App Notes For iOS Wi-Fi Setup

03/22/2017

This document provides details on how to implement Wi-Fi setup in applications using methods provided in the Ayla iOS SDK, and outlines common issues seen during the Wi-Fi setup process. AylaSetup class in Ayla SDK provides methods to connect an Ayla device to the Ayla cloud. Sample code for using AylaSetup is provided in *SetupViewController* in the Aura app.

Connect Phone To Device AP

When a new device requires Wi-Fi Setup or a registered one fails to connect to a known Wi-Fi Network, it will enable its own AP. The setup is performed directly once connected to this AP so the first step is to direct the user to the Wi-Fi configuration in iOS Settings app, a way to do this is to use the URL scheme for the settings app:

Swift

```
UIApplication.shared.openURL(URL(string:UIApplicationOpenSettingsURLString)!)
```

This will take the user to your app settings, from there they can move back to the see all the Settings and then into the Wi-Fi section where the user should select the AP corresponding to the device to be configured. After the user selects the AP the settings app will display the Captive Portal interface of the device which will attempt to redirect the user to the app.

Since this process involves a high degree of user interaction it's prone to fail at one of the described stages. Some of the problems the process might encounter are listed below, along with suggestions on how to mitigate them.

- The user could select another Wi-Fi network. Make sure to clearly tell the users the SSID they should find and connect to.
- The Captive Portal interface could take some time to show up or not show up at all. Sometimes this requires the user to disconnect from device AP and retry. This information could be added to a troubleshooting page.
- App settings ask for user permission to redirect to app, so it could be prevented by the user! Inform the user to return to the app in case of a successful connection of the device to the Ayla Cloud, but a failed redirection.

Create An AylaSetup Instance

Once the user has returned to the app and while still connected to the device AP, you'll require an AylaSetup instance. This instance will communicate to the device over its Wi-Fi network.

Swift.

```
let setup = AylaSetup(sdkRoot: AylaNetworks.shared())
```

Optionally, if your device supports updates of Wi-Fi connection process, you designate a listener for Wi-Fi State changes from the device:

```
setup.addWiFiStateListener(self)
```

Be sure to indicate your class conforms to `AylaDeviceWifiStateChangeListener` and implement the method `wifiStateDidChange` which will be called if the device has an update. Bear in mind, these updates require the device to support them. For other devices you'll get an "unknown" state.

You can also get the Setup status from the SDK point of view via the `status` property in the `setup` object:

```
@property (nonatomic, readwrite) AylaSetupStatus status;
```

See the appledoc of `AylaSetupStatus` for more information on the different status available.

Connect Via SDK To Device

Once the phone is connected to the device and user has returned to the app, a secondary connection will be established between the SDK and the device to communicate.

The SDK will attempt to locate the device LAN IP using mDNS. Should the SDK fail to determine the device IP address, it will resort to the `fallbackDeviceLANIP` property configurable via `AylaSystemSettings`.

At this point, the key exchange will take place enabling Secure Wi-Fi Setup. Once the device and SDK are connected, the method will return with a reference to the `AylaSetupDevice`. Note that `AylaSetupDevice` is also accessible from the `setupDevice` property in `setup`.

Initiate AP Scan

After connecting to the device via the SDK, the user needs to select from the list of APs visible to the device:

Swift

```
let task = setup.startDeviceScan(forAccessPoints: {
            self.fetchCurrentAPList()
            }) { (error) in
                let message = "Wi-Fi Scan Failed"
                self.updatePrompt(message)
                self.displayError(error as NSError, message: message)
        }
```

With the new error handling model in Swift 3 it's useful to cast as NSError in case of failure to access more information such as error `code` and `userInfo`.

Fetch AP List

The next step after requesting the device to start scanning for nearby networks is to fetch the list. Although it's possible to request the AP list from the device immediately after instructing it to start scanning for them, it's recommended to wait a few seconds to make sure it retrieves the latest scan results. Alternatively you can provide a way to refresh the list by fetching the device Access Points again. The list of AP can be obtained with the `fetchDeviceAccessPoints` method:

Swift

```
setup.fetchDeviceAccessPoints({ (scanResults) -> Void in
            self.scanResults = scanResults
            //display AP List
            }, failure: { (error) -> Void in
                //handle error
        })
```

This method will return an instance of `AylaWifiScanResults`, containing an array of `AylaWifiScanResult` in the `result` property, each result contains information on an AP for the user to select the one the one to which the device should connect.

Connect to the AP

Once the user has selected an AP, they must provide the credentials to connect to it. Performing this API securely is supported on every device, even if it doesn't support securing all other Wi-Fi Setup API calls.

