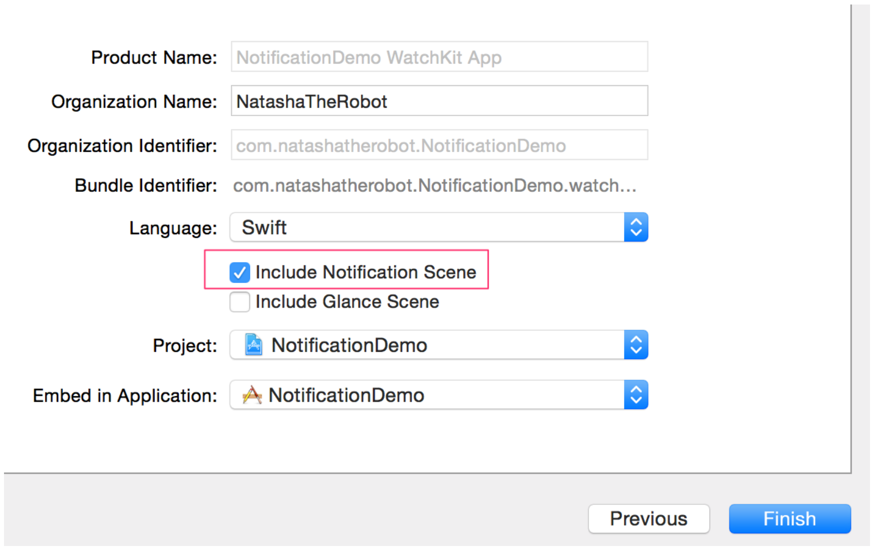
WatchKit Notifications Tutorial

Start a New WatchKit Project

1. Follow the **Hello, WatchKit Tutorial** to set up a new WatchKit project.
2. However, when you're creating a new WatchKit App Target, make sure to **check the Include Notification Scene** option

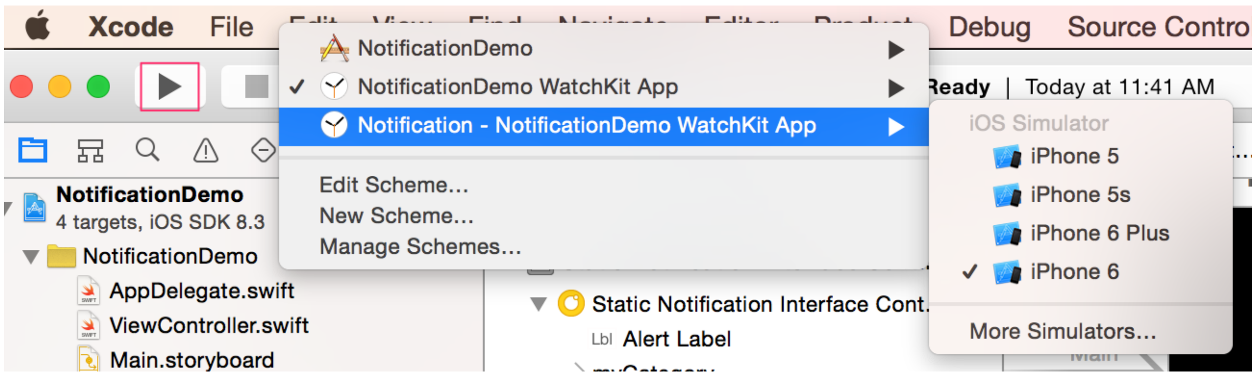


Product Name: NotificationDemo WatchKit App
Organization Name: NatashaTheRobot
Organization Identifier: com.natashatherobot.NotificationDemo
Bundle Identifier: com.natashatherobot.NotificationDemo.watch...
Language: Swift
[✓] Include Notification Scene
[ ] Include Glance Scene
Project: NotificationDemo
Embed in Application: NotificationDemo

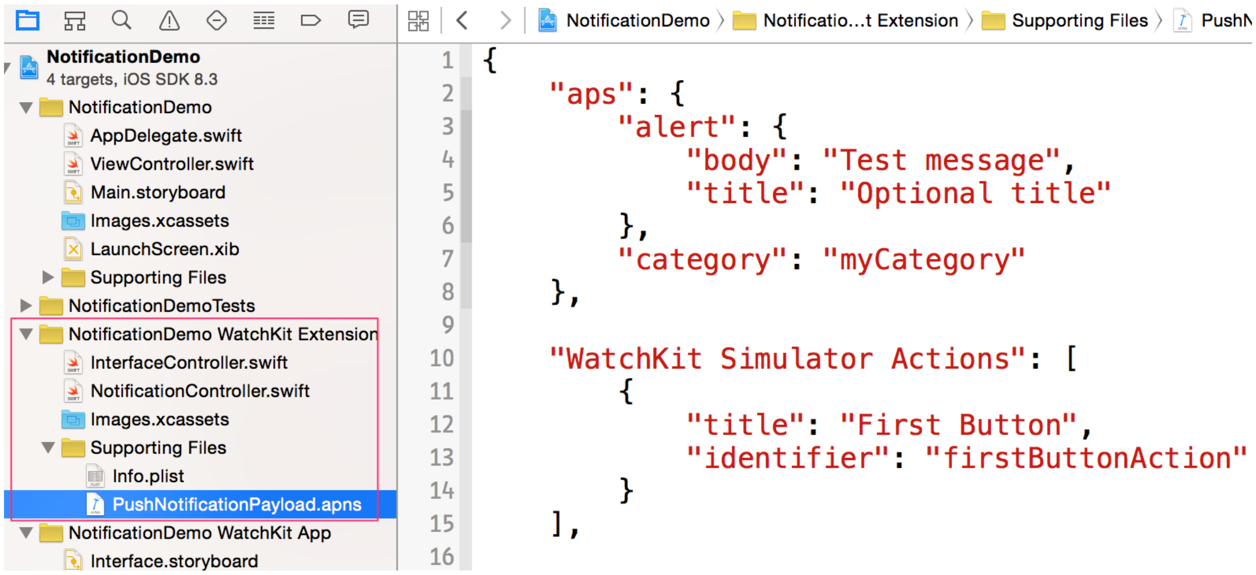
3. Make sure your Watch App runs properly.

