cts测试套件的环境搭建及使用

1. Android Compatibility

Android Compatibility又名Android兼容性计划，主要包括三个部分：Android源代码，Android兼容性定义文档（CDD文档），兼容性测试套件（CTS）。

* Android源代码：Android开发源码，进行修改开发；
* CDD：[Compatibility Definition Document](https://source.android.com/compatibility/android-cdd)，定义一系列Android设备的要求，也包括很多其他硬件兼软件时间方面的要求；
* CTS：提供限制，保证开发的应用在所有兼容Android设备上正常运行。

1. 基本介绍

CTS是一个自动化测试工具，也是一套单元测试，它运行在桌面机器上，并直接在附加设备或模拟器上执行测试用例。它的目的是在集成到工程师构建设备的日常工作流程（例如通过连续构建系统）中。其目的是尽早发现不兼容性，并确保软件在整个开发过程中保持兼容性，保证开发的应用在所有兼容Android设备上正常运行。

注：具体的使用方法详见：<https://source.android.google.cn/compatibility/cts>

1. 工作流程



1. 基本原理

基本原理就是利用测试APK调用对应的系统接口，保证系统对应的接口按照谷歌的要求返回对应的值。

1. 作用

通过CTS 测试，确保了APP与Android Framework之间有一致的调用接口（API），这使得APP开发者编写的同一款程序可以运行在不同系统版本（向前兼容）、不同硬件平台、不同厂商制造的不同设备上。



1. 基本组成

* CTS tradefed自动化测试框架会在桌面设备上运行，并管理测试执行情况；
* 单独的测试用例会在被测设备 (DUT) 上执行。测试用例采用 Java 语言编写为 JUnit 测试，并打包为 Android .apk文件，以在实际目标设备上运行。（一个apk文件就是一个module，而一个module就代表一组测试case。V1是以plan为测试方式，V2版本框架以module为核心）。

1. 测试类型
2. 从case类型上划分

* 单元测试：即基本API测试，比如HashMap，ArrayList等单个类。
* 组合API测试：测试一系列组合API，比如四大组件的使用。
* 稳定性测试：测试系统在压力下的耐久性（未来版本）。
* 性能测试：根据定义的基准测试系统的性能，例如每秒渲染帧数（未来版本）。

1. 从执行测试的方式上划分

* PC端跑plan：这种测试是通过在PC端通过命令行的方式组织所有的测试case，然后在车机上执行，最终收集所有的测试结果。
* CtsVerify：这种测试是为了弥补上面无法测试到的类型，是安装一个apk到被测试的设备上，需要人为输入，比如GPS，传感器等的测试。（CTS vertify应用，直接安装到手机上，主要用于硬件及CTS测试套件难以测试的一些偏功能性的接口测试，如：Camera、GPS和各种Sensor等）

其中单元测试覆盖领域：

* 签名测试：每个 Android 版本中都包含一个 XML 文件，用于描述这一版本所含的所有公开 API 的签名。CTS 包含一个实用工具，用于根据设备上可用的 API 检查这些 API 签名。签名检查的结果会记录在测试结果 XML 文件中
* 平台API测试：按照 SDK[类索引](https://developer.android.google.cn/reference/classes)所述内容来测试平台（核心库和 Android 应用框架）的 API，以确保 API 的正确性，包括正确的类、属性、方法签名以及正确的方法行为；此外还需进行负面测试，以确保不正确的参数处理产生预期行为
* DALVIK虚拟机测试：这类测试侧重于测试 Dalvik 可执行格式的文件
* 平台数据模型测试：CTS 会测试通过内容提供程序（如 SDK [android.provider](https://developer.android.google.cn/reference/android/provider/package-summary) 软件包中所述）提供给应用开发者的核心平台数据模型：通讯录、浏览器、设置等。
* 平台Intents测试：CTS 会测试核心平台 Intent
* 平台权限测试：CTS 会测试核心平台权限
* 平台资源测试：CTS 会测试核心平台资源类型（如 SDK [可用资源类型](https://developer.android.google.cn/guide/topics/resources/available-resources)中所述）的处理是否正确。这包括对以下资源的测试：简单值、可绘制资源、九宫格、动画、布局、样式和主题背景，以及加载备用资源。

