POKKT SDK v3.0.0 Integration Guide for Marmalade (Android)

Contents:

1. Introduction:

Thank you for choosing Pokkt SDK for Marmalade. This document contains all the information that is needed by you to setup the SDK with your project

There is a sample app provided with the SDK. We will be referencing this app during the course of explanation in this document. It is suggested that you should check that app to understand the following process in detail.

2. Installation:

All we need is the file provided: *PokktNativeExtension.zip*. And also will provide you one sample project which is *PokktSample.zip* file. Use this as a reference. *VideoScreen.cpp* and *OfferwallScreen.cpp* class is useful for you. Please check how it is implemented. And also here we are giving you complete implementation details.

Export this package and make this as subproject of your project. Please keep as it is this subproject and don't do any changes in this subproject.

Note: Please **do not copy** the code points from this PDF file as it may introduce unwanted characters and space in your code. Instead please refer to sample app source code provided with the sample app.

3. Manifest Changes:

Step 1: Configure “AndroidManifest.xml” by adding permissions

Notes: All these changes has been added in given extension project so you no need to add but if you have your own changes then add it in the same manifest or add this changes in your manifest file.

Required permissions:

Common:

READ_PHONE_STATE
INTERNET
ACCESS_NETWORK_STATE

Video specific:

WRITE_EXTERNAL_STORAGE
WAKE_LOCK

Add following code snippet within <application...> tag in manifest file.

Common:

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

Video specific:

```
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Calendar Event:

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

Step 2: Add Activity definition

Add following Activities in manifest. Add both activities to support both features (offerwall and video).

Offerwall:

```
<activity
        android:name=“com.app.pokktsdk.ShowOfferwallActivity"
        android:configChanges=“keyboard|keyboardHidden|navigation|
        orientation|screenLayout|uiMode|screenSize"
        android:windowSoftInputMode="adjustPan">
</activity>
```

Video:

```
<activity android:name="com.app.pokktsdk.PlayVideoCampaignActivity"
android:configChanges="keyboard|keyboardHidden|navigation|orientation|
screenLayout|uiMode|screenSize" android:screenOrientation="landscape"
android:windowSoftInputMode="adjustPan">
</activity>
```

Step 3: Add BroadcastReceiver (required only for offerwall)

Add following code snippet within </application> tag in manifest.

Offerwall:

```
<receiver android:name="com.app.pokktsdk.AppInstallBroadcastReceiver" >
 <intent-filter android:priority="1000" >
        <action android:name="android.intent.action.PACKAGE_INSTALL" />
        <action android:name="android.intent.action.PACKAGE_ADDED" />
        <data android:scheme="package" />
 </intent-filter>
</receiver>
```

Step 4: Add meta-data information

Add the following within </application> tag in manifest.

```
<meta-data android:name="offerwallDelegate" android:value="
    OfferwallEventsHandler" />

<meta-data android:name="videoDelegate" android:value="
    VideoEventsHandler" />
```

Step 5: Google analytics (optional)

Add following services and receiver in manifest for google analytics (Optional):

```
<receiver android:name="com.google.android.gms.analytics.AnalyticsReceiver"
android:enabled="true">
        <intent-filter>
                <action android:name="com.google.android.gms.analytics.
                        ANALYTICS_DISPATCH" />
        </intent-filter>
</receiver>

