App Preparation and Submission Guide

Copyrights and Trademarks

Contents

Overview

This guide is a collection of practices and guidelines to help you prepare and submit your mobile app.

It is intended to make our submission process easier and more transparent, and to help you understand both the basic things we look for in an app submission as well as the most important things we need to see to speed along the process.

At the end of this guide, you'll find some examples and advice to address issues we see frequently. However, it's not meant to be exhaustive or a replacement for the Oculus Best Practices Guide, which has more detailed information on how best to build your app. We encourage you to check it out. You may also want to consult the Mobile *Performance Guidelines*, *Design Guidelines* and *User Interface Guidelines* (download *here*).

If you have any feedback about this guide, please let us know on the official *Oculus Mobile SDK Development forum*. We want to make it as useful as possible.

Getting Your App on the Oculus Store

We're always looking for fun and creative VR apps and experiences that you want to share with others. To give you an idea of what to expect once your app is ready to submit, there are two main steps involved in getting your app on the Oculus Store: app review and commercial readiness.

After you submit your app, it will go through an initial review. A member of our developer relations team may contact you with questions or follow-up instructions. Not all apps will be ready for publication upon first review. In the event that your app doesn't pass our initial review, we encourage you to incorporate the feedback of our developer relations team and resubmit your app when you're ready.

If your app passes our review, we'll present you with a distribution agreement as well as a technical and business contact to get your app ready for publication. Once everything is ready, we'll work with you to get your app out to customers.

So without further ado, let's get started! First, we'll help you create your signature file. Next, we'll walk you through the process of submitting your app for review. Finally, we'll share some commonly seen issues to help make your app's review process as smooth as possible.

Create Your Signature File

Oculus mobile apps require two distinct signatures at different stages of development:

- Oculus Signature File (required during development, remove for submission)
- Android Application Signature (required for submission)

Oculus Signature File (osig)

During development, your application must be signed with an Oculus-issued Oculus Signature File, or *osig*. This signature comes in the form of a file that you include in your application in order to access protected low-level VR functionality on your mobile device. Each signature file is tied to a specific device, so you will need to generate osig files for each device that you use for development. When your application is submitted and approved, Oculus will modify the APK so that it can be used on all devices.

Please see our osig self-service portal for more information and instructions on how to request an osig for development: *https://developer.oculus.com/tools/osig/*

Android Application Signing

Android uses a digital certificate (also called a *keystore*) to cryptographically validate the identity of application authors. All Android applications must be digitally signed with such a certificate in order to be installed and run on an Android device.

All developers must create their own unique digital signature and sign their applications before submitting them to Oculus for approval. For more information and instructions, please see Android's "Signing your Applications" documentation: *http://developer.android.com/tools/publishing/app-signing.html*

Make sure to save the certificate file you use to sign your application. Every subsequent update to your application must be signed with the same certificate file, or it will fail.

Note: Your application must be signed by an Android certificate before you submit it.

Android Application Signing and Unity

Unity automatically signs Android applications with a temporary debug certificate by default. Before building your final release build, create a new Android keystore and assign it with the *Use Existing Keystore* option, found in *Edit* > *Project Settings* > *Player* > *Publishing Options*. For more information, see the “Android” section of Unity's documentation here: *http://docs.unity3d.com/Manual/class-PlayerSettings.html*.

Submitting Your App

In this section we will look at the detailed steps and requirements of the submission process.

Submission Overview

A complete submission contains three core components:

1. Application File
2. Image Assets
3. Description Document

You can find a sample submission folder with all of these elements properly formatted and structured at *http://static.oculus.com/submissions/TemplateDocuments.zip*. Please use this to model your app submission. Submissions that don't match this format will be considered incomplete.

Sending Your App to Oculus

Begin by emailing submissions@oculus.com with your company name and a request for a submission folder. We'll send you a link to a Dropbox folder titled Uploads_[YourCompanyName]. This will be the location for all application revisions moving forward for your organization.

In the Dropbox folder, create a new folder ("SubmissionFolder") named ApplicationName_SubmissionDate with the date formatted as YYYYMMDD. Each subsequent revision should be in its own folder.

For example, if you submit a game called "Super Awesome VR" on July 4, 2014, create a folder named SuperAwesomeVR_20140704. If you then create another submission on July 9, create a new folder named SuperAwesomeVR_20140709.

When your Dropbox folder is complete, please email *submissions@oculus.com* indicating which revision you would like us to process and any other details you want to convey. At this point, consider the folder "locked" while we review your submission. Any subsequent changes you make should be part of your next revision.