 注意：CTS 目前支持 64 位 Linux 和 Mac OS 主机。CTS 无法在Windows 操作系统上运行。在虚拟机和ubuntu下都测试通过，但是虚拟机上无法在终端显示正在测试情况，会打印出null的信息，但是在纯linux环境ubuntu下是正常的。

1. CTS源码结构

cts // Android兼容性测试套件标准

apps // cts测试所使用的apk源码

   ├─CtsVerifier

       ├─jni //mk,cpp C++代码

       ├─res //项目资源放置并且编译应用程序的地方

       ├─src //有关verifier的源码

       ├─tests //src目录对应的测试用例

build // cts套件的编译配置

common // cts套件的通用代码（包括设备端的检查锁屏、外设、wifi、phone，）

development // cts套件兼容之前使用eclipseADT开发apk的脚本

hostsidetests // cts套件新增host测试项

libs // cts套件使用的第三方库

   ~~├─ Annotation  用于指定哪些特性需要执行测试用例，为了能执行所有的应用设备都要有特定的注解。~~

├─ deviceutillegacy

├─ json  json库文件

├─ runner   包含Makefile的文件来运行tests/core/runner目录下的核心测试

├─ view

├─ testserver  http测试服务

├─ vogar-expect  是Java的命令行测试实用工具

├─ wrappedgtest

suite // cts套件

tests // cts套件进行的测试项，这个目录包含Java测试用例，测试不同类型的东西，主要是从Java应用程序中获得的特性。

tools // cts套件使用的测试工具源码 ，此目录包含可供您使用的不同类型的工具。

1. 关于编译
2. 编译步骤

* Android源码编译完成后，可以编译CTS
* source ./build/envsetup.sh
* lunch（选择33） mek\_8q-eng
* make cts

注：编译可能存在的问题：

./tests/tests/telecom2/res/.gitignore路径下缺少文件，但是该文件是空文件，添加之后可以编译通过

1. 编译好cts后生成的文件位置

CTS测试套件涉及到的目录有：

* Package CTS : out/host/linux-x86/cts/android-cts.zip
* Test packages : mydroid/out/host/linux-x86/cts/android-cts/testcases
* Test tools: mydroid/out/host/linux-x86/cts/android-cts/tools
* Test results : mydroid/out/host/linux-x86/cts/android-cts/results
* Cts make file : mydroid/build/core/tasks/cts.mk

注：

* 如果需要提交Google确认版本测试结果，我们采取下载官方CTS测试包进行测试并提交测试结果，不会采取源码编译的方法。（测试结果提交方只承认官方下载的测试包测试出来的结果）
* 下载地址：<https://source.android.google.cn/compatibility/cts/downloads>
* 现在的测试版本为：android-cts-9.0\_r7-linux\_x86-arm

1. Android设备配置
2. 将设备恢复出厂设置：设置 > 备份和重置 > 恢复出厂设置

**警告：**这将清空设备中的所有用户数据。

1. 将设备的语言设置为英语（**美国**）：**设置 > 语言和输入法 > 语言**
2. 如果设备具有 GPS 或 WLAN/移动网络功能，则打开位置信息设置：**设置 > 位置信息 > 开启**
3. 连接到满足以下要求的 WLAN 网络：支持 IPv6、可以将被测设备 (DUT) 视为隔离客户端（请参阅上文的[物理环境](https://source.android.google.cn/compatibility/cts/setup#physical_environment)部分），并可连接到互联网。连接网络的具体操作方法为：**设置 > WLAN**
4. 确保设备上未设置锁定图案或密码：**设置 > 安全 > 屏幕锁定 > 无**
5. 在设备上启用 **USB 调试**：**设置 > 开发者选项 > USB 调试**。
6. **注意：**在 Android 4.2 及更高版本中，默认情况下会隐藏**开发者选项**。要显示这些选项，请依次转到**设置 > 关于手机**，然后点按**版本号**七次。返回上一屏幕以查找**开发者选项**。要查看其他详细信息，请参阅[启用设备上的开发者选项](http://developer.android.google.cn/studio/run/device.html#developer-device-options)。
7. 确保将时间设置为 12 小时格式：**设置 > 日期和时间 > 使用 24 小时制 > 关闭**
8. 依次选择：**设置 > 开发者选项 > 不锁定屏幕 > 开启**
9. 依次选择：**设置 > 开发者选项 > 允许模拟位置 > 开启**

