Developer’s Guide

Ayla Mobile SDK Developer’s Guide

Version: 5.0

Date Released: May XXX, 2016

Document Number: AY006SDK4-5.0

Ayla Networks

Table of Contents

Introduction

This document describes the Ayla Mobile SDK.

Audience

This document is for developers.

Related Documentation

Related documentation is located in the GitHub. Note: A license agreement is required for access.

Customer Support

Support and customer documentation is located at http://support.aylanetworks.com

Overview

The Ayla Mobile SDK is designed to provide mobile application developers on the iOS and Android platforms a way to easily interact with the Ayla Cloud Service and connected devices. The 5.0 release of the SDK has been designed from the ground up to provide unparalleled speed, efficiency and ease of integration for developers.

Asynchronous Methods and Events: Listeners

Most of the SDK methods used by developers are asynchronous and follow a “listener” pattern, where the caller supplies callback blocks that are called when the operation completes. In the case of a successful API call, the “Success” listener is called and delivered any requested objects. If errors occur, a different “Error” listener is called and delivered the specific error that occurred.

Additionally, some components of the SDK provide a more permanent “listener” interfaces that allow application developers to be notified of events that occur in the SDK outside of a direct API call from the developer. Devices added to or removed from the user’s account, changes to online / offline state of devices, or the refreshing of the users authorization to the Ayla Cloud Service are all examples of events that can delivered by SDK components to interested parties.

AylaSessionManager, AylaDeviceManager and *AylaDevice* all provide methods to allow classes to register as listeners to events generated by the SDK in response to changes to objects in the system.

Hassle-free device management

The Ayla Mobile SDK automatically keeps itself up-to-date without any interaction required from the application developer. As the mobile application will find itself in various types of network situations, the SDK will adapt its methods of keeping in sync with devices and the Ayla Cloud Service.

The SDK contains several mechanisms to keep the list of devices as well as the status of each device up to date:

Device List: Poll	Polls the Ayla Cloud Service for the list of devices registered to the account
Device status: LAN mode	Maintains a local network connection with the mobile device to receive datapoint events
Device status: Mobile DSS	Maintains a persistent connection with the AylaCloudService to receive notifications of changes to devices and their properties
Device status: Poll	If LAN mode and Mobile DSS are not available, devices will poll the service as a last resort

The list of devices currently is always obtained from the service via polling. The device list changes infrequently- only when devices are added or removed from the users account.

Devices will attempt to enter LAN mode at all times. If the LAN connection is established, this will be the source for updates for device properties.

If LAN mode could not be established, such as when the user is away from their home WiFi network, Mobile DSS is used as the source of updates for device properties.

If Mobile DSS is not reachable or disabled, devices will poll the service to detect updates.

Application developers require no additional work, as the SDK will always use the best method available to keep devices up to date.

API Documentation

Both the Android and iOS versions of the SDK provide detailed inline documentation within the source code. Documentation sets may be built from the sources for the appropriate platform: AppleDoc documentation may be built using the [TBD-IOS] target in Xcode; JavaDoc documentation may be built using the gradle generateDebugJavadoc command on the command line.

SDK Components

The SDK consists of several components that work together to provide developers easy access to user account information, registered devices, and device statuses. The SDK also provides several methods for keeping device statuses up to date and automatically uses the best method based on network conditions and device reachability.

The main components of the Ayla SDK developers will interact with are:

- *AylaNetworks* – SDK configuration and initialization
- *AylaLoginManager* – Used to sign in, sign up or authenticate users and provide methods to reset passwords, etc.
- *AylaSessionManager* – Created after sign-in, keeps session up to date by refreshing authorization. This represents a user login session.
- *AylaDeviceManager* – Created by the *AylaSessionManager* after sign-in, maintains the list of devices and ensures they are up-to-date
- *AylaSetup* –Object that is used to assist connecting to and configuring devices that are not yet on the local WiFi network
- *AylaRegistration* – Object used to assist in registering devices (that are already connected to a network) with the user's account

AylaNetworks

The *AylaNetworks* class is used to configure the SDK with your application-specific information. The *AylaNetworks* class must be initialized before any other SDK operations are performed. The initialization of the SDK requires the developer to provide the SDK with an *AylaSystemSettings* object filled out with information pertinent to the application under development.

