Target setup for Android

This documentation serves as a guide for setting up the developer's system to initiate work with Android for CGI Studio. It is tailored for experienced users, omitting explanations for the more straightforward aspects.

- Android CMake Specific Content
  - Candera Device Packages
- Android Application Development Environment Setup Guide
  - Configuration and System Requirements
- Android Application Guides
  - NativeViewLibrary
  - CGI-Player Launcher
  - CourierSampleApp
  - MLC

Android CMake Specific Content

Candera Device Packages

Supported Device Packages

Candera provides device implementations for following device types:

Name	Description	Possible values
Android		ON/OFF
Other proprietary devices	Can be provided upon special request	ON/OFF

Device-Specific CMake Settings

Name	Description	Possible values
CGIDEVICE_NAME	The name of the device package	Each device package following the Candera device CMake rules provided in the folder <CANDERA_SRC>/CanderaPlatform/Device
CGIDEVICE_TARGET_BUILD	Enable or disable building for target	ON/OFF
CGISTUDIO_PATH_3RD_PARTY	Path to the 3rd party repository location	A string value containing the system file location of 3rd party software, auto-detected by finding possible locations of the name "[ic01_]cgi_studio_3psw"
CGIDEVICE_TARGET_ <CGIDEVICE_NAME> _DRIVER_PATH_INCLUDE	If you don't use the default driver path, the path to the driver include directory can be changed with this setting.	Path to driver include directory (by default set to include directory in 3rd party repository)
CGIDEVICE_TARGET_ <CGIDEVICE_NAME> _DRIVER_PATH_LIB	If you don't use the default driver path, the path to the driver library directory can be changed with this setting.	Path to driver library directory (by default set to library directory in 3rd party repository)

To build a certain device package selected it from the drop-down list of the CMake property **"CGIDEVICE_NAME"**:

For each device package, either building OpenGL based host simulation or target binaries can be selected via the Boolean CMake property "***CGIDEVICE_TARGET_BUILD***".

The following table illustrates configurations available for the supported device package:

CGIDEVICE_NAME	CGIDEVICE_TARGET_BUILD	PUB_CGI_CPU	PUB_CGI_OS	Default driver path following <CGISTUDIO_PATH_3RD_PARTY> /[src/lib]/

The CMake properties "PUB_CGI_CPU", "PUB_CGI_OS" usually cannot be configured via the CMake user interface, but are autonomously detected by
the build system. The same applies for the variable "PUB_CGI_COMPILER".

If you don't use the CGI Studio 3rd party software folder structure, the driver paths can be specified in the variables

- CGIDEVICE_TARGET_<CGIDEVICE_NAME>_DRIVER_PATH_INCLUDE and
- CGIDEVICE_TARGET_<CGIDEVICE_NAME >_DRIVER_PATH_LIB

Depending on these properties, either MS Visual Studio solutions or Unix makefiles will be generated into the specified build
directory.

Android Application Development Environment Setup Guide

To ensure that the system is setup properly to build and run the Android Demo Applications provided, this guide has been created.

Configuration and System Requirements

By following these steps, you should have a consistent and well-configured development environment for your
Android application on Windows 10/11. Remember to restart your system after making changes related to CMake
in Android Studio and after updating Environment Variables.

Operating System

Ensure that your development machine is running Windows 10/11. This guide has been refined to work on Windows 10/11. Other Operating Systems might work, but are not supported.

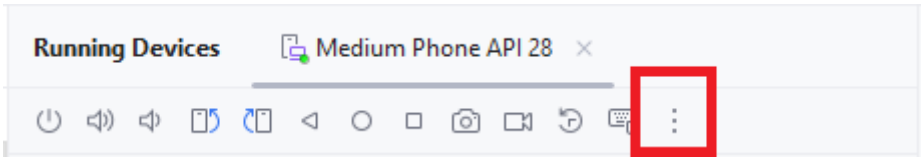
Android Studio

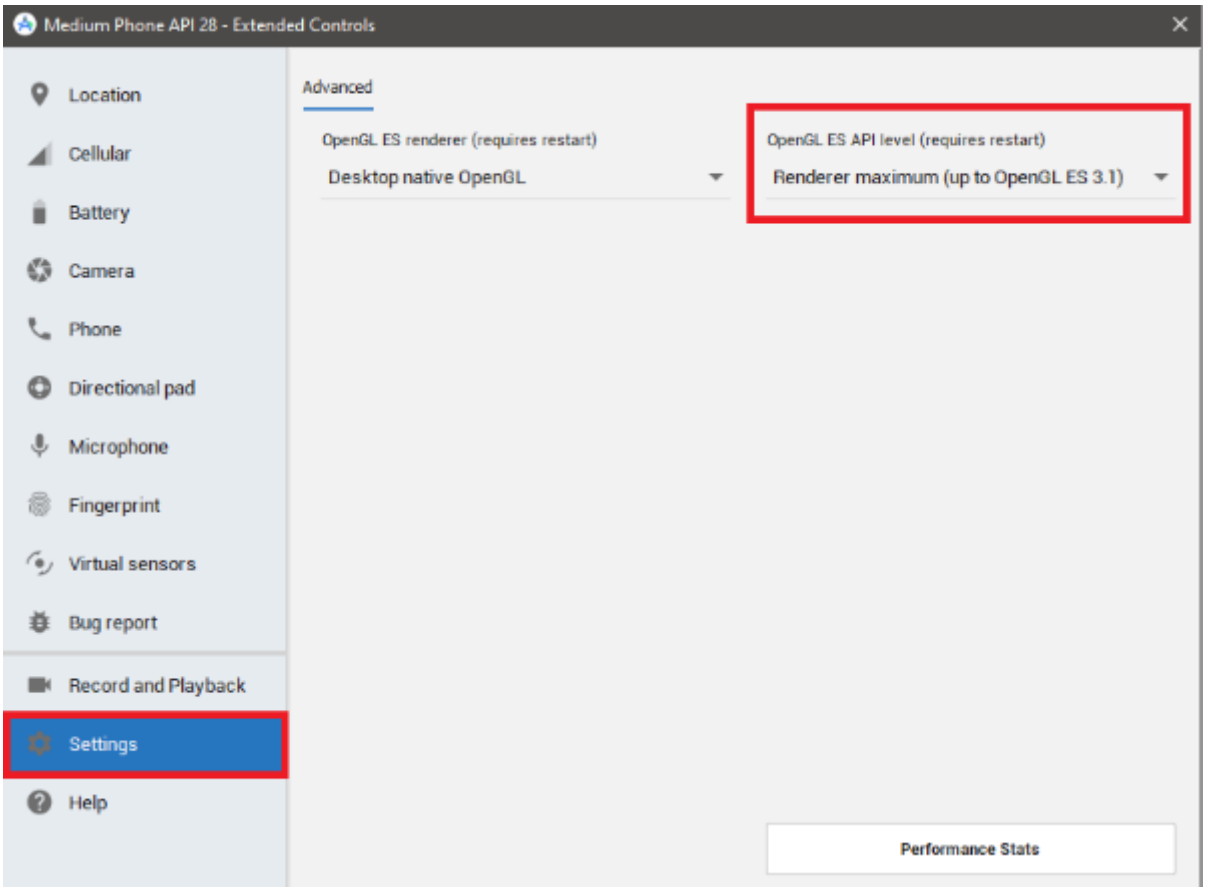
Android Studio Jellyfish (2023.3.1) through Android Studio Meerkat (2024.3.2) have been tested and confirmed to work. Android Studio can be downloaded from the official Android Studio archive:

