**研发二部**

**测试团队**

测试框架说明书

框架名称 <手机应用测试框架>

文档类别 <测试框架>

文档编号 <TestFramework\_Android\_IOS\_V2.0>

编 写 人 <唐晓文>

版 本 <V2.0>

密 级 <高>

修改时间 <2014-05-29>

版本修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 日期 | 描 述 | 作者 | 审核 |
| V1.0 | 2013-07-25 | 创建文档 | 唐晓文 |  |
| V1.1 | 2013-09-27 | 补充完成文档 | 唐晓文 |  |
| V1.2 | 2014-02-20 | 新增“接口测试”部分 | 唐晓文 |  |
| V2.0 | 2014-05-29 | 新增IOS应用测试相关知识 | 唐晓文 |  |

## 一、框架简介

随着智能手机应用开发的普及，手机应用的范围和用户数不断增大，对手机应用产品质量要求越来越高。为此，我们对手机应用相关技术及测试用例进行研究，整理出智能手机应用测试框架（主要是基于Android和IOS），以便测试人员能很快地熟悉相关测试技术，使产品达到基本的质量要求。

## 二、测试体系框架图

整个框架由测试方法和测试工具2类对象组成。

按照测试过程中使用频率高低划分为4类：

红色——高、黄色——中、蓝色——低、绿色——工具

## 三、框架测试方法

### 1、安装卸载更新测试

#### 1.1应用安装测试

##### 1.1.1Android应用安装测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | Android应用安装测试。 | **整理人员** | 唐晓文 |
| **简介** | 安装是Android应用是使用的前提。 | | |
| **详细技术说明** | Android应用程序打包后会生成一个apk文件（安装包）。  **一个合法的apk安装包至少包括以下成分：**  1、根目录下的“AndroidManifest.xml”文件，用以向Android系统声明所需Android权限等运行应用所需的条件。  2、根目录下的classes.dex（dex指Dalvik Exceptionable），应用（application）本身的可执行文件（Dalvik字节码）。  3、根目录下的res目录，包含应用的界面设定。（如果仅是一个后台执行的“service”对象，则不必需）  4、Apk根目录下的META-INF目录也是必须的，它用以存放应用作者的公钥证书与应用的数字签名。  **Android数字证书概述：**  Android系统要求每一个安装进系统的应用程序都是经过数字证书签名的，数字证书的私钥则保存在程序开发者的手中。Android将数字证书用来标识应用程序的作者和在应用程序之间建立信任关系，不是用来决定最终用户可以安装哪些应用程序。这个数字证书并不需要权威的数字证书签名机构认证，它只是用来让应用程序包自我认证的。  **Android数字签名证书包含以下几个要点：**  1、所有的应用程序都必须有数字证书，Android系统不会安装一个没有数字证书的应用程序。2、Android程序包使用的数字证书可以是自签名的，不需要一个权威的数字证书机构签名认证3、如果要正式发布一个Android应用，必须使用一个合适的私钥生成的数字证书来给程序签名，而不能使用adt插件或者ant工具生成的调试证书来发布。  4、数字证书都是有有效期的，Android只是在应用程序安装的时候才会检查证书的有效期。如果程序已经安装在系统中，即使证书过期也不会影响程序的正常功能。  5、Android使用标准的java工具 Keytool 和 Jarsigner 来生成数字证书，并给应用程序包签名。  Android系统不会安装运行任何一款未经数字签名的apk程序，无论是在模拟器上还是在实际的物理设备上。Android的开发工具(ADT插件和Ant)都可以协助开发者给apk程序签名，它们都有两种模式：调试模式(debug mode)和发布模式(release mode)。  在调试模式下，android的开发工具会在每次编译时使用调试用的数字证书给程序签名，开发者无须关心。（用debug签名的应用都不能发布安装）  当要发布程序时，开发者就需要使用自己的数字证书给apk包签名，可以有两种方法。  (1)在命令行下使用JDK中的和Keytool(用于生成数字证书)和Jarsigner(用于使用数字证书签名)来给apk包签名（详见“补充信息”）  (2)使用ADT Export Wizard进行签名(如果没有数字证书可能需要生成数字证书)  **Android数字证书的有效期：**  1、数字证书的有效期要包含程序的预计生命周期，一旦数字证书失效，持有该数字证书的程序将不能正常升级。  2、如果多个程序使用同一个数字证书，则该数字证书的有效期要包含所有程序的预计生命周期。3、Android Market强制要求所有应用程序数字证书的有效期要持续到2033年10月22日以后。  **应用安装涉及到如下几个目录：**  system/app 系统自带的应用程序，无法删除  data/app 用户程序安装的目录，有删除权限。安装时把apk文件复制到此目录  data/data 存放应用程序的用户数据  Data/dalvik-cache 将apk中的dex文件安装到dalvik-cache目录下(dex文件是dalvik虚拟机的可执行文件,其大小约为原始apk文件大小的四分之一)  系统ROM的文件目录结构请参见资料包中的“Android系统目录详解.txt”文件  **应用安装过程：**  复制APK安装包到data/app目录下，解压并扫描安装包，把dex文件(Dalvik字节码)保存到dalvik-cache目录，并在data/data目录下创建对应的应用数据目录。  **Android应用安装的4种方式：**  1、系统应用安装：开机时完成，没有安装界面。（内置程序，不测）  2、市场下载应用安装：通过market应用完成，没有安装界面。  3、ADB工具安装：由命令控制，没有安装界面。  4、第三方应用安装：通过SD卡里的APK文件安装，有安装界面，由packageinstaller.apk应用处理安装及卸载过程的界面。由于调用后，整个安装过程都是由一个系统级应用Installer完成的，所以安装过程中的流程分支不需要测试。安装过程中如果发生冲突事件，也是由系统处理所以不用测试。（还有其他3种方法，但是我们公司的应用均没使用，所以不用测）  **安装路径的差异带来的影响（ROM\SD）：**   1. Android从2.2版才开始支持将应用移至SD卡的功能，2.2以下的系统都只能把应用装在手机内存中。 2. 应用移至SD卡是有限制的，并不是所有的程序都能移动，因为在开发软件的过程中，有些功能要求应用必须装在手机内存里才能正常工作。比如输入法应用，如果装在SD卡上，那么每次开机都要重新设置；有的桌面小插件如果装在SD卡上，会在桌面小工具列表中找不到相关的工具。这样的例子还有很多。所以如果您看到有些程序无法移动到SD卡，说明这个软件做了保护，我们最好还是保持它的的现状不要变动。   **安装成功的几个条件：**   1. 数字签名文件正确合法。 2. 程序版本比系统版本低或等于系统版本。   **安装失败的几类原因：**  错误一：证书过期 问题说明：数字签名证书过期 解决办法：更改数字签名文件的有效期，重新为软件签名  错误二：证书错误 问题说明：安装的软件没有签名或者签名与之前版本不一致 解决办法：为软件重新签名  错误三：证书正确，手机时间不正确 问题说明：手机的时间超出软件签名证书的期限 解决办法：把手机时间调到证书的期限内，然后再装。在安装过后再调回正确时间即可  错误四：文件受损 问题说明：安装程序文件被破坏或者不完整，有可能是在程序下载或者打包时引起的文件破坏 解决办法：解压APK包，检查程序文件包的完整性。  错误五：内存空间不足 问题说明：安装程序时ROM或者SD卡空间不足 解决办法：删除一些文件腾出空间 | | |
| **一级**  **用例** | 1、校验安装界面（logo图标、应用名称、权限申明是否正确）  2、检查安装包APK版本的正确性（可使用"apk版本检测"工具查看）  3、二维码扫描下载安装和下载安装包拷贝到SD卡安装，安装到ROM。（正常安装）  4、安装到SD卡。（使用豌豆荚或91助手等工具）  5、把已安装好的应用切换安装（从SD卡切换到ROM，从ROM到SD卡），看能否正常使用。若需求规定只能安装到ROM看是否有限制措施。  6、如果应用是放在手机应用市场上的，还要测试在手机应用市场里测试能否正常下载安装。  7、应用版本在>=系统版本上都能正常安装。（测系统版本兼容，可在AVD上测试） | | |
| **二级**  **用例** | 8、在不同品牌的手机上安装，安装后能启动正常运行。（手机平台兼容，必须真机测试）  9、如果在同一平台上有多款应用，每款应用的证书是不是一致的。  10、安装时，检验数字签名文件是否正确，有效期是否正确。（首版本的签名一定要正确，否则后面不能改了，改了就不能升级）  11、安装时选择禁用所有权限，安装成功后测试被禁用权限的地方处理是否正常。（权限）  12、以第三方软件安装时，选择"我不信任该程序"所有权限使用时需提醒，看涉及到权限的各功能点在提醒被拒绝后，应用程序能否做正确处理。（权限） | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | **快速测试兼容性：**  对于真机测试上的安装卸载测试，可以使用Testin云端测试平台测试，可以覆盖更多款机型。  Testin云端测试平台地址：http://www.testin.cn/portal.action?op=Portal.realauto  测试账号/密码：tangxiaowen/123456  测试步骤：上传APK安装包，选择机型，执行测试，获得测试结果，下载测试报告。  **如何把应用安装到SD卡上：**  一般默认都是直接安装在手机内存上，你也可以看看设置中，有没有安装位置的设置；  如果没有相关设置，可通过两种方式安装到SD卡：  1、安装后在【设置-应用程序-管理应用程序】中将软件“移到SD卡”；  2、通过数据线连接电脑，然后打开91手机助手，会弹出框让你设置安装位置，选择安装到SD卡后，下次通过91手机助手安装的软件都会安装到SD中。  需要注意，并不是所有的软件都可以转移的，否则会出现异常；输入法、widget建议还是安装在手机上  注意：要求android2.2及以上系统。有些厂家机型不支持移动到SD卡。  **给APK包签名的详细步骤：**   1. 使用Keytool创建签名文件。   （1） 使用管理员权限，进入cmd命令行，cd到jdk/bin目录下；  （2）然后输入命令： keytool -genkey -alias mySign.keystore -keyalg RSA -validity 20000 -keystore mySign.keystore  生成的mySign.keystore就在jdk/bin目录中。  解释：-alias 后面跟的是别名这里是 asaiAndroid.store 密码：66472803  -keyalg 是加密方式这里是RSA  -validity 是有效期 这里是20000  -keystore 就是要生成的keystore的名称 这里是asiAndroid.keystore  （3）然后按回车键，按回车后首先会提示你输入的密码：这个在签名时要用的，要记住，然后会再确认你的密码。 然后依次输入姓名、组织单位、组织名称、城市区域、省份名称、国家代码。设置完信息，最后再次输入原来的密码进行确认，再输入yes/no确定完成签名，并生成mySign.keystore签名文件。  2、在Eclipse的ADT中打包时选择刚才生成的mySign.keystore签名文件即可。    **如何打包解包APK：**  使用apktool工具可以打包、解包、反编译APK。详见测试工具库。  **如何查看APK文件版本：**  使用测试工具库的“apk版本检测”小工具即可检测APK安装包信息。  **如何使用ADB安装应用：**   1. 在CMD中打开Android安装路径，   例如：C:\Program Files\Android\android-sdk\platform-tools）   1. 搜索设备，建立连接，执行命令：adb devices 2. 安装应用，执行命令：   adb install 文件本地路径+文件名（多台设备的话使用-s指定安装设备）  例如：adb -s emulator-5556 install helloWorld.apk  **签名策略：**同一个开发者的多个程序尽可能使用同一个数字证书，这可以带来以下好处。  (1)有利于程序升级，当新版程序和旧版程序的数字证书相同时，Android系统才会认为这两个程序是同一个程序的不同版本。如果新版程序和旧版程序的数字证书不相同，则Android系统认为他们是不同的程序，并产生冲突，会要求新程序更改包名。  (2)有利于程序的模块化设计和开发。Android系统允许拥有同一个数字签名的程序运行在一个进程中，Android程序会将他们视为同一个程序。所以开发者可以将自己的程序分模块开发，而用户只需要在需要的时候下载适当的模块。  (3)可以通过权限(permission)的方式在多个程序间共享数据和代码。Android提供了基于数字证书的权限赋予机制，应用程序可以和其他的程序共享概功能或者数据给那那些与自己拥有相同数字证书的程序。如果某个权限(permission)的protectionLevel是signature，则这个权限就只能授予那些跟该权限所在的包拥有同一个数字证书的程序。 | | |

##### 1.1.2 IOS应用安装测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | IOS应用安装测试 | **整理人员** | 唐晓文 |
| **简介** | 测试IOS应用的安装 | | |
| **详细技术说明** | **IOS应用安装包类型：**  什么是IPA文件？什么是DEB？什么是PXL文件？  IPA是Apple官方支持的程序用用文件，可以通过iTunes来安装到iPhone上，通常主要IPA程序和游戏为主；  DEB是受iPhone上的Cydia等管理工具支持的一种Debian发行版软件安装包格式，通常Deb程序以系统工具为主；  PXL是一种支持脚本方式的iPhone上的软件安装包，不仅可以支持游戏类型，也可以支持系统工具，现今支持PXL软件管理的主要是91手机助手。更多介绍请参见以下链接：  <http://blog.csdn.net/dakouchuanqi/article/details/5797788>  **IOS应用的安装方式：**  要从ITunes上下载安装APP，必须要有ITunes账号。首先在ITunes上申请账号，申请完后用账号登录ITunes，就可以在APP商店中选择相应的APP安装。这是官方操作，如果想不用账号随便安装软件必须把IPhone越狱，但越狱后手机不保修。  在开发过程中如何安装测试版本程序到IPhone真机上，方法有如下两种：  1、购买苹果的开发者账号，99美刀获取证书（只支持100个用户安装）  2、越狱手机，可以随意安装。  目前我们使用的是第一种，经过开发人员把证书授权到安装包中，就可以使用ITools直接在手机上安装。  手机应用发布后，需要测试程序是否能在苹果商店上下载安装升级。可以把手机连接电脑使用ITunes下载安装（如下例子所示），也可以使用手机上的APP Store下载安装。  由于IOS上安装应用比较封闭和智能所以测试时不需要花太多精力去关注，主要是检测程序上线发布后，在APP Store上能正常下载安装即可。上线前还要做好检测，确保能够通过APP Store的审查，顺利发布。 | | |
| **一级**  **用例** | 1、在手机上的APP Store搜索被测应用并顺利下载安装成功。  2、安装完成后能顺利启动应用，且安装、启动时间可接受。  3、使用不同安装工具安装应用：91助手、ITunes、ITools能够正常安装。  4、越狱的手机上能够安装成功，启动成功。 | | |
| **二级**  **用例** | 暂无 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 暂无 | | |

#### 1.2应用更新测试

##### 1.2.1 Android应用更新测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | Android应用更新测试 | **整理人员** | 唐晓文 |
| **简介** | 测试Android应用的更新升级。 | | |
| **详细技术说明** | **网络在线自动更新步骤原理：**  首先通过配置文件中的网络地址，获取服务器上的应用的版本、文件名以及下载地址，读出存放软件的版本信息。接着获取到我们手机上应用的版本信息，即版本号（versionCode）和版本名称（versionName）。versionCode是整数型，而versionName是字符串。由于versionName是给用户看的，不太容易比较大小，升级检查时，就可以检查versionCode。把获取到的手机上应用版本与服务器端的版本进行比较，应用就可以判断处是否需要更新软件。如果小于服务器的版本号，就下载新版本，更新。  **网络下载更新的流程：**  http://static.oschina.net/uploads/img/201301/05181807_cXp2.jpg  相关知识链接：<http://www.oschina.net/question/565065_85874>  **系统设置对更新过程的影响：**  Android系统的应用安装，在系统设置里面有一项，是否安装未知源，所在在软件更新的时候，需要检测这个选项，如果打钩，则只允许安装Market源提供的安装程序，如果没有打钩的话，系统安装应用时会提示用户设置，如果选择设置，设置好后，无法返回安装界面；如果选择取消，则推出安装程序。所以，如果是更新的话，一定要在下载之前就检测许可安装源的设置，或者在下载前检测是否已经下载过新的安装程序，避免重复下载安装程序。  **更新失败的原因：**   1. 服务器上更新包不存在，下载失败。 2. 已安装的版本的签名和更新包版本签名不一致。 3. 缺少安装应用程序权限。（需要在配置文件声明） 4. 系统设置不允许安装未知源应用。（需修改系统设置，允许安装未知源应用） 5. 新升级版本和老版本不兼容。 | | |
| **一级用例** | 1. 检查是否能自动更新。（主要是检查原版本配置文件的更新地址配置是否正确，服务器端发布的版本是否正确） 2. 验证数字签名是否正确一致。（只要能正常安装成功就一致） 3. 测试没升级的以前的旧版本，是否都能正常使用。（避免服务器端或数据库端更改造成旧版本无法使用） 4. 跨版本升级，测试功能是否正常（例如：从1.0跨越1.2、1.5升级到2.2）。 5. 权限检测。系统设置为限制安装未知源软件，下载更新包之前有没有检测权限。 6. 系统限制安装后，设置完成后不需要重复下载安装包。（测试是否重复下载） 7. 下载时断网了，下载失败，是否会在网络重新连接上后断点续传。（总之处理要合理） 8. 下载时SD卡空间不足，是否做出判断提示。 9. 下载完成后，点击取消安装，再次打开应用时会否提示安装，且点击“立即升级”后无需再次下载，直接安装。 | | |
| **二级**  **用例** | 1. 上线时验证各版本更否自动升级更新安装。（确定以前的各个版本versionCode是正确的，则可以不用所有都测，只测一个即可） 2. 更新时拒绝某些权限，升级后能否正常使用。（权限） 3. 更新时设置权限为使用到时询问提醒，升级后相关功能能否正常使用。（权限） | | |
| **三级**  **用例** | 1、下载过程中关机、断电后，再次打开是否会断点续传。 | | |
| **补充**  **信息** | 暂无 | | |

##### 1.2.2 IOS应用更新测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | IOS应用更新测试 | **整理人员** | 唐晓文 |
| **简介** | 测试IOS应用的升级更新 | | |
| **详细技术说明** | 升级更新后要考虑的问题：  1、数据结构变化造成的新老版本不兼容  2、每个老版本是否都能升级成最新版本，新版本的一些功能改造是否和一些老版本有冲突。 | | |
| **一级**  **用例** | 1、在从没安装过被测应用的手机上安装最新版本，安装成功，正常启动。  2、在安装过老版本的机器上升级新版本。每个老版本都能升级成功，功能使用正常。  3、从新版本升级到老版本，用户之前保存的数据无丢失且新老功能均能正常使用。当客户端有新版本时，有更新提示。  4、当版本为非强制升级版时，用户可以取消更新，老版本能正常使用。用户在下次启动app时，仍能出现更新提示。  5、当版本为强制升级版时，当给出强制更新后用户没有做更新时，退出客户端。下次启动app时，仍出现强制升级提示。  6、当客户端有新版本时，在本地不删除客户端的情况下，直接更新检查是否能正常更新。  7、当客户端有新版本时，在本地不删除客户端的情况下，检查更新后的客户端功能是否是新版本。  8、当客户端有新版本时，在本地不删除客户端的情况下，检查资源同名文件如图片是否能正常更新成最新版本。如果以上无法更新成功的，也都属于缺陷。 | | |
| **二级**  **用例** | 暂无 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 暂无 | | |

#### 1.3应用卸载测试

##### 1.3.1 Android应用卸载测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | Android应用卸载测试 | **整理人员** | 唐晓文 |
| **简介** | 测试Android应用的卸载 | | |
| **详细技术说明** | **卸载原理：**  卸载就是把安装应用时产生的三个文件夹删掉。（data/app, Dalvik-cache，data/data）  **Android手机上卸载应用的几种方式：**   1. 手机自带的应用管理卸载（在“设置”里面） 2. 手机管理工具卸载、桌面长按键卸载。 3. Shell命令卸载。 | | |
| **一级用例** | 1. 卸载后相应安装目录中的数据是否能完全删除掉（产品要求特殊保留的除外）。 2. 卸载后相应的桌面组件是否也被卸载、通知栏是否还显示应用相关消息。 3. 测试各种卸载方式能否正常卸载。 4. 是否运营商定制应用，是定制应用无法卸载。（测试无法删除） | | |
| **二级**  **用例** | 暂无 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 暂无 | | |

##### 1.3.2 IOS应用卸载测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | IOS应用卸载测试 | **整理人员** | 唐晓文 |
| **简介** | 测试IOS应用的卸载 | | |
| **详细技术说明** | 一般IOS上卸载应用会把所有相关数据全部清除。  **IOS手机上卸载应用的几种方式：**  1、手机管理工具卸载（ITunes、ITools删除）  2、桌面长按键卸载 | | |
| **一级用例** | 1、卸载后相应安装目录中的数据是否能完全删除掉（产品要求特殊保留的除外）  2、各种卸载方式是否能正常卸载。  3、是否运营商定制应用，是定制应用无法卸载。（测试是否能无法删除） | | |
| **二级**  **用例** | 暂无 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 暂无 | | |

### 2、功能测试

**常用功能点：**下拉刷新列表、通知功能、外部功能跳转、定位功能、权限相关测试、横竖屏切换功能、上传下载文件、重力感应功能、启动退出、编辑框校验、外部接口调用、视频播放、音频播放、后台服务、内置广告、登录、注册、找回密码、Wigdet插件测试、搜索功能测试。（第一期只包含这些，以后再不断扩充）

（每个测试点中包括了相应的冲突测试等其他测试类型）

以下属于功能测试范畴，但不单独以测试点的形式记录在框架中，只是会在上面功能点中穿插以下情况的用例：

**环境差异测试：**有无SIM卡，SIM卡停机或无效，有无SD卡，访客模式，飞行模式、联机模式、充电模式、Root过的手机和没Root过的手机。

**额外操作：**清本地数据、清缓存、更改权限设置、回收内存。

#### 2.1下拉刷新列表测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 下拉刷新列表测试 | **整理人员** | 陈小红、唐晓文 |
| **简介** | 测试手机应用的下拉刷新列表 | | |
| **详细技术说明** | **手机应用中的下拉刷新列表功能具有以下特点：**   1. 下拉提示用户刷新，拉到一定长度放手回弹，即可刷新列表，获取到最新的数据，因此列表中的数据最基本的特点就是按时间顺序倒序排列。 2. 列表页面首次加载一般只取固定条数的信息，例如：10条、20条。一进入页面就会刷新，获取最新的10条信息，下拉刷新的最后更新时间也就定为进入页面的时间。当然有些应用根据产品要求，可能会要求在进入页面时不刷新，仍旧加载用户上次已获取的列表，等用户自主下拉再刷新。 3. 为了避免一次性加载全部数据过慢或内存溢出，一般在列表的下方都有一个加载更多的按钮。每按一次或触发一次就获取固定条数的历史数据，直到获取完为止。 4. 列表中的项一般点进去跳转到信息详细页面。 5. 当列表没有数据的时候，给出提示信息。 6. 断网或报错的时候，给出提示信息，不会影响用户其他操作，也不会自动退出。   **获取新数据的下拉刷新功能流程详解：** 1、下拉，显示提示头部界面，这个过程提示用户"下拉刷新" 2、下拉到一定程度，超出了刷新最基本的下拉界限，我们认为达到了刷新的条件，提示用户可以"松手刷新"了，效果上允许用户继续下拉。  3、用户松手，可能用户下拉远远不止提示头部界面，所以这一步，先反弹回仅显示提示头部界面，然后提示用户"正在加载"。 4、加载完成后，隐藏提示头部界面，列表数据刷新。  **获取旧数据的列表下方的加载更多项功能详解：**   1. 点击或触发加载更多按钮，就会在列表下方加载固定条数的信息，按时间倒序排列加载。 2. 如果还有数据的话，列表再滑到最低端仍旧显示更多按钮，如果数据已经全部加载完，则不显示加载更多按钮，而是显示底边界。   **注意事项**  1、非正常网络时，下拉列表刷新；  2、下拉列表刷新，页面数据缓存；  3、飞滑时，列表处理；  4、使用到的应用权限（2G、3G、wifi的访问权限）；  5、列表中数据量加载过大内存溢出。 | | |
| **一级**  **用例** | 1. 下拉列表，是否有显示提醒用户下拉刷新、正在执行刷新等字样，是否记录了最近一次更新时间； 2. 正在刷新加载时，原来的列表展示是否正常，是否有进度圈显示； 3. 下拉列表回弹后，是否正常执行刷新（有新内容加载时、没有新内容加载时均正确）； 4. 刷新完成后，列表数据是否正确(包括加载的数据条数、数据信息详细—绑定的内容字段、图片是否正确)； 5. 下拉滑动，松手反弹效果是否实现，是否流畅； 6. 上下滑动列表，滚动条坐标定位是否准确； 7. 下拉列表中的项，是否按时间倒序排序； 8. 列表下滑到最后一行，应有边界提示；列表上拉到第一行也应有边界提示。 9. 列表下方的加载“更多”按钮，可实现每次加载固定条数的历史数据（数据条数正确、排序正确、详细信息和图片加载正确）。 10. 当列表没有信息的时候，下拉刷新给出无数据的提示信息。 | | |
| **二级**  **用例** | 1. 列表加载成功后，清缓存、清内存、清数据后，再刷新都能正常加载； 2. 图片列表下拉，图片加载是否流畅，是否有延迟，加载过的图片再上下滑动查看是否被缓存； 3. 下拉刷新提示正在刷新后，再下拉，应该回弹，且不会影响当前的正在加载的刷新操作。 4. 列表数据过大(图片过大)，下拉刷新是否正常（包括加载速度是否合理，占用资源是否正常，列表数据(图片)展示是否正常） 5. 网络异常下拉刷新，数据加载是否有超时处理，给出相应提示； 6. 正在加载中，按返回键、home键是否会对应用运行有影响。 7. 加大数据量测试列表一次下拉（例如：新增10000条数据一次性下拉获取），是否有内存溢出或变慢。 8. 加大数据量，测试列表加载更多功能，直到获取完全部列表看是否内存溢出。（例如：列表总共有10万条历史数据，一直按列表下方的加载更多按钮获取历史数据，全部获取完后滑动列表均正常）——注：具体的测试数据量选取还要看产品运营中，可能出现的数据量做参考。如果量很大就有必要测这个用例。 | | |
| **三级**  **用例** | 1. 当应用没有网络访问权限时，下拉刷新和获取更多数据时，应给出正确提示。（限制网络权限方式又分为，强制限制网络访问权限和提示用户限制网络访问权限用户可选择开启，不管是哪种方式其提示都应一致，且不妨碍用户继续使用其他功能） | | |
| **补充**  **信息** | 造大量测试数据，可以使用date factory 在测试数据库中生成大量测试数据。  工具安装包和使用方式请参见工具库。 | | |

