Happyfiit Android SDK Documentation

August 2015

Intializing the Engine

Every time you app is launched you should call the `initialize()` function of the `Happyfiit` class. The best place to do this is in your app `onCreate()` method.

```
public class MyApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();

        Map<String, String> initParams = new HashMap<>();
        initParams.put(ApplicationConfig.DEVELOPER_KEY, "your key");
        initParams.put(ApplicationConfig.DEVELOPER_SECRET, "your secret");
        initParams.put(ApplicationConfig.USER_EMAIL, "user's email address");

        Happyfiit.initialize(getApplicationContext(), initParams);
    }
}
```

`Initialize` function takes up a minimum of 2 parameters. The first one is the application context. The second is a map containing initialization properties in the form of key value pairs. These should include the app key and secret sent to you while registering to the Happyfiit platform. If the user's email address is available to the application, it is advisable to include this in the initialization parameters too so that Happyfiit can use this information for identifying a user and keep track of his/her progress across multiple devices and Android application installations. This is an optional field though and when not provided Happyfiit will generate and assign a random user id that will only be available for the lifetime of the application installation (cached in

application preferences). After a new application installation, Happyfiit will not be able to identify the user and lose track of his past statistics.

All available parameter names can be looked up and retrieved from the `ApplicationConfig` class.

You can contact info@happyfiit.com for initiating the registration process.

First Run

User Consent

In case it is the first time the Happyfiit SDK is used on the device and before any data is sent to the Happyfiit platform, a user consent dialog will be shown to the user notifying her about the presence of the SDK, how the Happyfiit Rewards Platform works and where to go should she has more queries. At the bottom of the popup dialog the user will have the option to either opt-in or opt-out of the platform. In case of the latter, any calls made to the Happyfiit API apart from the `initialize()` function will throw a `PlatformDeactivatedException`.

In order to avoid presenting the user consent dialog to the user right after finishing a workout, it is recommended to manually invoke the `start()` method of the Happyfiit API that should take care of this before a workout activity is started. Best place to call this method would be inside the workout activity's `onCreate()` method as shown in the code snippet below.

```
public class WorkoutActivity extends Activity {

    private Happyfiit happyfiit;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        try {
            happyfiit = Happyfiit.getInstance();
        } catch (HappyfiitInitializationException e) {
            //handle exception
        }

        happyfiit.start(new HappyfiitCallbackHandler(this) {
            @Override
            public void onFailure(Exception ex) {

            }
```

```
        });
    }
}
```

You need to obtain a reference to the Happyfiit singleton object before you can invoke any methods of the SDK. This can be achieved by calling the `getInstance()` method of the Happyfiit class, which should raise a `HappyfiitInitializationException` in case the SDK was not initialized properly. The `start()` method can then be invoked to present the user consent dialog in case of first time use of the API. Since the method is executed asynchronously, an instance of a `HappyfiitCallbackHandler` needs to be passed in as argument to notify the calling code about execution errors if any.

Historical Workouts

It is recommended that you provide Happyfiit with the workouts of the user so that our algorithm can more accurately track the user's course of fitness. This is a one-off procedure that takes place on the first time Happyfiit is launched on the device. The process is facilitated by the `getWorkoutHistory()` callback that needs to be provided using an overloaded version of the `initialize` function. Detailed information and example code available in the sections below.

Pushing Workout data

Workout and Opportunity definitions

A workout includes information that has been collected by the fitness app during a user's physical activity and may be uploaded to the Happyfiit Platform for evaluating whether this should be rewarded or not. Information about a workout includes its type (e.g. running, push-ups, etc.) and a collection of associated workout statistics (e.g. duration, distance, etc.) with their accompanying scores.

A workout may be created using a Builder as shown below:

```
Workout workout1 = new Workout.Builder()
                .type("running")
                .addWorkoutStatistic("distance", "meter", "100")
                .addWorkoutStatistic("duration", "sec", "10")
                .build();
```

Before uploading a workout to the Platform, this needs to be encapsulated inside an Opportunity object.

```
Opportunity op = new Opportunity.Builder()
                .addWorkout(workout1)
                .build();
```

An `opportunity` can include one or multiple workouts (e.g. case of uploading user's workout history), as well as additional information that may be available on the device and can be shared with the Happyfiit platform in order to optimize its rewarding algorithm (e.g. performance aggregates, user personal bests, etc.).

```
Opportunity op = new Opportunity.Builder()
                .addWorkout(workout1)
                .addParameter("mostCaloriesBurnt", 665)
                 .addParameter("farthestDistanceKM", 6.09)
                .build();
```

Pushing a workout

Upon creation of the Opportunity object this can be pushed to the server by invoking the `pushOpportunity` function of the Happyfiit API as shown below:

```
happyfiit.pushOpportunity(op, new HappyfiitCallbackHandler(activityContext) {

            @Override
            public void onFailure(Exception ex) {
                //handle exception
            }
        });
```

This is executed asynchronously on the device, meaning the call will not block until a response is retrieved from the Happyfiit server. The function takes up two arguments, the opportunity containing the workout data and an instance of a `HappyfiitCallbackHandler` used for notifying the calling code about the execution result when this is available. The callback handler needs to be provided with the activity context, so that the rewards popup can be shown in case of an achievement.

