

Unity Flurry SDK (unity-flurry-sdk)

A Unity plugin for Flurry SDK

Flurry Push for messaging and **Flurry Config** for remote configuration are supported by our plugin!

Table of contents

- [Installation](#)
 - [Android](#)
 - [iOS](#)
- [Example](#)
- [API Reference](#)
- [Support](#)
- [License](#)

Installation

The Flurry SDK Unity plugin is available via [the Unity Asset Store](#) and [Github](#).

- Install from [the Unity Asset Store](#)
 1. Open [the Unity Asset Store](#) in your browser.

2. Click on the "Open in Unity" button, then open your Unity project.
 3. Download and import "Flurry SDK Plugin" in your Package Manager.
- Download and install from [Github](#)
 1. Download the Flurry Unity package from [flurry-sdk-6.1.0.unitypackage](#).
 - If you are using Apple Xcode < 12, please use releases [flurry-sdk-3.3.0.unitypackage](#), or [flurry-sdk-3.3.0-push.unitypackage](#) if you want to use Flurry Push.
 2. Open your project in Unity Editor, choose menu **Assets > Import Package > Custom Package...** to bring up the File chooser, and select the package downloaded.
 - Add Flurry code

```
using FlurrySDK;
```

Android

Note: `FlurryUnityApplication.java` is now bundled in the `aar` format. Please manually remove `FlurryUnityApplication.java` imported by the previous release from the `Assets/Plugins/Android` folder.

- To improve analytics identities, please see [Manual Flurry Android SDK Integration](#) for adding Google Play Services library in your app by including `play-services-ads-identifier` libraries.
- **Flurry Push**

In order to use [Flurry Push](#) for [Android](#), please follow the additional steps below:

1. Follow [Set up a Firebase Cloud Messaging client app with Unity](#). Complete to the 5th step for importing Firebase SDK. There should be a file `google-services.json` in your project's `Android` folder now. You do not need to provide any setup codes here.
2. Please rename the following Android manifest template file

`Assets/Plugins/Android/AndroidManifest_Flurry-template.xml` that comes with the Flurry SDK plugin to `AndroidManifest.xml`, and merge the contents with yours if needed. You need to replace `AndroidManifest.xml` generated by Firebase with Flurry's.

```
<application
...
<!-- Flurry Messaging services; do not modify -->
<service android:name="com.flurry.android.marketing.messaging.FCM.FlurryMessageListenerService"
  android:exported="false">
  <intent-filter>
    <action android:name="com.google.firebase.MESSAGING_EVENT" />
  </intent-filter>
</service>

<receiver
  android:name="com.flurry.android.marketing.messaging.notification.NotificationCancelledReceiver"
  android:enabled="true"
  android:exported="false">
</receiver>

<receiver
  android:name="com.flurry.android.marketing.messaging.notification.NotificationClickedReceiver"
  android:enabled="true"
  android:exported="false">
</receiver>
```

3. If you want to customize Flurry Push notification, please configure an Android entry point Application and update the metadata section in your `AndroidManifest.xml`. Example can be found at [FlurryUnityApplication.java](#).

```

<application
  android:name="com.flurry.android.FlurryUnityApplication"
  android:label="@string/app_name"
  android:icon="@drawable/app_icon">

  <!-- Flurry Agent settings; please update -->
  <meta-data android:name="flurry_apikey" android:value="FLURRY_ANDROID_API_KEY" />
  <meta-data android:name="flurry_with_crash_reporting" android:value="true" />
  <meta-data android:name="flurry_with_continue_session_millis" android:value="10000L" />
  <meta-data android:name="flurry_with_include_background_sessions_in_metrics" android:value="true" />
  <meta-data android:name="flurry_with_log_enabled" android:value="true" />
  <meta-data android:name="flurry_with_log_level" android:value="2" />
  <meta-data android:name="flurry_with_messaging" android:value="true" />

```

4. Set up "Android Authorization" in Flurry [Push Authorization](#).

- Flurry plugin released `aar` libraries in the package. If your apps change the default searching path, please remember to include the `aar` type.

```
implementation fileTree(dir: 'libs', include: ['*.jar', '*.aar'])
```

iOS

For further details on configuring xcode for push notifications see here: [Flurry Push for Unity iOS](#).

There are some minor differences between the Android and iOS plugin:

- iOS does not make use of the messaging listeners in C-sharp. Delegate methods `didReceiveMessage/didReceiveActionWithIdentifier` in `FlurryUnityPlug.mm` may be optionally modified to customize app behavior.
- iOS does not have an equivalent method for Android's `GetReleaseVersion` method.

