POKKT SDK Integration Guide (v 6.0)

Native(Android)

Overview

Thank you for choosing **Pokkt SDK** for **Android**. This document contains all the information required to set up the SDK with your project. We also support mediation for various third party networks. To know the supported third party networks and their integration process go to mediation section.
Before implementing plugins it is mandatory to go through project configuration and implementation steps, as these sections contain mandatory steps for basic SDK integration and are followed by every plugin.

You can download our SDK from pokkt.com.

In the package downloaded above you will find:

1. Docs:
   Contains documentations for step wise step integration for SDK.
2. PokktSDK_v6.0.jar
   Pokkt SDK in *jar* format.
3. PokktSDK_v6.0.aar
   Pokkt SDK in *aar* format.
4. PokktAds Demo
   Source code for *PokktAds Demo*(Sample app) which showcase implementation of Pokkt SDK through code for better understanding.
5. PokktAds Demo.apk
   Application package of PoktkAds Demo, so that you can directly install this apk on your device and have a look how our SDK works instead of compiling the source code.
6. Dependency jars:
   moat.jar and *comscore.jar* both teese are required by Pokkt SDK and should be added to project

minSdkVersion supported is **11**.

ScreenName: This one parameter is accepted by almost all API's of Pokkt SDK. This controls the placement of ads and can be created on Pokkt Dashboard.

We will be referencing **PokktAds Demo** app provided with SDK during the course of explanation in this document. We suggest you go through the sample app for better understanding.

Project Configuration

Dependencies

- Add *PokktSDK_v6.0.jar* or *PokktSDK_v6.0.aar* to your project.
- Add *moat.jar* and *comscore.jar* to your project.
- We expect **Google play services** integrated in project, although it;s optional but we recommend you to integrate it, as it's required to fetch **AdvertisingID** for device,which is useful to deliver targeted advertising to Android users.

Manifest

Permissions Declarations

Add the following permissions to your project manifest

1. Mandatory permissions.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

- android.permission.INTERNET = Required for SDK communication with server.

- android.permission.READ_PHONE_STATE = Required for creating unique identifier for you application based on the unique id of the device like IMEI.

2. Optional permissions.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_CALENDAR" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.SEND_SMS" />
```

- android.permission.ACCESS_NETWORK_STATE = Required to detect changes in network, like if WIFI is available or not.
- android.permission.ACCESS_WIFI_STATE = Required to detect changes in network, like if WIFI is available or not.
- android.permission.CHANGE_WIFI_STATE = Required to detect changes in network, like if WIFI is available or not.
- android.permission.WAKE_LOCK = Required to prevent device from going into the sleep mode during video play.
- android.permission.WRITE_EXTERNAL_STORAGE = Required to store media files related to ads in external SD card, if not provided we will use app cache folder to store media files, which will result in unnecessary increase in application's size. It is recommended to ask for this permission as low end devices generally have less internally memory available.
- android.permission.WRITE_CALENDAR = Some Ads create events in calendar.
- android.permission.ACCESS_COARSE_LOCATION" = Some Ads show content based on user's location.
- android.permission.ACCESS_FINE_LOCATION = Some Ads show content based on user's location
- android.permission.CALL_PHONE = Some Ads are interactive and they provide you a way to call directly from the content.
- android.permission.SEND_SMS = Some Ads are interactive and they provide you a way to send message.

Activity Declaration

Add the following activity in your AndroidManifest for Pokkt SDK integration.

```
<activity
        android:name="com.pokkt.sdk.userinterface.presenter.activity.PokktAdActivity"
        android:configChanges="keyboard|keyboardHidden|navigation|
                               orientation|screenLayout|uiMode|
                               screenSize|smallestScreenSize"
        android:hardwareAccelerated="true"
        android:label=”Pokkt”
        android:screenOrientation="landscape"
        android:windowSoftInputMode="stateAlwaysHidden|adjustUnspecified" />
```

You can change the **android:screenOrientation="landscape"** to of your choice, the way you want to display the ads.

Service Declaration

Add the following service in your AndroidManifest for receiving InApp notifications. How to set up InApp notifications.

