Are you up to the Challenge of Creating Apple WatchKit App for IOT?

by **Satya K Vivek** | November 07, 2018

If you're up to the challenge of creating an Apple WatchKit App, here we will provide a simple and detailed guide outlining the basic steps. The smartwatch segment of the wearables market promises to be one of the most exciting — and a key part of market growth will be driven by the applications available for the devices.

By integrating the power of Internet of Things (IoT) with that of Apple Watch, the users can take the advantage of the next level of User eXperience.

Some interesting use cases include measure your sleep activity and have the alarm wake you up, make coffee as you soon as you step out of bed, doctors can proactively diagnose that your heart is at risk and controlling the lighting in your house. These are just some obvious examples, and it's currently easiest to imagine its impact in the home and on your health and that's why Apple also launched HealthKit and HomeKit.

However building applications for smartwatches cannot simply be an exercise in taking existing applications and transferring them to a smaller screen. Rather, you'll need a complete rethink of how your customers will engage with the application, and the specific use cases in which they'll use it.

Apple Watch and watchOS

The original Apple Watch was released April 24th of 2015, becoming the best-selling wearable of the year with 4,2 million devices sold during the 2nd quarter of 2015; beating its Android competitors such as Fitbit and Xiaomi.

Since the original release of the Apple Watch, the device has rapidly been improved with updated versions. Apple Watch Series 1 and Series 2 were both released during 2016, Series 3 in 2017 and Series 4 in 2018. Each one of the Series featured updates to the previous model, such as upgraded processing power, improved memory and storage and Bluetooth connectivity. The Series 3 also featured a model with its own LTE cellular network, which gives

the opportunity to use the watch applications without using the phone's network. The Series 4, Apple just released has WatchOS 5. Included in this are a new walkie-talkie mode, improved notifications, the Podcasts app and other tweaks.

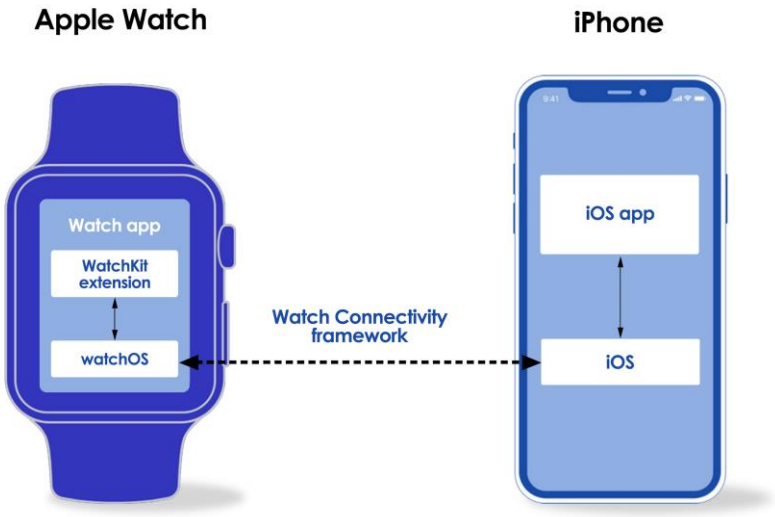
As new versions of the watchOS were released, more access for the developers was granted. Apple Watch Series 3 and Series 4 comes in two versions: GPS, and GPS + Cellular. The Series 2 had GPS but not cellular, and the Series 1 didn't have either. The cellular models are very capable, provided of course you have an airtime contract. You can talk to other compatible Watch users with the Walkie-Talkie app, make and receive phone calls and texts, use Siri and stream from Apple Music and Apple Podcasts. You can also receive notifications from some cellular-aware apps.

How the Apple Watch Works with iPhone

A Watch application cannot be installed on the Apple Watch on its own; it always has to have a paired iPhone with the application installed on it. For the Watch application to function, the watch has to be connected to the paired iPhone via Bluetooth or Wi-Fi. The communication and the connection session are established with the Watch Connectivity framework.

A WatchOS App's Architecture

The watch architecture consists of three main elements: iOS application, Watch application and the WatchKit Extension.



iOS Application

The application installed on the user's iPhone is called the parent application. iOS application installed on the iPhone that has a Watch application, will automatically be installed on the Apple Watch as well. The Watch application installed on the Apple Watch is completely dependent on the iOS application. *Some processes have to be delegated over to the parent applications since the watchOS does not have all of the same frameworks as iOS.* Sometimes the application only acts as a display for the data that has been generated at the parent application.

The parent application shares data with the Watch application through the WatchKit Extension. By creating a WCSession utilising the Watch Connectivity framework, the applications can create a connection for transferring data such as user settings and application states.

Watch Application

The Watch App is a bundle of resources that resides on the parent application and are installed on the Apple Watch. These resources include the application storyboard, image and audio files and any localisation files used for the Watch application. Within the Watch App bundle resides the WatchKit Extension.

WatchKit Extension

The WatchKit Extension is the area of the application hierarchy that connects the Watch and the parent applications together. The extension contains the code written for the Watch application. The extension uses the WatchKit framework to manipulate the interface of a Watch application: it uses the classes of the framework to configure the Watch App's elements and create responses to the user interactions.