**注意：**此模拟位置设置仅适用于 Android 5.x 和 4.4.x。

1. 依次选择：**设置 > 开发者选项 > 通过 USB 验证应用 > 关闭**

**注意：**此验证应用步骤在 Android 4.2 中为必需步骤。

1. 启动浏览器并关闭任何启动/设置屏幕。
2. 使用 USB 数据线连接用于测试设备的台式机

**注意：**将运行 Android 4.2.2 或更高版本的设备连接到计算机时，系统会显示一个对话框，询问您是否接受允许通过此计算机进行调试的 RSA 密钥。选择“允许 USB 调试”。

在设备上安装和配置帮助程序应用。

1. **注意：**对于 CTS 2.1 R2 - 4.2 R4 的版本，请通过以下命令设置您的设备（或模拟器），以便执行无障碍测试：  
   adb install -r android-cts/repository/testcases/CtsDelegatingAccessibilityService.apk  
   在设备上，依次启用：**设置 > 无障碍 > 无障碍 > Delegating Accessibility Service**

**注意**：对于 7.0 之前的 CTS 版本，请在声明 android.software.device\_admin 的设备上设置您的设备，以使用以下命令执行设备管理测试：  
adb install -r android-cts/repository/testcases/CtsDeviceAdmin.apk

依次点击“设置”>“安全”>“选择设备管理器”，然后启用两个 android.deviceadmin.cts.CtsDeviceAdminReceiver\* 设备管理器。确保 android.deviceadmin.cts.CtsDeviceAdminDeactivatedReceiver 和任何其他预加载的设备管理器均保持停用状态。

1. 将 CTS 媒体文件复制到设备上，如下所示：

**注意：**对于 CTS 2.3 R12 及更高版本，如果设备支持视频编解码器，则必须将 CTS 媒体文件复制到设备上。

导航 (cd) 到下载并解压缩媒体文件的目标路径。

更改文件权限：chmod u+x copy\_media.sh

运行 copy\_media.sh：

要复制分辨率不超过 720x480 的剪辑，请运行：./copy\_media.sh 720x480

如果您不确定最大分辨率，请尝试运行 ./copy\_media.sh all，以便复制所有文件。

如果 adb 下有多个设备，请将 -s（序列号）选项添加到末尾。例如，要将分辨率不超过 720x480 的文件复制到序列号为 1234567 的设备，请运行：./copy\_media.sh 720x480 -s 1234567

1. 测试计划分类

CTS V2版本（Android 7.0或更高版本）

1. 按模块分类

* CtsBluetoothTestCases
* CtsMediaTestCases
* CtsMediaStressTestCases
* CtsNdefTestCases
* CtsPermissionTestCases
* ……（分类比较多就不一一写出了）

1. 按计划分类

* cts - 从预装的 CTS 套件运行 CTS。
* cts-camera - 从预装的 CTS 套件运行 CTS-camera。
* cts-java - 从预装的 CTS 套件运行核心 Java 测试。
* cts-pdk - 运行有助于验证与 PDK 融合的产品的测试。
* everything - 兼容性套件的通用配置。
* ……

1. 测试流程

* Linux环境的准备（ADB、AAPT、JDK等）：apt install aapt、apt install adb；

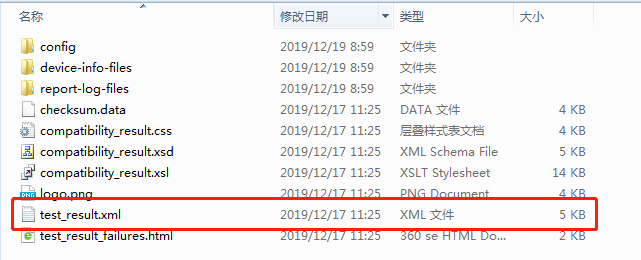
注：此安装方式适用于Ubuntu16+，在Ubuntu14上有问题

* 使用adb连接车机，执行adb remount 赋予执行权限，使用adb devices查看并确认待测设备状态；
* 对车机环境进行详细设置准备，按照本文中十Android设备配置章节步骤执行，在整体运行时该项要求是必须的；
* 对于整体cts运行，可先将需要使用到的Media文件按需求push到车机中；
* 运行cts脚本./cts-tradefed，进入控制台
* 输入控制台命令，如：run cts -m CtsSampleDeviceTestCases --skip-preconditions