https://developer.android.com/studio/archive.

Android Studio Emulator

When using the Android Studio Emulator for testing, ensure that the Android version and ABI match those of the Android application. Additionally, make sure that, for the emulator, the OpenGL ES API level (requires a restart) is set to "Renderer maximum (up to OpenGL ES 3.1)." This OpenGL ES API level can be configured in the Android Studio Emulator's Extended Controls.

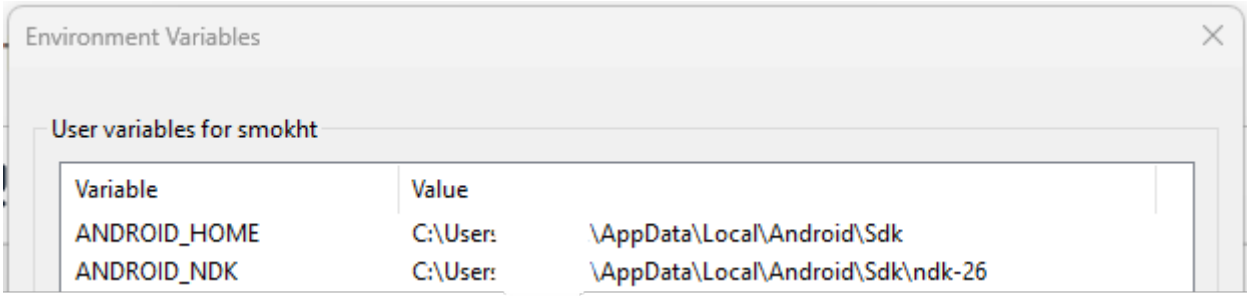




Android Software Development Kit (SDK)

After installing and setting up Android Studio, the SDK path has to be added to the System Environment Variables under the User Variables. After saving make sure to restart the system to apply the changes correctly.

- **Variable Name:** ANDROID_HOME
- **Variable Value:** %LOCALAPPDATA%\Android\Sdk



Android Native Development Kit (NDK)

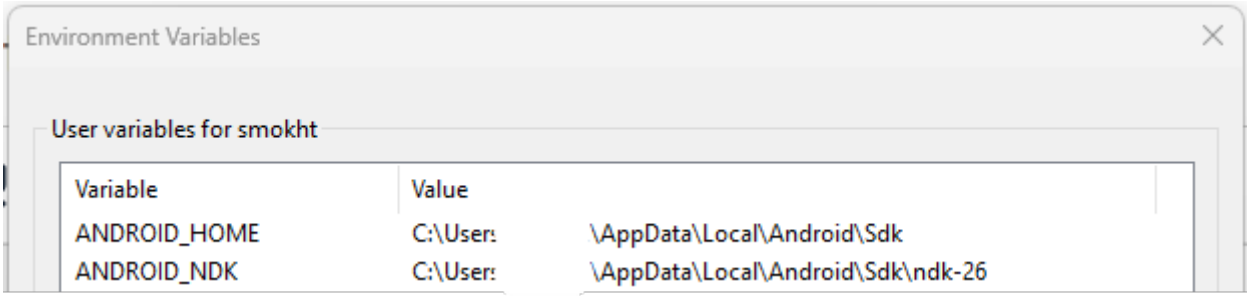
To make use of NDK 26, we need to manually install this. We would also require some files from the NDK 17 release for the build in this release. Download both NDK 26

https://dl.google.com/android/repository/android-ndk-r26b-windows.zip and NDK 17

https://dl.google.com/android/repository/android-ndk-r17c-windows-x86_64.zip from the official download pages.

1. Navigate to %LOCALAPPDATA%\Android\Sdk in Windows File Explorer.
2. Create a new directory named ndk-26.
3. Unzip the contents of android-ndk-r26b-windows.zip into the newly created %LOCALAPPDATA%\Android\Sdk\ndk-26 directory.
4. Unzip the contents (**except llvm)** from android-ndk-r17c-windows-x86_64.zip\android-ndk-r17c\toolchains\ to %LOCALAPPDATA%\Android\Sdk\ndk-26\toolchains.
5. After installing and setting up the NDK, the NDK path has to be added to the System Environment Variables under the User Variables. After saving make sure to restart the system to apply the changes correctly.