WatchKit

WatchKit is the framework for third-party developers for creating applications for the watchOS. WatchKit is watchOS's equivalent to the iOS's UIKit framework, as they both are used to create interfaces and interaction for the application. A Watch application includes one or more interfaces which contain buttons, sliders, tables and many other visual elements. WatchKit uses the classes of the framework to configure these elements and to establish connection to the user interactions.

The WatchKit framework provides a WKExtensionDelegate protocol which is a collection of methods that can be implemented to manage app-level behaviour of the WatchKit Extension. The protocol mainly handles application states, snapshots and background tasks.

- **Application states:** WatchKit notifies changes in the application's execution state to the extension delegate object. The state changes are triggered by major events in the lifetime of the application. The WKExtensionDelegate can respond to state transition events by implementing the delegate's application lifecycle methods. These methods can, for example, be used to preload resources, configure initial user interfaces and save application states and user data.
- **Background tasks:** Background tasks give the application a small amount of time to run in the background. Several different handler types are provided for the background tasks, each for a specific type of activity. Some of these tasks can be manually scheduled by the application, while some are automatically scheduled by the system.
- **Snapshots:** Snapshots are glimpses of the application state that are displayed in the dock when the watch's side button is pressed. The system keeps the most recently used apps in the dock so that they can be resumed quickly. These apps receive priority for background tasks. The dock can hold up to 10 applications at the same time.
- **Watch Connectivity:** Watch Connectivity is a framework for establishing a two-way communication between the applications. The communication is used for data sharing. The shared data can be just small pieces of data, such as dictionaries, or entire files. The framework is an essential part for making a Watch application. The framework can also be used to update the watch's complications. The data can be sent instantly or as background transfers.

- **WCSession:** To initiate communication between the applications, both the Watch and the iOS applications have to establish communication using the WCSession class. Both applications have to set up their own sessions at some point of their execution.
- **Sending data:** Most of the time, data is sent to the counterpart device as a dictionary, most commonly in the form of [String:Any]; an exception to this being the transfer of files which include a URL path to the file. Many of the data transfer methods send the data in the background, but the data can also be sent instantly if necessary.
- **Receiving data:** When the Watch application receives background data from the parent application, it is not guaranteed that it will wake up and handle the incoming data right away. Most of the time, the application will postpone the data handling to preserve battery usage. On watchOS 4 the application will immediately handle the data if it is the frontmost application and within 10 minutes for all other applications. For older watchOS's, there is no guarantee to when the data will be processed; sometimes it will not be processed till the application is launched.

User interaction with Apple Watch

WatchOS apps can provide four different components for the user: full apps, glances, notifications, and complications.

- **Full Apps**: behave in a similar way to iPhone apps, and can have multiple screens and a range of possible interactions.

- **Glances**: are single screens of content that can be accessed by swiping up from the watch face. They don't have any interactive elements—they're only for displaying information. If the user taps on the screen, the full app is launched.

- **Notifications**: appear when the watchOS app's counterpart iOS app receives a notification. Notifications usually come from the Apple Push Notification service, but they can also be "local" notifications, which the iOS app schedules for later delivery.

- **Complications**: are small elements that are embedded into certain watch faces. They're not interactive, but they let apps add a little more information to the most quickly accessible part of the phone's interface.

Challenges

App owners should take a wise decision while preparing for a new Watch App based on the following tips:

- **App Categories**: App developers must first organise their strategy on basis of what category the proposed App fits into. By categorising your app, you are keeping a footnote for its practical purposes and usage. It also helps in smoothly slide up the process successfully into the next stage of planning. Some prominent categories that will simplify your app design process: productivity tools such as calendars and alarms, social media utilities and health and fitness app category.
- **Keep user actions simple**: The way a user interacts with the Watch app should be kept to an efficient minimum. Apple watch is all about things that would take between one and three seconds.Tracking time using simple choices such as a yes-no or start-stop actions, suits the wearable format perfectly.

- **Make a solid smartphone app**: App for Apple watch is very different from merely transporting a copy of a smartphone app to the Watch. This is an important consideration which many developers forget that Apple Watch is intended to be a

companion. Therefore, a large portion of development work needs to go into building a smartphone app that pairs seamlessly with the Apple Watch app.

- **Choose the right look**: The Apple Watch only has a small display size as compared to other devices. This means you're simply not going to be able to get as much text on the screen at the same time as you can with a mobile app. Chose the colour palate wisely which complements the available real estate.

Conclusion

The Apple Watch enables you to take a step ahead into the world of technology, no matter whether you're a tech geek or a sports addict. This dependable and versatile wearable gadget helps you organise your daily activities efficiently and achieve your full potential.

As an incredibly personal device, Apple Watch will understand each and every one of your needs, including those your brain can't recognise yet, and it will message the programmable world to address them intelligently and automatically. Gadgeon has very good experience in developing Apple Watch Apps and iOS/Android Apps that communicate with the devices for various applications like Personal health care, Home automation, Remote monitoring, Personal security systems, Climate monitoring etc.