```
<service
        android:name="com.pokkt.sdk.notification.NotificationService"
        android:exported="false"
        android:label="PokktNotificationService" />
```

Implementation Steps

SDK Configuration

1. Set **Application Id** and **Security key** in Pokkt SDK. You can get it from Pokkt dashboard from your account. These are unique per app registered.

```
PokktAds.setPokktConfig("<Pokkt Application ID>", "<Pokkt Security Key>");
```

2. If you are using server to server integration with Pokkt, you can also set **Third Party UserId** in PokktAds.

```
PokktAds.setThirdPartyUserId("<Third party user Id>");
```

3. When your application is under development and if you want to see Pokkt logs and other informatory messages, you can enable it by setting **shouldDebug** to **true**. Make sure to disable debugging before release.

```
PokktAds.Debugging.shouldDebug(<true>);
```

Ad Types

Video

- Video ad can be rewarded or non-rewarded. You can either cache the ad in advance or directly call show for it.
- We suggest you to cache the ad in advance so as to give seamless play behaviour, In other case it will stream the video which may lead to unnecessary buffering delays depending on the network connection.

Rewarded

1. To cache rewarded ad call:

```
PokktAds.VideoAd.cacheRewarded("<ScreenName>");
```

2. To show rewarded ad call:

```
PokktAds.VideoAd.showRewarded("<ScreenName>");
```

Non Rewarded

1. To cache non-rewarded ad call:

```
PokktAds.VideoAd.cacheNonRewarded("<ScreeName>");
```

2. To show non-rewarded ad call:

```
PokktAds.VideoAd.showNonRewarded("<ScreeName>");
```

You can check if ad is available or not before making *cache* or *show* request.

```
PokktAds.VideoAd.checkAdAvailability("<screen name>", <true / false>);
```

Interstitial

Rewarded

1. To cache rewarded ad call:

```
PokktAds.Interstitial.cacheRewarded("<ScreenName>");
```

2. To show rewarded ad call:

```
PokktAds.Interstitial.showRewarded("<ScreenName>");
```

Non Rewarded

1. To cache non-rewarded ad call:

```
PokktAds.Interstitial.cacheNonRewarded("<ScreeName>");
```

2. To show non-rewarded ad call:

```
PokktAds.Interstitial.showNonRewarded("<ScreenName>");
```

You can check if ad is available or not before making *cache* or *show* request.

```
PokktAds.Interstitial.checkAdAvailability("<screen name>", <true / false>);
```

Banner

- Add **PokktBannerView** to your layout, we use it as placeholder to populate banner ad into it.

```
<com.pokkt.sdk.banners.PokktBannerView
        android:id="@+id/pokkt_banner_view_top"
        android:layout_width="320dp"
        android:layout_height="50dp"
        android:layout_centerHorizontal="true"/>
```

- Load banner

```
PokktAds.Banner.loadBanner(“<screenName>”, <pokktBannerView>);
```

- You can remove Banner using:

```
PokktAds.Banner.destroyContainer(<pokktBannerView>);
```

- You can also set banner to auto-refresh using:

```
PokktAds.Banner.ShouldAutoRefresh(<true/false>)
```

Ad Delegates

Ad delegates are optional, but we suggest to implement them as it will help you to keep track of the status of your ad request.

Video

```
PokktAds.VideoAd.setDelegate(new PokktAds.VideoAd.VideoAdDelegate() {

   @Override
   public void videoAdCachingCompleted(String screenName, boolean isRewarded, double
reward) {

   }

   @Override
   public void videoAdCachingFailed(String screenName, boolean isRewarded, String
errorMessage) {

   }

   @Override
   public void videoAdDisplayed(String screenName, boolean isRewarded) {

   }

   @Override
   public void videoAdFailedToShow(String screenName, boolean isRewarded, String
errorMessage) {

   }

   @Override
   public void videoAdClosed(String screenName, boolean isRewarded) {

   }

   @Override
   public void videoAdSkipped(String screenName, boolean isRewarded) {

   }

   @Override
```

```
  public void videoAdCompleted(String screenName, boolean isRewarded) {

  }

  @Override
  public void videoAdGratified(String screenName, boolean isRewarded, double reward) {

  }

  @Override
  public void videoAdAvailabilityStatus(String screenName, boolean isRewarded, boolean
availability) {

