Qualcomm

Qualcomm Technologies, Inc.

Qualcomm® Snapdragon™ eXtended Reality SDK

User Guide

August 5, 2019

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	May 2016	Initial release
B	Sept 2016	VR SDK 1.1 Update
C	March 2017	VR SDK 2.0 Update
D	July 2017	VR SDK 2.1 Update
E	January 2019	XR SDK 3.0 Update
F	August 2019	XR SDK 4.0 Update

Contents

Tables

1 Introduction

1.1 Purpose

This document provides information about installing the Qualcomm® Snapdragon™ eXtended Reality (XR) SDK.

1.2 Background

High performance, low latency virtual reality on Android requires access to numerous new features and software optimizations on the target platform. The Snapdragon XR SDK provides access to these features which would otherwise be unavailable to developers.

In addition, the Snapdragon XR SDK implements many of the core low latency XR rendering functionality developers require to create high quality XR content.

The core Snapdragon XR SDK features include:

- Asynchronous timewarp
    - Barrel distortion
    - Chromatic aberration correction
    - Display stabilization/reprojection
    - Single buffered rendering
    - Layering (masks/overlays)
- 3DOF/6DOF sensor fusion (DSP, 1000 Hz)
- Stereo eye tracking (DSP, 120 Hz)
- CPU/GPU power management

2 Installation

To install the Snapdragon XR SDK, unzip the distribution package to any preferred location.

Note: To avoid long path name truncation using Windows Command Prompt, specify a short file path location. It is recommended to use Windows PowerShell to avoid truncated path names.

2.1 System requirements

A Snapdragon 835-based device running Android 7.1 loaded with the LA 2.0 (1463+) CRM, or later, is required to utilize the Snapdragon XR SDK libraries.

2.2 SDK contents

The Snapdragon XR SDK contains the items listed in Table 2-1.

Table 2-1 SDK contents

/3rdParty	3rd party libraries used by the SDK
/controllers	Device services
/doc	SDK API documentation
/framework	Common utility code used by the SDK samples
/samples	Samples demonstrating use of the SXR SDK APIs
/sxrApi	Native C/C++ headers and libraries for interfacing with the SDK APIs

3 Using the SDK

3.1 Integrating with native OpenGL ES applications

3.1.1 SDK dependencies

- SXR HOME
    - Set SXR_HOME environment variable to the root of the Snapdragon XR SDK installation.
- Android SDK
    - Set ANDROID_HOME environment variable to the root of the Android SDK installation
- Java JDK
    - Set JAVA_HOME environment variable to the root of the JDK 1.8 installation

3.1.2 SDK libraries

After installing the Snapdragon XR SDK, modify the application's "Android.mk" file to pull in "libsxrapi.so".

This is done using the "PREBUILT_SHARED_LIBRARY" and "LOCAL_SHARED_LIBRARIES" commands in the Android NDK build system.

See the Snapdragon XR sample or the Android NDK build documentation for more information.

Manually copy the file "sxrApi.jar" from the Snapdragon XR SDK to the "libs" directory in your application.

3.1.3 Application manifest

In the application's build file "AndroidManifest.xml", change the "<application>" entry to include "android:hasCode="true"".

This tells the Android build system that there will be Java code in the native application.

If this is not done, the Android build system will not add the "sxrApi.jar" file to the final APK.

3.1.4 Application Java activity

To include the Snapdragon XR SDK jar file in the final APK, your application must have at least a shell Java Activity class.

This can be nothing more than an Activity that extends "android.app.NativeActivity". Its only function can be to call "System.loadLibrary("sxrapi")" and your application's library. Be sure to modify "AndroidManifest.xml" to change the Activity name to this new Java class.

3.1.5 Android activity lifecycle

Add the following Snapdragon XR calls to your native code:

android_main()

> Call "sxrInitialize()" before entering the main loop

> Call "sxrShutdown()" before the application exits

On APP_CMD_RESUME

> Call "sxrBeginXr()" if XR has been initialized

On APP_CMD_PAUSE

> Call "sxrEndXr()" if XR has been initialized and "sxrBeginXr()" has been called

3.1.6 Main application loop

At the beginning of each frame, call "sxrGetPredictedDisplayTime()" to request the estimated time from the current point to the point when the frame being generated will appear on the display.

The predicted time should be supplied to the "sxrGetPredictedHeadPose()" to get a quaternion representation of the current head orientation (and position if the device is equipped for 6DOF).

See the Snapdragon XR API documentation for further details on these functions.

3.1.7 Sample SimpleVR

The sample SimpleVR demonstrates the use of the Snapdragon XR SDK.

From the sample directory, execute the gradle wrapper batch file to compile and install the application. The gradle wrapper will install the desired version of gradle.

> **gradlew build**

3.1.8 Sample ControllerTest

Setup

- Install the ControllerTest apk
- Install the XimmerseControllerService apk
- Install the BTConfig[1.0.0-Flip] apk from https://github.com/Ximmerse/SDK_Flip/tree/master/Tools

Pairing

1. Launch the XimmerseControllerService app

2. Press and hold the Back button then press and hold the Start button on the controller

3. Select Bind in the XimmerseControllerService app

4. Release both buttons on the controller

ControllerTest - Demonstrates the use and integration of the xImmerse 3DoF controller.

XimmerseControllerService - Service that talks to Ximmerse Controller Daemon and provides the values to the SXR SDK in a format that is required by the SXR SDK.

BTConfig[1.0.0-Flip] – Ximmerse deamon that talks to the bluetooth controller.