Run the Static Notification

1. Open **Interface.storyboard**. You'll notice there is now a **Static** and a **Dynamic Notification Interface Controller**.
2. Change the active scheme to the **Notification - [YOUR APP NAME]** and **Run** to view the Notification



3. Notice that the Notification has pre-filled in data that is not in the Storyboard. To access the test data, open the **PushNotificationPayload.apns** in your WatchKit App Extension (located in the Supporting Files folder)


NotificationDemo 〉 Notificatio...t Extension 〉 Supporting Files 〉 PushN

NotificationDemo
4 targets, iOS SDK 8.3
- NotificationDemo
  - AppDelegate.swift
  - ViewController.swift
  - Main.storyboard
  - Images.xcassets
  - LaunchScreen.xib
  - Supporting Files
- NotificationDemoTests
- NotificationDemo WatchKit Extension
  - InterfaceController.swift
  - NotificationController.swift
  - Images.xcassets
  - Supporting Files
    - Info.plist
    - PushNotificationPayload.apns
- NotificationDemo WatchKit App
  - Interface.storyboard

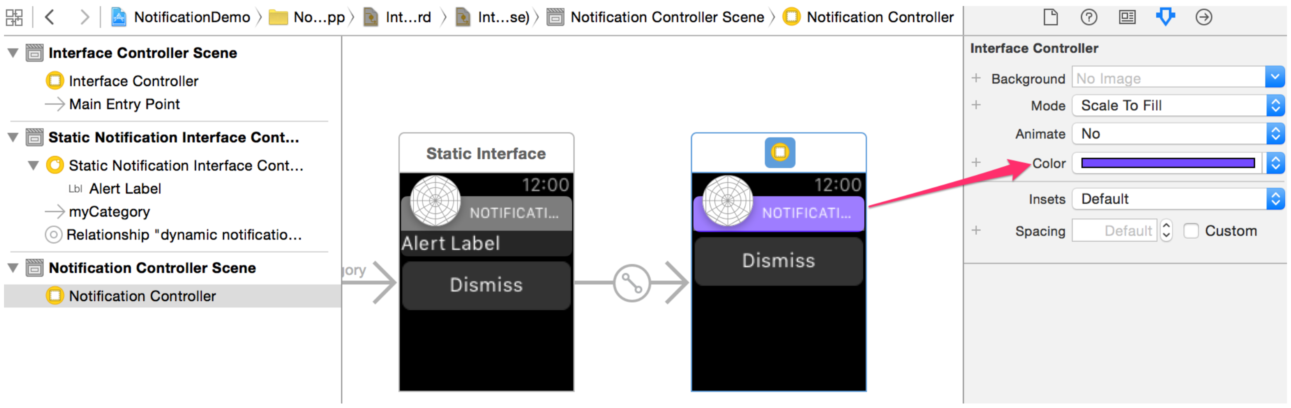
```
{
    "aps": {
        "alert": {
            "body": "Test message",
            "title": "Optional title"
        },
        "category": "myCategory"
    },

    "WatchKit Simulator Actions": [
        {
            "title": "First Button",
            "identifier": "firstButtonAction"
        }
    ],
```

4. Change the **title** of the **First Button** to something else. **Run** the Notification scheme to see how that changed what is displayed.
5. Change the other attributes in the Notification payload to see how that changes what is displayed.

