotificationCenter.current().add(gameStartRequest) { (error) in // ... }

// Game was cancelled

UNUserNotificationCenter.current()
    .removePendingNotificationRequests(withIdentifiers: [gameStartIdentifier])
```

```
// Pending Notification Removal

let gameStartIdentifier = "game1.start.identifier"
let gameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                             content: content,
                                             trigger: startTrigger)

UNUserNotificationCenter.current().add(gameStartRequest) { (error) in // ... }

// Game was cancelled

UNUserNotificationCenter.current()
    .removePendingNotificationRequests(withIdentifiers: [gameStartIdentifier])
```

```
// Pending Notification Removal

let gameStartIdentifier = "game1.start.identifier"
let gameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                             content: content,
                                             trigger: startTrigger)

UNUserNotificationCenter.current().add(gameStartRequest) { (error) in // ... }

// Game was cancelled

UNUserNotificationCenter.current()
    .removePendingNotificationRequests(withIdentifiers: [gameStartIdentifier])
```

```
// Pending Notification Removal

let gameStartIdentifier = "game1.start.identifier"
let gameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                             content: content,
                                             trigger: startTrigger)

UNUserNotificationCenter.current().add(gameStartRequest) { (error) in // ... }

// Game was cancelled
```

```
UNUserNotificationCenter.current()
    .removePendingNotificationRequests(withIdentifiers: [gameStartIdentifier])
```

```
// Pending Notification Removal

let gameStartIdentifier = "game1.start.identifier"
let gameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                             content: content,
                                             trigger: startTrigger)

UNUserNotificationCenter.current().add(gameStartRequest) { (error) in // ... }

// Game was cancelled

UNUserNotificationCenter.current()
    .removePendingNotificationRequests(withIdentifiers: [gameStartIdentifier])
```

```
// Pending Notification Update

let gameStartIdentifier = "game1.start.identifier"
let gameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                             content: content,
                                             trigger: startTrigger)

UNUserNotificationCenter.current().add(gameStartRequest) { (error) in // ... }

// Game start time was updated

let updatedGameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                                    content: content,
                                                    trigger: newStartTrigger)

UNUserNotificationCenter.current().add(updatedGameStartRequest) { (error) in // ... }
```

```
// Pending Notification Update

let gameStartIdentifier = "game1.start.identifier"
let gameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                             content: content,
                                             trigger: startTrigger)

UNUserNotificationCenter.current().add(gameStartRequest) { (error) in // ... }

// Game start time was updated
let updatedGameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                                    content: content,
                                                    trigger: newStartTrigger)

UNUserNotificationCenter.current().add(updatedGameStartRequest) { (error) in // ... }
```

```
// Pending Notification Update

let gameStartIdentifier = "game1.start.identifier"
let gameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                             content: content,
                                             trigger: startTrigger)

UNUserNotificationCenter.current().add(gameStartRequest) { (error) in // ... }

// Game start time was updated

let updatedGameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                                    content: content,
                                                    trigger: newStartTrigger)

UNUserNotificationCenter.current().add(updatedGameStartRequest) { (error) in // ... }
```

```swift
// Pending Notification Update

let gameStartIdentifier = "game1.start.identifier"
let gameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                             content: content,
                                             trigger: startTrigger)

UNUserNotificationCenter.current().add(gameStartRequest) { (error) in // ... }

// Game start time was updated

let updatedGameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                                    content: content,
                                                    trigger: newStartTrigger)

UNUserNotificationCenter.current().add(updatedGameStartRequest) { (error) in // ... }
```

```
// Pending Notification Update

let gameStartIdentifier = "game1.start.identifier"
let gameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                             content: content,
                                             trigger: startTrigger)

UNUserNotificationCenter.current().add(gameStartRequest) { (error) in // ... }

// Game start time was updated
let updatedGameStartRequest = UNNotificationRequest(identifier: gameStartIdentifier,
                                                    content: content,
                                                    trigger: newStartTrigger)

UNUserNotificationCenter.current().add(updatedGameStartRequest) { (error) in // ... }
```

Notification Management

Example

Notifications scheduled for a sports game

- Score Updates

```
// Delivered Notification Removal

let gameScoreIdentifier = "game1.score.identifier"
let gameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                             content: scoreContent,
                                             trigger: trigger)

UNUserNotificationCenter.current().add(gameScoreRequest) { (error) in // ... }

// Wrong game score was published

UNUserNotificationCenter.current()
    .removeDeliveredNotifications(withIdentifiers: [gameScoreIdentifier])
```

```
// Delivered Notification Removal

let gameScoreIdentifier = "game1.score.identifier"
let gameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                             content: scoreContent,
                                             trigger: trigger)

UNUserNotificationCenter.current().add(gameScoreRequest) { (error) in // ... }

// Wrong game score was published

UNUserNotificationCenter.current()
    .removeDeliveredNotifications(withIdentifiers: [gameScoreIdentifier])
```

```swift
// Delivered Notification Removal

let gameScoreIdentifier = "game1.score.identifier"
let gameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                             content: scoreContent,
                                             trigger: trigger)

UNUserNotificationCenter.current().add(gameScoreRequest) { (error) in // ... }

// Wrong game score was published

UNUserNotificationCenter.current()
    .removeDeliveredNotifications(withIdentifiers: [gameScoreIdentifier])
```

```
// Delivered Notification Removal

let gameScoreIdentifier = "game1.score.identifier"
let gameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                             content: scoreContent,
                                             trigger: trigger)

UNUserNotificationCenter.current().add(gameScoreRequest) { (error) in // ... }

// Wrong game score was published
```

```
UNUserNotificationCenter.current()
    .removeDeliveredNotifications(withIdentifiers: [gameScoreIdentifier])
```

```swift
// Delivered Notification Removal

let gameScoreIdentifier = "game1.score.identifier"
let gameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                             content: scoreContent,
                                             trigger: trigger)

UNUserNotificationCenter.current().add(gameScoreRequest) { (error) in // ... }

// Wrong game score was published

UNUserNotificationCenter.current()
    .removeDeliveredNotifications(withIdentifiers: [gameScoreIdentifier])
```

```
// Delivered Notification Update

let gameScoreIdentifier = "game1.score.identifier"
let gameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                             content: scoreContent,
                                             trigger: trigger)