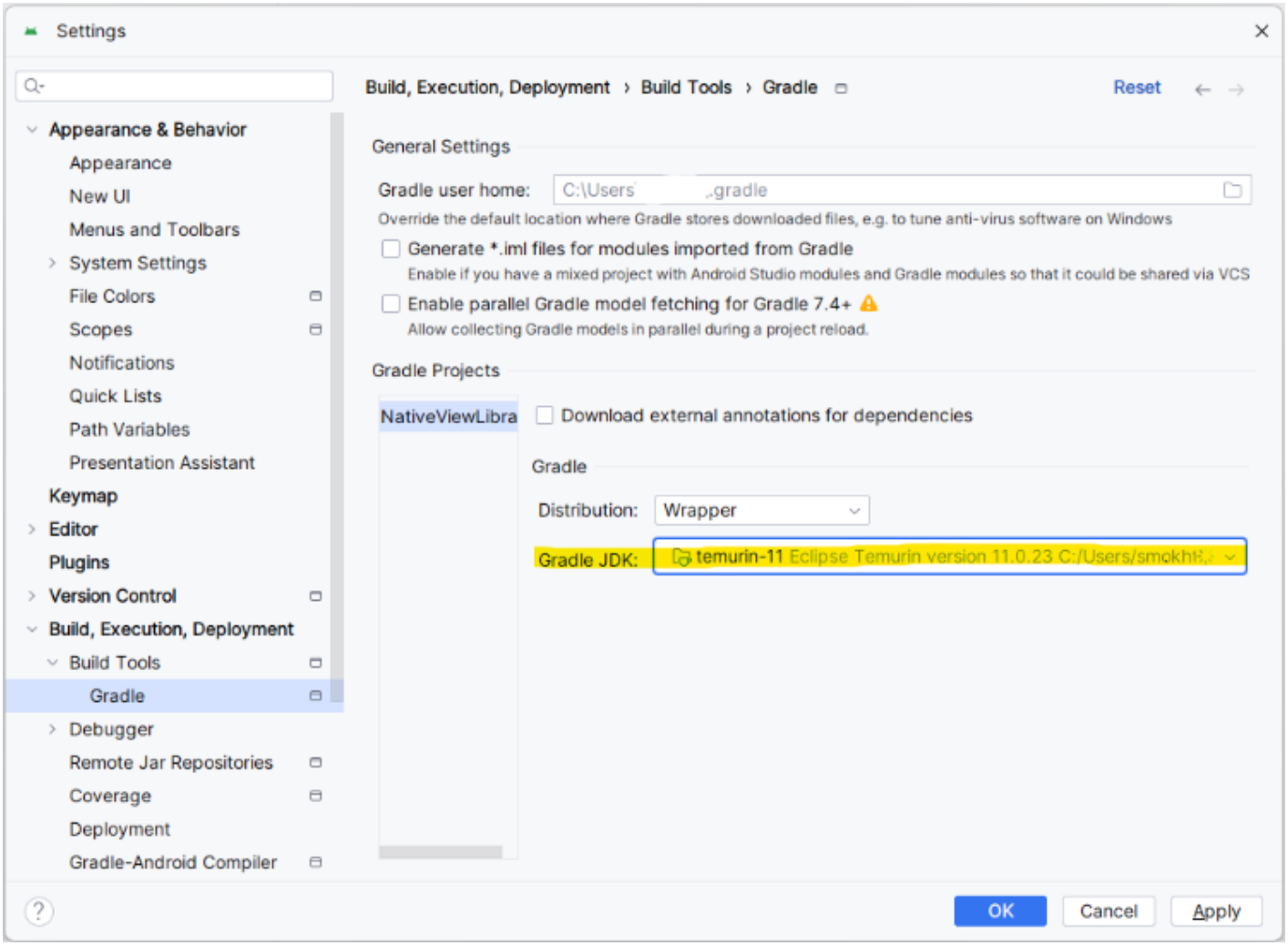
- **Variable Name:** ANDROID_NDK
- **Variable Value:** %LOCALAPPDATA%\Android\Sdk\ndk-26



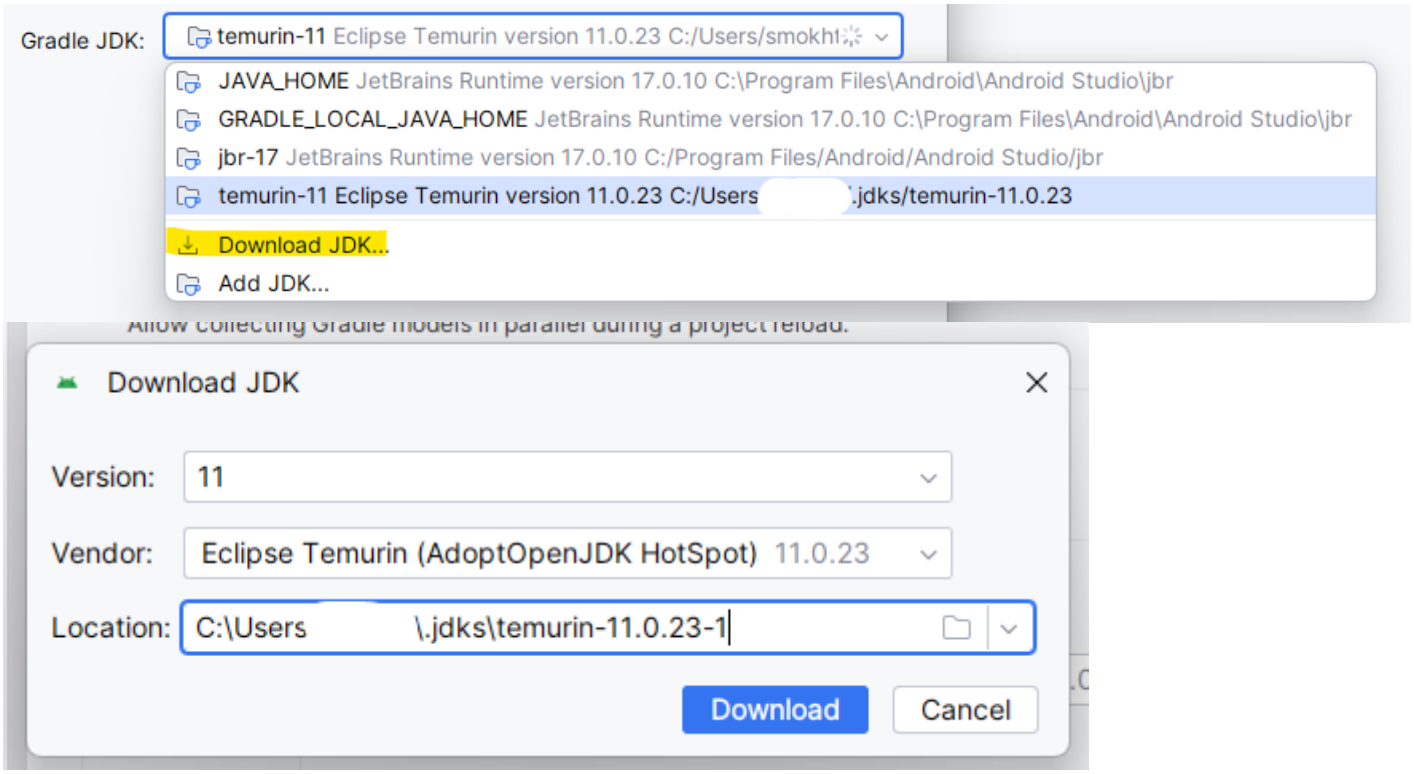
Java Development Kit (JDK)

The JDK can be downloaded and used within Android Studio directly. For Android Applications we make use of the Eclipse Temurin 11. When the Android Demo Application has been opened with Android Studio, the Android JDK should be set. On the specific Android Application wiki page the used JDK version is written down.

