Developers Guide

# Ayla Local Devices Developers Guide

iOS & Android

Version: 1.0
Date Released: July 12, 2017
Document Number: AY006DLD2-1

Ayla Networks

## Copyright Statement

## Trademarks Statement

## Referenced Documents

Ayla Networks does not supply all documents that are referenced in this document with the equipment. Ayla Networks reserves the right to decide which documents are supplied with products and services.

## Contact Information

Ayla Networks TECHNICAL SUPPORT and SALES

Contact Technical Support: https://support.aylanetworks.com
or support@aylanetworks.com

Contact Sales: https://www.aylanetworks.com/company/contact-us

Ayla Networks REGIONAL OFFICES

| | HEADQUARTERS | |
|---|---|---|
| Chicago | Silicon Valley | Boston |
| 10 N. Martingale Road, Suite 400 | 4250 Burton Drive, Suite 100 | 275 Grove Street, Suite 2-400 |
| Schaumburg, IL 601073 | Santa Clara, CA 95054 | Newton, MA 02466 |
| | Phone: +1 408 830 9844 | |
| | Fax: +1 408 716 2621 | |

# Table of Contents

# 1 Introduction

This document is directed towards mobile application developers who are writing an Ayla-connected mobile application that interoperates with devices that do not have their own persistent connection to the internet.

## 1.1 About this Document

With the information and links provided in this document, iOS and Android developers can create apps that remotely interact with IoT devices.

## 1.2 Intended Audience

Developers for iOS and Android applications that connect with IoT devices.

Application developers can use the general guidelines and links in this document to develop apps for iOS and Android compute devices.

## 1.3 Related Documentation

Related documents and resources include:

- OEM Dashboard User Guide (AY006UDB3)
- Developer Portal User Guide (AY006UDP3)
- AURA app and Ayla SDK (Both are available on GitHub for registered OEMs who have signed the Ayla software license agreement.)

## 1.4 Customer Support

Support is available at http://support.aylanetworks.com. The support site provides access to the knowledge base. You can also create a support ticket for additional help.

## 1.5 Abbreviations and Acronyms

Most abbreviations and acronyms are spelled out the first time that they are used in the document and are not always listed in the following table. However, abbreviations and acronyms that are frequently used in Ayla user documentation or commonly known by the intended audience of the document may also or only be spelled out in this table.

- iOS – the Apple phone operating system
- Android – the Google phone operating system

# 2 Access to Development Tools

To access the iOS and Android tools and development resources, you need to:

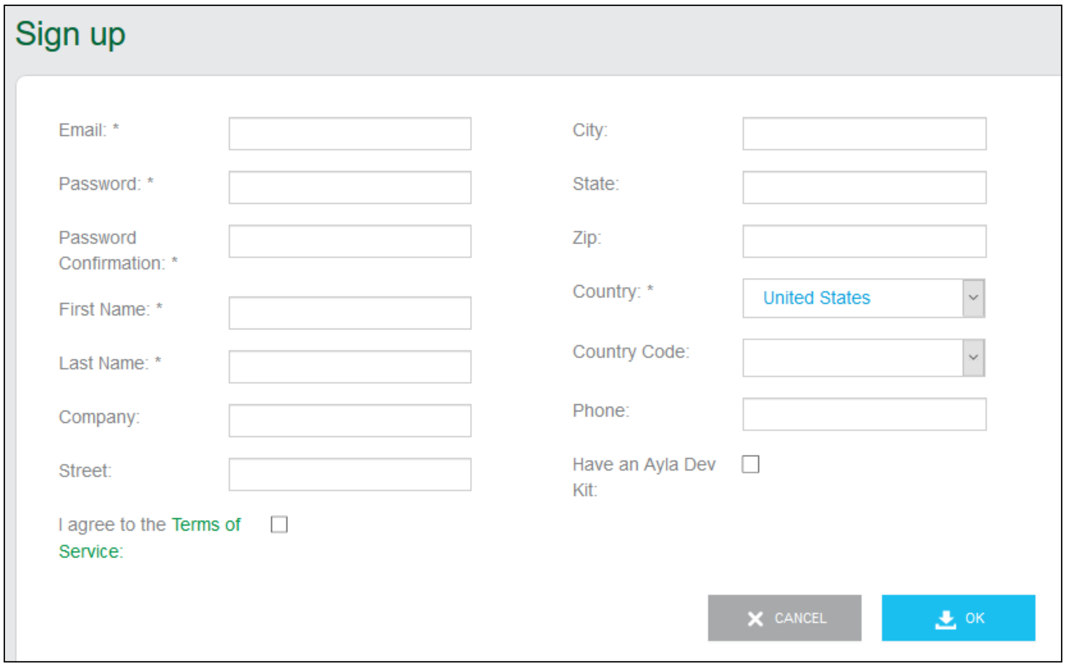
1. Get an Ayla Developer account.
2. Get a GetHub account.
3. Request access to Ayla development tools