UNUserNotificationCenter.current().add(gameScoreRequest) { (error) in // ... }

// Score game was updated

let updateGameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                                   content: newScoreContent,
                                                   trigger: newTrigger)

UNUserNotificationCenter.current().add(updateGameScoreRequest) { (error) in // ... }
```

```
// Delivered Notification Update

let gameScoreIdentifier = "game1.score.identifier"
let gameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                             content: scoreContent,
                                             trigger: trigger)

UNUserNotificationCenter.current().add(gameScoreRequest) { (error) in // ... }

// Score game was updated
let updateGameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                                   content: newScoreContent,
                                                   trigger: newTrigger)

UNUserNotificationCenter.current().add(updateGameScoreRequest) { (error) in // ... }
```

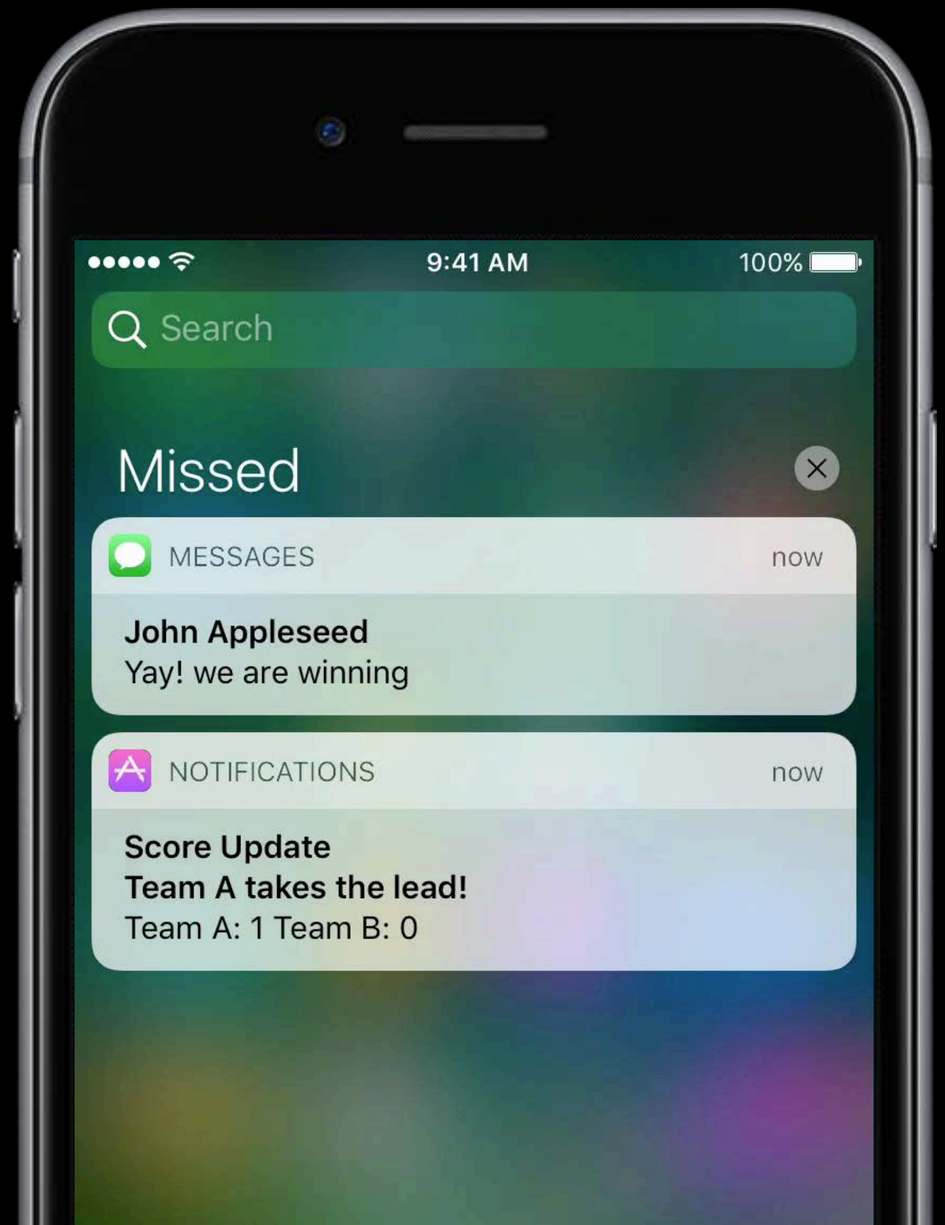
```swift
// Delivered Notification Update

let gameScoreIdentifier = "game1.score.identifier"
let gameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                             content: scoreContent,
                                             trigger: trigger)

UNUserNotificationCenter.current().add(gameScoreRequest) { (error) in // ... }

// Score game was updated

let updateGameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                                   content: newScoreContent,
                                                   trigger: newTrigger)

UNUserNotificationCenter.current().add(updateGameScoreRequest) { (error) in // ... }
```

```
// Delivered Notification Update

let gameScoreIdentifier = "game1.score.identifier"
let gameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                             content: scoreContent,
                                             trigger: trigger)

UNUserNotificationCenter.current().add(gameScoreRequest) { (error) in // ... }

// Score game was updated

let updateGameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                                   content: newScoreContent,
                                                   trigger: newTrigger)

UNUserNotificationCenter.current().add(updateGameScoreRequest) { (error) in // ... }
```

```swift
// Delivered Notification Update

let gameScoreIdentifier = "game1.score.identifier"
let gameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                             content: scoreContent,
                                             trigger: trigger)

UNUserNotificationCenter.current().add(gameScoreRequest) { (error) in // ... }

// Score game was updated

let updateGameScoreRequest = UNNotificationRequest(identifier: gameScoreIdentifier,
                                                   content: newScoreContent,
                                                   trigger: newTrigger)