1. In Android Studio, open [File | Settings | Build, Execution, Deployment | Build Tools | Gradle].
2. Choose Gradle JDK: Temurin-11 Eclipse Temurin Version 11.



If this version is not available in the dropdown, download it by clicking on “Download JDK...” in the dropdown of Gradle JDK.



Ninja

The "ninja" version used is 1.10.2 and can be downloaded from https://github.com/ninja-build/ninja/releases/download/v1.10.2/ninja-win.zip. Copy "ninja.exe" to "C:\Programs\" and add this path "C:\Programs\" to the Environment Variables -> System Variables -> Path.

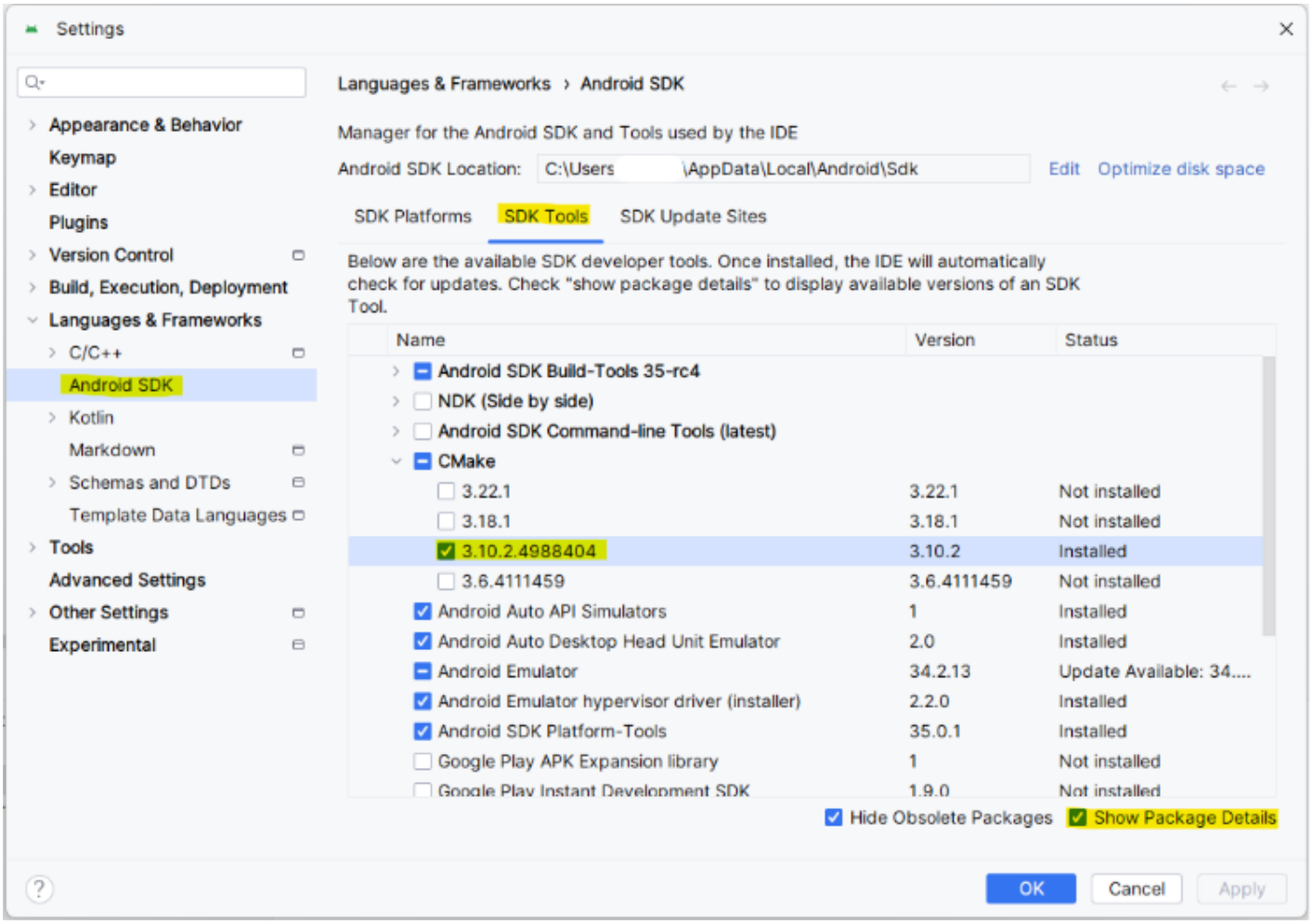
CMake

For the current Android demo applications that use the NativeView approach (e.g., NativeViewLibrary and AndroidCgiPlayerLauncher), we rely on specific CMake versions.

- **Up to CGI 3.14 -> CMake 3.10.2 is used**
- **From CGI 3.15 -> CMake 3.22.1 is used**
- **The supported CMake version can be also found in the Android application module (app) build.gradle.**

These CMake versions can be installed directly through the Android Studio SDK Manager.

1. In Android Studio, open [File | Settings | Languages & Frameworks | Android SDK] and navigate to the SDK Tools tab.
2. Click on Show Package Details on the bottom right of the page.
3. In the list, make sure only CMake 3.10.2 is selected and installed.
4. After installing and closing the settings page, make sure to restart the system to apply the changes correctly.



Alternative CMake Approach

For applications other than the NativeView Library approach, (e.g. SampleApp) the '.so' library must be configured and generated with CMake outside of Android Studio. It should be built using 'ninja' with Windows CMD. We recommend using **CMake 3.24.2**, which can be installed through the CMake installer executable available for download on the official CMake website:

https://cmake.org/download/. Once the build is complete, manually copy the '.so' library to the corresponding ABI directory in the project. Detailed configuration and build steps will be provided in the respective Android Application Wiki page.

Other Dependencies

Depending on your project, additional dependencies might be required (e.g., specific libraries, tools).
Document and install these as needed.

Troubleshooting

- If encountering any issues during the setup, refer to the official documentation for Android Studio,
  CMake, and NDK for troubleshooting steps.

Environment tested with Android Applications

- NativeViewLibrary
- AndroidCgiPlayerLauncher
- SampleApp (Previously known as CourierSampleApp)

Android Application Guides

This document mandates adherence to the Android Application Development Environment Setup Guide, as a prerequisite for system compliance.