  }
});
```

Interstitial

```
PokktAds.Interstitial.setDelegate(new PokktAds.Interstitial.InterstitialDelegate() {

  @Override
  public void interstitialCachingCompleted(String screenName, boolean isRewarded, double
reward) {

  }

  @Override
  public void interstitialCachingFailed(String screenName, boolean isRewarded, String
errorMessage) {

  }

  @Override
  public void interstitialDisplayed(String screenName, boolean isRewarded) {

  }

  @Override
  public void interstitialFailedToShow(String screenName, boolean isRewarded, String
errorMessage) {

  }
```

```
  @Override
  public void interstitialClosed(String screenName, boolean isRewarded) {

  }

  @Override
  public void interstitialSkipped(String screenName, boolean isRewarded) {

  }

  @Override
  public void interstitialCompleted(String screenName, boolean isRewarded) {

  }

  @Override
  public void interstitialGratified(String screenName, boolean isRewarded, double reward) {

  }

  @Override
  public void interstitialAvailabilityStatus(String screenName, boolean isRewarded, boolean
availability) {

  }
});
```

Banner

```
PokktAds.Banner.setDelegate(new PokktAds.Banner.BannerAdDelegate() {

  @Override
  public void bannerLoaded(String screenName) {

  }

  @Override
  public void bannerLoadFailed(String screenName, String errorMessage) {

  }
});
```

Pokkt ad player configuration

Pokkt Ad player works the way App is configured at Pokkt dashboard, but we provide a way to override those settings using **PokktAdPlayerViewConfig**.

Application should prefer configuration provided through code by developer or what's configured for the app in dashboard, can be controlled any time through the dashboard itself. If you want to make changes to this configuration after your app distribution, you can contact **Pokkt Team** to do the same for your app through admin console.

```
PokktAdPlayerViewConfig adPlayerViewConfig = new PokktAdPlayerViewConfig ();
 // set properties values to adPlayerViewConfig
PokktAds.setAdPlayerViewConfig(adPlayerViewConfig );
```

Various setters for the properties that can be managed through this are:

1. **Back button**
   Defines if user is allowed to close the Advertisement by clicking on back button or not.
   Setter Name **:** setBackButtonDisabled(boolean backButtonDisabled)
   Values**:**
   True = Back button is disabled and user cannot close the Ad.
   False = Back button is not disabled and user can close the Ad.

2. **Default skip time**
   Defines the time after which user can skip the Ad.
   Setter name**:** setDefaultSkipTime(int defaultSkipTime)
   Values**:**
   Any Integer value.
   Default value is 10 seconds .
3. **Should allow skip**
   Defines if user is allowed to skip the Ad or not.
   Setter name**:** setShouldAllowSkip(boolean shouldAllowSkip)
   Values**:**
   True = User can skip Ad.
   False = User can't skip Ad.

4. **Should allow mute**
   Defines if user is allowed to mute the Video Ad or not.

***Setter name*:** setShouldAllowMute(boolean shouldAllowMute)
***Values*:**

True = User can mute video Ad.
False = User can't mute video Ad.

5. **Should confirm skip**

Defines if confirmation dialog is to be shown before skipping the Ad.
***Setter name*:** ShouldConfirmSkip
***Values*:**

True = Confirmation dialog will be shown before skipping the video.
False = Confirmation dialog will not be shown before skipping the video.

6. **Skip confirmation message**

Defines what confirmation message to be shown in skip dialog.
***Setter name*:** setShouldSkipConfirm(boolean shouldSkipConfirm)
***Values*:**

Any String message.
Default value is "Skipping this video will earn you NO rewards. Are you sure?".

7. **Affirmative label for skip dialog**

Defines what should be the label for affirmative button in skip dialog.
***Setter name*:** setSkipConfirmYesLabel(String skipConfirmYesLabel)
***Values*:**

Any String message.
Default value is "Yes".

8. **Negative label for skip dialog**

Defines what should be the label for affirmative button in skip dialog.
***Setter name*:** setSkipConfirmNoLabel(String skipConfirmNoLabel)
***Values*:**

Any String message.
Default value is "No".

9. **Skip timer message**

Defines message to be shown before enabling skip button. Don't forget to add placeholder "**##**" in your custom message.
This placeholder is replaced by property "Default skip time" assigned above.
***Setter name*:** setSkipTimerMessage(String skipTimerMessage)
***Values*:**

Any String message.
Default value is "You can skip video in ## seconds"

10. Incentive message

Defines message to be shown during video progress, that after what time user will be incentivised.

***Setter name*:** setIncentiveMessage(String incentiveMessage)

***Values*:**

Any String message

Default value is “more seconds only for your reward !”

11. Should collect feedback

Defines message to be shown during video progress, that after what time user will be incentivised.

***Property name*:** setShouldCollectFeedback

***Values*:**

True = If you want to collect feedback from the user for the Ad.

False = If you don’t want to collect feedback from the user for the Ad.

User Details

For better targeting of ads you can also provide user details to our SDK using.