<service android:name="com.google.android.gms.analytics.AnalyticsService"
        android:enabled="true"
android:exported="false"/>
```

Step 6: In-app notification:

Add following service in manifest for in-app notification:

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

Step 5: Include Google Play Services SDK

Please follow the instructions below to include the Google Play Services SDK
http://developer.android.com/google/play-services/setup.html

Why is this required?

Google has now changed policy with respect to recognising the devices. It no longer allows the developer to read the Android_ID. Instead a new Advertisers ID is needed to be use.
Please find more details below
https://developer.android.com/google/playservices/id.html

4. Code Integration:

Implementation Steps

- Common

1. For all invocation of Pokkt SDK developer will make use of methods available in *PokktNativeManager* class.This class only have static methods.
2. Before calling any other methods from the *PokktNativeManager* please make sure that you have called the *initPokkt* already.(This does not apply to session related methods namely *startSession* and *endSession*)
3. For almost all methods call *PokktMarmaladeConfig* instance is required. *PokktMarmaladeConfig* is plain object which will hold all the values required by the SDK which you need to assign.
4. In *PokktMarmaladeConfig* you can assign *applicationId*, *securityKey* and *IntegrationType* which are must for all type of integrations.
5. If you are doing server to server integration with pokkt you can also mention *thirdPartyUserId* in *PokktMarmaladeConfig*.
6. Apart from above mentioned parameters you can assign additional ones based on your integration type.(please refer to OfferWall and Video sections below.)
7. While in development please call *PokktNativeManager.setDebug(true);* to see pokkt debug logs and toast messages. please make sure to change this to *PokktNativeManager.setDebug(false);* for production build.

- Session