#### 2.2通知功能测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 通知功能测试 | **整理人员** | 陈小红、唐晓文 |
| **简介** | 测试Android应用中的通知功能。 | | |
| **详细技术说明** | **Android应用中通知功能的特点：**   1. 通知功能从Android1.5版本开始引入，类型可分为：抽屉通知、进行中的通知、弹出通知、服务器端推送通知。 2. 抽屉通知：就是我们一般见到的，需要用户自行到通知界面去查看的通知，不管你点开查看过没都可以被删除。 3. 进行中的通知：这类通知是为了让用户了解后台正在运行的进程。例如：正在播放的音乐播放器、正在后台运行的省电程序、正在保护系统的安全软件等。另外也可以对下载上传、视频编码这种持续时间较长的任务提供反馈。这种进行中的通知是不可以被移除的。 4. 弹出通知：会绕过抽屉通知直接出现在用户面前。一般情况下很少使用，只在需要及时地反馈并且必须打断用户的场合下才会使用。例如Talk应用使用这种形式的通知来提醒用户有好友邀请他加入视频聊天，因为这个邀请会在几秒后自动失效。 5. 服务器端推送通知：主要是一种运营手段，利用通知功能把服务器端的消息“群发”到各个安装了该应用的手机上。简单来说就是要获取服务器上不定时更新的信息，例如：网易推送的新闻消息。一般实现起来有两种方法：第一种是客户端使用Pull（拉）的方式，就是隔一段时间就去服务器上获取一下信息，看是否有更新的信息出现。第二种就是：服务器使用Push（推送）的方式，当服务器端有新信息了，则把最新的信息Push到客户端上。Push方式比Pull方式更优越。因为Pull方式不仅有延迟还费电费流量。 6. 一般抽屉通知包含以下元素：Icon头像图标、Title标题名字、消息内容、时间戳（在 Android 通知的右上角要显示时间戳。一般紧急信息才加时间戳）、应用的图标。如下图：   http://dl.iteye.com/upload/attachment/0069/6632/3cd06ca3-7f70-3b35-8db1-28d86f126a9e.jpg  这只是个系统布局，开发人员也可以自己自定义布局，但是自定义布局需要注意测试兼容性。   1. 同一个应用发来的通知可以叠加成一个通知。如果已经有正在等待处理的相同通知了，那么您应用不应当在创建一条新的通知，而是将多条通知合并。合并的通知提供了总体的信息描述，并且告知用户有多少条通知正在等待处理。   http://dl.iteye.com/upload/attachment/0069/6634/8e590365-bf6e-3145-ac57-bf077adf41a5.jpg   1. 点击通知可以跳转到相应页面：如果用户点击了一条通知，此时应该将相关的应用程序打开到可以对通知中提到的内容进行操作的状态页面，再点击“返回”和“向上”按钮则会在内部应用程序中跳转，不会再跳回通知界面。但如果用户点击了一个合并的通知，应该去到列表页面（即内容详情页的上一层级）。对于弹出通知，点“返回”会关闭弹出通知。如果用户从这条弹出通知进入了发出通知的应用程序，“向上”和“返回”的逻辑会与标准通知的逻辑保持一致，在应用程序内进行导航跳转。（以上一系列的跳转规则需要开发人员实现，不是系统自定的，所以需要测试） 2. 通知的清理删除：自动删除具有时效性的已过期通知。用户查看过的聊天消息或邮件等通知，应该在查看过后自动移除这些通知。用户也可以手动清理未查看的通知。 3. 通知发送到达后，在状态栏上会有提示。一般是一段简短的文字摘要和图标。默认情况下，待处理的通知是以图标形式显示在状态栏中，从屏幕上方向下滑即可打开通知抽屉。在抽屉中左滑、右滑可以删除通知。 4. 在应用程序中可以设置是否发送通知。   **创建发送一般通知的基本原理：**  一个Activity或者一个Service都可以创建一个状态通知。（对Activity来说，仅仅当它正在前端显示并且其窗口成为焦点时，它才能执行各项操作，所以大多数情况下你只需要在Service中创建状态通知。） 创建通知必须使用这两个类: Notification和 NotificationManager。  当用户选择了一条通知时，Android就会启动一个被这个Notification 定义了的Intent（即跳转到应用相应页面，此时测试需要检查跳转页面是否正确）。  可以在设备上设置通知到来时的提醒方式，比如声音，振动或者屏幕闪烁。  可以用Notification类的一个实例来定义你的状态通知属性，比如状态图标，通知内容或者额外的一些设置比如播放一段提示音。NotificationManager是Android的系统服务，它负责管理和执行所有的状态通知。接收到通知后，Notification对象所定义的通知的详情会在状态栏或者通知窗口中显示。  创建一个状态通知，需要以下几步： 1.得到NotificationManager的引用：  2.初始化Notification：  3.定义通知的内容和PendingIntent:（包括所有元素：图片、标题、内容、时间戳、头像等）  4.传送Notification到NotificationManager:  就这样，一个通知就发送完毕了。  **更新通知原理：**  当你的应用程序中的通知事件在持续执行时，你可以更新你的通知中的信息。例如，当一个新的SMS消息到达之前你已经读了先前的消息，消息应用程序就会更新现有的通知以便显示新收到的消息的总数。更新现有的通知而不是在原来的基础上增加新的通知，这种做法的确是更好的，因为它避免了通知窗口中的混乱局面。  因为每个通知都被NotificationManager给予了一个整型的ID作为唯一的标识，所以你可以通过条用 setLatestEventInfo()方法并传新的参数来修改通知，改变这个通知的一些属性，然后调用notify()就可以了。  **服务器端推送消息的几种实现原理：**  1、客户端轮询(Pull)方式：应用程序应当阶段性的与服务器进行连接并查询是否有新的消息到达，你必须自己实现与服务器之间的通信，例如消息排队等。而且你还要考虑轮询的频率，如果太慢可能导致某些消息的延迟，如果太快，则会大量消耗网络带宽和电池。Android操作系统允许在低内存情况下杀死系统服务，所以我们的推送通知服务很有可能就被操作系统Kill掉了。  2、SMS(Push)方式：在Android平台上，你可以通过拦截SMS消息并且解析消息内容来了解服务器的意图，并获取其显示内容进行处理。这个方案的好处是，可以实现完全的实时操作。但是问题是这个方案的成本相对比较高，需要向移动公司缴纳相应的费用。我们目前很难找到免费的短消息发送网关来实现这种方案。  3、持久连接(Push)方式：这个方案可以解决由轮询带来的性能问题，但是还是会消耗手机的电池、流量。一般来说，每个手机平台都有自家提供 Push 功能，让应用开发者能够很方便地把 Push 能力集成到应用中。Android 上有 GCM (Google Cloud Messaging，原来叫 C2DM），iOS 上有 APNs（Apple Push Notification service），Windows Phone 上有 MPNs（Microsoft Push Notification service）。但悲剧在于，GCM 在国内基本上是不可用的。有如下两个原因：其一、国内大部分 Android 手机都是不带 Google 服务的，也就用不了 GCM 这个 Google服务。这是主要的问题。其二、在国内Google的服务，大多是不稳定的。无论是使用第三方提供的Push服务还是自己研发的Push服务，目前来说这种方式比较普遍，实现的原理都是：客户端开启后台连接服务，心跳监听服务器端是否推送消息，监听到有消息后通知客户端用户。 | | |
| **一级**  **用例** | 1、通知权限校验(是否允许应用程序发送通知，拒绝后是否强制允许并发送通知)——测试权限。  2、检查通知设置功能（设置开启可发送通知，设置为禁止不能发送通知）——测试应用配置。  3、检查手机系统设置关闭通知功能，是否能自动打开并发送通知。——测试系统设置。  4、通知发送是否成功-应用程序存在通知信息，手机是否能正常接收通知信息。  5、通知信息是否展示成功-手机接收通知后，通知信息是否正确（图标、内容、时间戳等）。  6、点击通知是否进入正确页面，进入页面后点击“返回”或“向上”按钮，是否跳转到应用程序内部响应页面。  7、多条相同应用发送的同类信息是否合并为一条通知显示，合并显示的内容是否正确，条数是否显示正确。点击后是否进入列表页而非详细页。  8、如果是弹出通知，点击“返回”键后是否会消失，点击通知后能否打开应用进入相应页面，进入页面后点击“返回”或“向上”按钮不会跳出本应用。  9、通知提醒方式是否正确-设置对应通知提醒方式后，通知信息是否正确提醒：如只显示通知消息、声音提醒、震动提醒、LED闪烁提醒、弹出提示。  10、清除通知是否能正常手动清除(或清除时是否强制不清除)。  11、对于有时效性的通知，是否能自动清除。  12、对于用户已经查看过的通知，在通知列表中是否自动清除。  13、对于自定义的通知布局，测试布局的系统版本兼容性和界面兼容性。  14、Push消息，能否到达，通知、正确展示、点击后进入正确页面。  15、Push大量消息，是否会有延迟、堵塞。  16、Push推送是否会对同一个用户重复推送多条消息。 | | |
| **二级**  **用例** | 17、通知是否存在重复发送问题。  18、通知发送失败后，是否会重发。  19、Push服务推送失败后，是否会重发。  20、Push客户端后台监听程序被强制关闭后，是否会重启。 | | |
| **三级**  **用例** | 21、Push服务的客户端后台监听服务长时间运行的可靠性（无内存泄露、流量消耗均匀无异常、无报错、无自动停止--在系统资源不足的情况下也不被系统杀掉）。  22、大量数据Push服务的压力测试（如果使用的是第三方Push服务，则不需要测试，由第三方公司保障） | | |
| **补充**  **信息** | 暂无 | | |

#### 2.3外部功能跳转测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 外部功能跳转测试 | **整理人员** | 苟红梅、唐晓文 |
| **简介** | 测试Android应用中对于调用外部各功能点的测试 | | |
| **详细技术说明** | **外部功能跳转测试在这里主要指以下两个大概念：**  1、调用其他应用的功能。（主要是指被测应用调用外部应用功能，例如：相册照片的选取、广告点击后的跳转、链接点击跳转、分享、调用外部地图程序等等）  2、使用其他应用的数据。（主要是指被测应用扫描使用其他应用的数据，例如：阅读应用扫描手机中所有txt文件加入书架中、音乐播放应用获取手机中所有可播放的音乐文件、应用获取手机联系人、获取通话记录等等）  **Android应用调用外部功能原理：**  在一个Android应用中，主要是由四种组件组成的（Activity、Intent Receiver、Service、Content Provider），他们之间是独立的，可以互相调用，协调工作，最终组成一个真正的Android应用。而这些组件之间的通讯主要是由Intent完成的。  Intent负责对应用中一次操作的动作、动作涉及数据、附加数据进行描述，Android则根据此Intent的描述，负责找到对应的组件，将 Intent传递给调用的组件，并完成组件的调用。Intent在这里起着一个媒体中介的作用，专门提供组件互相调用的相关信息，实现调用者与被调用者之间的解耦。  Android上不同程序里面的Activity之间互相调用，调用的方式就是向系统提出请求，系统会根据你提供的条件去寻找适当的程序Activity来供你调用。Android系统解析Intent主要是通过查找已注册在各个AndroidManifest.xml中的所有IntentFilter及其中定义的Intent，最终找到匹配的Intent。在这个解析过程中，Android是通过Intent的action、type、category这三个属性来进行判断的，具体判断方式不在此详述。  要通过Intent调用外部功能具体实现有两种方式：一种是通过查找定义好的“动作”来启动跳转到相符合的activity，另一种是通过查找activity名，直接打开跳转（这类多用于同一应用内的跳转）。所以以下仅针对第一种方式（根据动作跳转）做详细介绍：  在Android中，对Intent的定义是执行某操作的一个抽象描述。它到底描述了什么。  首先，是要执行的动作(action)的一个简要描述，如VIEW\_ACTION(查看)、EDIT\_ACTION(修改)等，Android为我们定义了一系列的标准动作（可以被其他应用调用）。此外，我们还可以根据应用的需要，定义我们自己的动作，并可定义相应的Activity来处理我们的自定义动作（自定义的动作能被系统扫描到，并提供给其他应用调用）。其次，是执行动作要操作的数据（data），Android中采用指向数据的一个URI来表示，如在联系人应用中，一个指向某联系人的URI可能为：content://contacts/1。(注：URI是Android系统中的唯一资源标示符，用于寻找资源)  Action和data组合起来用就能描述一些工作，例如：  VIEW\_ACTION content://contacts/1-- 显示标识符为"1"的联系人的详细信息  EDIT\_ACTION content://contacts/1-- 编辑标识符为"1"的联系人的详细信息  VIEW\_ACTION content://contacts/-- 显示所有联系人的列表  PICK\_ACTION content://contacts/-- 显示所有联系人的列表，并且允许用户在列表中选择一个联系人，然后把这个联系人返回给父activity。例如：电子邮件客户端可以使用这个Intent，要求用户在联系人列表中选择一个联系人。  除了action和data这两个重要属性外，还有一些附加属性，如：category（类别）、type（数据类型）、component（组件）、extras（附加信息）。在此不详细描述讲解。  总之，action、data/type、category和extras 一起形成了一种语言。  这种语言使系统能够理解诸如“查看某联系人的详细信息”之类的短语。  随着应用不断的加入到系统中，它们可以添加新的action、 data/type、category来扩展这种语言。应用也可以提供自己的Activity来处理已经存在的这样的“短语”，从而改变这些“短语”的行为。使用这些短语就能使用外部应用的功能。  以“分享”功能为例实现如下：分享功能是一个菜单，这个菜单可以打开一个页面，在打开时也传给了这个页面一个Intent。这个Intent里面定义了动作筛选条件，页面把根据这个条件选出来的应用加载到列表中，只有定义了这类动作的应用才会被筛选出来显示在这个页面中。例如：分享照片功能，列出来的是所有定义了分享图片动作的应用，点击进去就是各应用在配置文件中定义好的Activity页面，且该页面也具有相应的分享图片功能。（所以，分享功能打开能有多少个可以使用，不是开发人员控制的，而是系统筛选的结果，开发人员只负责把筛选条件填对。但对于自定义开发的分享除外，自定义开发的分享可以指定分享到哪些应用，如果该应用在手机上没有安装，则自动跳转到应用下载页面）  **调用其他应用功能的两种具体实现方式：**  1、使用自定义Action调用：在被调用的应用的AndroidManifest.xml配置文件中声明action，调用该程序该功能的时候只需要知道action名称即可。  2、使用包类名调用：无需在被调用的应用的AndroidManifest.xml配置文件中声明，只要知道被调用程序的包名和Activity类名，即可以此调用相应的程序。  第一种比较常用。在AndroidManifest.xml配置文件中如果设置了权限限制，则在调用时需要声明权限才能调用相应的Action和Activity。（测试时注意权限，特别是调用系统应用）  **Android应用使用外部数据（应用之间的数据共享）：**  通常在android应用中，数据都是在本应用沙盒之内的，其他外部应用不能够访问，那么如果一个应用需要访问另外一个应用的数据，那就需要把另外一个应用的数据公布出来，比如android中的通讯录数据，这些数据是以ContentProvider方式提供与其他应用访问的。我们也可以定义自己的ContentProvider来使跨应用共享数据。数据具体的存贮方式可以为数据库、文件，持久化或非持久化存储的其他形式。  **特别说明：**  1、Android中的Activity、Content Provider、Service、Broadcast Receiver等。这些如果在Androidmanifest.xml配置不当，会被其他应用调用,引起风险。最好全部都配置相应权限，获得权限才能访问（在测试过程中注意检查暴露在外部可调用的东西要确保有权限限制，否则就是安全漏洞。这个在安全测试中会有详细讲解）  2、不同的系统版本、不同厂家开发的定制系统中，系统应用的实现会有所不同，所以我们在调用外部应用时，会发生兼容性问题，例如：调用相册选择图片，在三星上可以选中返回图片，在小米上就不行。（在这要特别注意兼容性测试） | | |
| **一级用例** | 1. 检查外部链接能否正常跳转进入所指向的页面； 2. 检查调用外部功能，是否能够正常进入外部应用相应界面，并传入正确数据（例如：分享照片到记事本，记事本能否打开，照片能否正常带入记事本编辑页面并做相应操作）； 3. 测试调用外部应用时，回传参数是否正确（如：调用拍照后返回照片，选择相册图片后返回图片） 4. 检查外部数据能否正常调用（例如：通讯录联系人是否能正常获取使用、SD卡中的数据能否正常读写）； 5. 当没有数据访问权限时，访问外部数据，能否恰当处理并给出正确提示。 6. 当没有外部应用使用权限时，调用外部应用的功能，能否恰当处理并给出正确提示。 7. 检查（如广告）需进行人为配置的外链接，通过正常情况配置后，访问时能否跳转到所指向的链接地址页； 8. 检查（如广告）需进行人为配置的外链接，所配置的链接地址错误，在访问时链接的跳转处理； 9. 跳转到外部应用后，按返回键能否回到跳出前的原应用页面（例如：点击某个外链接后，访问链接所指向的页面，点击手机上的物理返回键，检查能否返回到外链接的点击页）； 10. 调用系统应用时，需要在不同系统版本和不同手机品牌间做兼容性测试。因为，每个厂家和系统版本中的应用实现有可能不同。 | | |
| **二级**  **用例** | 1. 在访问android应用中所包含的外链接过程中，断开网络检查程序对无网或网络终断时的外链接访问处理；（这个本质上测的是网络链接访问，而不是外部功能跳转） 2. 在访客模式下，测试调用外部数据（联系人、通讯录等）功能，是否能合理处理。 | | |
| **三级**  **用例** | 13、被测应用如果有提供给外部的功能或内容，需要测试是否加了权限限制。 | | |
| **补充**  **信息** | 暂无 | | |

#### 2.4定位功能测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 定位功能测试 | **整理人员** | 唐晓文 |
| **简介** | Android应用中的定位功能测试 | | |
| **详细技术说明** | **Android系统提供的3种定位方式：**  1、基于移动互联网3G/2G：根据基站的位置定位。采集到手机上的基站ID号（cellid）和其它的一些信息（MNC，MCC，LAC等等），然后通过把它们传给百度api或者谷歌api获取位置信息。其特点是精度低、速度快，缺点是必须得有信号。  2、基于WIFI定位：和基站定位类似，这种方式是通过获取当前所用的wifi的一些信息，然后发给网络上的定位服务以获得经纬度坐标。因为它和基站定位其实都需要使用网络，所以在Android里也统称为Network方式（基于网络定位）。  3、基于GPS卫星导航。其特点是精度高、速度慢、耗电。首次定位耗时长。一般需要10-60秒时间才能开始第1次定位，如果是在室内则基本上无法定位。优点是：GPS走的是卫星通信的通道，在没有网络连接的情况下也能用。  一般来说NETWORK得到的位置精度一般在500-1000米，GPS得到的精度一般在5-50米。  手机一般会优先使用GPS定位，定位不上时使用基于网络的基站定位。  手机中可设置是否允许Google使用位置信息。  还有一种方式也可以用来定位大致位置（定位到省、市），这种方式与Android系统无关，实现原理是获取到手机号后根据手机号码号段定位手机号码归属地，不是实时的地点，而是理论上的用户所在地。  **Android应用中定位实现原理：**  在Android应用中实现定位使用到的类是LocationManager。  LocationManager可以监听获取当前位置信息，可以定义要返回的信息（设置精度要求、经纬度、方位、海拔、是否耗话费、时间等）  **具体实现起来会这样做：**启动两个监听（既监听网络定位，又监听GPS定位，获取实时位置），先使用网络定位来得到1个精度较差的位置，再使用GPS来得到更准确的位置。渐进式精度定位响应更快。这里设置了关闭GPS也没用，程序只要拥有这个权限就可以打开，再关闭。定位最好设置超时时间，否则有可能遇到问题时，程序一直卡在定位上，用户无法退出使用其他功能。  如果超时，可以使用上一次定位的位置，如果上次定位位置为null只能告知用户定位失败。  Ps：LocationManager有个函数可以得到上次定位时的最后位置，通常在应用刚启动时立刻得到1个位置，这样应用看上去会比较快。  **如果不这样做：**生硬的直接使用GPS定位，GPS模块从启动到获取数据之间时间会比较长，可能有2~3分钟时间，那么你的程序可能会被这个函数阻塞几分钟。正是基于这样的考虑，android上要想获取定位信息，必须使用异步方式。  **测试时需要特别注意：**  1、由于实现定位功能调用的是系统框架方法，需要做系统兼容性测试。  2、实现定位功能需要获取定位权限（测无权限）。  3、定位功能可以由系统开启或关闭（测关闭功能）。  4、网络定位需要网络，需要测试在不同网络下的定位（测关闭网络、各网络下的兼容性）。  5、如何模拟测试外地定位是否正确。（一些手机的开发模式下可以设置“允许模拟地点”，再结合DDMS工具就可以实现模拟外地数据） | | |
| **一级**  **用例** | 1、网络和GPS都开启的情况下，测试定位是否成功，且速度快，定位精度准确。  2、测试只有移动网络情况下，定位是否成功，且速度快，定位精度准确，无报错。  3、测试只有WIFI的情况下，定位是否成功，且速度快，定位精度准确，无报错。  4、测试只有GPS的情况下，定位是否成功，且速度快，定位精度准确，无报错。  5、测试在手机设置中关闭GPS后（wifi、2G/3G也关闭），定位是否成功，是否精准（程序应能自动打开GPS，使用完后关闭）  6、定位失败后是否给出提示，和合理处理（使用上一次定位成功的地址）。  7、没有定位权限时，如何处理，给出相应提示还是用上一次定位地点。  8、定位过程中，按返回键退出。 | | |
| **二级用例** | 9、在不同的系统版本、不同厂家机型中测试定位功能。（系统不一样，定位功能可能失效） | | |
| **三级用例** | 10、在不同网络环境中测试定位速度和精度（移动、联通、电信）  11、在不同环境中测试定位速度和精度（室内、室外、电梯中）  12、在本地测试外地定位（使用DDMS工具中的模拟定位地址来测试，DDMS只能在AVD模拟器上开启模拟位置功能）。 | | |
| **补充说明** | DDMS工具、AVD模拟器的安装包和使用文档见测试工具库  只要安装了Android SDK，这两个工具就安装成功了。 | | |

#### 2.5横竖屏切换功能测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 横竖屏切换功能测试 | **整理人员** | 宪敏、唐晓文 |
| **简介** | 测试Android横竖屏切换功能测试 | | |
| **详细技术说明** | **横竖屏切换功能简介：**  横竖屏自动切换功能是在重力感应作用下，自动切换窗口的横屏显示和竖屏显示功能。可以在手机中设置是否自动切换，也可以在应用中设置每个页面是否允许横竖屏切换，或者专门实现横竖屏切换功能，不管当时重力感应到的是哪个方位（例如：漫画应用的横竖屏切换）这类切换不考重力感应，也不靠系统实现，是自己强制的切换。  android系统的手机绝大部分只支持三种屏幕方向的重力感应自动切换，竖直，水平，反方向水平，实际上Android本身是支持反方向竖直的重力感应切换的。  **横竖屏切换时页面内部会发生什么：**  在做android开发时，在默认情况下当屏幕从竖评变到横屏时会触发  onConfigurationChanged事件。  在默认情况下横屏切换竖屏会重新加载画面并显示和横屏一样的画面，这样会有2个问题，  1、布局问题：在竖屏显示的布局到横屏中由于宽和高发生了变化，所以多少都会影响到布局（除非你定制2套画面，然后再加个判断在横屏时怎么显示，在竖屏时怎么显示），当然最简单的办法就是在项目的 AndroidManifest.xml中找到你所指定的activity 中加上  只竖屏显示的话(android:screenOrientation="portrait")  只横屏显示的话（android:screenOrientation="landscape"）  2、画面重新载入问题：在默认情况下横竖切换时会重新载入画面，导致一些不必要的资源浪费，更严重的是画面上保持的数据（特别是游戏方面）也都被重置了（当然你可以在重置前保存这些数据到数据库或者到文本文件中）。也可以在代码中强制避免横竖切换时重新载入画面。不过如果需要在重新载入过程中保存之前的操作内容或数据，则需要保存之前的数据。然后在activity的 onCreate()中取出来加载到切换后的新页面中。  **测试时需要注意的问题：**  1、对于需求中明确指出不支持横竖屏切换的页面是否做了限制。  2、对于支持横竖屏（4个方向）切换的页面，是否有相应界面（横屏界面、竖屏界面）显示。  3、横竖屏切换后功能是否一致、数据是否一致（编辑中的数据能否保留不丢失）。  4、横竖屏界面功能应该一致，横屏有的竖屏也应该有。 | | |
| **一级**  **用例** | 1、横屏查看各功能是否实现，页面是否显示正确（显示的是横屏界面）  2、竖屏查看各功能是否实现，页面是否显示正确（显示的是竖屏界面）  3、横竖屏切换（4个方向），查看各功能是否实现，页面是否显示正确，数据是否保留。  4、横屏和竖屏所包含的功能应该一致，无缺漏。  5、测试不支持横竖屏切换的页面，在重力感应下会否切换（测试时应开启重力感应、屏幕切换设置）。  6、关闭手机“自动旋转屏幕”设置后，支持切换的页面是否还能切换。  7、对于用户自定义的横屏、竖屏，测试其不随重力感应而切换变化。 | | |
| **二级**  **用例** | 8、不同屏幕分辨率下的横、竖屏界面兼容性测试。  9、清缓存、清内存后是否能正常展示、使用。 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 暂无 | | |

#### 2.6上传下载文件测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 上传下载文件测试 | **整理人员** | 宪敏、唐晓文 |
| **简介** | 测试Android应用中上传下载文件测试 | | |
| **详细技术说明** | **上传下载文件功能简介：**  上传：先通过用户选择提交文件，传输文件保存到服务器端，然后把相应信息写入数据库中。  下载：每次到下载页面时，从数据库中读出文件路径，根据路径在服务器目录中寻找，找到后传给手机客户端（传的方式又有：整体传输、分段传输、断点续传）。  **Android手机应用上传文件：**  上传一般为前台操作，上传时显示进度圈，页面模态化，其他功能不能操作。  上传时能否选中本地文件（有无SD卡访问权限，能否扫描出可上传文件），是调系统应用选择文件还是自己开发选择页面（扫描相应文件展示在页面中供用户选择上传）。选择好文件后，把文件打开加入内存缓冲区，再建立服务器链接，逐步上传直到完成文件，关闭服务，销毁缓冲区临时对象。重要的大文件上传最好有断点续传，避免意外造成的中断。  **Android手机应用下载文件：**  下载文件一般是在后台异步处理，可以在通知栏中显示下载进度也可以在应用内部开一个页面来显示，下载完成后从通知中点击进去就能见到下载文件。下载中断后能否断点续传。  视频文件下载比较特殊和特殊，大多数选择分段下载，这就涉及到后面下载完成后的重组、下载过程中的丢包处理。  下载的形式：可以在web view或浏览器中点击链接下载，可以点击应用页面中的下载按钮下载（这种一般可以暂停、重启下载）。  下载后的保存地址：一种是默认、一种是让用户指定（在手机上一般很少让用户指定地址，一般是保存在应用的专属文件夹）。 | | |
| **一级**  **用例** | 1. 正常上传、下载文件成功。 2. 使用通知来显示下载进度的，进度条显示正常，点击通知能够跳转到下载文件页面。 3. 在指定页面显示下载进度的，进度条显示正常，暂停下载再重启下载，进度条显示正常，且能继续接着下载，下载完成后能在下载列表中打开文件。 4. 上传后的文件是否按规则重命名。 5. 上传、下载的文件存放路径是否正确。 6. 上传、下载后的文件能正常打开使用，打开后无乱码。 7. 没有选择文件，提交空文件上传。 8. 限制上传文件类型，判断文件是否为合法文件 9. 限制上传文件大小，上传超大文件是否会有内存溢出。 10. 重复上传同一个文件（文件名相同）是否成功。 11. 上传、下载文件失败后，给出相应提示。 12. 上传多个文件时，部分失败后的处理，是否回滚（例如5个图片上传成功3个，2个失败；上传文件失败但是文件描述信息已存入数据库）。 13. 检查断网时候上传、下载的提示内容。 14. 上传过程中，按Home键切换到桌面，再切换回应用，还能继续上传成功。 15. 上传过程中，按返回键，取消上传。 16. 上传过程中，页面应该为模态，其他功能操作用户应不能操作。 17. 下载过程中，按Home键、返回键、电源解锁键，均不会对下载造成影响。 18. 没插SD卡的时候测试文件下载，是否给出相应提示。 19. SD卡被拔出后，进入下载页面和已下载文件列表页显示仍然正在下载项、进度和已下载项。 20. 手机内存不足、SD卡空间不足时，下载时能否给出提示。 | | |
| **二级**  **用例** | 21、程序中有未上传、下载完的数据，断网后重新连网是否能继续上传、下载成功。手动删除了未下载完成的文件，程序是否会自动重新下载。  22、不同系统版本下的兼容性测试。  23、上传下载过程中，清缓存、清内存后上传、下载是否能正常进行。 | | |
| **三级**  **用例** | 24、下载文件时，服务器是否存在一个存储目录文件数过多，造成访问速度下降的问题。  25、在上传、下载过程中热插拔SD卡（拔出时暂停或提示，插入后继续下载）。  26、多用户同时上传，测试上传性能、速度是否有影响。  27、同一个客户端持续上传多次文件，内存占用是否异常，是否会造成内存溢出。  28、多个客户端同时上传，服务器能否支撑最大多少用户同时上传（能建立多少链接、上传速度可接受）。  29、多个用户同时下载，服务器能支持多少用户同时下载（能建立多少链接、下载速度可接受）。 | | |
| **补充**  **信息** | 暂无 | | |

