



Emitter & Push SDK Integration Guide iOS

Document Revision October 15, 2013

Table of Contents

INTRODUCTION	3
SUPPORTED PLATFORMS	3
OBJECTIVE OF THIS DOCUMENT	3
GETTING STARTED	4
SETTING YOUR APP AS IN PRODUCTION OR IN DEVELOPMENT	5
IOS DEVELOPER PROGRAM ENROLLMENT	6
CONFIGURING YOUR APP FOR APPLE PUSH NOTIFICATIONS	7
APPLE DEVELOPER MEMBERS CENTER	7
REGISTERED APP ID	8
CREATING AN SSL CERTIFICATE	10
EXPORTING THE SSL CERTIFICATE	11
SETTING UP THE CERTIFICATE ON MOBILE CONNECT	12
INTEGRATING EMITTER & PUSH SDK INTO YOUR APP	14
DOWNLOADING AND INTEGRATING THE EMITTER & PUSH SDK LIBRARY	14
REQUIRED LIBRARIES	14
HOW TO ADD THE EMITTER & PUSH SDK FILES TO YOUR PROJECT	14
LINK AGAINST THE STATIC LIBRARY	15
SETTING UP YOUR APP DELEGATE	15
<i>Import the required header files</i>	15
<i>Initialize Emitter & Push SDK Instance</i>	16
<i>Register for Remote Notifications</i>	16
TEST YOUR PUSH INTERGRATION	17
EMITTER & PUSH PARAMETERS	18
PARAMETERS	18
LOCATION SERVICE	18
EMITTER METHODS AND PARAMETERS	19
EMITTER CONFIGURATION PARAMETERS	19
EMITTER EVENTS	21

Introduction

This document describes how to integrate the RadiumOne Emitter & Push SDK for iOS into your application.

Supported Platforms

All mobile and tablet devices running iOS 5.0 with Xcode 4.5 and above.

Objective of this document

This document describes how to:

- Create application and push certificate (you can skip it if you can not use push)
- Add the Emitter & Push SDK files to your project
- Integrate Your Code and set up App Delegate

Getting Started

Before you start, you should have the following files:

Lib	Contains the library and headers of SDK for iOS. Library support multiple architectures: <ul style="list-style-type: none">• arm7• arm7s• arm64• i386• x86_64
DemoApplication	Fully functional iOS application that integrates all features of the Emitter & Push SDK.
Doc	SDK integration guide to help new developers integrate the RadiumOne SDK for iOS into their applications. This is the document that you are now reading.

Setting your App as in Production or in Development

When creating or editing an app on Mobile Connect you can set the status of the app to either “Production” status or “Development” status. “Production” status for an app is considered to be a live app that is in the hands of real users and will have notifications running on live servers. A “Development” status of an app is one that you are still performing testing on and will not be viewed by any of your real life audience because it will stay on test servers.

It is important to know this difference because Apple will treat these two servers separately. Also device tokens for development will not work on production and vice versa. We recommend a development app version and production app version for your app on Mobile Connect to keep Push SSL certificates for each separate. You can also continue testing and experimenting on one app without worrying about it affecting your live app audience in any way.

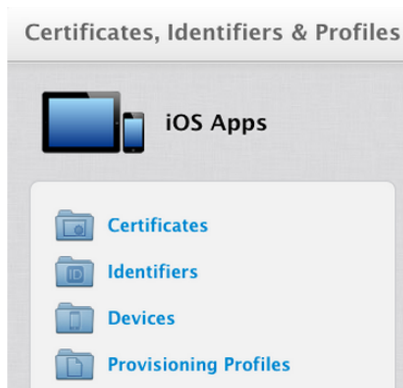
iOS Developer Program Enrollment

This integration assumes that you are enrolled in the iOS Developer Program. If you are not, please enroll [here](#). Being in the Apple Developer Program is a required component of having your iOS app communicate with the RadiumOne Mobile Connect service and is necessary for the next step of setting up your app with the Apple Push Notification Service. It is also essential that you have “Team Agent” role access in the iOS Developer Program to complete this process.