Application Manifest File Requirements

- The manifest must contain the application's package name in the `<manifest>` tag's package attribute. This package name must be unique.
- `versionName` will be displayed on the Store.
- `versionCode` will be used to deliver updates to users. This value must be an integer greater than 0, and must increment with subsequent revisions. If you're using Unity to build your app, you should edit this value within Unity instead of manually modifying the manifest file, since Unity will overwrite it.
- VR Mode must be properly configured by including the following tag in the `<application>` element in the manifest file of all VR APK files:

```
<meta-data android:name="com.samsung.android.vr.application.mode"
         android:value="vr_only"/>
```

- You must structure your main activity as shown below. In particular, note that your main activity's intent filter should be sent to `android.intent.category.INFO` instead of the more common `android.intent.category.LAUNCHER`. This is to ensure that your app only appears in Oculus Home and can't be launched from the phone's launcher. The only exception to this rule is if you are creating an app that can operate both with and without VR functionality. In that case, use the LAUNCHER category and replace `vr_only` with `vr_dual` in the `com.samsung.android.vr.application.mode` meta data tag.

```
<activity
```

```
android:name="YOUR ACTIVITY"
android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen"
android:label="@string/app_name"
android:launchMode="singleTask"
android:screenOrientation="landscape"
android:configChanges="screenSize|orientation|keyboardHidden|keyboard
android:excludeFromRecents="true"><intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.INFO" />
</intent-filter>
</activity>
```

- The minSdkVersion and targetSdkVersion must be set to 19.

```
<uses-sdk android:minSdkVersion="18" android:targetSdkVersion="18" />
```

- The OpenGL ES version required by the application. It must be either 0x0002000 for OpenGL ES 2.0 or 0x0003000 for OpenGL ES 3.0. <uses-feature android:glEsVersion="0x00030000" android:required="true" />
- The Platform UI enables the passthrough camera. For this to work properly, you will also need to add camera permissions to your manifest. <uses-permission android:name="android.permission.CAMERA" />
- You must also include the following activity in your manifest:

```
<activity
android:name="com.oculusvr.vrlib.PlatformActivity"
android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen"
android:launchMode="singleTask"
android:screenOrientation="landscape"
android:configChanges="screenSize|orientation|keyboardHidden|keyboard"/>
```

- You must remove any unnecessary permissions from your AndroidManifest.xml file. Please be careful to only include permissions that are absolutely necessary for your app to function. These permissions are displayed to the user at installation and if they are overly broad, it may cause the user to cancel the install.

Note: You should not add the noHistory attribute to your manifest.

Image Asset Requirements

Inside your main submission folder, create a new folder called Images.

Images should only be provided as 2560 X 1440 PNG files. They will be automatically resized for optimal display while maintaining the aspect ratio.

Images must not contain side-by-side or post-distortion rendering.

- Main Image (x1): This image will be the primary image displayed whenever the primary context is your application itself. It will be the main image on the Product Details Page dedicated to your app.
- Thumbnail Image (x1): This image will be the identifier for your game when it's displayed in the context of other applications. When a user is browsing or searching, this image should help users identify your app. This can be the same as or different from the main image. If you don't provide a thumbnail image, your main image will be used as your thumbnail.
- Gallery Images (x5): We require at least 5 gallery images. These will only be displayed within the context of your application and should showcase the features of your app.

These images should be saved in a subfolder named ImageGallery and named with the prefix 1_, 2_, 3_, etc. to indicate the order you would like them to be displayed on the Store.

For additional information as to how these images will be used, please refer to SubmissionsTemplate/OculusArtGuidelines_2014.pdf.

Application Description

Create a new copy of SubmissionsTemplate/GearVR_SubmissionDetails.docx in SubmissionFolder and fill out all of the fields.

If you would like your application to be available in other languages, please create an additional copy of the GearVR_SubmissionDetails.docx template for each language, and fill in all fields in the target language. You may append the appropriate language code to the end of each document's name for clarity's sake if you wish, e.g., GearVR_SubmissionDetails_en-US.docx, GearVR_SubmissionDetails_ko-KR.docx.

Initial Submission

When you have completed all of these steps, please mail *submissions@oculus.com* with the location of the folder. Please also submit a short paragraph about your game, describing how it plays, how customers will enjoy it, how long it is, and any other salient features. Please refrain from using "marketing speak," because this will not be your pitch for customers. It’s just a way for us to understand what you’re trying to accomplish and how your game works.

Common Issues

How to address common issues that we frequently see.

This section isn't meant to be exhaustive or a replacement for the *Oculus Best Practices Guide*, which has more detailed information on how best to build your app. We encourage you to check it out.

SDK Version

Always use the latest SDK to develop your app.

Application Signing

Your application must be signed with an Android digital certificate before you submit it. If you are using Unity, be sure to replace the Unity-generated temporary debug certificate with a permanent, private keystore that you have generated and assigned to your app. See *Application Signing* for more information.