Optionally the user can override the `onAchievement()` function of the `HappyfiitCallbackHandler` class in order to customize the way a reward will be shown to the user. It is recommended to use this function for creating native application notifications (e.g. popup, present icon etc.) prior to showing the Happyfiit Reward popup. This can help the application developer maintain control over the user experience by customizing the look and feel of the notification. Note however that when overriding this function you need to make sure that at some point you call the `showReward()` function for presenting the Happyfiit popup to the user in order to redeem the reward. Following code snippet illustrates this process.

```
happyfiit.pushOpportunity(op, new HappyfiitCallbackHandler(this) {

        @Override
        public void onAchievement(Reward reward) {
                //1. Create a native to the application notification (optional step)

                //2. Show Happyfiit reward popup so that user can redeem it (required step if
        onAchievement method is overriden)
```

```
                happyfiit.showReward(activityContext, reward);
        }

        @Override
        public void onFailure(Exception ex) {
                //handle exception
        }
});
```

Pushing Historical Workouts

User workouts that have been collected by the mobile application in the past and have not been uploaded to the Happyfiit Platform but may do so in order to provide the latter with more data about the user's course of fitness is known as the user's workout history. This can optionally be provided to Happyfiit by using the overloaded version of the `initialize` function as shown below:

```
public class MyApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();

        Map<String, String> initParams = new HashMap<>();
        initParams.put(ApplicationConfig.DEVELOPER_KEY, "your key");
        initParams.put(ApplicationConfig.DEVELOPER_SECRET, "your secret");
        initParams.put(ApplicationConfig.USER_EMAIL, "user's email address");

        Happyfiit.initialize(getApplicationContext(), initParams, firstRunHandler());
    }

    private FirstRunHandler firstRunHandler() {
        return new FirstRunHandler() {
            @Override
            public Opportunity getWorkoutHistory() {

                //create user's workout history

                Workout workout1 = new Workout.Builder()
                .type("running")
                .addWorkoutStatistic("distance", "meter", "100")
                .addWorkoutStatistic("duration", "sec", "10")
                .time(new LocalDateTime(2015, 3, 25, 18, 0, 37))
                .build();

                Workout workout2 = new Workout.Builder()
                .type("running")
                .addWorkoutStatistic("distance", "meter", "100")
                .addWorkoutStatistic("duration", "sec", "9")
                .time(new LocalDateTime(2015, 3, 26, 18, 1, 20))
```

```
                .build();

                List<Workout> workouts = new ArrayList<>(Arrays.asList(workout1, workout2));

                Opportunity op = new Opportunity.Builder()
                        .history(true)
                        .addWorkoutList(workouts)
                        .addParameter("pb", 10)
                        .build();

                return op;
            }
        };
    }
}
```

This version of `initialize` accepts an additional argument of type `FirstRunHandler` which will be used by the SDK only if it is the first time `Happyfiit` is running on the device. The user's workout history can then be provided by creating an instance of this class and implementing its `getWorkoutHistory()` function to prepare and return a list of past workouts, contained inside an `Opportunity` object. For reasons of separating the opportunities that should be checked for achievements and can trigger a reward from the ones that contain historical data and should only be used for archiving on the server, the history flag of the Opportunity object must be set to true in this case.

SDK Installation Instructions

The Happyfiit SDK is packaged as an Android Library file (aar). This needs to be included in the application dependencies by referencing it in the gradle build script as shown below.

```
dependencies {
…..
    compile ('com.happyfiit:android-sdk:1.0.0@aar') {
        transitive=true
    }
 }
```

The "`transitive=true`" attribute is needed so that 3rd party libraries that the SDK depends on are also included in the generated apk.

You can get the latest version of the SDK by browsing the Happyfiit repository in Github. For this you need to configure the following entries under the build script's "repositories" section:

```
repositories {
    ...
```

```
    maven { url "https://raw.githubusercontent.com/happyfiit/maven-repo/master/releases" }
    maven { url "https://raw.githubusercontent.com/happyfiit/maven-repo/master/snapshots" }
}
```

Please contact the Happyfiit team for any other information you may require concerning the installation process.

Sample Application

A sample android application that demonstrates the use of the Happyfiit SDK for managing user workouts and displaying rewards is available at following URL: https://github.com/happyfiit/sample-app