## 2.1 Sign-up for a Ayla Developer Account

| NOTE | If you have an Ayla Developer account, you can skip this procedure. |
|---|---|

To get an account:

1. Go to https://developer.aylanetworks.com/registrations/new



Sign up

| | | | |
|---|---|---|---|
| Email: * | | City: | |
| Password: * | | State: | |
| Password Confirmation: * | | Zip: | |
| First Name: * | | Country: * | United States |
| Last Name: * | | Country Code: | |
| Company: | | Phone: | |
| Street: | | Have an Ayla Dev Kit: | ☐ |
| I agree to the Terms of Service: | ☐ | | |

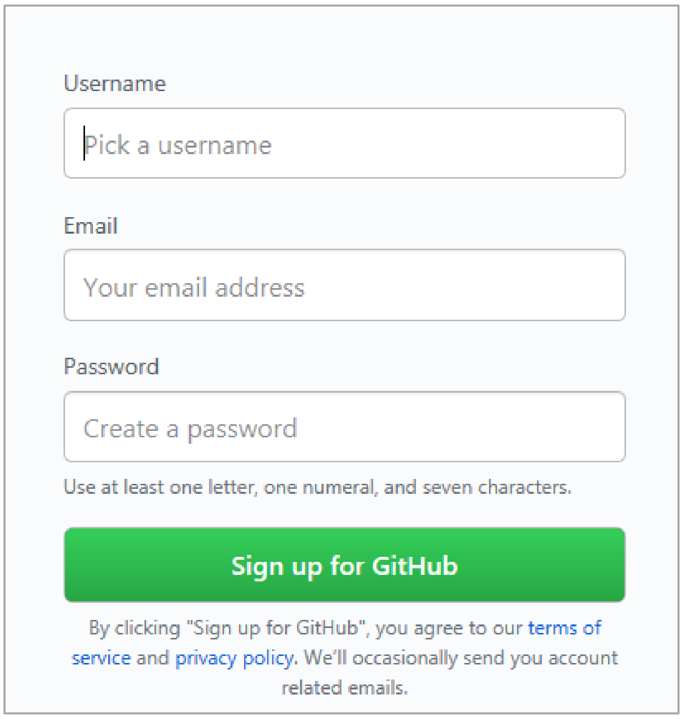
2. On the Sign up page, enter required and appropriate optional details, and then click **OK**.
3. You will receive an email confirmation.
4. Review the details and click the link to confirm the account.

## 2.2 Sign-up for a GitHub account

| NOTE | If you have a GitHub account, you can skip this procedure. |
|---|---|

If you don't already have a GitHub account:

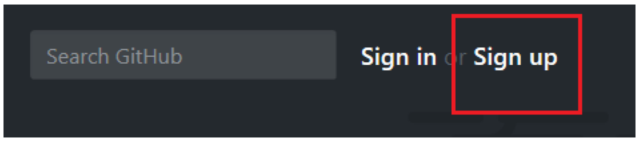
1. Go to https://github.com/


Username
Pick a username
Email
Your email address
Password
Create a password
Use at least one letter, one numeral, and seven characters.
Sign up for GitHub
By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We'll occasionally send you account related emails.