Configuring your App for Apple Push Notifications

Apple Developer Members Center

Make sure you are logged into the [Apple Developer Members Center](#). Once you are logged in you will locate your application in the *Identifiers* folder list.



Registered App ID

The screenshot shows the 'Register iOS App ID' form in the Apple Developer portal. The left sidebar contains a navigation menu with categories: Certificates, Identifiers, Devices, and Provisioning Profiles. Under 'Identifiers', 'App IDs' is selected. The main content area is titled 'Registering an App ID' and includes a description of App ID strings. It features three sections: 'App ID Description' with a 'Name' field; 'App ID Prefix' with a dropdown menu showing '6DH58Q45LS (Team ID)'; and 'App ID Suffix' with two radio button options: 'Explicit App ID' (selected) and 'Wildcard App ID'. The 'Explicit App ID' section has a 'Bundle ID' field with the example 'com.yourCompany.yourApp'. The 'App Services' section at the bottom allows enabling various services, with 'Push Notifications' checked. At the bottom of the form are 'Cancel' and 'Continue' buttons.

Registering an App ID

The App ID string contains two parts separated by a period (.)—an App ID Prefix that is defined as your Team ID by default and an App ID Suffix that is defined as a Bundle ID search string. Each part of an App ID has different and important uses for your app. [Learn More](#)

App ID Description

Name:

You cannot use special characters such as @, &, *, ', "

App ID Prefix

Value:

App ID Suffix

☒ **Explicit App ID**

If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.

To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.

Bundle ID:

We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (*).

☐ **Wildcard App ID**

This allows you to use a single App ID to match multiple apps. To create a wildcard App ID, enter an asterisk (*) as the last digit in the Bundle ID field.

Bundle ID:

Example: com.domainname.*

App Services

Select the services you would like to enable in your app. You can edit your choices after this App ID has been registered.

Enable Services:

- ☐ **Data Protection**
 - ☐ Complete Protection
 - ☐ Protected Unless Open
 - ☐ Protected Until First User Authentication
- ☒ **Game Center**
- ☐ **iCloud**
- ☒ **In-App Purchase**
- ☐ **Inter-App Audio**
- ☐ **Passbook**
- ☒ **Push Notifications**

If you have not registered an App ID yet it is important that you do so now. You will need to click the “+” symbol, fill out the form, and check the *Push Notifications* checkbox. Please keep in mind it is possible to edit these choices after the App ID is registered.

App Services


Select the services you would like to enable in your app. You can edit your choices after this App ID has been registered.

- Enable Services:
- ☐ Data Protection
 - ☐ Complete Protection
 - ☐ Protected Unless Open
 - ☐ Protected Until First User Authentication
 - ☒ Game Center
 - ☐ iCloud
 - ☒ In-App Purchase
 - ☐ Inter-App Audio
 - ☐ Passbook
 - ☒ Push Notifications

You can expand the application and when doing so you will see two settings for push notifications they will have either yellow or green status icons like here:

Your application

com.yourCompany.yourAppWithPush



Name: Your application

Prefix: 6DH58Q45LS

ID: com.yourCompany.yourAppWithPush

Application Services:

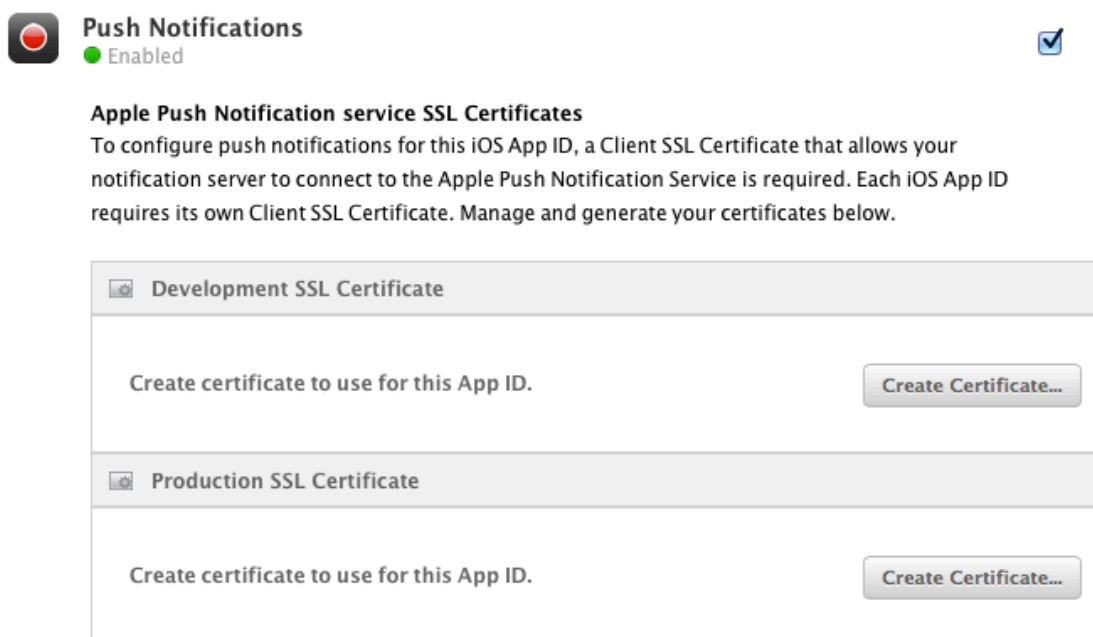
Service	Development	Distribution
Data Protection	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Game Center	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
iCloud	<input type="radio"/> Disabled	<input type="radio"/> Disabled
In-App Purchase	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
Inter-App Audio	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Passbook	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Push Notifications	<input checked="" type="radio"/> Configurable	<input checked="" type="radio"/> Configurable

Edit

You will need to click *Edit* to continue. If for some reason the edit button is not visible it is because you do not have “Team Agent” role access. This role is necessary for getting an SSL certificate.

Creating an SSL Certificate

To enable the Development or Production Push SSL Certificate please click *Edit* (It is important to note that each certificate is limited to a single app, identified by its bundle ID and limited to one of two environments either development or production. There is more info [here](#))



You will see a *Create Certificate* button, after clicking it you will see the “Add iOS Certificate Assistant”. Please follow the instructions presented in the assistant which includes launching the “Keychain Access” application, generating a “Certificate Signing Request (CSR)”, generating an SSL Certificate, etc.

If you follow the assistant correctly, after downloading and opening the SSL Certificate you should have it added under “My Certificates” or “Certificates” in your “Keychain Access” application. Also when you are returned to the Configure App ID page the certificate should be badged with a green circle and the label “Enabled”.