- iOS does not yet have an equivalent method for Android's LogPayment method, however if SetIAPReportingEnabled is set to true Flurry will automatically track in app purchases.

Example

- `Example.cs`

```
using System.Collections.Generic;
using UnityEngine;

using FlurrySDK;

public class FlurryStart : MonoBehaviour
{
    #if UNITY_ANDROID
        private readonly string FLURRY_API_KEY = FLURRY_ANDROID_API_KEY;
    #elif UNITY_IPHONE
        private readonly string FLURRY_API_KEY = FLURRY_IOS_API_KEY;
    #else
        private readonly string FLURRY_API_KEY = null;
    #endif

    void Start()
    {
        // Note: When enabling Messaging, Flurry Android should be initialized by using AndroidManife
        // Initialize Flurry once.
        new Flurry.Builder()
            .WithCrashReporting(true)
            .WithLogEnabled(true)
            .WithLogLevel(Flurry.LogLevel.DEBUG)
            .WithReportLocation(true)
```

```
.WithMessaging(true, new MyMessagingListener())
.Build(FLURRY_API_KEY);

// Example to get Flurry versions.
Debug.Log("AgentVersion: " + Flurry.GetAgentVersion());
Debug.Log("ReleaseVersion: " + Flurry.GetReleaseVersion());

// Set Flurry preferences.
Flurry.SetLogEnabled(true);
Flurry.SetLogLevel(Flurry.LogLevel.VERBOSE);

// Set user preferences.
Flurry.SetAge(36);
Flurry.SetGender(Flurry.Gender.Female);
Flurry.SetReportLocation(true);

// Set user properties.
Flurry.UserProperties.Set(Flurry.UserProperties.PROPERTY_REGISTERED_USER, "True");

// Log Flurry events.
Flurry.EventRecordStatus status = Flurry.LogEvent("Unity Event");
Debug.Log("Log Unity Event status: " + status);

// Log Flurry timed events with parameters.
IDictionary<string, string> parameters = new Dictionary<string, string>();
parameters.Add("Author", "Flurry");
parameters.Add("Status", "Registered");
status = Flurry.LogEvent("Unity Event Params Timed", parameters, true);
Debug.Log("Log Unity Event with parameters timed status: " + status);
...
Flurry.EndTimedEvent("Unity Event Params Timed");

// Log Flurry standard events.
```

```

        status = Flurry.LogEvent(Flurry.Event.APP_ACTIVATED);
        Debug.Log("Log Unity Standard Event status: " + status);

        Flurry.EventParams stdParams = new Flurry.EventParams()
            .PutDouble (Flurry.EventParam.TOTAL_AMOUNT, 34.99)
            .PutBoolean(Flurry.EventParam.SUCCESS, true)
            .PutString (Flurry.EventParam.ITEM_NAME, "book 1")
            .PutString ("note", "This is an awesome book to purchase !!!");
        status = Flurry.LogEvent(Flurry.Event.PURCHASED, stdParams);
        Debug.Log("Log Unity Standard Event with parameters status: " + status);
    }
}

```

- **Config.cs**

```

// Register Config listener
Flurry.Config.RegisterListener(new MyConfigListener());
Flurry.Config.Fetch();

public class MyConfigListener : Flurry.IConfigListener
{
    public void OnFetchSuccess()
    {
        Debug.Log("Config Fetch Completed with state: Success");
        Flurry.Config.Activate();
    }

    public void OnFetchNoChange()
    {
        Debug.Log("Config Fetch Completed with state: No Change");
        complete();
    }
}

```

```

public void OnFetchError(bool isRetrying)
{
    Debug.Log("Config Fetch Completed with state: Fail - " + (isRetrying ? "Retrying" : "End"));
    complete();
}

public void OnActivateComplete(bool isCache)
{
    Debug.Log("Config Fetch Completed with state: Activate Completed - " + (isCache ? "Cached" :
    complete());
}

private void complete()
{
    string welcome_message = Flurry.Config.GetString("welcome_message", "Welcome!");
    Debug.Log("Get Config Welcome message: " + welcome_message);
}
}

```

- **Messaging.cs**

```

// Set Messaging listener
new Flurry.Builder()
    .WithMessaging(true, new MyMessagingListener())
    ...

public class MyMessagingListener : Flurry.IMessagingListener
{
    // If you would like to handle the notification yourself, return true to notify Flurry
    // you've handled it, and Flurry will not show the notification.
    public bool OnNotificationReceived(Flurry.FlurryMessage message)
    {
        Debug.Log("Flurry Messaging Notification Received: " + message.Title);
    }
}

