Apple iOS Application Analytics SDK

Integration Guide

Contents

DOCUMENT CHANGE LOG **4**

INTRODUCTION **5**

BEFORE YOU START **5**

SETTING UP YOUR APPLICATIONID 5
SENDING ANALYTICS DATA TO BANGO 6

SAMPLE IMPLEMENTATION **7**

METHODS AND INTERFACES **11**

CONFIGURATION METHODS 11
applicationChannel *11*
applicationType *11*
applicationVersion *11*
continueSessionSeconds *11*
debugLog *12*
findLocation *12*
logEnabled *12*
postURL *12*
setUserId *13*
SESSION METHODS 14
onStartSession *14*
onStartSessionWithEventParameters *14*
onEndSession *14*
onEndSessionWithEventParameters *15*
onIdle *15*
onIdleWithEventParameters *15*
onResume *16*
onResumeWithEventParameters *16*
LOGGING METHODS 17
onDemandVideoStart *17*
onDemandVideoStartWithEventParameters *17*
onDemandVideoStop *17*
onDemandVideoStopWithEventParameters *18*
onLiveVideoStart *18*
onLiveVideoStartWithEventParameters *18*
onLiveVideoStop *19*
onLiveVideoStopWithEventParameters *19*
onSearch *19*
onSearchWithEventParameters *20*
onPageView *20*
onPageViewWithEventParameters *20*
onEvent *21*
onError *21*
onErrorWithEventParameters *21*

REFERENCE INFORMATION **22**

CUSTOM PARAMETERS 22
CAMPAIGN PARAMETERS 24

FAQ 25

I AM TRYING TO SEND ANALYTICS DATA IN THE BACKGROUND BUT IT DOES NOT SEEM TO BE WORKING. WHAT IS THE ISSUE? 25
HOW DO I KNOW WHEN TRANSMISSION HAS COMPLETED AND WHETHER IT HAS SUCCEEDED FOR FAILED? 26
I AM GETTING ISSUES WHEN I TRY TO TRANSMIT DATA MANUALLY. WHAT IS HAPPENING? 26
HOW DO I ADD THE API AS A STATIC LIBRARY? 26

Document change log

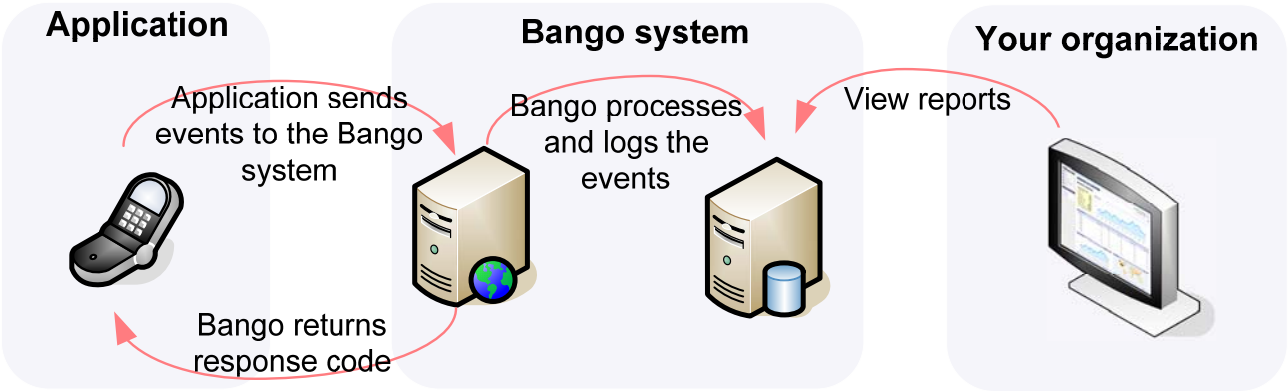
Date	Version	Change detail
15 September 2010	1.0	Created
07 October 2010	1.0.1	Corrected BGO_VALUECUR and BGO_COSTCUR parameter names – incorrectly specified with two 'R's.
29 March 2011	2.0.0	Added the ability to transmit data when the device goes into idle
01 June 2011	3.0.0	Removed dependency on ASIAsyncHttp library Classes renamed to be prefixed with BGO_ to avoid conflicts with other libraries Dependency added to Reachability library (If users have their own Reachability dependency then either can be used)
12 July 2011	4.0.0	Updated the way that data is transmitted in the background
7 October 2011	5.0.0	Removed core data dependency. The core data persistence layer has been replaced with a SQLite database. Added BGOAnalyticsManagerDelegate functionality. Added FAQ section.
22 November 2011	5.0.1	Corrected BGO_APP_SESSION_ERROR event name to BGO_APP_ERROR SDK version changed from number to string to allow revision version number Fixed bug that resulted in exclusion of latest profile from transmit
31 January 2012	5.0.2	Bango DB Storage moved from documents to caches folder Fixed bug that caused framework to crash when accessing local db from multiple threads Changed use of equality operator to assignment in Analytics manage initialiser
14 February 2012	5.0.3	Synchronised access to database to improve thread safety Added better implementation of sharedPersistenceManager

