Android 10

CTS 测试操作指南

1.0

2020.02.26

文档履历

版本号	日期	制/修订人	内容描述
1.0	2020.02.26		Android 10 CTS 测试操作指南

目录

1. 测试环境搭建

1.1 获取 CTS 测试包

Google 在官方网站中提供了测试包的下载地址，从下面网址中获取：https://source.android.google.cn/compatibility/cts/downloads

注意: 目前 CTS 测试包会分 ARM 和 X86 2 个版本。这是针对不同的被测设备平台，而不是指测试主机的类型。我们的产品均基于 ARM 芯片，下载 ARM 版本即可。

1.2 Ubuntu 主机测试环境搭建

对测试主机的具体要求如下：

1. 安装 Ubuntu16.04 LTS 64bit 系统

2. 添加 adb 工具，配置系统使主机能够通过 adb 连接平板。

3. 安装 JDK1.8。

4. 安装 aapt 工具。

5. 安装 CTS 测试工具。将 1.1 节中获取的 zip 压缩文件解压到测试主机中，解压后得到名字为 android-cts 的文件夹，注意不要修改此文件夹及其子目录和文件的名字。

6.Android O 之后的测试包要求主机能连接外网。在更换新的 CTS 测试包的第一次测试一定要联网，否则无法跑起来，只要跑起来了一次，后面基于该版本测试工具的测试不一定要联网，但最好保持电脑端的 ipv6 的连接。注意：需要设置环境变量，使系统能够找到 adb，java 版本和 aapt 工具。

1.3 Windows 主机测试环境搭建

Windows 上运行 CTS 主要是方便在 Windows 环境下开发的同事对 CTS 进行调试。Windows 中的环境要求如下：

1. 添加 adb 工具，配置系统使主机能够通过 adb 连接平板。

2. 安装 JDK1.8。

3. 安装 aapt 工具。

4. 安装 CTS 测试工具。将 1.1 节中获取的 zip 压缩文件解压到测试主机中，解压后得到名字为 android-cts 的文件夹，注意不要修改此文件夹及其子目录和文件的名字。

注意：CTS 测试工具没有提供在 windows 下运行的脚本，这个脚本的实质是启动一个 java 程序。参照 CTS 工具中 shell 脚本的内容写了一个 BAT 批处理程序，用于在 Windows 环境下运行 CTS。如果出现无法正常运行 BAT 批处理程序的问题，请把 CTS 测试工具放在磁盘的根目录，文件夹命名不能太长。

1.4 网络环境

CTS 测试过程中会连接国外的网络如：youtube。使用国内的网络无法访问这些网站。需要使用能正常访问这些网站的网络环境进行测试。

测试过程会测试 ipv6 网络，所以 wifi 需要支持 ipv6。

2. 测试前平板配置

平板固件烧录完成后需要进行相关配置才能测试 CTS。进行 CTS 测试的设备都必须是安全机器，并已经烧写 google attestation key。主要的配置如下：

1. 恢复出厂设置。这一项在有必要的情况下进行。刚烧好固件运行起来的可跳过此步骤。如果烧录好固件后被用作其它用途再进行 CTS 测试，则需要先恢复出厂设置或重烧固件。

2. 在设置中选择语言为 English(United States)。

3.Settings > Display > Brightness 设置为最小。

4.Settings > Display > Sleep 设置最长休眠时间。

5. 选项 Settings > Security > Screen Lock 为 none，确保设备上未设置锁定图案或密码。

6. 选项 Setting> Wi-fi, 连接支持 ipv6 和能连接国外网络的网络。

7. 勾选 USB 调试。Settings > Developer options > USB debugging。(注意，在 4.2 之后的系统中 Developer options 默认是不显示的，需要进入 Settings > About tablet，然后迅速连续敲击 Build number 七次，返回上一级菜单查找开发者选项)。