Performance

Your app shouldn't crash, freeze, hang, enter an extended unresponsive state, or cause device or system reboot. Playback of videos, movies, and audio must not stutter. Graphics and icons should be rendered stereoscopically and displayed at a comfortable viewing distance. Text should be legible and displayed without any cutoffs or overlaps in the VR environment.

Comfort

All apps should endeavor to reduce or eliminate simulator sickness. For example, user-controlled head tracking and orientation behavior should be maintained at all times.

Interaction with the Oculus Home Menu

- Load from Home
    - Initial Interactive State - App must begin accepting input, respond to head tracking, and display graphics within 4 seconds of app startup.
        - If the app takes longer than 4 seconds to load, appropriate user feedback must be displayed. At minimum, a progress indicator is required. Engaging animations, tutorials/tips, or other visual cues are encouraged.
        - Contractually required license screens/animations/logos or legal disclaimers must NOT remain on-screen for more than 5 seconds each and must be rendered stereoscopically and with head-tracking.
- Return to Home
- App must transition to the Oculus Home screen quickly and in a manner that does not cause user discomfort.

Power Level State Handling

Power level state handling and detection is a mitigation strategy for heat build-up. It's recommended that each app use this feature.

Apps are strongly encouraged to use the minimum possible CPU and GPU clock combination that still allows them to maintain 60 FPS. Higher clock levels will lead to faster heat build-up and reduced battery life. If you have to use higher levels, consider optimizing first and raising the GPU and CPU levels only as a last resort.

Gamepad Notifications

If your app requires a Bluetooth gamepad, please make this apparent to the user with notifications in order to avoid a bad user experience. We suggest one of the following methods:

1. If no gamepad is connected at app launch, present a notification that no controller is detected and that it's required to use the app.
2. If the user attempts an input (swiping or tapping the touchpad) that is normally handled by the gamepad, present a notification that a controller is required for that action. This is more appropriate in apps where a gamepad is optional or just adds extra functionality.

Reserved User Interactions

Back button interactions:

- Long-press: he user presses the button down and holds it for > 0.75 seconds. A longpress should always open the Universal Menu.
    - Apps must implement the Universal Menu accessed by a long-press. This will happen through integration with the latest SDK. The Universal Menu will provide features such as the passthrough camera, a shortcut to Oculus Home, the ability to reorient, brightness, and do-not-disturb mode.
- Short-press: The user presses the button down and releases it before the long-press time is up.
    - A "back" action is interpreted by the application dependent on its current state, but generally it will retreat one level in an interface hierarchy. For example, if the app menu is up and at its initial, top-level screen, a short-press will exit the app menu. If no app menu or other satisfactory stateful condition is current (determined by the application), the short-press will provide a confirmation dialog and then exit the app and return to Oculus Home.

Volume-button interactions:

- Volume buttons must adjust the volume using the VR volume UI provided by the Oculus Mobile SDK.

LTE and WiFi

Using LTE requires significantly more power than using WiFi. All apps that require a significant use of data (beyond squirts for login or leaderboard purposes, for instance) should require a WiFi connection to be in use. Apps with large bandwidth needs such as movie streaming applications must require WiFi.

Agreements

We'll supply you with a Distribution Agreement once your app has been initially reviewed.

Your End User License Agreement

The Distribution Agreement encourages you to supply an end-user license agreement (EULA) for your users. If you supply one, your EULA will be incorporated into the purchase/distribution flow of your app. If you don't supply a EULA, certain pre-established terms will govern the use of your app as set forth in the Distribution Agreement.

Age and Ratings

Users must be at least 13 years old to use Oculus Home. We have two ratings that you may select upon submission: 13+ or 17+. We may have brief descriptive fields available to indicate the nature of your app's content.

Restricted Use of Oculus Name and Logos

The trademarks of Oculus, including its name, logo, and those of it or its partners' products such as “Rift” or “Gear VR” must not appear anywhere in your app. The exception to this is when referencing Oculus hardware or services, for example in a “Quit to Oculus Home?” dialog box or similar UI.

User Data and Permissions

Only request permissions that are required by your application.

If your application transmits data about the user (e.g. for analytics purposes, save state backup, multiplayer pairing, etc.) to an external server, you must declare this in your app submission. Applications distributed through the Store must not damage or destroy data on the user's device.

Monetization and Commerce at Launch

The launch of the Gear VR Innovator Edition will support a free distribution method. This means that your app must not include any monetization features at this time. We plan to introduce commerce within the months following the launch. This is a good time to build your audience, experiment, and show off your work. Having your app available for distribution before commerce is ready doesn’t prevent it from being monetized at a later date.

That about covers it. Thanks for reading, and let us know if we can help in any way. Can’t wait to see what you create!