Introduction

Bango application analytics is an analytics solution for mobile applications. When an event happens in a mobile application it is sent to Bango. Bango will then store this data and it will be available to be reported on. This document provides detail on integrating your mobile application with our recording server through the Bango SDK.

Please note, the Bango Analytics integration is for use by those with significant development experience.

This diagram shows the application analytics architecture:



Before you start

It is recommended that you use a separate Analytics account for integration testing so that your test data is separate from your live data. Once the integration is ready to go live you can switch to your live account.

Please contact your account manager or Bango customer services for information or sign up for a one month free trial to get started at Bango.com.

How application analytics works

Setting up your ApplicationId

Each application that you wish to use Bango analytics for needs an ApplicationId. An ApplicationId is a simply a unique reference for your application. If your application is distributed on different platforms you have a choice with how to use the ApplicationId:

1. Use a single ApplicationId for all versions of the application across all platforms. This will mean that the data from all versions of the application will be aggregated for display in the app analytics reports. You can filter the reports by OS version to see data from individual apps.
2. Use a unique ApplicationId for every different version of the application. This will mean that the data from each version of the platform will be displayed separately in the app analytics reports. This gives you the highest level of granularity for your data.

To set up an ApplicationId visit the ‘Manage’ section of the Bango.com Management Tools and click on ‘Analytics set-up’ in the left menu then click on ‘App analytics’.

Sending analytics data to Bango

Once an event has occurred in a mobile application it can be sent straight away or stored in the mobile device’s memory to be sent at a later time. Bango has a concept of batches and events. You can send one batch with each transmission and each batch may contain many events. This model suits the storing of events on the mobile device’s memory ready for sending as a batch when the application closes. This means that the sending of event data does not create any network traffic when the application is in use. This model is less network intensive because data common to each event is only sent once with each batch.

Sample Implementation

1. Create an XCode project and Unzip frameworks folder and drop into your “classes” directory.
Add these files to your project. Please refer to the FAQ section for more information about building the Bango source into a static library.

2. Paste this bit of code into the `application:didFinishLaunchingWithOptions:` method of your applications delegate *AppDelegate.m.

```
/*
  BGO_ANALYTICS_DID_FINISH_LAUNCHING_WITH_OPTIONS_START
   */

    // Init BangoAnalytics - setting the managed object context
  BGOAnalyticsManager *analytics = [BGOAnalyticsManager sharedManager];

  // set the uncaught exception handler
  NSSetUncaughtExceptionHandler(&uncaughtExceptionHandler);

  // config options here
  analytics.findLocation = YES;
  analytics.debugLog = YES;
  analytics.applicationChannel = @"DirectToDevice";
  analytics.applicationType = @"test-application for analytics";
  analytics.continueSessionSeconds = 10;
  analytics.postUrl = BGO_SETTING_POST_URL_DEBUG;

  // enable / disable event logging
  analytics.logEnabled = YES;

  // set transmission to automatic
  analytics.automaticDataTransmission = YES;

  // start the session
  [analytics onStartSessionWithApplicationId:BANGO_APPLICATION_ID];

  // try and post any data which is not the current profile we have just
created
  // no need to use this if automaticDataTransmission is enabled
  // [analytics commenceTransmissionWithMode:TRANSMIT_EXCLUDE_LATEST];

  /*
   BGO_ANALYTICS_DID_FINISH_LAUNCHING_WITH_OPTIONS_END
   */
```