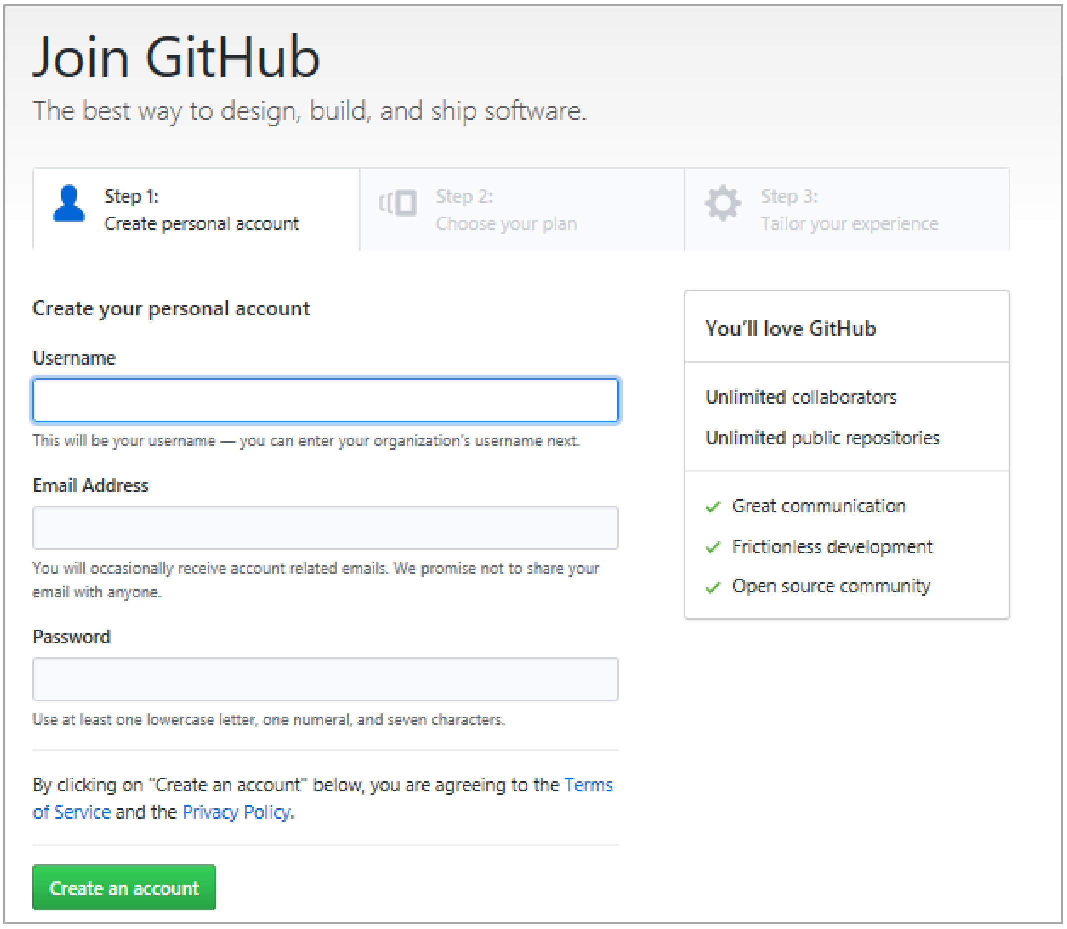
2. Enter details in the form and click **Sign up for GitHub**.

### Alternate sign-up

1. Go to [https://github.com/](https://github.com/)
2. Click the **Sign up** link.



3. Complete the form.



Join GitHub

The best way to design, build, and ship software.

Step 1: Create personal account | Step 2: Choose your plan | Step 3: Tailor your experience

Create your personal account

Username

This will be your username — you can enter your organization's username next.

Email Address

You will occasionally receive account related emails. We promise not to share your email with anyone.

Password

Use at least one lowercase letter, one numeral, and seven characters.

By clicking on "Create an account" below, you are agreeing to the Terms of Service and the Privacy Policy.

