User Manual

Ayla Local Devices

Developers' Guide for iOS and Android platforms

Version: 5.5

Date Released: March 23, 2017

Document Number: TBD

Ayla Networks

Table of Contents

Introduction

The Ayla network has traditionally required that devices be able to communicate directly with the Ayla cloud servers via a connection to the Internet. As more devices come to market that rely on Bluetooth LE, NFC or other means of communication, the need to manage these devices within an IoT ecosystem has grown significantly. Ayla Local Devices, new to the 5.5 release of the Ayla Mobile SDK, allow devices without persistent Internet connections to join the Ayla family of connected devices.

1.1. Audience

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.

1.2. Related Documentation

OEM Dashboard User Guide (AY006UDB3)

Developer Portal User Guide (AY006UDP3)

AURA app and Ayla SDK (Both are available on GitHub for registered OEM's who have signed the Ayla software license agreement.)

1.3. 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.

2. Before the coding begins

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.

2.1. 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.

2.2. 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. AylaLocalDevice

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:

`unique_hardware_id`	Unique string that identifies this device. Each device instance must have a different identifier in this field.
`connectLocal()`	Connects the mobile application to the device locally (e.g. via Bluetooth, or other protocol)
`disconnectLocal()`	Disconnects the mobile application from the device

`isConnectedLocal()`	Returns true if the device is locally connected; false otherwise
`GetValueForProperty()`	Called by the SDK to obtain the current value for a property. The mobile application should obtain this value from the device.
`setValueForProperty()`	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()`	Returns 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.

3.1. 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 DeviceClassPlugin object. The DeviceClassPlugin 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. 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.

4. 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 Ayla-specific 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 on Android.

4.1. AylaBLEDemoBoard

Included in the SDK is an example of an implementation of an AylaBLEDevice, AylaBLEDemoBoard. 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 AylaBLEDemoBoard are of particular interest, and should be studied as examples when developing a BLE device for the Ayla network:

`onServicesDiscovered`	Called by the framework when BLE services have been discovered. The application should save any GATT services here to use to communicate with the device
`getManagedPropertyNames`	Returns a list of names of properties that are managed by the device. All properties that can be read or written by the application should be listed here.
`getValueForProperty / setValueForProperty`	Called by the framework to read or write values to the device
`isPropertyReadOnly`	Returns true if a property can only be read and not written
`getCandidateJson`	Returns the JSON required to register the device on the Ayla cloud service. The contents of this structure depend on the template that was created on the Ayla cloud service for this device.
`getCharacteristicsToFetch`	Called by the framework after establishing a connection with the BLE device. This method should return the contents of the superclass (if the device supports the Ayla GATT profile) as well as any additional characteristics that should be read. Only after all of the returned characteristics have been read will the

	call to connectLocal() return, ensuring that by the time connectLocal() has returned these characteristics will be available.
`onCharacteristicChanged`	A callback from the Bluetooth LE subsystem, this method is called when an indication or notification comes in. Implementers should update property values as appropriate.

5. 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.