UNUserNotificationCenter.current().add(updateGameScoreRequest) { (error) in // ... }
```

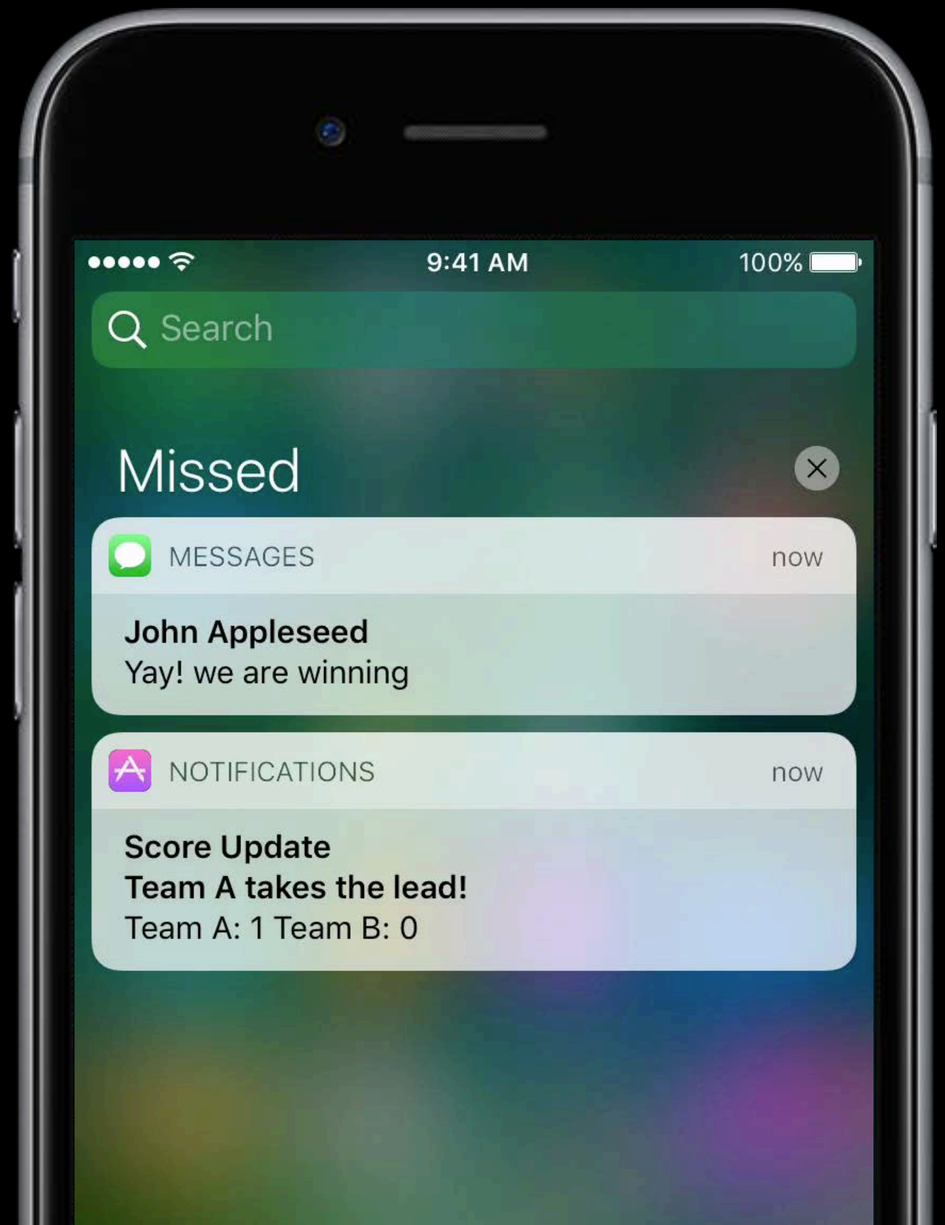
Notification Management

Example



Notification Management

Example



Notification Actions

Julien Barlerin iOS Frameworks QA Engineer

Default Action

Default Action

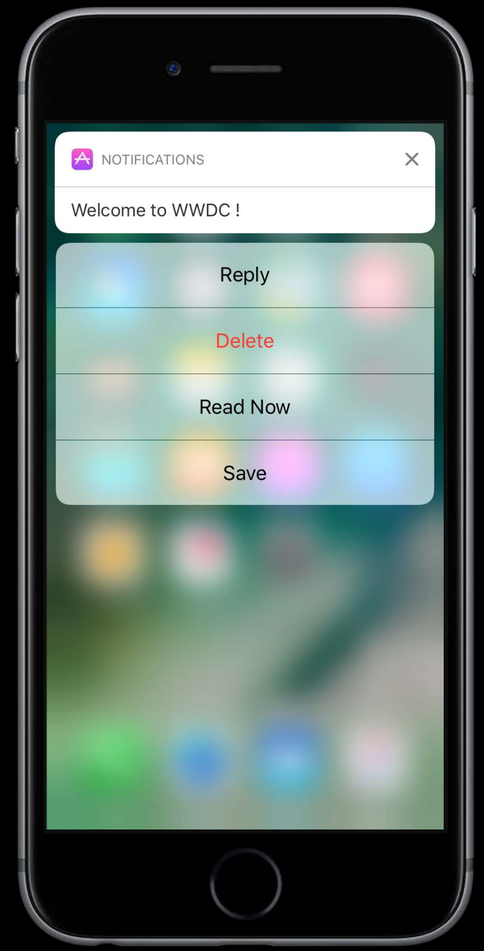
Default Action

Default Action

Actionable Notifications

Actionable Notifications

Buttons with customizable title



Actionable Notifications

Buttons with customizable title

Text input

Actionable Notifications

Buttons with customizable title

Text input

Background or foreground

Actionable Notifications

Buttons with customizable title

Text input

Background or foreground

iOS and watchOS

Actionable Notifications

iOS

Actionable Notifications

iOS

Actionable Notifications

watchOS

Actionable Notifications

watchOS

Actionable Notifications

watchOS

Actionable Notifications

watchOS

Quick Interaction Techniques for watchOS	Presidio	Wednesday 11:00AM

Actionable Notifications

Registration

Actionable Notifications

Registration

```
let action = UNNotificationAction(identifier:”reply",title:"Reply",options:[])

let category = UNNotificationCategory(identifier: "message", actions: [action],
        minimalActions: [action], intentIdentifiers: [], options: [])

UNUserNotificationCenter.current().setNotificationCategories([category])
```

Actionable Notifications

Registration

```
let action = UNNotificationAction(identifier:”reply",title:"Reply",options:[])

let category = UNNotificationCategory(identifier: "message", actions: [action],
        minimalActions: [action], intentIdentifiers: [], options: [])

UNUserNotificationCenter.current().setNotificationCategories([category])
```

Actionable Notifications

Registration