8. 勾选 Settings > Developer options > Stay Awake，保持屏幕常亮。

9. 去掉勾选 Settings > Developer options > Verify apps over USB。

10. 打开浏览器，跳过浏览器设置界面。

11. 打开相机 app，跳过相机设置界面，使其在测试的时候能正常打开相机。

12. 拷贝多媒体文件，从 android 10 版本开始的 CTS 套件，可以不用手动拷贝多媒体文件，但测试指令要加上多媒体路径的参数，详细请查看 3. 启动 cts 测试部分章节。

（1）CTS 测试文件可以在如下网址中获取到，注意；从 Android6.0 开始使用 1.2 版本的媒体文件，现在 Android10.0 请使用 1.5 版本的媒体文件。多媒体文件包括视频和图片两种类型。
https://source.android.google.cn/compatibility/cts/downloads#cts-media-files

（2）Ubuntu 下直接执行 copy_media.sh 脚本即可将视频媒体文件拷贝到此电脑的平板。执行命令：
./copy_media.sh all

（3）Ubuntu 下直接执行 copy_images.sh 脚本即可将媒体文件拷贝到此电脑的平板。执行命令：./copy_images.sh

13. 连接能够上国外网络的 Wifi AP。

14. 将平板通过 USB 连接到测试主机。在 USB debugging 弹框中勾选 Always allow from this computer，点击 OK。

3. 启动 CTS 测试

3.1 CTS 完整测试

启动 cts 测试，需要在终端进入 1.2 步骤所解压的 android-cts 文件夹下的 tools 目录下，执行命令./cts-tradefed，会启动测试平台。注意，从 android 10 开始，cts-instant 相关的测试合并到 CTS 测试包中，故 android 10 之后 CTS 的完整测试包含了 cts-instant 相关的测试。

3.1.1 常用测试命令

Host	Description
help	Display a summary of the most commonly used commands
help all	Display the complete list of available commands
version	Show the version.
exit	Gracefully exit the CTS console. Console closes when all currently running tests are finished.
Run	Description
run cts	In Android 10, run the default CTS plan and CTS-Instant together (that is, the full CTS invocation). For Android 9 and lower, run the default CTS plan only. Use this comprehensive option (including preconditions) for device validation. See cts.xml for inclusions.The CTS console can accept other commands while tests are in progress.If no devices are connected, the CTS desktop machine (or host) will wait for a device to be connected before starting tests. If more than one device is connected, the CTS host will choose a device automatically.
run cts-instant	For Android 9, run the default CTS-Instant plan.

Host	Description
run cts --module-parameter INSTANT_APP	In Android 10, run the default CTS-Instant plan.
run cts --module-parameter INSTANT_APP --module/-m test_module_name	In Android 10, run the specified CTS-Instant test module or modules.
run retry	For Android 9 and higher only. Retry all the tests that failed or weren't executed from the previous sessions. For example, run retry --retry -s or run retry --retry --shard-count with TF sharding. run cts --retry isn't allowed for Android 9 and higher.
--device-token	For Android 8.1 and lower versions. Specifies that a given device has the given token. For example, --device-token 1a2b3c4d:sim-card.
--enable-token-sharding	For Android 10 only. Automatically matches the test that requires respective SIM type. No need to provide a device serial number to execute SIM-related test cases. Supported SIMs: SIM_CARD, UICC_SIM_CARD, and SECURE_ELEMENT_SIM_CARD.

Host	Description
run cts-dev	Run the default CTS plan (that is, the full CTS invocation) but skip preconditions to save run time for iterative development of a new test. This bypasses verification and setup of the device's configuration, such as pushing media files or checking for Wi-Fi connection, as is done when the --skip-preconditions option is used. This command also skips device-information collection, and all system status checkers. It also runs the tests on only a single ABI. For device validation, avoid this optimization and include all preconditions and checks. See cts-dev.xml for exclusions.The CTS console can accept other commands while tests are in progress.If no devices are connected, the CTS desktop machine (or host) will wait for a device to be connected before starting tests. If more than one device is connected, the CTS host will choose a device automatically.
run retry	**For Android 9**. Retry all tests that failed or were not executed from the previous sessions. For example, run retry --retry session id -sdevice serial, or run retry --retry session id --shard-count with TF sharding.run cts --retry is not allowed for Android 9.
--plan test_plan_name	Run the specified test plan.
--module/-m test_module_name [--module/-m test_module2...]	Run the specified test module or modules. For example, run cts --module CtsGestureTestCases executes the gesture test module (this can be shortened to run cts -m Gesture). run cts -m Gesture --test android.gesture.cts.GestureTest#testGetStrokes runs the specific package, class, or test.
--subplan subplan_name	Run the specified subplan.

