Android

Show Contents

Prebuilt libraries

The easiest way to get started is using the official prebuilt libraries (https://bintray.com/google/webrtc/google-webrtc) available at JCenter. These libraries are compiled from the tip-of-tree and are meant for development purposes only.

On Android Studio 3 add to your dependencies:

```
implementation 'org.webrtc:google-webrtc:1.0.+'
```

On Android Studio 2 add to your dependencies:

```
compile 'org.webrtc:google-webrtc:1.0.+'
```

The version of the library is `1.0.<Cr-Commit-Position>`. The hash of the commit can be found in the .pom-file. The third party licenses can be found in the THIRD_PARTY_LICENSES.md file next to the .aar-file.

Getting the Code

Android development is only supported on Linux.

1. Install prerequisite software (/native-code/development/prerequisite-sw/)
2. Create a working directory, enter it, and run:

   ```
   fetch --nohooks webrtc_android
   gclient sync
   ```

This will fetch a regular WebRTC checkout with the Android-specific parts added. Notice that the Android specific parts like the Android SDK and NDK are quite large (~8 GB), so the total checkout size will be about 16 GB. The same checkout can be used for both Linux and Android development since you can generate your Ninja (https://ninja-build.org/) project files in different directories for each build config.

See Development (/native-code/development/) for instructions on how to update the code, building etc.

Compiling

1. Generate projects using GN:

   Make sure your current working directory is src/ of your workspace. Then run:

   ```
   gn gen out/Debug --args='target_os="android" target_cpu="arm"'
   ```

   You can specify a directory of your own choice instead of out/Debug , to enable managing multiple configurations in parallel.

   - To build for ARM64: use `target_cpu="arm64"`
   - To build for 32-bit x86: use `target_cpu="x86"`
   - To build for 64-bit x64: use `target_cpu="x64"`

2. Compile using:

   ```
   ninja -C out/Debug
   ```

ABOUT
FAQ (/faq/)
Architecture (/architecture/)
Videos (/videos/)
Release Notes (/release-notes/)
Reference Apps (/reference-apps/)
License (/license/)
Press (/press/)

DEVELOPER
Contributing (/contributing/)
Bug Reporting (/bugs/)
Troubleshooting (/troubleshooting/)
Continuous Testing (/testing/continuous/)
Conformance Testing (/testing/conformance/)
Wireshark (/testing/wireshark/)

NATIVE CODE
Overview (/native-code/)
Development (/native-code/development/)
Prerequisites (/native-code/prerequisites/)
Android (/native-code/android/)
iOS (/native-code/ios/)
Native APIs (/native-code/native-apis/)

WEB APIS
Overview (/web-apis/)
Development (/web-apis/development/)
Standardization (/web-apis/standardization/)
Chrome (/web-apis/chrome/)
Firefox (/web-apis/firefox/)
Interop (/web-apis/interop/)

CONTACT
Google+ (https://plus.google.com/+webrtc/posts)
Twitter (https://twitter.com/webrtc)
discuss-webrtc (https://groups.google.com/group/discuss-webrtc)

Using the Bundled Android SDK/NDK

In order to use the Android SDK and NDK that is bundled in `third_party/android_tools` , run this to get it included in your `PATH` (from `src/`):

```
. build/android/envsetup.sh
```

Then you'll have `adb` and all the other Android tools in your `PATH` .

Running the AppRTCMobile App

AppRTCMobile is an Android application using WebRTC Native APIs via JNI (JNI wrapper is documented here (https://webrtc.googlesource.com/src/+/master/sdk/android/README)).

For instructions on how to build and run, see examples/androidapp/README (https://webrtc.googlesource.com/src/+/master/examples/androidapp/README).

Using Android Studio

1. Build the project normally (out/Debug should be the directory you used when generating the build files using GN):

   ```
   ninja -C out/Debug AppRTCMobile
   ```

2. Generate the project files:

```
build/android/gradle/generate_gradle.py --output-directory $PWD/out/Debug \
--target "//examples:AppRTCMobile" --use-gradle-process-resources \
--split-projects --canary
```

3. *Import* the project in Android Studio. (Do not just open it.) The project is located in `out/Debug/gradle`. If asked which SDK to use, choose to use Android Studio's SDK. When asked whether to use the Gradle wrapper, press "OK".
4. Ensure target `webrtc > examples > AppRTCMobile` is selected and press Run. AppRTCMobile should now start on the device.

If you do any changes to the C++ code, you have to compile the project using ninja after the changes (see step 1).

Note: Only "arm" is supported as the target_cpu when using Android Studio. This still allows you to run the application on 64-bit ARM devices. x86-based devices are not supported right now.

Running WebRTC Native Tests on an Android Device

To build APKs with the WebRTC native tests, follow these instructions.

1. Ensure you have an Android device set in Developer mode connected via USB.
2. Compile as described in the section above.
3. To see which tests are available: look in `out/Debug/bin`.
4. Run a test on your device:

```
out/Debug/bin/run_modules_unittests
```

5. If you want to limit to a subset of tests, use the `--gtest_filter flag`, e.g.

```
out/Debug/bin/run_modules_unittests \
--gtest_filter=RtpRtcpAPITest.SSRC:RtpRtcpRtcpTest.*
```

6. **NOTICE:** The first time you run a test, you must accept a dialog on the device!

If want to run Release builds instead; pass `is_debug=false` to GN (and preferably generate the projects files into a directory like `out/Release`). Then use the scripts generated in `out/Release/bin` instead.

Running WebRTC Instrumentation Tests on an Android Device

The instrumentation tests (like AppRTCMobileTest and libjingle_peerconnection_android_unittest) gets scripts generated in the same location as the native tests described in the previous section.