```
let action = UNNotificationAction(identifier:”reply",title:"Reply",options:[])

let category = UNNotificationCategory(identifier: "message", actions: [action],
        minimalActions: [action], intentIdentifiers: [], options: [])

UNUserNotificationCenter.current().setNotificationCategories([category])
```

Actionable Notifications

Registration

```
let action = UNNotificationAction(identifier:”reply",title:"Reply",options:[])

let category = UNNotificationCategory(identifier: "message", actions: [action],
        minimalActions: [action], intentIdentifiers: [], options: [])

UNUserNotificationCenter.current().setNotificationCategories([category])
```

Introducing SiriKit	Presidio	Wednesday 05:00PM

Actionable Notifications

Registration

```
let action = UNNotificationAction(identifier:”reply",title:"Reply",options:[])

let category = UNNotificationCategory(identifier: "message", actions: [action],
        minimalActions: [action], intentIdentifiers: [], options: [])

UNUserNotificationCenter.current().setNotificationCategories([category])
```

Actionable Notifications

Presentation

Actionable Notifications

Presentation

Remote Notifications

```
{
  aps : {
        alert : “Welcome to WWDC !”,
        category: "message"
        }
}
```

Actionable Notifications

Presentation

Remote Notifications

```
{
  aps : {
        alert : “Welcome to WWDC !”,
        category: "message"
        }
}
```

Local Notifications

```
content.categoryIdentifier = "message"
```

Dismiss Action

NEW

Dismiss Action

NEW

Dismiss Action

NEW

Dismiss Action

NEW

Dismiss Action

Category option

NEW

Dismiss Action

Category option

NEW

```
customDismissAction: UNNotificationCategoryOptions
```

Dismiss Action

Category option

NEW

```
customDismissAction: UNNotificationCategoryOptions


let category = UNNotificationCategory(identifier: "message", actions: [action],
    minimalActions: [action], intentIdentifiers: [], options: [])
```

Dismiss Action

Category option

NEW

```
customDismissAction: UNNotificationCategoryOptions


let category = UNNotificationCategory(identifier: "message", actions: [action],
    minimalActions: [action], intentIdentifiers: [], options: [.customDismissAction])
```

Notification Actions

Summary

Notification Actions

Summary

Default action

Notification Actions

Summary

Default action

Actionable Notifications

Notification Actions

Summary

Default action

Actionable Notifications

Dismiss action

Notification Actions

Response handling

Notification Actions

Response handling

NEW

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol

func userNotificationCenter(_ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: () -> Void)
```

Notification Actions

Response handling

NEW

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol

func userNotificationCenter(_ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: () -> Void)
```

Notification Actions

Response handling

NEW

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol
```

```
func userNotificationCenter(_ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: () -> Void)
```

Notification Actions

Response handling

NEW

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol

func userNotificationCenter(_ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: () -> Void)
```

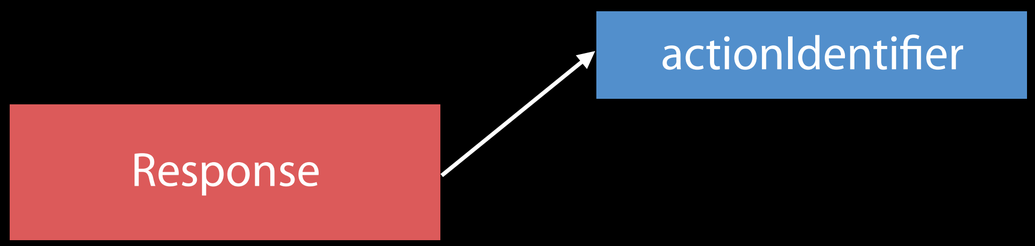
Response

Notification Actions

Response handling

NEW

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol

func userNotificationCenter(_ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: () -> Void)
```