NativeViewLibrary



Requirements

This document mandates adherence to the Android Application Development Environment Setup Guide, as a prerequisite for system compliance.

> The project requirements

- NativeViewLibrary Android application uses the **Android Gradle Plugin version 7.2.1** and **Gradle version 7.3.3**.
- A **custom NDK 26** is required, as described in the Android Native Development Kit (NDK) chapter.
- **CMake 3.22.1** is required as described in the CMake chapter.
- **Eclipse Temurin JDK 11** is required as described in Java Development Kit (JDK) chapter.
- Supported API levels: **23–28** (The application is compatible with later Android versions up to API 36 on Android 16).
- Supported ABIs: **arm64-v8a, x86_64, x86**

- The NativeViewLibrary Android application integrates the NativeView library, which is provided externally from **../libs/nativeview-lib** and already imported into the project.

Building and Running the Android Application

To build and run the Android application, perform the following steps:

1. **Substitute the CGI Studio root as described in the windows-commands/subst article.**

   Open a Command Prompt in the CGI Studio root directory (e.g., "...\cgistudio") and type:

```
subst x: .
```

   Press **enter**. The CGI root is now available under the **X:** drive in Windows Explorer.

2. **Configure local.properties:**

   - Open **local.properties.sample** file from the root of the Android application.
   - Copy the content of the sample to the **local.properties** file in the root of the Android application.
   - Update the **USERNAME** in the SDK and NDK paths to match your system.
   - Configure from where the asset should load under the **Asset Configuration** block.
   - Set the ABI to the target CPU architecture under the **Target ABI** block.
   - Save **local.properties** and **sync project with Gradle files** (Ctrl+Shift+O).

> Note: If the SDK or NDK paths do not exist, Android Studio may reset **local.properties** after syncing the project with Gradle files.

3. **Open the Android Application:**

   Launch Android Studio and open the project located at:

```
X:\cgi_studio_apps\References\Android\NativeViewLibrary
```

4. **Create an Android asset**

   Run the following batch file:

```
X:\bin\SceneComposer\SceneComposer_Android_OpenGLES30.bat
```

- Choose an asset template (e.g., “3D Basic”) and save this as Test.bin
  - **If from sdcard:** to the **/sdcard/** directory of the target or emulator. In Android Studio, this can be done in the "Device Explorer" of the target or emulator.
  - **If from APK:** to the following folder:

```
X:\cgi_studio_apps\References\Android\NativeViewLibrary\sample\src\main\assets
```

5. **Build variant:**

   - In Android Studio, open [View | Tool Window | Build Variants].
   - In the Build Variants tool window, select debug or release under **Active Build Variant.**

6. **Run the application**:

   - Select the target device or emulator in Android Studio.
   - Click **Run** (the play button)
   - The first time the Application starts, it will request permissions. Accept and restart the application. The asset should now be running.

7. **APK**

After the application is running, the current built APK can be found in the build directory. This can located by opening [Build | Analyze APK]. The usual directory would be:

```
X:\cgi_studio_apps\References\Android\NativeViewLibrary\sample\build\intermediates\apk\
```

> If the build was for release, the APK will be signed.

Troubleshooting

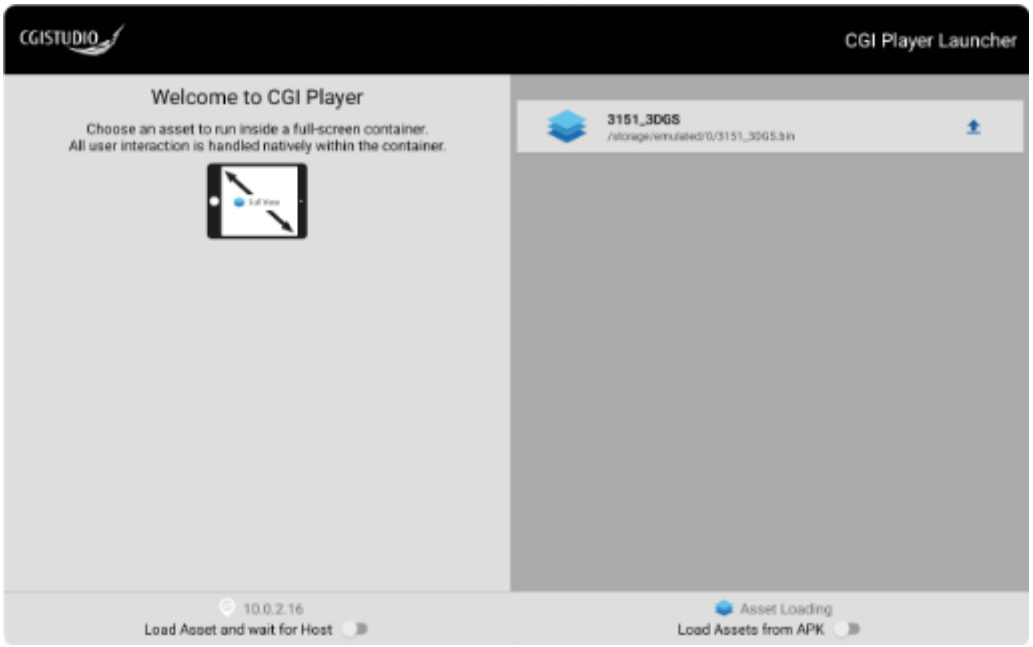
- If encountering any issues during the setup, refer to the official documentation for Android Studio,
  CMake, and NDK for troubleshooting steps.

Platform Tests

- Google Pixel C, ARM Cortex A57, Android 8.1.0, API 27
- Huawei MediaPad M5, ARM Cortex A73, Android 9, EMUI 9.1.0, API28
- Emulator Pixel 2, x86_64, Android 6, API 23 (Without Play/APIs)
- Emulator Medium Phone, x86_64, Android 9, API 28 (Without Play/APIs)

- Emulator Pixel 2, x86, Android 9, API 28 (Without Play/APIs)
- Emulator Google Pixel Tablet, x86_64, Android 14.0, API 34
- Emulator Google Pixel Tablet, x86_64, Android 16.0, API 36
- Emulator Google Pixel 9 Pro XL, x86_64, Android 16.0, API 36

CGI-Player Launcher



Requirements

This document mandates adherence to the Android Application Development Environment Setup Guide, as a prerequisite for system compliance.