Create an account

You'll love GitHub

Unlimited collaborators

Unlimited public repositories

✓ Great communication

✓ Frictionless development

✓ Open source community

## 2.3 Request Access to Ayla Mobile Documentation

| | |
|---|---|
| NOTE | If you have access to Ayla's GitHub > Mobile Documentation repos, you can skip this procedure. |

**Prerequisites**

Before access can be provided to the Customer or Prospect to complete, the following must be completed beforehand:

- Account in SFDC (GiHub)
- Opportunity and opportunity's details in SFDC
- mNDA completed and upload to the Opportunity in SFDC

If you don't already access to the GitHub > Ayla Mobile Documentation account:

1. Send an email request to: Support@aylanetworks.com, Subject: Access to Mobile SDK repo & Aura repo
2. When your request is received and processed, you will get a return email with a link to the Software License Agreement (SLA)

3. Read and sign the SLA.
4. You will receive an email that provides a link to the Ayla repos.

# 3 Access and Create Local Device Solution

The Ayla repositories contain details needed to develop a Local Device application.

## 3.1 Coding Preparation

Local devices need some initial configuration within the Ayla network before development can begin. Please contact your Ayla support representative to discuss these requirements should you have any questions.

### 3.1.1 Hardware unique identifier

Devices on the Ayla network, local or otherwise, must have a means of being uniquely identified. A serial number or other unique identifier must be present and readable by the mobile application before the device can be registered on the Ayla network.

### 3.1.2 Local device OEM access

Local device support must be enabled for your OEM. Talk to your Ayla representative to ensure that your OEM account has been granted access to the Local Device functionality before beginning mobile app development.

## 3.2 AylaLocalDevice Class

The AylaLocalDevice class is a subclass of AylaDevice that also exposes interfaces required for local device control and registration.

AylaLocalDevices contain the following additional fields and methods:

| Field | Description |
|---|---|
| `hardwareAddress` | Unique string that identifies this device. Each device instance must have a different identifier in this field. |
| `- (AylaGenericTask*) connectLocalWithSuccess:failure:` | Connects the mobile application to the device locally (e.g. via Bluetooth, or other protocol) |
| `- (AylaGenericTask*) disconnectLocalWithSuccess: failure:` | Disconnects the mobile application from the device |
| `isConnectedLocal` | true if the device is locally connected; false otherwise |

| Field | Description |
|---|---|
| `- (nullable id)valueForProperty:` | Called by the SDK to obtain the current value for a property. The mobile application should obtain this value from the device. |
| `- (nullable AylaGenericTask*) setValue: forProperty:success:failure:` | Called by the SDK to set the current value for a property. The mobile application is responsible for delivering the new value to the hardware. |
| `requiresLocalConfiguration` | True if the device requires additional setup before it can be connected. |

AylaLocalDevices are configured to intercept calls made to their properties, allowing the device object to have control over the reading and writing of values to the device. This is accomplished through AylaLocalProperty objects. AylaLocalDevices manage AylaLocalProperties instead of normal AylaProperties. AylaLocalProperties convert calls to create or fetch datapoints into calls to the AylaLocalDevice object to set or read property values.

### 3.2.1 AylaLocalDeviceManager

In order for the SDK to know what device classes to use for each managed device, developers must implement and install an instance of a AylaDeviceClassPlugin object. The AylaDeviceClassPlugin is called when devices are received from the cloud service, and should return the type of class that should be instantiated given a particular device's model and OEM model values.

Additionally, local devices need to provide a method of discovering new devices and registering them. BLE devices have some of this work already done in the AylaBLEDeviceManager class, which is used as a base in Aura for the AuraLocalDeviceManager class. This class in Aura handles both the LocalDeviceManager interface to provide support for discovery and registration, as well as the DeviceClassPlugin to provide support for the Ayla BLE Demo Board device.

### 3.2.2 Local Device software updates

Software updates may be issued to local devices. The image for update is prepared and uploaded to the Ayla Cloud service, and an update command is generated for the devices that need to be updated.

When the local device is connected, the Ayla SDK will fetch pending commands from the cloud. If a software update command is pending, the SDK will automatically fetch the update file, verify the checksum and then call back the application via the device's onOTAReceived(command, path) method.

Once the image has been received, the developer is responsible for performing the actual update on the local device, as the details of this operation will vary from device to device.

When the local device has been updated, or failed to update, the application is responsible for acknowledging completion of the command by calling the local device's setOTAStatus(status, commandId) method.