注：该命令是运行Sample模块的测试，具体可参考官网中的执行命令来执行测试需求

1. 测试结果

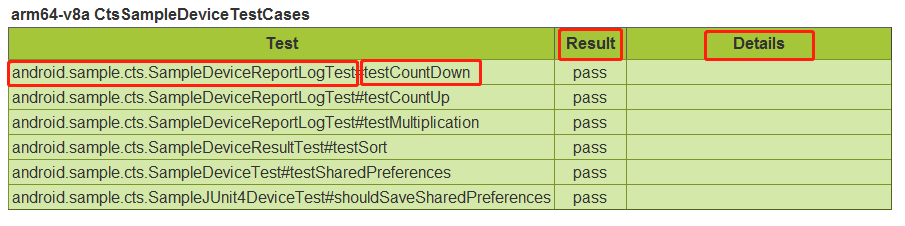
* 测试任务执行完成后，会在./android-cts/results文件夹中生成对应的测试结果及./android-cts/log文件夹中有对应的日志文件，
* results文件夹中的test\_result.xml文件包含实际结果



* 打开test\_result.xml文件可以查看已执行的测试计划的详细信息，如：CTS计划名称、执行开始和结束时间，提供通过、失败、超时或无法执行的测试数量的汇总摘要，如下图所示：



* 也可以查看到所执行实际测试的详细信息，包括测试包、测试套件、测试用例和执行的测试结果：通过、失败、超时或未执行，如果测试失败会提供详细信息以供诊断原因，如下图所示：



注意：在测试执行过程中，按<ctrl+C>会终止CTS进行，但测试报告无法生成，以后也无法继续测试未完成的计划。

1. 错误分析方法

1、确认运行failed项是Google本身的原因还是需要我们分析修复的；

2、查看log文件的相关信息，配合查看CTS源码进行分析（也可以采用在测试case中增加LOG信息，再自行编译用来测试执行）

1. 测试用例开发（CTS v2版本）
2. 目录结构

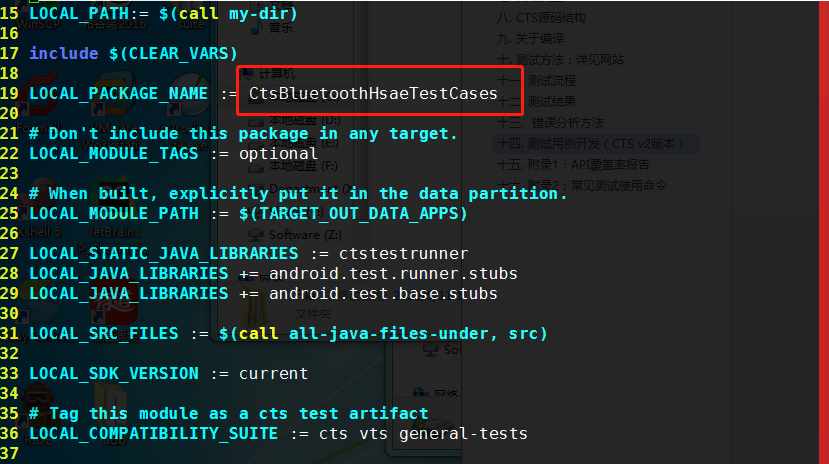


1. 构建新的测试模块
   * + 运行以下命令创建测试目录并将示例文件复制到该目录中，mkdir cts/tests/module-name && cp -r cts/tests/sample/\* cts/tests/module-name
     + 转到 cts/tests/module-name，然后按照上文建议的命名惯例替换所有出现的“[Ss]ample”。
     + 更新 SampleDeviceActivity 以运行您需要测试的功能。
     + 更新 SampleDeviceTest 以确保该 Activity 成功或记录其错误。
     + 可以添加其他 Android 目录，如 assets、jni、libs 和 res（“窗口控件类”中使用的的资源文件）。如需添加 JNI 代码，请在项目根目录下的 src 旁创建一个目录，并将原生代码和 Android.mk Makefile 放入该目录中
2. 案例流程举例（bluetooth模块）