1. Starting with this version Pokkt SDK is adding session tracking for which we have *startSession* and *endSession* methods in *PokktNativeManager.*
2. You should call *startSession* at the start of his application and once only. You will need to provide PokktMarmaladeConfig instance for this method with *applicationId*, *securityKey* and *IntegrationType* assigned.
3. You should call *endSession* at the end of his application and once only.

- OfferWall

1. In *PokktMarmaladeConfig* for OfferWall you can set two additional parameters which are *offerWallAssetValue* and *closeOnSuccessFlag. offerWallAssetValue* is required if you only want to show campaigns of certain value on OfferWall. It has default value as empty. *closeOnSuccessFlag* is required if you wish to close the OfferWall after user has completed one offer. It's default value is false.
2. Before calling another method for offerWall in *PokktNativeManager*, please make sure that you have already called *pokktInit* first.
3. You will need to inherit *PokktDelegate.cpp* class in your class where you want to listen offerWall related events and you will have to write all offerWall related method. Refer to the *OfferwallScreen.cpp* class to get the better ideas.
4. To show offerWall you can call *PokktNativeManager.getCoins (PokktMarmaladeConfig);*
5. In the screen or activity where you have button to show offerWall, in that activity onResume you should call *PokktNativeManager. getPendingCoins();* so that you get a callback to award points to the user after he has come back to your game after finishing with OfferWall. You will get a callback for this call in your *PokktNativeManager::CoinResponseEvent* implementation or *PokktNativeManager::CoinResponseWithTransIdEvent* implementation in case of success otherwise in *PokktNativeManager::CoinResponseFailedEvent* in case of failure.
6. You can call *PokktNativeManager.checkCampaignAvailable();* to check whether the campaigns are available before showing OfferWall button to user. You will get a callback for this call in your *PokktNativeManager::CampaignAvailabilityEvent* implementation.

- Video

