APS Analytics for iOS

Pro or Enterprise Subscription Required
This AMPLIFY Appcelerator Services feature requires a Pro or Enterprise Subscription.

This page describes how to use the AMPLIFY Appcelerator Services Analytics for native iOS applications, built with Objective-C and the iOS APIs.

Not developing a native iOS application with Objective-C?

See the following topics to use the Appcelerator Analytics Service on other platforms:
For native Android applications built with Java, see APS Analytics for Android.
For Titanium Applications, see Appcelerator Analytics.

Introduction

Appcelerator Analytics collects real-time data about your application's usage, which can then be viewed in the **Analytics dashboard**. By default, the Analytics dashboard provides information about app installs, the number of sessions, and average app session length (organized by app name, platform, and geography). Your app can also utilize custom analytic events and event funnels.

This document provides an overview of the features provided by Analytics and how to use them using the Titanium SDK. **If you are developing an iOS application with Objective-C or Swift or an Android application with Java**, see APS Analytics for iOS or APS Analytics for Android for details on using Analytics.

For platform-specific details about how analytics captured, see Analytics Architecture.

For information about viewing analytics data, see Managing Client Applications.

Terminology

Analytics refers to data about how your application has been used, as well as information about how users interact with your application. Analytics data is transmitted in the form of *events*.

Events are operational milestones in the application. Some events are generated automatically, such as those that mark an installation, or the beginning and end of a session. Others may be **custom events**, which have a meaning specific to an application, such as tapping a specific button or opening a certain window.

A feature event represents an action a user could take in an application, such as 'liking an item' or launching a video'. Applications use the Titanium or APSAnalytics API to create custom events.

Event funnels let you define custom, ordered event sequences that let you track a specific user process, such as finding a product and making a purchase.

The **Analytics dashboard** organizes, analyzes, and presents analytics data captured for your applications. You also use the Analytics dashboard to create and view event funnels.

Getting Started

To integrate the Performance service with a new or existing iOS application:

1. Go to the Dashboard and create a new native iOS application.
2. Download the Services SDK and get your Analytics application key.
3. Unpack the `appcelerator-sdk-ios-<VERSION>.zip` file.
4. Drag the `Appcelerator.framework` folder into your Xcode project's root folder if you are using Xcode 6 and above, or the `Frameworks` folder if you are using Xcode 5 or below.
5. Select **Copy items into destination…** and click **Finish**.

6. Select your project in the Project Navigator and click the **Build Phases** tab.
7. Expand the **Link Binary With Libraries** section and add the `libsqlite3.dylib` and `libz.dylib` frameworks.
8. Click the **Build Settings** tab, then click the **All** button in the top-left corner of the tab.
9. Expand the **Linking** section and add `-ObjC` to Other Linker Flags.
10. In your application delegate implementation file, import `Appcelerator/Appcelerator.h`.

AppDelegate.m

```
#import <Appcelerator/Appcelerator.h>
```

11. In the application delegate's `application:didFinishLaunchingWithOptions` method, enable the service by calling the APSServiceManager's `enableWithAppKey:` method.

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    [[APSServiceManager sharedInstance] enableWithAppKey:@"APS_APP_KEY"];
    return YES;
}
```

To get your APS App key:

a. Go to the Dashboard.
b. Select your application from the **Apps** drop-down menu.
c. Click the **Overview** tab.
d. Click the **Services** button.
e. Click **Show Key** under the Analytics and Cloud section.

The iOS application can now send user session events and make additional method calls using the `APSAnalytics` class.

Advanced Initialization Options

Session Timeout

By default, after the application has been backgrounded for 30 seconds, the Analytics service ends the current user session and starts a new one when the application enters the foreground again. To adjust the timeout, use the `sessionTimeout` property.

```
// Sets the timeout to 15 seconds instead of 30 seconds.
[APSAnalytics sharedInstance].sessionTimeout = 15;
```

Creating Custom Events

You use the APSAnalytics API to log and report custom events. Feature Events are for capturing a user action, such as selecting a specific menu option or launching a video.

Currently, the optional `data` parameter of the Titanium.Analytics methods, which are used for logging a dictionary object, cannot be accessed through the Dashboard.

Feature Events

You use the `sendAppFeatureEvent:payload` method to generate a feature event that captures a specific application or user activity. A feature event should represent an action, such as launching a video, or 'new item', 'launch video', and so forth. The name you assign to a feature event should incorporate the application state into the event name, rather than long descriptive names. The following naming convention is suggested, where *group.event* refers to the parent event:

`group.event.sub-event`

Feature event names should be as generic as possible. For instance, if you want to track when users select a certain menu option, use a name like **"user.menu.selection",** not **"joeuser.menu.selection"**. The first option is better because it groups all the same types of an event into a single metric that's easy to view on Dashboard. The person analyzing the data only has to look at a single number to get an overview of how many users have selected that menu option. The second might be fine for very small user bases, but if you have more than 100 users, it means that the person analyzing the data would have to look through 100 different event names to be able to generate any useful data.

For example, to track a user's menu selection, you might use the following code, where the ten-digit number uniquely identifies the selection in your code:

Good Practice: Track the State with the Naming Syntax

```
[[APSAnalytics sharedInstance] sendAppFeatureEvent:@"select.item.12345678910"
payload:nil];
```

You should avoid using long, descriptive event names, as shown below:

Bad Practice: Avoid Long Descriptions

```
 [[APSAnalytics sharedInstance] sendAppFeatureEvent:@"Select Item THIS IS THE
DESCRIPTION OF THE EVENT -12345678910" payload:nil];
```

Geo Events

Use the `sendAppGeoEvent` to send real-time geographic data to the Analytics service. Pass the method a CLLocation object containing the location data. For example, you can use the `locationManager: didUpdateLocations` delegate method to send geo-events when the device receives new location data.

GeoController.m

```
// Delegate method from the CLLocationManagerDelegate protocol
- (void)locationManager:(CLLocationManager *)manager
     didUpdateLocations:(NSArray *)locations
{
    CLLocation *newLocation = [locations lastObject];
    [[APSAnalytics sharedInstance] sendAppGeoEvent:newLocation];
    NSLog(@"%f, %f", newLocation.coordinate.latitude,
newLocation.coordinate.longitude);
}

@end
```