The iOS Wi-Fi Lifecycle

This thread has been locked by a moderator.

26

For important background information, read Extra-ordinary Networking before reading this.

Share and Enjoy
—
Quinn “The Eskimo!” @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@" + "apple.com"`

The iOS Wi-Fi Lifecycle

Most developers don’t care about about iOS manages its Wi-Fi interface. However, for some apps these details matter. This post gives an overview of that process.

IMPORTANT This post covers a lot of ground, and its focus is on concepts. I’ve skipped over a bunch of details, and it’s also quite possible that I’ve got some things wrong. Lemme know if you find any particularly egregious problems.

Network Interfaces

A typical iPhone has two general-purpose network interfaces: WWAN (cellular) and Wi-Fi. Each of these has their own lifecycle.

On modern iPhones [1] the WWAN lifecycle is quite simple: The system tries hard to keep the interface up as much as possible. That’s because important system services, like push notifications, run over WWAN.

The story for Wi-Fi is more complex, *way* more complex. The rest of this post talks about Wi-Fi, and it only really covers the basics.

IMPORTANT This post is focused on iOS. The Wi-Fi lifecycle on iPad OS is similar to that of iOS; the main difference is that most iPadOS devices don’t have an WWAN interface. OTOH, the Wi-Fi lifecycle is very different on macOS, tvOS, and watchOS.

[1] Historically iOS would bring up the WWAN interface on demand. This is no longer a concern, although on-demand interfaces still exist. See Connect by name in TN3151 Choosing the right networking API.

Wi-Fi Lifecycle Basics

iOS maintains a list of known Wi-Fi networks. Each of those networks has an auto-join flag, which tells iOS whether it should try to automatically join the network. The user can control this using Settings > Wi-Fi > TheirNetworkName > Auto-Join.

There are various circumstances where iOS will try to auto-join a network. The details are not officially documented, but some common examples are:

- Waking the device by pressing the Side button
- Bringing a Wi-Fi app to the front

A Wi-Fi app is one that that has the `UIRequiresPersistentWiFi` property set in its `Info.plist` [1].

Additionally, if the device doesn’t have WWAN, or WWAN is down, iOS will work harder to join a Wi-Fi network in order to maintain important system services like push notifications.

There are various circumstances where iOS will leave a Wi-Fi network. Again, the details are not documented but common examples are:

- Device sleep, which typically happens shortly after screen lock
- After 30 minutes of not having a Wi-Fi app active

Finally, the user can:

- Manually join a network using Settings > Wi-Fi
- Temporarily leave a Wi-Fi network using Control Centre [2]
- Turn Wi-Fi on and off using Settings > Wi-Fi

[1] The behaviour of Wi-Fi apps is one of the places where I’m least confident about the accuracy of this post. What I’ve described here is most definitely the behaviour from the early days of iOS. Things may have changed. I’m working to confirm this but that’s harder than you might think.

[2] This *does not* turn off Wi-Fi. For the details, see Use Bluetooth and Wi-Fi in Control Center.

Wi-Fi and the Wider Internet

When iOS auto-joins a Wi-Fi network, it evaluates whether the network leads to the wider Internet. If not, it immediately turns around and leaves that network. While it’s doing this evaluation the Wi-Fi interface is up but it can’t become the default route.

The exact criteria used for this evaluation is not documented, but there are two obvious requirements:

- The network must have a viable IP configuration.
- The network must not be captive.

A viable IP configuration typically means that the DHCP server on the network returned an IPv4 address, subnet mask, and router. However, that’s only part of the story. For example:

- There doesn’t need to be a DHCP server at all! Remember that iOS has a list of known networks, and a known network might have a static IP configuration.
- There doesn’t need to be an IPv4 address. iOS supports IPv6-only networks.

iOS doesn’t do the viable IP evaluation if you manually join a Wi-Fi network. In that case, it’ll stay on that network until it leaves for some other reason, like device sleep.

The captive network test involves sending an HTTP request to a server at Apple [1]. The exact details are not documented but the gist is:

- If the request returns the expected information, the network is not captive.
- If the request redirects, the network is captive.
- If the request fails, the system assumes it’s not captive.

If the network is identified as a captive network, the system’s captive network support kicks in. This is super complex, *far* too complex to fully explain here. For more about this, see Hotspot Network Subsystem Programming Guide (and marvel at Figure 1-1!).

However, for the purposes of this discussion, you can think of the entire captive network process as the system taking a very long time to evaluate whether the network leads to the wider Internet. So, the interface remains up but it can’t become the default route.

[1] This is the old school test. A network can explicitly advertise its captive status using the techniques described in How to modernize your captive network.

Network

Reply

Posted 2 days ago by eskimo

Add a Comment