3. Add this method to the application delegate to log uncaught exceptions – this is optional.
N.B. The code will not log exceptions to Bango if a valid session does not exist.

```
void uncaughtExceptionHandler(NSException *exception) {
  BGOAnalyticsManager *analytics = [BGOAnalyticsManager sharedManager];
```

```
  NSMutableString *exceptionString = [NSMutableString stringWithString:@"name:
"];
  [exceptionString appendString:exception.name];
  [exceptionString appendString:@" reason: "];
  [exceptionString appendString:exception.reason];

  NSNumber * errorLevel = [NSNumber numberWithInt:BGO_ERROR_INTERNAL_CRITICAL];

  [analytics onErrorWithEventParameters:nil message:exception.name
eventClass:@"Uncaught Exception"

eventErrorLevel:errorLevel eventErrorId:exceptionString];
}
```

Declare it in the header file;

```
void uncaughtExceptionHandler(NSException *exception);
```

4. If you would like you application to send all logged events when the user sends the application to the background, paste the following code into `applicationWillResignActive:(UIApplication *)application` method your application delegate. Note this is not the only way to achieve this effect and will only work on versions of iOS > 4.0. For more information please consult our FAQ section.

```
/*
   BGO_ANALYTICS_APPLICATION_WILL_RESIGN_ACTIVE_START
   */

    // Log Idle Event
    UIDevice * device = [UIDevice currentDevice];
    BOOL backgroundSupported = NO;
    if ([device respondsToSelector:@selector(isMultitaskingSupported)]) {
        backgroundSupported = device.multitaskingSupported;
    }

    if (backgroundSupported) {
        UIApplication * app = [UIApplication sharedApplication];

        backgroundTaskId_ = [app beginBackgroundTaskWithExpirationHandler:^{
            [app endBackgroundTask:backgroundTaskId_];
            backgroundTaskId_ = UIBackgroundTaskInvalid;
        }];

        // Start the long-running task and return immediately.
      dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT,
0), ^{

            // Do the work associated with the task.
          BGOAnalyticsManager *analytics = [BGOAnalyticsManager sharedManager];
            [analytics onIdleWithEventParameters:nil
                                    eventDetail:@"applicationWillResignActive"
                                   eventValue:nil fromBackground:YES];

            [app endBackgroundTask:backgroundTaskId_];
```

```
            backgroundTaskId_ = UIBackgroundTaskInvalid;
        });
    } else {
        // No multitasking support so make a regular call to onIdle...
      BGOAnalyticsManager *analytics = [BGOAnalyticsManager sharedManager];
        [analytics onIdleWithEventParameters:nil
                                eventDetail:@"applicationWillResignActive"
                                   eventValue:nil];

    }

   /*
    BGO_ANALYTICS_APPLICATION_WILL_RESIGN_ACTIVE_END
    */
```

5. Paste following code into the `applicationWillEnterForeground:(UIApplication *)application` of your application delegate class.

```
/*
    BGO_ANALYTICS_APPLICATION_WILL_ENTER_FOREGROUND_START
    */

     // Log Idle Event
   BGOAnalyticsManager *analytics = [BGOAnalyticsManager sharedManager];
   [analytics onResumeWithEventParameters:nil
eventDetail:@"applicationWillEnterForeground" eventValue:nil];

   /*
    BGO_ANALYTICS_APPLICATION_WILL_ENTER_FOREGROUND_END
    */
```

6. Paste the following code into `applicationWillTerminate:(UIApplication *)application` method of your application delegate.

```
/*
    BGO_ANALYTICS_APPLICATION_WILL_TERMINATE_START
    */

     // Log Idle Event
   BGOAnalyticsManager *analytics = [BGOAnalyticsManager sharedManager];
   [analytics onEndSessionWithEventParameters:nil
eventDetail:@"applicationWillTerminate" eventValue:nil];

   /*
    BGO_ANALYTICS_APPLICATION_WILL_TERMINATE_END
    */
```

7. Ensure that you have imported all the necessary API classes:

```
/*
BGO_ANALYTICS_IMPORTS_START
```

```
 */
#import "BGOAnalyticsConstants.h"
#import "BGOAnalyticsManager.h"
#import "BangoConfigConstants.h"
#import "BGOTransmissionModeConstants.h"
/*
 BGO_ANALYTICS_IMPORTS_END
 */
```

9. Bango App Analytics API for iOS and its dependent libraries require the following frameworks and libraries to be included in your project for it to build successfully.