```



```

        return false;
    }

    // If you would like to handle the notification yourself, return true to notify Flurry
    // you've handled it, and Flurry will not launch the app or "click_action" activity.
    public bool OnNotificationClicked(Flurry.FlurryMessage message)
    {
        Debug.Log("Flurry Messaging Notification Clicked: " + message.Title);
        return false;
    }

    public void OnNotificationCancelled(Flurry.FlurryMessage message)
    {
        Debug.Log("Flurry Messaging Notification Cancelled: " + message.Title);
    }

    public void OnTokenRefresh(string token)
    {
        Debug.Log("Flurry Messaging Token Refresh: " + token);
    }

    public void OnNonFlurryNotificationReceived(IDisposable nonFlurryMessage)
    {
        Debug.Log("Flurry Messaging Non-Flurry Notification.");
    }
}

```

- Publisher.cs

```
// Register Publisher Segmentation listener
Flurry.PublisherSegmentation.RegisterListener(new MyPublisherSegmentationListener());
Flurry.PublisherSegmentation.Fetch();

public class MyPublisherSegmentationListener : Flurry.IPublisherSegmentationListener
{
    public void OnFetched(IDictionary<string, string> data)
    {
        string segments;
        data.TryGetValue("segments", out segments);
        Debug.Log("Flurry Publisher Segmentation Fetched: " + segments);
    }
}
```

API Reference

See [Android-\(FlurryAgent\)](#) / [iOS-\(Flurry\)](#) for the Flurry references.

- **Methods in Flurry.Builder to initialize Flurry Agent**

```

Builder WithAppVersion(string appVersion); // iOS only. For Android, please use Flurry.setVersionName
Builder WithContinueSessionMillis(long sessionMillis);
Builder WithCrashReporting(bool crashReporting);
Builder WithGppConsent(string gppString, ISet<int> gppSectionIds); // Android only
Builder WithDataSaleOptOut(bool isOptOut);
Builder WithIncludeBackgroundSessionsInMetrics(bool includeBackgroundSessionsInMetrics);
Builder WithLogEnabled(bool enableLog);
Builder WithLogLevel(Flurry.LogLevel logLevel); // LogLevel = { VERBOSE, DEBUG, INFO, WARN, ERROR, ASSERT
Builder WithReportLocation(bool reportLocation); // Android only
Builder WithMessaging(bool enableMessaging, IMessagingListener messagingListener);
Builder WithPerformanceMetrics(Flurry.Performance performanceMetrics); // Performance = { NONE, COLD, WARM, HOT
Builder WithSslPinningEnabled(bool sslPinningEnabled); // Android only

void Build(string apiKey);

```

- **Methods to set Flurry preferences**

```

void SetContinueSessionMillis(long sessionMillis);
void SetCrashReporting(bool crashReporting);
void SetIncludeBackgroundSessionsInMetrics(bool includeBackgroundSessionsInMetrics);
void SetLogEnabled(bool enableLog);
void SetLogLevel(Flurry.LogLevel logLevel); // LogLevel = { VERBOSE, DEBUG, INFO, WARN, ERROR, ASSERT
void SetSslPinningEnabled(bool sslPinningEnabled); // Android only

```

- **Methods to set user preferences**

```

void SetAge(int age);
void SetGender(Flurry.Gender gender); // Gender = { Male, Female }
void SetReportLocation(bool reportLocation);
void SetSessionOrigin(string originName, string deepLink);
void SetUserId(string userId);
void SetVersionName(string versionName); // Android only. For iOS, please use Flurry.Builder.WithApp

void AddOrigin(string originName, string originVersion);
void AddOrigin(string originName, string originVersion, IDictionary<string, string> originParameters);
void AddSessionProperty(string name, string value);

```

- **Methods to set privacy preferences**

```

bool SetGppConsent(string gppString, ISet<int> gppSectionIds); // Android only
void SetDataSaleOptOut(bool isOptOut);
void DeleteData();
void OpenPrivacyDashboard();

```

- **Methods in Flurry.UserProperties to set user properties**

```

// Standard User Properties: Flurry.UserProperties = {
//     PROPERTY_CURRENCY_PREFERENCE, PROPERTY_PURCHASER, PROPERTY_REGISTERED_USER, PROPERTY_SUBSCRIBER
void Set(string propertyName, string propertyValue);
void Set(string propertyName, List<string> propertyValues);
void Add(string propertyName, string propertyValue);
void Add(string propertyName, List<string> propertyValues);
void Remove(string propertyName);
void Remove(string propertyName, string propertyValue);
void Remove(string propertyName, List<string> propertyValues);
void Flag(string propertyName);