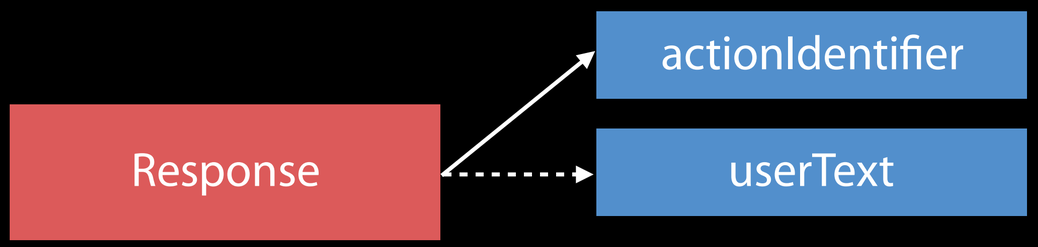


Notification Actions

Response handling

NEW

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol

func userNotificationCenter(_ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: () -> Void)
```

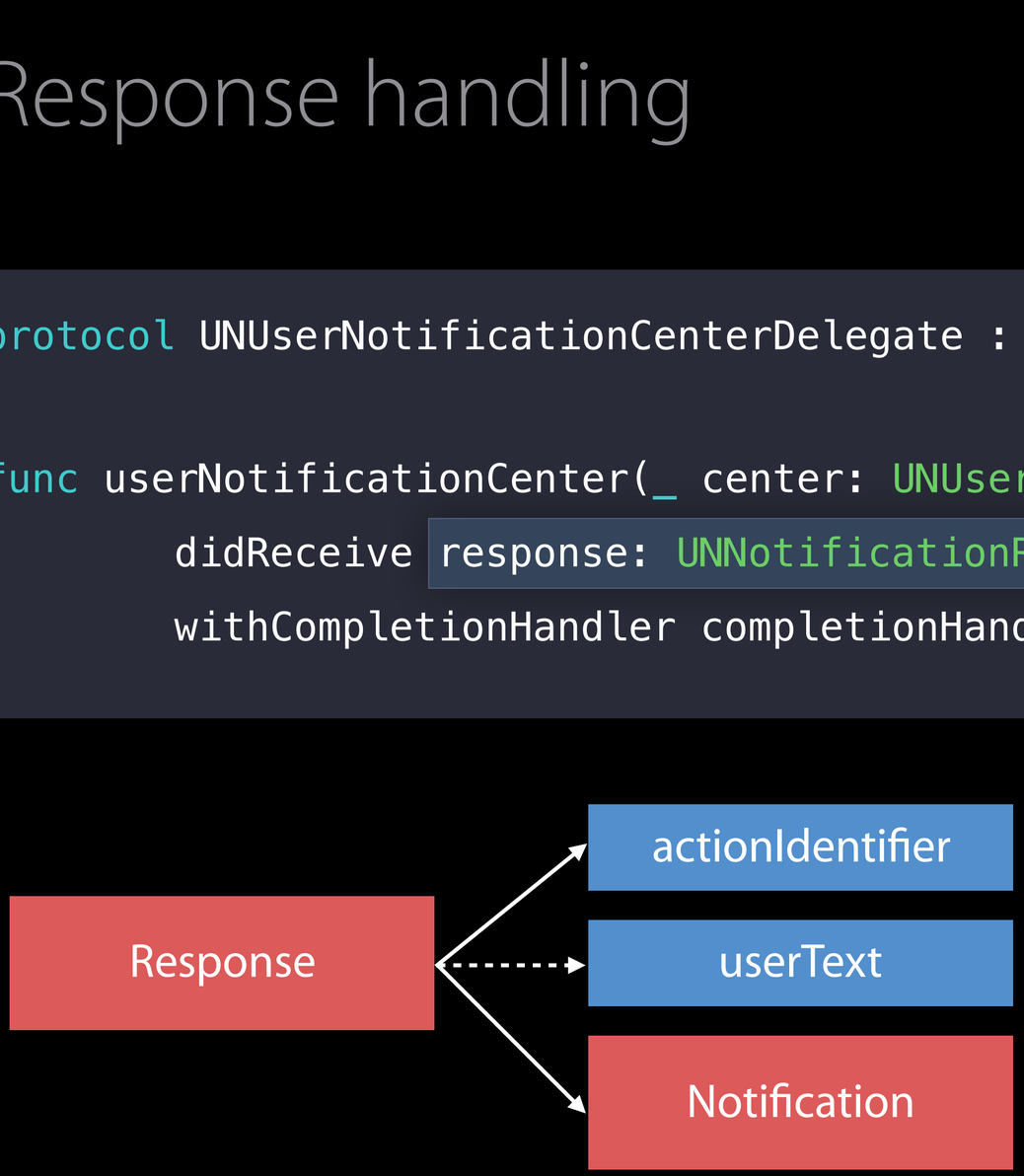


Notification Actions

Response handling

NEW

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol

func userNotificationCenter(_ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: () -> Void)
```

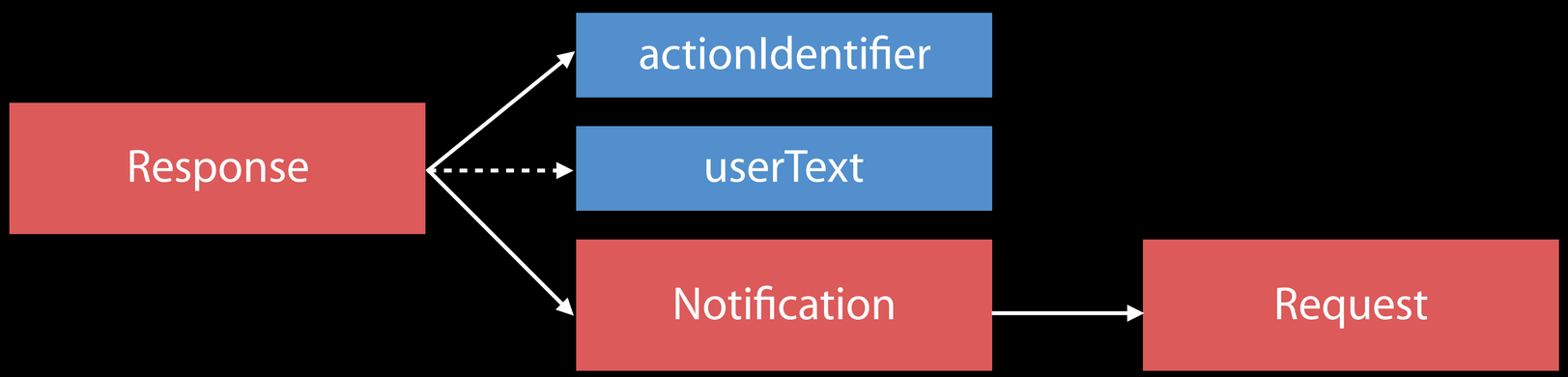


Notification Actions

Response handling

NEW

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol

func userNotificationCenter(_ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: () -> Void)
```

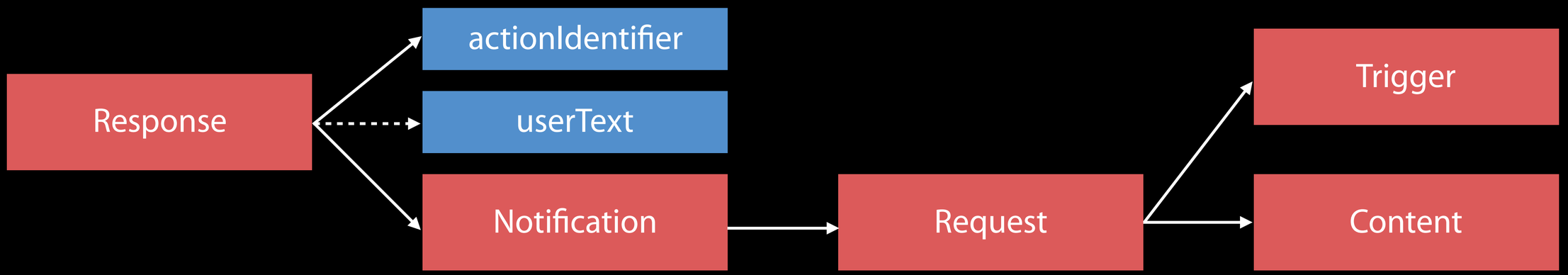


Notification Actions

Response handling

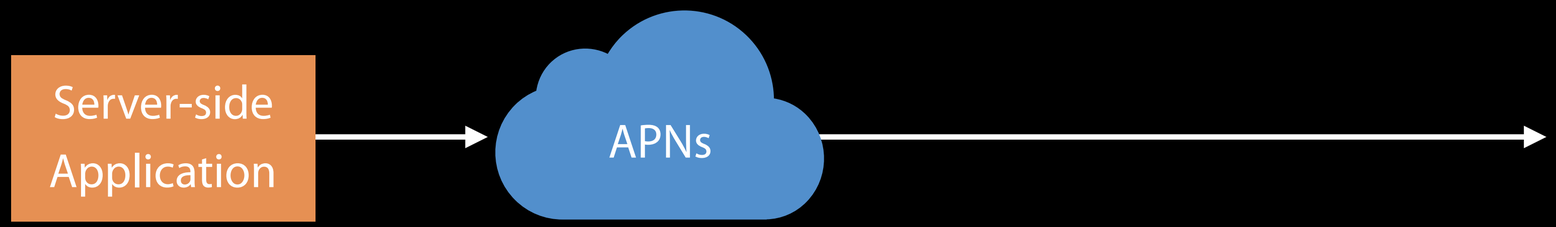
NEW