- `CFNetwork.framework`
- `SystemConfiguration.framework`
- `MobileCoreServices.framework`
- `CoreGraphics.framework`
- `CoreLocation.framework`
- `UIKit.framework`
- `libsqlite.dylib`

Right click (or shift click) on frameworks in the file tree on the left of the IDE. Then choose *add existing framework*, and add all the frameworks above.

Methods and Interfaces

Configuration Methods

Method:

applicationChannel

Description: Gets or set the method used to distribute the application.

Inputs:

applicationChannel	NSString[50]	The method used to distribute the application, for example “AppStore”

Outputs:

applicationChannel	NSString[50]	The method used to distribute the application, for example “AppStore”

Method:

applicationType

Description: Gets or sets the type of application.

Inputs:

applicationType	NSString[50]	The type of application, for example “Lite”

Outputs:

applicationType	NSString[50]	The type of application, for example “Lite”

Method:

applicationVersion

Description: Gets or sets the application version to be used, if this is not supplied then the application version number of the application is used.

Inputs:

applicationVersion	NSString[50]	The application version to be used

Outputs:

applicationVersion	NSString[50]	The application version to be used

Method:

continueSessionSeconds

Description: Gets or sets the period in seconds, that after we receive onSessionEnd call we will still count events in that of the previous session.

Inputs:

continueSessionSeconds	Integer	This is the period in seconds

Outputs:

continueSessionSeconds	Integer	This is the period in seconds

Method:

debugLog

Description: Gets or sets whether or not debug logging should take place, by default this is set to false and should be switched off for production release.

Inputs:	debugLog	Boolean	Should debug events be logged
Outputs:	debugLog	Boolean	Should debug events be logged

Method:

findLocation

Description: Gets or sets whether or not an attempt should be made to discover the user's location, by default this is set to true.

Inputs:	findLocation	Boolean	Should location data be found
Outputs:	findLocation	Boolean	Should location data be found

Method:

logEnabled

Description: Gets or sets whether or not logging should take place, by default this is set to true.

Inputs:	logEvents	Boolean	Should events be logged
Outputs:	logEvents	Boolean	Should events be logged

Method:

postURL

Description: Gets or sets the URL to post the data to, if this is not supplied then the default bango url is used.

Inputs:	url	NSString	The URL to post the data to
Outputs:	url	NSString	The URL to post the data to

Method:	**setUserId**
Description:	Gets or sets the unique identifier that you have for the user of the application. For example, this could be an email or a GUID or just a numerical value. If this is not supplied then one will be generated based on device information.
Inputs:	userId | NSString[50] | The unique identifier for the user
Outputs:	userId | NSString[50] | The unique identifier for the user

Session Methods

Method:

onStartSession

Description: This method initiates logging, this must be called after the configuration methods but before any other logging or any other session methods.

Inputs:

applicationKey	NSString[50]	The ApplicationId that has been set up for the application using the Management Tools

Outputs:

None		

Method:

onStartSessionWithEventParameters

Description: This method initiates logging, this must be called after the configuration methods but before any other logging or any other session methods.

Inputs:

applicationKey	NSString[50]	The ApplicationId that has been set up for the application using the management tools
eventDetail	NSString[200]	Any extra detail that you would like to attach to the event
eventValue	NSString[50]	This is a value that you can associate with the event
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onEndSession

Description: This method indicates that the session has ended and will initiate the transmitting of data to the specified url.

Inputs:

None		

Outputs:

None		

Method:

onEndSessionWithEventParameters

Description: This method indicates that the session has ended and will initiate the transmitting of data to the specified url.

Inputs:

eventDetail	NSString[200]	Any extra detail that you would like to attach to the event
eventValue	NSString[50]	This is a value that you can associate with the event
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onIdle

Description: This method indicates that the session has entered an idle state, for example the application has lost focus.

Inputs:

None		

Outputs:

None		

Method:

onIdleWithEventParameters

Description: This method indicates that the session has entered an idle state, for example the application has lost focus.

Inputs:

eventDetail	NSString[200]	Any extra detail that you would like to attach to the event
eventValue	NSString[50]	This is a value that you can associate with the event
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onResume

Description: This method indicates that the session has resumed after an idle state, for example the application has gained focus after losing focus.