#### 2.7启动退出功能测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 启动退出功能测试 | **整理人员** | 向兴菊、唐晓文 |
| **简介** | 测试Android应用启动、退出功能 | | |
| **详细技术说明** | **Android应用的启动：**  当我们从手机桌面点击启动一个应用时，Android应用程序框架就会去各个应用的配置文件中去匹配相应的应用名称，找到后launcher就会默认启动应用中的MainActivity这个Activity（也就是主页面）。  一般在启动应用时，特别是首次启动应用时，会做一些初始化处理，如果处理不好会造成加载时间长甚至黑屏等问题，所以我们在测试过程中需要分析首次启动应用时，应用会做哪些动作，每个动作又会存在或引发怎样的问题。  一个Android应用在启动时会做以下几类动作：  1、检查版本更新。  2、获取页面上服务器动态数据，加载页面。  3、获取定位信息、手机信息、用户信息等，传输到服务器。  4、自动登录。  以上动作无论失败与否都不应该影响应用正常打开启动，或给予用户任何感知。  首次启动耗时也是判断用户体验的重要指标。二次启动一般是指退出应用后，程序还在内存中，这时再启动会比第一次快。（一般会拿二次启动的时间和首次启动时间做对比，看是否需要优化。）  为了不在启动时初始化加载内容使得页面阻塞，一般采用如下方法启动：  1、将onCreate中初始化的内容，移动到线程中做初始化，加载等  2、初始化完成之后，通过Handler发送消息。  3、Hander中收到消息后，再初始化完整界面。  （即先加载部分界面，待获取到完整数据或做完要做的处理动作后再加载完全部界面）  除了上面提到的在桌面上启动应用外，还有部分应用有开机自启动程序。启动的测试也包括这类程序的启动。开机自启动需要系统权限的许可。开机自启动程序主要是为了监听接收服务器传来的push消息等。  **Android应用的退出：**  一般退出应用的方式有：在首页按返回键退出；长按Home键在后台程序管理中关闭应用程序。  退出应用有如下两种实现方式：  1.退出方法里不使用killProcess直接finish()。运行程序，点击返回键退出，从设置里看该应用程序，显示可以“强制停止”，说明程序仍在运行。  2.退出方法里先finish()，然后使用killProcess。运行程序，再点击返回键退出，从设置里看该应用程序，“强制停止”按钮不可用，说明程序已经停止运行。  （即如果想完全销毁关闭程序则选用第二种。第一种方式在Android中比较常用，这样退出的程序不占用CPU但是还是在占用内存，好处就是下次启动能更快） | | |
| **一级**  **用例** | 1. 从桌面点击应用图标正常启动应用。 2. 点击返回键退出应用，检查后台是否还在存在该应用。 3. 点击Home键退出后，再从桌面点击应用图标或长按Home键从后台点击应用，都应该打开离开前的界面。 4. 在应用中点击“退出”菜单退出。 5. 从手机的“设置 – 强制停止”中退出应用后，再打开应用，看是否能启动。 6. 点击widget插件启动App 7. 强制退出再启动 8. 开机后检查，开机自启动程序是否启动，功能是否正常。 | | |
| **二级**  **用例** | 9. 当没有开机自启动权限或被限制开机自启动时，开机后程序是否启动或报错。  10. 在不同品牌的手机上测试应用启动兼容性以及首次启动耗时、二次启动耗时。（可使用Testin平台测试，具体方法请参见测试工具云平台测试章节） | | |
| **三级用例** | 11. 当手机剩余内存较小的时候，启动应用。确定应用能在系统资源紧张时启动。 | | |
| **补充说明** | Testin云平台运行见测试工具相关章节。 | | |

#### 2.8编辑框校验测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 编辑框校验测试 | **整理人员** | 宪敏、唐晓文 |
| **简介** | 测试编辑框各种校验 | | |
| **详细技术说明** | **测试功能简介：**检验文本内容是否符合要求和规范（不能输入不合法内容或恶意内容）。  **Android中的文本控件：**  1、文本框（Text View）：只读显示控件，用于文本显示。  2、编辑框（Edit Text）：常用的编辑框，用于文本编辑。与网页中的编辑框相似，可以设置一些限制属性，就可以无需考虑数据提交时的校验问题。和网页中的文本框有所不同的是，多行输入的文本框可以输入图片。  这两个控件的关系是：Edit Text继承的是Text View，所以TextView里面有的属性，Edit View里面也有。  **文本编辑框（EditText）控件中与内容校验相关的属性：**  android:hint="请输入数字！"//设置显示在空间上的提示信息  android:numeric="integer"//设置只能输入整数，如果是小数则是：decimal  android:singleLine="true"//设置单行输入，一旦设置为true，则文字不会自动换行。  android:password="true"//设置只能输入密码  android:digits //设置只接受某些数字  Android：singleLine//是否单行或者多行，回车是离开文本框还是文本框增加新行  android：numeric //只接受数字  android：phoneNumber //输入电话号码  android：editable //是否可编辑  android:autoLink=”all” //设置文本超链接样式当点击网址时，跳向该网址。  android:autoText如果设置，将自动执行输入值的拼写纠正。  android:digits设置允许输入哪些字符。如“1234567890.+-\*/% ()”  android:drawableBottom在text的下方输出一个drawable，如图片。如果指定一个颜色的话会把text的背景设为该颜色，并且同时和background使用时覆盖后者。  android:drawableLeft在text的左边输出一个drawable，如图片。  android:drawablePadding设置text与drawable(图片)的间隔，与drawableLeft、drawableRight、drawableTop、drawableBottom一起使用，可设置为负数，单独使用没有效果。  android:drawableRight在text的右边输出一个drawable，如图片。  android:drawableTop在text的正上方输出一个drawable，如图片。  android:editable设置是否可编辑。  android:editorExtras设置文本的额外的输入数据。  android:ellipsize设置当文字过长时,该控件该如何显示。有如下值设置：”start”—?省略号显示在开头;”end”——省略号显示在结尾;”middle”—-省略号显示在中间;”marquee” ——以跑马灯的方式显示(动画横向移动)  android:freezesText设置保存文本的内容以及光标的位置。  android:hintText为空时显示的文字提示信息，可通过textColorHint设置提示信息的颜色。  android:imeOptions附加功能，设置右下角IME动作与编辑框相关的动作，如actionDone右下角将显示一个“完成”，而不设置默认是一个回车符号。  android:imeActionId设置IME动作ID。  android:imeActionLabel设置IME动作标签。  android:inputMethod为文本指定输入法，需要完全限定名(完整的包名)。例如：com.google.android.inputmethod.pinyin，找不到会报错。  android:inputType设置文本的类型，用于帮助输入法显示合适的键盘类型。  android:linksClickable设置链接是否可点击。  android:password以小点”.”显示文本  android:phoneNumber设置为电话号码的输入方式。  android:privateImeOptions设置输入法选项。  android:scrollHorizontally设置文本超出TextView的宽度的情况下，是否出现横拉条。  注：有很多校验功能是无法用属性来实现的，就要具体情况具体分析，例如：判断输入的注册用户名是否已存在，修改密码时输入的原密码是否正确，就需要访问远程服务器查看。这些都属于业务级别的校验了。  **Android中支持的可输入内容有：**  与其他非智能机不同，Android手机上可以输入的内容很多，基本上电脑中可以输入的他都可以输入。英文、中文、英文标点符号、中文标点符号、全角半角、其他国家语言、特殊字符（注音、制表符、希腊符号、偏旁部首、拼音、序号、数学符号）。测试中要特别注意特殊字符的输入保存，避免乱码产生。在编辑中还支持选中部分、选中全部、复制、粘贴、剪切动作。（二次输入确认密码时，不应该允许复制）  **其他与校验无关但在编辑文本上会遇到的处理方式：**  1、有些产品会要求输入完毕且判断输入内容符合后，自动关闭软键盘。  2、软键盘弹出时，输入框应上移不被输入法遮挡，应让用户完整的看到整个编辑框里的内容。 | | |
| **一级**  **用例** | 1、根据需求校验编辑框，常见的内容校验有如下几种：非空校验、纯整数校验、小数校验、日期校验、邮箱格式校验、电话格式校验、手机格式校验、邮编格式校验、网址校验、敏感词校验、自定义正则表达式校验。  2、测试编辑框字数长度校验，测为空、最大值、最小值，计算字数正确，全半角计算正确（不过多数手机输入法没有全角半角切换。例如：讯飞、百度）。  3、在多行编辑框中，输入内容过多后，能否自动显示下拉滚动条，能否自动换行。  4、在编辑框中输入特殊字符（每类字符随机输入一个或多个），保存后再查看是否显示为乱码。  5、在编辑框中输入图片，图片插入后位置是否合适（居左、居右、居中，根据产品要求来）。  6、输入框内光标聚焦后，点击手机键盘的回车键，查看输入框有没有无限撑大。  7、在输入框中输入“TAB”、“回车”、“空格”是否能输入（或者有限制不能输入）。  8、业务校验，输入新注册用户判断是否已存在、输入确认密码是否正确、输入原密码是否正确。  9、对于制定了输入类型的编辑框，测试获取焦点后是否打开的是指定的输入类型。（例如：该编辑框只能输入数字，输入法打开后默认就是数字输入界面，或者只能显示数字输入界面，不能切换）  10、编辑框中输入的网址、是否能点击跳转到浏览器打开相应网页。（如果产品要求不跳转，则测试它无法跳转）  11、单行文本框输入过多文字后，能否自动显示横向滚动条。  12、在校验编辑框录入字数时，实时显示的已输入字数要能被用户看到，且增减计数正确。 | | |
| **二级**  **用例** | 13、在需要到服务器端获取数据校验输入内容时（例如：校验用户名是否唯一，原密码是否正确），无网络或无网络访问权限，是否会给出提示（至少不会导致卡死或报错）  14、对于指定了输入法的编辑框，测试是否能打开相应输入法。如果没有安装相应输入法会不会报错（例如：指定打开百度输入法，但是手机上没安装百度输入法）。 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 暂无 | | |

#### 2.9外部接口调用测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 外部接口调用测试 | **整理人员** | 苟红梅、唐晓文 |
| **简介** | 测试Android应用中所涉及调外部接口调用功能点的测试； | | |
| **详细技术说明** | **测试对象简介：**  接口测试主要用于检测外部系统与所测系统之间以及内部各系统之间的交互点。测试的重点是检查数据交互、传递、和控制管理过程以及系统间的相互依赖关系等。  可分为：内部接口和外部接口；  内部接口：是指系统内部模块与模块、类与类之间消息（或数据）传递的测试；如：导入、导出；  外部接口：即调用子系统、外部第三方系统或功能；如支付功能，调用银联接口、或支付宝、快钱等。  根据发布方式不同外部接口又分为：基于soap协议的Web Service方式、Hessian方式、rest方式。（各类接口都会约定相应的参数和xml、Jsion文件格式，用于传参和解析回传内容，测试需要检验是否各类文件是否符合设计格式）  在Android应用中我们会经常涉及到调用外部接口实现一些功能，例如：查询外部数据用于页面展现、调用外部积分接口实现积分。也许我们的应用只实现了一个个承接数据的容器，所有业务数据的获取和更新都是使用接口实现，这样我们应用的质量和性能就很大程度的依赖于接口质量和性能。  在Android应用中如果有功能是通过外部接口调用实现的，测试时需要分为两部分测试。  一是测接口：如果接口是我方开发的则需要单独针对接口做测试，如果不是我方开发则由提供方并保证接口可用，我方只做联调测试；  二是测调用接口的功能：在Android应用中直接测试调用了外部接口的功能。  **Android应用中是如何实现外部接口调用的：**  首先需要有互联网访问权限，真正实现封装发送、接收数据的是Android框架底层的工作。我们只需要把参数传对，把获取到的内容解析正确展示即可。不同的接口可能底层协议不同，但是都是同样的作用结果。在这整个过程中传值、大数据量及中文传输都有可能出现问题。接口调用一般都会设置超时时间，为避免应用服务器无响应的后续操作。  **在测试外部接口时一般会向开发人员确认如下问题：**  1、远程接口是应该抛出异常，还是封装返回值？ （一般不返回。不返回的话测试时就要注意，需要查看接口调用日志才能确定它是否处理失败抛异常）  2、接口需不需要容错处理，还是交给调用方容错。（一般两边都会做一定的容错，根据开发详设来做验证。）  3、接口升级的考虑，怎么做到客户端容易升级？ （最好接口向后兼容，接口改动后之前调用该接口的功能都不需要改造，但仍要做基础功能验证）  4、读写接口（readService、writeService）是否分离？（readService通常是不需要启用事务的，而writeService则需要分离它们两个可以防止在不必要的时候启用事务）  5、参数Map、还是类比较好？（类比较好）  6、是否有鉴权和身份验证处理？（需要鉴权的话，还要对鉴权功能做测试）  7、是否记录调用日志？（测日志记录准确性） | | |
| **一级用例** | 1. 检查接口参数传递时有校、无效、重复的参数值及数据类型、参数数量、格式进行验证； 2. 检查接口传递的值生成的报文、查询数据库，确保事务数据正确的保存； 3. 检查接口返回的报文处理结果、数据类型、返回报文格式等是否正确； 4. 检查接口的错误处理：在测试接口提交数据或获取数据时，尝试在处理过程中中断事务，检查其结果情况； 5. 在测试调用接口提交数据或获取数据时，尝试中断服务器网络连接，检查系统能否正常处理错误； 6. 在测试调用接口提交数据或获取数据时，尝试中断服务器与外部第三方的连接（包括：客户端场景中断、外部接口服务中断、双方均中断），检查系统在该情况下能的容错情况； 7. 在向外部接口传递数据过程中由于各种原因服务终断，检查是否存在部份数据传输成功，部份失败的情况。、 8. 测试接口鉴权、身份验证是否正确（测认证通过/不通过两种情况） | | |
| **二级**  **用例** | 1. 接口日志记录是否正确（时间、内容、异常、错误原因等） | | |
| **三级**  **用例** | 10、测试接口的响应时间、处理能力（性能测试） | | |
| **补充**  **信息** | 接口测试可以使用接口测试工具：Soap UI(只能测web service接口)、Jmeter（各种接口都可以）进行测试。（工具具体使用方式和安装包请参见测试工具库） | | |

#### 2.11视频播放测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 视频播放测试 | **整理人员** | 苟红梅、唐晓文 |
| **简介** | 测试Android应用中视频播放的测试 | | |
| **详细技术说明** | **功能简介：**  Android应用中播放视频一种是播放本地视频、第二种是播放网络视频。我们这里讨论的主要是网络视频的下载和播放。在Android手机中播放视频可以在网页或网页容器控件中播放，也可以在专门的播放控件中播放。  **手机上常见的视频格式：**  严格来说视频格式其中涵盖了两个概念，一个是封装格式，一个是编码格式，我们经常说一个视频文件时AVI格式或者MP4格式指的都是封装格式，而非编码格式。  常见封装格式  AVI：微软在90年代初创立的封装标准，是当时为对抗quicktime格式(mov)而推出的，只能支持固定CBR恒定比特率编码的声音文件。  FLV：h针对于h.263家族的格式。  MKV：万能封装器，有良好的兼容和跨平台性、纠错性，可带 外挂字幕。  MOV：MOV是Quicktime封装（苹果推出）。  MP4：主要应用于mpeg4的封装。适合所有手机，特别是带存储卡的手机，优点：图像清晰。文件大小适中。  RM/RMVB：Real Video，由RealNetworks开发的应用于rmvb和rm 。  TS/PS：PS封装只能在HDDVD原版。  WMV：微软推出的，作为市场竞争。  3GP：适合所有手机，特别是内存小的手机。优点：文件小。  （AVI,RMVB，WMV适合有存储卡,带操作系统的智能手机。优点：视频质量好，缺点：文件较大。）这些就是网络上比较常见的封装格式以及其特点。  大多数使用手机自带的摄像功能拍摄出来的视频默认格式为：MP4  简单的来说一个视频的画质取决于四个因素，视频分辨率和帧率，编码格式和码流。分辨率影响视频文件画面的大小，帧率影响画面的流畅性，编码格式和码流影响画面质量。在手机上播放一个视频想要取得比较好的画质首先视频分辨率不能低于屏幕分辨率，否则的话视频不能点对点的充分利用好屏幕，那么画质必然会有所损失。有些视频播放应用甚至还会报错无法播放。在一些恶劣的网络环境下，降低码率将有助于提高视音频的连贯性和实际效果。  通过安装软件应用Android手机几乎支持所有视频格式（avi mkv rmvb mp4 mov wmv asf wma wav mpg ts mpa dvd au mp3 mid ivf aiff ogm ogg cdaflic d2v aac roq flac drc dsm swf pls pmp）。  **影响Android手机应用视频播放效果的因素：**   1. 视频软件或包含视频的软件所运行的操作系统版本； 2. 视频在终端运行时所支持的网络带宽； 3. 视频在终端运行时所支持的内存大小； 4. 视频文件的大小、分辨率； 5. 在不同性能的手机上播放（低端机速度慢）；   **技术上可能会做的一些处理：**  1、分段下载、边下边播。  2、下载过程中丢包重传。  3、根据网速调整传输速率。  4、支持拖动跳越下载（下载中拖动，直接跳过部分分段下载指定分段）。  5、支持全屏切换、横竖屏切换。  6、支持断点续传。 | | |
| **一级用例** | 1. 检查视频在终端的播放画质是否清晰； 2. 检查视频在终端的播放画面是否流畅； 3. 检查视频在终端的播放声音是否清晰清脆，且声音和画面同步； 4. 检查视频在播放过程中暂停、播放、前进、后退等功能按钮是否正确（如：播放过程中暂停、播放过程中快进、播放过程中后退等）； 5. 检查视频在播放、快进、后退过中缓冲进度条和播放进度条的变化是否正确，时间计算显示是否正确； 6. 检查视频在播放过程中点击画面能否暂停，暂停的状态点击画面后能否继续播放； 7. 检查视频执行全屏、最小化、横竖屏切换后视频是否继续播放； 8. 检查向左、向右滑动屏幕是否能快进、后退播放视频。 9. 测试工具栏、状态条在全屏播放时的显示和隐藏（XX秒不触摸屏幕后，自动隐藏）。 10. 视频下载或缓存是否保存在指定路径。 11. 对于分段下载的视频，下载完成后，能否正确的合成播放；分段丢失或者被手动删除后是否给出提示（例如：文件已被损坏无法播放）。 12. 如果服务器端文件格式类型多样，还要测试各种类型的视频文件是否能在客户端流畅播放。 13. 测试手机上无SD卡时，在线播放视频；点击播放列表播放视频。 | | |
| **二级**  **用例** | 1. 在不同的品牌终端上播放视频，视频能否正常运行、执行最小化、最大化、暂停、播放等各功能按钮是否正常； 2. 在不同分辨率下播放视频，能否正常播放显示、画质清晰不变形。 3. 在不同的品牌手机上双击视频画面能否暂停、继续播放； 4. 视频在终端播放过程中，突然来电、来短信、邮件等对视频的播放处理检查。 5. 在视频处于播放过程中，切换终端到手机桌面或做其它动作（如：查看短信、接听电话）按Home键将视频切换后台运行状态，检查视频的处理； 6. 在视频播放过程中，断开网络检查视频在无网情况下的处理，重新接上网络后是否能继续播放； 7. 限制网络访问权限的时候，看应用如何处理。 8. 限制SD卡读写权限时，能否正常下载播放视频，或给出合理提示。 9. 使用2G、3G、wifi不同网络及不同运营商网络播放视频，检查检查视频在不同网络及运营商网络下的运行情况及友好处理； | | |
| **三级**  **用例** | 21、在手机SD卡剩余容量不足的情况下，下载或播放视频能否给出提示。  22、测试在播放过程中是否存在内存泄露。  23、测试服务器端和带宽支持多少客户端并行播放。 | | |
| **补充**  **信息** | 暂无 | | |

#### 2.12音频播放测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 音频播放测试 | **整理人员** | 苟红梅、唐晓文 |
| **简介** | 测试Android应用中的音频播放功能 | | |
| **详细技术说明** | **测试功能简介：**  Android应用中的音频播放一般具有如下功能：暂停、播放、上一首、下一首、循环播放模式选择、歌词同步显示、后台播放、插拔耳机的播放暂停、拉动进度条定位、边下边播。  最常见的例如：酷狗音乐、酷我听书等应用。  **音频有哪些格式：**  ----------------常见格式--------------------  MIDI：MID文件格式由MIDI继承而来。  MP3：是一种有损压缩格式。  WMA：微软开发，支持流技术，即一边读一边播放。  WAV：微软开发，标准格式的WAV文件和CD格式一样。  RealAudio系列（RA、RM、RMX等）：主要适用于在网络上的在线音乐欣赏，现在大多数的用户仍然在使用56Kbps或更低速率的Modem，所以典型的回放并非最好的音质。有的下载站点会提示你根据你的Modem速率选择最佳的Real文件。其格式的特点是可以随网络带宽的不同而改变声音的质量，在保证大多数人听到流畅声音的前提下，令带宽较富裕的听众获得较好的音质。  ----------------不常见格式-----------------  OGG格式：特点是支持多声道、免费、开源。  APE格式、FLAC格式、ACC格式  CD格式：\*.cda格式  一般测试选用常用格式测试，如果服务器端的音频文件统一只有一种格式测无需测多种格式兼容。  **Android手机上音频播放测试注意项：**   1. 音频播放时，有哪些意外事件可以打断播放，时间结束后又自动播放的。（插拔耳机、来电显示接完电话后挂机、来短信铃声、来通知铃声） 2. 在播放过程中、清内存、清缓存、清数据，都不能影响当前音乐播放。 3. 在找不到播放资源时要做处理提示（不管是本地资源、网络资源）。 4. 下载过程中如果有丢包情况，要有补救措施（至少在下载完成时要检查下载包的完整性）。 | | |
| **一级用例** | 1. 检查音频在应用中的正常播放（进度条显示、歌曲时间显示、缓冲条显示、歌词同步显示、歌曲名显示、播放按钮显示为||暂停功能，循环模式显示、歌曲时长显示与歌曲详细信息里的时长一致）； 2. 检查音频在线播放（边下边播）时，是否清晰流畅，没有卡带、跳跃等现象，进度条和缓冲进度条显示正常，时间进度显示正常，播放进度条总慢于缓冲进度条，缓冲中要有缓冲提示； 3. 播放本地已下载音乐时，是否清晰流畅，播放进度条显示正常，时间进度显示正常、无缓冲进度条、无缓冲提示。 4. 检查音频在播放过程中暂停、播放等功能按钮是否正确（如：播放过程中暂停，暂停后播放时长是否仍然在走动更新）； 5. 检查音频（本地音频和网络音频）在播放过程中，拖动进度条，是否流畅，能准确定位（首、尾、中），定位时当前歌曲时间也随定位变更显示。 6. 检查音频在播放时间的变化是否正确与实际时长相同（时间显示准确）； 7. 检查音频播放、停止、暂停不同状态下，功能按钮图标在状态流转时的变化； 8. 检查音频在单曲循环、顺序播放、列表循环、随机播放等模式能否正常生效、流转、名称和显示的图标对应正确。 9. 检查下载成功的音频，是否保存进指定路径，下载完成后播放音频是否完整。 10. 检查音频在网络不稳定的时候下载（不断关闭开启网络测试，关闭网络暂停，重启网络后点击播放键继续），下载完成后在本地播放流畅清晰无丢包情况，音频时长与下载文件一致； 11. 在无网络的情况下，在线播放音频是否给出相应提示。 12. 测试播放不同格式的音频（MIDI\MP3\WMA\WMV\RA—只测主流格式）播放，均能支持播放。 13. 在播放音频过程中按返回键直到退出应用、按Home键、按锁屏键、屏幕解锁，均不会影响音频正常播放。 14. 在音频播放过程中（无耳机），插入耳机-音频继续播放，拔出耳机-播放停止，再插入耳机-继续播放。 15. 在音频播放过程中，从“文件管理”中进入应用文件夹，删除SD卡中正在播放的那个音频文件，仍旧能够继续播放。直到播放完毕后，下次再播放则提示找不到该文件。 16. 音频在播放过程中，突然来电、来短信、来通知，音频中断播放，等事件的提示音播放完毕后音频继续播放。 17. 在音频播放过程中，拨打电话，音频播放自动暂停，待打完电话挂机后，音频自动继续播放。 18. 在音频播放过程中，点击其他音频文件，能够停止正在播放的音频，播放新的音频文件。 19. 测试手机在无SD卡的情况下，在线播放音频；点击播放列表播放音频。 | | |
| **二级**  **用例** | 1. 在不同的品牌终端上播放音频，音频能否正常运行、声音、音质是否清晰； 2. 在不同网络下（2G\3G\WIFI\移动、联通、电信）测试在线播放音频是否流畅。 3. 在无网络访问权限下，测试在线播放网络音频。 4. 在无SD卡读写权限下，测试在线播放网络音频和播放本地音频。 5. 测试播放大音频文件，是否存在内存占满无法播放的问题。 | | |
| **三级**  **用例** | 25、测试长时间持续运行音频播放（例如：依次播放1万首歌曲、同一首歌循环播放24小时等），无资源占用增加，正常运行无报错，内存使用正常。  26、在线播放的服务器端性能测试，测试最大能同时支撑多少用户在线听歌或下载。 | | |
| **补充**  **信息** | 暂无 | | |

#### 2.13后台服务测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 后台服务测试 | **整理人员** | 陈晓红 |
| **简介** | Android后台服务即service，是android系统的四大组件之一，没有可视化界面，运行于后台的一种服务程序。 | | |
| **详细技术说明** | **Android后台服务特点介绍：**  Android后台服务无界面、能长时间运行，有自启动能力，需要始终保持，负责处理一些隐秘的任务，例如后台播放音乐。Service需要别人启动它，关闭可以由别人关闭也可以自身执行关闭。而一些安全软件，如360等，会有强制结束进程的功能，如果不做保持，就会被其杀掉。  当系统内存不足需要杀死某些进程或服务时，它比处于前台的应用优先级低，但是比后台的其他应用程序优先级高。但是被系统杀死后，在系统资源恢复后又会自动启动运行。  **Android后台服务保持机制：**  利用Android系统广播每分钟一次，在系统广播里每分钟检查一次Service的运行状态，如果已经被结束了，就重新启动Service。如果实现了这个动作，开机启动也就自然实现了，无需再开发。  **Service的生命周期：**  只继承了onCreate(),onStart(),onDestroy()三个方法。  当第一次启动Service时，先后调用了onCreate(),onStart()这两个方法。  当停止Service时，则执行onDestroy() 方法。  这里需要注意的是，如果Service已经启动了，当再次启动Service时，不会再执行onCreate()方法，而是直接执行onStart()方法。 | | |
| **一级**  **用例** | 1. 服务能正常启动(开机自动启动，手动启动)； 2. 服务的本身功能能够正常实现（具体根据需求测试，例如：定时发送广告等）。 3. 能够长时间保持（长时间待机或关闭主应用程序，后台服务依然运行）； 4. 手动强制关闭服务，过一段时间（1分钟左右）后能自动启动，启动后功能运行正常；（如果是很重要的服务还要考虑暂停期间会造成什么后果） | | |
| **二级**  **用例** | 5、后台服务功能需要网络运行，当无网络的情况下会不会报错，网络重新连接后又能否继续正常运行。 | | |
| **三级**  **用例** | 6、测试启动一定时间内占用内存情况、耗电量情况（使用工具监控，内存占用不会随着运行时间的加长而越来越大）  7、是否有会造成重复启动服务的点，重复启动后是否会对程序运行照成影响。 | | |
| **补充**  **信息** | 监控工具可以使用DDMS（安装包和具体使用方法可以到测试工具库获取） | | |

#### 2.14内置广告测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 内置广告测试 | **整理人员** | 唐晓文 |
| **简介** | 一般应用中内置的广告为第三方平台开发的。 | | |
| **详细技术说明** | **一般Android应用实现广告功能的方式有三种：**   1. 内置广告条 2. 弹出广告提示框 3. 推送广播通知（这方法就是PUSH通知的方法，在此不赘述）   （各种更变态的强制广告展现方式在不断涌现，欢迎补充）  第三方广告条平台通过内置广告条的点击率和开发者分成，每个平台的分成规则不同。所以我们需要保证广告条的可见可用。广告条程序由第三方广告条平台提供实现。  **添加第三方广告条的方法：**  1、申请获取第三方广告条分成账号。  2、把第三方提供的广告条SDK加入应用工程。  3、修改AndroidManifest.xml文件加入广告条需要的一些权限。  4、在AndroidManifest.xml文件按格式加入申请来的账号。  5、在显示广告条的界面加入广告条控件。  注：广告条的更新由提供者控制，不属于我们的测试范围。 | | |
| **一级**  **用例** | 1. 测试广告能正常启动，显示正确。 2. 与第三方广告平台联调、确保点击次数、计费等记录正确（一般都会提供联调地址）。 3. 检查从第三方广告平台申请的账号在配置文件中是否填写正确。（填写不正确收不到钱） 4. 页面跳转，广告条仍可见可用。 5. 在断网的情况下，广告条可用可见，如不可见也不能影响应用中的其他功能使用。 | | |
| **二级**  **用例** | 6、限制网络访问权限时（或限制其他广告条需要的权限），广告条仍可见可用。 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 暂无 | | |