```
protocol UNUserNotificationCenterDelegate : NSObjectProtocol

func userNotificationCenter(_ center: UNUserNotificationCenter,
        didReceive response: UNNotificationResponse,
        withCompletionHandler completionHandler: () -> Void)
```



Remote Notifications

Remote Notifications



Service Extension

Service Extension

Basics

Service Extension

Basics

Non UI iOS Extension

Service Extension

Basics

Non UI iOS Extension

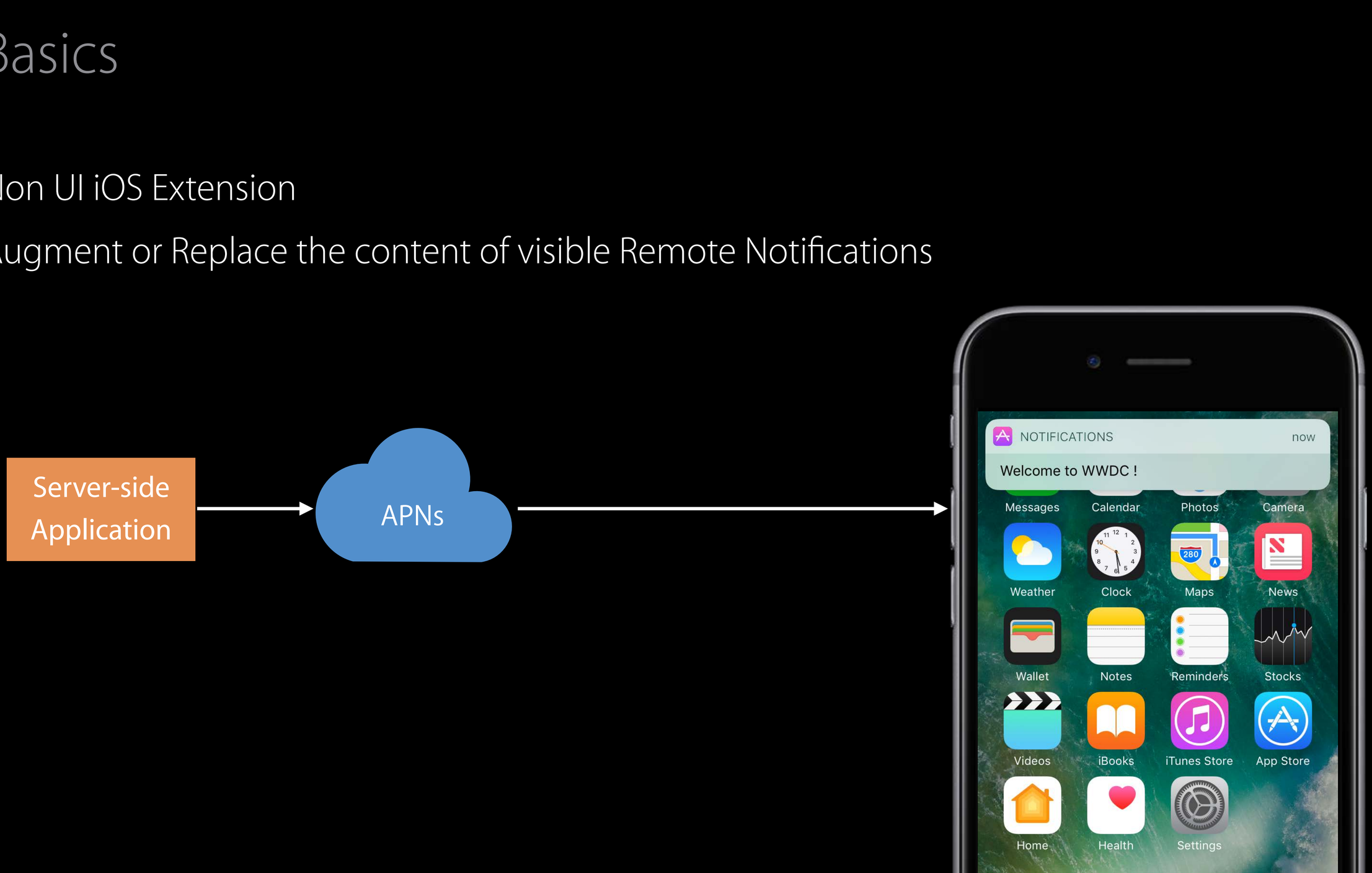
Augment or Replace the content of visible Remote Notifications

Service Extension

Basics

Non UI iOS Extension