## 3.3 Bluetooth LE Devices

Included in the Ayla SDK is an implementation of AylaLocalDevice for Bluetooth LE devices called AylaBLEDevice. This implementation includes methods to obtain Aylaspecific

information from devices that support the Ayla GATT profile, though it is not required that the device support this.

AylaBLEDevice provides implementations of the required methods of AylaLocalDevice in a Bluetooth LE context, as well as helper methods to help deal with Bluetooth LE communication.

### 3.3.1 AylaGenericThermostat

Included in Aura is an example of an implementation of an AylaBLEDevice, AylaGenericThermostat. This class is designed to set up and communicate with a virtual BLE thermostat, and is used as an example of a working implementation of an AylaBLEDevice.

The following methods of AylaGenericThermostatCandidate are of particular interest, and should be studied as examples when developing a BLE device for the Ayla network:

| Field | Description |
|---|---|
| `servicesToDiscover` | Unique string that identifies this device. Each device instance must have a different identifier in this field. |
| `- (nullable NSArray<CBUUID *>*)vendorCharacteristicsToFetchForService:` | Returns a list of names of characteristics that are managed by the device. All properties that can be read or written by the application should be listed here. |
| `valueForProperty: / setValue:ForProperty:` | Called by the framework to read or write values to the device |
| `- (void) didUpdateValueForVendorCharacteristic:error:` | A method called when an indication or notification comes in. Implementers should update property values as appropriate. |

## 3.4 Aura Implementation

The Aura sample application that ships with the Ayla Mobile SDK contains support for the AylaBLEDemoBoard, including registration. To discover Ayla BLE devices within Aura, select "Local" as the registration type in the Registration screen and scan for devices. Devices supporting the Ayla GATT profile will be discovered and may be registered in Aura, as long as the account has Local Device access enabled. Support for ther devices can be added if an identifying service and candidate are registered in createLocalCandidate method of AuraLocalDeviceManager.

# 4 GitHub Repositories

## 4.1 Android Repos

There are two repositories for Android:

- https://github.com/AylaNetworks/Android_AylaSDK
- https://github.com/AylaNetworks/Android_Aura

The README.md file in each folder provides instructions and where more information is available.

## 4.2 iOS Repos

There are two repositories for iOS:

- https://github.com/AylaNetworks/iOS_AylaSDK
- https://github.com/AylaNetworks/iOS_Aura

The README.md file in each folder provides instructions and where more information is available.

## 4.3 Build App

Follow the instructions to create the app.

Generate local HTML file that contains API reference.

## 4.4 On Ayla Developer Portal

NOTE Read the Ayla Developer Portal Users Guide.

Create device templates.

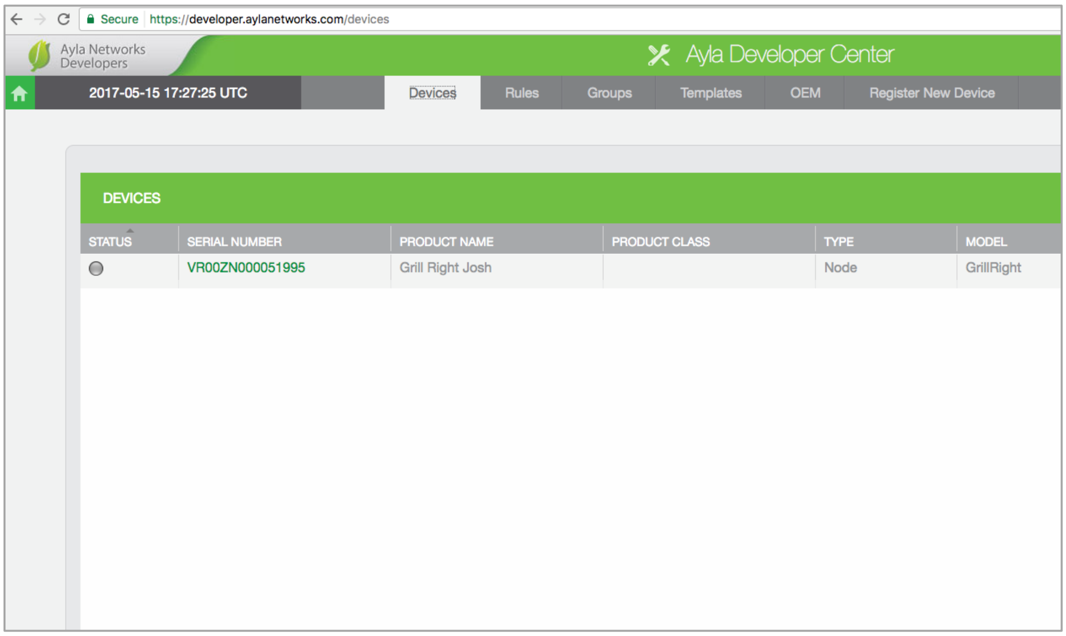
Each BLE device profile should map to a unique template.