The *AylaSystemSettings* object requires the following fields to be filled out:

appId	Application ID for the mobile app, supplied by Ayla Networks
appSecret	Application secret for the mobile app, supplied by Ayla Networks
serviceType	Service the application should connect to. May be one of: Dynamic, Field, Development, Staging or Demo
serviceLocation	Service location the application should connect to. May be one of: USA, China, Europe
deviceDetailProvide r	Interface that must be implemented by the application developer that provides information to the SDK about devices found on the account

Most of the system settings are related to associating the application with the application developer's account with Ayla Networks and the specific services (location and type) that the mobile app wishes to connect to.

The *DeviceDetailProvider* is a bit different, as it requires the developer to implement an interface that will be called by the Mobile SDK. Currently this interface contains a single method, *getManagedPropertyNames*. The method is given a device and is expected to return an array of strings for the names of properties the SDK should manage for the provided device.

Managed properties are discussed in more detail in the *AylaDevice* section of this document.

Once an *AylaSystemSettings* object has been created by the application developer, it can be passed to *AylaNetworks* initialize method to initialize the SDK.

AylaLoginManager

The *AylaLoginManager* is one of the few SDK components that is available before a user has signed in to the service. It contains methods to sign-in the user, sign up a new user, have the service re-send a confirmation email to the user, or have the service send an email to the user with a link to reset their password.

Signing In

The *AylaLoginManager* contains a single method to sign in the user, signIn. This method takes in an application-defined name for the session and an *AylaAuthProvider* object that performs the actual sign-in operation and provides the authorization to the SDK. The SDK comes with three pre-defined *AylaAuthProvider* objects that can be used for the most common methods of signing in:

- *UsernameAuthProvider*, which is used to sign in using a username / password
- *AylaOAuthProvider*, which is used to sign in using Google or Facebook OAuth
- *CachedAuthProvider*, which is used to sign in by refreshing an existing authorization from a previous sign-in operation, and can also store / retrieve credentials in the mobile applications settings

Additional *AylaAuthProvider* objects may be created to sign in to the service via custom means.

If the sign-in operation is successful, the caller is returned an *AylaAuthorization* object which may be cached and used for subsequent sign-in operations. An *AylaSessionManager* is also created, which may be accessed by the name provided to the *signIn* API call via a call to *AylaNetworks getSessionManager* method.

AylaSessionManager

Once sign-in has completed, an *AylaSessionManager* is created by the SDK to keep the session up to date. The session manager will refresh the authorization before it expires and provides the authorization received from the sign in operation to subsequent requests to the Ayla Cloud Service.

The AylaSessionManager also provides the *AylaDeviceManager* object, which is responsible for maintaining the list of registered devices.

Applications may register with *AylaSessionManager* via the addListener method to receive notifications of events related to the session, such as the session being closed due to user sign-out or an authorization issue, or the authorization being refreshed. Applications that sign in using cached authorizations should be sure to update cached authorizations when the *AylaSessionManager* refreshes them

AylaDeviceManager

The *AylaDeviceManager* is created by *AylaSessionManager* once the user has been authenticated. The *AylaDeviceManager* is responsible for maintaining a list of the devices registered to the account as well as keeping those devices up to date.

When the user is first authenticated and a session created, the session's *AylaDeviceManager* goes through a series of steps to initialize. Applications may register with *AylaDeviceManager* via the *addListener* interface to be notified of these state changes as well as a number of other device-related events.

Once the *AylaDeviceManager* has completed initialization, listeners are notified via a *deviceManagerInitComplete* call. If any errors were encountered during initialization, they will be provided to the listener via this interface as well.