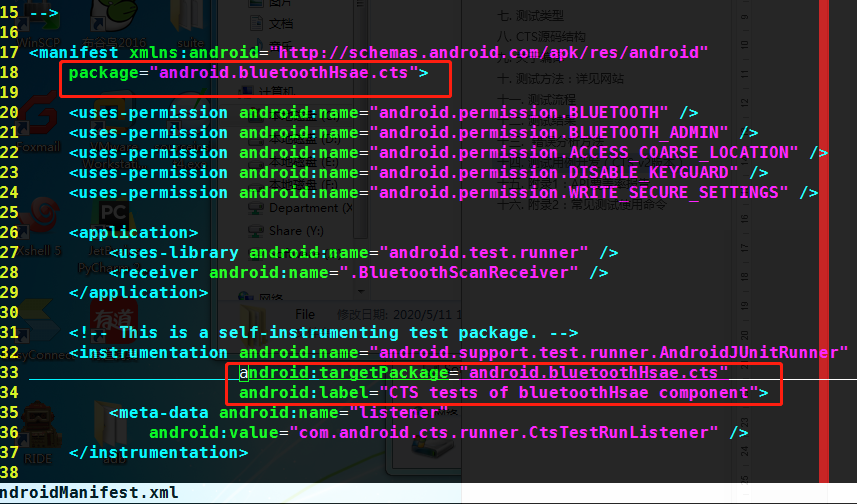
注：1、经过分析后发现CTS测试源码中无蓝牙搜索的测试case，我们可以增加这部分的case，蓝牙搜索相关的API：startDiscovery()、cancelDistory()、isDiscovery()

2、不建议在原代码文件夹内直接修改源码文件，可以在当前路径下添加文件夹，以下执行流程按照此思路进行

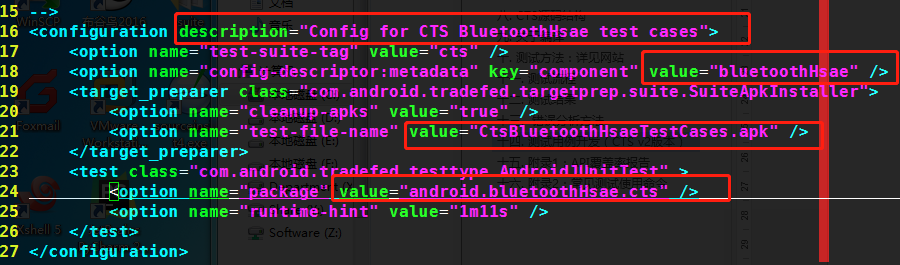
* + - cp -r cts/tests/tests/ bluetooth/\* cts/tests/ bluetoothHsae/
    - 进入bluetoothHsae目录下，修改Android.mk文件



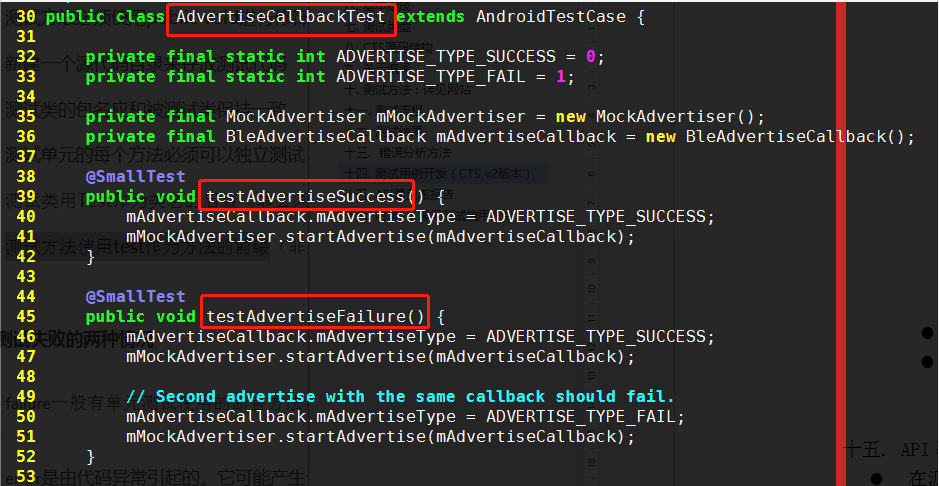
* + - 进入bluetoothHsae目录下，修改AndroidManifest.xml文件