## 4.5 Code & Test

Code the mobile app solution and test functionality.

### 4.5.1 Developer Portal Notes

LocalDevices connected to Ayla's service show up on the Developer Portal with a gray circle.



| STATUS | SERIAL NUMBER | PRODUCT NAME | PRODUCT CLASS | TYPE | MODEL |
|---|---|---|---|---|---|
| ● | VR00ZN000051995 | Grill Right Josh | | Node | GrillRight |

NOTE Local Devices on the Ayla Developer Portal have a grey circle even if they are connected and transmitting.

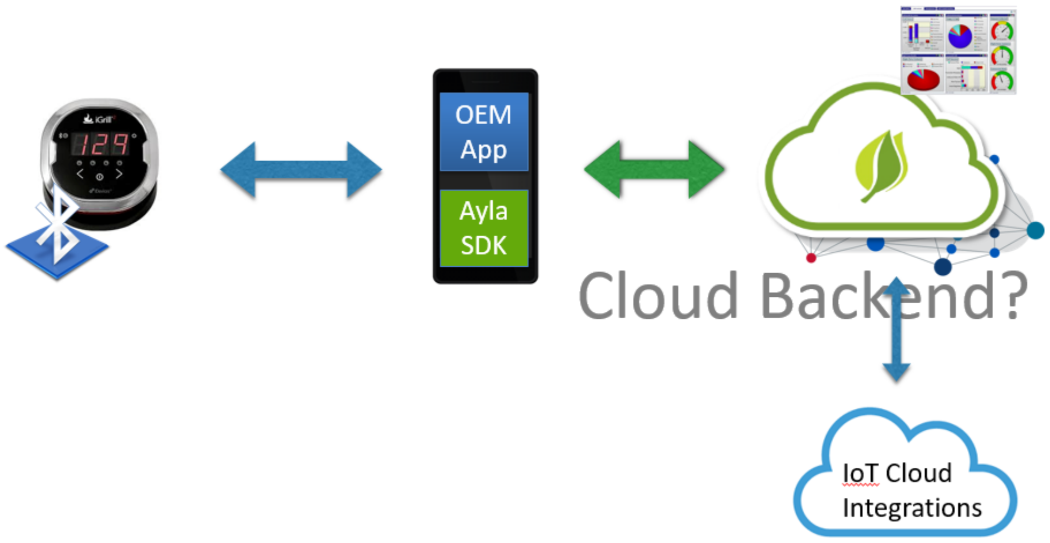
### 4.5.2 Reference Code

Ayla provides Local Device reference code in the Aura application (available in the Github Repo). This reference code is built around the Grill-Right meat thermometer device sold by Oregon Scientific (Model AW133).

# Appendix A – PaaG Example

The Phone as a Gateway is a Local Device application. PaaG is an add-on features, available for approved OEMs. This control is available with an attribute “phone_as_gateway”.

The concept is presented in this graphic.



A PaaG application enables the mobile device that connects to the Ayla Cloud via a native app. The setup handle a mobile device RF connection; device registration with the Ayla cloud that enables continued secure communication between the device and the cloud when the mobile application is running (device foreground or background).

With PaaG, each local device has a unique hardware id. This hardware id is mandatory in the new “discover” API. Cloud service uses this ID to determine if the newly discovered device has been registered. When a device is discovered, a DSN is linked to this local device and is unchangeable.

Ayla Networks

4250 Burton Drive, Santa Clara, CA 95054
Phone: +1 408 830 9844
Fax: +1 408 716 2621