After initialization, application developers may get the list of devices from *AylaDeviceManager* via *getDevices.*

AylaDeviceManager polls the service at regular intervals for the list of devices, and will notify any listeners if the list changes. The poll interval defaults to 15 seconds, and can be modified by calling *setPollInterval*.

In most applications, the only interaction application developers will need to have with *AylaDeviceManager* is registering as a listener and updating any user interface elements when notified that the device list has changed.

AylaDevice

AylaDevice represents a physical device registered to the users account. *AylaDevices* are obtained via the *AylaDeviceManager* once it has completed its initialization, and contain information about the current state of the device as well as methods to manipulate the current state of the device.

AylaDevice objects also support registration of listeners to be notified of changes to the device. Mobile applications may register with each *AylaDevice* to be notified of changes to the device’s properties, state, or any other changes to the device. These notifications may be used to provide changes to the user interface or display messages to the end user regarding the state of the device.

AylaDevice objects are kept up to date by the SDK. No additional effort is required. Several methods are used by the SDK to ensure that the objects accurately reflect the state of the device:

LAN mode	When the mobile app and device are on the same local network, the SDK will connect directly to the device. Changes to the device are sent to the SDK via local network calls, and datapoints can be created on the device via a direct connection. This is the fastest method for keeping devices updated, though it is only possible while the user and device are on the same network.
Mobile DSS	If LAN mode is not possible, Mobile DSS is used to keep devices up to date. Mobile DSS is a websocket-based service that will send events to the SDK when devices change. Mobile DSS is available when the mobile device is connected to the Internet and can reach the Ayla DSS service.
Polling	Finally, if LAN mode and Mobile DSS are not available, the SDK will poll the Ayla Cloud Service for changes to device state.

Regardless of the method used to update devices within the SDK, users of the SDK will be notified via the *AylaDevice* listener interface whenever a device changes.

Application developers will frequently use *AylaDevice* objects both for monitoring the state of registered devices as well as controlling those devices. Developers should register listeners for each AylaDevice that needs to be monitored.

Custom Device Classes

As of the 5.5 release, developers may specify their own AylaDevice subclasses to be managed by the Ayla Device Manager. To specify a class to be instantiated for a particular device, developers may register a DeviceClassPlugin object with the Ayla SDK via a call to *AylaSDK.sharedInstance().installPlugin*, passing in an object that implements the DeviceClassPlugin interface.

The DeviceClassPlugin interface contains a single method:

```
Class<? extends AylaDevice> getDeviceClass(String model, String
oemModel, String uniqueId);
```

Whenever an AylaDevice object is received from the cloud service, the Device Manager will call this method passing in the model, oemModel and unique identifier (if present) of the device. The method should return a Class object indicating the class that should be created for the device object. This class will be instantiated and the resulting object managed by AylaDeviceManager.

AylaLocalDevice

As of the 5.5 release of the Ayla Mobile SDK, devices that do not on their own have Internet connectivity may be brought into the Ayla network. Local devices rely on the mobile device to communicate with the Ayla cloud on their behalf.

The Ayla Mobile SDK provides support for local devices at a high-level, as well as additional support for Bluetooth LE devices via the AylaBLEDevice class, a subclass of AylaLocalDevice.

Local devices can be controlled from the mobile device when a direct connection exists between them. The Ayla SDK takes care of translating calls to read and write Ayla Property objects into calls into the Ayla Local Device to read or write values to or from the local device, allowing seamless interoperation with other Ayla devices.

See your Ayla representative for more information about Ayla Local Devices and how to get your own BLE or other device on to the Ayla network.

AylaProperty / AylaDatapoint

AylaDevice objects contain a set of *AylaProperty* objects. Each *AylaProperty* represents a state of the device, such as whether a switch is on or off, or what the current temperature is on a thermostat. An *AylaProperty* also may represent a control interface, such as setting the desired temperature of a thermostat, or turning a wall switch on or off.

An *AylaProperty* may have its value changed by creating an *AylaDatapoint* for that property. The "value" of an *AylaProperty* is really just a representation of the latest *AylaDatapoint* that was created for the property.