#### 2.15登录测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 登录测试 | **整理人员** | 宪敏、唐晓文 |
| **简介** | 测试在Android应用中的登录功能 | | |
| **详细技术说明** | **Android应用登录功能的需求分析：**  1.实现用户名和密码的输入  2.提取用户名和密码信息  3.查询远程用户信息判断输入账号是否正确  4.登录时有进度条  5.超时处理  6.登录成功跳转  7.只允许单点登录  8.登录后只要不注销，下次进来自动登录  9.登录成功后保持登录  **Android应用中登录中的一些特殊处理：**  1、网页上只能使用电脑键盘输入，不需要控制输入法。Android应用中使用软键盘输入，为了增加用户体验可以在输入法上做定制化显示。（例如：键盘上只显示需要输入的数据类型、“完成”按钮可以设置为“下一个”把焦点移到下一个。甚至实现不同系统输入法切换。）  2、登录后如何保持住登录状态：这个可以用session机制来处理，当用户连接到服务器，服务器可以生成一个session返回给 手机客户端，客户端保存这个session,每次发送操作都把session作为参数发给服务器，服务器可以判断这个session是否过时和存在。  3、如何实现自动登录：一次登录成功后，就把用户名和密码保存在客户端本地，下次打开客户端则去读取，如果读取失败则给出提示让用户重新输入。若本地支持保存有多个账号信息时，还需考虑自动登录哪个。 | | |
| **一级**  **用例** | 1. 页面title、文字描述是否正确； 2. 输入正确的用户名和正确的密码登录。 3. 登录客户端成功后，退出应用再次打开客户端，自动登录成功显示为已登录状态。 4. 输入正确的用户名和正确的密码，但未区分大小写 5. 输入正确的用户名和错误的密码，出现密码错误的提示并清空密码输入框。 6. 输入错误的用户名和错误的密码，给出相应提示。 7. 不输入用户名和密码/或均为空，直接点击登录，给出正确提示。 8. 只输入用户名，密码为空/或为空格/或为回车/或为tab，出现“请输入密码”提示框，用户名为空/或为空格，只输入密码。 9. 检查用户名和密码输入框的字数长度限制（大于、等于、小于）。 10. 输入没注册过的用户名，提示该用户未注册请注册。 11. 点击“取消”按钮或返回上级按钮，取消登录返回上级页面。 12. 检查正常用户名不允许的特殊字符（包括输入法中可以输入的所有类型字符）、空的用户名，系统（操作系统和应用系统）的保留字符。 13. 不合法的密码有：空密码（除有特殊规定的），错误的密码，字符大于密码的限制、正常密码不允许的特殊字符，系统（操作系统和应用系统）的保留字符。 14. 密码输入框输入显示（一般显示为\*或者黑点） 15. 同一个账号只能在一个手机应用中登录，不能多点登录，或者一个登录另一个就自动退出。 16. 测试无网络时登录，是否能给出正确提示。 17. 对于界面有链接的界面，要测试界面上的所有的链接都正常跳转到正确页面。 18. 打开登录界面，自动弹出输入法，且输入法是按照需求定制显示正确。 19. 登录失败次数限制，测试失败次数为n-1，n，n+1三种情况。（n为可接收的失败次数） 20. 当本地保存有多个可自动登录的账号密码时，打开应用时会选择最后退出的账号自动登录。 | | |
| **二级**  **用例** | 1. 页面的默认焦点是否控制在用户名输入框中，并打开输入法。 2. 登录客户端成功后退出应用，手动删除本地保存的账号信息，再次打开客户端则显示为登录页面要求输入用户名密码，不会自动登录。 3. 输入用户名和密码点击登录，按返回键撤销，成功撤销返回上一级页面，且登录动作继续进行直至登录成功。 4. 安全测试。在输入框中做sql注入测试，看是否能绕过登录或操控数据库获取信息。（详情见安全测试章节SQL注入测试方法） | | |
| **三级**  **用例** | 1. 验证登录的次数是否有限制。从安全角度考虑，有些安全级别高的软件会考虑这方面的限制。 2. 测试账号和密码的复制粘贴功能。 | | |
| **补充**  **信息** | 可以使用LR对于登录进行性能测试，测试登录的时间，并发登录的时间等等。  **【免登录功能测试】**  很多应用提供免登录功能，当应用开启时自动以上一次登录的用户身份来使用app  1) app有免登录功能时，需要考虑系统版本差异，要做兼容性测试。  2) 考虑无网络情况时能否正常进入免登录状态。  3) 切换用户登录后，要校验用户登录信息及数据内容是否相应更新，确保原用户退出。  4) 根据MTOP的现有规则，一个帐户只允许登录一台机器。所以，需要检查一个。  5) 帐户登录多台手机的情况。原手机里的用户需要被踢出，给出友好提示。  6) app切换到后台，再切回前台的校验。  7) 切换到后台，再切换回前台的测试。  8) 密码更换后，检查有数据交换时是否进行了有效身份的校验。  9) 支持自动登录的应用在进行数据交换时，检查系统是否能自动登录成功并且数据操作无误。  10) 检查用户主动退出登录后，下次启动app，应停留在登录界面。 | | |

#### 2.16找回密码测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 找回密码测试 | **整理人员** | 陈小红 |
| **简介** | 用户找回密码 | | |
| **详细技术说明** | **手机端应用找回密码，存在根据多种方式找回密码：**  1.若根据手机号验证来的，那么可以通过获取短信验证码来找回密码，从而重新设置新密码；  2.若根据邮箱来的，那么可以通过发送找回密码的邮件来重新设置密码  3.若根据密保问题来的，那么可以通过正确回答密保问题来重新设置密码 | | |
| **一级**  **用例** | 1.验证用户名称是否符合规范(用户名格式是否正确、特殊字符输入、字段为空时)  2.验证发送验证信息后是否能正常收取(是否成功获取验证码短信、是否成功收取找回密码邮件、是否能正常进行密保问题验证)  2.1验证码长度校验  2.2验证码特殊字符校验  2.3 验证码是否正确校验  2.4 邮箱地址校验  2.5 邮件发送是否成功校验  2.6 邮件数据信息校验  2.7密保问题显示是否正确校验  3.验证是否能正常修改密码(通过获取的短信验证码、收取的邮件信息、密保问题回答，是否能正常修改密码)  3.1.密码长度校验  3.2密码大小写校验  3.3密码输入特殊字符校验  3.4邮件链接点击后能正确进入密码重置页面，密码与二次密码是否相同校验。  3.5密码、密保问题为空校验。  3.6强密码校验（测试是否符合强密码的特点）  4.验证修改密码后是否正常登录(修改密码后，是否能正常登录，二次登录新密码是否能正常登录)  5、点击“找回密码”进入确认找回密码页面，点击确认找回，显示滚动条，再按返回键取能立即返回上层页面，且找回密码动作仍继续进行直至完成。  6、测试无网络时找回密码，是否能给出正确提示。  7、短信找回密码时，无2G3G信号时能否给出正确提示。  8、检查确定一次找回密码动作，不会重复发送多条同样的短信或者同样的邮件。 | | |
| **二级**  **用例** | 9、无网络访问权限时，找回密码是否给出正确提示。 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 暂无 | | |

#### 2.17权限相关测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 权限相关测试 | **整理人员** | 陈小红 |
| **简介** | Android系统中权限用来描述是否允许做某件事。Android系统中权限分为普通级别(Normal)，危险级别(dangerous)，签名级别(signature)和系统/签名级别(signature or system)。系统中所有预定义的权限根据作用的不同，分别属于不同的级别。 | | |
| **详细技术说明用例** | 目前Android系统定义了许多权限，通过SDK文档用户可以查询到哪些操作需要哪些权限，然后按需申请。在Android中，用户可以允许系统安装来自"未知源"(也就是非Google官方的，或手机预置市场的)应用程序。签名对于权限认证没有太大的直接效果。但是android系统有些权限是基于签名的。比如：system等级的权限有专门对应的签名，签名不对，权限也就获取不到。应用需要的权限应当在users-permission属性中申请，所申请的权限应是已经被系统或某个应用程序定义过的，否则视为无效申请。同时，使用权限的申请需要遵循权限授予条件，非platform认证的应用程序无法申请高级权限。  Android权限(Permission)  指Android中的一系列"Android.Permission.\*"对象，是本文的中心内容（其他系统刷机解锁权限和Linux文件权限不在此做特别说明，只说和应用开发使用紧密相关的权限）。  Google在Android框架内把各种对象(包括设备上的各类数据，传感器，拨打电话，发送信息，控制别的应用程序等)的访问权限进行了详细的划分，列出了约一百条"Android.Permission"。应用程序在运行前必须向Android系统声明它将会用到的权限，否则Android将会拒绝该应用程序访问通过该"Permission"许可的内容。  比方说，搜狗输入法提供了一个智能通讯录的功能，用户可以在输入联系人拼音的前几个字符，或首字母，输入法就能自动呈现相关联系人的名字。为了实现这个功能，输入法必须声明它需要读取手机中联系人的能力，也就是在相关代码中加上声明"android.permission.READ\_CONTACTS"对象。  应用程序的权限级别大致分为两种：  第一种低级点的(permission的protectlevel属性为normal或者dangerous)，其调用者apk只需声明即可拥有其permission。  第二种高级点的(permission的protectlevel属性为signature或者signatureorsystem),其调用者apk就需要和被调用的apk一样拥有相同的签名。  如何检查权限声明内容，可以查看AndroidManifest.xml文件内容，查找permission章节。  （开发者可以自己定义自己开发的功能的权限或数据权限，以防外部应用调用使用。例如有可能被其他程序调用用于破坏系统。调用原理可以参见本文“外部功能跳转测试”章节）  若想拥有所需的权限，必须在AndroidManifest.xml文件中包含一个或更多的标签来声明此权限。应用程序安装的时候，应用程序请求的permissions通过package installer来批准获取。package installer通过检查该应用程序的签名来确定是否给予该程序索取的权限。在用户使用过程中不会检查权限（但是有部分厂商的定制系统中拓展开发有系统应用可以控制使用途中可以检查权限），也就是说要么在安装时就批准该权限，使其按照设计可以使用该权限;要么就不批准，这样用户也就根本无法使用该feature，也不会有任何提示告知用户尝试失败。当应用试图在没有权限的情况下做受限操作，应用将被系统杀掉以警示。（高版本的Android系统中已经不会强制杀掉程序）  **Android应用权限管理表现在功能上则体现为：**  1、信任该程序或不信任该程序：不信任该程序则会在每次应用中使用到某权限的时候弹出提示框提示该应用正在使用某权限或查看某类数据。  2、如果应用的某项权限被限制了，则会有两种方式，一种是完全限制不能被使用，对于这种类型的权限限制，应用中使用到该权限的功能最好做一下处理；还有一种是每次用到该权限时询问用户是否授权，对于这类权限限制产生的操作流分支，应用程序最好能判断处理。 | | |
| **一级**  **用例** | 1、测试安装包里的AndroidManifest.xml配置文件里声明的权限都正确（没有缺少的权限也没有多余声明的权限）  2、给予相应的权限，功能正常执行。  3、禁止相应的权限，功能能正常提示。  4、安装时信任该应用，在每个使用到权限的地方均不会弹出提示。  5、安装时不信任该应用，在每个使用到权限的地方均会弹出提示，但是提示弹出后不会影响应用功能的完成。  6、安装时不信任该程序，所有权限设置为询问访问。使用功能时，弹出询问框，选择禁止，功能不执行且给出相应提示无权限。  7、安装时不信任该程序，所有权限设置为询问访问。使用功能时，弹出询问框，选择允许，功能成功执行。  8、安装时不信任该程序，所有权限设置为询问访问。使用功能时，弹出询问框，选择允许且选择记住该操作，功能成功执行。且下次再执行次功能无任何提示，成功执行。  9、如果是开放于仅供内部开发的应用间声明的权限必须是签名级的。 | | |
| **二级**  **用例** | 暂无 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 暂无 | | |

#### 2.18Wigdet插件测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | Widget插件测试 | **整理人员** | 唐晓文 |
| **简介** | Widget是微小的应用程序视图，可以被嵌入到其它应用程序中（比如桌面）并接收周期性的更新。是Android 1.5 版本引进的功能。 | | |
| **详细技术说明** | **如何创建删除Widget：**   1. 通过在HomeScreen中长按，在弹出的对话框中选择Widget部件来进行创建。同一个Widget部件可以同时创建多个。 2. 长按部件后并拖动到垃圾箱里进行删除。 3. 安装和卸载都跟随主应用程序的安装卸载。   **Widget的尺寸：**  有六种基于4 x 4 (纵向)或 4 x 4 (横向)单元的主屏幕网格的标准Widget尺寸。  就像把屏幕分成了如下表格块：   |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  |   常用尺寸为：2 x 2，4 x 1，4 x 2，4 x 3  如果创建时当前屏幕空白位置不够放置该尺寸Widget，系统应用会提示“此屏没有足够的空间”。  **Widget的界面控件：**  AppWidget的实现主要涉及到以下3类：  1）AppWidgetProvider 2）RemoteViews——用于界面布局 3）AppWidgetManager——用于管理更新  界面中使用的组件必须是RemoteViews所支持的，目前原生API中支持的组件如下：  FrameLayout 、LinearLayout 、RelativeLayout 、AnalogClock 、Button  Chronmeter 、ImageButton 、ImageView 、ProgressBar 、TextView  如果使用了除此之外的组件，则在Widget创建时会导致android.view.InflateExceptionn异常。  这就导致有一些功能或样式无法实现，如很基本的list或文本编辑框都是无法直接实现的。如果想自定义Widget中的View的话只能通过修改framework来提供相应组件的支持。如果程序使用了不支持的组件，测试时需特别注意。  **AppWidget在实现上的分类：**   1. 带配置活动的Widget（即设置了confiuration Activity的Widget） 2. 不带配置活动的Widget（即没有设置confiuration Activity的Widget）   **AppWidget的生命周期:**  1、创建第一个时: onReceive🡪onEnabled🡪onReceive🡪onUpdate  2、创建非第一个时：onReceive🡪onUpdate  3、删除非最后一个时：onReceive🡪onDelete  4、删除最后一个时：onReceive🡪onDelete🡪onReceive🡪onDisable  5、启动(无confiuration Activity)：情况同1  6、启动（带confiuration Activity）：情况同2  7、拖动时： 无任何事件触发（可以不用测）  8、定时更新：情况同2 9、启动时位置不够：onReceive🡪onEnabled🡪onReceive🡪onUpdate🡪onReceive🡪  onDeleted🡪onReceive🡪onDisabled **注：**  最后一个widget被移除时触发（应做一些清理工作，比如删除一个临时的数据库等，不应该影响下次新建使用） 每次状态的变化会触发onReceive，但该函数一般不需要重写。  最重要的是onUpdate事件，具体执行更新操作。  **Widget的主要设置参数如下:** minWidth: 定义Wdiget组件的宽度  minHeight: 定义Wdiget组件的高度  updatePeriodMillis: 更新的时间周期（定义了App Widget框架调用onUpdate()方法来从AppWidgetProvider请求一次更新的频度。如果不是用来显示实时时间，频率越低越好，每小时不超过一次以节省电源。还可以允许用户在配置中调整这个频率，一些人可能想每15分钟一次股票报价，或者一天只要更新四次。） initialLayout: Widget的布局文件  configure: 如果需要在启动前先启动一个Activity进行设置，在这里给出Activity的完整类名(后面会说到，与一般Activity的实现有些许差别) | | |
| **一级用例** | 1. Widget插件是否随着应用安装成功。 2. Widget插件是否能拖动到桌面创建，创建后是否有数据加载展示。 3. 点击Widget是否能启动主应用程序，手机重启后点击Widget是否还能启动主应用程序。 4. Widget插件是否能拖动删除。 5. APK卸载后Widget是否也被卸载。 6. 换肤操作后，widget皮肤是否自动切换，切换成功后关机重启能否保持。 7. Widget显示数据是否能自动更新，或是否按照配置的特定周期时间更新。如：手机天气的天气信息是否能自动更新，多久更新一次，更新频率是否太频繁，是否达到要求的效果。 8. Widget界面上的所有操作能否响应执行。 9. 清空程序缓存后，Widget是否还能正常工作。 10. 清内存后，Widget是否还能正常工作。 11. 重启手机后，widget是否还在，数据加载是否正确，能否正常工作。 12. 每种尺寸的Widget都是独立的，都要单独测试。 13. 多个不同尺寸的同一Widget，同时加载到桌面，运行正常互不影响，能同时刷新。 14. 加载多个同一尺寸的Widget，操作一个Widget时其他的是否要同步。 15. 后台更新程序是否会被暂停，是否有保护。（例如：远程获取信息展示中途被打断） | | |
| **二级**  **用例** | 1. 程序中未加载完成的数据，Widget是否能继续加载成功。 2. 不同屏幕分辨率下的界面兼容性测试。 3. 不同系统版本下的兼容性测试。 4. 检查断网时候的显示内容。 5. 清缓存、清内存后是否能正常展示、使用。 | | |
| **三级**  **用例** | 21、根据生命周期测试需要特殊处理的事件。例如：全部插件删除后是否要销毁特定内容等。 | | |
| **补充**  **信息** | 暂无 | | |

#### 2.19 注册测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 注册测试 | **整理人员** | 向兴菊 |
| **简介** | 注册测试，包括常见的几大类注册。 | | |
| **详细技术说明** | **手机客户端应用的几类注册方式：**   1. 使用昵称注册   不允许昵称重复、不允许特殊字符、长度有限制、大小写区分、全半角区分   1. 使用手机号注册   手机号格式正确长度限制、自动获取手机号、是否限制只允许中国移动号码注册、不许重复手机号注册。已加入黑名单的手机号不能再注册。能否获取白名单。短信认证，时效问题。   1. 使用邮箱注册   邮箱格式是否正确、长度限制、特殊字符、邮箱认证。  **额外功能测试：**   1. 是否注册送积分。 2. 是否记录注册日志。 3. 注册成功后能否自动登录。 4. 是否有手机号、邮箱黑名单。 5. 之前被加入黑名单的手机号、昵称、邮箱还是否能被注册。 | | |
| **一级**  **用例** | 1. 正常注册（符合任何限制条件） 2. 输入框校验：内容格式、字符类型特殊字符、字符长度、是否必填项校验。 3. 重复注册已存在的用户校验。（唯一标识是否区分大小写、全半角） 4. 密码是否可以复制粘贴，密码是否以\*之类的加秘符号显示，两次密码输入不一致是否提醒。 5. 其他产品需求限制。例如：一个手机号只能注册一个账号之类。 6. 注册成功后，到数据库查看是否已保存，且密码已加密保存。 7. 注册中途退出，或只完成了前面几步没完成全部步骤，是否注册成功。 8. 用手机号注册时，能否自动获取手机号。 9. 曾经被加入黑名单的手机号、昵称、邮箱不能被注册。 10. 注册成功后，自动登录成功。 11. 短信认证是否成功（正确码、错误码、空、发送失败），验证码的时效性。 12. 根据产品逻辑验证邮箱绑定是否成功。 | | |
| **二级**  **用例** | 1. SQL注入测试。 2. 测试日志记录是否正确，，且日志记录失败不影响注册成。 3. 注册后加其他额外动作是否已完成。（例如：加积分等） | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | SQL注入测试详情请参见安全测试模块 | | |

#### 2.20 搜索功能测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 搜索功能测试 | **整理人员** | 向兴菊、唐晓文 |
| **简介** | 应用中涉及到的搜索功能测试 | | |
| **详细技术说明** | **应用中搜索的实现方式：**   1. 直接用SQL访问数据库搜索（搜索本地数据库、远程数据库） 2. 调用接口实现搜索（例如：调用OneSearch接口） 3. 搜索返回多条数据：返回所有搜索数据、搜索后分页首次只返回第一页数据其他数据存在缓存中，等用户操作翻页后再一页页返回。 4. 搜索返回单条数据。 | | |
| **一级**  **用例** | 1. 搜索页面检查，文字显示正确。 2. 使用默认条件搜索（页面中直接点击搜索） 3. 依次修改可选条件搜索，确保所有选择性条件生效。 4. 修改输入条件搜索，确认输入的搜索条件生效。 5. 修改区间条件搜索，确认区间范围搜索生效。（大于、等于、小于三种情况） 6. 组合可选、输入条件搜索 7. 搜索框的输入校验特殊字符、中文、英文、数字（每种类型纯输入一次，最后再组合输入一次） 8. 操作后检查搜索条件及查询结果：查询结果匹配正确、字段显示正确、排序正确、返回条数正确。 9. 检查搜索结果为空时的提示语是否正确。 10. 检查搜索失败的提示语是否正确。 11. 在搜索过程中显示搜索进度条，页面呈现模态不允许操作界面功能。 12. 在搜索过程中按返回键，立即返回上一页，搜索停止。 13. 在搜索过程中按Home键，切换到桌面再切换回搜索界面时继续搜索，得出结果。 | | |
| **二级**  **用例** | 14. 在无网络的情况下搜索，提示语是否正确。  15. 在无网络访问权限时搜索，提示语是否正确。  16、搜索框的SQL注入校验、跨站脚本攻击校验。  17、搜索本地数据库时，数据库被手动删除（清数据）了，此次搜索为空，不影响操作。 | | |
| **三级**  **用例** | 18、搜索远程数据库时，数据库宕机，给出响应提示。  19、一次性搜索返回大量数据时，不会造成内存溢出崩溃。（大数据量测试） | | |
| **补充**  **信息** | 暂无 | | |

#### 2.21 接口测试（服务器端）

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 接口测试 | **整理人员** | 唐晓文 |
| **简介** | 应用中调用的与服务器端进行交互的接口功能测试 | | |
| **详细技术说明** | **Android客户端是如何与服务器交互：**  一般的web系统中会采用Web Service与服务端进行交互。但是在传递大量的数量时，Web Service显得有些笨拙，当然也有一些特殊情况必须要用到，但我们在此主要讨论常用的，其他的接口方式可以参见web测试框架接口测试模块。在移动客户端中，一般采用直接发送HTTP GET或POST请求的方式和服务器端交互、获取数据。实际上也就类似于浏览器和服务器之间的关系，也是基于http协议相互传递数据，根据发过去的URL和携带的一些参数，服务器就知道要向客户端发送哪些数据。  开发人员要做的是把客户端的调用做得更周全些，像浏览器一样发送http请求。所以与web测试相比，BUG就不仅仅是会出现在服务器端，客户端发送请求也可能存在BUG。测接口只是测服务器端，还要到界面上测客户端调用是否正确，两边都正确了才没问题。（在此我们主要讨论接口测试，也就是服务器端的测试）  **接口调用如何实现：**  Android SDK集成了Apache HttpClient模块。  要注意的是，这里的Apache HttpClient模块是HttpClient 4.0（org.apache.http.\*），而不是Jakarta Commons HttpClient 3.x（org.apache.commons.httpclient.\*）  在HttpClient模块中用到了两个重要的类：HttpGet和HttpPost。这两个类分别用来提交HTTP GET和HTTP POST请求。开发人员就是使用这两个类按照以下步骤去发起http请求。  无论是使用HttpGet，还是使用HttpPost，都必须通过如下3步来访问HTTP资源。  1、创建HttpGet或HttpPost对象，将要请求的URL通过构造方法传入HttpGet或HttpPost对象。  2、使用DefaultHttpClient类的execute方法发送HTTP GET或HTTP POST请求，并返回HttpResponse对象。  3、通过HttpResponse接口的getEntity方法返回响应信息，并进行相应的处理。  **注意：**如果使用HttpPost方法提交HTTP POST请求，还需要使用HttpPost类的setEntity方法设置请求参数。Entity又分为两类：一类是StringEntity用于传普通的字符串类参数，另一类是multiEntity用来传结构化的参数和上传文件。在使用测试工具模拟发送请求时，要特别注意这两种方式，不同的方式在Jmeter里需要采用不同的设置，否则接口不响应、无返回数据。  **容易出现的问题：**  1、编码问题，客户端传输的数据编码不符合规范，或者服务器端解码时不符合规范解码有问题。导致中文字符显示为乱码。  IOS的原理与上面类似。 | | |
| **一级**  **用例** | 1、按照接口规范传正规参数。  2、检查数据格式校验（不使用接口规范中定义的json格式或者xml格式进行测试）  3、检查参数是否合法（缺少参数、非法参数、参数格式不规范、多出没定义的参数、参数长度超长、大小超出约束范围）  4、检查接口处理是否正确（获取、修改、删除数据或其他既定功能是否正确完成）  5、接口停用后，客户端能继续其他操作，并给出友好提示，不会导致崩溃。  6、接口传输数据量大时会否有问题（上传、下载）  7、检查接口传输过来的数据是否加载显示完整、是否有乱码。 | | |
| **二级**  **用例** | 1、测试接口性能和响应能力  2、接口访问过程中，网络闪断 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 因为接口大多是基于http get或post的轻量级接口，测试接口使用Jmeter工具里的http请求采样器或者xml-rpc采样器即可（也可以使用httpclient工具包写java程序调用） | | |

### 3、稳定性及异常测试

#### 3.1冲突测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 冲突测试 | **整理人员** | 唐晓文 |
| **简介** | 所有冲突测试本质上都是手机应用的状态流转测试。Android手机提供的打断事件与非智能机有着一些机制上的不同。所以我们要从他的应用程序打断机理出发，定制我们的系统测试用例的打断事件。 | | |
| **详细技术说明** | **【Android应用中的冲突事件详解】**  每个冲突事件实际上都是一种状态的流转，都会触发相应的事件函数，所以我们要从他的应用程序打断原理出发，定制我们的系统测试用例的打断事件。我们只需要分析、检查程序是否在相应的事件中做了合理的处理。测试时需要注意区分等价类（相同意义的操作可以不用重复）。  **系统基本事件（系统中断）：**  1. Home键回home app，当前测试的应用就会放到后台处于stop状态，当然如果开发人员改了接口我们也会看到不同的东西。  2. 长按Home键，切换应用程序。  3. Back键。  4. 长按Power键，调出关机窗口。  5. 改变手机方向（重力感应切屏）  6. 内容冲突检测（content provider）  7. 锁屏、解锁  **Interaction interruptions（互动的中断）：**  1. Class 0 SMS接收短信（Class 0:手机模拟器AVD中模拟的一种SMS到来的方式。 自动显示在手机终端上，无论手机或者SIM的容量是否已满，但是不能自动保存在手机或者SIM卡上。一般只是接收，其接收的时候只自动显示内容，不会被保存在手机或SIM卡中。）  2. 弹出的Alarm闹铃  3. 来电显示  4. Bluetooth的匹配请求  5. Bluetooth的文件发送接收请求窗口  6. 低电量信息提示窗口  这些都会打断当前Activity的生命周期，所以如果这些都能够包含在我们的测试用例中（尽可能多的Activity界面都应该检查这些，当然了，不可能每个UI都检查，但是关键业务点需要重点测试这些打断事件）  在正常的业务流程，我们不会检查这些，而这些我们的开发人员自己也会做到，所以在测试过程中很难以发现更多的问题了。如果加上这些检查，我们的测试覆盖应该是非常全面的而且实践证明也是非常有效的。  **【IOS应用中的冲突事件详解】**  Iphone外置的按键有如下：Home键、power键、音量控制键、静音键。测试时需要注意一些快捷键对应用产生的影响。IOS的中断事件和Android列出的类似，在此不赘述。  IOS和Android的打断事件原理差不多，但是早期的IOS不支持多任务处理，所以在运行中断后不会进入后台运行或是暂停的状态，一般会停止运行。这点需要在系统兼容性测试时注意。 | | |
| **一级用例** | 1、测试关键业务被电源按键、上下键、Home键、返回键、操作键、照相键干扰或打断。  2、测试关键业务被插拔USB连电脑干扰。  3、测试关键业务被插拔电源线干扰。  4、测试关键业务被插拔耳机线干扰（特别是和声音输入输出相关的功能要特别测试）。  5、测试关键业务被电量低提示干扰。  6、测试关键业务被闹铃、来电、来短彩信、来通知干扰。  7、测试关键业务被Bluetooth的匹配请求、发送接收文件干扰。  8、对于使用了content provider进行内容共享的数据，检查在多应用共同使用的情况下是否会发生冲突。  9、APP切换到后台，再回到app，检查是否停留在上一次操作界面。  10、APP切换到后台，再回到app，检查功能及应用状态是否正常，IOS4 和IOS5 的  版本的处理机制有的不一样。  11. app切换到后台，再回到前台时，注意程序是否崩溃，功能状态是否正常，尤其  是对于从后台切换回前台数据有自动更新的时候。  12. 手机锁屏解屏后进入app注意是否会崩溃，功能状态是否正常，尤其是对于从  后台切换回前台数据有自动更新的时候。  13. 当App使用过程中有电话进来中断后再切换到app，功能状态是否正常  14. 当杀掉app进程后，再开启app，app能否正常启动。  15. 出现必须处理的提示框后，切换到后台，再切换回来，检查提示框是否还存在，  有时候会出现应用自动跳过提示框的缺陷。  16. 对于有数据交换的页面，每个页面都必需要进行前后台切换、锁屏的测试，这  种页面最容易出现崩溃。 | | |
| **二级用例** | 暂无 | | |
| **三级用例** | 暂无 | | |
| **补充说明** | 暂无 | | |

