Google Analytics Documentation.

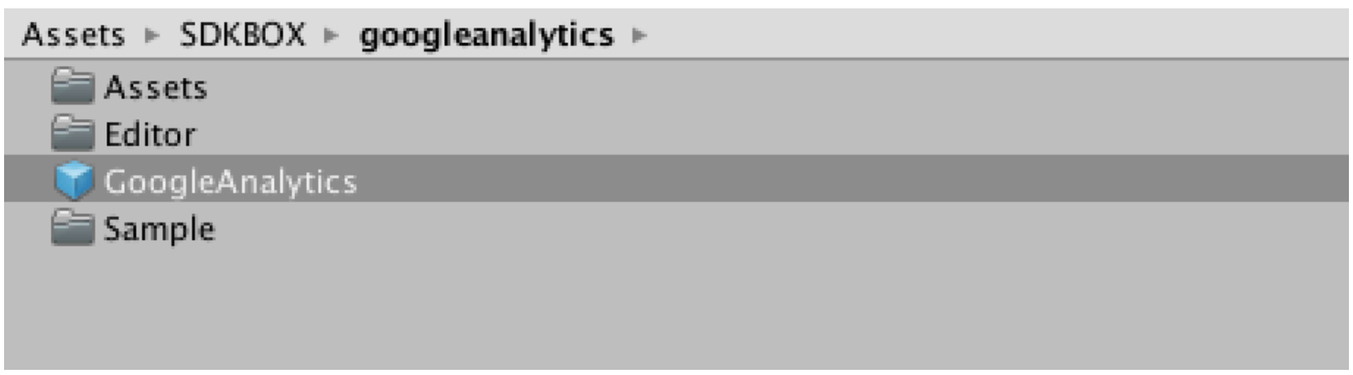
For more information, visit our website @ www.sdkbox.com

Importing SDKBOX GoogleAnalytics

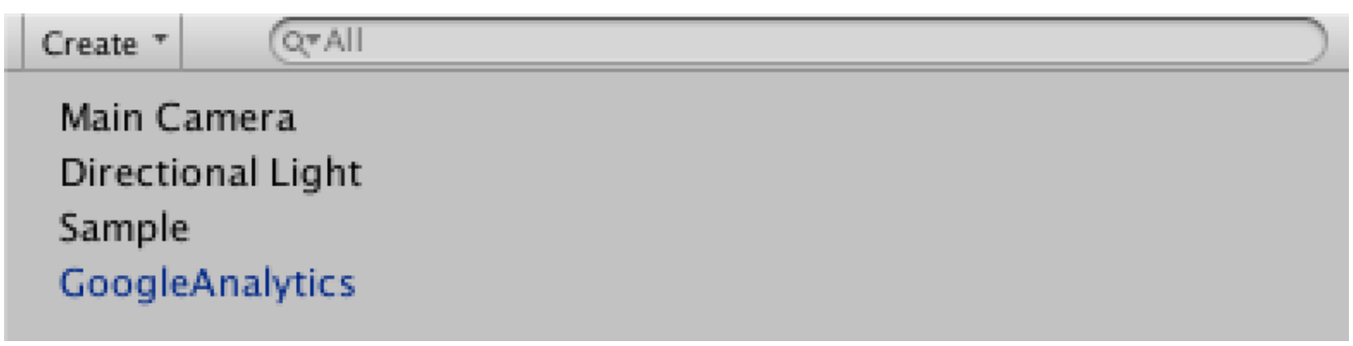
First import the `sdkbox_googleanalytics` unity package into your project. This will create a directory SDKBOX under Assets. If you have other SDKBOX plugins installed, they will appear here too. A plugins directory may also be present if you have other SDKBOX plugins. This directory contains files for iOS and Android.

Using the GoogleAnalytics Plugin

To begin using GoogleAnalytics plugin, find the GoogleAnalytics prefab in the Assets/SDKBOX/googleanalytics directory.