Swift

```
setup.connectDeviceToService(withSSID: ssid, password: password, setupToken: token!, latitude: 0.0, longitude: 0.0, success: { (wifiStatus) -> Void in
            // Call to confirm connection.
            self.confirmConnectionToService()

            }) { (error) -> Void in
                var message = "Unknown error"

            if let errorCode = AylaWifiConnectionError(rawValue: UInt16(error.aylaResponseErrorCode)) {

                    switch errorCode {
                    case .connectionTimedOut:
                        message += "Connection Timed Out"
                    case .invalidKey:
                        message += "Invalid Wi-Fi Key Entered"
                    case .notAuthenticated:
                        message += "Failed to Authenticate"
                    case .incorrectKey:
                        message += "Incorrect Wi-Fi Key Entered"
                    case .disconnected:
                        message += "Disconnected from Device"
                    default:
                                message += "AylaWifiConnectionError Code \(errorCode.rawValue)"
                    }
                }
                self.updatePrompt("Setup Failed")
                self.displayError(error as NSError, message: message)
        }
```

```
}
```

Along with this call you can specify the location of the device, that due to the primitive nature of the parameters in objective-c cannot be made nullable. Therefore, to omit them give them a value of 0.0 ((double)zero).

If the device supports AP-STA mode (the device can be both a WiFi AP and a WiFi Station/Client), the mobile remains connected to the device's AP even after this method finishes. At this point, the designated `AylaDeviceWifiStateChangeListener` will receive calls to `wifiStateDidChange` with updates of the connection process from the device.

The mobile phone disconnecting from the Ayla device does not affect the device connecting to the Ayla Device Service (ADS). The app should therefore proceed to confirm whether the device connected to ADS even if a network error is received.

After sending the credentials to the device, the SDK will attempt to confirm the connection to AP has taken place and at that point will send the signal to the device to stop AP mode. However, this doesn't mean the device has connected to the Ayla cloud. A confirmation call is still required to enable communication with the device over ADS.

Reconnecting The Phone To the Original AP

iOS doesn't provide a way to manage wireless connections programmatically. Therefore, after the device stops its AP, the phone OS will attempt to reconnect to a known SSID. During this process many things can go wrong.

- If you have added an `AylaConnectivityListener` to the `connectivity` property in `AylaNetworks` class, you'd expect to receive a `connectivity: didObserveNetworkChange:` call, however, if the AP switching happens too fast in the OS it would fail to notify the change occurred. Therefore, do not rely on this delegate method to determine when to poll for device connection to ADS or some other stage requiring the device has stopped its AP Mode and the phone has reconnected to the original network.
- If the phone has been connected to multiple APs within the same area, according to Apple documentation it would attempt to reconnect to a wireless networks following a set of rules described here: https://support.apple.com/en-us/HT202831 However, due to all the factors involved in the process, it's possible the phone will reconnect to a different network. Keep this in mind if you attempt to register a device automatically right after performing Wi-Fi Setup (For instance, on a device with same-lan registration where both the phone and the device MUST be on the same Wi-Fi network).

Confirming Device Connectivity To Ayla Device Service (ADS)

To determine if the device is available and ready to use, it's necessary to confirm the device has connected to ADS. This is achieved via confirmDeviceConnected API:

Swift

```
setup.confirmDeviceConnected(withTimeout:
defaultCloudConfirmationTimeout, dsn:(deviceDSN)!, setupToken:token!,
success: { (setupDevice) -> Void in
                // Wi-Fi Setup Complete
            }) { (error) -> Void in
                let message = "Confirmation step has failed."
                self.displayError(error as NSError, message: message)
        }
```

On successful return of this call, the IoT device registration type, DSN, and LAN IP Address are returned. These values are useful during the device registration process which typically immediately follows Wi-Fi setup.

Exit Setup

Unlike the Android SDK, it's not required that you call exit on setup to finish the process.

Flow Diagram

https://docs.google.com/a/aylanetworks.com/drawings/d/1roBPWEbwAwHCWveC-5BNaBVh23-PhUVd7CMOCzzK640/edit?usp=sharing

Glossary of Terms

ADS - Ayla Device Service

AP Mode - The device acts as an access point, and allows mobile phone to connect to it.

STA Mode - The device is connected to a Wi-Fi access point.

AP-STA mode - The device supports simultaneous AP and Station mode. The mobile phone can remain connected to the device AP even after the device has connected to another Wi-Fi access point.

DSN - Ayla assigned unique Device Serial Number

Latitude (include the value to be used, to ignore/skip setting)

Longitude (include the value to be used, to ignore/skip setting)