Run the Dynamic Notification

1. In the Storyboard, select the **Dynamic Notification Controller** and change it's color to easily identify it


NotificationDemo 〉 No...pp 〉 Int...rd 〉 Int...se) 〉 Notification Controller Scene 〉 Notification Controller

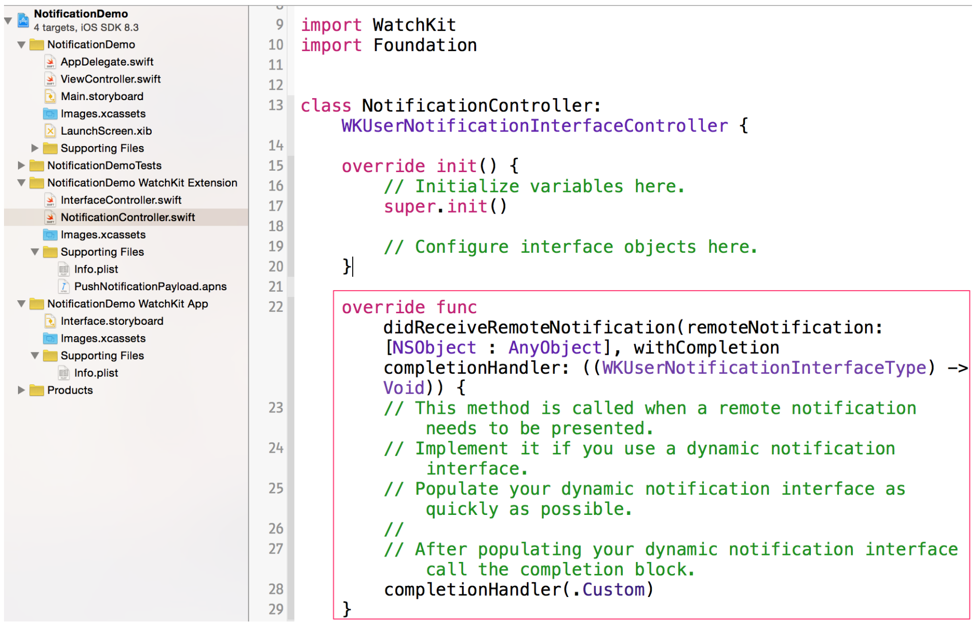
- Interface Controller Scene
  - Interface Controller
  - Main Entry Point
- Static Notification Interface Cont...
  - Static Notification Interface Cont...
    - Alert Label
  - myCategory
  - Relationship "dynamic notificatio...
- Notification Controller Scene
  - Notification Controller

Interface Controller

Background	No Image
Mode	Scale To Fill
Animate	No
Color	🟪
Insets	Default
Spacing	Default ☐ Custom

2. Now, go to **NotificationController.swift** in your WatchKit Extension.

3. Uncomment (**Command + /**) the **didReceiveRemoteNotification:withCompletion:** method.



```
import WatchKit
import Foundation


class NotificationController:
    WKUserNotificationInterfaceController {

    override init() {
        // Initialize variables here.
        super.init()

        // Configure interface objects here.
    }

    override func
        didReceiveRemoteNotification(remoteNotification:
        [NSObject : AnyObject], withCompletion
        completionHandler: ((WKUserNotificationInterfaceType) ->
        Void)) {
        // This method is called when a remote notification
            needs to be presented.
        // Implement it if you use a dynamic notification
            interface.
        // Populate your dynamic notification interface as
            quickly as possible.
        //
        // After populating your dynamic notification interface
            call the completion block.
        completionHandler(.Custom)
    }
```

4. **Run** the app to see the **Dynamic** Notification.
5. Pass the **.Default** option to the completion handler.

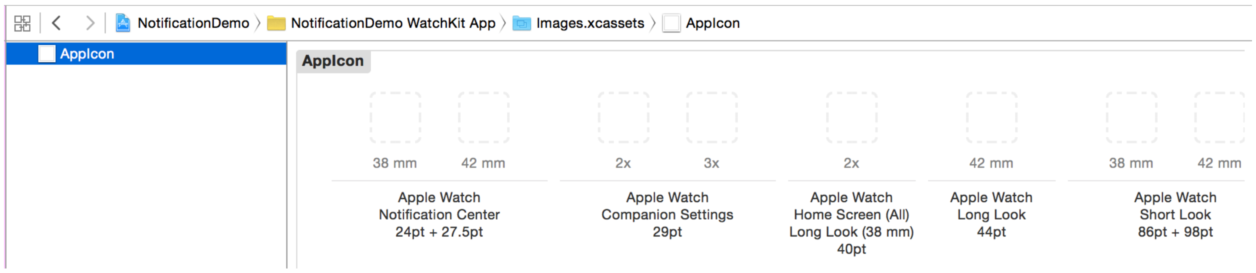
```
override func didReceiveRemoteNotification
    (remoteNotification: [NSObject : AnyObject],
    withCompletion completionHandler:
    ((WKUserNotificationInterfaceType) -> Void)) {
    // This method is called when a remote notification
        needs to be presented.
    // Implement it if you use a dynamic notification
        interface.
    // Populate your dynamic notification interface as
        quickly as possible.
    //
    // After populating your dynamic notification interface
        call the completion block.
    completionHandler(.Default)
}
```

6. Run the Notification scheme to see which notification shows up.

Customizing The Notification

Change Icon

1. In your Watch App's **Images.xcassets** folder, select the AppIcon option. Notice that you need **8 icons** for your Watch App!