#### 3.2异常测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 异常测试 | **整理人员** | 唐晓文 |
| **简介** | 异常测试这里指针对在手机应用在使用过程中可能会遇到的异常情况做测试。 | | |
| **详细技术说明** | **Android、IOS应用在使用过程中可能会出现的系统异常有：**  1、关机断电：当手机电量不足时，会自动关机，此时需要测试评估对于应用正在操作的重要业务是否会造成影响。如果有影响则采取相应处理，针对处理方案进行测试。  2、断网：当手机应用在访问网络过程中断网，会暂停当前操作，给出相应提示，等网络回复后再允许用户启动继续访问。  3、断GPS：当手机应用正在使用定位功能定位时，GPS断了则会使用网络继续定位，如果网络也断了那就获取上一次定位成功的位置作为此次定位结果，如果没有上一次定位地址，则提示用户定位失败。  4、服务器宕机：当应用正在与服务器交互或者准备与服务器交互时，服务器宕机了，应给出合理提示（服务器正忙请稍后）。宕机后能读缓存展现的尽量读缓存或默认值展现，把宕机带来的体验差感觉降到最低。  5、远程数据库断连：当应用在与远程数据库连接交互时，数据库断连访问超时，应给出适当提示。且应该把部分执行成功的事务回滚。  6、系统内存不足：当系统内存不足时，会按照一定规律依次杀掉应用进程以释放内存。当我们的应用被切换到后台后就有可能会被杀掉，如果此时还有用户数据备有保存，则需要在销毁前的事件中保存用户数据，以避免用户数据丢失。（特别是文档编写页面）  7、网络不稳定，网络时断时续。 | | |
| **一级**  **用例** | 1、测试断网对需要访问网络的功能的影响。  2、测试远程数据库断连对数据库访问业务的影响（主要是删除、修改、新增动作）。  3、测试服务器宕机对整个应用的影响。宕机后打开应用依次访问各个功能。  4、在应用的重要功能页面，强制停止应用进程，看进程在销毁前是否保存了重要的业务数据、是否做了必要操作等。 | | |
| **二级**  **用例** | 5、测试关机断电对重要业务功能的影响。  6、测试断GPS对定位的影响（可以在定位过程中走进电梯测试） | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 暂无 | | |

#### 3.3稳定性测试

##### 3.3.1 Android应用稳定性测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | Android应用稳定性测试 | **整理人员** | 唐晓文 |
| **简介** | 测试Android手机应用能否长时间运行稳定不报错。 | | |
| **详细技术说明** | **Android应用有可能出现的稳定性问题：**   1. 功能一次运行不会报错，但是反复运行足够多次后导致应用崩溃。 2. 应用不能承受非法操作，只能规规矩矩的点击设计好的功能键，随意操作点击就会崩溃。 3. 手机在极端情况下是否能正常运行：   1）频繁收到内存警告。  2）电池电量低。  3）网络不稳定的情况下。  **测试方法概述：**  测试第一个问题可以使用iTestin录制脚本反复执行，监控。  测试第二个问题可以使用Monkey工具发送随机事件流来检测应用的稳定性。  这两种方法都与内存泄露测试方法相似，但是不同点在于稳定性测试需要长时间稳定运行，一般事件的设定为12\*n个小时。  **使用到的工具：**  Itestin/monkey  工具不在此详细讲解，请参见本文中的工具使用章节 | | |
| **详细操作步骤** | **方法一：使用iTestin测试，andtools监控**（andtools在能耗测试中有讲解使用方法）   1. 使用Itestin录制全功能遍历脚本，从首页开始，最后回到首页以便能够循环回放执行（录制脚本时，间隔时间尽量放长一些） 2. 在设备上安装andtools设定监控内存使用和监控的时长，例如：12小时 3. 在andtools中打开被测应用。手机连接电脑iTestin，设置好循环数开始运行脚本（循环数要大约能够持续运行12小时） 4. 执行完后查看iTestin日志。 5. 中途没有中断、内存占用正常，不超过75%即可认为通过测试。   **方法二：使用monkey执行随机测试（**Monkey的详细操作见测试工具章节**）**   1. 在手机上打开被测应用，页面停留在首页。 2. 手机连接电脑，使用ADB进入手机shell。 3. 在手机执行monkey命令，（因为是测试稳定性，所以运行的时间需要设置长一些，点击时间间隔也需要设置长一些。） 4. 执行完后查看日志。若有报错等信息会自动停止执行。 5. 日志有报错导致测试停止测不通过。全部顺利执行完毕测视为通过。   **其他测试项：**  1、手机在频繁收到内存警告的情况下使用应用。  2、手机在电池电量低的情况下使用应用。  3、在网络不稳定的情况下使用应用。 | | |
| **补充**  **信息** | 里面用到的测试工具可在工具库中找到。 | | |

##### 3.3.2 IOS应用稳定性测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | IOS应用稳定性测试 | **整理人员** | 唐晓文 |
| **简介** | 测试IOS手机应用能否长时间运行稳定不报错。 | | |
| **详细技术说明** | **IOS应用有可能出现的稳定性问题：**   1. 功能一次运行不会报错，但是反复运行足够多次后导致应用崩溃。 2. 应用不能承受非法操作，只能规规矩矩的点击设计好的功能键，随意操作点击就会崩溃。 3. 手机在极端情况下是否能正常运行：   1）频繁收到内存警告。  2）电池电量低。  3）网络不稳定的情况下。  **测试方法概述：**  测试第一个问题可以使用iTestin录制脚本反复执行，监控。  测试第二个问题可以使用IOSmonkey2.js脚本发送随机事件流来检测应用的稳定性。  这两种方法都与内存泄露测试方法相似，但是不同点在于稳定性测试需要长时间稳定运行，一般事件的设定为12\*n个小时。  **使用到的工具：**  Itestin/IOSmonkey2.js  **IOSmonkey2.js脚本下载地址如下：**  https://github.com/douban/ynm3k/blob/master/robot4ios/util/iOSMonkey2.js | | |
| **详细操作步骤** | **方法一：使用iTestin测试，用instruments监控**  注意：IOS设备需要越狱，操作方法和Android应用测试相同，在此不赘述。  使用Instruments监控方法请参见5.2.2章IOS应用能耗测试。  **方法二：使用IOSMonkey2.js脚本测试，用instruments监控**  1、手机连上电脑，在Instruments中选择运行Automation，在Target中选择被测应用。  2、在左下角的Scripts中的Add下拉框中选择“iOSMonkey2.js“脚本。  3、点击左上角的“Run“按钮开始运行，同时监控手机应用的运行情况，是否报错闪退等。  4、脚本运行完成后，可以在日志模块中查看运行情况。  5、猴子测试脚本在每个操作后面都做了截图保存，并且输出了被操作控件所有的父控件，这样可以根据截图和控件之间的关系发现、定位问题。  **Monkey测试还可以在以下特殊情况下进行，更容易发现问题：**  1、手机在频繁收到内存警告的情况下使用应用。  2、手机在电池电量低的情况下使用应用。  3、在网络不稳定的情况下使用应用。 | | |
| **补充**  **信息** | 暂无 | | |

### 4、兼容性测试

#### 4.1系统版本适配

##### 4.1.1 Android系统版本适配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 系统版本适配测试 | **整理人员** | 唐晓文 |
| **简介** | 测试应用是否能在不同版本的系统上正常运行。 | | |
| **详细技术说明** | **【目前Android的系统版本】**   1. Android1.1（2009年2月发布）：包含闹钟，API 示例，浏览器，计算器，摄像头，联系人，开发工具包，拨号应用，电子邮件，地图 （包含街景），信息服务，音乐，图片，设置。 2. Android1.5（2009年5月发布）：新增录像，蓝牙A2DP，自动蓝牙连接，上传视频到 YouTube 以及 Picasa，复制/粘贴功能。 3. Android1.6（2009年9月发布）：新增Android 应用市场集成，照相、摄像以及浏览， 多选/删除功能，手势搜索，语音搜索应用集成，极大提升了语音阅读功能，对非标准分辨率有了更好的支持。 4. Android2.0/2.1（2009年10月发布）：提升硬件速度，更多屏幕以及分辨率选择，大幅度的用户界面改良，支持 Exchange活动墙纸，大幅改进虚拟键盘，蓝牙2.1，Google地图 3.1.2。 5. Android2.2：完整的 Flash 10.1 支持，最高5倍速度提升，最多支持8个设备连接的移动热点功能，有着专用链接的改进版主屏幕，大量Exchange支持改进，支持 Exchange 2010，摄像头/视频改进，多键盘语言支持，Android 云信息将应用安装在记忆卡上，蓝牙语音拨号，支持720P视频录制。 6. Android2.3（2010年12月发布）：用户界面更美观，提升游戏体验，提升多媒体能力，增加官方进程管理，改善电源管理，NFC进场通信，全局下载管理，改进的虚拟键盘，原生支持前置摄像头，SIP网络电话。 7. Android3.0（2011年2月发布）：优化针对平板，全新设计的UI增强网页浏览功能，n-app purchases功能。 8. Android3.1（2011年5月发布）：改进3.0BUG，经过优化的Gmail电子邮箱，全面支持GoogleMaps，将Android手机系统跟平板系统再次合并从而方便开发者，任务管理器可滚动，支持USB输入设备（键盘、鼠标等），支持 Google TV.可以支持XBOX 360无线手柄，widget支持的变化，能更加容易的定制屏幕widget插件。 9. Android3.2（2011年7月发布）：支持7英寸设备，引入了应用显示缩放功能。 10. Android4.0（2011年10月发布）：全新的ui，全新的Chrome Lite浏览器，有离线阅读，16标签页，隐身浏览模式等。截图功能。更强大的图片编辑功能。自带照片应用堪比Instagram，可以加滤镜、加相框，进行360度全景拍摄，照片还能根据地点来排序。Gmail加入手势、离线搜索功能，UI更强大。新功能People：以联系人照片为核心，界面偏重滑动而非点击，集成了Twitter、Linkedin、Google+等通讯工具。有望支持用户自定义添加第三方服务。新增流量管理工具，可具体查看每个应用产生的流量。正在运行的程序可以像电脑一样的互相切换。人脸识别功能。系统优化、速度更快。支持虚拟按键，手机可以不再拥有任何按键。更直观的程序文件夹。平板电脑和智能手机通用。支持更大的分辨率。专为双核处理器编写的优化驱动。增强的复制粘贴功能。语音功能。全新通知栏。更加丰富的数据传输功能。更多的感应器支持。语音识别的键盘。全新的3D驱动，游戏支持能力提升。全新的谷歌电子市场。增强的桌面插件自定义。   **其他厂家定制的系统版本：**   |  |  |  | | --- | --- | --- | | **定制安卓系统** | | | | **软件名称** | **测试版本** | **官网地址** | | MIUI | V4 | [点击跳转](http://www.miui.com/getrom.php) | | Fly Me（魅族） | 2 | [点击跳转](http://flyme.meizu.com/flysys/system.jsp) | | 点心OS（Tapas OS） | 2 | [点击跳转](http://www.dianxinos.com/richpage?accordion_name=ossolution&richid=51) | | 百度云ROM | V2 | [点击跳转](http://bbs.rom.baidu.com/download.php) | | 腾讯tita | 1.3.0 | [点击跳转](http://tita.qq.com/bbs/download.php) | | CyanogenMod（CM） | 10.1 | [点击跳转](http://download.cyanogenmod.com/) | | 力卓（Lidroid） | 基于Android 4.1.2 | [点击跳转](http://apps.lidroid.com/rom/) | | HTC Sense UI | HTC Sense 4.1 | -- | | 三星touchwiz | 5 | -- | | 摩托Blur UI | Blur 1.5 | [点击跳转](http://www.motorola.com/Support/CN-ZH/Consumer-Support/Software/) | | 华为Emotion UI | 基于安卓4.1 | [点击跳转](http://www.huafans.cn/emotiondownload.php) | | 乐OS（联想） | 2 | [点击跳转](http://bbs.lephone.cc/leos.html) | | 乐蛙OS | 乐蛙OS 4.0 | [点击跳转](http://bbs.lewaos.com/forum.php) |   各定制系统简介：   1. MIUI：是小米科技旗下基于Android系统深度优化、定制、开发的第三方手机操作系统。 2. Fly Me：主要用于魅族MX系列手机，可使用云服务端为用户提供便利的数据存储和同步。 3. 点心OS：是一款基于Android操作平台、针对中国用户使用习惯打造的互联网智能手机操作系统，点心操作系统在底层技术架构优化的基础上，为用户提供流畅的操作体验；同时根据本土化需求，为用户提供贴心的功能设计及丰富多彩的移动互联网应用。 4. 百度云ROM：由百度公司推出，是基于Android 4.0进行开发，初次刷机需要使用到百度云一键刷机工具，而后可以通过OTA方式进行无线升级。 5. 腾讯tita：是腾讯推出的一款基于Android操作系统的定制版ROM，内部已经将该系统定名为tita，腾讯tita基于Android 4.0体系定制，拥有全新的界面计划，体验、机能全面进级，最注重就是简捷易操纵方面。 6. CyanogenMod：是一个基于开源Android系统（Cyanogen团队开发），供某些手机使用的二级市场固件。它提供一些在官方Android系统或手机厂商没有提供的功能。最新版本10.2，基于Android标准版4.3 7. Lidroid（力卓）：是由力卓社区ROMER团队基于安卓原生系统开发的第三方安卓手机操作系统，ROM在稳定流畅的基础上，针对用户使用习惯，进行系统优化，开发、定制多种独有人性化功能。支持的品牌机型：三星、谷歌、HTC、联想。 8. HTC Sense：是宏达开发的一种针对Android的智能手机触控UI（只做了系统UI修改）。自HTC Hero开始HTC所有装置都将与HTC Sense整合。2013发布的最新版本HTC sense 5.0，基于安卓4.1.2。 9. TouchWiz：是三星公司对于安卓手机进行二次优化的界面。这项技术可以实现动态图标的拖放和界面平滑切换。TouchWiz的操作模式则和之前几款手机完全不同了，无论是菜单操作还是输入法界面都很适合手指进行控制。目前最新版本为TouchWiz 5.0，基于Android4.0操作系统。 10. Moto Blur：是Motorola第一款Android手机所搭载的定制化UI。摩托罗拉Blur是基于谷歌Android平台开发的应用界面，除了基本的Android特性之外，Blur最突出的特色是注重网络社交功能。 11. 华为Emotion：是华为基于Android进行开发的，以“简单易用、功能强大、情感喜爱”为核心设计理念，以易发现、易享受、易分享为目标的智能终端人机交互系统。 12. 乐OS（联想）：是联想基于Android平台的操作系统。目前版本（乐OS1.0、乐OS 1.5、乐OS2.0、乐OS2.5、乐MAGIC 3.0、乐MAGIC 3.5、乐MAGIC 4.0） 13. 乐蛙OS：主要是华为、中兴的智能机在使用。   **各品牌手机市场占用率（2011年发布）：** HTC 31%、Moto 20%、三星14%、华为9%、索爱7%、魅族6%、联想4%、中兴3%、google2% 其他4%  **市场占用率排名前十的手机型号（2011发布）：**  HTC Desire、摩托罗拉MileStone、HTC Hero、HTC Legend、华为C8500、联想乐Phone  索尼爱立信X10i、摩托罗拉XT800、三星I9000、HTC Wildfire  **各个标准版本的系统市场占用率（2012年发布）如下图：**    所以我们选择的主要测试标准版本是：Android2.3/4.0  **对于不同操作系统版本测试时需要注意：**   1. Android2.2版本以后才支持把应用安装在SD中。 2. Android2.2版本之前不支持Flash。 3. 根据不同版本特点评估版本差异性对应用的影响。（是否有功能不支持） 4. 不同系统对于同样功能的处理有所不同。（例如：各个系统的相册应用实现不一致，如果我们的应用中需要调用相册这个系统应用选择照片，可能有些应用就无法实现相册图片选择）   **目前主要Android测试版本：**  1、Android2.3/4.0标准版  2、其他厂家的系统（三星、htc）  小技巧：有些智能机安装的是双系统（例如：小米），一款机器可以测试两款系统兼容性。 | | |
| **一级**  **用例** | **【Android系统的兼容性测试】**   1. 目前主要系统版本的兼容性适配测试（2.3/4.0） 2. 目前主要品牌手机的兼容性适配测试（三星、HTC、华为） 3. 对于调用了系统应用的功能需要专门做系统兼容性测试。 | | |
| **二级**  **用例** | 1、使用Testin云测试平台测试各个厂家不同系统版本的安装、启动、卸载适配。 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | Testin云测试平台的使用方式请参见测试工具介绍模块 | | |

##### 4.1.2 IOS系统版本适配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | IOS系统版本适配测试 | **整理人员** | 唐晓文 |
| **简介** | 测试手机应用在不同IOS系统版本上运行的情况。 | | |
| **详细技术说明** | **目前IOS各类机型及特点：**    注意：苹果公司有这样一条规定——软件系统无法从高版本降低至低版本。例如一台已经安装了IOS6.1的系统设备就无法再安装IOS6.0的系统（除非越狱）。所以我们可以根据此规则缩小测试范围。  **补充说明：**一代IPhone只支持GPRS上网，IPhone3G/3GS可以支持3G上网了，到了IPhone4就是多任务处理了，之前版本都是单任务的处理方式。到了IPhone5就支持4G上网了。  **IOS主流操作系统版本（2014年5月数据）：**   |  |  | | --- | --- | | **系统名称** | **版本** | | IOS5（市场占有率3%） | 5.0,5.1 | | IOS6（市场占有率11.8%） | 6.0,6.1 | | IOS7（市场占有率84.9%） | 7.0 |   基本上只要测试IOS6和IOS7即可覆盖绝大多数IOS设备。  查看IOS各个版本的当前市场占有率的链接如下：  <http://david-smith.org/iosversionstats/>  **IOS设备兼容性：**  IOS设备主流的有3类：IPhone、IPad、ITouch（都是IOS操作系统）  三者之间的主要区别就是硬件配置、屏幕大小、是否能打电话上的区别。  ipad是大号的itouch、是不带通话功能的大号iphone  iphone是带通话功能的itouch、是带通话功能的小号的ipad  itouch是小号的ipad 不带通话功能的iphone  目前在Ipad和Itouch上加软件也能实现wifi和3G通话功能。  **测试版本选择：**  1、按照目前市场占有率来看，我们开发的应用只需要兼容IOS7和IOS6就可以满足97%的用户。查看IOS各个版本的当前市场占有率的链接如下：  <http://david-smith.org/iosversionstats/>  2、设备兼容性选择IPhone和Ipad进行测试。  **主要测试重点：**  系统兼容性的主要测试重点在于测试程序是否正确使用了该软件系统版本的API，再一个就是设备屏幕分辨率。  **常见问题举例：**  1、IOS6之前，系统不支持自动布局，如果IOS5系统上运行程序就会崩溃。  2、应用程序需要解析JSON格式的数据，在没用第三方库的情况下，IOS4.3系统会出现异常。 | | |
| **一级**  **用例** | **【IOS系统的兼容性测试】**   1. 目前主要系统版本的兼容性适配测试（IOS6/IOS7） 2. 目前主要手机的兼容性适配测试（IPhone4S、IPhone5S） 3. 对于调用了系统应用的功能需要专门做系统兼容性测试。 4. 不同的IOS设备兼容性适配测试（IPhone、Ipad） | | |
| **二级**  **用例** | 1、使用Testin云测试平台测试各系统版本的安装、启动、卸载适配。 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | Testin云测试平台的使用方式请参见测试工具介绍模块 | | |

#### 4.2界面适配

##### 4.2.1 Android界面适配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 界面适配测试 | **整理人员** | 唐晓文 |
| **简介** | 测试在不同的手机界面下应用是否能正常运行展现。 | | |
| **详细技术说明** | **术语解释：**   1. 手机屏幕尺寸：指的是手机屏幕实际的物理尺寸。现有的Android手机主要屏幕尺寸有：2.8、3.1、3.2、3.7、4、4.2、4.3、5.0(单位/英寸) 2. 分辨率（dpi）： dpi（dots per inch）分辨率、解析度，每英寸像素数，多用于屏幕显示领域。 3. 像素：px 4. dip（device independent pixels）设备独立像素. 不同设备有不同的显示效果,这个和设备硬件有关，多为了支持WVGA、HVGA和QVGA 使用，不依赖像素。dip的公式：px /dip=dpi/160 所以dip类似于英寸、长度（dp=dip，sp类似于dip） dip=160\*inch 5. 相对分辨率=长px\*宽px   **一些相关概念的说明：**   1. 平时我们说手机的分辨率是 320\*480的，其实的这里的分辨率是相对分辨率。意思是：水平方向上的像素数是320，垂直方向上像素数是480，分辨率是160（默认是160，意思是每英寸像素数160）那么水平方向：320 /160=2英寸垂直方向:480/160=3英寸。于是乎屏幕对角线 是根号下4\*9=3.6（这就是常说的3.6英寸屏幕） 2. 一个手机的屏幕参数有三个：长宽像素之积（相对分辨率）、真实分辨率、对角线长度（真实分辨率默认是160所以不常说，如果不是160你可以通过另外两个参数求出真实分辨率） 3. 模拟器的分辨率都是160，所以像素越大，屏越大。   **Android本地应用手机屏幕适配方法详解：**  Android本地应用的屏幕适配一般有三种方式解决  1、按像素比 y/开发时用的屏幕像素=x/用户设备像素  2、按长度 用dpi（假设屏幕尺寸基本不变）  3、按密度 放在l、m、h文件夹（假设屏幕尺寸基本不变，dpi越大，px越大）  常用的是第三种方式，在此仅就第三种方法详细讲解一下原理和测试中可能遇到的问题：  android从2.1版本开始从一个用于存放图片资源的drawble文件夹分为了drawable-mdpi、drawable-ldpi、drawable-hdpi三个，这三个主要是为了支持多分辨率。  (1)drawable-hdpi里面存放高分辨率的图片,如WVGA (480x800),FWVGA (480x854)  (2)drawable-mdpi里面存放中等分辨率的图片,如HVGA (320x480)  (3)drawable-ldpi里面存放低分辨率的图片,如QVGA (240x320)  系统会根据机器的分辨率来分别到这几个文件夹里面去找对应的图片。  注意：这三个文件夹都是按真实分辨率dpi来对应找文件的。  图片大小的确定：low:medium:high:extra high的比例为3:4:6:8。  如果手机是高分辨率，但高分辨率文件夹里没有东西，中等分辨率文件夹里有东西，程序就会去中等里找图片并且把它按比例放大。同理如果低文件夹里没有就会去找高文件夹，把找到的图片缩小展示。  例如：在一个低分辨率的手机上，Android就会选择ldpi文件夹下的图片，但是如果没有在ldpi的文件夹下找见相关的资源文件，Android系统会首先从hdpi文件夹中选择文件，然后对图片资源进行缩放处理，显示在屏幕上；如果hdpi文件夹下也没有的话，会在默认的drawable文件夹中寻找。  以上是系统提供的一种通用适配方案，如果想要完全适配具体的分辨率，最好的方式是单独针对具体的分辨率建立相应文件夹来做适配。文件夹命名示例如下：  屏幕相对分辨率：1024x600  分辨率：1（160）  对应文件夹：values-mdpi-1024x600  屏幕相对分辨率：1024x600  分辨率：1.5（240）  对应文件夹：values-hdpi-683x400 ---由1024/1.5 ，600/1.5得到，需要四舍五入。  以此类推，一般情况下需要创建出values 、values-mdpi 、values-hdpi文件夹，以备在一些没有规定的尺寸屏幕上找不到资源的情况。  **图片放错文件夹可能会出现的问题：**  1、本身图片是同一张图片，由于没有在hdpi文件夹中找到对应图片，系统在默认文件夹下找见了图片资源，但是这时系统会认为改图适用于中等分辨率的屏幕上，如果直接放到高分辨率的手机上不能适配，所以系统会自动的将图片放大，所以虽然是同一张图片，但是现实出来就一个正常效果，一个放大效果。如下图所示：  http://img.my.csdn.net/uploads/201301/16/1358303139_9385.jpg图片在hdpi文件夹中  http://img.my.csdn.net/uploads/201301/16/1358303177_6285.jpg图片不在hdpi文件夹中但在默认文件夹下  2、如果同一张图片，放置在ldpi的文件夹下，在低分辨率的手机上显示正常，但是如果该图片被放在hdpi文件夹中，系统认为该图是为高分辨率显示的，在低分辨率手机上就要将图片缩小处理，所以显示出来的效果就是图变小了。  **总之：测试时需注意检查在相应分辨率下，是否显示的是正确分辨率文件下的图片（避免图片放错文件夹）。**  **屏幕布局实现自适应原理：**  以上说的都是图片资源文件夹，布局文件夹同理。只需要在res目录下创建不同的layout文件夹，比如layout-640x360,layout-800x480,所有的layout文件在编译之后都会写入R.java里，而系统会根据屏幕的大小自己选择合适的layout进行使用。  可以为不同分辨率提供不同的布局文件（文件夹中的布局文件名称要相同）例如：  layout-320x240 注： 320 (大分辨率) 要写在前面，(240)小分辨率写在后面  layout-480x320 注： 480 (大分辨率) 要写在前面，(320)小分辨率写在后面  **横竖屏切换如何实现自适应：**  在对应的分辨率下建立图片资源文件夹，分辨满足横屏、竖屏时获取图片资源。  drawable-land-hdpi,当屏幕为横屏，且为高密度时，加载此文件夹中的资源  drawable-port-hdpi,当屏幕为竖屏，且为高密度时，加载此文件夹中的资源  在res目录下建立layout-port-800x600和layout-land两个目录，里面分别放置竖屏和横屏两种布局文件，这样在手机屏幕方向变化的时候系统会自动调用相应的布局文件，避免一种布局文件无法满足两种屏幕显示的问题。  **Android web应用如何实现介面适配：**  Android为了简化Web应用程序的开发，提供了下面的支持：  1.支持viewport性能，运行你根据屏幕的大小来调整应用程序的大小。  2.CSS和javaScript的特征，允许你根据屏幕的分辨率来提供不同的样式和图片。  因此，在开发Android的Web应用程序时，你可以不必考虑屏幕是否支持，因为Android已经帮你匹配了Web页面。  **Android手机主流分辨率：**  QVGA = 320 \* 240;  HVGA = 480 \* 320;（目前有测试机）  WVGA = 800 \* 480;（目前有测试机）  FWVGA = 854 \* 480;  DVGA = 960 \* 640; | | |
| **一级**  **用例** | 1. 在不同分辨率下测试所有界面的图片展现，图片是否有模糊变形。（要把所有适配的图片、布局文件都测到） 2. 横竖屏切换的界面兼容性测试（所有屏幕尺寸下的图片都能正确显示） 3. 测试Web应用界面在不同分辨率下的展示。（与网页适配测试类似） | | |
| **二级**  **用例** | 暂无 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 可以使用AVD模拟器来做屏幕适配测试（缺点：运行慢，可以删除一些应用以加快速度）。  **Android手机以及平板设备各类分辨率如下（**加粗的是主流分辨率**）：**  **QVGA = 320 \* 240;**  WQVGA = 320 \* 480;  WQVGA2 = 400 \* 240;  WQVGA3 = 432 \* 240;  **HVGA = 480 \* 320;（有测试机）**  VGA = 640 \* 480;  **WVGA = 800 \* 480;（有测试机）**  WVGA2 = 768 \* 480;  **FWVGA = 854 \* 480;**  **DVGA = 960 \* 640;**  PAL = 576 \* 520;  NTSC = 486 \* 440;  SVGA = 800 \* 600;  WSVGA = 1024 \* 576;  XGA = 1024 \* 768;  XGAPLUS = 1152 \* 864;  HD720 = 1280 \* 720;  WXGA = 1280 \* 768;  WXGA2 = 1280 \* 800;  WXGA3 = 1280 \* 854;  SXGA = 1280 \* 1024;  WXGA4 = 1366 \* 768;  SXGAMINUS = 1280 \* 960;  SXGAPLUS = 1400 \* 1050;  WXGAPLUS = 1440 \* 900;  HD900 = 1600 \* 900;  WSXGA = 1600 \* 1024;  WSXGAPLUS = 1680 \* 1050;  UXGA = 1600 \* 1200;  HD1080 = 1920 \* 1080;  QWXGA = 2048 \* 1152;  WUXGA = 1920 \* 1200;  TXGA = 1920 \* 1400;  QXGA = 2048 \* 1536;  WQHD = 2560 \* 1440;  WQXGA = 2560 \* 1600;  QSXGA = 2560 \* 2048;  QSXGAPLUS = 2800 \* 2100;  WQSXGA = 3200 \* 2048;  QUXGA = 3200 \* 2400;  QFHD = 3840 \* 2160;  WQUXGA = 3840 \* 2400;  HD4K = 4096 \* 2304;  HXGA = 4096 \* 3072;  WHXGA = 5120 \* 3200;  HSXGA = 5120 \* 4096;  WHSXGA = 6400 \* 4096;  HUXGA = 6400 \* 4800;  SHV = 7680 \* 4320;  WHUXGA = 7680 \* 4800;  **主流分辨率下的主流手机（2011年发布）：**  ==== 320x480 ====  1. 三星 S5830 Galaxy Ace  2. 索尼爱立信 X8 (E15i)  3. 华为 U8500  4. Motorola XT502  5. LG P503  6. 酷派 D530  7. LG GT540  8. 夏普 SH8118U  9. 天语 W606 (估计运行困难)  10. 中兴 R750  11. 华为 U8500-HiQQ (估计运行困难)  12. 酷派 W711(估计运行困难)  13. 首派 A60  14. 三星I7500  15. HTC G6 Legend  16. Motorola ME600  17. Motorola CLIQ  18. 三星 I5700  19. HTC Aria (G9)  20. LG GW620  21. 华为 C8600  22. Motorola ME501  23. LG P500  ==== 480x800 ====  1. HTC Desire (G7)  2. HTC A9191  3. 三星 I9003  4. 联想W101 乐phone (1.6系统非原生)  5. 夏普 SH8128  6. 三星 I9088  7. 中兴 V880  8. 三星 I909  9. Motorola A1680  10. 联想W100 乐phone(1.6)  11. 联想C101  12. 酷派 N930  13. 三星 I9088  14. HTC 双擎T9199  15. HTC A9191  16. 中兴 V9E  17. 三星I909 (SPhone)  18. 酷派 N930  19. 华为 U8800  20. HTC Desire HD (G10)  21. 三星 I9010  22. 夏普 SH7218U(新机)  23. Google Nexus One (G5)  24. Google Nexus S  25. 三星I9000  26. 三星T959  27. 三星W899  28. HTC Incredible S (S710e)  29. HTC Desire Z  30. HTC EVO 4G  31. 戴尔Mini5 (Streak)  32. 三星I9020  33. 戴尔Venue  ==== 640x960 ====  1. 魅族 M9  ==== 480x854 ====  1. Motorola XT702  2. Motorola XT711  3. Motorola ME722 (里程碑2)  4. Motorola XT800  5. Motorola ME811 (Droid X)  6. Motorola ME525  7. Motorola XT800+  8. 索尼爱立信 X10i  9. Motorola XT806Ix  ==== 320x240 ====  1. SAMSUNG S5570  2. HTC A3366 Wildfire  3. Motorola ME511  4. Motorola XT301  5. 华为 C8500  6. HTC A3380  7. 三星 I5508  8. HTC A3360  9. 中兴 X850  10. Motoroal XT300  11. 索尼爱立信 U201  ==== 600x1024 ====  1. 三星 Galaxy P1000 | | |