This allows a history of values to be stored for the property. Sets of datapoints may be retrieved via calls to *fetchDatapoints*. Results may be limited by a maximum count returned as well as a range of dates the datapoints were created.

The Ayla Mobile SDK uses the aforementioned methods to ensure that the *AylaProperty* objects always reflect the latest datapoint known to the service. Listeners to the owning *AylaDevice* are notified whenever the latest datapoint for a property changes.

Datapoints may be created for the property via a call to *createDatapoint*.

Applications that need to create datapoints in batches may use the *createDatapointBatch* method on *AylaDeviceManager*. Batch requests may include datapoints for multiple properties or devices.

Managed Properties

Devices may have many properties defined to record various pieces of information specific to the product. Some of these properties do not change in value, while others may not be useful to the mobile application.

The SDK is responsible for keeping property values (the value of the latest datapoint for a given property) up-to-date using several different methods. In order to reduce the number of properties the SDK is required to manage, the application developer may provide list of "managed" properties that are considered important to the application.

When devices need to update their properties, they will first check with the DeviceDetailProvider supplied to the SDK via the *AylaSystemSettings* object during initialization. If the *DeviceDetailProvider* returns an array of property names for a given device, only those properties will be fully managed by the SDK and kept up-to-date.

Other properties may be fetched from the service via calls to *fetchProperties* passing in specific names or null (which indicates all properties should be fetched), but may not be automatically kept up-to-date by the SDK.

AylaPropertyTrigger / AylaPropertyTriggerApp

An *AylaPropertyTrigger* may be attached to an *AylaProperty* to provide events when datapoints with certain conditions are created on the property. For example, a trigger might be created that fires whenever the temperature drops below a certain value, or one that fires whenever a door is opened.

An *AylaPropertyTrigger* defines the conditions that must be met in order for the trigger to "fire". For example, a trigger might be defined to fire when a datapoint is created for the property with a value that is less than 62 (degrees on a thermostat, for example). Another trigger might be defined to fire when a datapoint is equal to 1 (on the "door_opened" property, for example).

What actually happens when the conditions for a trigger are met is the responsibility of the *AylaPropertyTriggerApps,* which are attached to the trigger. *AylaPropertyTriggerApps* can be defined to send users an SMS message, email, or push notification.

Multiple *AylaPropertyTriggers* may be attached to an *AylaProperty*.

Multiple *AylaPropertyTriggerApps* may be attached to an *AylaPropertyTrigger.* All apps will be executed when the trigger's conditions have been met.

An *AylaPropertyTrigger* is created via a call to the *AylaProperty's* method *createTrigger*.
An *AylaPropertyTriggerApp* is created via a call to the *AylaPropertyTrigger's* method *createApp*.

AylaSchedule / AylaScheduleActions

AylaDevices support schedules, which may be used to trigger various actions. *AylaSchedules* may be set up to fire at certain times each day, on specific days of the week, specific dates, etc. and update properties via its *AylaScheduleActions*, described below.

AylaScheduleActions are attached to *AylaSchedule* objects and describe what changes should happen when the schedule fires. *AylaScheduleActions* specify the update that should occur to the *AylaProperty* when the schedule fires, as well as information about whether the action should take place at the start or end of the schedule.

AylaSetup

New devices need some help getting connected to the local network. The *AylaSetup* class provides methods to connect -to Wi-Fi devices that provide an Access Point and help configure them to join a local Wi-Fi network.

The *AylaSetup* class may be used without a running session so that devices may be configured without being associated with an account (registered).

On the Android platform, *AylaSetup* may be used to scan for known access points using a developer-provided filter, connect to the device's access point, have the device scan for available Wi-Fi access points to join, obtain information about the device and set up a secure session to pass the user's Wi-Fi credentials to the device so that it may join the network.

Due to restrictions on the iOS platform, *AylaSetup* cannot scan for device Wi-Fi access points or connect to these access points without user intervention. However, *AylaSetup* can still be used to connect to the device, have the device scan for available access points, and establish a secure session for transmitting the network credentials to the device.