```
PokktUserDetails pokktUserDetails = new PokktUserDetails();
pokktUserDetails.setName(" ");
pokktUserDetails.setAge(" ");
pokktUserDetails.setSex(" ");
pokktUserDetails.setMobileNumber(" ");
pokktUserDetails.setEmailAddress(" ");
pokktUserDetails.setLocation(" ");
pokktUserDetails.setBirthday(" ");
pokktUserDetails.setMaritalStatus(" ");
pokktUserDetails.setFacebookId(" ");
pokktUserDetails.setTwitterHandle(" ");
pokktUserDetails.setEducation(" ");
pokktUserDetails.setNationality(" ");
pokktUserDetails.setEmployment(" ");
pokktUserDetails.setMaturityRating(" ");

PokktAds.setUserDetails(pokktUserDetails);
```

Debugging

Other than enabling debugging for Pokkt SDK, it can also be used to:

1. Export log
   Export your log to your desired location, we generally have it in root directory of SD card, if permission for external storage is provided and in cache folder otherwise.

```
PokktAds.Debugging.exportLog(getActivity());
```

2. Export log to cloud
   You can also export log to cloud.

```
PokktAds.Debugging.exportLog(getActivity());
```

Analytics

We support various analytics in Pokkt SDK.
Below is mentioned how to enable various analytics with Pokkt SDK.

Google Analytics

Google analytics Id can be obtained from Google dashboard.

```
AnalyticsDetails analyticsDetail = new AnalyticsDetails();
 analyticsDetail.setSelectedAnalyticsType( AnalyticsType.GOOGLE_ANALYTICS);
 analyticsDetail.setGoogleAnalyticsID( "<Google Analytics Id>");

 PokktAds.Analytics.setAnalyticsDetails(analyticsDetail);
```

Flurry Analytics

Flurry application key can be obtained from Flurry dashboard.

```
AnalyticsDetails analyticsDetail = new AnalyticsDetails();
 analyticsDetail.setSelectedAnalyticsType( AnalyticsType.FLURRY);
 analyticsDetail.setFlurryApplicationKey("<Flurry Application Key>");

 PokktAds.Analytics.setAnalyticsDetails(analyticsDetail);
```

MixPanel Analytics

MixPanel project token can be obtained from MixPanel dashboard.

```
AnalyticsDetails analyticsDetail = new AnalyticsDetails();
 analyticsDetail.setSelectedAnalyticsType(AnalyticsType.MIXPANEL);
 analyticsDetail.setMixPanelProjectToken( "<MixPanel Project Token>");

 PokktAds.Analytics.setAnalyticsDetails(analyticsDetail);
```

Fabric Analytics

Analytics Id is not required in case of Fabric.

```
 AnalyticsDetails analyticsDetail = new AnalyticsDetails();
 analyticsDetail.setSelectedAnalyticsType(AnalyticsType.FABRIC);

 PokktAds.Analytics.setAnalyticsDetails(analyticsDetail);
```

IAP(In App Purchase)

Call trackIAP to send any In App purchase information to Pokkt.