Augment or Replace the content of visible Remote Notifications

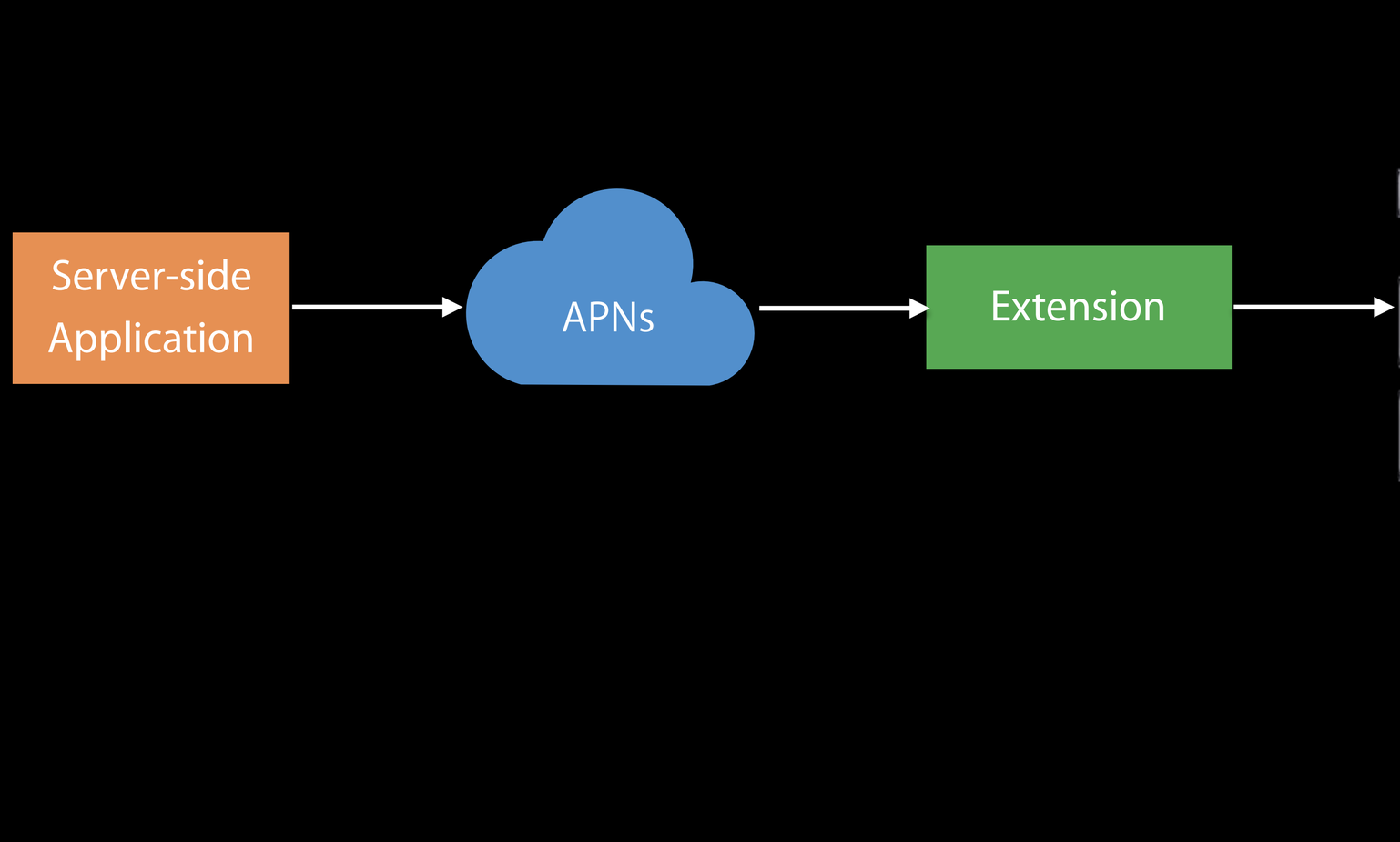


Service Extension

Basics

Non UI iOS Extension

Augment or Replace the content of visible Remote Notifications



Service Extension

Details

Service Extension

Details

Short execution time

Service Extension

Details

Short execution time

Fallback

Service Extension

Potential uses

Service Extension

Potential uses

End-to-end encryption

Service Extension

Potential uses

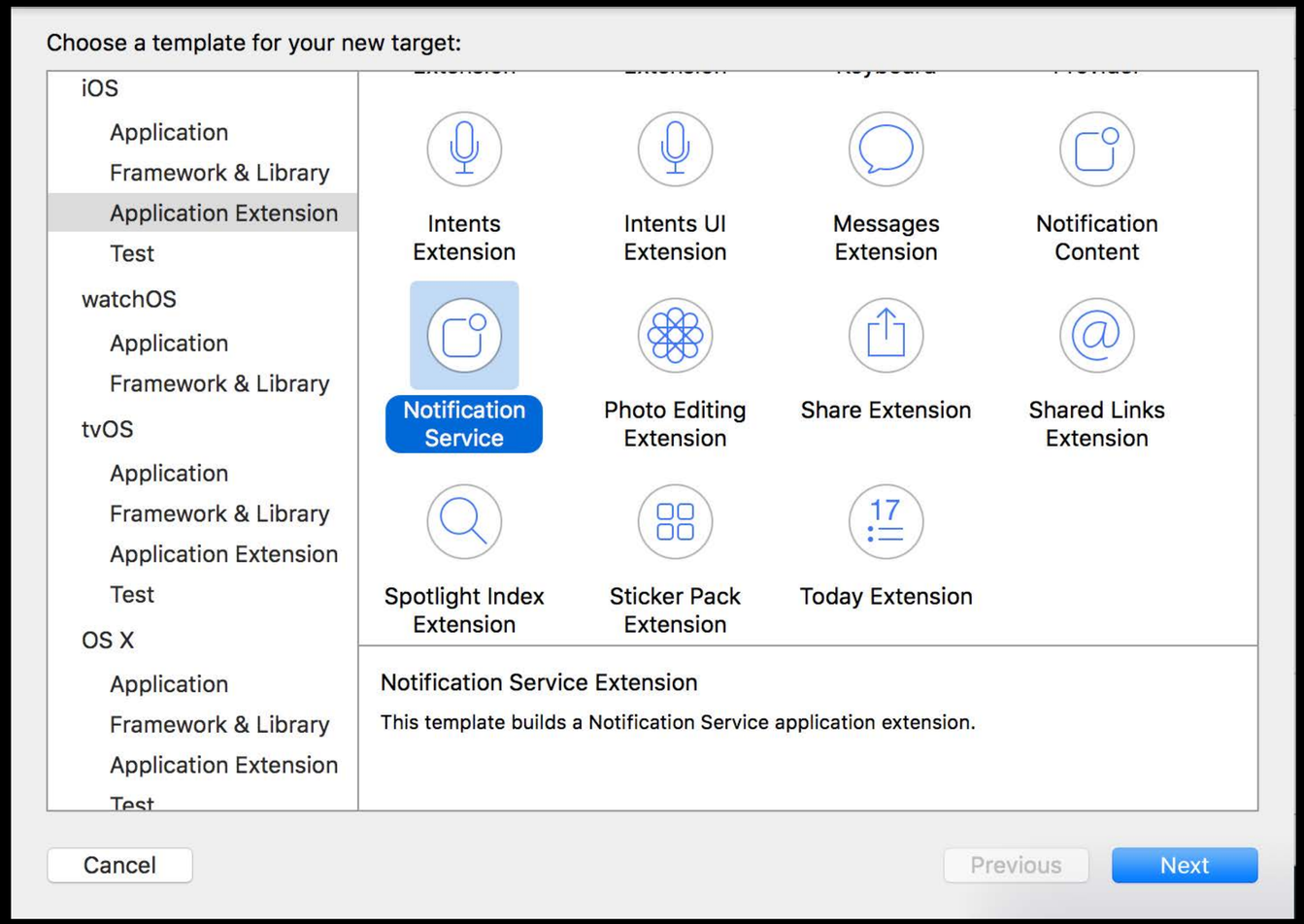
End-to-end encryption

Add Attachments

Service Extension

How to implement it

Service Extension