##### 4.2.2 IOS界面适配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | IOS界面适配测试 | **整理人员** | 唐晓文 |
| **简介** | 测试手机应用在不同屏幕的IOS手机上运行的情况。 | | |
| **详细技术说明** | **目前IOS各类机型及分辨率如下：**    **测试版本选择：**  1、按照目前市场占有率来看，目前手机上只需要兼容4S和5S就可以满足。  查看IOS各个版本的当前市场占有率的链接如下：  <http://david-smith.org/iosversionstats/>  2、设备兼容性选择IPhone和Ipad进行测试。 | | |
| **一级**  **用例** | **【IOS系统的兼容性测试】**  1、目前主要手机的兼容性适配测试（IPhone4S、IPhone5S）  2、不同的IOS设备兼容性适配测试（IPhone、Ipad） | | |
| **二级**  **用例** | 1、使用Testin云测试平台测试各版本的UI适配。 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | Testin云测试平台的使用方式请参见测试工具介绍模块 | | |

#### 4.3不同网络适配

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 网络适配测试 | **整理人员** | 唐晓文 |
| **简介** | 测试手机应用在不同网络环境下的运行情况。 | | |
| **详细技术说明** | **目前智能手机支持的网络按渠道分有两大类：**   1. 移动网络（2G/3G/4G，移动/联通/电信） 2. WIFI网络（移动、联通、电信）  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 中国移动 | | | | 中国联通 | | | | | 2G（GPRS） | | 3G | 无线宽带 | 2G（GPRS） | | 3G沃 | 无线宽带 | | cmwap | cmnet | TD-SCDMA | 1M\4M\8M | nuiwap | uninet | WCDMA | 1M\4M\8M | | 中国电信 | | | | 4G | | | | | 2G（GPRS） | | 3G天意 | 无线宽带 | 中国移动 | | | TDD | | 无 | 无 | CDMA | 1M\4M\8M | 中国联通、电信 | | | FDD |   **2G（GPRS）网络中的两种接入方式（WAP、NET）的区别：**  简单地说WAP是内网，NET是外网。  通过WAP连接后，得到的IP是10.X.X.X，是个A类，也就是内网IP。  而通过NET连接，就会获得一个类似20X.XXX.XXX.XXX这样的标准C类IP，可以浏览整个INTERNET上的任何网站——简单的说，这时候你的手机就跟你的电脑上网能力是一样的了。这两种连接方式，资费不同的原因就在于此，前者连接，中国移动基本是不需要任何成本的，而后者，中国移动需要对网间数据进行费用结算。可以通过一些特殊的方法实现从WAP链接net。  **测试注意事项：**  1、主要测试不同网络下应用的访问速度。（如果应用只在本地运行，不需要网络就不用测）  2、使用WIFI访问，网络带宽不一样，访问速度也不一样。  3、无论网速快还是慢，应用都要能正常运行，不能崩溃。  4、测手机应用不需要关心不同网络的链接原理，这是由Android/IOS系统底层处理的。如果是测手机则需要关心。 | | |
| **一级**  **用例** | 1. 测试覆盖不同网络（上表中列出的12组），目前我们主要选择中国移动3G、2G、4G、wifi进行测试。着重测试是否能连通访问，以及访问速度快慢问题。 2. 无网络时，执行需要网络的操作，给予友好提示，确保程序不出现crash。 3. 内网测试时，要注意选择到外网操作时的异常情况处理。 4. 在网络信号不好时，检查功能状态是否正常，确保不因提交数据失败而造成crash。 5. 在网络信号不好时，检查数据是否会一直处于提交中的状态，有无超时限制。 6. 如遇数据交换失败时要给予提示。 7. 在网络信号不好时，执行操作后，在回调没有完成的情况下，退出本页面或者 8. 执行其他操作的情况，有无异常情况。此问题也会经常出现程序crash。 | | |
| **二级**  **用例** | 暂无 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 有些宽带wifi网络会出现DNS解析错误无法访问某些IP地址的问题（例如：移动宽带） | | |

### 5、性能测试

#### 5.1大数据量测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 大数据量测试 | **整理人员** | 黄万冰 |
| **简介** | 这里指针对在手机应用使用过程中可能会遇到大数据量梳理的功能做的测试。 | | |
| **详细技术说明** | **手机应用可能会遇到的大数据量处理情况：**   1. 从远程服务器数据库获取大量数据。（获取到的数据存在本地缓存） 2. 从本地数据库读取大量数据。 3. 处理大文件（图片处理、视频处理）——处理时会把资源放入内存处理，注意内存溢出。 4. 上传下载大文件。 5. 远程数据库数据量太大，导致查询返回太慢。 6. 本地数据库或文件保存大量数据。（可能出现的问题：开启1000个事务去插入1000条数据sqlite耗时2分钟，开启1个事务插入1000条数据耗时0.2秒。有些问题需要在数据量较大的情况下才会显现）   要了解程序是如何实现的，才能根据实际情况判断需不需要加大数据量来测试。  **加大数据的方式：**   1. 使用工具（DATA Factory）或存储过程，在服务器端数据库中生成大量数据。加大服务器数据后，可使用jmeter测试接口查询效率。 2. 对于Android应用来说直接进入adb shell 操纵sqlite数据库加数据，或者拷贝数据库文件出来使用sqliteadmin工具添加数据。对于IOS应用来说，可以使用ITools工具把sqlite数据库拷贝出来加上测试数据后再上传到手机原文件目录下。 | | |
| **一级**  **用例** | 1. 对于列表要尽可能多的加载大量数据。（看需求最多支持多少条） 2. 上传下载文件或图片选用大文件测试。（看是否会造成崩溃） 3. 对于图片、视频的处理功能需要测试拿大文件测试。（看是否造成崩溃） 4. 加大要查询的数据库的数据量，测试查询效率。 5. 加大接口要处理的数据库的数据量，测试接口返回效率。 6. 批量操作本地数据库的功能，需要加大数据测试。 | | |
| **二级**  **用例** | 暂无 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | 文中提到的工具在测试工具库中可找到。 | | |

#### 5.2能耗测试

##### 5.2.1 Android应用能耗测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | Android能耗测试 | **整理人员** | 唐晓文 |
| **简介** | 测试Android手机应用运行时所需能耗。 | | |
| **详细技术说明** | **Android应用对CPU的消耗：**  应用切换到后台后（即不可见），不占用CPU，后台服务程序除外。  比较耗CPU的功能有：视频音乐播放、图片处理、上传下载文件、游戏。  中国电商企业Android版App应用CPU消耗平均水平为8.91%。  较优的合理值（平均CPU占用率）：8%  **Android应用对内存的消耗：**  实际上是虚拟机的内存占用。当超过虚拟机的内存分配最大值时，内存溢出。  比较消耗内存的功能：视频播放、大图片处理加载、3D渲染（游戏）、上传下载获取大数据。  可接受的合理值（平均内存占用率）：不超过70%  **Android应用对流量的消耗：**  所有访问网络的功能都会消耗流量，对用户来说这个值越小越好。  比较消耗流量的功能：自动同步、后台获取大数据，上传下载大图，在线视频听歌、游戏。  **Android应用对电量的消耗：**   1. RAM只要通上电就一直耗电。 2. CPU是主要的耗电大户。切换到后台的程序不运行就不占CPU。若还在运行，则会消耗CPU，也就是耗电。 3. 使用GPS定位比使用网络定位要耗电，在电量不足时能够默认使用网络定位最好。 4. 网络连接、传感器耗电较大。   比较耗电的功能：大数据量的传输、游戏、上网、处理图片、播放视频、解析大量的文本数据。  **程序上减少耗电的做法：**  1、检查网络连接，如果无网络可用，就不需要进行联网操作。  2、获取屏幕尺寸等信息可以使用缓存技术，不需要进行多次请求。  3、使用效率高的数据格式和解析方法。针对移动设备，最好能使用JSON之类的轻量级数据格式为佳。  4、在进行大数据量下载时，尽量使用GZIP方式下载。  5、后台服务service频繁获取服务器数据也很耗电。最好是定个较长时间获取。 | | |
| **详细操作** | **【单个手机测试应用能耗】**  **测试思路：**   1. 录制自动遍历应用中所有功能的脚本，监控工具开始后跑脚本，跑完后记录结果。 2. 开启监控工具后，手动操作应用里的功能，测试完后记录结果。   **使用工具：**andtools，可测试应用的CPU，内存、流量（无法测试耗电量），只能定在一定时间范围内进行监控。  **实现步骤：**，步骤如下：  1、安装andtools工具。  2、运行andtools，选择需要测试的程序，在选择测试类型（可选择CPU，内存，和流量），一次只能选一项进行测试。如图：    3、点击开始测试后，启动选择的程序，手动进行操作遍历所有功能，设定的监控时间到了以后会在通知里提示监控结束。（所以设置的时间长短，要足够满足操作完所有功能，如果只是针对某项功能监控的话可以设置时间短一些，总之时间长短要合适）  4、查看记录结果（只能得到平均值和最大值）。  **对于测试流量消耗来说测试一款手机和多款手机是一样的，所以批量测试主要是针对CPU\内存的测试。**  **【大批量手机测试】**  **测试思路：**使用工具录制遍历尽可能多的脚本   1. 进入Itestin网站，选择性能测试。 2. 然后根据提示上传APP应用。 3. 选择需要监控的市面上的一些机型。 4. 提交测试。提交测试后，等待测试的结果，可以看到该应用在各个机型上的CPU，内存，流量，电量进行监控。并绘制出图像出来。使用简单，缺点是结果并不是即使反馈结果，需要等到2小时以上。   **注：**  由于Itestin对每款终端的测试时间很短，测试的功能不全面，此结果只能作为参考，不够精准。 | | |
| **补充**  **信息** | 此项测试用到的工具安装包在工具库里有。  详细操作使用说明请参见工具库对应文档。 | | |

##### 5.2.2 IOS应用能耗测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | IOS能耗测试 | **整理人员** | 唐晓文 |
| **简介** | 测试IOS手机应用运行时所需能耗。 | | |
| **详细技术说明** | IOS应用测试能耗可以使用IOS自带的Instruments来进行测试。  下面就详细介绍下Instruments中带的用于监控应用资源消耗的各种小工具。  **1、网络活动监控器（Network Activity Monitor）**  Network Activity Monitor instrument 工具记录电脑网络传输信息。该工具可以运行在单个进程或系统当前所有运行的进程之上。跟踪面板可以被设置来默认显示以下网络相关的数据，但你也可以配置它来显示其他类型的数据。默认情况下，它显示以下信息：  1）每秒发送的字节数量  2）每秒收接收字节的数量  3）每秒发送包的数量  4）每秒接收包的数量  **2、内存监控器（Memory Monitor）**  Memory Monitor instrument 工具记录进程使用的实际内存和虚拟内存的数量。  该工具可以运行在单个进程或系统所有当前运行的进程之上。跟踪面板可以被设置来默认显示以下内存相关的数据，但你也可以配置它来显示其他类型的数据。默认情况下，它显示以下的信息： 1）虚拟内存页面交换进入的数量（virtual memory page ins ）  2）虚拟内存页面交换出去的数量（virtual memory page outs）  3）正在使用的虚拟内存空间的总数量  4）空闲物理内存的总数量  5）已用物理内存的总数量  **3、硬盘监控器（Disk Monitor）**  Disk Monitor instrument工具记录硬盘的读取和写入操作。该instrument工具  可以运行在单个进程或系统所有当前运行的进程之上。 跟踪面板可以被设置来默认显示以下和硬盘相关的数据，但你也可以配置它来显示其他类型的数据。默认情况下，它显示以下信息：  1）每秒写入硬盘的字节数量  2）每秒从硬盘读取的字节数量  3）每秒处理的写操作的数量  4）每秒处理的读操作的数量  **4、CPU监控器（CPU Monitor）**  CPU Monitor instrument工具记录系统的负载。该工具可以运行在单个进程或系统所有当前运行的进程之上。跟踪面板可以被设置来默认显示以下负载值，但你也可以配置它来显示其他类型  的数据。默认情况下，它显示以下信息：  1）系统产生的负载的数量  2）用户产生的负载的数量  3）系统总负载  **5、电量监控器（Energy Diagnostics）**  站在系统维度监控系统电量消耗情况，默认情况下，它显示以下信息：  1）屏幕亮度消耗电量  2）锁屏/唤醒消耗的电量  3）蓝牙消耗的电量  4）wifi消耗的电量  5）GPS消耗的电量 | | |
| **详细操作** | **1、监控网络流量具体步骤**  1.1启动Instruments，选择被测程序和测试类型“Blank“（空设置），并在Library中选择Net Activity运行库。  1.2点击左上角的“Run“按钮开始运行，并监控数据。数据表中会列出wifi流量情况（发出包数和接受包数，接收字节数和发出字节数）和非wifi流量情况即Cell类的网络流量情况（一切通过手机卡联网的数据流量，也包含发出包数和接受包数，接收字节数和发出字节数）  **实际使用场景举例：**在测试过程中我们可能需要测试这样的需求功能，即在有WiFi的情况下自动后台同步数据，无wifi的情况下不同步。使用这个流量监控就可以看到在关闭wifi只开启手机流量的时候，cell类的流量为空则表示不会在未经用户同意的情况下擅自使用同步功能，耗费流量。  **2、监控应用cpu、内存使用情况**  2.1在Instruments中的Trace选择页面中选择Activity Monitor选项。  2.2指定Target为被监控的程序后，点击”Run”开始启动监控，手机操作运行应用并监控cpu、内存使用情况。  **3、监控系统电量消耗情况**  3.1在Instruments中的Trace选择页面中选择Energy Diagnostics选项。  3.2选择被监控程序后点击”Run”开始启动监控，手机操作运行应用并监控电量数据。  3.3点击不同的电量监控项即可查看详细的数据（wifi、GPS耗电量等等）  在Energy Diagnostics的监控结果中不但有定性的监控数据，还有很多非常专业的定量数据。 | | |
| **补充**  **信息** | 此项测试用到的工具安装包在工具库里有。  详细操作使用说明请参见工具库对应文档。 | | |

#### 5.3内存泄露测试

##### 5.3.1 Android应用内存泄露测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | Android应用内存泄露测试 | **整理人员** | 唐晓文 |
| **简介** | 测试Android手机应用是否存在内存泄露。 | | |
| **详细技术说明** | **Android应用内存分配机制：**  为了能够使得Android应用程序安全且快速的运行，Android 的每个应用程序都会使用一个专有的Dalvik 虚拟机实例来运行，它是由Zygote 服务进程孵化出来的，也就是说每个应用程序都是在属于自己的进程中运行的。一方面，如果程序在运行过程中出现了内存泄漏的问题，仅仅会使得自己的进程被kill 掉，而不会影响其他进程（如果是system\_process 等系统进程出问题的话，则会引起系统重启）。另一方面Android 为不同类型的进程分配了不同的内存使用上限，如果应用进程使用的内存超过了这个上限，则会被系统视为内存泄漏，从而被kill掉。  **Android应用为什么会有内存泄露原因：**  Android 应用程序开发以Java语言为主，而Java编程中一个非常重要但却经常被忽视的问题就是内存使用的问题。Java的垃圾回收机制(Garbage Collection 以下简称GC)使得很多开发者并不关心内存使用的生命周期，只顾着申请内存，却不手动释放废弃的内存，而造成内存泄露，引起很多问题，甚至程序崩溃。Android的虚拟机Dalvik VM和java虚拟机JVM没有什么太大的区别，只是在字节码上稍做优化，所以Android应用开发中同样会出现内存泄露的问题。而且由于Android智能平台主要用于嵌入式产品开发，可用的内存资源更加稀少，更容易造成内存泄露。  **内存泄露的概念：**  一般我们常说的内存泄漏是指堆内存的泄漏。堆内存是指程序从堆中分配的，大小任意的（内存块的大小可以在程序运行期决定），使用完后必须显式释放的内存。应用程序一般使用malloc，calloc，realloc，new等函数从堆中分配到一块内存，使用完后，程序必须负责相应的调用free或delete释放该内存块，否则，这块内存就不能被再次使用，我们就说这块内存泄漏了。  在java程序中，如果已经不再使用一个对象，但是仍然有引用指向它，GC就无法收回它，当然该对象占用的内存就无法再被使用，这就造成内存泄露。可能一个实例对象的内存泄露很小，并不会引起很大的问题。但是如果程序反复做此操作或者长期运行，造成内存不断泄露，终究会使程序无内存可用，只好被系统kill掉。（我们这里讲的内存泄露主要指的是Dalvik虚拟机的内存泄露，而不是整个Android系统的内存泄露。）  **内存泄露造成的后果：**   1. 应用程序在使用过程中频繁崩溃，造成重要数据丢失。 2. 内存泄露占用虚拟机内存使应用运行缓慢。   **可能造成内存泄露的原因：**   1. 访问数据库游标没有关闭。（程序中经常会进行查询数据库的操作，但是经常会有使用完毕Cursor 后没有关闭的情况。如果我们的查询结果集比较小，对内存的消耗不容易被发现，只有在常时间大量操作的情况下才会复现内存问题） 2. 构造Adapter时，没有使用缓存的convertView（例如在列表下拉中，没有使用已经缓存的convertView，而每次都去新建view来用） 3. 没有在Activity的生命周期各方法中适当的释放资源。 4. Bitmap对象不在使用时调用recycle()释放内存（即处理大图没回收） 5. 没释放对象的引用   **所需工具：**DDMS+MAT（测试工具库获取） | | |
| **详细操作步骤** | **（一）工具准备**   1. DDMS—Update heap Gause GC   Heap是DDMS自带的一个很不错的内存监控工具，下图红色框中最左边的图标就是该工具的启动按钮，它能在Heap视图中显示选中进程的当前内存使用的详细情况。下图框中最右边的是GC工具，很多时候我们使用Heap监控内存的时候要借助GC工具，点击一次GC按钮就相当于向VM请求了一次GC操作。中间的按钮是Dump HPROF file，它的功能相当于给内存拍一张照，然后将这些内存信息保存到hprof文件里面，在使用我们的第二个工具MAT的时候会使用到这个功能。  QQ截图20111221142355.jpg   1. MAT(Memory Analyzer Tool)   Heap工具能给我们一个感性的认识，告诉我们程序当前的内存使用情况和是否存在内存泄漏的肯能性。但是，如果我们想更详细，更深入的了解内存消耗的情况，找到问题所在，那么我们还需要一个工具，就是MAT。这个工具是需要我们自己去下载的，可以下载独立的MAT RCP 客户端，也可以以插件的形式安装到Eclipse里面，方便起见，推荐后者。  安装方法：  1）登录官网<http://www.eclipse.org/mat/downloads.php>  2）下载MAT Eclipse插件安装包（红框所示，当然你也可是选择Update Site在线安装，个人觉得比较慢）  QQ截图20111221145828.jpg  3）安装  在Eclipse里面安装新软件，选择刚才下载的本地安装包进行安装  QQ截图20111221150352.jpg   1. **案例分析**   工具准备好了，那就来看看怎么使用。我们以Q+ for Pad为例，看看查找好友功能是否存在内存泄漏。  1、打开 eclipse 并切换到 DDMS 透视图，同时确认 Devices 、 Heap 和 logcat 视图已经打开了。  2、将Android设备（手机）链接到电脑，并确保使用“ USB  调试”模式链接。   1. 启动要测的应用，此时我们能看到下图所示的情况，被测应用有两个进程   4.jpg  4、选中main进程，点击Update Heap按钮,再点击GC按钮，查看该进程当前堆内存的使用情况  5.jpg  如何才能知道我们的程序是否有内存泄漏的可能性呢。这里需要注意一个值：Heap视图中部有一个Type叫做data object，即数据对象，也就是我们的程序中大量存在的类型的对象。在data object一行中有一列是“Total Size”，其值就是当前进程中所有Java数据对象的内存总量，一般情况下，这个值的大小决定了是否会有内存泄漏。可以这样判断： a) 不断的操作当前应用，同时注意观察data object的Total Size值； b) 正常情况下Total Size值都会稳定在一个有限的范围内，也就是说由于程序中的的代码良好，没有造成对象不被垃圾回收的情况，所以说虽然我们不断的操作会不断的生成很多对象，而在虚拟机不断的进行GC的过程中，这些对象都被回收了，内存占用量会会落到一个稳定的水平； c) 反之如果代码中存在没有释放对象引用的情况，则data object的Total Size值在每次GC后不会有明显的回落，随着操作次数的增多Total Size的值会越来越大，直到到达一个上限后导致进程被kill掉。   1. 在设备上操作被测功能：输入昵称，查找联系人，在查到的结果中不断向下翻页，最后退出（这个时候程序会不断的拉取联系人的头像，这是我们要关注的），然后我们点击下GC按钮，手动触发下垃圾回收，结果截图如下，图中红色标记的地方可以看出，执行查找联系人操作后，这两个数值明显增加了，GC操作也无法使之下降，我们可以怀疑，这个操作导致了内存泄漏。   QQ截图20111221190454.jpg   1. 使用MAT进一步分析，找到问题的根源。之前我们已经安装了MAT插件，所以这里我们只要选中main进程，点击Dump HPROF file按钮，就会跳转到MAT视图。在弹出的对话框中选择报告类型，一般选第一个就行。   QQ截图20111221190701.jpg  点击完成后，MAT会自动生成报告，列出几个内存占用比较大的可疑对象，MAT不会明确告诉我们这就是泄露，因为它也不知道这个东西是不是程序还需要的，只有程序员自己知道。图中很明显的占用内存较大的是头像资源，并且当退出查找联系人功能后，这些资源是需要释放的，陌生人的头像我们不需要缓存在内存里面。QQ截图20111221191241.jpg  点击Domain Tree按钮，可以按包进行分组。点开树形列表，可以更详细的看到类对象占用内存的大小。其中，Shallow Heap表示实例的内存使用总和，Retained Heap表示所有类实例被分配的内存总和，里面也包括他们所有引用的对象。  QQ截图20111221191520.jpg  从生成的数据中，我们发现有一千多个HashMapEntry对象，针对单个HashMapEntry对象继续追踪，最后找到了一个6KB左右的Bitmap，应该就是我们的头像资源。  QQ截图20111221195725.jpg  因此，为了回收内存，我们必须把刚才查找好友保存在内存中的头像资源释放。通过使用Heap 和MAT工具，能够更好的帮助测试人员发现及定位内存泄漏问题，也能帮助我们发下性能问题，找到内存优化的点。 | | |
| **补充**  **信息** | **小技巧：**  如果怀疑某个功能存在内存泄露，需要反复操作该功能，可以使用iTestin把需要反复操作的动作录制成脚本。录制成功后，把该脚本反复执行。同时观察DDMS中虚拟机内存回收情况，如果发现内存或者对象数直线上升，则需要开启MAT分析内存占用原因。  相关工具安装包见测试工具库。 | | |