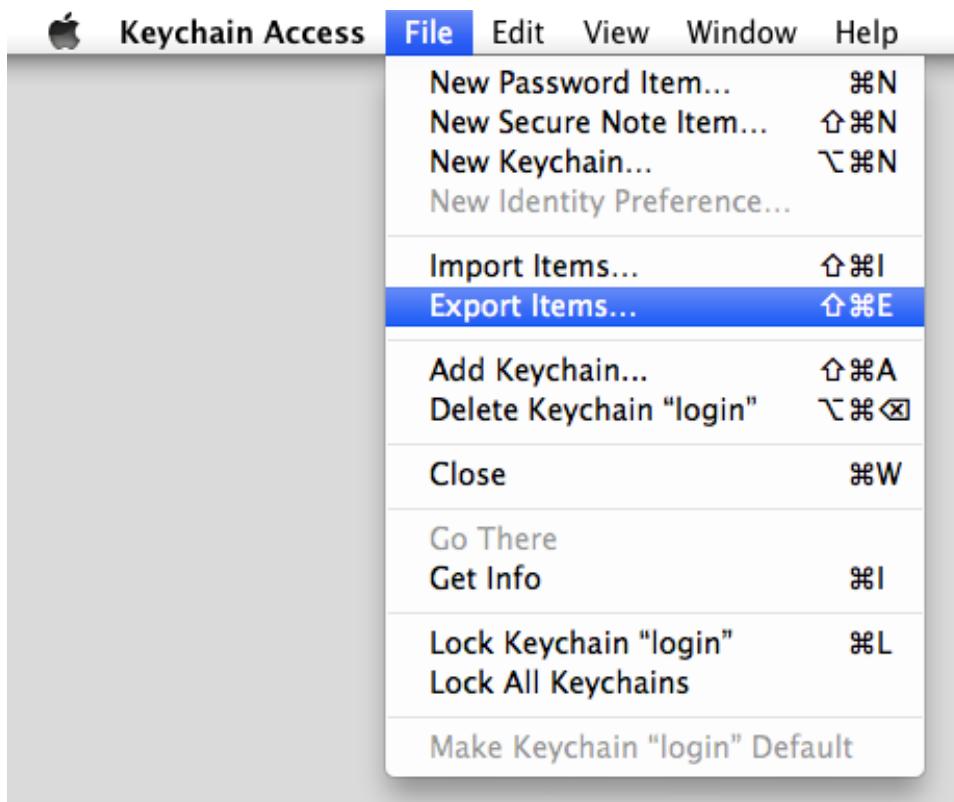
Download, Install and Backup

Download your certificate to your Mac, then double click the .cer file to install in Keychain Access. Make sure to save a backup copy of your private and public keys somewhere secure.



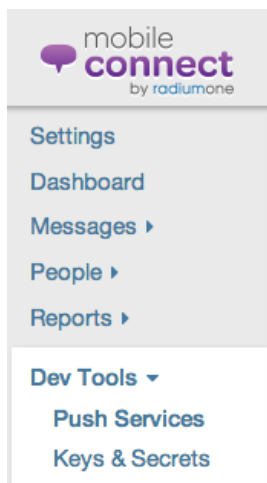
Exporting the SSL Certificate

If not already in the “Keychain Access” app that contains your certificate, please open it and select the certificate that you just added. Once you select the certificate go to File -> Export Items and export it as a Personal Information Exchange (.p12) file. Also when saving the file be sure to use the Personal Information Exchange (.p12) format.

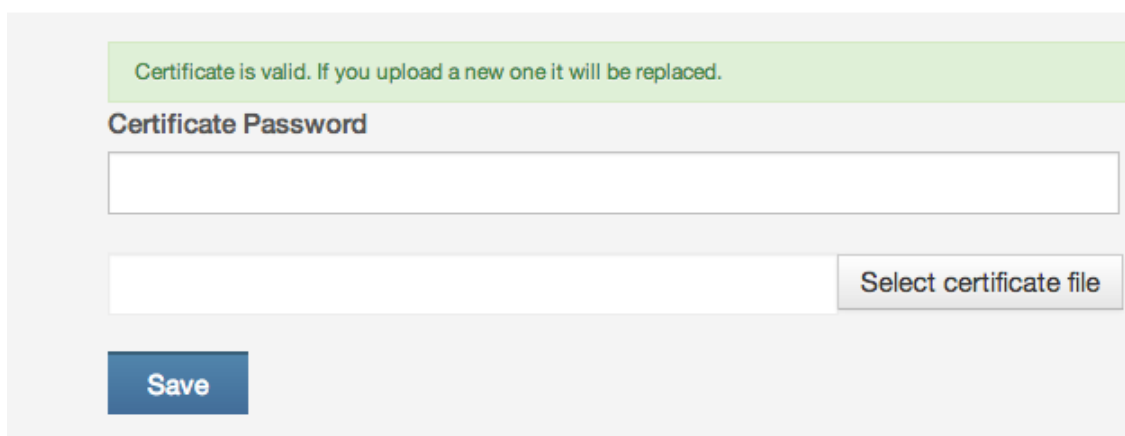


Setting up the certificate on Mobile Connect

Please make sure you are signed into your account on Mobile Connect and go to the application you want to upload this certificate to. Next, in the side menu go to Dev Tools -> Push Services -> Apple Push Notification Service.



Apple Push Notification Service

A screenshot of the Apple Push Notification Service configuration form. At the top, there is a green banner with the text 'Certificate is valid. If you upload a new one it will be replaced.' Below this is a section titled 'Certificate Password' with a text input field. Underneath the password field is another text input field for the certificate file. To the right of this field is a button labeled 'Select certificate file'. At the bottom left of the form is a blue button labeled 'Save'.

If your certificate has a password you can enter it here. Next, you can click *Select Certificate File* to choose the certificate file you exported in the “Exporting the SSL Certificate” step.