```swift
// Service Extension

import UserNotifications

class NotificationService: UNNotificationServiceExtension {

    override func didReceive(request: UNNotificationRequest, withContentHandler
contentHandler:(UNNotificationContent) -> Void) {
        // Modify the notification content
    }

    override func serviceExtensionTimeWillExpire() {
        // Called before the extension will be terminated by the system
    }

}
```

```swift
// Service Extension

import UserNotifications

class NotificationService: UNNotificationServiceExtension {

    override func didReceive(request: UNNotificationRequest, withContentHandler
contentHandler:(UNNotificationContent) -> Void) {
        // Modify the notification content
    }

    override func serviceExtensionTimeWillExpire() {
        // Called before the extension will be terminated by the system
    }

}
```

```swift
// Service Extension

import UserNotifications

class NotificationService: UNNotificationServiceExtension {

    override func didReceive(request: UNNotificationRequest, withContentHandler
contentHandler:(UNNotificationContent) -> Void) {
        // Modify the notification content
    }

    override func serviceExtensionTimeWillExpire() {
        // Called before the extension will be terminated by the system
    }

}
```

```swift
// Service Extension

import UserNotifications

class NotificationService: UNNotificationServiceExtension {

    override func didReceive(request: UNNotificationRequest, withContentHandler
contentHandler:(UNNotificationContent) -> Void) {
        // Modify the notification content
    }

    override func serviceExtensionTimeWillExpire() {
        // Called before the extension will be terminated by the system
    }

}
```

Service Extension

Example payload

```
{
  aps : {
        alert : “New Message Available”,
        mutable-content : 1
        },
  encrypted-content : “#myencryptedcontent”
}
```

Service Extension

Example payload

```
{
  aps : {
        alert : “New Message Available”,
        mutable-content : 1
        },
  encrypted-content : “#myencryptedcontent”
}
```

Service Extension

Example payload

```
{
  aps : {
        alert : “New Message Available”,
        mutable-content : 1
        },
    encrypted-content : “#myencryptedcontent”
  }
```

```swift
// Decrypt Remote Notification Payload in Service Extension and Update Notification Content

override func didReceive(request: UNNotificationRequest, withContentHandler contentHandler:
(UNNotificationContent) -> Void) {

    // Decrypt the payload
    let decryptedBody = decrypt(request.content.userInfo["encrypted-content"])

    let newContent = UNMutableNotificationContent()

    // Modify the notification content
    newContent.body = decryptedBody

    // Call content handler with updated content
    contentHandler(newContent)
}
```

```swift
// Decrypt Remote Notification Payload in Service Extension and Update Notification Content

override func didReceive(request: UNNotificationRequest, withContentHandler contentHandler:
(UNNotificationContent) -> Void) {

    // Decrypt the payload
    let decryptedBody = decrypt(request.content.userInfo["encrypted-content"])

    let newContent = UNMutableNotificationContent()

    // Modify the notification content
    newContent.body = decryptedBody

    // Call content handler with updated content
    contentHandler(newContent)
}
```

```swift
// Decrypt Remote Notification Payload in Service Extension and Update Notification Content

override func didReceive(request: UNNotificationRequest, withContentHandler contentHandler:
(UNNotificationContent) -> Void) {

    // Decrypt the payload
    let decryptedBody = decrypt(request.content.userInfo["encrypted-content"])

    let newContent = UNMutableNotificationContent()

    // Modify the notification content
    newContent.body = decryptedBody

    // Call content handler with updated content
    contentHandler(newContent)
}
```

```swift
// Decrypt Remote Notification Payload in Service Extension and Update Notification Content

override func didReceive(request: UNNotificationRequest, withContentHandler contentHandler:
(UNNotificationContent) -> Void) {

    // Decrypt the payload
    let decryptedBody = decrypt(request.content.userInfo["encrypted-content"])

    let newContent = UNMutableNotificationContent()

    // Modify the notification content
    newContent.body = decryptedBody

    // Call content handler with updated content
    contentHandler(newContent)
}
```

```swift
// Decrypt Remote Notification Payload in Service Extension and Update Notification Content

override func didReceive(request: UNNotificationRequest, withContentHandler contentHandler:
(UNNotificationContent) -> Void) {

    // Decrypt the payload
    let decryptedBody = decrypt(request.content.userInfo["encrypted-content"])

    let newContent = UNMutableNotificationContent()

    // Modify the notification content
    newContent.body = decryptedBody

    // Call content handler with updated content
    contentHandler(newContent)
}
```

Summary

Summary

Notifications Overview

Summary

Notifications Overview

User Notifications Framework

Summary

Notifications Overview

User Notifications Framework

- Registration

Summary

Notifications Overview

User Notifications Framework

- Registration
- Content

Summary

Notifications Overview

User Notifications Framework

- Registration
- Content
- Scheduling

Summary

Notifications Overview

User Notifications Framework

- Registration
- Content
- Scheduling
- Management

Summary

Notifications Overview

User Notifications Framework

- Registration
- Content
- Scheduling
- Management
- Actions

Summary

Notifications Overview

User Notifications Framework

- Registration
- Content
- Scheduling
- Management
- Actions

Service Extensions

More Information

https://developer.apple.com/wwdc16/707

Related Sessions

Advanced Notifications	Pacific Heights	Wednesday 10:00AM
Quick Interaction Techniques for watchOS	Presidio	Wednesday 11:00AM

Labs

Notifications Lab	Frameworks Lab C	Wednesday 11:00AM
Notifications Lab	Games, Graphics and Media Lab B	Friday 09:00AM

WWDC16