QT 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
SYMBIAN REQUIRED CAPABILITIES 6

SAMPLE IMPLEMENTATION 7

INTEGRATING USING PRE-BUILT LIBRARIES 8

METHODS AND INTERFACES 9

CONFIGURATION METHODS 9
setApplicationChannel 9
getApplicationChannel 9
setApplicationType 9
getApplicationType 9
setApplicationVersion 10
getApplicationVersion 10
setContinueSessionSeconds 10
getContinueSessionSeconds 10
setFindLocation 11
getFindLocation 11
setLogEnabled 11
getLogEnabled 11
setPostURL 12
getPostURL 12
setUseOfflineStorage 12
getUseOfflineStorage 12
setUserId 12
getUserId 13
SESSION METHODS 14
onStartSession 14
onEndSession 14
onIdle 15
onResume 15
LOGGING METHODS 16
onDemandVideoStart 16
onDemandVideoStop 16
onLiveVideoStart 16
onLiveVideoStop 17
onSearch 17
onPageView 17
onEvent 18
onError 18
POSTING METHODS 19
postEvents 19
DEBUGGING METHODS 20
ViewBatch 20

REFERENCE INFORMATION **21**
CUSTOM PARAMETERS 21
CAMPAIGN PARAMETERS 23

Document change log

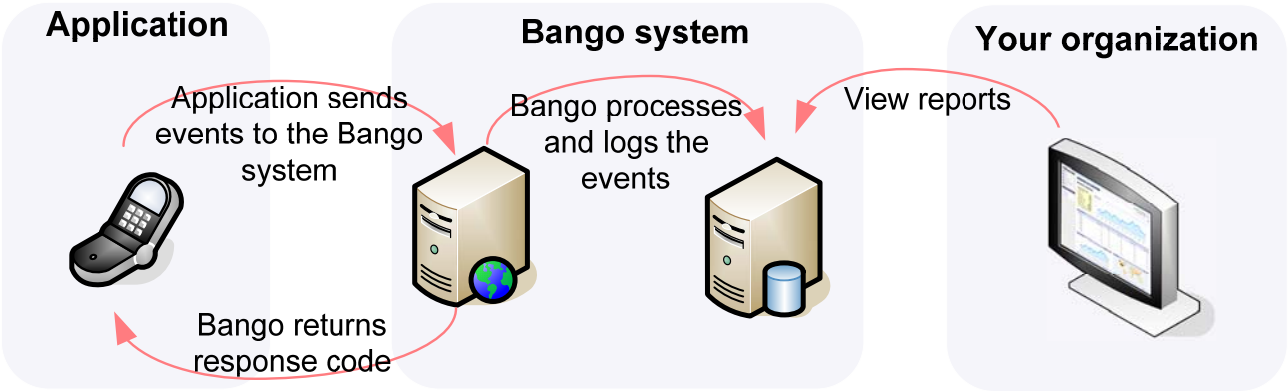
Date	Version	Change detail
21 June 2011	1.0.0	Created
19 July 2011	1.0.1	Missing parameter added to onEndSession - default behaviour changed to post on exit. Changed from using standard string class to using QT-specific QString class
21 July 2011	1.0.2	Default behaviour on onEndSession changed back to save on exit
16 August 2011	1.0.3	Changes to handle Carbide RegisterMetaType error.
07 October 2011	1.1.0	QJson library project removed Instructions on building app added

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 ‘Setup and Config’ section of the Bango.com Management Tools and click on ‘Analytics set-up’ in the left menu then click on ‘Application 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.

Symbian Required Capabilities

The SDK requires Symbian extended capabilities (http://developer.qt.nokia.com/wiki/Symbian_Capabilities) in order to access some of the required functionality on the device. To use these capabilities, an application will require approval by Symbian Signed (www.symbiansigned.com). Development certificates can be issued for use on a fixed set of devices during development and testing.

The required capabilities are:

Name	Reason
NetworkServices	Transmit data to specified URL
ReadDeviceData	Allow detection of device information
Location	Allow detailed location data

Sample Implementation

There are 2 methods that must be called for application analytics to perform correctly. These must be called **after** any of the configuration options have been set:

Insert a call to BangoAgent::onStartSession (QString), passing it the ApplicationId. We recommend placing this call in the constructor method in your application.

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent), ui(new Ui::MainWindow)
{
       // Configuration options set

       BangoAgent::onStartSession(ApplicationID);

       // your code

}
```

Insert a call to BangoAgent::onEndSession() when a session is complete. . We recommend placing this call in the destructor method in your application.