NotificationDemo 〉 NotificationDemo WatchKit App 〉 Images.xcassets 〉 AppIcon

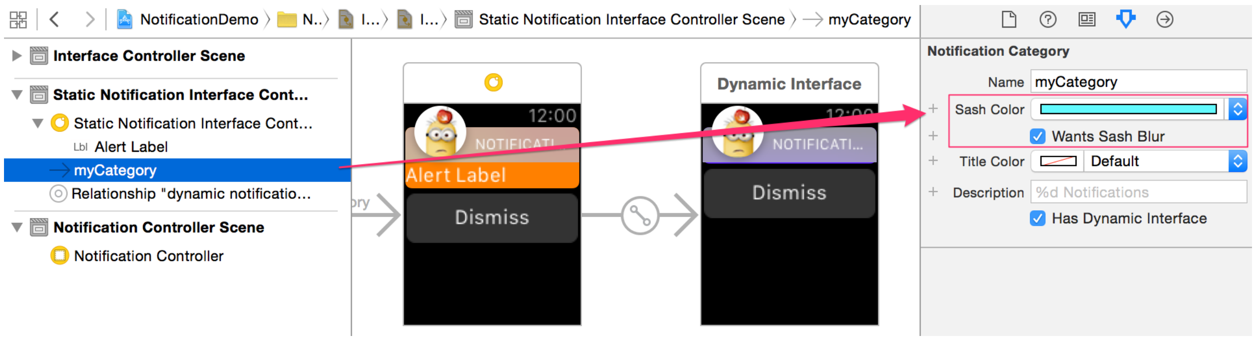
AppIcon

38 mm, 42 mm	2x, 3x	2x	42 mm	38 mm, 42 mm
Apple Watch Notification Center 24pt + 27.5pt	Apple Watch Companion Settings 29pt	Apple Watch Home Screen (All) Long Look (38 mm) 40pt	Apple Watch Long Look 44pt	Apple Watch Short Look 86pt + 98pt

2. Use the icons in the **AppIcons folder** of this tutorial to populate the icons.
3. Run the Notification scheme to see how the icon changes the Notification

Change the Sash Color

1. In the Storyboard, select the **myCategory** in the **Static Notification Controller** and change the **Sash Color**. Check and uncheck the **Wants Sash Blur** option.

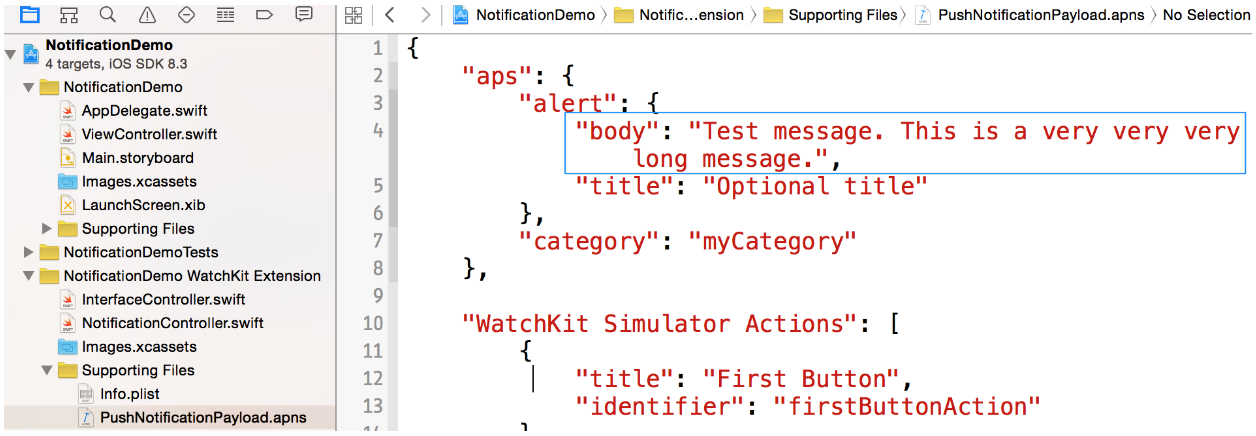

NotificationDemo 〉 N..〉 I...〉 I...〉 Static Notification Interface Controller Scene 〉 myCategory
Interface Controller Scene
Static Notification Interface Cont...
Static Notification Interface Cont...
Alert Label
myCategory
Relationship "dynamic notificatio...
Notification Controller Scene
Notification Controller
12:00
NOTIFICATI
Alert Label
Dismiss
Dynamic Interface
NOTIFICATI...
Dismiss

Notification Category
Name: myCategory
Sash Color
✓ Wants Sash Blur
Title Color: Default
Description: %d Notifications
✓ Has Dynamic Interface