Inputs:

None		

Outputs:

None		

Method:

onResumeWithEventParameters

Description: This method indicates that the session has resumed after an idle state, for example the application has gained focus after losing focus.

Inputs:

eventDetail	NSString[200]	Any extra detail that you would like to attach to the event
eventValue	NSString[50]	This is a value that you can associate with the event
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Logging Methods

Method:

onDemandVideoStart

Description: This method logs an onDemandVideoStart event.

Inputs:

None		

Outputs:

None		

Method:

onDemandVideoStartWithEventParameters

Description: This method logs an onDemandVideoStart event.

Inputs:

eventDetail	NSString[200]	Any extra detail that you would like to attach to the event. This should contain the video title
eventValue	NSString[50]	This is a value that you can associate with the event
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onDemandVideoStop

Description: This method logs an onDemandVideoStop event.

Inputs:

None		

Outputs:

None		

Method:

onDemandVideoStopWithEventParameters

Description: This method logs an onDemandVideoStop event.

Inputs:

eventDetail	NSString[200]	Any extra detail that you would like to attach to the event. This should contain the video title
eventValue	NSString[50]	This is a value that you can associate with the event
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onLiveVideoStart

Description: This method logs a LiveVideoStart event.

Inputs:

None		

Outputs:

None		

Method:

onLiveVideoStartWithEventParameters

Description: This method logs a LiveVideoStart event.

Inputs:

eventDetail	NSString[200]	Any extra detail that you would like to attach to the event. This should contain the video title
eventValue	NSString[50]	This is a value that you can associate with the event
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onLiveVideoStop

Description: This method logs a LiveVideoStop event.

Inputs:

None		

Outputs:

None		

Method:

onLiveVideoStopWithEventParameters

Description: This method logs a LiveVideoStop event.

Inputs:

eventDetail	NSString[200]	Any extra detail that you would like to attach to the event. This should contain the video title
eventValue	NSString[50]	This is a value that you can associate with the event
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onSearch

Description: This method logs a search event.

Inputs:

None		

Outputs:

None		

Method:

onSearchWithEventParameters

Description: This method logs a search event.

Inputs:

eventDetail	NSString[200]	Any extra detail that you would like to attach to the event
eventValue	NSString[50]	This is a value that you can associate with the event
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onPageView

Description: This method increments the page view count.

Inputs:

None		

Outputs:

None		

Method:

onPageViewWithEventParameters

Description: This method increments the page view count.

Inputs:

eventDetail	NSString[200]	Any extra detail that you would like to attach to the event
eventValue	NSString[50]	This is a value that you can associate with the event
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onEvent

Description: This method logs a custom event.

Inputs:

eventName	NSString[50]	The event name is identifier for the type of event that you would like to record. You can create your own event names simply sending it in this field.
eventDetail	NSString[200]	Any extra detail that you would like to attach to the event
eventValue	NSString[50]	This is a value that you can associate with the event
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onError

Description: This method logs an error event.

Inputs:

None		

Outputs:

None		

Method:

onErrorWithEventParameters

Description: This method logs an error event.

Inputs:

message	NSString[50]	Any extra detail that you would like to attach to the error
errorId	NSString[50]	The id you want to assign to the error
errorClass	NSString[50]	The class the error occurred in
eventParameters	NSDictionary	A NSDictionary collection of event parameters as defined by the custom parameters table

Outputs:

None		

Reference Information

Custom parameters

These are the custom parameters that can be attached to an event.

Bango parameter	Recommended use	Maximum size
bgo_region	The region parameter can be used to indicate the regional variation of an app or as a geographical indicator.	200 characters
bgo_section	The section parameter should be used to record which section of an app the event was generated in e.g. News, Finance, Sport. Use in conjunction with bgo_subsection to give a finer level of granularity.	200 characters
bgo_subsection	The subsection parameter should be used to define a sub-section of an app (e.g. US News, UK News, World News). Use in conjunction with bgo_section to give a greater level of detail.	200 characters
bgo_media	The media parameter should be used to define the media type being analysed (e.g. video, ringtone, audio). Use in conjunction with bgo_author to give a greater level of detail.	200 characters
bgo_author	The author parameter should be used to indicate the author of content. Use in conjunction with bgo_media to give a greater level of detail.	200 characters
bgo_gender	The gender parameter should be used to indicate the gender of the user. Used to record male or female, any other value will be ignored.	M, MALE, F or FEMALE
bgo_age	The age parameter should be used to indicate the age of a user. The following ages are mapped to a single letter: < 16 yrs: A 16 - 20 yrs: B 21 - 25 yrs: C 26 - 30 yrs: D 31 - 35 yrs: E 36 - 40 yrs: F 41 - 50 yrs: G 51 - 60 yrs: H 60 + yrs: I	A, B, C, D, E, F, G, H or I, any other value will be ignored.
bgo_identity	The identity parameter should be used to	200 characters

