Unified Analytics SDK

A custom analytics SDK

Abdelrahman Hajjaj

Overview

The unified analytics SDK is a custom unity SDK that uses Game Analytics and Facebook to track down multiple in-game operations and events.

Figure 1 Shows the main SDK configuration editor window.

Figure 1 - SDK Config Editor

Installation

To install the SDK to an existing unity project, you will need to import a custom package.

Importing a custom package is done via the “Assets Menu -> Import package -> Custom Package”.

Configuration

Configuring the SDK was intended to be developer friendly, and it allow easy and direct access to the “Config” editor.

From the “Unified Analytics” Menu, Click view Config (As shown in Figure 2), And the unity editor will automatically select and view the Configuration editor shown in Figure 1.

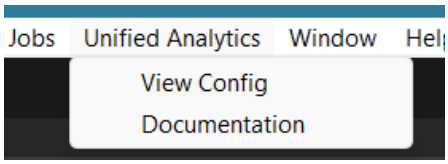

Figure 2 - Unified Analytics Menu

SDK Setting Entries

The first section in the configuration editor, it allows dynamic SDKs controls and handling for the basic setup of the supported SDKs.

The currently supported SDKs are “Game Analytics”, “Facebook SDK” and “Test Dummy SDK”

However, developers can still integrate other new SDKs to be supported and shown in the config editor, we will circle to that later in the SDK API section.

The SDKs Entries section is divided into 2 sections as shown on the right in figure 3 (1 and 2 are setting entries added) while 3 is the controller to add new entries.

The SDK support unlimited number of entries if they have a unique name, and their perspective SDK support those additions.

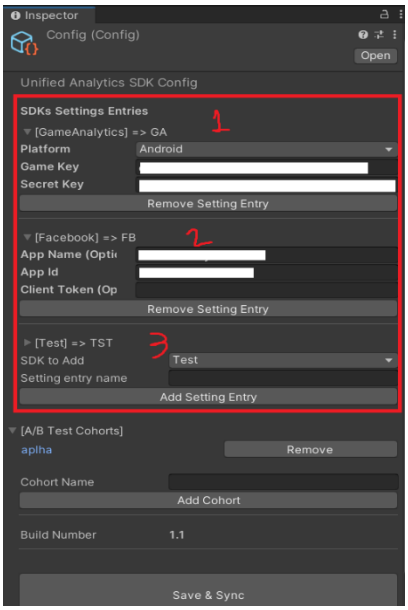

Figure 3 - SDKs Setting Entries

A/B Test Cohorts

The second section of the configuration editor, it allows the addition of Cohorts (Test Groups) mainly used in the game analytics SDK to add any number of A/B test cases.

Each client on the first install will get assigned a random cohort which will persist through sessions.

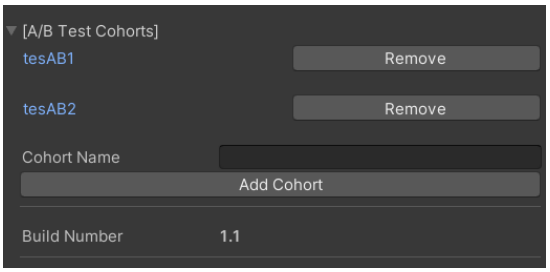

Figure 4 - A/B Test Cohort Editor

The Cohort name and the build number are combined to be sent in the "Game Analytics" events through sessions here is an example of events received by 2 unique devices each one of them is assigned to different cohort.

Figure 5 - Game Analytics Feed Example

SDK Developer API

The "Unified Analytics" SDK public API is intended for developers to control and access features supported by the SDK.

Developers can access and use the SDK by calling out "`UnifiedAnalyticsSDK.UnifiedAnalytics`" which is a singleton controller instance for the SDK.

Usage

A perquisite before initializing and using the SDK, it's required to instantiate a "Game Analytics" tracker prefab into the main scene.
Then using "`UnifiedAnalytics`.Initialize()" will start the initialization process for all SDK operations.

Config

The "`UnifiedAnalytics`" allows access to the "`Config`" instance which is the underlying instance for the Config editor shown in Figure 1, Allowing access to the SDKs entries, Cohort names and build number.

Usage

Developers can access the "`Config`" instance by using the "`UnifiedAnalytics`.Config" which reference a direct "`Config`" instance from the "`UnifiedAnalytics`" controller.

Tracker

It's Control on of the main operations of the "`UnifiedAnalytics`" SDK, which allows sending app events to both "Game Analytics" and "Facebook".

Usage

Developers can access the "`Tracker`" instance by using the "`UnifiedAnalytics`.Tracker" which allows multiple tracking operations using events.

Methods

`SendFacebookAppEven` Parameters: - [string] event name - [Dictionary] parameters	Allows sending custom event using the Facebook SDK. Requires 2 parameters, which are the event name and its parameters
`SendLevelStartEvent` Parameters: - [string] event name	Sends a Game analytics progression event indicating that a level has started. Requires 1 parameter, which is the level/Progression name.
`SendLevelCompletedEvent` Parameters: - [string] Level name - [bool] Player won?	Sends a Game Analytics progression event indicating that a level has been completed. Requires 2 parameters, which are the level/progression name and an indicator whether the player has won in the level.
`SendLevelFailedEvent` Parameters: - [string] Level name - [string] Reason	Sends a level failed progression event, indicating that the level has failed while supplying the reason. (i.e., player cancelled the level or terminated the app). Requires 2 parameters, which are the level/progression name and the failure reason.
`SendProgressionEvent` Parameters: - [Enum] `Progression Status` - [Dictionary] parameters	Sends a custom Game Analytics progression event. Requires 2 parameters, which are the progression status (Game Analytics) and the parameters for the event.