2. Run the Notification scheme to see how the different sash color changes the Notification

Wrap Text

1. In **PushNotificationPayload.apns**, update the alert body text to be longer.

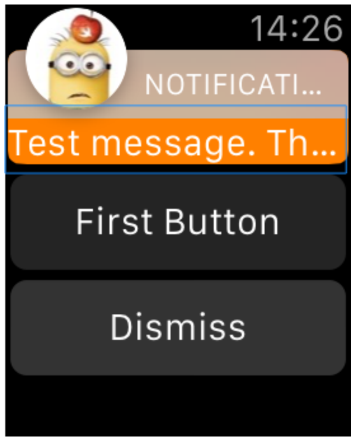


NotificationDemo
4 targets, iOS SDK 8.3
NotificationDemo
AppDelegate.swift
ViewController.swift
Main.storyboard
Images.xcassets
LaunchScreen.xib
Supporting Files
NotificationDemoTests
NotificationDemo WatchKit Extension
InterfaceController.swift
NotificationController.swift
Images.xcassets
Supporting Files
Info.plist
PushNotificationPayload.apns
NotificationDemo › Notific...ension › Supporting Files › PushNotificationPayload.apns › No Selection

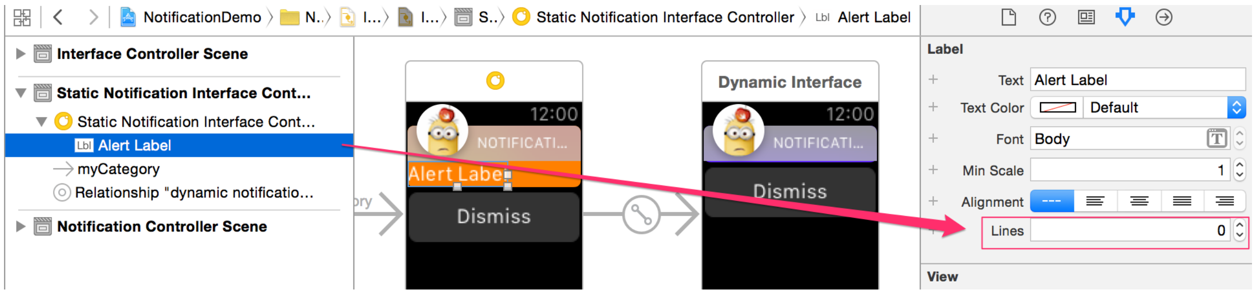
```
{
    "aps": {
        "alert": {
            "body": "Test message. This is a very very very
                long message.",
            "title": "Optional title"
        },
        "category": "myCategory"
    },

    "WatchKit Simulator Actions": [
        {
            "title": "First Button",
            "identifier": "firstButtonAction"
```

2. Run the Notification scheme to see how the message is displayed. Notice that it gets cut off:



3. In the Storyboard, change the **number of lines** for the **Alert Label** to **0**



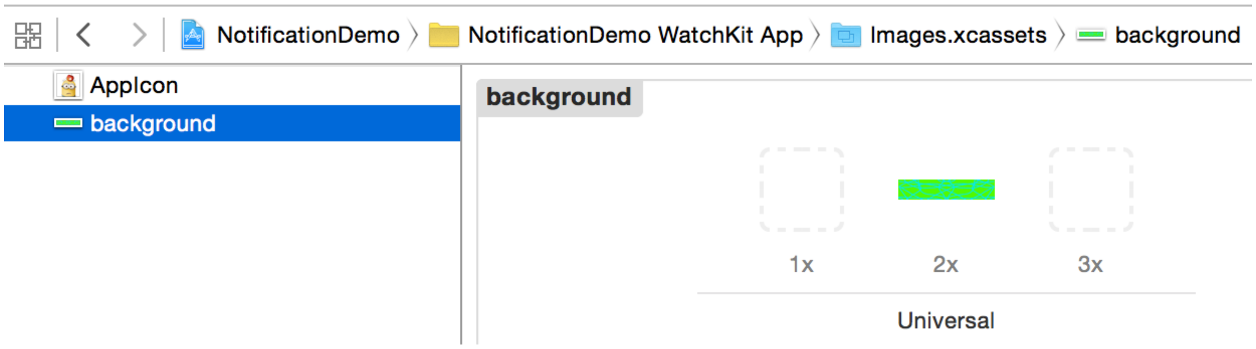
NotificationDemo › N.. › I... › I... › S.. › Static Notification Interface Controller › Alert Label
Interface Controller Scene
Static Notification Interface Cont...
Static Notification Interface Cont...
Alert Label
myCategory
Relationship "dynamic notificatio...
Notification Controller Scene
12:00
NOTIFICATI...
Alert Label
Dismiss
Dynamic Interface
12:00
NOTIFICATI...
Dismiss
Label
View