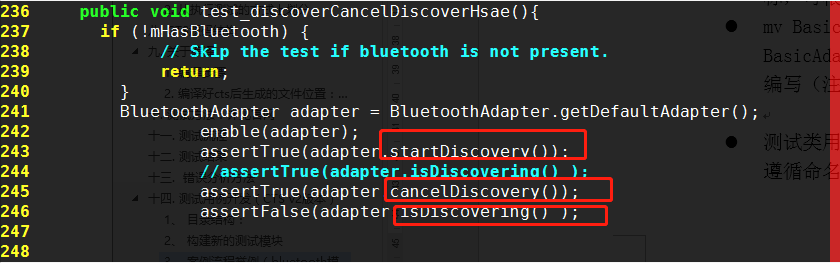
* + - 进入bluetoothHsae目录下，修改AndroidTest.xml文件



* + - mv ./src/android/bluetooth ./src/android/ bluetoothHsae，修改文件名称，可根据需求设计；
    - mv BasicAdapterTest.java BasicAdapterHsaeTest.java，接下来可以在BasicAdapterHsaeTest.java文件中增加对需要测试的API进行测试用例的编写（注意同步package的名称）；
    - 测试类用Test作为类名的后缀，测试方法使用test作为方法的前缀，其余遵循命名惯例
* 测试用例开始前的资源准备和预先设置：Setup()方法；
* 测试用例步骤执行及验证点检验：Assert()方法；
* 测试用例结束后的环境清理：Teardown()方法；



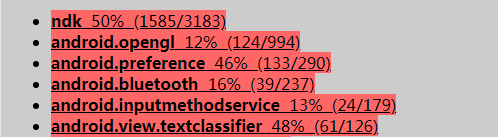
* + - 完成对startDiscovery()、cancelDistory()、isDiscovery()这三个API的测试用例的编写，如下：

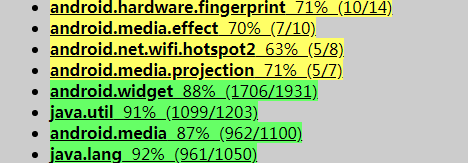


* + - 重复编译操作，详见本文“关于编译”章节内容；
    - 编译好的cts测试文件在Android/out/host/linux-x86/cts/android-cts文件夹中会包含CtsBluetoothHsaeTestCases.apk测试包，即可进行测试执行操作
    - 运行测试case，查看结果是否如预期。

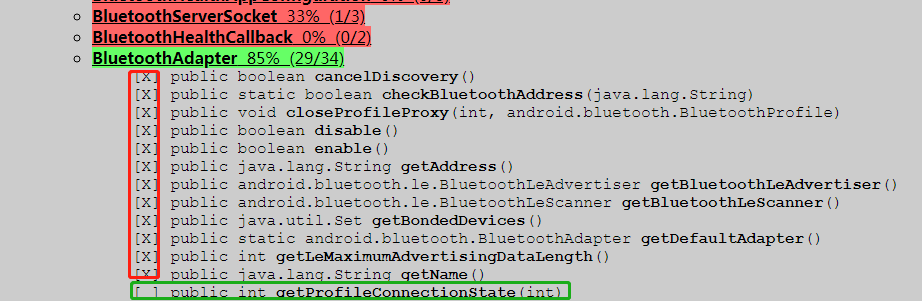
1. API覆盖率报告

* 在源码根目录：make cts-test-coverage；
* 编译完成后生成的报告：/out/host/linux-x86/cts-api-coverage/test-coverage.html；
* 用浏览器打开test-coverage.html文件，覆盖率50%以下的包是红色，覆盖率50%～80%的包是黄色，覆盖率80%以上的是绿色；





* 点开依次点开“anoid.bluetooth”—>“BluetoothAdapter”可查看各包的详细覆盖情况。查看到前面有X的就是已经在CTS的case中覆盖了的API，可根据需要对一些CTS中未覆盖的API进行测试，或者如有对framework有修改的话，可以写针对性的用例然后生成覆盖率报告看是否有遗漏；