> The project requirements

- CGI-Player Launcher Android application uses the **Android Gradle Plugin version 7.2.1** and **Gradle version 7.3.3**.
- A **custom NDK 26** is required, as described in the Android Native Development Kit (NDK) chapter.
- **CMake 3.22.1** is required as described in the CMake chapter.
- **Eclipse Temurin JDK 11** is required as described in Java Development Kit (JDK) chapter.
- Supported API levels: **23–28** (The application is compatible with later Android versions up to API 36 on Android 16).
- Supported ABIs: **arm64-v8a, x86_64, x86**
- The CGI-Player Launcher Android application integrates the NativeView library, which is provided externally from **../libs/nativeview-lib** and already imported into the project.

Building and Running the Android Application

To build and run the Android application, perform the following steps:

1. **Substitute the CGI Studio root as described in the windows-commands/subst article.**

   Open a Command Prompt in the CGI Studio root directory (e.g., “...\cgistudio”) and type:

```
subst x: .
```

   Press **enter**. The CGI root is now available under the **X:** drive in Windows Explorer.

2. **Open the Android Application:**

   Launch Android Studio and open the project located at:

```
X:\cgi_studio_apps\References\Android\AndroidCgiPlayerLauncher
```

3. **Configure local.properties:**

   - Open **local.properties.sample** file from the root of the Android application.
   - Copy the content of the sample to the **local.properties** file in the root of the Android application.
   - Update the **USERNAME** in the SDK and NDK paths to match your system.
   - How the asset can be loaded is described in the **Asset Configuration** block.
   - Set the ABI to the target CPU architecture under the **Target ABI** block.
   - If you need Vulkan support, change `angle_vulkan=off` to `angle_vulkan=on` in **local.properties**.
   - Save **local.properties** and **sync project with Gradle files** (Ctrl+Shift+O).

> Note: If the SDK or NDK paths do not exist, Android Studio may reset **local.properties** after syncing the project with Gradle files.

4. **Create an Android asset**

   Run the following batch file:

```
X:\bin\SceneComposer\SceneComposer_Android_OpenGLES30.bat
```

   - Choose an asset template (e.g., “3D Basic”) and save this as Test.bin

- **If from sdcard:** to the **/sdcard/** directory of the target or emulator. In Android Studio, this can be done in the "Device Explorer" of the target or emulator.
- **If from APK:** to the following folder:

```
X:\cgi_studio_apps\References\Android\AndroidCgiPlayerLauncher\app\src\main\assets
```

5. **Build variant:**

- In Android Studio, open [View | Tool Window | Build Variants].
- In the Build Variants tool window, select debug or release under **Active Build Variant.**

6. **Run the application**:

- Select the target device or emulator in Android Studio.
- Click **Run** (the play button)
- The first time the Application starts, it will request permissions. Accept and restart the application.
- The assets are loaded from the location that is set in the **Asset Location** option in the application.
- The desired asset can be clicked on and the asset should be running.

6.1 **Launching an asset via ADB (command-line)**

The asset path can be found in the CGI-Player Launcher app. Replace the asset path and asset filename in the command. Use this method to start the CGI-Player Launcher and open a selected asset from cmd:

```
adb shell am start -n at.cgistudio.player/.presentation.nativeview.NativeViewActivity -e extra_asset_path "/storage/emulated/0/3151_3DGS.bin" -e extra_asset_name "3151_3DGS.bin" -e extra_load_from_assets false -e extra_wait_for_connection false
```

7. **APK**

After the application is running, the current built APK can be found in the build directory. This can located by opening [Build | Analyze APK]. The usual directory would be:

```
X:\cgi_studio_apps\References\Android\AndroidCgiPlayerLauncher\app\build\intermediates\apk\
```

> If the build was for release, the APK will be signed.

Troubleshooting

- If encountering any issues during the setup, refer to the official documentation for Android Studio,
  CMake, and NDK for troubleshooting steps.

Platform Tests

- Google Pixel C, ARM Cortex A57, Android 8.1.0, API 27
- Huawei MediaPad M5, ARM Cortex A73, Android 9, EMUI 9.1.0, API28
- Emulator Pixel 2, x86_64, Android 6, API 23 (Without Play/APIs)
- Emulator Medium Phone, x86_64, Android 9, API 28 (Without Play/APIs)
- Emulator Pixel 2, x86, Android 9, API 28 (Without Play/APIs)
- Emulator Google Pixel Tablet, x86_64, Android 14.0, API 34
- Emulator Google Pixel Tablet, x86_64, Android 16.0, API 36
- Emulator Google Pixel 9 Pro XL, x86_64, Android 16.0, API 36

CourierSampleApp

System Requirements

This document mandates adherence to the Android Application Development Environment Setup Guide, as a prerequisite for system compliance.

By following these steps, you should have a consistent and well-configured development environment for your Android application on Windows 10/11. Remember to restart your system after making changes related to CMake in Android Studio and after updating Environment Variables.



Project Structure

CourierSampleApp is using the Android Gradle Plugin Version 3.5.3 with Gradle Version 5.4.1. The Application is tested with ndk-26 that can be installed from the document in chapter Configuration and System Requirements. The supported API range is 24-28. The supported ABIs are arm64-v8a, x86_64, x86. The CourierSampleApp Android Application is NOT using the NativeView library. The build process is different than the Android Applications that use the NativeView library.

Building and Running the Android Application

To be able to build and run the Android Application, the following steps must be done.

1. CGI Studio

These steps are required to prepare the asset from the solution to use in the CMake and Android Studio steps.

1. Subst [windows-commands/subst] the root of CGI Studio.

- Open a Command Prompt in the root of CGI Studio directory e.g., “...\cgistudio” and type: “subst
  x: .” and click enter. The CGI root is now available under the subst "x" drive in Windows Explorer.

2. Run CGI Studio by double clicking on “X:\bin\SceneComposer\SceneComposer_Android_OpenGLES30.bat”,
3. Open the following solution for CourierSampleApp from X:\cgi_studio_courier_apps\src\CourierSampleApp\SCSolution\Android\Solution.scs”.
4. Save this solution as:
   1. AssetLibCff_CourierSampleApp.bin and then copy it into the /sdcard/ directory of the target/emulator. For the emulator this can be done in the "Device Explorer" of the emulator in Android Studio.
   2. AssetLibCff_Android_Target.bin and then copy it to “X:\cgi_studio_courier_apps\src\CourierSampleApp\Assets”. This one is needed to build the ".so" library required by the Android Project.

2. CMake

These steps are to prepare the ".so" library required by the Android application. Make sure the steps of "Alternative CMake Approach" have been done prior to the next step.