Host	Description
-- module/-m test_module_name -- test test_name	Run the specified module and test. For example, run cts -m Gesture --test android.gesture.cts.GestureTest#testGetStrokes runs the specific package, class, or test.
--retry	Retry all tests that failed or were not executed from the previous sessions. Use list results to get the session id.
--retry-type not_executed	Retry only tests that were not executed from the previous sessions. Use list results to get the session id.
--shards number_of_shards	**For Android 8.1 and lower versions**. Shard a CTS run into given number of independent chunks, to run on multiple devices in parallel.
--shard-count number_of_shards	**For Android 9**. Shard a CTS run into given number of independent chunks, to run on multiple devices in parallel.
--serial/-s deviceID	Run CTS on the specific device.
--include-filter module_name [--include-filter module2...]	Run only with the specified modules.
--exclude-filter module_name [--exclude-filter module2...]	Exclude the specified modules from the run.
--log-level-display/-l log_level	Run with the minimum specified log level displayed to STDOUT. Valid values: [VERBOSE, DEBUG, INFO, WARN, ERROR, ASSERT].
--abi abi_name	Force the test to run on the given ABI, 32 or 64. By default CTS runs a test once for each ABI the device supports.
--logcat, --bugreport, and --screenshoot-on-failure	Give more visibility into failures and can help with diagnostics.
--device-token	Specifies a given device has the given token, such as --device-token 1a2b3c4d:sim-card.
--skip-device-info	Skips collection of information about the device.**Caution**: Don't use this option when running CTS for approval.

Host	Description
--skip-preconditions	Skip preconditions to save run time for iterative development of a new test. This bypasses verification and setup of the device's configuration, such as pushing media files or checking for Wi-Fi connection.
List	Description
list modules	List all available test modules in the repository.
list plans or list configs	List all available test plans (configs) in the repository.
list subplans	List all available subplans in the repository.
list invocations	List `run' commands currently being executed on devices.
list commands	List all `run' commands currently in the queue waiting to be assigned to devices.
list results	List CTS results currently stored in repository.
list devices	List currently connected devices and their state.`Available' devices are functioning, idle devices, available for running tests.`Unavailable' devices are devices visible via adb, but are not responding to adb commands and won't be allocated for tests.`Allocated' devices are devices currently running tests.
Dump	Description
dump logs	Dump the tradefed logs for all running invocations.
Add	Description
add subplan --name/-n subplan_name--result-type[pass | fail | timeout | notExecuted][--session/-s session_id]	Create a subplan derived from previous session; this option generates a subplan that can be used to run a subset of tests. The only required option is --session. Others are optional but, when included, must be followed by a value. The --result-type option is repeatable; for example add subplan --session 0 --result-type passed --result-type failed is valid.

3.1.2 完整测试指令

测试命令：run cts

前文提到，Q 版本后的 cts_instant 是包含在 CTS 测试中，即运行 cts，会自动执行 cts 和 cts_instant 的测试。若只想要测试 instant 相关的模块或者用例，可加一个参数，如 run cts --module-parameter INSTANT_APP，详情请查看上面的指令表格。