On both platforms, *AylaSetup* can be used to verify the Wi-Fi status of the device after credentials have been passed to ensure that it has joined the network. *AylaSetup* also may be used to ensure that the device was successfully able to connect to the Ayla Cloud Service. The following steps are required to set a new device up on the user's Wi-Fi network:

Android	iOS
Scan for Wi-Fi access points that match a specified string pattern (find Wi-Fi devices nearby)	Instruct user to find the Wi-Fi access point in iOS Wi-Fi settings
Connect to the device's access point	Instruct user to connect to the device's access point
Set up a secure connection with the device	
Send a command to the device to scan for access points	
Display the discovered access points to the user to choose which one to connect the device to	
Prompt the user for the Wi-Fi password, if the network requires one	
Submit the SSID and password to the device over the secure session so that it can join the user's Wi-Fi network	
Poll the device for its Wi-Fi status to recognize when it has connected or if it has encountered an error	
Re-connect to the original Wi-Fi network	Instruct the user to re-connect to the original Wi-Fi network
Poll the Ayla Cloud Service to ensure the device was able to connect to the service	

AylaRegistration

Once a device has been configured to join the network, it needs to be associated with the user's account. This process is called "registration".

Devices may support one of several different methods of registration:

Same LAN	Device must be on the same network as the mobile device
Button-push	Requires the user to press a button on the device
Display	Device has a display with a code that needs to be provided by the user
DSN	Device Serial Number is used to register
AP-mode	Device provides an access point to share a token
Node	Device is a node of a gateway, gateway is required to register

Each registration type has different criteria that must be met in order to register. The *AylaRegistration* class provides methods to work with each of these registration types.

For registration flows that require a registration candidate (Same LAN, Button-Push) *AylaRegistration* provides a method to fetch the candidate from the Cloud Service.

Other registration methods do not require a candidate to be fetched from the cloud.

Once a candidate is fetched, if required, or the user has provided other registration information to the mobile application, the application may call the *registerDevice* method on *AylaRegistration* to register the device with the user's account.

Getting Started

This section outlines the steps needed to get an Ayla-connected mobile app up and running quickly using the Ayla SDK. The basic steps to begin using the Ayla SDK include:

- Create an instance of *AylaSystemSettings* configured for your application
- Create a *DeviceDetailProvider* to provide the SDK information about the devices your app will manage
- Call *AylaNetworks* initialize method with the application's *AylaSystemSettings*
- Sign in to the Ayla Cloud Service using *AylaLoginManager*
- Register to listen for events from the *AylaSessionManager* for the login session
- Register to listen for events from the *AylaDeviceManager*
- Register to listen for events from each *AylaDevice*

Each of these steps is outlined in more detail below:

Create AylaSystemSettings

The *AylaSystemSettings* class is used to initialize the SDK. The class contains several members that need to be provided by the developer before being passed to *AylaNetworks initialize* method.

AylaSystemSettings contains the following members:

`deviceDetailProvider`	User-provided class with methods to provide information about devices known to the mobile application
`appId`	Ayla-provided string that identifies the app
`appSecret`	Ayla-provided string that identifies the app
`serviceLocation`	Location of the Ayla service the app should connect to: USA, China, or Europe
`serviceType`	Type of service the application should connect to: Development, Field, Staging, Demo or Dynamic

DeviceDetailProvider

The *DeviceDetailProvider* is an interface that allows the application to provide information about specific devices to the SDK. Currently there is a single method of this interface, *getManagedPropertyNames*.

This method takes in an *AylaDevice* object and is expected to return an array of strings of the names of properties the SDK should manage.

Devices may have many properties, though often only a few of them are of concern to the application developer. By providing an implementation to this interface, the SDK can be made aware of which properties of each device should be managed by the SDK.

If all properties of a device are meant to be managed, they should either all be included in the array returned from this method (preferred), or the method should return null for those devices.