1. Open CMake and choose "X:\cmake\Apps\Courier\CourierSampleApp" as source code and e.g. X:\build for where to build the binaries. Click Configure.
2. Specify "Ninja" as the generator for this project and choose "Specify toolchain file for cross-compiling". Click Next.
3. Choose "X:\cgi_studio_devices\src\Android\ToolchainFiles\android.toolchain-ndk26-generic.cmake". Click Finish.
4. The first "error" will ask to set the "ANDROID_ABI", choose the ABI required. This should match the target/emulator CPU Architecture. Click Configure.
5. The next "error" will ask to set the ANDROID_PLATFORM. This should match the target/emulator Android version. Click Configure until there are no red lines anymore and then click on Generate.
6. Open CMD and navigate to "X:\build" by typing first "x:" and clicking enter, and then "cd X:\build" and clicking enter.
7. Type "ninja" and click enter for the build to start.
8. After the build is done, copy "X:\build\libApplicationLib.so" to the directory matching the ABI previously selected e.g. "X:\cgi_studio_apps\References\Android\SampleApp\app\native-libs\x86_64\libApplicationLib.so".

3. Android Studio

These steps are to prepare the Android Studio to build and run the Android application.

1. Open the Android Application in Android Studio from “X:\cgi_studio_apps\References\Android\SampleApp”.

2. Open the “local.properties.sample” file and change the “USERNAME” in the SDK and NDK path to match the
   system it will be used on.

local.properties.sample | D:\...\local.properties.sample

Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not work properly.

```
## COPY THE BELOW TO LOCAL.PROPERTIES
## MAKE SURE TO CHANGE USERNAME TO YOUR USERNAME.
## MAKE SURE THE PATHS EXIST, ELSE ANDROID STUDIO RESETS IT IMMEDIATELY.

sdk.dir=C\:\\Users\\USERNAME\\AppData\\Local\\Android\\Sdk
ndk.dir=C\:\\Users\\USERNAME\\AppData\\Local\\Android\\Sdk\\ndk-26
```

3. Copy the contents of the "local.properties.sample" to “local.properties”. Note that if the SDK or NDK paths do not exist, Android Studio will reset the “local.properties” again after trying to Sync Project with Gradle Files.
4. Make sure the JDK is set to 11 in [File | Settings | Build, Execution, Deployment | Build Tools | Gradle]

   like described in chapter Java Development Kit (JDK).
5. Choose the supported ['arm64-v8a', 'x86_64', 'x86'] ABI "mAbi" and set the application name "mApplicationName" in “build.gradle (Project: Module :app)”.

build.gradle

Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not work properly.

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'

def mAbi :String = 'x86_64' // choose from: 'arm64-v8a', 'x86_64', 'x86'
def mPackageName :String = "at.cgistudio.sampleapp"
def mApplicationName :String = "Name" // choose the application name
def mMinApi :Integer = 24
def mMaxApi :Integer = 28
def mVersionCode :Integer = 6
def mVersionName :String = "1.6"
def mFlavorDimension :String = "split"