##### 5.3.2 IOS应用内存泄露测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | IOS应用内存泄露测试 | **整理人员** | 唐晓文 |
| **简介** | 测试IOS应用是否存在内存泄露点。 | | |
| **详细技术说明** | **IOS下内存管理的特点：**  1、不支持垃圾自动回收，所以每次用完后要销毁。（当使用retain、new、alloc或copy方法创建一个对象时，该对象的保留计数器值为1。当不再使用该对象时，一定要负责向该对象发送一条release 或autorelease消息。这样，该对象将在使用寿命结束时被销毁。）  2、当通过任何其他地方获得一个对象时，则假设该对象的保留计数器值为1，而且已经被设置为自动释放，你不需要执行任何操作来确保该对象被清理。如果你打算在一段时间内拥有该对象，则需要保留（retain)它并确保在操作完成时释放它。  3、如果你保留了某个对象，你需要（最终）释放或自动释放该对象。必须保持retain方法和release方法的使用次数相等。(可通过 [obj retainCount];输出计数器的数值）  **根据以上特点，我们可以用如下方法进行测试：**  1、通过代码分析来进行静态测试（本质上是扫描代码创建和销毁对象的语句是否符合以上规则）  静态分析可以使用Xcode里面的“分析”（Analyze）功能进行测试。静态分析器在检查代码时会枚举所有的代码路径，分别检查所有的函数和方法。  2、使用Instruments中（Allocations和Leaks这两个组件组合使用）来动态检查内存泄露问题。Allocations组件是监控调用了alloc方法申请内存以后的内存使用情况。Allocations可以监控记录对象在生命周期内的内存引用计数变化。在对象被成功释放后，Allocations不再追踪。  Leaks会追踪内存泄露，一般要与Allocations一起使用比较容易定位问题。 | | |
| **一级**  **用例** | **1、使用Xcode里面的“分析”（Analyze）进行代码静态分析。**  静态分析不需要构建应用或执行程序，只是按一定的规则扫描源码。  1.1 启动Instruments中的Analyze工具运行静态分析。  1.2 点击静态分析的结果查看代码的问题。  这项测试一般让开发人员做比较合适，便于排查问题。  **2、使用Instruments中（Allocations和Leaks这两个组件）进行动态分析内存泄露。**  前提条件：测试手机已连上苹果电脑，手机上已安装好被测应用。  2.1在xocde中启动Instruments工具，在工具的操作界面选择leaks，如下图所示：  http://hi.csdn.net/attachment/201202/22/0_1329888744Sp07.gif  2.2运行会出现下面界面：  http://hi.csdn.net/attachment/201202/22/0_13298887721KZb.gif  2.3点击左上角的“Launch Executable”下拉框选择被测应用，如下图所示：  http://hi.csdn.net/attachment/201202/22/0_1329888796yqrx.gif  2.4 点击左上角的Record按钮开始启动测试，接着会自动启动应用记录内存回收情况（同时我们在手机上不断对应用进行使用操作，尽量遍历所有功能），在操作过程中leaks将默认每10秒钟自动检查一次内存泄露。  http://hi.csdn.net/attachment/201202/22/0_1329888823luYC.gif  2.5根据下图所示你也可以设置手动检测内存泄露。（定时的自动检测不受控制没有太多的针对性。一般我们常设置为手动检测，然后在我们认为可能发生内存泄露的时候点击“Check for leaks”按钮来检测是否存在内存泄露，例如：应用加载完一个新的内存模式后，或者是退出游戏后进入menu菜单时都可以手动点击检测。 ）  http://hi.csdn.net/attachment/201202/22/0_1329888841Rmrc.gif  2.6 一般应用运行了一段时间后，会发现一些内存泄露点，如下图所示（出现红柱就表示有内存泄露，我们可以点击堆栈，查看到哪一行代码泄露）：  http://hi.csdn.net/attachment/201202/22/0_1329888865YiI3.gif  2.7单击下图标识的按钮，将在右边打开一个窗口提供各种泄漏细节信息：  http://hi.csdn.net/attachment/201202/22/0_1329888884beqr.gif  2.8单击一个内存泄露点。详细视图将显示给你一个完整的堆栈跟踪到你泄露的内存分配。在这个例子中，单击第一个泄漏显示，泄漏发生在[ NSString initwithutf8string ]。如果你进一步跟踪堆栈你会看到最后的应用程序是[ instrumentstestviewcontroller viewDidLoad ]。  http://hi.csdn.net/attachment/201202/22/0_1329888902J6na.gif  2.9在扩展的详细视图，打开Xcode窗口，双击右边的问题，显示代码如下：  http://hi.csdn.net/attachment/201202/22/0_1329888916HIa2.gif  这样就能定位到具体问题，开发人员经过分析代码就可以知道如何修改了。  参考资料链接：http://blog.csdn.net/jianxuanlu/article/details/7282975 | | |
| **补充**  **信息** | 如何用Instruments监控应用程序cpu、内存、I/O等关键指标：  1、打开Instruments程序，选择Activity Monitor之后指定Target为你要测试的程序名。  2、开启录制按钮对被测应用性能指标进行监控。  3、在监控运行中可以看到各项指标值，点击进去可以看到详情。  4、Instruments还提供多次监控指标值对比功能，用于分析监控数据。 | | |

#### 5.4系统性能测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 系统性能测试 | **整理人员** | 唐晓文 |
| **简介** | 测试手机应用整体性能。 | | |
| **详细技术说明** | **手机应用类型说明：**  分两类应用程序：一种是客户端的应用程序，使用SDK 进行开发，安装包安装后使用；另一种是Web应用程序，使用Web标准技术进行开发，用户通过Web浏览器进行访问或者web容器控件进行访问。简单的说就是以下两类：   1. 本地应用（Native APP）——相当于电脑中安装的软件。 2. Web应用（Web APP）——相当于电脑上使用浏览器打开的网站，也可以在客户端应用程序中嵌入Web页面（使用WebView）。嵌入到应用程序的Web页面要根据情况进行特殊的设计。可以在应用程序和web页之间定义接口，允许Web页中的javascript调用应用程序的API。   对于性能测试来说，这2类应用需要根据它们的实现原理采不同的方法进行测试。  **针对不同应用的性能测试思路：**   1. 对于本地应用来说除了要测试服务器端，还需要测试本地客户端的性能，避免出现性能缺陷导致崩溃。还要测试与服务器端或其他系统交互的接口性能，避免接口造成的系统处理瓶颈。最后就是要测试服务器端的处理性能。 2. web app的性能测试和web网站性能测试类似，主要测试服务器处理能力，在此不赘述。   **手机应用的负载点分析**（导致一个应用性能不佳的因素）**：**  本地客户端：主要测试的是内存泄露，处理速度慢，资源占用过高等问题。（还与手机硬件配置、操作系统版本有关系）  服务器端（有服务器端的才需要）：服务器最大连接数、吞吐量、网络带宽因素、程序处理速度  接口：响应时间、吞吐量、处理效率。  更多具体的负载点需要根据具体项目具体分析。  **衡量性能的性能指标：**  本地客户端——首次加载时间，二次加载时间，使用过程中的能耗（电量、流量、内存、cpu）  Web客户端——首次加载时间，二次加载时间，使用过程中的能耗（电量、流量、内存、cpu）  服务器端（或接口）——最优并发数、最大吞吐量、平均响应时间  **测试方法策略：**  性能测试（有目标再加压，求可接受范围内最大值）  负载测试（一直压，直到资源不足，还能正常使用）  压力测试（一直压，直到系统无法处理，求极限值）  稳定性测试（以一定压力运行长时间，保持稳定）  容量测试（大数据量处理效率）  场景设计方法（先单场景，后混合场景—按业务情况混合）  设计场景中的具体加压数值时，需要参考业务量和业务比例。  **制定场景测试通过标准**（以下为示例）**：**  1、服务器平均响应时间不超过多少毫秒、错误率小于多少，资源占用必须在什么范围内。  2、客户端在XXXX条件下仍旧能正常运行。  3、客户端在Monkey随机加压下，随机点击10000次不崩溃。  4、客户端在使用时资源占用情况……，闲置时资源占用情况。 | | |
| **一级**  **用例** | 【本地应用客户端】   1. 使用Monkey/IOSMonkey加压测试内存泄露和程序稳定性（具体方法可参见“内存泄露测试”章节） 2. 测试首次加载启动时间 3. 测试二次加载启动时间 4. 主要页面的响应时间   【web应用客户端】  提交百度移动测试平台“性能检测”——>”即时监测”则会立即显示出web页面的性能数据。  （原理同YSlow工具，同样是分析网页请求效率，给出问题和优化建议。还支持监测不同地区不同运营商网络下web app页面性能。立即出结果）  【服务器端/接口】   1. 测试服务器端最优并发数。 2. 测试服务器端的最大吞吐量。 3. 手机资源不足时，遍历应用程序功能 | | |
| **二级**  **用例** | 【本地应用客户端】  使用监控工具测试应用的cpu消耗、内存消耗、流量消耗（具体方法可参见“能耗测试”章节）  【服务器端/接口】  测试服务器端在默认配置下的极限并发数。  测试服务器端在最优并发数下能否持续稳定运行。（具体操作可参见“稳定性测试”章节） | | |
| **三级**  **用例** | 【本地应用客户端】  负载测试：在手机资源快耗尽的时候（内存不足、SD卡空间不足、cpu占用率较高等情况），客户端能否正常安装、正常运行（遍历所有功能）。 | | |
| **补充**  **信息** | 百度移动云测试中心中的“性能监控”可以监控不同区域不同网络下客户端的访问情况。  百度移动云测试中心简称MTC（Mobile TestingCenter）  平台地址：<http://developer.baidu.com/yunzhongce>  登录账号：使用百度账号注册成为开发者  在“解决方案与服务”—>”其他”—>”性能监测（APM）”模块中使用。 | | |

### 6、安全性测试

#### 6.1本地数据安全

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 本地数据安全 | **整理人员** | 黄万冰、唐晓文 |
| **简介** | 本地数据安全测试这里主要是针对SD卡数据恶意篡改和sqlite数据库安全做的测试。 | | |
| **详细技术说明** | **【Android应用的本地数据安全】**  **Android应用的4种本地存储方式：**  分别是：SharePreference、SQLite、Content Provider和File。  由于Android系统中，数据基本都是私有的，都存放于“data/data/程序包名”目录下。  若要实现数据共享，正确方式是使用Content Provider。  **SQLite：** SQLite是一个轻量级的数据库，支持基本SQL语法，是常被采用的一种数据存储方式。所有的数据库都是储存于“data/data/应用包名/databases”目录下。其只有五种数据类型，分别是：NULL——空值、INTEGER——整数、REAL——浮点数、TEXT——字符串、BLOB——大数据(可用于存储图片)  **SharedPreference：** 除SQLite数据库外，另一种常用的数据存储方式，其本质就是一个xml文件，常用于保存应用的参数设置。其存储位置在/data/data/<包名>/shared\_prefs目录下。SharedPreferences对象本身只能获取数据而不支持存储和修改，存储修改是通过Editor对象实现。一个应用可以创建多个这样的xml文件。它只能存储boolean，int，float，long和String五种简单的数据类型，无法进行条件查询。  **File：** 即常说的文件（I/O）存储方法，可以把数据存在ROM或SD卡中，所以又分为内部存储（ROM）和外部存储（SD），常用于存储大数量的数据，但是缺点是更新数据将是一件困难的事情。存储位置：/data/data/<包>/files/\*\*\*.\*\*\*。（SD卡中有这个路径，ROM里也有这个路径）  **ContentProvider:** Android系统中能实现所有应用程序共享的一种数据存储方式，由于数据通常在各应用间的是互相私密的，所以此存储方式较少使用，但是其又是必不可少的一种存储方式。例如音频，视频，图片和通讯录，一般都可以采用此种方式进行存储。每个Content Provider都会对外提供一个公共的URI（包装成Uri对象），如果应用程序有数据需要共享时，就需要使用Content Provider为这些数据定义一个URI，然后其他的应用程序就通过Content Provider传入这个URI来对数据进行操作。它指向的文件在哪，哪就是它的数据存储路径。  注：/data/data文件夹需要有root权限才能访问。尽量不要使用真机测试，可以使用集成在elipse的DDMS工具界面查看模拟器AVD设备获取到该文件夹下的文件。  **Android4种本地存储可能存在的安全风险：**  1、SQLite数据库（关键数据保存后没加密、SQL注入）  Android中操作数据库可使用SQLiteOpenHelper或ContentProvider的方式。使用SQLiteOpenHelper操作数据库时，数据库存放在data/data/<packagename>/databases/目录，这个目录只能是应用自己访问，相对是安全的，但是对于root用户来说，这个目录也是透明的，因此，私密数据也要做到加密存储。  使用ContentProvider操作数据时，本质上也是使用SQLiteOpenHelper，这时需要在AndroidManifest.xml来注册这个Provider，注册Provider就对外提供了访问这个数据库的接口，其他应用就可以访问这个数据库了，为了数据库数据安全就需要控制访问，如果不想对外提供访问只需在AndroidManifest.xml注册Provider时设置android:exported="false"；如果想提供对外访问能力，最好设置android:readPermission和android:writePermission这两个属性，来分别指定对这个ContentProvider中数据读和写操作的权限。android数据库还要预防数据注入的攻击。  详细举例如下：  1）由于陌陌的留言板位置数据缓存在用户sqlite文件中，只要用户刷新过附近的人的留言板，加载过对方的留言板内容，那么你加载过的那条留言的GSP经纬度数据就存储在自己的sqlite中，对方的位置尽在你掌握之中。  2）登录使用的用户信息账号密码保存在SQLite数据库中，使用明文存储，手机被root后其保存的账号密码可以较容易的被获取到。特别是如果账号密码是保存在共享存储的数据库中，那更加加大了泄露的危险。  3）sqlite作为轻量级数据库，对于数据库查询，如果开发者采用字符串连接方式构造SQL语句，就会产生sql注入，导致敏感数据泄露的可能。可以输入类似数据检测是否有漏洞：  2' and ‘1'=’1  2′ and ’1′=’2  2′ or \_id <> ‘  2、Shared Preferences：和SQLITE数据库一样可以被ROOT权限用户获取，但安全性相对较好，一般用于存储配置信息无敏感数据，风险不大。  3、Content Provider：主要用于数据共享，存储的是文件型数据。主要就是要注意不要把敏感数据共享出去。  4、File（SD卡）： SD卡是一个公共的存储空间，只要在AndroidManifest.xml中申请了如下权限，就可以操作SD卡。也就是说SD卡数据是最不安全的，很容易被其他应用读取篡改，如果SD中存放数据，建议存放一些无关紧要的数据，重要数据加密存储或者存放到ROM中。  5、File（ROM）：虽然 ROM数据存储在/data/data/<package name>/files目录中，只允许当前应用访问，安全性较好，但是root后的机器也是可以被读取篡改的，这里的数据也不是绝对的安全，ROM的敏感数据也要加密存储。  **可用的解决办法：**  数据安全最重要的是解决数据加密的问题，密钥存储问题，选择合适的加密方法。  加密方法，密钥存储建议使用Native本地库实现。（在此不详细解释）  **【IOS的本地数据安全和源代码安全】**  由于IOS的封闭特性，使得IOS的数据安全相对于android要高一些。我们能正常获取到的数据有sqlite本地数据库的数据和应用的plist配置文件以及cookie文件。只要保障这些数据中没有敏感数据，且重要数据被加密即可保障本地数据的安全。（对于本地的重要数据，可以加密存储或将其保存到keychain中，以保证其不被篡改。）  以上说的是本地数据安全，除了数据安全安装包的程序安全也要注意。如果没有混淆加密代码会被黑客反编译查看到部分源码，进而获取业务逻辑和通信协议，利用其进行破坏。  对于IOS来说通过file、class-dump、theos、otool等工具，黑客可以分析编译之后的二进制程序文件，不过相对于这些工具来说，IDA的威胁最大。IDA是一个收费的反汇编工具，对于Objective-C代码，它常常可以反汇编到可以方便阅读的程度，这对于程序的安全性，也是一个很大的危害。因为通过阅读源码，黑客可以更加方便地分析出应用的通信协议和数据加密方式。 | | |
| **一级**  **用例** | **1、检查sqlite数据库本地数据存储安全（Android）：**  1.1、运行cmd，切换到android-sdk目录，运行adb shell。  1.2、切换至root用户后，进入目录：data/data/项目包名/databases。操作方式同linux  1.3、利用sqlite3 db命令打开数据库。若某些手机不支持sqlite3命令，可把需要查看的数据库拷贝出来，用loadsql等工具进行查看。  1.4、用sql语句查看数据库，主要关注默写隐私数据，如密码等，是否进行了加密保护处理。  注：如果是使用真机测试必须root，建议用模拟器运行避免权限干扰。类似问题可以用同样方法测试。  对于IOS来说，用ITools工具取出应用的sqlite数据库即可，其他操作同上。  **2、检查sqlite数据库是否存在SQL注入漏洞：**  在可以访问sqlite数据库，执行删除、修改、查询操作的功能界面上，依次录入各类SQL注入语句，查看返回结果。  3、对于Android应用应检查Shared Preferences的xml文件、ROM中保存的文件数据、SD卡中保存的数据、Content Provider共享存储中是否有敏感数据或不宜给外人共享的数据。如果有且必须要保存，则需要测试是否加密。  4、对于IOS来说，应检查配置文件plist中是否保存有敏感数据  5、解压安装包，查看是否所有文件代码均已加密。反编译安装包是否能获取源码。 | | |
| **二级**  **用例** | 暂无 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | Android安装包反编译工具可使用Apktool（具体参见框架工具部分）  IOS安装包反编译工具可使用IDA（在本文档中无介绍，感兴趣者可自行搜索） | | |

#### 6.2远程服务器/客户端安全

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 远程服务器/客户端安全 | **整理人员** | 黄万冰、唐晓文 |
| **简介** | 对手机应用在与远程服务器打交道过程中可能存在的安全问题进行测试。 | | |
| **详细技术说明** | **手机应用访问远程服务器可能出现的安全问题：**  基于http协议与远程服务器打交道的手机应用会面临着和web网站类似的一系列安全问题。主要包括以下几种：   1. SQL Injection(SQL注入)：   内容比较多，测试手段也比较多。请参见此文档：     1. Cross-site scritping(XSS):   跨站脚本漏洞（Cross Site Scripting，常简写作XSS）是Web应用程序在将数据输出到网页的时候存在问题，导致攻击者可以将构造的恶意数据显示在页面的漏洞。因为跨站脚本攻击都是向网页内容中写入一段恶意的脚本或者HTML代码，故跨站脚本漏洞也被叫做HTML注入漏洞（HTML Injection）。与SQL注入攻击数据库服务器的方式不同，跨站脚本漏洞是在客户端发动造成攻击，也就是说，利用跨站脚本漏洞注入的恶意代码是在浏览器中运行的。  手机应用程序中的web控件以及显示在手机浏览器中的web app，都有可能存在类似漏洞，测试原理和web的测试原理相似：简单的说，就是根据相关语言定义一些可能导致跨站脚本漏洞的函数（一般为输出函数），然后去找检查这些函数的参数是否由外部传入且未经过安全处理。比如PHP，就是检查echo、print函数的参数是否来自外部并且没有经过防跨站处理就直接显示到页面；对ASP来说，就是response.write之类的输出函数。   1. CSRF:(跨站点伪造请求) 2. Email Header Injection(邮件标头注入) 3. exposed error messages(错误信息) 4. 网络传输重要信息使用明文传输   以上6种方法中，目前在测试中比较常用的是SQL注入和第6点网络传输安全。 | | |
| **一级**  **用例** | 1. **SQL注入的测试方法**   1）找到带有参数传递的页面,如：搜索页面,登录页面,提交评论页面等等。  2）在参数或表单中加入某些特殊的SQL语句或SQL片断,如在登录页面的URL中输入HTTP://DOMAIN /INDEX.ASP?USERNAME=HI' OR 1=1—  3）验证是否能入侵成功或是出错的信息是否包含关于数据库服务器的相关信息。  4）如果能，说明存在SQL安全漏洞。   1. **跨站脚本攻击的测试方法：**   2.1．找到带有参数传递的URL,如 登录页面,搜索页面,提交评论,发表留言 页面等等  2.2．在页面参数中输入如下语句(如:Javascrīpt,VB scrīpt, HTML,ActiveX, Flash)来进行测试：   |  | | --- | | <scrīpt>alert(document.cookie)</scrīpt> |   2.3．当用户浏览 时便会弹出一个警告框，内容显示的是浏览者当前的cookie串,这就说明该网站存在XSS漏洞   1. **Email Header Injection(邮件标头注入)测试方法**   3.1、如果表单用于发送email,表单中可能包括“subject”输入项（邮件标题），我们要验证subject中应能escape掉“\n”标识  3.2、输入email如：hello\ncc:spamvictim@example.com  3.3、查看结果，如果允许用户使用这样的subject，那他可能会给利用这个缺陷通过我们的平台给其他用户发送垃圾邮件。   1. **exposed error messages(错误信息)测试方法**   4.1.查看访问错误的页面，或返回的错误信息。  4.2.若给出的信息包含程序代码，或服务器IP等信息。需要进行屏蔽。  **5、检测网络传输中是否使用明文传输**  5.1在Windows下，可以使用Fiddler软件来将自己的电脑设置成代理服务器，从而截取应用的网络请求  5.2截取到应用发送的请求后，分析目标应用在通信协议上是否有安全问题。例如：明文发送用户登录名和密码、使用get方式传输用户名和密码（发送时使用了Http Get的方式，而GET的URL数据一般都会保存在服务器的Access Log中，所以黑客一旦攻破服务器，只需要扫描Acesss Log，则可以轻易获得所有用户的明文密码）。 | | |
| **二级**  **用例** | 暂无 | | |
| **三级**  **用例** | 暂无 | | |
| **补充**  **信息** | IOS应用安全测试知识相关链接：  http://www.csdn.net/article/2014-04-30/2819573-the-secret-of-app-dev-security | | |

### 7、易用性测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试点** | 易用性测试 | **整理人员** | 唐晓文 |
| **简介** | 测试应用是否好用，便于使用。 | | |
| **详细技术说明** | **易用性所包含的内容：**  包含易理解性、易学习性和易操作性和吸引用户使用的能力。  易用性测试方法有：静态测试；动态测试；动态和静态结合测试。  UI界面易用性测试，我们以谷歌发布的Android 4.0设计规范作为我们的测试依据。  **Android 4.0设计准则内容包含如下几点：**   1. 不使用其他平台特有的图标（页面元素要符合Android系统） 2. 不在界面的下方选用选项卡tabs（Android系统里，选项卡（tabs）会固定放在屏幕上方，不会放在下方。这点与ios 系统不同。） 3. 不在操作栏上使用返回样式的按钮 4. 不在内容列表里使用向右箭头 5. 不要有太多需要用户选择的选项。 6. 只给出当前可操作的必要操作，隐藏无法操作的按钮或功能。 7. 要让用户知道他在哪，怎么回去。 8. 提示、说明的句子要尽可能的简短，不能重复描述。 9. 永不丢失用户花心思创建、编辑的东西，即便是他们还没保存。 10. 不要骚扰、打断用户的操作。 11. 尽量使用通用的操作方式，类似的功能操作必须一致。 12. 出了问题后，应给出便于理解的提示，而不是专业的技术术语。 13. 把复杂的工作流程分解为多个小步骤。 14. 把专业的操作简单化，黑盒化。用户不需要知道原理，只需要得到结果。 15. 重要的核心操作一定要快。 16. UI尽可能多的兼容不同尺寸、分辨率的屏幕。 17. 触摸反馈（使用颜色和光作为触摸的反馈，加强手势行为的结果，暗示哪些操作能用，哪些操作被禁用。无论何时用户触摸了一个可操作区域，你的app都要提供视觉反馈。使用户知道哪些可操作。） 18. 上下滑动和左右滑动要有边界。当用户尝试滚动超过内容边界时，要给出一个明确的视觉线索。 19. 首屏应该同时满足新用户和老用户。 20. 如果页面是首页则不应该出现返回或向上按钮。（向上层级按钮和返回键的区别，向上按钮可以确保用户始终在app中，而返回按钮可以进入另一个app中，是系统级别的返回） 21. 确保页面在横屏、竖屏时期都具有相同的功能。 22. 用横划来代替上一个、下一个按钮。 23. 通知不能太多频繁，骚扰用户。注意不要在用户做关键流程的时候发无关通知打断思路。 24. 不要通知用户他不能操作的功能。 25. 通知里面要包含app图标作为第二图标，这样用户能够区分出是哪个app发出的通知。导航到正确的地方。 26. 当应用发送了多条通知时最好合并展示。 27. 进行中的通知不能被删除（例如：在播放音乐）。 28. 在每个界面上，应该只使用一个进度指示，并应根据周围环境的大小适应。例如，最大的进度圈在空白的内容中显示良好，但在一个小对话框就不适用。 | | |
| **一级**  **用例** | 1. 应用界面是否符合Android设计规范。 2. 应用界面是否直观、不拥挤、布局合理、无当前不可操作的功能按钮。 3. 列表的滚动操作要有边界提示。（到了边界高亮提示） 4. 应用操作是否具有一致性。（确保相似的操作以相似的方式进行，例如：确定取消按钮的位置名称一致，成功失败提示语一致，提示框按钮样式等一致，术语命名一致，快捷键菜单统一） 5. 应用是否有帮助系统（功能实用引导、帮助文档等）。 6. 应用使用起来是否舒适、流畅。（提示准确明确，操作不需要思考，无歧义，所见即所得，错误提示有效能看懂，根据提示操作能成功，响应速度快。） 7. 应用界面、提示文字的正确性（拼写正确，无语法错误） 8. 界面字体的风格是否一致，相同区域的文字字体大小是否一致。 9. 页面深度层级一般控制在3层左右。 10. 对于有延时可能的操作必须要有，滚动条提示操作正在进行中，并禁止用户进行其他操作。 11. 页面在横竖屏切换之后，横屏、竖屏界面上的功能要一致，不能有缺失。 12. 通知不能过于频繁（检查是否存在某些操作设置导致通知频繁）。 13. 通知要有logo标识或者说明是哪个应用的，而且跳转位置正确。 14. 发送了多条通知是否合并展示。且单条点开的跳转位置和多条点开跳转位置最好不同。 15. 每个界面上，应该只使用一个进度指示。且相同类型的进度指示，样式应该一致。 | | |
| **二级**  **用例** | 1. 一些会重复提示的操作，最好带有记住功能（例如：提示是否开放权限，有个“记住我的选择“选框） 2. 提示、说明的句子要尽可能的简短。 3. 不要让用户选择太多，如果无法避免选择也要给他提供后退的后路。 | | |
| **三级**  **用例** | 19、注意不要出现Android应用使用IOS的特有操作习惯，IOS应用也最好不要出现Android应用的特有操作习惯。 | | |
| **补充**  **信息** | 1. Android4.0设计规范可以到WIKI搜索下载。 2. 建议使用百度云测试平台的“云众测”搜集真实用户的体验建议，提高用户体验度。   百度移动云测试中心简称MTC（Mobile TestingCenter）  平台地址：<http://developer.baidu.com/yunzhongce>  登录账号：使用百度账号注册成为开发者 | | |

### 8、自动化测试

#### 8.1基于UI的Android自动化测试

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | 基于UI的Android自动化测试 | **整理人员** | 唐晓文 |
| **功能**  **简介** | 基于页面的GUI的Android自动化测试 | | |
| **详细技术说明** | **适用自动化的测试场景：**   1. 需要反复执行的测试 2. 需要长时间自动运行(测试稳定性) 3. 回归拨测   **可以选用的测试工具有：**  MonkeyRunner\ITestin\Robtium | | |
| **测试**  **说明** | **测试步骤流程如下：**   1. 准备好测试用例 2. 根据测试用例录制、编写脚本（单个脚本应包括3部分：准备、运行、清场还原） 3. 准备测试数据，环境 4. 执行脚本 5. 出具测试报告，得出结果 | | |
| **工具库地址** | 工具使用详情请参见“自动化测试工具”章节，在此不赘述。 | | |