If this method returns null for a particular device, all properties will be fetched from the service and managed.

A wall switch, for example, might have a single property that is used to turn on or off the switch and provides the current state for the switch. A thermostat, on the other hand, might have a property for the minimum temperature, another for maximum temperature, and a third for the current temperature. The *DeviceDetailProvider* implementation for this app might look something like this:

```
String[] getManagedPropertyNames(AylaDevice device) {
    if ( device.getOemModel().equals("My_Switch") {
        return { "OnOff_Property" };
    } else if ( device.getOemModel().equals("My_Thermostat") {
        return {"Max_Temp", "Min_Temp", "Current_Temp"};
    }
}
```

The SDK will call getManagedPropertyNames with each device it is managing.

Signing In

Once the SDK has been initialized, the next step is to sign in the user. There are several methods available to sign in, each of which has an *AylaAuthProvider* class used to perform the sign-in operation and provide the authorization to the SDK.

The most common method of signing in is via a username and password. The *UsernameAuthProvider* class provides the means to sign in using a username and password. This class is initialized with the user's username and password.

Additionally, the *AylaOAuthProvider* class may be used to sign in with Google or Facebook using OAuth authentication. This class is initialized with the type of account to authorize with (currently Google and Facebook are supported) as well as a web view to display the appropriate interface to the user for the selected authentication type.

Finally, once the user has signed in, the obtained credentials may be saved and used to sign the same user in again. The *CachedAuthProvider* class provides a means of storing an *AylaAuthorization* object (the result of a successful sign-in operation, regardless of the method of authentication used to sign in) as well as signing in using a previously-saved authorization.

When the appropriate *AylaAuthProvider* has been initialized, it may be passed to *AylaLoginManager's* signIn method. If the sign-in process is successful, the caller will be called back with the *AylaAuthorization* object that can be saved using the *CachedAuthProvider's cacheAuthorization* method, and later retrieved via the *getCachedProvider* method.

Listening for Session changes

After successfully signing in, the *AylaSessionManager* will keep the authorization up to date by periodically refreshing it with the Ayla service. If an error occurs when refreshing the authorization, the session will be closed.

If the application is caching the authorization to be used to sign in later, it needs to update the cached authorization whenever the *AylaSessionManager* updates it.

AylaSessionManager provides a listener interface that is called whenever the session is closed or the authorization object has changed. Developers may register with *AylaSessionManager* via its *addListener* interface to be notified of these events.

Working with Devices

Once the user has signed in, the *AylaSessionManager* creates an *AylaDeviceManager*, which is responsible for maintaining the list of devices registered to the user.

The *AylaDeviceManager* passes through several states during initialization:

`Uninitialized`	The initial state of AylaDeviceManager

`FetchingDeviceList`	Currently fetching the list of devices from the service
`FetchingDeviceProperties`	Currently fetching property details for each device registered to the account
`FetchingLanConfig`	Fetching LAN configuration information for LAN mode, required for secure communications with the devices
`Ready`	The normal state of AylaDeviceManager after it has completed initialization.
`Error`	An unrecoverable error has occurred
`Paused`	Set when the application enters the background, will pass through most states when resumed

After signing in, *AylaDeviceManager* will automatically pass through these states and eventually reach either the Ready or Error states. Application developers should register a listener interface with *AylaDeviceManager* to be notified of the state changes.

AylaDeviceManager supports a listener interface to notify listeners of changes to its state. Objects implementing the *DeviceManagerListener* interface may register with *AylaDeviceManager* to receive these notifications.

For convenience, when the *AylaDeviceManager* has reached the Ready state, listeners will be notified via a call to *deviceManagerInitComplete*. If an error occurred, listeners will be notified via a call to *deviceManagerInitFailure* with the error that occurred and the state the *AylaDeviceManager* was in when the error occurred.

Listeners are also called whenever the list of devices has changed, such as when a device is added or removed from the users account, or a device has been shared with this account from another account. The *deviceListChanged* method is called in these situations.