Text: Alert Label
Text Color: Default
Font: Body
Min Scale: 1
Alignment
Lines: 0

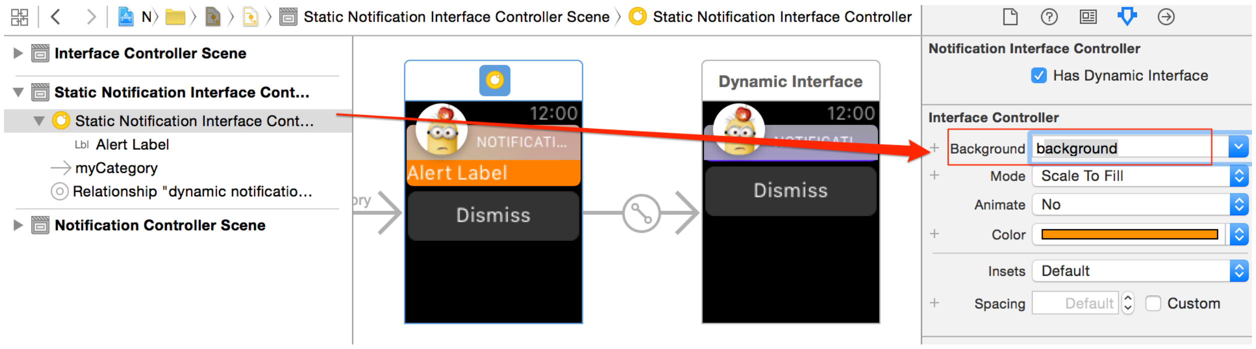
4. Run the Notification scheme to see the text now wrap around

Change the Background Image

1. Add the **background image** located in the **BackgroundImage folder** in your tutorial to **Images.xcassets** in your **WatchKit App**


NotificationDemo 〉 NotificationDemo WatchKit App 〉 Images.xcassets 〉 background
AppIcon
background
background
1x
2x
3x
Universal

2. In the Storyboard, select the Static Notification Interface Controller, set the background to the background image you just added.


Static Notification Interface Controller Scene 〉 Static Notification Interface Controller
Interface Controller Scene
Static Notification Interface Cont...
Static Notification Interface Cont...
Alert Label
myCategory
Relationship "dynamic notificatio...
Notification Controller Scene
12:00
NOTIFICATI...
Alert Label
Dismiss
Dynamic Interface
Dismiss

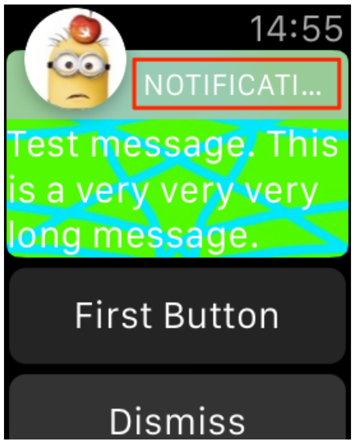
Notification Interface Controller
Has Dynamic Interface ✓

Interface Controller
Background: background
Mode: Scale To Fill
Animate: No
Color:
Insets: Default
Spacing: Default ☐ Custom

3. Change the **Mode** to **Aspect Fill** for better image scaling.
4. Run the Notification scheme to see what the Notification looks like with this background

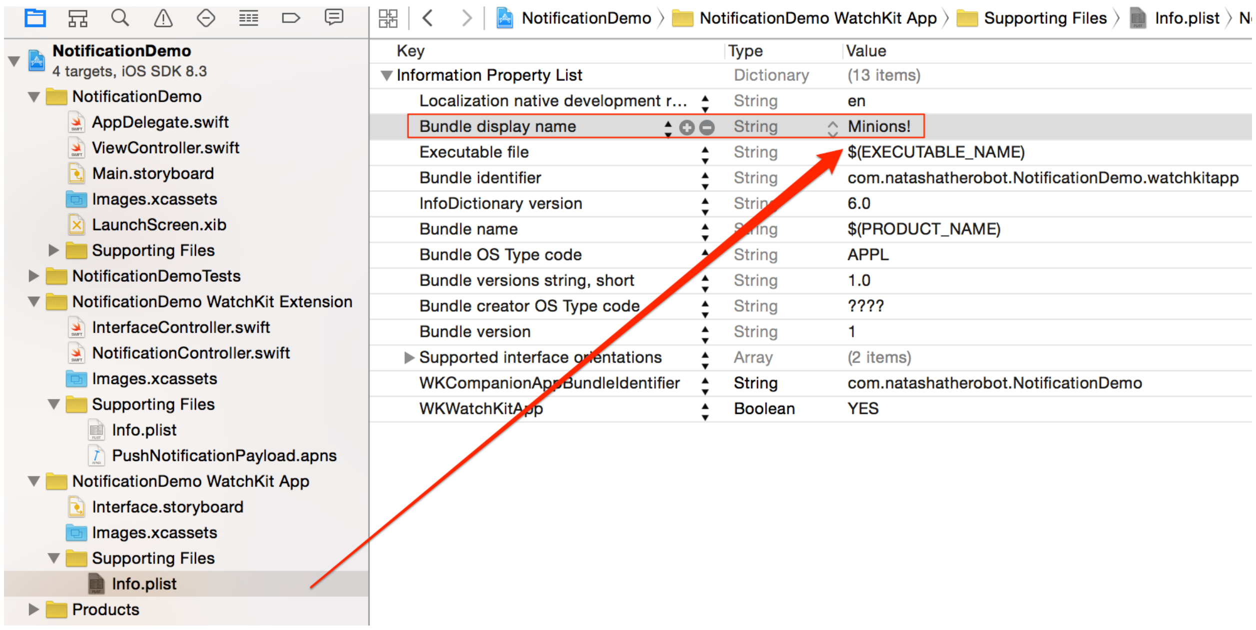
Change the WatchKit App Title

1. Notice that our **App Title** (NotificationDemo) is cut off on the Notification



Test message. This is a very very very long message.