```
MainWindow::~MainWindow()
{
       BangoAgent::onEndSession();

       // your code
}
```

You can then use either the generic method to log an event:

BangoAgent::onEvent(eventName, eventDetail, eventValue, eventParameters)

Alternatively you can use a predefined event method to log a predefined event type:

BangoAgent::onSearch(eventName, eventDetail, eventValue, eventParameters)
BangoAgent::onPageView(eventName, eventDetail, eventValue, eventParameters)

Please see the "Methods And Interfaces" section below for further information.

Integrating Using Pre-built Libraries

You can also use pre-built libraries to integrate your app.

1. Open the BangoNokiaQTRoot.pro project in QT Creator, and pick a suitable toolchain.
2. Right-click on the BangoNokiaQTLibrary and select "Build".
3. Check that the .dsos (BangoNokiaQTLibrary.dso and BangoNokiaQTLibrary{000a0000}.dso) are in your toolchain's directory.
4. Copy the header files (BangoNokiaQTLibrary_global.h and BangoAgent.h) and the .dll (BangoNokiaQTLibrary.dll) into your application folder, and add a link to them in your .pro file by adding the following.

```
LIBS +=  -lBangoNokiaQTLibrary
addFiles.sources += BangoNokiaQTLibrary.dll
addFiles.path += !:/sys/bin
DEPLOYMENT += addFiles
```

5. Configure your app to use the same toolchain as selected above.
6. Make sure to include the header file wherever you wish to make a call to the library.
7. Set a proper signing key for the sis file, and build.

The SDK includes libraries pre-built for the ARMV5 Symbian^3 toolchain. To use these, copy the .dsos to the toolchain directory (this is "C:\QtSDK\Symbian\SDKs\Symbian3Qt*xxx*\epoc32\release\armv5\lib" on the QT SDK default installation, where *xxx* is the QT version), and follow the above instructions from step 4.

Methods and Interfaces

Configuration Methods

Method:	**setApplicationChannel**		
Description:	The method used to distribute the application.		
Inputs:	applicationChannel	QString[50]	The method used to distribute the application, for example “AppStore”
Outputs:	None		

Method:	**getApplicationChannel**		
Description:	Returns method used to distribute the application.		
Inputs:	None		
Outputs:	applicationChannel	QString[50]	The method used to distribute the application, for example “AppStore”

Method:	**setApplicationType**		
Description:	The type of application.		
Inputs:	applicationType	QString[50]	The type of application, for example “Lite”
Outputs:	None		

Method:	**getApplicationType**		
Description:	Returns the type of application.		
Inputs:	None		
Outputs:	applicationType	QString[50]	The type of application, for example “Lite”

Method:

setApplicationVersion

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

Inputs:

applicationVersion	QString[50]	The application version to be used

Outputs:

None		

Method:

getApplicationVersion

Description: Returns the application version to be used.

Inputs:

None		

Outputs:

applicationVersion	QString[50]	The application version to be used

Method:

setContinueSessionSeconds

Description: 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:

None		

Method:

getContinueSessionSeconds

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

Inputs:

None		

Outputs:

continueSessionSeconds	Integer	This is the period in seconds

Method:

setFindLocation

Description: 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:

None		

Method:

getFindLocation

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

Inputs:

None		

Outputs:

findLocation	Boolean	Should location data be found

Method:

setLogEnabled

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

Inputs:

logEvents	Boolean	Should events be logged

Outputs:

None		

Method:

getLogEnabled

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

Inputs:

None		

Outputs:

logEvents	Boolean	Should events be logged

Method:

setPostURL

Description: The URL to post the data to, if this is not supplied then the default Bango URL is used.

Inputs:

url	QString	The URL to post the data to

Outputs:

None		

Method:

getPostURL

Description: Returns the URL to post the data to.

Inputs:

None		

Outputs:

url	QString	The URL to post the data to

Method:

setUseOfflineStorage

Description: Whether or not offline storage should be used, the recommend setting for this is true. This will mean that data will not be lost should there be problem transmitting the data.

Inputs:

useOfflineStorage	Boolean	Should off-line storage be used

Outputs:

None		

Method:

getUseOfflineStorage

Description: Returns whether or not offline storage should be used.

Inputs:

None		

Outputs:

useOfflineStorage	Boolean	Should off-line storage be used

Method:

setUserId

Description: 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	QString[50]	The unique identifier for the user

Outputs:

None		

Method:	**getUserId**		
Description:	The unique identifier for the user of the application.		
Inputs:	None		
Outputs:	userId	QString[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	QString[50]	The ApplicationId that has been set up for the application using the Management Tools
eventDetail	QString[200] (Optional)	Any extra detail that you would like to attach to the event
eventValue	QString[50] (Optional)	This is a value that you can associate with the event