Once the *AylaDeviceManager* has entered the Ready state, the list of devices may be obtained via a call to *getDevices*.

DeviceListeners

AylaDevice objects keep themselves up-to-date. When a device detects it has changed in any way, it calls its *DeviceListeners* to notify them about the change. Classes implementing the *DeviceListener* interface may register with *AylaDevice* objects for these notifications via a call to *addListener*.

Device listeners are notified of the following events:

deviceChanged	The device's properties or attributes changed
deviceError	The device encountered an error while trying to update itself
deviceLanStateChanged	The device has entered LAN mode or exited LAN mode

Applications should register with each device they are interested in receiving updates from.

Properties

Properties represent characteristics of devices that can change. Things like on / off state, current temperature, "motion detected", etc. are all examples of properties.

The Ayla Mobile SDK manages these properties, providing interfaces to update them as well as listener interfaces to know when they have been updated from outside the mobile application.

The set of properties for a device can be obtained via a call to *getProperties*. These properties may be used to control the device, such as turning a light on or setting a fan speed to 30%. Properties that are meant to control the device like this may be updated by creating an *AylaDatapoint* on the property using the *createDatapoint* method on *AylaProperty*.

When datapoints are created from outside the application, such as from the device itself, a schedule firing, or from another controlling application, listeners to the AylaDevice object that owns the property will be notified that it has changed via a call to *deviceChanged*.

Property Triggers

Properties can be configured to fire events when certain conditions are met. Like AylaSchedule, an *AylaPropertyTrigger* contains a set of "applications" called *AylaPropertyTriggerApps* that are "run" when the trigger's condition has been met.

AylaPropertyTriggers are configured to compare the value of the latest datapoint (also referred to as the property's current value) to a pre-defined value. Multiple triggers may be installed on a single property, each with its own set of applications that are run.

Currently *AylaPropertyTriggerApps* can be set to:

- Email a contact
- Send a contact an SMS message
- Send a contact a push notification (APNS, GCM, Baidu are supported)

These triggers allow applications to be notified when certain conditions have been met whether the mobile app is running or not.

Sharing Devices

AylaDevices may be shared with other users with separate accounts. *AylaSessionManager* provides methods to create, update and delete shares with other users.

To share a device with another user, an *AylaShare* object is created and initialized with the email address of the user to share with, the serial number (DSN) of the device to share, start / end dates when the share should be valid, and the role (read-only, read / write) the share should be granted with.

Once the *AylaShare* object is created, it can be passed to *AylaSessionManager* via the createShare API.

Devices that are shared with the current user account will appear in the list of devices from *AylaDeviceManager*, and can be treated like other devices in the system. These *AylaDevice* objects will be identifiable as "shared" devices by the presence of an *AylaGrant* object returned from *AylaDevice's getGrant*.

User-defined data: AylaDatum

For data that needs to be stored, but is not an *AylaProperty*, *AylaDatum* provides a way to store key / value pairs associated with the user account or specific devices.

AylaDatum objects may be created and saved via API calls to *createDatum*. Both *AylaSessionManager* and *AylaDevice* support this API. The appropriate method should be called depending on whether the information stored should be associated with the user account (*AylaSessionManager*), or a particular device registered to the user account (*AylaDevice*).

AylaDatum objects may be updated, deleted, or fetched as well. The contents of an AylaDatum object are application-specific and is not processed by the SDK or Ayla Cloud Service.

Contacts

The *AylaContact* class may be used to create an "address book" of contacts that are known to the Ayla Cloud Service. These contacts may be referred to when creating *AylaPropertyTriggerApps* or *AylaDeviceNotificationApps*.

AylaContacts may be fetched, created, updated or removed by calling the appropriate method on *AylaSessionManager*.

Connectivity

Upon initialization, *AylaNetworks* provides an *AylaConnectivity* object that may be used to receive notifications of network connectivity changes. Applications may register as listeners to the *AylaConnectivity* object via a call to *registerListener*. Listeners will be notified via their *onReceive* method whenever the network configuration on the mobile device changes.