2. Go to **Info.plist** in your **WatchKit App**, change the **Bundle display name** to something shorter


NotificationDemo 〉 NotificationDemo WatchKit App 〉 Supporting Files 〉 Info.plist 〉 N

- NotificationDemo
  4 targets, iOS SDK 8.3
  - NotificationDemo
    - AppDelegate.swift
    - ViewController.swift
    - Main.storyboard
    - Images.xcassets
    - LaunchScreen.xib
    - Supporting Files
  - NotificationDemoTests
  - NotificationDemo WatchKit Extension
    - InterfaceController.swift
    - NotificationController.swift
    - Images.xcassets
    - Supporting Files
      - Info.plist
      - PushNotificationPayload.apns
  - NotificationDemo WatchKit App
    - Interface.storyboard
    - Images.xcassets
    - Supporting Files
      - Info.plist
  - Products

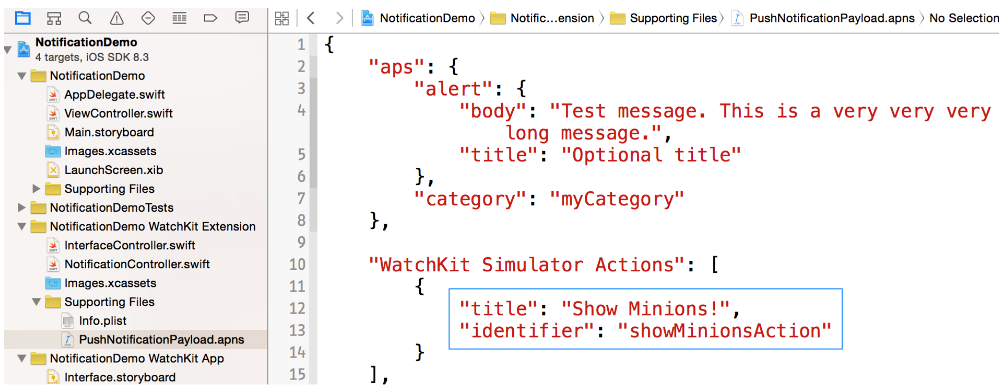
Key	Type	Value
Information Property List	Dictionary	(13 items)
Localization native development r...	String	en
Bundle display name	String	Minions!
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	com.natashatherobot.NotificationDemo.watchkitapp
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
Supported interface orientations	Array	(2 items)
WKCompanionAppBundleIdentifier	String	com.natashatherobot.NotificationDemo
WKWatchKitApp	Boolean	YES

3. Run the Notification scheme to see what the Notification looks like with this new display name!

Configuring Notification Actions

Now, let's write the code to configure the actions when the user taps a custom button in the Notification. In this tutorial, I'm going to configure an action to see what Mayhem the Minions are currently in.

1. In **PushNotificationPayload.apns**, change the WatchKit Simulator Actions **title** to **Show Minions!** and the **identifier** to **showMinionsAction**.



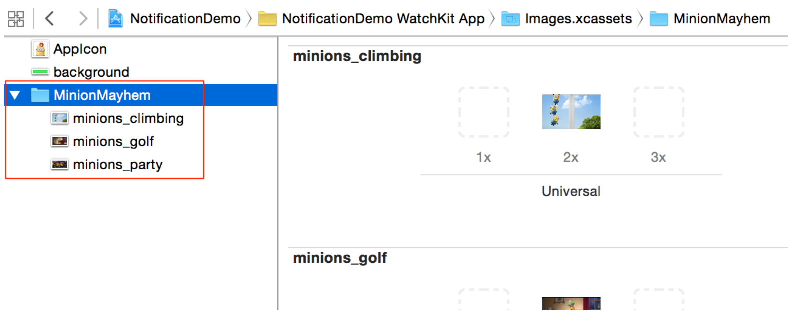
NotificationDemo › Notific…ension › Supporting Files › PushNotificationPayload.apns › No Selection

NotificationDemo (4 targets, iOS SDK 8.3)
- NotificationDemo
  - AppDelegate.swift
  - ViewController.swift
  - Main.storyboard
  - Images.xcassets
  - LaunchScreen.xib
  - Supporting Files
- NotificationDemoTests
- NotificationDemo WatchKit Extension
  - InterfaceController.swift
  - NotificationController.swift
  - Images.xcassets
  - Supporting Files
    - Info.plist
    - PushNotificationPayload.apns
- NotificationDemo WatchKit App
  - Interface.storyboard

```
{
    "aps": {
        "alert": {
            "body": "Test message. This is a very very very
                long message.",
            "title": "Optional title"
        },
        "category": "myCategory"
    },

    "WatchKit Simulator Actions": [
        {
            "title": "Show Minions!",
            "identifier": "showMinionsAction"
        }
    ],
```