1. 附录1：运行中的一些点
2. 采用run cts命令运行整个cts计划中的所有测试case，现发现运行一段时间后测试无输出信息打印，测试出现假死现象，后排查出该问题只出现在VMware虚拟机中，在纯Linux系统下运行该现象未复现，建议测试PC为纯Linux操作系统；
3. 目前计划采用的方式是整体执行，对failed的模块逐个执行的策略。
4. 每次执行CTS时会自动搜索历史报告文件，如果文件多，启动速度就越来越慢，可以将results文件夹中的一些测试结果删除或转移，提高运行速度
5. 其实是V1和V2的很重要的一点区别就是在这里：

* v1版本的plan文件相当于是一个集合，需要执行哪些测试case呢，就把需要执行的测试case添加到plan中，最后把这个集合中的测试case拿出来执行即可。
* v2版本则不然，它默认就把所有的测试case给全部拿到并执行，除非配置了不需要执行哪些case，否则的话默认执行全部的case。

1. 附录2：常见测试使用命令
2. list invocations 列出设备上当前正在执行的“运行”命令
3. l r 命令查看当前正在运行的所有session （list results 查看测试结果）
4. l c 查看当前列队中正在执行的命令（list command）
5. 整体运行cts计划，运行命令为run cts ~~–plan CTS（现阶段运行发现执行一段时间没有输出）~~
6. （根据cts运行原理执行plan的时候会跑全部的case）
7. 运行某个模块的执行命令：run cts -- module/-m <test\_module\_name>
8. 运行指定的模块和测试命令-m <test\_module\_name> -- test <test\_name>，例如：run cts -m CtsAlarmClockTestCases --test android.alarmclock.cts.DismissAlarmTest#testAllExtent
9. 在特定设备上运行命令：--serial/-s <deviceID>

* 如果需要运行测试结果failed的测试项，可以使用l r 先查看session\_ID再使用run cts –retry [session\_ID]（Android 8）
* run retry --retry [session-id] （Android 9）重新尝试运行在以前会话中失败或未执行的所有测试

1. --skip-device-info，跳过收集设备相关信息的过程（建议使用，不然运行时间长）。注意：运行 CTS 以寻求批准时，请勿使用此选项。
2. --skip-preconditions，绕过对设备配置的验证和设置，例如推送媒体文件或检查 Wi-Fi 连接（建议使用，不然运行时间长且容易验证失败）
3. --disable-reboot，避免车机、手机在运行过程中重启（建议使用）
4. --exclude-filter CtsDevicePolicyManagerTestCases,在执行过程中会使得车机~~重启~~adb无法识别到设备，故目前在整体运行过程中不运行次模块
5. --abi <abi\_name>强制要求测试在给定的 ABI（32 或 64）上运行。默认情况下，CTS 会为设备支持的每个 ABI 运行一次测试，可以使用adb shell getprop ro.product.cpu.abi命令查看到具体的ABI。

* armeabiv-v7a: 第7代及以上的 ARM 处理器。2011年15月以后的生产的大部分Android设备都使用它.
* arm64-v8a: 第8代、64位ARM处理器，很少设备，三星 Galaxy S6是其中之一。
* armeabi: 第5代、第6代的ARM处理器，早期的手机用的比较多。

1. 注意：您可以通过使用 run cts-dev 命令（而非 run cts）来缩减在 Android 7.0 (Nougat) 及更高版本中的运行时间。此命令会跳过前提条件、设备信息收集和所有系统状态检查工具。它还仅在单个 ABI 上运行测试。对于设备验证，请忽略此优化操作并添加所有前提条件和检查。
2. **通用执行命令**：run cts -s 1016a80e82899a18 --abi arm64-v8a -m CtsBluetoothHsaeTestCases --skip-device-info --skip-preconditions --disable-reboot
3. 附录3：ACTS测试

Android Connectivity Testing Suite (ACTS) ，Android 框架 API 和芯片组认证之间的测试，可验证 Android 框架所用的蓝牙、WLAN 和移动网络各方面的功能。因为ACTS 测试利用特权 Android API 解锁原本不可用的更深层次的测试所以只能使用 ACTS 来测试工程和 userdebug 版本