Google Universal Analytics for Unity

Wrapper for Universal Analytics by Google, from Jetro Lauha / Strobotnik Ltd.

Table of Contents

Introduction

This package contains a helper class and usage examples for integrating analytics to projects created with Unity. The code offers a way to use Google Analytics service with implementation based on the new Measurement Protocol specification, which is part of *Google Universal Analytics*.

In April 2014 Google announced that Universal Analytics is not anymore in beta!

This document has been revised in May 2014.

Upgrade Guide

Here upgrade instructions for updating your analytics code to version **1.4.0**.

- **Rename** all instances of HitType.Appview to HitType.Screenview.
- **Rename** all addContentDescription calls to addScreenName.
- **Remove** all calls to addApplicationVersion. There is no need to add calls to setApplicationVersion (a new method) if you have set the app version to Analytics component. That version will automatically be added to all hits.
- **Check** if you want to use the new offline caching of hits. The Analytics component has this automatically enabled.

Additionally, note that following methods have been added, mostly due to additions to Measurement Protocol: setUserID, setIPOverride, setUserAgentOverride, setApplicationName, setApplicationID, setApplicationVersion, addApplicationInstallerID. When you use methods which begin with “set”, the given setting will apply to all analytics hits afterwards.

Setting Up a Profile in Google Analytics

1. Go to www.google.com/analytics/ and create an account if you don't already have one.

2. Click **Admin**, select the account you want to use, and then in the "Property" column choose "**Create new property**" from the drop-down menu.

3. Below the "What would you like to track?" question, select "Mobile app" and follow the step "a" below. (Or if you have some external reason to select "Website", you can do that as well and follow step "b".)

    a. *This step applies if you selected "Mobile app".* Enter name of your app in the App Name field. **Note**: Just to be sure, it's probably a good idea to use only letters and numbers, so select a name with no special characters – for example: CarSimulator2. This way you shouldn't hit any potential encoding problems later.
    b. *This step applies if you have some reason to select a "Website" instead.* In that case, Keep the "Select a tracking method" as "⊙ Universal Analytics BETA". Enter a Website Name for your property, and a valid Web Site URL for your app.

4. Select an Industry Category and correct Reporting Time Zone.

5. Click "Get Tracking ID". Now you have a new Property ID top of the page which looks like UA-XXXXXXX-Y (where X and Y are some numbers). Copy-paste that to your analytics initialization code as the trackingID parameter for the GoogleUniversalAnalytics class initialize() method (see `Analytics.cs`).

6. Return to Admin Home and make sure your new Property is still selected in the Property column. *If you selected "Mobile app" in step 3, you already have an auto-generated "All Mobile App Data" view which can be used, and the next step "a" for creating a new view is optional. However, if you selected "Website" in step 3, you must follow the next step "a" to create a custom view to be used for viewing of the analytics hits generated by your app.*

    a. *This step applies only if you chose "Website" in step 3.* In the "View" column choose "Create new view" from the drop-down menu. Select "Mobile app", and also enter a name for your App in the Reporting View Name field (see note about naming in step 3a). Click "Create View".

7. You entered an app name in either either step 3a or step 6a. Copy-paste that app name to your analytics initialization code as the appName parameter for the GoogleUniversalAnalytics class initialize() method (see `Analytics.cs`).

8. (Optional) View Settings of your new Profile. Select the currency you want to use. If you want to do Ecommerce tracking (send transaction and item hits), enable it as well.

Integrating Analytics Code To Your Unity Project

1. **Add** `Analytics.cs` and `GoogleUniversalAnalytics.cs` to your project Assets (this will happen automatically if you just import the package using Asset Store). You probably don't need to make any changes to the `GoogleUniversalAnalytics.cs`, but feel free to adapt `Analytics.cs` for your needs based on each project.

2. **Add a new** GameObject to the first scene of your project. **Name** the GameObject to e.g. "Analytics". Then **drag** the `Analytics.cs` script to it. You don't need to drag the `GoogleUniversalAnalytics.cs` script (not a MonoBehaviour), instead it is internally referred by `Analytics.cs`.

3. **Change** the tracking ID, app name and app version information (see previous chapter, "*Setting Up a Profile in Google Analytics*").