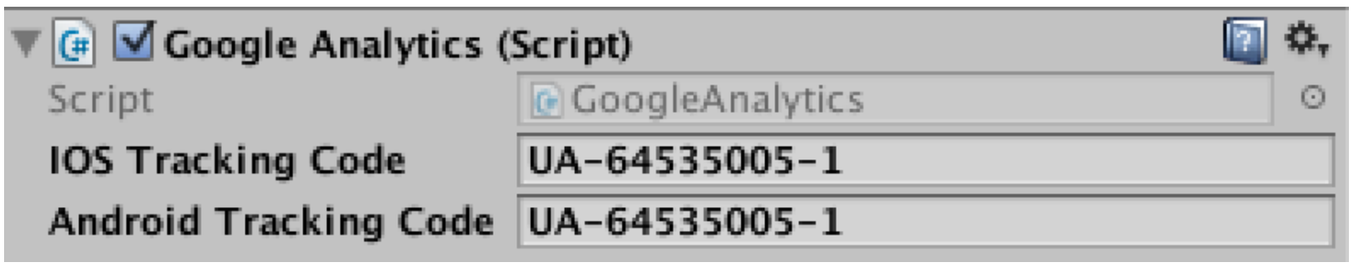


Drag an instance of this prefab into the scene where you want to start using analytics. You should only have a single instance of this prefab at any time.



Select the GoogleAnalytics game object in the hierarchy and in your inspector pane you can configure the object to complete the setup.

Configuring the GoogleAnalytics Plugin



Description of Fields

iOS Tracking Code

This is the tracking code from your Google Analytics account. You can use the same code for both platforms if you like, or you can have different codes for each. It's up to you.

Android Tracking Code

This is the tracking code from your Google Analytics account. You can use the same code for both platforms if you like, or you can have different codes for each. It's up to you.

GoogleAnalytics API

The analytics session is being explicitly started at plugin initialization time.

```
public void startSession();
```

You normally will never stop a session manually. But if you need to, you can call this manually.

```
public void stopSession();
```

Manually request dispatch of hits. By default, data is dispatched from the Google Analytics SDK for Android every 5 minutes.

```
public void dispatchHits();
```

Change the dispatch info time period to the desired amount of seconds.

```
public void dispatchPeriodically(int seconds);
```

Stop periodically sending info. Then manually the `dispatchPeridically` or `dispatchHits` should be called.

```
public void stopPeriodicalDispatch();
```

Set user ID for this tracking session.

```
public void setUser(string userID);
```

Set value for custom dimension.

```
public void setDimension(int index, string value);
```

Set value for custom metric.

```
public void setMetric(int index, string value);
```

Log screen info. title is the title of a screen. Screens are logical units inside your app you'd like to identify at analytics panel.

```
public void logScreen(string title);
```

logEvent("Achievement", "Unlocked", "Slay 10 dragons", 5);

```
public void logEvent(string eventCategory, string eventAction, string eventLabel, int
 value);
```

Log an exception. It is a basic support for in-app events.

```
public void logException(string exceptionDescription, bool isFatal);
```

Measure a time inside the application.

```
static void logTiming(string timingCategory, int timingInterval, string timingName, s
tring timingLabel);
```

Log a social event.

```
public void logSocial(string socialNetwork, string socialAction, string socialTarget);
```

While running on dry run, the tracked events won't be sent to the actual analytics account.

```
public void setDryRun(bool enable);
```

Enable advertising tracking when in google's ad vendors.

```
public void enableAdvertisingTracking(bool enable);
```

Create a tracker identified by the google analytics tracker id XX-YYYYYYYY-Z. If the tracker already existed, no new tracker will be created. In any case, the tracker associated with tracker id will be set as default tracker for analytics operations.

```
public void createTracker(string trackerId);
```

*** Enable a tracker identified by a trackerId. If the tracker does not exist, * nothing will happen.**

```
public void enableTracker(string trackerId);
```