参数：

-s：平板序列号可通过"l d"（list device 的首字母缩写）命令查看

--logcat-on-failure：抓取 fail 项 log

--shard-count x：使用 x 台机器并行测试

--abi(-a) arm64-v8a 或者 --abi(-a) armeabi-v7a：指定测试 64/32 系统

--preconditions-arg skip-media-download：避免每次都在线下载多媒体文件

--module-arg TestModlue:local-media-path:/home/a/android-cts-media-1.5：指定本地多媒体文件的路径，其中 TestModlue 为 CtsMediaTestCases，CtsMediaStressTestCases，CtsMediaBitstreamsTestCases 三个，完整测试需要指定三个模块的多媒体路径。一般配合上面的去掉多媒体下载的指令使用。

比如：启动 CTS 32bit（armeabi-v7a）环境测试：

run cts --shard-count 2 -s < 平板序列号 1> -s < 平板序列号 2> -a armeabi-v7a --logcat-on-failure --preconditions-arg skip-media-download --module-arg CtsMediaTestCases:local-media-path:/ home/ user/ android-cts-media-1.5 --module-arg CtsMediaStressTestCases:local-media-path:/ home/ user/ android-cts-media-1.5 --module-arg CtsMediaBitstreamsTestCases:local-media-path:/home/user/android-cts-media-1.5

启动 CTS 64bit（arm64-v8a）环境测试：

run cts --shard-count 2 -s < 平板序列号 1> -s < 平板序列号 2> -a arm64-v8a --logcat-on-failure --preconditions-arg skip-media-download

3.2 CTS 补测

3.2.1 Retry 测试

除谷歌允许的 FAIL 外, 如果 CTS 测试 fail 项超过允许的 FAIL 项，要进行补测。测试命令：（android9.0 使用 run retry，而 android9.0 以前则继续使用 run cts）

run retry --retry <session_id> -s <serial>

session_id: 可通过"l r" 命令查看，比如通过如下命令启动补测：

run retry -r session_id --shard-count 2 -s 平板序列号 1 -s 平板序列号 2

3.2.2 创建补测测试计划以及启动 subplan 测试

Add: 可以通过 help add 命令查看帮助。

a/add s/subplan: create a subplan from a previous session

Options:

--session : The session used to create a subplan.

--name/-n : The name of the new subplan.

--result-type : Which results to include in the subplan. One of passed, failed, not_executed.Repeatable.

例：a s --session 2 --name 2 --result-type failed --result-type not_executed

通过如下命令启动补测：

run cts --subplan subplan_name，subplan_name 是上述中创建的名字

3.3 针对性测试

以下针对性测试，若含有 instant 部分，则会自动执行 cts 和 cts_instant 的测试，若只想单独调试 cts_instant 的模块，请加参数 --module-parameter INSTANT_APP。

可以针对某个测试包，测试类或者具体测试用例进行测试。如下图所示。

run <plan> --module/-m <module> --test/-t <test_name>: run a specific test from the module. Test name can be <package>.<class>, <package>.<class>#<method> or <native_name>. 例：

1．测试整个 module

run cts -m CtsVideoTestCases

2．测试整个 class

run cts -m CtsVideoTestCases -t android.video.cts.VideoEncoderDecoderTest

3. 测试一项 test

run cts -m CtsVideoTestCases -t android.video.cts.VideoEncoderDecoderTest#testAvcOther0Qual0320x0240

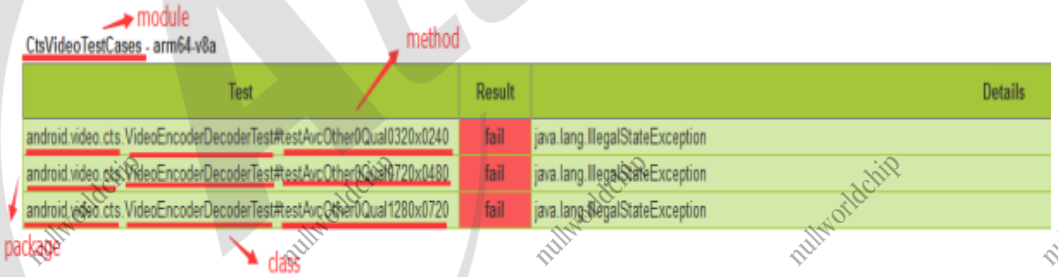

图 1: 测试项的 module, class, testcase 项分布

3.4 跳过某些有问题的测试项

比如测试的时候 CtsCameraTestCases 项目出现了问题，导致机器卡死或者测试中断，导致无法进行其他的项目的测试，这个时候可以选择跳过测试该项目，在执行的命令加上：--exclude-filter CtsCameraTestCases 或者 --exclude-filter "CtsCameraTestCases android.camera.cts.Takephotos"。