android {
```

6. Run the Application by selecting the target/emulator and clicking on “run app”.
7. The first time the Application starts, it will request permissions. Click accept and restart the application. The solution should be running.

Troubleshooting

- If encountering any issues during the setup, refer to the official documentation for Android Studio,

CMake, and NDK for troubleshooting steps.
- If you encounter any issues linked to outdated source code, reach out to the Android developers for
  assistance in implementing a resolution.
- Check for updates regularly and keep your development environment components up to date.

Platform Tests

- Google Pixel C, ARM Cortex A57, Android 8.1.0, API 27
- Huawei MediaPad M5, ARM Cortex A73, Android 9, EMUI 9.1.0, API28
- Emulator Pixel 2, x86_64, Android 6, API 23 (Without Play/APIs)
- Emulator Medium Phone, x86_64, Android 9, API 28 (Without Play/APIs)
- Emulator Pixel 2, x86, Android 9, API 28 (Without Play/APIs)

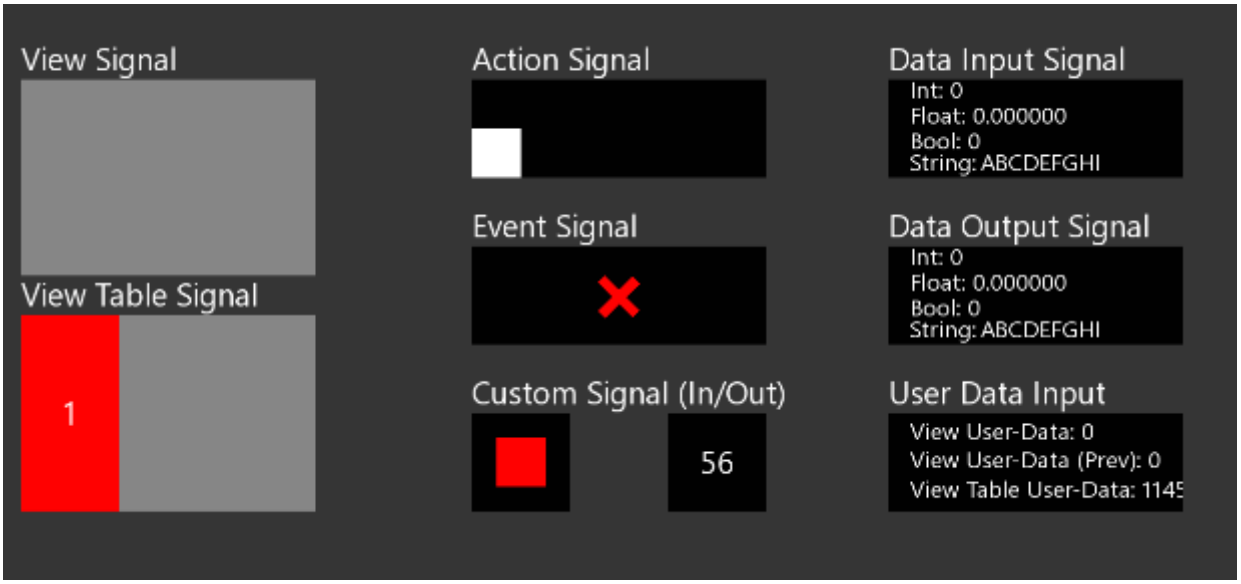
MLC

System Requirements

This document mandates adherence to the Android Application Development Environment Setup Guide, as a prerequisite for system compliance.

By following these steps, you should have a consistent and well-configured development environment for your Android application on Windows 10/11. Remember to restart your system after making changes related to CMake in Android Studio and after updating Environment Variables.

<Signals>



Project Structure

NativeViewLibrary is using the Android Gradle Plugin Version 3.5.3 with Gradle Version 5.4.1. The Application is tested with ndk-26 that can be installed from the document in chapter Configuration and System Requirements. The supported API range is 24-28. The supported ABIs are arm64-v8a, x86_64, x86. The MLC Android Application is NOT using the NativeView library. The build process is different than the Android Applications that use the NativeView library.

Building and Running the Android Application

To be able to build and run the Android Application, the following steps must be done.

1. CGI Studio

These steps are required to prepare the asset from the solution to use in the CMake and Android Studio steps.

1. Subst [windows-commands/subst] the root of CGI Studio.
   - Open a Command Prompt in the root of CGI Studio directory e.g., “...\cgistudio” and type: “subst
     x: .” and click enter. The CGI root is now available under the subst "x" drive in Windows Explorer.
2. Run CGI Studio by double clicking on “X:\bin\SceneComposer\SceneComposer_Android_OpenGLES30.bat”,
3. Open the following solution for MLC <Signals/Cluster> from X:\cgi_studio_matlab_connector\apps\sample\<Signals/Cluster>\src\SCSolution\Android\Solution.scs”.
4. Save this solution as:
   1. AssetLibCff.bin and then copy it into the /sdcard/ directory of the target/emulator. For the emulator this can be done in the "Device Explorer" of the emulator in Android Studio.
   2. AssetLibCff_Android_Target.bin and then copy it to “X:\cgi_studio_matlab_connector\apps\sample\<Signals/Cluster>\src\Asset”. This one is needed to build the ".so" library required by the Android Project.

2. CMake

These steps are to prepare the ".so" library required by the Android application. Make sure the steps of "Alternative CMake Approach" have been done prior to the next step. Note: Currently the ninja build can only succeed when the [cmake | CMAKE_BUILD_TYPE = Release] in CMake during configuration.

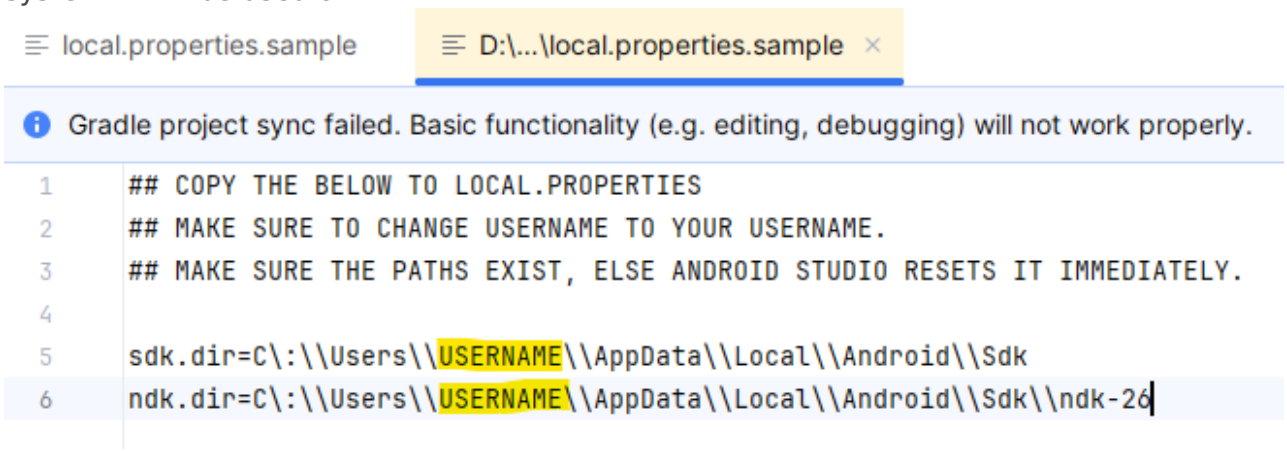
1. Open CMake and choose "X:\cgi_studio_matlab_connector\apps\sample\Signals\src\Application" as source code and e.g. X:/build for where to build the binaries. Click Configure.
2. Specify "Ninja" as the generator for this project and choose "Specify toolchain file for cross-compiling". Click Next.
3. Choose "X:\cgi_studio_devices\src\Android\ToolchainFiles\android.toolchain-ndk26-generic.cmake". Click Finish.
4. The first "error" will ask to set the "ANDROID_ABI", choose the ABI required. This should match the target/emulator CPU Architecture. Click Configure.
5. The next "error" will ask to set the ANDROID_PLATFORM. This should match the target/emulator Android version. Click Configure until there are no red lines anymore and then click on Generate.
6. Open CMD and navigate to "X:\build" by typing first "x:" and clicking enter, and then "cd X:\build" and clicking enter.
7. Type "ninja" and click enter for the build to start.

8. After the build is done, copy "X:\build\libApplicationLib.so" to the directory matching the ABI previously selected e.g. "X:\cgi_studio_apps\References\Android\SampleApp\app\native-libs\x86_64\libApplicationLib.so".

3. Android Studio

These steps are to prepare the Android Studio to build and run the Android application.

1. Open the Android Application in Android Studio from “X:\cgi_studio_apps\References\Android\SampleApp”.
2. Open the “local.properties.sample” file and change the “USERNAME” in the SDK and NDK path to match the system it will be used on.



3. Copy the contents of the "local.properties.sample" to “local.properties”. Note that if the SDK or NDK paths do not exist, Android Studio will reset the “local.properties” again after trying to Sync Project with Gradle Files.
4. Make sure the JDK is set to 11 in [File | Settings | Build, Execution, Deployment | Build Tools | Gradle]

   like described in chapter Java Development Kit (JDK).
5. Choose the supported ['arm64-v8a', 'x86_64', 'x86'] ABI "mAbi" and set the application name "mApplicationName" in “build.gradle (Project: Module :app)”.



6. Run the Application by selecting the target/emulator and clicking on “run app”.
7. The first time the Application starts, it will request permissions. Click accept and restart the application. The solution should be running.

Troubleshooting

- If encountering any issues during the setup, refer to the official documentation for Android Studio,
  CMake, and NDK for troubleshooting steps.
- If you encounter any issues linked to outdated source code, reach out to the Android developers for
  assistance in implementing a resolution.
- Check for updates regularly and keep your development environment components up to date.

Platform Tests

- <Signals> Emulator Medium Phone, x86_64, Android 9, API 28 (Without Play/APIs)