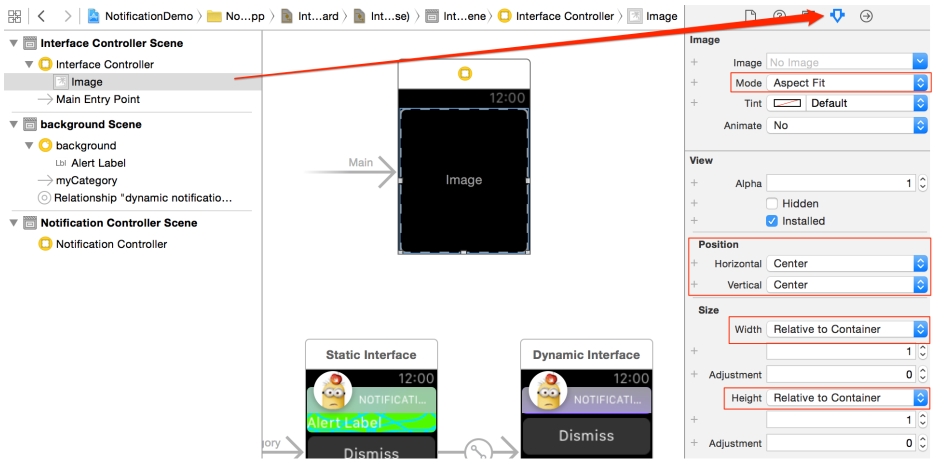
2. Add the images from the **ActionImages** folder in the tutorial folder to **Images.xcassets** in your **WatchKit App**.



NotificationDemo › NotificationDemo WatchKit App › Images.xcassets › MinionMayhem

- AppIcon
- background
- MinionMayhem
  - minions_climbing
  - minions_golf
  - minions_party

3. In the Storyboard, add an **Image** to the **Interface Controller,** and configure the layout as appropriate



NotificationDemo › No…pp › Int…ard › Int…se) › Int…ene › Interface Controller › Image

- Interface Controller Scene
  - Interface Controller
    - Image
  - Main Entry Point
- background Scene
  - background
    - Alert Label
  - myCategory
  - Relationship "dynamic notificatio…
- Notification Controller Scene
  - Notification Controller

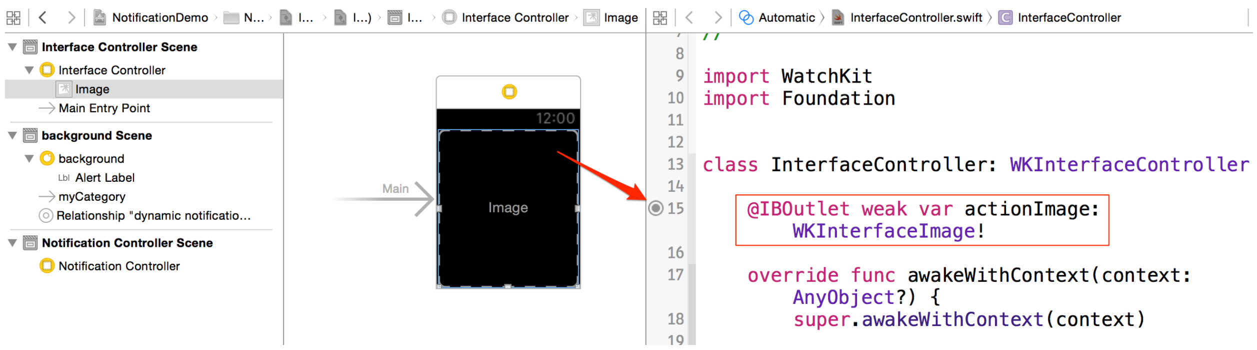
Image
- Image: No Image
- Mode: Aspect Fit
- Tint: Default
- Animate: No

View
- Alpha: 1
- Hidden
- ✓ Installed

Position
- Horizontal: Center
- Vertical: Center

Size
- Width: Relative to Container
- 1
- Adjustment: 0
- Height: Relative to Container
- 1
- Adjustment: 0

4. Create an **IBOutlet** for setting the Image in the **Interface Controller**.



```
//

import WatchKit
import Foundation


class InterfaceController: WKInterfaceController

    @IBOutlet weak var actionImage:
        WKInterfaceImage!

    override func awakeWithContext(context:
        AnyObject?) {
        super.awakeWithContext(context)
```

5. In the **Interface Controller**, override the **handleActionWithIdentifier:forRemoteNotification:** method to handle your action identifier

```
class InterfaceController: WKInterfaceController {

    @IBOutlet weak var actionImage: WKInterfaceImage!

    let actionImageNames = ["minions_climbing", "minions_golf", "minions_party"]

    override func handleActionWithIdentifier(identifier: String?,
        forRemoteNotification remoteNotification: [NSObject : AnyObject]) {

        if let identifier = identifier {
            if identifier == "showMinionsAction" {
                let randomImageIndex = Int(arc4random_uniform(UInt32
                    (actionImageNames.count)))
                actionImage.setImageNamed(actionImageNames[randomImageIndex])
            }
        }

    }
}
```

6. Run the Notification scheme. Click the **Show Minions!** button to see what the Minions are up to!