	record an identity you have for a user. This is separate from the Bango User ID and should be used for an identity you might have for the user.	
bgo_value	The value parameter should be used to define a monetary value associated with an event. Can be used in conjunction with bgo_valuecur to give a greater level of detail.	200 characters
bgo_valuecur	The value currency parameter should be used to define the currency (code, symbol or description) for the value defined above. To be used in conjunction with bgo_value.	10 characters
bgo_cost	The cost parameter should be used to define a cost associated with an event. Can be used in conjunction with bgo_costcur to give a greater level of detail.	200 characters
bgo_costcur	The cost currency parameter should be used to define the currency (code, symbol or description) for the cost defined above. To be used in conjunction with bgo_cost.	10 characters
bgo_custom1	To be used to record a custom parameter, at reporting stage you will be able to map meaning to each custom parameter	200 characters
bgo_custom2	To be used to record a custom parameter, at reporting stage you will be able to map meaning to each custom parameter	200 characters
bgo_custom3	To be used to record a custom parameter, at reporting stage you will be able to map meaning to each custom parameter	200 characters
bgo_custom4	To be used to record a custom parameter, at reporting stage you will be able to map meaning to each custom parameter	200 characters
bgo_custom5	To be used to record a custom parameter, at reporting stage you will be able to map meaning to each custom parameter	200 characters
bgo_custom6	To be used to record a custom parameter, at reporting stage you will be able to map meaning to each custom parameter	200 characters
bgo_custom7	To be used to record a custom parameter, at reporting stage you will be able to map meaning to each custom	200 characters

	parameter	
bgo_custom8	To be used to record a custom parameter, at reporting stage you will be able to map meaning to each custom parameter	200 characters
bgo_custom9	To be used to record a custom parameter, at reporting stage you will be able to map meaning to each custom parameter	200 characters
bgo_custom10	To be used to record a custom parameter, at reporting stage you will be able to map meaning to each custom parameter	200 characters
bgo_videotype	The video type parameter should be used to indicate if the video was live or on-demand	LIVE, ONDEMAND

Campaign parameters

These are parameters that are used to track campaign information across web pages. If you want to send campaign information out from your application these parameters can be used. For a full guide please see http://bango.custhelp.com/app/answers/detail/a_id/1189. You can also use these parameters within your application to associate information with events.

Bango parameter	Recommended use	Maximum size
bgo_campaign	Use to track the different advertising campaigns or product promotions that your business creates. An example of this would be a mobile content store tracking the effectiveness of their “Hot Summer Ringtone Sale” advertising campaign.	200 characters
bgo_source	Visitors to your website must come from somewhere. That is, each referral to a website has an origin, or Source . Examples of sources are the Google search engine, an email newsletter, or a referring web site. There may be several Sources for each campaign. For example, the “Hot Summer Ringtone Sale” is advertised in both an email newsletter and a banner ad. In this case, both “newsletter” and “banner ad” would be possible Sources.	200 characters
bgo_medium	The Medium helps to qualify the source. Together, the Source and Medium provide specific information about the origin of a referral. For example, if the Source is “Google,” the medium might be	200 characters

	“CPC,” indicating a sponsored link. Or, the Medium might be “organic,” indicating a link in the unpaid search results. In the case of a “newsletter” Source, examples of Medium include “email” and “print.”	
bgo_content	The Content variable indicates the version of an ad on which a visitor clicked. Labelling your content versions allows you to determine which one is most effective at attracting profitable leads. For example, if you had two versions of a banner ad, you could use the Content parameter to identify which one is bringing more visits to your site.	200 characters
bgo_term	The Term is a keyword or phrase that matches what a user types into a search engine. For example, a link in a cost-per-click (CPC) ad would be tagged with the Term that triggered the ad. In our example, the Term might be “hot ringtones.”	200 characters
bgo_owner	The Owner parameter can be used to assign an individual, group or organisation responsibility for a campaign link.	200 characters

FAQ

I am trying to send analytics data in the background but it does not seem to be working. What is the issue?