```
InAppPurchaseDetails inAppPurchaseDetails = new InAppPurchaseDetails();
inAppPurchaseDetails.setProductId("<productId>");
inAppPurchaseDetails.setPurchaseData("<purchaseData>");
inAppPurchaseDetails.setPurchaseSignature("<purchaseSignature>");
inAppPurchaseDetails.setPurchaseStore(IAPStoreType.GOOGLE);
inAppPurchaseDetails.setPrice(<100.00>);

PokktAds.Analytics.trackIAP(inAppPurchaseDetails);
```

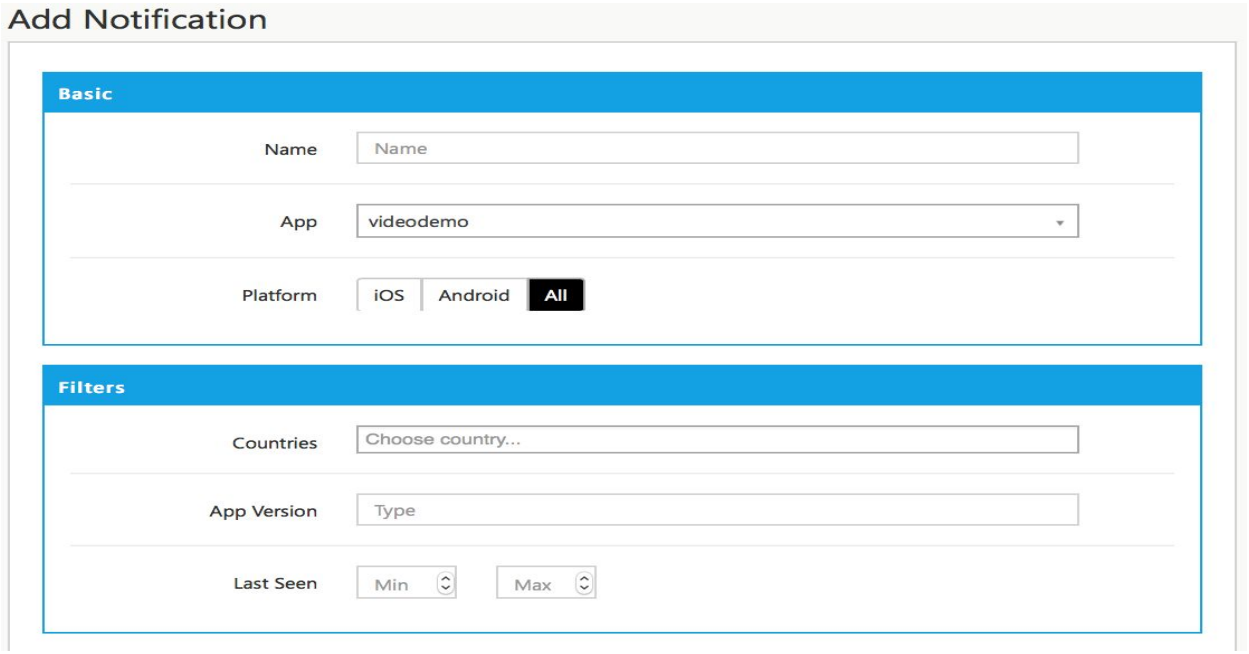
Pokkt Dashboard

InApp Notifications

Open developer dashboard -> Manage App -> Notifications

Create notifications

1. Basic notification information required.



   a. Name:
      A friendly name for notification. It will help you to distinguish between different notifications.
   b. App:
      Select your app for which you want to assign notifications.
   c. Platform:
      Select OS platform for which you want to target notifications.
   d. Countries:
      Select countries where this notifications will be shown to users. Let's say if you have users in multiple countries, you can selectively target notifications to them.
   e. App version:

Enter your app version for which you want to show notifications. Let's say you have multiple version installed among users and you want to send different notifications to different users based on their versions.

f. Last seen:

Set minimum and maximum limit in days for which user can remain away from the app.

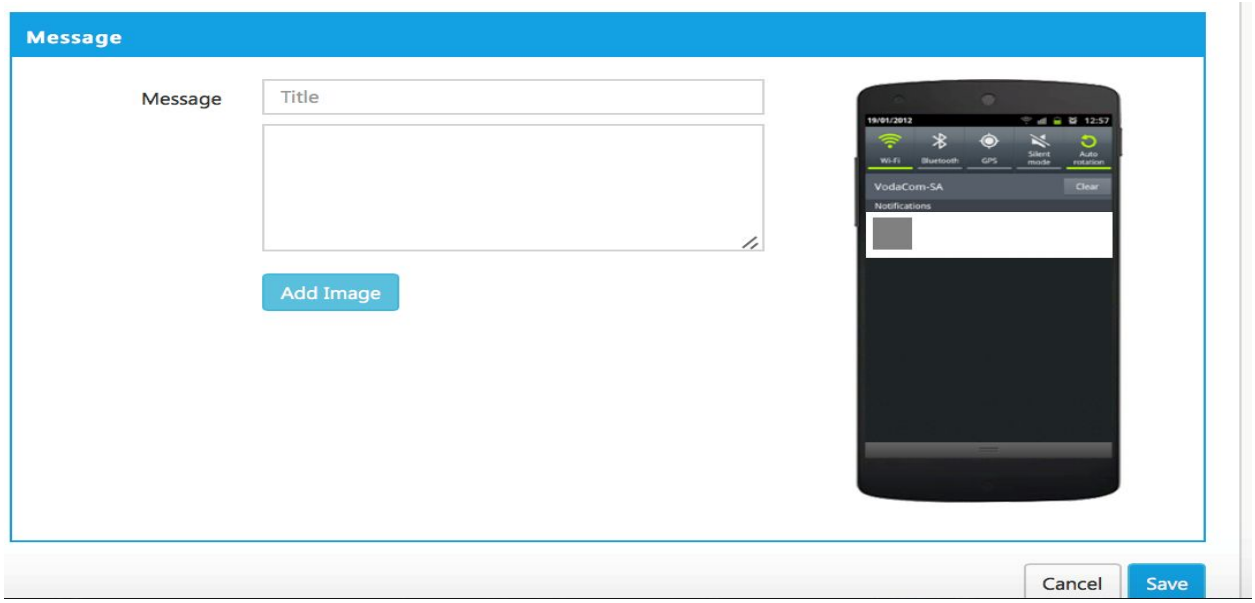
Let's say if min = 2 and max = 4, and user hasn't open your app for atleast 2 days, you can remind by showing notification, but if 4 days have passed app will not show any more notifications to user.