Once the file is uploaded, click *Save*.

If your file uploaded correctly you will get a green badge with a checkmark and text saying, “Certificate is valid. If you upload a new one it will be replaced.”

Now your app is setup with APNs (Apple Push Notification Service), next you will add the Mobile Connect Library to your iOS project

Integrating Emitter & Push SDK into your App

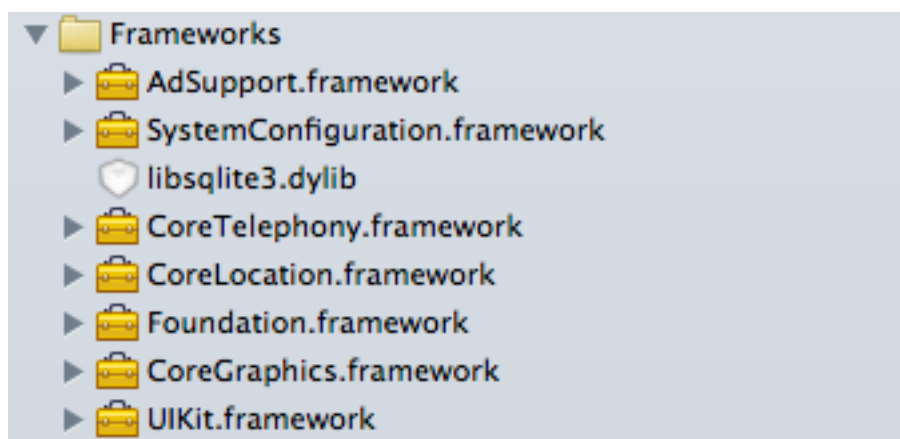
After completing the prior steps for connecting your app with APNs now you will have to setup your app to integrate with Mobile Connect. The following steps will setup this communication with Mobile Connect as well manage your app's connection with APNs so you can send notifications and receive back user reports.

Downloading and Integrating the Emitter & Push SDK Library

You can download and add the latest version of the Mobile Connect Library to the same directory as your project.

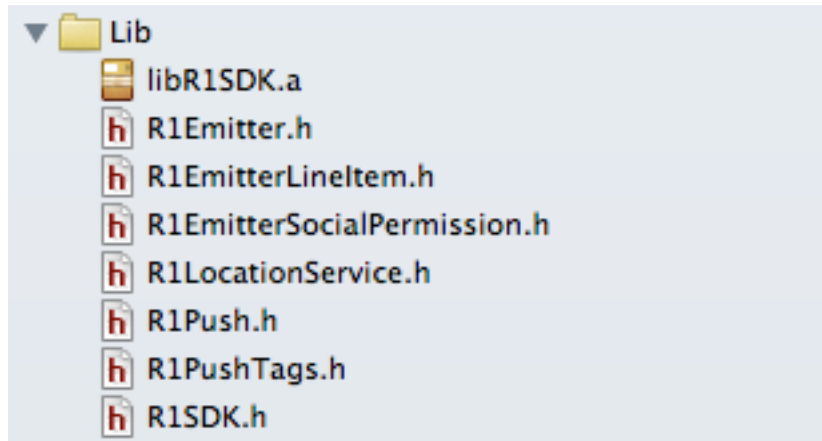
Required Libraries

To correct linking you must add few required frameworks:



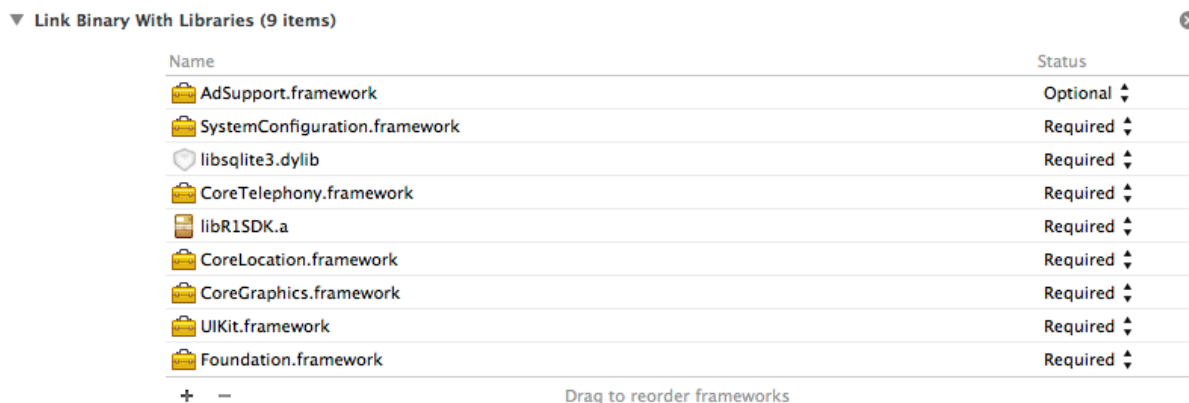
How to Add the Emitter & Push SDK files to Your Project

1. Open up your iOS project in xCode.
2. Select File -> Add Files to "[your project]"
3. Select Folder with SDK library and headers in dialog
4. When the dialog box appears, check the Copy Items into destination group's folder checkbox.



Link against the static library.

Check the libR1SDK.a file in the “Link Binary With Libraries” section in the Build Phases tab for your target. If it is absent add it.



Setting up your App Delegate

You will need to initialize Emitter & Push SDK in your App Delegate.

Import the required header files

At the top of your application delegate include any required headers:

- If you want use only emitter (without push)

```
#import "R1SDK.h"
```

- If you want use emitter and push

```
#import "R1SDK.h"
#import "R1Push.h"
```

Initialize Emitter & Push SDK Instance

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    R1SDK *sdk = [R1SDK sharedInstance];

    sdk.applicationId = @"[Your Application Id]";
    // Only if you want use push
    sdk.clientKey = @"[Your Client Key]";

    // Optional parameters for emitter & push
    sdk.applicationUserId = @"[Optional application user id]";

    [sdk start];

    // Your Code here

    // Only if you want use push
    [[R1Push sharedInstance] handleNotification:[launchOptions valueForKey:UIApplicationLaunchOptionsRemoteNotificationKey]
                                     applicationState:application.applicationState];

    // Only if you want use push
    [[R1Push sharedInstance] registerForRemoteNotificationTypes:(UIRemoteNotificationTypeBadge |
                                                                UIRemoteNotificationTypeSound |
                                                                UIRemoteNotificationTypeAlert)];

    return YES;
}
```

Register for Remote Notifications

```
- (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
    // Only if you want use push
    [[R1Push sharedInstance] registerDeviceToken:deviceToken];
}

- (void)application:(UIApplication *)application didFailToRegisterForRemoteNotificationsWithError:(NSError *) error
{
    // Only if you want use push
    [[R1Push sharedInstance] failToRegisterDeviceTokenWithError:error];
}

- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo
{
    // Only if you want use push
    [[R1Push sharedInstance] handleNotification:userInfo
                                     applicationState:application.applicationState];
}
```

Push is disabled by default. You can enable it in *application:didFinishLaunchingWithOptions:* method or later.

```
[[R1Push sharedInstance] setPushEnabled:YES];
```

If you enabled it in *application:didFinishLaunchingWithOptions:* method Push Notification UIAlertView will be showed at first application start.

Test your Push intergration

1. After you have this implemented, run and compile your code.
2. Add the application to your phone
3. When you first open the app on your phone it will ask you for permission to send notifications. If you are not prompted to receive notifications, you should revisit the prior steps.

To make sure your device token is registered:

Log into your account on *Mobile Connect* -> Go to your *App* and select it
-> Select the *People* section in the left-hand side menu -> Then select *Devices*

If your device token registered properly you should see it under Device ID

Emitter & Push Parameters

Parameters

applicationUserId

Optional current user identifier.

```
[R1SDK sharedInstance].applicationUserId = @"12345";
```

location

The current user location coordinates. Use it only if your application already uses location services.