4. It's good idea to at least glance through the `Analytics.cs` code, and modify it to suit your needs. The example integration makes an anonymous client ID for analytics, sends client SystemInfo data on first launch and sends automatic app screen hits when a new level is loaded, with a given prefix (see OnLevelWasLoaded). The Analytics component is made persistent with DontDestroyOnLoad.

5. For all important screens which aren't separate scenes, add calls to Analytics.gua.sendAppScreenHit("screen name") when you enter the screen. Similarly, when important events or transactions happen, add analytics calls for those. Make custom helper methods to Analytics class based on what you need. A customized hit consists of these calls: Analytics.gua.beginHit(...), one or more Analytics.gua.addXXXXX(...) calls, and finally Analytics.gua.sendHit().

6. Check the Doxygen documentation for the GoogleUniversalAnalytics class for some more information (and to perhaps get more ideas what you could track as well).

7. In your web browser, go to the start page of Google Analytics and find your newly added app name & click it to go to App Overview analytics page. From left sidebar, go to the Real-Time → Overview page.

8. Run your Unity project and see if you can see a new screen view event appearing in the real-time page (assuming you have set up the app name and tracking id properly).

9. Note: If you checked “Use HTTPS” in the Analytics component, HTTPS does not seem to work when you run inside the editor, so you need to test with a stand-alone build.

10. Make yourself familiar with Google User-ID Policy:
    https://developers.google.com/analytics/devguides/collection/protocol/policy

    Specifically, you must give your end users proper notice about what and how you collect data, and either get consent from your end users, or provide them with the opportunity to opt-out.

    You must not send any data which allows to personally identify an individual or permanently identify a particular device. The clientID generation code in `Analytics.cs` already includes app install -time generated random data, so it is not a permanent identification.

Note that most of the sent hit data is not processed in real-time.
You can typically see the new data after a 24-hour period or so.

Offline-Cache, Throttling and Internet Access Status

There is “Use Offline Cache” checkbox in the Analytics component (default is enabled).

When offline cache is **disabled**, all hits are sent immediately if network seems to be reachable, or discarded if client is offline or analytics server is not accessible.

When offline cache is **enabled**, network reachability is automatically monitored and proper access to the Google Analytics (GA) service is also separately verified before sending analytics hits. All analytics hits will be saved to a cache file if network status is determined to be non-functional (client is offline, or there is no access to the GA server, or client can only fetch a network login screen).

When client is online and internet access is verified, the cached hits are automatically sent one by one with throttling. Queue time (measured in milliseconds) is automatically added to all cached hits, so that they are attributed to correct time. The cache file is deleted once all hits have been sent. Note that the throttling only applies to hits sent from the offline cache, not hits sent when client is online with verified network access.

NOTE: Measurement Protocol specification warns that if requests are too old, then hits may not be processed. This limitation is set in the Google Analytics service and you cannot change it. The official age limit in the specification is **4 hours**. However, in google-analytics-measurement-protocol group, it has been stated that “docs should actually read 4 hours past midnight of the configured timezone in the profile”.

Applications should be designed to send only moderate amount of events per session (and limit to use of small payloads). Google's servers may throttle incoming hits if you send too many very rapidly. There is 500 hit limit per session (see Limits and Quotas), after which Google Analytics will stop processing hits for that session.

The offline cache file name is set in the `Analytics.cs`, string offlineCacheFileName. Default file name is "GUA-offline-queue.dat". The file is saved to Application.persistentDataPath. Also some offline cache specific metadata entries are kept in PlayerPrefs.

Limits and Quotas

You can verify up to date info about limits and quotas from Google's documentation.

Here's a quick recap, applicable to Normal accounts:

- 10 million hits per month per **property** (tracking ID)
- 200,000 hits per **visitor** per day
- 500 hits per **session**
- Maximum of 20 custom dimension metrics (200 for premium accounts)

Other Documentation

Please extract Assets/Analytics/doxygen_docs.zip and check out the API docs for the GoogleUniversalAnalytics class:

```
docs/html/class_google_universal_analytics.html
```

Also, it doesn't hurt to peek at the source code!