There are various methods to achieve this and most of them will only work on versions of iOS that are greater than 4.0. In older versions of iOS you could choose to use the applicationWillTerminate method to perform certain tasks like saving state etc. but the application had a very limited window to complete tasks before it was considered unresponsive and terminated by iOS.
In version 4.0 and later developers can use code blocks otherwise known as lambda functions to perform various tasks. We use the dispatch queue in conjunction with blocks to run methods after an application has left the foreground to send data to the Bango service. If you intend to use this method or any other thread based calls to send data, you have to ensure that you use the background variants of the Api calls since NSConnection callbacks return on the main thread which may not be available in some cases (especially if the application is now is the background).
Users can place this code in multiple places e.g. in the applicationWillResignActive or applicationDidEnterBackground. There is also a mechanism to subscribe via NSNotificationCenter to UIApplicationDidEnterBackgroundNotification and add the code block for posting data in the selector that responds to said notification. All of these are valid but only one of them should be used – although this has no effect on the data, we cannot guarantee how the application will behave and some users have experienced crashes.
Please note that if you use the background Api methods from the main application thread, this will call the Bango service synchronously and cause your application to appear to hang while the data is sent over the connection and a response is obtained or the call times out. Although this should work it does not make for the best user experience and should be avoided.

How do I know when transmission has completed and whether it has succeeded for failed?

In version 5.0 we have added the new protocol that you can implement if you want a class to be notified when data transmission has been completed. The protocol has 2 optional methods called transmitSuccess and transmitFailure. All you need to do in your class to get those methods called in your class is to set an instance of that class as the BGOAnalyticsManager delegate using the following code

```
[BGOAnalyticsManager sharedManager].delegate = self;
```

Please note that the shared analytics manager can only have one delegate so if you set a different class to be the delegate the last instance is the one that will be called.

I am getting issues when I try to transmit data manually. What is happening?

This may happen if you have enabled automatic transmission mode. If you have enabled automatic transmission mode but are still sending data manually then it is quite likely that you might end up trying to transmit data twice. We go through some length to ensure that this has no effect on the integrity of the data stored by the Api but we cannot guarantee the behaviour of your application. In short if you want to send data manually then do not enable automatic transmission.

How do I add the API as a static library?

It is useful to be able to have code that you do not change very often/at all as a separate target in your XCode project – in this case as a static library. Although XCode compiler is very optimised such that it does not attempt to build code that has not been changed, having a separate target means that you can clean and rebuild your application as many times as you like and the compiler will not rebuild other targets. You can still clean all targets to get the compiler to rebuild them.

To add the Bango iOS Api as a separate target follow these steps:

1) Select your project file in the project browser.
2) In the editor window select the add target button (located in the bottom left hand corner of the screen).
3) From the list of target templates select the Cocoa Touch Static Library followed by the next button.
4) Enter the name of the Api e.g. bangoapi then select the finish button. This will create an empty static library target.
5) Drag the bango source folder into the project and make sure that the correct target is selected i.e. bangoapi
6) Select the bangoapi target and expand the Copy headers section.
7) Drag the following header files into the project sub section:
    - BGOPersistenceManager.h
    - BGOAnalyticsManagerDelegate.h
    - BGOAnalyticsManager.h
    - schema/BGODataType.h
    - schema/BGOProfile.h
    - schema/BGOEvent.h
    - schema/BGOEventParameter.h
8) Expand the Link Binary with Libraries section under the Build Phases tab.
9) Add the following frameworks and libraries to the list – these will also need to be added you your application link section as you would have to do if you were using a single target:
    - SystemConfiguration.framework
    - CFNetwork.framework
    - UIKit.framework

- Foundation.framework
- libsqlite3.dylib
- CoreLocation.framework

10) Locate the bangoapi precompiled header file (.pch) and add the following import statements
    - `#import <Foundation/Foundation.h>`
    - `#import <UIKit/UIKit.h>`
11) Under Build settings, locate the Other linker flags value and change it to **-ObjC -all_load**.
12) Finally select your application target and add the new target to the Link Binary with Libraries section.