Agile Link Architecture Specification

v0.3 2014.01.15

# Overview

The Agile Link SDK is a software development kit provided by Ayla Networks to manufacturers in order to aid in the development of mobile applications utilizing the Ayla service network.

The goal of the Agile Link SDK is to make it as easy as possible for developers to authenticate with the Ayla network, obtain and manage networks of connected devices, and allow the command and control of these devices.

# Architecture goals

The primary goal of producing this SDK is to provide a starting point for developers to produce iOS and Android applications with minimal effort. While the types and capabilities of devices connected to an Ayla network can vary widely, there are many tasks that need to be performed on all of them:

## Agile Link Functionality

* User authentication / login
* Device discovery / setup
* Log out / cache clearing
* Polling the list of connected devices
* Polling the state of each connected device
* Adding / removing devices from the collection
* Enabling / disabling LAN mode as appropriate
* Device management UI
  + Display a list of all connected devices
  + Display a filtered list of all connected devices (favorites, etc.)
  + Display details of a user-selected device
  + UI to remove a device
  + UI to add a device
  + UI to remove a device
  + UI to change the state of a device
* Handling connectivity changes (app-modal message when network is not accessible)

## Environment

Setting up the development environment should be a simple task. To achieve this, dependencies should be managed by the build system (Gradle for Android) or by an external tool such as CocoaPods (iOS).

The Android version of the app should support both Eclipse and Android Studio so as not to force development in a certain environment. It should support Android version 4.2+ (JellyBean).

The iOS version of the app should support Xcode 6.1 and iOS 7 at a minimum.

# Software Architecture

To facilitate ease of development, the Agile Link SDK will be distributed as a functional application built to use The sample application will use / derive from the classes provided in the Agile Link SDK as an example of how a developer might use the SDK to implement a client’s own devices.

The system can be divided into several components:

* Session Manager
  + Store all configuration parameters
  + Handle user login
    - Local authentication (LAN mode only, property updates)
  + Handle log out / cache cleanup
* Device Manager
  + Fetch / poll list of devices
  + Enter LAN mode if appropriate
  + Handle adding devices
  + Handle removing devices
  + Handle device groupings (favorites, etc.)
    - Groups, bindings, scenes for Zigbee
  + Poll device statuses for changes
  + Triggers / Trigger Apps
  + Schedule support
* Device Object
  + Base object class, meant to be derived from
  + Contains AylaDevice object
  + Returns list of properties to be polled by Device Manager
  + Provides UI elements for list views, grid views and detail views
  + Derived classes can support additional functionality / properties / etc
* Sample UI
  + Login screen
  + Device list
  + Favorites list
  + Add device
  + Remove device
  + Device details page

## Object Details

The following sections describe in detail the functionality and interfaces of the system objects. For ease of reading, pseudocode is used to define APIs or notifications in a platform-independent manner.

### Notifications

On Android platforms, objects can be notified by implementing a listener interface and registering themselves with the appropriate system object.

On iOS devices, objects can be notified by registering for notifications via the NSNotificationCenter.

## Session Manager

The Session Manager is a static / singleton object used to initiate a login session.

#### Interfaces

void startSession(SessionParameters params)

void stopSession()

DeviceManager deviceManager()

SessionParameters sessionParameters()

#### Notifications

void sessionStarted(error error)

void lanModeEnabled(Boolean enabled)

## Session Parameters

This class contains configuration information required to start a session

#### Members

context (Android only, needed for resources, etc.)

deviceSsidRegex

appVersion

pushNotificationSenderId

appId

appSecret

username

password

serviceType (= AML\_STAGING\_SERVICE)

loggingLevel (=AML\_LOGGING\_LEVEL\_ERROR)

DeviceCreator deviceCreator

#### Interfaces

## DeviceCreator

Device deviceForAylaDevice(AylaDevice aylaDevice)

The DeviceCreator object implements deviceForAylaDevice. This method is called by the DeviceManager to create user-defined Device objects for each AylaDevice returned by the service. This allows the user to define his own classes that will be managed by the application framework.

## Device Manager

The Device Manager is obtained from the Session Manager once login has successfully completed.

#### Methods

Gateway getGatewayDevice()

Array<Device> deviceList()

Array<Device> getFilteredDeviceList(filter function)

Boolean isLANModeEnabled()

void setDeviceListPollInterval(int timeInMs)

void setDeviceStatusPollInterval(int timeInMs)

#### Notifications

void deviceListChanged()

void deviceStatusChanged(Device changedDevice)

## Device : AylaDevice

The Device object is a base class representing the common properties of a device connected to the network. End users should create new class objects derived from the Device class that contain device-specific information.

Creation of device objects are handled by the deviceCreator method passed in to the Session Manager via the Session Parameters. This allows the framework to create and manage devices of the object type desired by the implementer.

#### Methods

AylaDevice getDevice()

void updateStatus()

AylaProperty getProperty(String propertyName)

// UI methods

View getListItemView(Context context, View convertView, ViewGroup parent)

View getGridItemView(Context context, View convertView, ViewGroup parent)

Fragment getDetailsFragment(Context context)

String toString()

String getDeviceState();

ArrayList<String> getPropertyNames()

Implementers of the Device class should pay particular attention to these methods:

updateStatus()

This method is responsible for fetching information about the device’s status. It will be called whenever the device manager status timer is called. The default implementation fetches properties (returned from getPropertyNames()). Custom devices may require additional functionality, which should be implemented in an override of this method.

toString()

This method is called to determine the text displayed in the default list view or grid view. The default implementation returns the friendly name of the device.

getDeviceState()

This method is called along with toString to provide additional information about the state of the device, such as “ON”, “OFF”, “Open”, “Closed”, etc. It is optional, and defaults to an empty string.

getPropertyNames()

This method should be overridden to add properties to be fetched during device status updates.

getListItemView()

getDetailsFragment()

getGridItemView()

These methods can be overridden to replace or augment the view objects responsible for displaying the device in a list, grid or full-screen detail view. The default implementations return simple views configured with the results of toString() and getDeviceState(). Implementers should create custom views that contain device-specific information or controls to be presented to the user.

## Gateway : Device

The Gateway object is derived from the Device object, and contains additional interfaces used to query gateway-owned devices or to configure the gateway.

#### Interfaces

Array<Device> getNodes()

# Building Agile Link

## Android Studio

git clone https://github.com/AylaNetworks/Agile\_Link\_Android.git

cd Agile\_Link\_Android

mkdir libraries

cd libraries

git clone https://github.com/AylaNetworks/Android\_AylaLibrary.git

cd Android\_AylaLibrary

git checkout -b zigbee\_gradle origin/zigbee\_gradle

Then open Android Studio and select "Open existing project"

Open build.gradle in the project root directory.

## Eclipse

[TBD]

## iOS

[TBD]