References for Designing Your Analytics

- Remember to make yourself familiar with Google User-ID Policy:
  https://developers.google.com/analytics/devguides/collection/protocol/policy
  (See end of chapter "*Integrating Analytics Code To Your Unity Project*")

- Courses in the Google's Analytics Academy
  https://analyticsacademy.withgoogle.com

- Making Sense of Data course by Google
  https://datasense.withgoogle.com/course

- Gamasutra: "Muddled Mobile Metrics" by Trevor McCalmont
  http://gamasutra.com/blogs/TrevorMcCalmont/20130514/192214/Muddled_Mobile_Metrics.php

If you have more suggestions for the list above, please mail them to contact@strobotnik.com.

Some Hints About How to Find Stuff from Analytics Results

We're assuming you have now collected your initial bunch of data, maybe over a few days and probably consisting of your alpha/beta testers. To begin data reviewing, open your web browser. Go to the start page of Google Analytics and find your newly added app name & click it to go to App Overview analytics page. The App Overview page itself should be pretty self-explanatory -- similarly, try to visit each sub-page to get an overall idea what's there.

To see usage of screens in your application, go to Engagement→Screens. In the table it's easy what screens are used the most and average time users are spending in each screen.

You can find the "first launch" system info from Engagement→Events→Top Events, and select the "SystemInfo_since_v001" event category (if you haven't renamed it). There you can find a table with many statistics. Note that the ones with numerical values show a cumulative value in Event Value, which is not very useful, while the Average Value probably is. You can also click the Event Action to get detailed breakdown of submitted values. The code in `Analytics.cs` granularizes some of the values for slightly easier interpretation, e.g. memory sizes and fill rate. These system info events help you to accumulate device statistics of your users so you know what kinds of devices are actively used to run your application. The idea with the naming of the event category is that once you make a major update your software, you can change the version number in the category to e.g. "v100" (first public release v1.0.0) or "v200" (major update v2.0.0). After pushing an update with this change, all active users will re-submit their system info on next launch. This way you can compare e.g. device type usage percentages to earlier systeminfo category so you can get some idea how many active users have newer devices. See also next chapter, *"Example of Creating a Custom Report"*.

After you have familiarized yourself with the basics, try out using secondary dimensions. For example, go to Users→Demographics→Location. Top of the table, click Secondary Dimension and select Users→Screen Resolution. Now you see what screen resolutions are most used by country.

The Engagement→Engagement Flow screen should be useful for analyzing screens your users go through, and to view how many users "drop-off" in certain screen.

For some advanced level exploration of data, it may be good idea to try out Google Analytics Query Explorer: http://ga-dev-tools.appspot.com/explorer/

Example of Creating a Custom Report

You may have noticed that for example the Audience → Devices and Network → Devices report page doesn't show as fine-grained data you might want in some cases. For example, if you select Operating System as the Primary Dimension, hits from desktop builds may be detected by Google as "(not set)".

However, if you're following the integration example of `Analytics.cs`, that is, your app sends events named like “SystemInfo_since_v001” for each user's first launch, you can also use custom reports to view that data. Those events also contain an operatingSystem field. Here are instructions how to make such a custom report.

1. When you're in the Reporting section of your Analytics profile, select "Customization" from the top bar (next to Reporting).
2. Select "+New Custom Report"
3. Enter Title to General Information: "OS from Events"
4. In Report Content->Metric Groups, select "+ add metric", and pick "Sessions" which is selectable under Visitors.
5. In Report Content->Dimension Drilldowns, select "+ add dimension", and pick "Event Label" which is selectable under Engagement.
6. Select "+add filter" in the Filters, and pick "Event Category" which is selectable under Engagement.
7. Modify the latter dropdown from "Exact" to "Regex".
8. Write "SystemInfo" in the last empty textfield of the newly created filter. Now your filter should have exactly these values: Include, "Event Category", Regex, SystemInfo
9. Select "+add filter" again in the Filters to create another one, and pick "Event Action" which is selectable under Engagement.
10. Keep the latter dropdown as "Exact".
11. Write "operatingSystem" in the last empty textfield of the newly created filter. Now the second filter should have exactly these values: Include, "Event Action", Exact, operatingSystem
12. Select "Save", and you'll be taken to the new report. It's now also added to list of Custom Reports for later access.

Feedback, Feature Suggestions and Bug Reports

Send by email: contact@strobotnik.com. Write “GoogleUniversalAnalytics” to subject.