```
[R1SDK sharedInstance].location = ...;
```

Location Service

If your application not used location information before this SDK installation, you can use `locationService` from this sdk:

```
[R1SDK sharedInstance].locationService.enabled = YES;
```

In this case location information will setted automatically to emitter and to push if it activated.

Location service doesn't get location always. When location received SDK turned off location in `CLLocationManager` and wait 60 minutes.

Emitter methods and parameters

Emitter configuration parameters

The following is the list of configuration parameters, most of which contain the values that are sent to the tracking server.

appName

Default : `CFBundleName` string from the application bundle

The application name associated with emitter. If you wish to override this property, you must do so before making any tracking calls.

```
[R1Emitter sharedInstance].appName = @"Custom application name";
```

appId

Default: nil

The application identifier associated with this emitter. If you wish to set this property, you must do so before making any tracking calls. **Note:** that this is not your app's bundle id, like e.g. com.example.appname.

```
[R1Emitter sharedInstance].appId = @"12345678";
```

appVersion

Default: `CFBundleShortVersionString` string from the application bundle.

The application version associated with this emitter. If you wish to override this property, you must do so before making any tracking calls.

```
[R1Emitter sharedInstance].appVersion = @"1.0.2";
```

appScreen

The current application screen set for this emitter.

```
[R1Emitter sharedInstance].appScreen = @"MainViewController";
```

sessionStart

If true, indicates the start of a new session. Note that when a emitter is first instantiated, this is initialized to true. To prevent this default behavior, set sessionTimeout to a negative value

By itself, setting this does not send any data. If this is true, when the next emitter call is made, a parameter will be added to the resulting emitter information indicating that it is the start of a session, and this flag will be cleared.

```
[R1Emitter sharedInstance].sessionStart = YES;  
// Your code here  
[R1Emitter sharedInstance].sessionStart = NO;
```

sessionTimeout

Default: 30 seconds

Indicates how long, in seconds, the application must transition to the inactive or background state for before the tracker will automatically indicate the start of a new session when the app becomes active again by setting sessionStart to true. For example, if this is set to 30 seconds, and the user receives a phone call that lasts for 45 seconds while using the app, upon returning to the app, the sessionStart parameter will be set to true. If the phone call instead lasted 10 seconds, sessionStart will not be modified.

```
[R1Emitter sharedInstance].sessionTimeout = 15;
```

Emitter Events

In context of A-SDK, events are the actions that define a conversion. Events that are common to most mobile applications are called Standard Events and A-SDK already implements most such events.

State Events

Some of the standard events are emitted when the state of the application is changed by the OS and, therefore, they do not require any additional code to be written in the app:

Launch	- emitted when the app starts
First Launch	- emitted when the app starts for the first time
Post Upgrade Launch	- emitted when the app starts for the first time after a version upgrade
Suspend	- emitted when the app is put into the background state
Resume	- emitted when the app returns from the background state

Interaction Events

Interaction Events are emitted due to some user action within the app. It is up to the app designer to decide which user actions need to be tracked and what information is passed to the tracking server. Therefore, the application code needs to include the emitter callbacks in order to emit these events.

Note: The last argument in all of the following emitter callbacks, *otherInfo*, is a dictionary of “key”, “value” pairs or nil

Action

A generic user action.

```
[[R1Emitter sharedInstance] emitAction:@"Button pressed"
                                label:@"About"
                                value:10
                                otherInfo:@{@"custom_key":"value"}];
```

Login

User login within the app

```
[[R1Emitter sharedInstance] emitLoginWithUserID:[R1Emitter sha1:@"usrId"]
    userName:@"userName"
    otherInfo:@{@"custom_key":"value"}];
```

Registration

User registration within the app

```
[[R1Emitter sharedInstance] emitRegistrationWithUserID:[R1Emitter sha1:@"userId"]
    userName:@"userName"
    email:@"user@email.com"
    streetAddress:@"streetAddress"
    phone:@"phone"
    city:@"city"
    state:@"state"
    zip:@"zip"
    otherInfo:@{@"custom_key":"value"}];
```

Facebook Connect

Connect to Facebook services

```
NSArray *permissions = @[[R1EmitterSocialPermission
socialPermissionWithName:@"photos" granted:YES]];

[[R1Emitter sharedInstance] emitFBConnectWithUserID:[R1Emitter sha1:@"12345"]
    userName:@"user_name"
    permissions:permissions
    otherInfo:@{@"custom_key":"value"}];
```

Twitter Connect

Connect to Twitter services