1. In *PokktMarmaladeConfig* for Video you can set five additional parameters which are *autoCacheVideo, skipEnabled, defaultSkipTime, screenName* and *incentivised.*
2. *autoCacheVideo* is required if you want to automatically cache video on user device. It has default value as true. if you set it as false then video will not be automatically cached and you will have to call

PokktNativeManager.cacheVideoCampaign(); to start caching videos on device.

3. If you want to enable/disable the skip button on video screen please set *skipEnabled* as true/false.The default value for *skipEnabled* is false.
4. If you have enabled skipped button by setting *skipEnabled* as true then you can control after how many seconds the skip button will be visible in video by setting *defaultSkipTime* to appropriate value. Since most videos will be 30 sec or less please set *defaultSkipTime* as 10 or less. You can also give your own skip message by setting *customSkipMessage* on *PokktMarmaladeConfig*
5. *screenName* has default value as *default* and can be used by you to give different screen name for different places in your app where you are showing video ads. You will control ad targeting based on these screen names which should match exactly with screen names defined in dashboard. ScreenName can not contain white spaces and only special characters allowed are hyphen and underscore.
6. You can choose to show video with or without incentive to user by setting *incentivised* as true or false. Video gratification will only happen for incentivised playback.
7. You can disable the back button while video is playing by setting *backButtonDisabled* on *PokktMarmaladeConfig*.
8. You will need to inherit *PokktDelegate.cpp* class in your class where you want to listen **Video** related events handler and you will have to write all **Video** related method handler. Refer to the *VideoScreen.cpp* class to get the better ideas.
9. You can call *PokktNativeManager::isVideoAvailable()* to check if the campaign is available before you try to play video.