g. Message:

Message you want to show in notifications bar.

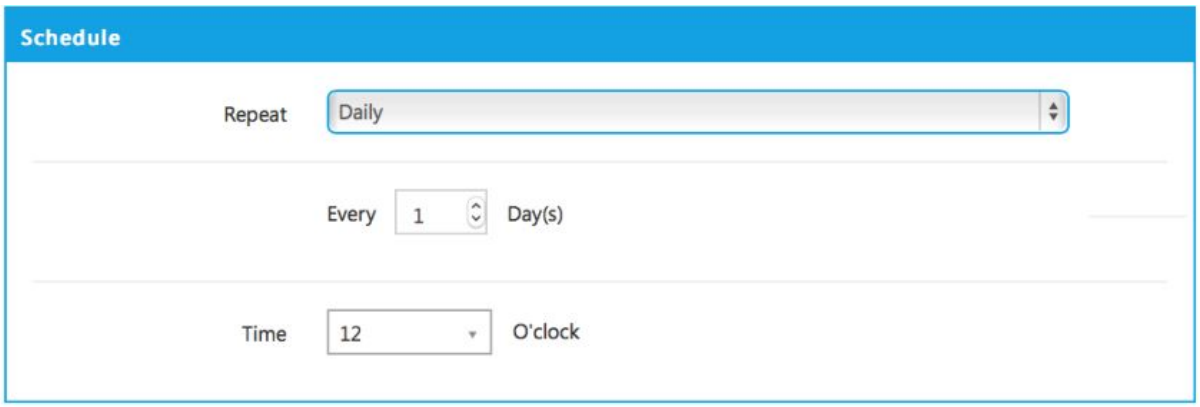
h. Title:

Title for the notification to be shown in notification bar.

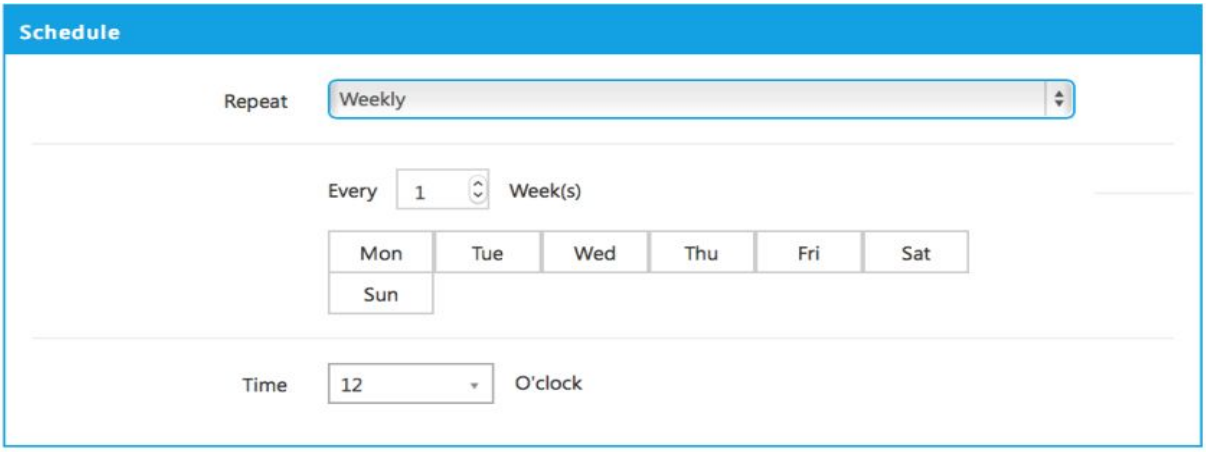


2. Schedule notification

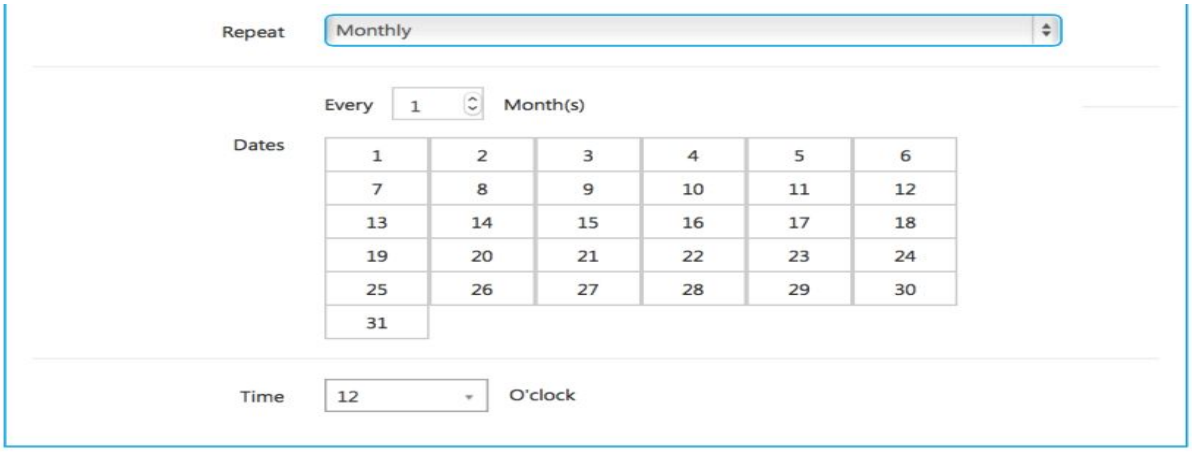
a. Daily



b. Weekly



c. Monthly



Proguard

If you are using proguard in your app, add following rules to your proguard file.

```
            # Pokkt SDK
-keep class com.pokkt.** { public *; }
-dontwarn com.pokkt.**
# moat
-keep class com.moat.** { *; }
-dontwarn com.moat.**
# comscore
-keep class com.comscore.** { *; }
-dontwarn com.comscore.**
-dontnote com.comscore.**
#facebook
-dontwarn com.facebook.**
# fabric
-dontwarn com.crashlytics.**
-dontwarn io.fabric.**
# flurry
-dontwarn com.flurry.**
# MixPanel
-dontwarn com.mixpanel.**
```