```
NSArray *permissions = @[[R1EmitterSocialPermission
socialPermissionWithName:@"photos" granted:YES]];

[[R1Emitter sharedInstance] emitTConnectWithUserID:[R1Emitter sha1:@"12345"]
    userName:@"user_name"
    permissions:permissions
    otherInfo:@{@"custom_key":"value"}];
```

Upgrade

Application version upgrade

```
[[R1Emitter sharedInstance] emitUpgradeWithOtherInfo:@{@"custom_key":"value"}];
```

Trial Upgrade

Application upgrade from trial to full version

```
[[R1Emitter sharedInstance]
emitTrialUpgradeWithOtherInfo:@{@"custom_key":"value"}];
```

Screen View

Information about the current screen

```
[[R1Emitter sharedInstance] emitScreenViewWithDocumentTitle:@"title"
    contentDescription:@"description"
    documentLocationUrl:@"http://www.example.com/path"
    documentHostName:@"example.com"
    documentPath:@"path"
    otherInfo:@{@"custom_key":"value"}];
```

Transaction

```
[[R1Emitter sharedInstance] emitTransactionWithID:@"AE3237DAA"  
    storeID:@"storeId"  
    storeName:@"store_name"  
    cartID:@"ABBCCD"  
    orderID:@"ABCDEF"  
    totalSale:3.2  
    currency:@"EUR"  
    shippingCosts:1.8  
    transactionTax:2.5  
    otherInfo:@{ "custom_key": "value"}];
```

Transaction Item

```
R1EmitterLineItem *lineItem = [R1EmitterLineItem itemWithID:@"product_id"  
    name:@"product_name"  
    quantity:5  
    unitOfMeasure:@"parts"  
    msrPrice:1.3  
    pricePaid:3.4  
    currency:@"USD"  
    itemCategory:@"items"];  
  
[[R1Emitter sharedInstance] emitTransactionItemWithTransactionID:@"transactionId"  
    lineItem:lineItem  
    otherInfo:@{ "custom_key": "value"}];
```

Create Cart

```
[[R1Emitter sharedInstance] emitCartCreateWithCartID:@"cartId"  
    otherInfo:@{ "custom_key": "value"}];
```

Delete Cart

```
[[R1Emitter sharedInstance] emitCartDeleteWithCartID:@"cartId"  
    otherInfo:@{ "custom_key": "value"}];
```


Add To Cart

```
R1EmitterLineItem *lineItem = [R1EmitterLineItem initWithID:@"product_id"
                                name:@"product_name"
                                quantity:5
                                unitOfMeasure:@"unit"
                                msrPrice:10
                                pricePaid:10
                                currency:@"USD"
                                itemCategory:@"category"];

[[R1Emitter sharedInstance] emitAddToCartWithCartID:@"cart_id"
                                lineItem:lineItem
                                otherInfo:@{@"custom_key":"value"}];
```

Delete From Cart

```
R1EmitterLineItem *lineItem = [R1EmitterLineItem initWithID:@"product_id"
                                name:@"product_name"
                                quantity:5
                                unitOfMeasure:@"unit"
                                msrPrice:10
                                pricePaid:10
                                currency:@"USD"
                                itemCategory:@"category"];

[[R1Emitter sharedInstance] emitDeleteFromCartWithCartID:@"cart_id"
                                lineItem:lineItem
                                otherInfo:@{@"custom_key":"value"}];
```

Custom Events

In addition to the Standard events listed above, A-SDK allows mobile apps designers to track events that are specific to their application.

There is no limit to what user actions could be tracked by the app.

Pressing the “like” button, changing the screen mode from portrait to

landscape, zooming in or out on an image – are all examples of the user actions that can cause events to be emitted.

To include tracking of custom events for the mobile app, the following callbacks need to be included in the application code.

```
// Emits a custom event without parameters

[[R1Emitter sharedInstance] emitEvent:@"Your custom event name"];

// Emits a custom event with parameters

[[R1Emitter sharedInstance] emitEvent:@"Your custom event name"
    withParameters:@{@"key":"value"}];
```