10. You can call *PokktNativeManager::getVideo(PokktMarmaladeConfig);* to play video with incentives and *PokktNativeManager::getVideoNonIncent(PokktMarmaladeConfig);* to play video without incentives.
11. Please reward user only from the*PokktNativeManager::VideoGratifiedEvent* implementation.

- Export Logs

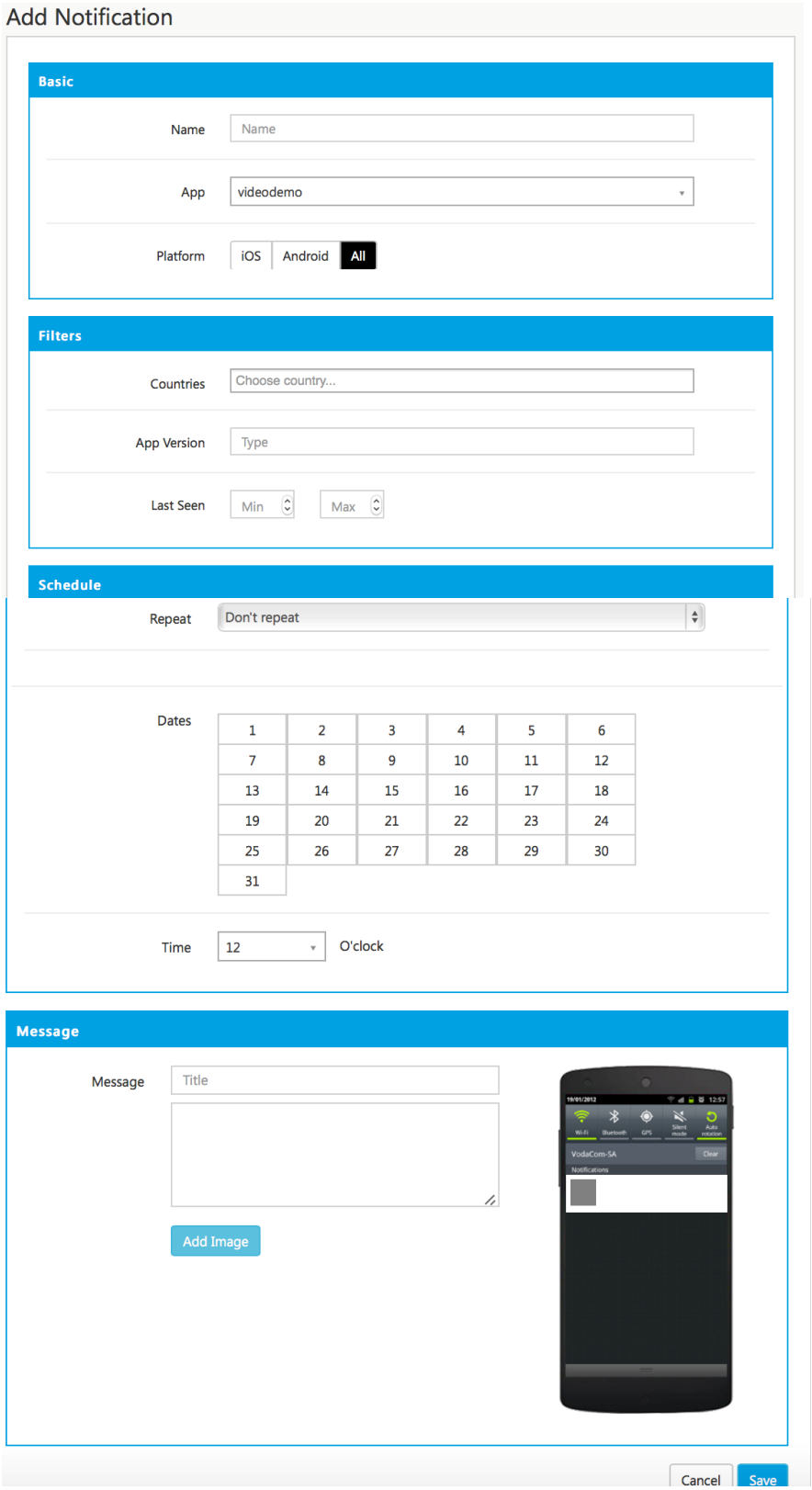
1. Developer should call *PokktNativeManager::exportLogs()* to export the Pokkt SDK logs to folder of your choice.
2. This API shows a folder chooser dialog where user can choose a particular folder.
3. User can also create a new folder where user wants to export the logs.

- Optional Parameters

- *PokktMarmaladeConfig* also has provision for developers to provide extra user data available with them to pokkt.We currently support following data points: *name, age, sex, mobileNo, emailAddress, location, birthday, maritalStatus, facebookId, twitterHandle, education, nationality, employment and maturityRating.*

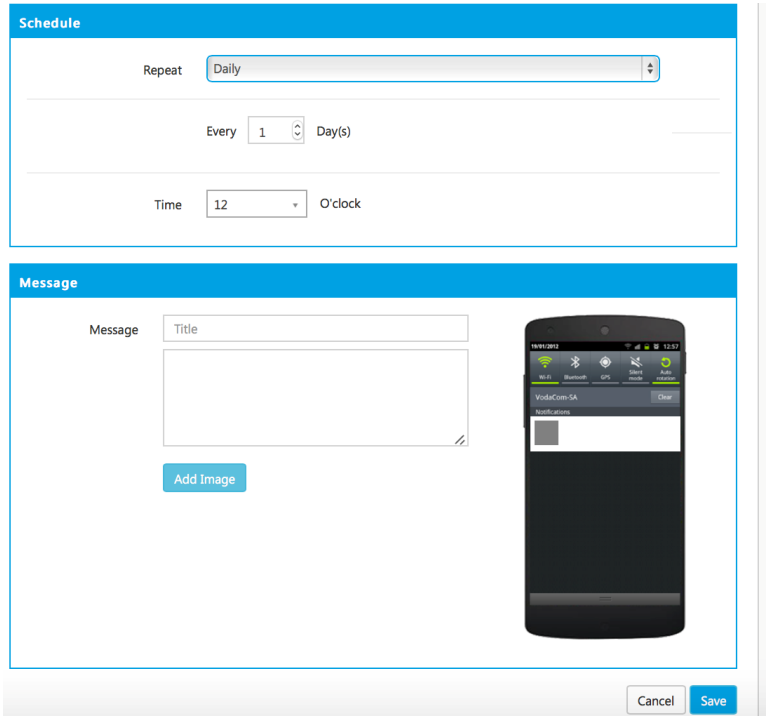
- In-App Notifications

Developer can add In-App notifications in their dashboard.

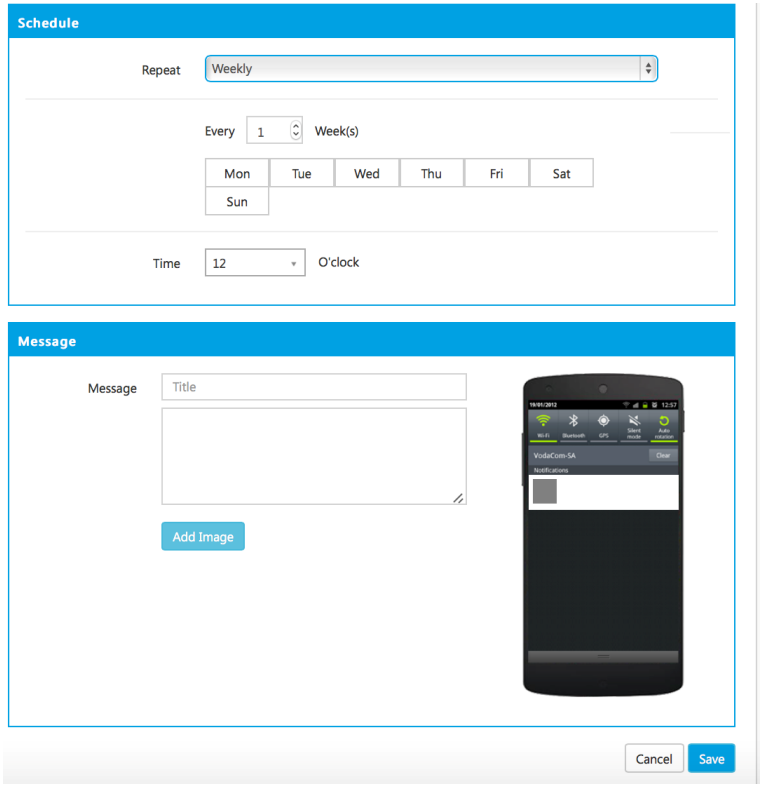


Repeat schedule can be daily, weekly monthly.

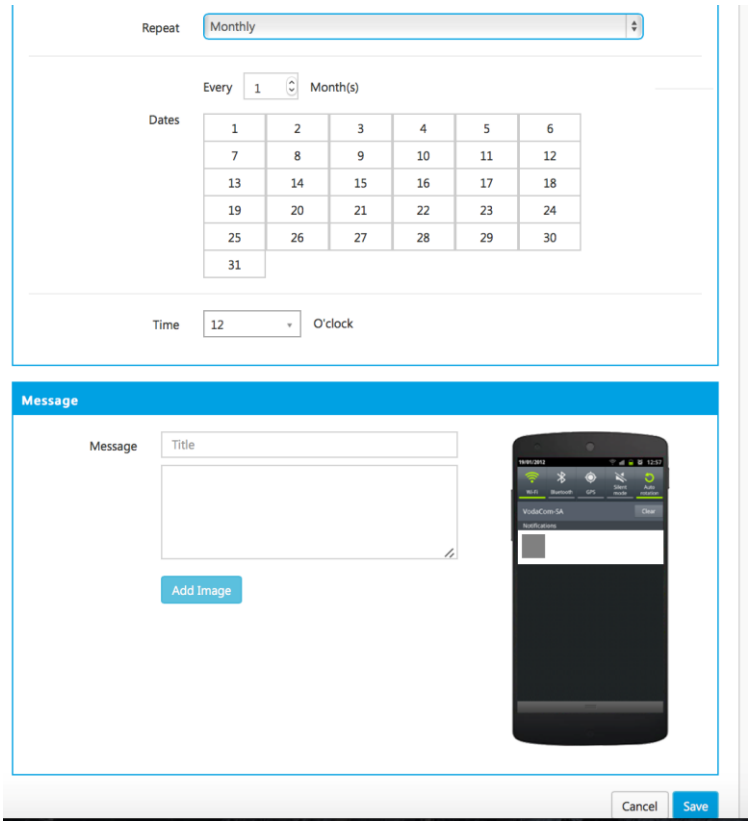
Daily Repeat can have options like frequency of repeat and time in hours of notification.



Weekly repeat can have options like frequency of repeat in weeks, days of repeat and time in hours of notification.



Monthly repeat can have options like frequency of repeat in months dates of repeat and time in hours for notification.



. For don't repeat case, there are options like dates and time in hour for notification.

The notifications are listed and can be edited. Notifications can also be deactivated/activated

5. Functionalities:

5.1 Offerwall

There are five events to listen too. These are:

1. *CoinResponseEvent*
2. *CoinResponseWithTransIdEvent*
3. *CoinResponseFailedEvent*
4. *CampaignAvailablityEvent*
5. *OfferwallClosedEvent*

Add those handler where you want to listen offerwall related events. So please check *OfferwallScreen.cpp* for reference.

Reference on how to consume them:

```
// handle pokkt offerwall events
PokktNativeManager::setListener(PokktNativeManager::CoinResponseEvent,
pokkt_event_selector(OfferwallScreen:: handleCoinResponse), this);

void OfferwallScreen::handleCoinResponse(std::string coins)
{
    int coinsInt = 0;
    std::stringstream(coins) >> coinsInt;
    _coinsEarned += coinsInt;

    char buf[100];
    sprintf(buf, "%d", _coinsEarned);
    pointsEarnedLbl->SetText(buf);
}


PokktNativeManager::setListener(PokktNativeManager::CoinResponseWithTransId
Event, pokkt_event_selector(OfferwallScreen::handleCoinResponseWithTrId),
this);

void OfferwallScreen::handleCoinResponseWithTrId(std::string coinsWithTrId)
{
    // extract comma separated values
    std::stringstream stream(coinsWithTrId);
    std::string value;
    std::vector<std::string> values;
    while (getline(stream, value, ','))
    {
        values.push_back(value);
        if (stream.peek() == ',')
            stream.ignore();
    }
    if (values.size() < 2)
        return; // the values received are not proper
    std::string points = values[0];
```

```
std::string transId = values[1];
 if (points == "-1")
    {
        // no points earned
    }
    else
    {
        // points earned equal to coins with transaction id
    }
}
```

```
PokktNativeManager::setListener(PokktNativeManager::CoinResponseFailedEve
nt, pokkt_event_selector(OfferwallScreen::handleCoinResponseFailed), this);

void OfferwallScreen::handleCoinResponseFailed(std::string message)
{
}
```

```
PokktNativeManager::setListener(PokktNativeManager::CampaignAvailabilityEvent,
pokkt_event_selector(OfferwallScreen::handleCampaignAvailibity), this);

void OfferwallScreen::handleCampaignAvailibity(std::string message)
{
    bool available = message == "true";

    POKKTLOG("[POKKT-CPP] Offerwall Campaign availability: %s", (available ?
"available" : "no campaigns"));
}
```

```
PokktNativeManager::setListener(PokktNativeManager::OfferwallClosedEvent,
pokkt_event_selector(OfferwallScreen::handleOfferwallclosed), this);

void OfferwallScreen::handleOfferwallclosed(std::string message)
{
    POKKTLOG("[POKKT-CPP] Offerwall Closed!");
    PokktNativeManager::getPendingCoins();
}
```

There are two ways to invoke offerwall.

Open Asset Value: In this case, POKKT platform provides all offers with any asset value.
Sample code snippet: PokktMarmaladeConfig::setOfferWallAssetValue(“”);
PokktNativeManager::GetCoins(PokktMarmaladeConfig);

Fixed Asset Value: In this case, POKKT platform provides all offers with fixed asset value.
Sample code snippet: PokktMarmaladeConfig::setOfferWallAssetValue(assetValue);
PokktNativeManager::GetCoins(PokktMarmaladeConfig);

Pending Coins: In case after completing activity, if status of transaction is pending, then call *getPendingCoins()* method. You should always call this method in your calling activity’s resume *(UnPausedNotificationRecieved)* method so that you can check for the pending coins for user.
Sample code snippet: (Check *main.cpp* in sample project)

```
int32 UnPausedNotificationRecieved(void *systemData, void *userData)
{
    s3eDebugTracePrintf("----------App Resumed----------");
    PokktNativeManager::getPendingCoins();
    return 0;
}
```

Check for campaign availability: If you are using optional meta-tag as mentioned in Step 4 of manifest changes, you can call the following to check for campaign availability:

PokktNativeManager::CheckOfferwallCampaign();

The result will be noticed with the event:
PokktNativeManager::CampaignAvailabilityEvent

Moreover, you can listen to the following event to know whether offerwall has been closed or not:
PokktNativeManager::OfferwallClosedEvent

5.2 Video:

There are 7 events to manage the video caching and its playback, these are:

1. *VideoClosedEvent*
2. *VideoDisplayedEvent*
3. *VideoSippedEvent*
4. *VideoCompletedEvent*
5. *VideoGratifiedEvent*
6. *DownloadCompletedEvent*
7. *DownloadFailedEvent*

Add those handler where you want to listen Video campaign related events. So please check *VideoScreen.cpp* for reference.:

Reference on how to consume them:

```
// handle pokkt video ad events

PokktNativeManager::setListener(PokktNativeManager::VideoClosedEvent,
pokkt_event_selector(VideoScreen::handleVideoClosed), this);

void VideoScreen::handleVideoClosed(std::string message)
{
}


PokktNativeManager::setListener(PokktNativeManager::VideoDisplayedEvent,
pokkt_event_selector(VideoScreen::handleVideoDisplayed), this);

void VideoScreen::handleVideoDisplayed(std::string message)
{
}

PokktNativeManager::setListener(PokktNativeManager::VideoSkippedEvent,
pokkt_event_selector(VideoScreen::handleVideoSkipped), this);

void VideoScreen::handleVideoSkipped(std::string message)
{
}


PokktNativeManager::setListener(PokktNativeManager::VideoCompletedEvent,
pokkt_event_selector(VideoScreen::handleVideoCompleted), this);

void VideoScreen::handleVideoCompleted(std::string message)
{
}
```

```
PokktNativeManager::setListener(PokktNativeManager::VideoGratifiedEvent,
pokkt_event_selector(VideoScreen::handleVideoGratified), this);

void VideoScreen::handleVideoGratified(std::string coins)
{
    float coinsFloat = 0;
    std::stringstream(coins) >> coinsFloat;
    _coinsEarned += coinsFloat;
    char buf[100] = "";
    sprintf(buf, "%f", _coinsEarned);
    pointsEarnedLbl->SetText(buf);
}

PokktNativeManager::setListener(PokktNativeManager::DownloadCompletedEvent,
pokkt_event_selector(VideoScreen::handleDownloadCompleted), this);

void VideoScreen::handleDownloadCompleted(std::string coinsToEarn)
{
}

PokktNativeManager::setListener(PokktNativeManager::DownloadFailedEvent,
pokkt_event_selector(VideoScreen::handleDownloadFailed), this);

void VideoScreen::handleDownloadFailed(std::string message)
{
}
```

A video file is cached on user's device. You can set the auto-caching option in the beginning, as mentioned earlier in this document. In case of manual caching, call the following to start video caching:

```
PokktNativeManager::cacheVideoCampaign();
```

Before playing video or showing button to play video, Application should check whether video is cached or not by calling the following:

```
PokktNativeManager::IsVideoAvailable();
```

You should listen to *DownloadCompletedEvent* to check whether download is completed or not, you can show the play buttons once you receive this event.

Furthermore, Application can decide to play video as incentivised (user will be gratified after watching complete video) or non-incentivised (user will not be gratified after watching complete video). You must provide the screen-name parameter for it. Followings are the method calls to me made:

```
PokktMarmaladeConfig::setScreenName("screen_name");
PokktNativeManager::getVideo(PokktMarmaladeConfig);
PokktNativeManager::getVideoNonIncent(Pokkt
MarmaladeConfig);
```

Next, you can listen to *VideoGratifiedEvent* to get the coins earned, if at all, by watching the last video.

6. Debugging and Logging

You can enable the SDK logs by setting the debugging option to true anytime.

```
PokktNativeManager::setDebug(<true/false>);
```

You can use the following command to log some debug messages:

```
PokktNativeManager::showLog("pokkt init...");
```

You can use the following command to display a message as Toast on android device:

```
PokktNativeManager::showToast("pokkt init...");
```

7. Important Points

- Please do not copy the code points from this pdf as it may introduce unwanted characters and space in your code. Instead please refer to sample app source code in pokkt bundle.
- Please also refer to sample app source code for better understanding of implementation.