eventParameters	QMap<QString, QString> (Optional)	A hash table 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. If force event posting is set to true, events will be posted from the main application thread, after the onEndSession event is logged.

Inputs:

forceEventPosting	Boolean (Optional)	Whether to force posting of events. Defaults to false.
eventDetail	QString[200](Optional)	Any extra detail that you would like to attach to the event
eventValue	QString[50](Optional)	This is a value that you can associate with the event
eventParameters	QMap<QString, QString> (Optional)	A hash table 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. The normal location for this would be in the onPause method of application.

Inputs:

eventDetail	QString[200](Optional)	Any extra detail that you would like to attach to the event
eventValue	QString[50](Optional)	This is a value that you can associate with the event
eventParameters	QMap<QString, QString> (Optional)	A hash table 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. The normal location for this would be in the onResume method of application. However it must be noted that the onResume method is also called after the onCreate Method.

Inputs:

eventDetail	QString[200](Optional)	Any extra detail that you would like to attach to the event
eventValue	QString[50](Optional)	This is a value that you can associate with the event
eventParameters	QMap<QString, QString> (Optional)	A hash table 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:

eventDetail	QString[200](Optional)	Any extra detail that you would like to attach to the event. This should contain the video title
eventValue	QString[50](Optional)	This is a value that you can associate with the event
eventParameters	QMap<QString, QString> (Optional)	A hash table collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onDemandVideoStop

Description: This method logs an onDemandVideoStop event.

Inputs:

eventDetail	QString[200] (Optional)	Any extra detail that you would like to attach to the event. This should contain the video title
eventValue	QString[50] (Optional)	This is a value that you can associate with the event
eventParameters	QMap<QString, QString> (Optional)	A hash table collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onLiveVideoStart

Description: This method logs a LiveVideoStart event.

Inputs:

eventDetail	QString[200] (Optional)	Any extra detail that you would like to attach to the event. This should contain the video title
eventValue	QString[50] (Optional)	This is a value that you can associate with the event
eventParameters	QMap<QString, QString> (Optional)	A hash table collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onLiveVideoStop

Description: This method logs a LiveVideoStop event.

Inputs:

eventDetail	QString[200] (Optional)	Any extra detail that you would like to attach to the event. This should contain the video title
eventValue	QString[50] (Optional)	This is a value that you can associate with the event
eventParameters	QMap<QString, QString> (Optional)	A hash table collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onSearch

Description: This method logs a search event.

Inputs:

eventDetail	QString[200] (Optional)	Any extra detail that you would like to attach to the event
eventValue	QString[50] (Optional)	This is a value that you can associate with the event
eventParameters	QMap<QString, QString> (Optional)	A hash table collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onPageView

Description: This method increments the page view count.

Inputs:

eventDetail	QString[200] (Optional)	Any extra detail that you would like to attach to the event
eventValue	QString[50] (Optional)	This is a value that you can associate with the event
eventParameters	QMap<QString, QString> (Optional)	A hash table collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onEvent

Description: This method logs a custom event.

Inputs:

eventName	QString[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	QString[200] (Optional)	Any extra detail that you would like to attach to the event
eventValue	QString[50] (Optional)	This is a value that you can associate with the event
eventParameters	QMap<QString, QString> (Optional)	A hash table collection of event parameters as defined by the custom parameters table

Outputs:

None		

Method:

onError

Description: This method logs an error event.

Inputs:

message	QString[50](Optional)	Any extra detail that you would like to attach to the error
errorId	QString[50](Optional)	The id you want to assign to the error
errorClass	QString[50](Optional)	The class the error occurred in
eventParameters	QMap<QString, QString> (Optional)	A hash table collection of event parameters as defined by the custom parameters table

Outputs:

None		

Posting Methods

Method:	**postEvents**		
Description:	This method attempts to post all events that have occurred to the specified URL.		
Inputs:	None		
Outputs:	None		

Debugging Methods

Method:	**ViewBatch**
Description:	This method returns the JSON string for events and profile currently in memory.
Inputs:	None
Outputs:	QString | JSON of batch currently in memory

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