注意，跳过某个用例时，模块与用例名要用空格隔开，且要有双引号包住。其他与此类推，跳过多项时，重复输入参数。

3.5 循环测试

如需多次执行测试，无需等待正在执行的测试完成，直接输入多次测试命令即可，这些命令会被缓存起来，被依次调用。

4. 测试结果分析与调试

4.1 从 CTS 报告中获取出错信息

测试后的结果保存在 android-cts\results 目录下，可以通过浏览器打开 test_result_failures_suite.html 文件显示出测试的结果，在错误项的 Details 栏中给出了基本的错误信息。

测试过程抓取的 log 保存在 android-cts\logs 目录下。

4.2 查找相关 log 信息

测试时加上 --logcat-on-failure 参数，cts 工具会自动将 fail 的测试 log 保存在 android-cts\logs 下，打开 log 文件，搜索 TestRunner。测试过程的 log 以 ``TestRunner: started:…." 开始，以 ``TestRunner: finished: ….." 结束。

5. cts-on-gsi 测试

GSI 英文的全称为：generic system image，即通用系统镜像。从 Android O 开始，起谷歌引入了一个新的架构。这个新架构主要有这两点:

（1）原来的 system 分区切分成 system 分区与 vendor 分区。

（2）用 HIDL 沟通 system 分区与 vendor 分区。

面对这种改变，以后进行 GMS 认证需要进行 Treble Compliance Testing，其中有一项就是 cts-on-gsi。这就意味着 CTS 测试有两种：

（1）和之前一样的普通的 CTS 测试。

（2）刷入了谷歌提供的 GSI 的 system 镜像的 CTS 测试，即 cts-on-gsi。

所以 cts-on-gsi 也是 CTS 的一种，故放在本文档的第 5 章节。

5.1 环境搭建

5.1.1 获取测试包和 GSI

cts-on-gsi 需要 VTS 测试包工具和 GSI 系统包，这两个没有完全开放，仅 Android 合作伙伴可获取

5.1.2 Ubuntu 主机测试环境搭建

基本和前面的 1.2 的描述的环境配置要求一样，cts-on-gsi 是使用 VTS 测试包测试的，所以其实就是使用 VTS 测试环境进行测试，VTS 环境只比普通的 CTS 环境多了 python 环境。所以在 1.2 的基础上还需要做如下配置：

1、主机需要连接可以访问国外网站的网络。

2、配置 python 环境

（1）安装 Python 开发包：sudo apt-get install python-dev

（2）安装 Protocol Buffer 工具

sudo apt-get install python-protobuf

sudo apt-get install protobuf-compiler

（3）安装 Python 虚拟环境相关工具

sudo apt-get install python-virtualenv

sudo apt-get install python-pip

3、安装 google-api-python-client（非必须）

有了以上环境，vts 测试的基本条件就满足了，但是在实际测试时，会提示无法从网络获取 google-api-python-client，在终端输入如下命令即可下载安装：$ pip install --upgrade google-api-python-client

4、添加 fastboot 工具，通过 fastboot 工具烧写 GSI。

5.1.3 Windows 主机测试环境搭建

cts-on-gsi 不建议在 Windows 环境进行完整测试，在 Windows 上进行调试可以复用 1.3 所述的环境。使用 CTS 的环境可以调试 cts-on-gsi 的单个 fail。因为普通的 CTS 的测试用例都包含了 cts-on-gsi 里面的测试用例。