```

- **Methods to get Flurry versions**

```
int GetAgentVersion();  
string GetReleaseVersion();  
string GetSessionId();
```

- **Methods to log Flurry events**

```

enum EventRecordStatus {
    FlurryEventFailed,
    FlurryEventRecorded,
    FlurryEventUniqueCountExceeded,
    FlurryEventParamsCountExceeded,
    FlurryEventLogCountExceeded,
    FlurryEventLoggingDelayed,
    FlurryEventAnalyticsDisabled,
    FlurryEventParametersMismatched
}

EventRecordStatus LogEvent(string eventId);
EventRecordStatus LogEvent(string eventId, IDictionary<string, string> parameters);
EventRecordStatus LogEvent(string eventId, bool timed);
EventRecordStatus LogEvent(string eventId, IDictionary<string, string> parameters, bool timed);

void EndTimedEvent(string eventId);
void EndTimedEvent(string eventId, IDictionary<string, string> parameters);

EventRecordStatus LogEvent(Flurry.Event eventId, Flurry.EventParams parameters);

void OnPageView(); // Deprecated, API removed, no longer supported by Flurry.

void OnError(string errorId, string message, string errorClass);
void OnError(string errorId, string message, string errorClass, IDictionary<string, string> parameters);

void LogBreadcrumb(string crashBreadcrumb);

EventRecordStatus LogPayment(string productName, string productId, int quantity, double price,
                             string currency, string transactionId, IDictionary<string, string> parameters);

```

- **Methods to set Flurry.EventParams**

```

EventParams EventParams();
EventParams EventParams(EventParams paramsSource);
IDictionary<object, string> GetParams();
EventParams Clear();
EventParams Remove(EventParamBase param);
EventParams Remove(string key);
EventParams PutAll(EventParams paramsSource);
EventParams PutString(StringEventParam param, string value);
EventParams PutString(string key, string value);
EventParams PutInteger(IntegerEventParam param, int value);
EventParams PutInteger(string key, int value);
EventParams PutLong(IntegerEventParam param, long value);
EventParams PutLong(string key, long value);
EventParams PutDouble(DoubleEventParam param, double value);
EventParams PutDouble(string key, double value);
EventParams PutBoolean(BooleanEventParam param, bool value);
EventParams PutBoolean(string key, bool value);

```

- **Methods to enable IAP reporting (iOS)**

```
void SetIAPReportingEnabled(bool enableIAP);
```

- **Methods to set the iOS conversion value sent to Apple through SKAdNetwork (iOS)**

```

void UpdateConversionValue(int conversionValue)
void UpdateConversionValueWithEvent(Flurry.SKAdNetworkEvent flurryEvent); // SKAdNetworkEvent = { No

```

- **Methods in Flurry.Performance for Flurry Performance Metrics**

```
void StartResourceLogger();  
void LogResourceLogger(string id);  
void ReportFullyDrawn();
```

- **Methods in Flurry.Config for Flurry Config**

```
void Fetch();  
void Activate();  
void RegisterListener (IConfigListener configListener);  
void UnregisterListener(IConfigListener configListener);  
string GetString(string key, string defaultValue);  
  
interface IConfigListener  
{  
    void OnFetchSuccess();  
    void OnFetchNoChange();  
    void OnFetchError(bool isRetrying);  
    void OnActivateComplete(bool isCache);  
}
```

- **Methods for Messaging (Flurry Push)**


```

interface IMessagingListener
{
    bool OnNotificationReceived(FlurryMessage message);
    bool OnNotificationClicked(FlurryMessage message);
    void OnNotificationCancelled(FlurryMessage message);
    void OnTokenRefresh(string token);
    void OnNonFlurryNotificationReceived(IDisposable nonFlurryMessage);
}

class FlurryMessage
{
    string Title;
    string Body;
    string ClickAction;
    IDictionary<string, string> Data;
}

```

- **Methods in Flurry.PublisherSegmentation for Flurry Publisher Segmentation**

```

void Fetch();
void RegisterListener (IPublisherSegmentationListener publisherSegmentationListener);
void UnregisterListener(IPublisherSegmentationListener publisherSegmentationListener);
IDictionary<string, string> GetData();

interface IPublisherSegmentationListener
{
    void OnFetched(IDictionary<string, string> data);
}

```

Support

- [Flurry Developer Support Site](#)

License

Copyright 2022 Yahoo Inc.

This project is licensed under the terms of the [Apache 2.0](#) open source license. Please refer to [LICENSE](#) for the full terms.