5.2 测试前平板配置

5.2.1 选择 GSI 镜像

选择合适的 GSI 进行烧写，GSI 包的名字后缀一般是安全补丁的编号或者日期。与 ARM 架构相关的 GSI 包有 arm64 和、arm32 两种 ABI。目前 Google 提供三种 GSI 包，分别是 O Update 10、P Update 10、10 launch devices。这里以出厂为 Android10 系统的情况举例。如下图所示，aosp_arm_img-5956348.zip 是 google 提供的 gsi，解压后分别得到如图中的几个文件。

图 2: gsi 镜像以及解压出来的文件

假如目前设备的安全补丁日期是 20191105 的，则选择当月提供的 gsi 包，否则会出现刷 gsi 启动异常。Android 10 烧写 GSI 不再需要烧写 vbmeta。

安全补丁日期可以在 About tablet 中查看。即 GSI 固件需要选取和当前 sdk 安全补丁日期一样的 img。选择 arm（即 arm32）。

5.2.2 烧写 GSI 步骤

1、进入 android 界面 ``设置 -> 系统 -> 开发者选项"，点选 oem 解锁选项（首次解锁设备需要点选，解锁后可以忽略该步骤以及第 3 步）

2.、让设备进入 bootloader 模式，在 adb 控制台输入：adb reboot bootloader

3、设备解锁：在 bootloader 控制台输入：fastboot oem unlock，此指令解锁设备，设备只要解锁过一次，只要没主动上锁，以后不用每次都要解锁。

然后进入 fastboot 模式，fastboot reboot fastboot（这条指令是紧接着上一步，若不需要 oem 解锁，这直接用 adb reboot fastboot 命令进入 fastboot 模式）

4、fastboot 控制台输入刷录 GSI 指令，例如：fastboot flash system system.img (解压 gsi 包后得到的 system.img)，此指令表示将 system.img 镜像烧写到 system 分区。

5、等待刷录镜像完成，在 fastboot 控制台进入 bootloader 模式，擦除用户数据：fastboot reboot bootloader，fastboot -w

6、输入 fastboot reboot 让系统重启，即可重启系统。

流程如下：oem 解锁 -->> 进入 bootloader 模式 -->> 设备解锁 -->> 进入 fastboot 模式 -->> 刷录镜像 -->> 进入 bootloader 模式，清除缓存 -->> 重启系统

5.2.3 平板设置

请参考前文的第 2 章测试前平板设置，操作一样。

5.3 启动 cts-on-gsi 测试

同样 3.1 章节，需要在终端进入 tools 目录，注意，cts-on-gsi 是用 VTS 套件，启动测试平台应该是./vts-tradefed

常用测试命令介绍可参考 3.1.1 章节。cts-on-gsi 仅仅会启动主 ABI 的测试。

测试命令：run cts-on-gsi

参数：

-s 平板序列号：平板序列号可通过"l d"（list device 的首字母缩写）命令查看

--logcat-on-failure ： 抓取 fail 项 log

比如：

启动 cts-on-gsi 测试：

run cts-on-gsi --shard-count 2 -s 平板序列号 1 -s 平板序列号 2 --logcat-on-failure --precondition-arg skip-media-download

其他请参考前面第 3 节启动 CTS 测试章节，包括指定多媒体的参数、补测等基本是一样的，这里不作重复。

只要把相关的 run cts 改为 run cts-on-gsi。

VTS 测试包环境的操作和 CTS 环境包环境的操作是一样的。以下内容也基本一致，请参考以上 CTS 的内容。

5.3.1 cts-on-gsi 补测

除谷歌允许的 FAIL 外, 如果 CTS 测试 fail 项超过允许的 FAIL 项，要进行补测。

测试命令：

android9.0 使用 run cts-on-gsi-retry，而 android10.0 则使用 run retry --retry <session_id> 的指令。

session_id: 可通过"l r" 命令查看

比如通过如下命令启动补测：

run retry -r session_id --shard-count 2 -s 平板序列号 1 -s 平板序列号 2

6. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner. The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This document neither states nor implies warranty of any kind, including fitness for any particular application.tates nor implies warranty of any kind, including fitness for any particular application.