#### 8.2 IOS自动化测试

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | IOS应用自动化测试 | **整理人员** | 唐晓文 |
| **功能**  **简介** | 通过编写脚本或录制脚本自动化测试IOS应用的功能 | | |
| **详细技术说明** | 实现IOS自动化测试常用的手段有两种：一种是写脚本测试，一种是录制回放测试。  1）编写脚本测试可以使用IOS自带的Instruments 中的UI Automation  2） 录制回放测试可以使用ITestin工具录制（需要IPhone手机越狱），也可以用UI Automation录制脚本测试。  对于web app应用则可以在浏览器中使用QTP进行自动化测试，在此不讲述这类测试方法，主要讲的是Native app的自动化测试。  **使用UI Automation开发脚本的前提条件：**  1）系统运行权限限制，所以对程序的签名有一定的要求（必须是开发者签名的测试程序）。  2）应用程序控件识别能力，必须是大部分控件都能识别。（开发程序的时候就要处理UI控件的属性为Accessibility，这样才能识别控件）  3）因为测试脚本要针对控件操作，所以在编写脚本前，必须要了解被测应用界面控件信息。（可以在详细日志区通过查看相关控件的截图来定位控件，获取控件名称等信息）  4）脚本语言使用的是js，因此需要有一定的js语言基础。 | | |
| **测试**  **说明** | **测试执行思路如下：**   1. 准备好测试用例 2. 根据测试用例编写脚本（单个脚本应包括3部分：准备、运行、清场还原） 3. 准备测试数据，环境 4. 执行脚本 5. 出具测试报告，得出结果   **UI Automation开发脚本步骤：**  1、连接手机、打开Instrument，选择Automation，在Target中选择被测应用程序。  2、在左下角的New Script点击“Add“下拉框的”Create“按钮新建脚本。  3、在脚本编辑区域，编写脚本。（获取相关控件，对相关控件做操作）  4、脚本编写完成后，点击”Run”按钮进行回放测试。  关于编写脚本有很多技巧和语言上的问题需要学习解决（例如：模拟滑动效果、输出自动判断的测试结果、添加延迟等等），在此不详细讲述。感兴趣的同学可以看书《IOS测试指南》  **UI Automation录制脚本步骤：**  如果对于写脚本感到很吃力也可以使用Automation的录制脚本功能进行脚本创建。步骤如下：  1、连接手机、打开Instrument，选择Automation，在Target中选择被测应用程序。  2、点击正下方的红色录制按钮，开始录制脚本。  3、录制开始后，在手机上打开应用做一些功能操作。手动操作的同时会自动生成自动化脚本。（录制功能支持真机也支持模拟器录制）  4、录制生成的脚本会显示在脚本编辑区域。  5、脚本录制完成后，根据测试需要再修改加工下脚本、删除一些不必要的多余操作，使其更符合我们的测试目的，回放起来更流畅准确。  6、脚本完成后即可点击左上角的回放功能执行测试脚本了。  **关于ITestin上录制脚本的操作和Android应用的一样，故在此不在赘述。** | | |
| **补充**  **说明** | 暂无 | | |

## 四、框架测试工具

### 1、操作调试类工具

#### 1.1 ADB

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | ADB | **整理人员** | 唐晓文 |
| **功能**  **简介** | adb是Android Debug Bridge的缩写，意为Android系统的调试桥。通过adb我们可以在Eclipse中方便通过DDMS来调试Android程序，其实这就是用于谷歌Android系统的debug调试工具。除了上述的操作功能之外，我们还可以通过adb管理设备或手机模拟器的状态。还可以进行很多手机操作，比如刷ROM系统升级、运行shell命令等等。 | | |
| **详细技术说明** | adb的工作方式比较特殊，采用监听Socket TCP 5554等端口的方式让IDE和Qemu通讯，默认情况下adb会daemon相关的网络端口，所以当我们运行APK安装器时adb进程就会自动运行。  ADB可以管理设备或手机模拟器的状态，还可以进行以下的操作：  a、快速更新设备或手机模拟器中的代码，如应用或Android系统升级；（常用）  b、在设备上运行shell命令；（常用）  c、管理设备或手机模拟器上的预定端口；  d、在设备或手机模拟器上复制或粘贴文件。  **一些常用的操作：**  进入Shell: adb shell  通过上面的命令，就可以进入设备或模拟器的shell环境中，在这个Linux Shell中，你可以执行各种Linux 的命令，另外如果只想执行一条shell命令，可以采用以下的方式：  adb shell [command]  如：adb shell dmesg会打印出内核的调试信息。  (Android的linux shell做了大量精简，很多linux常用指令都不支持)  上传文件: adb push  下载文件: adb pull  安装程序: adb install  卸载软件: adb shell rm /data/app/  补充一点，通过adb安装的软件(\*.apk)都在”/data/app/”目录下，所以安装时不必制定路径，卸载只需要简单的执行”rm”就行。  结束adb: adb kill-server  显示android模拟器状态:  adb devices (端口信息)  adb get-product (设备型号)  adb get-serialno (序列号)  等待正在运行的设备: adb wait-for-device  端口转发:  adb forward adb forward tcp:5555 tcp:1234  (将默认端口TCP 5555转发到1234端口上)  查看bug报告: adb bugreport  访问数据库:  adb shell sqlite3  访问数据库SQLite3  记录无线通讯日志：  adb shell logcat -b radio  一般来说，无线通讯的日志非常多，在运行时没必要去记录，但我们还是可以通过命令，设置记录在应用程序配置文件:“AndroidManifest.xml”中。 | | |
| **使用**  **说明** | **上传文件：**  1、启动一个模拟器。  2、在cmd中执行命令：adb push "<要上传文件的路径>" sdcard/<上传后文件名称>  **安装应用：**   1. 启动模拟器，在cmd中执行命令：adb devicse 2. adb install "<要安装文件的本机路径>“   **进入模拟器shell：**   1. 启动模拟器，在cmd中执行命令：adb devicse 2. adb shell 3. 输入命令exit，退出shell | | |
| **工具库地址** | 【29】ADB\_win32  更多详情请参见“Android\_ADB\_使用详解.pdf“文档 | | |

#### 1.2 AVD

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | AVD | **整理人员** | 唐晓文 |
| **功能**  **简介** | Android设备模拟器。可以创建多款不同配置的测试设备。对于测试来说需要了解模拟器创建的设备存储路径在哪以及能够模拟哪些真机操作，怎么模拟。 | | |
| **详细技术说明** | 1. **工具路径：**Android SDK安装路径Tools文件夹下。 2. **可以模拟的操作：**   AVD的关机原理  AVD的上网功能  AVD的快照功能  详细内容请参见工具库“30【Android设备模拟器】AVD\_win32“文件夹中的”简单Android\_AVD的创建.pdf“文件   1. **配置文件路径：**   系统会在系统默认文件路径下自动创建一个.android文件夹，所创建的有关avd的配置信息则被保存在该.android目录下。例如：C:\Users\tangxiaowen\.android\avd   1. **创建SDCard镜像文件**   进入Android SDK目录下的tools子目录，运行  mksdcard -l sdcard 512M e:\sdcard.img  这样就创建了一个512M的SD卡镜像文件:查看E盘，E盘下会有一个sdcard.img 文件。   1. **命令行下运行AVD时加载sdcard**   进入Android SDK目录下的tools子目录，运行  emulator -avd MyAVD -sdcard android.img  如果不想每次都加-sdcard $sdcard\_path，有两个办法：  一：可以把sdcard.img 复制到C:\Documents and Settings\Administrator\.android\avd\下对应的AVD目录下。  二：修改配置文，C:\Documents and Settings\Administrator\.android\avd\AVD\_NAME.avd\，打开config.ini，加入以下行：sdcard.path=你的SD卡路径。   1. **AVD路径设置**   前面提到，不管用Eclipse还是从命令行创建Android Emulator时，相应的文件是被放置到 “C:\Documents and Settings\Administrator\.android“ 下面。如果想改变AVD的路径设置，也有2个办法：  解决办法一：  新添加一个环境变量，名字为 "ANDROID\_SDK\_HOME”，然后把变量值改为你想将".android”文件夹(AVD信息)所放置的位置。  解决办法二：  修改配置文件中的路径，比如我的AVD名称是MyAVD, 打开C:\Documents and Settings\Administrator\.android\avd目录中的文件MyAVD.ini文件，找到target=android-5path  =C:\Documents and Settings\Administrator\.android\avd\MyAVD.avd  修改其路径即可。  遇到ERROR: unknown virtual device name 这样的问题一般是由于AVD文件路径引用错误造成的。 | | |
| **使用**  **说明** | **创建AVD**  **1）命令行方式创建AVD（不建议用此方式）**  进入Android SDK目录下的tools子目录，运行：  android create avd -n my\_android2.2 -t 2  如：android create avd –name adk\_1\_5\_version --target 2  在创建avd文件得第二步，控制台提示“是否自定义自己得设备”，默认选项是"no"，如果输入 "y"，接下来可以一步步根据提示，定制自己得模拟器性能参数。  这里t代表target目标平台的API Level，我们可以使用android list targets来查看这里t的可选参数。  进入Android SDK目录下的tools子目录，运行：android list targets  **2）开发工具创建AVD**  1、在Eclipse中。选择Windows > Android SDK and AVD Manager或直接从开始中启动。  2、点击左侧面板的Virtual Devices，再右侧点击New。  3、填入Name，选择Target的API，SD Card大小任意，Skin随便选，Hardware目前保持默认值。  4、点击Create AVD即可完成创建AVD。 | | |
| **工具库地址** | 【30】AVD\_win32  更详细的使用资料请参见工具库文档“简单Android\_AVD的创建.pdf“ | | |

#### 1.3 DDMS

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | DDMS | **整理人员** | 唐晓文 |
| **功能**  **简介** | DDMS 的全称是Dalvik Debug Monitor Service，是 Android 开发环境中的Dalvik虚拟机调试监控服务。 | | |
| **详细技术说明** | **DDMS能提供的功能：**  测试设备截屏，针对特定的进程查看正在运行的线程以及堆信息、Logcat、广播状态信息、模拟电话呼叫、接收SMS、虚拟地理坐标等。  **DDMS怎么工作的：**   1. 每一个Android应用都运行在一个Dalvik虚拟机实例里，而每一个虚拟机实例都是一个独立的进程空间。虚拟机的线程机制，内存分配和管理，Mutex等等都是依赖底层操作系统而实现的。所有Android应用的线程都对应一个Linux线程，虚拟机因而可以更多的依赖操作系统的线程调度和管理机制。 2. DDMS在IDE与设备或模拟器之间的起着中间人的角色。 3. DDMS启动时会与ADB之间建立一个device monitoring service用于监控设备。当设备断开或链接时，这个service就会通知DDMS。 4. 当一个设备链接上时，DDSM和ADB之间又会建立VM monitoring service用于监控设备上的虚拟机。 5. 通过ADB Deamon与设备上的虚拟机的debugger建立链接，这样DDMS就开始与虚拟机对话了。   **关于DDMS的监听端口分配：**  对于设备上每个虚拟机的debugger，DDMS都会开启一个监听端口，端口从8600开始。如果有更多的终端或者更多App进程将按照这个顺序依次类推。DDMS通过8700端口接收所有终端的指令。  **如何启动DDMS**（两种）：  1、直接到Android SDK 安装路径的Tools文件夹下双击ddms.bat即可。  2、在Eclipse中直接点击启动DDMS插件即可。  DDMS对Emulator和外接测试机同等效用,如果系统检测到它们(VM)同时运行,那么DDMS将会默认指向Emulator。  **DDMS左边面板介绍：**   1. 左边显示了所有当前能找到的所有模拟器或设备列表和每个设备当前正在运行的虚拟机列表。虚拟机是按程序的包命来显示的。 2. 通过这些列表可以找到运行着想调试的activity的虚拟机。每个虚拟机旁边的是“debugger pass-through”端口，链接到其中一个端口就会链接到设备上对应的虚拟机。不管如何，在用DDMS时，只需要链接到8700端口，因为DDSM转发所有的通信到当前选择的虚拟机。这样，就不用在每次切换虚拟机是重新配置debugger端口。 3. 当一个正在运行的程序调用waitForDebugger()函数时，客户端名字旁边会显示一个红色的icon,知道debugger连上对应的虚拟机，这是debugger会变成绿色。 4. 如果看到叉icon，着意味着DDMS用于不能打开虚拟机的端口而不能建立debugger与虚拟机建立连接。如果看到所有的虚拟机是这样，很可能是有另外一个DDSM实例在运行。   **DDMS右边面板介绍：**  【info】这里显示关于选中的VM的一般的信息，包括进程ID，包名，和虚拟机版本。  【Threads】 线程视图列出了此进程的所有线程。  ID:虚拟机分配的唯一的线程ID,在Dalvik里，它们是从3开始的奇数。  Tid：linux的线程  ID：For the main thread in a process, this will match the process ID.  Stauts：线程状态。（其中包括以下几类状态——running:正在执行程序代码、sleeping：执行了Thread.sleep() 、monitor：等待接受一个监听锁。wait: Object.wait() 、native：正在执行native代码 、vmwait：等待虚拟机，zombie：线程在垂死的进程，init：线程在初始化（我们不可能看到），starting：线程正在启动（我们不可能看到））  utime：执行用户代码的累计时间  stime：执行系统代码的累计时间   name：线程的名字  【VM Heap】  展示一些堆的状态，在垃圾回收其间更新。当选定一个虚拟机时， VM Heap视图不能显示数据，可以点击右边面包上的带有绿色的”Show heap updates”按钮，然后在点击”Cause GC “实施垃圾回收更新堆的状态。     【Allocation Tracker】  在这个视图里，我们可以跟踪每个选中的虚拟机的内存分配情况。点击”Start Tracking”后点击”Get Allocations “就可以看到。  【Emulator Control 】  在这里，可以模拟一些设备状态和行为。  Telephony Status：改变电话语音和数据方案的状态，模拟不同的网络速度。  TelePhony Actions：发送模拟的电话呼叫和短信到模拟器。  Location Controls：发送虚拟的定位数据到模拟器里，我们就可以执行定位之类的操作。可以收工的在Manual里输入经度纬度发送到模拟器，也可以通过GPX和KML文件。  【其他】  1，File Explorer  通过Device > File Explorer 就打开File Explorer。这里可以浏览文件，上传上载删除文件，当然这是有相应权限限制的。  2，Screen Capture  通过Device > Screen Capture 就可以截图。  3，Exploring Processes  通过Device  > Show process status ，这里的信息是通过shell命令”ps -x”输出的。  4，Examine Radio State  通过Device > Dump radio，检测广播状态。  5，Stop a Virtual Machine  通过Actions > Halt VM ，停止一个虚拟机。 | | |
| **使用**  **说明** | **用来查看进程：**    1显示的是运行的应用名称，2现实的是对应的进程号，3显示的是对应的监听端口。  **用来监控进程的虚拟机内存占用、回收情况：**   1. 选中被测应用进程，点击左上角按钮，添加内存监控。 2. 点击右边“VM Heap”虚拟机堆监控，也就是虚拟机的内存监控tab页。 3. 内存监控页面的数据是在每触发一次内存回收才刷新。（可以等手机自动回收，也可以按“Cause GC”按钮手动触发GC内存回收） 4. 一般边操作手机功能，变查看内存情况。如果内存居高不下多次回收，对象和内存没有明显减少则存在内存泄露。     **用来监控系统日志：**    **用来模拟设备的一些状态和行为**（Emulator Control 模块）**：**  Telephony Status：改变电话语音和数据方案的状态，模拟不同的网络速度。  TelePhony Actions：发送模拟的电话呼叫和短信到模拟器。  Location Controls：发送虚拟的定位数据到模拟器里，我们就可以执行定位之类的操作。有三种方式可以实现：  Manual：手动为终端发送二维经纬坐标。  GPX：通过GPX文件导入序列动态变化地理坐标,从而模拟行进中GPS变化的数值.  KML：通过KML文件导入独特的地理标识,并以动态形式根据变化的地理坐标显示在测试终端  注意：此功能只能在模拟器（AVD）中使用，真机无法使用。  **监控网络流量**（Network模块）：可以调采样速度，按开始停止按钮即可。这里监控的是被选中应用的流量使用情况。 | | |
| **工具库地址** | Android SDK安装后自动带有DDMS，无需再安装。 | | |

#### 1.4 Android打包工具Apktool

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | Apktool | **整理人员** | 唐晓文 |
| **功能**  **简介** | APK文件解压重打包的工具 | | |
| **详细技术说明** | Android的安装包为APK后缀的安装包。  有时在测试过程中我们需要把打好的apk包里面的签名文件换成其他签名文件。  这时就需要用到这款工具“Apktool”（网友自行开发的小工具）。  可以解压、反编译、打包apk文件。无需安装，打开即可用。  **前提条件：**需要有JDK环境 | | |
| **使用**  **说明** | **双击打开工具后，根据提示操作即可。**    **注意事项：**  1、反编译非系统apk前需要把APK命名为123.apk  2、反编译系统apk时需要将Rom的framework-res.apk放置在此目录下，选择数字5执行安装！如果是HTC的Rom，你还需要将com.htc.resources.apk也安装一次！  3、反编译之前如果目录里有上次反编译产生的apk,framework文件夹清除，否则不能正常反编译！数字7可以快速清理！  4、路径默认放在D盘根目录下面，大家不要自己改动否则可能会出现一些文件丢失的情况！ | | |
| **工具库地址** | 测试工具库中可以下载。 | | |

#### 1.5 ITunes和ITools

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | ITunes和ITools | **整理人员** | 唐晓文 |
| **功能**  **简介** | ITunes是IPhone的官方联机管理工具。  ITools是PC上的一款IOS手机管理工具。 | | |
| **详细技术说明** | **【iTunes】**  iTunes前身是音乐管理器。目前主要用于iPhone与iTunes的同步。  同步内容范围很广包括音乐、视频、通讯录、日程、邮件、书签、铃声、照片等等。下载安装苹果商店的应用。平时工作中使用较少，只是它是安装ITools工具的前提条件，ITools依赖它的一些组件运作。注意：据网上用户反馈，在ITunes上使用和手机同步功能有可能会造成一些文件和安装程序丢失。  **【ITools】**  安装程序应用，查看应用的文件夹，查看文件系统，上传导出文件，备份恢复信息，查看日志，导出崩溃日志文件，录制屏幕动作。（测试时常用） | | |
| **使用**  **说明** | **iTunes的使用见以下链接页面：**  http://jingyan.baidu.com/article/76a7e409a01708fc3b6e15ba.html  **iTools的使用见以下链接页面：**  http://jingyan.baidu.com/article/495ba84130d36838b30ede23.html | | |
| **工具库地址** | 类似的软件还有91助手。 | | |

#### 1.6 IOS常用小工具

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | IOS常用小工具 | **整理人员** | 唐晓文 |
| **功能**  **简介** | IOS常用小工具有：用于查看plist配置文件的plistEditor，用于修改实时内存的iGameGuardian工具，用于查看cookie文件的python脚本BinaryCookieReader.py，用于修改实时定位坐标的工具“任我行”。 | | |
| **详细技术说明** | **【plistEditor】**  IOS应用的配置文件后缀为plist，在测试过程中可直接打开并编辑修改Plist配置文件，绿色软件无需安装。  **【BinaryCookieReader.py】**  这是一个Python脚本，运行前要安装Python环境，主要是把Cookies文件编译成可以看得懂的文本文件。  **【iGameGuardian】**  可用于实时查询修改IPhone设备中的内存。特别是测试游戏的时候，可以实时修改游戏分值，但是要在越狱的手机上才能使用。  **【任我行】**  在测试定位功能时，我们需要切换当前地理位置的坐标，这时就需要这款工具来修改手机上的坐标位置，让应用切换坐标，测试坐标切换后应用是否能正常运行。需要购买使用。 | | |
| **使用**  **说明** | **iGameGuardian的使用如下：**  <http://www.pc6.com/edu/61100.html>  **任我行的使用方式如下：**  http://www.locationholic.com/CN/index\_CN.htm  其他工具使用比较简单，在此不赘述。 | | |
| **工具库地址** | 以上工具，可在测试工具库中下载（【IOS应用】测试工具包） | | |

#### 1.7 IOS自带的Instruments工具

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | Instruments | **整理人员** | 唐晓文 |
| **功能**  **简介** | Instruments工具是IOS自带的测试工具。不仅可以运行自动化测试还可以查看内存泄露、网络连接、CPU内存电量使用情况等一系列数据。 | | |
| **详细技术说明** | **工具运行环境：**  Instruments工具集成在IOS开发工具Xcode中，所以要使用这个必须安装IOS开发环境。  最好使用苹果电脑真机安装，虚拟机上安装IOS操作系统会有很多问题很难解决。  **工具各功能模块简介：**  xcode内存管理工具－instrument  1、左上角的区域有3个按钮，分别是暂停（暂停自动化测试或暂停监控）、启动/停止（这个按钮在自动化测试中是启动停止的意思，在监控数值时是开始监控的意思）、是否循环（在回放自动化测试脚本时按下这个按钮，就会循环不停的回放）。  2、中间区域是负责展示运行状态，例如：加了一个监控就会新生成一行，监控状态就显示在对应的第二列中。例如上图：加了Leaks，右边对应显示这个监控项监控到的信息。  3、左下角区域主要是一些设置项的显示。  4、右下角主要是显示日志和脚本信息。  5、最右边的区域显示的是详细日志。在4中选择一条日志，就会在这显示出详细信息。 | | |
| **使用**  **说明** | 在之前的测试点中有详细描述，在此不赘述。 | | |
| **工具库地址** | 暂无 | | |

### 2、压力测试工具

##### 2.1Monkey

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | Monkey | **整理人员** | 唐晓文 |
| **功能**  **简介** | Monkey是一个命令行工具 ，可以运行在模拟器里或实际设备中。它向系统发送伪随机的用户事件流，实现对正在开发的应用程序进行压力测试。 | | |
| **详细技术说明** | **Monkey运行原理：**  在Monkey运行的时候，它生成事件流，并把它们发给系统。同时，Monkey还对测试中的系统进行监测，对下列三种情况进行特殊处理：   1. 如果限定了Monkey运行在一个或几个特定的包上，那么它会监测试图转到其它包的操作，并对其进行阻止。 2. 如果应用程序崩溃或接收到任何失控异常，Monkey将停止并报错。 3. 如果应用程序产生了应用程序不响应(application not responding)的错误，Monkey将会停止并报错。   **Monkey的参数选项：**  大致分为四大类：   1. 基本配置选项，如设置尝试的事件数量。 2. 运行约束选项，如设置只对单独的一个包进行测试。 3. 事件类型和频率。 4. 调试选项。   更多Monkey指令参数请参见帮助文档。  按照选定的不同级别的反馈信息，在Monkey中还可以看到其执行过程报告和生成的事件。  **注：**  Monkey执行的事件动作是随机的，不可复现的，对于问题的排查定位不是很方便。只能通过出错后它报出的错误日志来分析。使用Monkey一定要注意每一页的测试数据比较饱满，测试动作比较丰富，否则很可能卡死在一个页面上一直动不了。 | | |
| **使用**  **说明** | **案例步骤说明：**   1. 手机中启动应用，打开到被测页面，用usb连接电脑。 2. 使用ADB连接手机，进入abd shell（启动cmd，如果没有设置环境变量就先CD到Android安装路径下的Tools文件夹下，执行如下命令）   adb devices  adb shell   1. 输入monkey指令: monkey -p com.cmcc.wificity -v -v -v 10   **指令说明：**  -p参数： 表示指定测试的程序是com.example.android.apis  -v参数： 表示查看monkey生成的一些详细的随机的事件名。  （3个-v表示能看到的信息最多，一个-v也可以）  100参数： 表示发出的测试事件总数为100  **跑完后给出的结果：**  总共跑了10个事件    发送的每种事件的比例如下：0事件占15%，1事件占10%，2事件占2%依此类推。    出发的每个动作都有记录，如下图所示：    丢弃的：键＝0，指针＝0，轨迹球＝0，翻转＝0。  ##网络统计经过时间为136ms，其中0ms是用在手机上的，136ms用于无线网络上，没有连接的时间为0ms。    从例子中可以看出，该程序在这次测试中没有问题，若程序出现问题终端将打印出异常供程序员查找错误。 | | |
| **工具库地址** | Android的SDK自带monkey，安装SDK后可直接使用。  安装包可去测试工具库取。 | | |

##### 2.2IOSMonkey

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | IOSMonkey | **整理人员** | 唐晓文 |
| **功能**  **简介** | IOSMonkey是一个JS脚本，可以在Instruments工具的Automation模块中运行。通过无规律的频繁操作，实现对正在开发的应用程序进行压力测试。 | | |
| **详细技术说明** | **IOSMonkey运行原理：**  实际上就是使用一个自动化测试脚本，运行该脚本检测应用运行稳定情况。  **工具运行环境：**  IOS开发环境（苹果电脑，安装了Xcode、Instruments），IPhone真机连接电脑。  **IOSMonkey2.js脚本下载地址：**  https://github.com/douban/ynm3k/blob/master/robot4ios/util/iOSMonkey2.js | | |
| **使用**  **说明** | 1、手机连上电脑，在Instruments中选择运行Automation，在Target中选择被测应用。  2、在左下角的Scripts中的Add下拉框中选择“iOSMonkey2.js“脚本。  3、点击左上角的“Run“按钮开始运行，同时监控手机应用的运行情况，是否报错闪退等。  4、脚本运行完成后，可以在日志模块中查看运行情况。  5、猴子测试脚本在每个操作后面都做了截图保存，并且输出了被操作控件所有的父控件，这样可以根据截图和控件之间的关系发现、定位问题。 | | |
| **工具库地址** | 暂无 | | |

### 3、自动化测试工具

#### 3.1MonkeyRunner

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | MonkeyRunner | **整理人员** | 黄万冰 |
| **功能**  **简介** | monkeyrunner工具主要用于功能测试、自动化测试、或用于运行单元测试套件，也可以将其用于其它目的。 | | |
| **详细技术说明** | **工具原理：**  monkeyrunner工具提供了一个API，使用此API写出的程序可以在Android代码之外控制Android设备和模拟器。通过monkeyrunner，您可以写出一个Python程序去安装一个Android应用程序或测试包，运行它，向它发送模拟击键，截取它的用户界面图片，并将截图存储于工作站上。  **MonkeyRunner的测试类型**  1、多设备控制：monkeyrunner API可以跨多个设备或模拟器实施测试套件。您可以在同一时间接上所有的设备或一次启动全部模拟器（或统统一起），依据程序依次连接到每一个，然后运行一个或多个测试。您也可以用程序启动一个配置好的模拟器，运行一个或多个测试，然后关闭模拟器。  2、 功能测试： monkeyrunner可以为一个应用自动贯彻一次功能测试。您提供按键或触摸事件的输入数值，然后观察输出结果的截屏。  3、 回归测试：monkeyrunner可以运行某个应用，并将其结果截屏与既定已知正确的结果截屏相比较，以此测试应用的稳定性。  4、 可扩展的自动化：由于monkeyrunner是一个API工具包，您可以基于Python模块和程序开发一整套系统，以此来控制Android设备。除了使用monkeyrunner API之外，您还可以使用标准的Python os和subprocess模块来调用Android Debug Bridge这样的Android工具。  **运行monkeyrunner**  您可以直接使用一个代码文件运行monkeyrunner，抑或在交互式对话中输入monkeyrunner语句。不论使用哪种方式，您都需要调用SDK目录的tools子目录下的monkeyrunner命令。如果您提供一个文件名作为运行参数，则monkeyrunner将视文件内容为Python程序，并加以运行；否则，它将提供一个交互对话环境。  monkeyrunner的命令语法为：  monkeyrunner -plugin <plugin\_jar> <program\_filename> <program\_options>  **常用API可查看相关的API文档** | | |
| **使用**  **说明** | **详细使用说明（例子）：**  1、在$Android\_Root\tools下新建一个monkeyrunnerprogram1.py文件，内容如下：    2、将手机上的被测应用打开，页面定位在被测页，并将光标定位在第一项上。  3、在$Android\_Root\tools目录下运行一下命令：  monkeyrunner monkeyrunnerprogram1.py   1. 在运行过程中我们可以看见光标不断向下移动，并且可以在当前目录下保存我们自定义的截图：运行前（shotbegin.png），运行后（shotend.png） | | |
| **工具库地址** | Android的SDK自带monkeyrunner。可以直接使用（安装包可到测试工具库获取）  **使用的注意事项：**   1. monkeyrunner如果要用真机进行测试，需要对机器有root权限。 2. monkeyrunner会和电脑中装的豌豆荚，360手机助手等手机连接软件冲突。 3. 在自动化脚本中，尽可能的程序各个模块使用ID的调用，用ID进行的点击等操作不会受限于sleep时间。 4. 在脚本中，可以插入python代码，对monkeyrunner自动化进行优化，从而达到真正自动化目的。如：切图比较结果等。 | | |

#### 3.2Robotium

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | Robotium | **整理人员** | 唐晓文 |
| 功能  简介 | 该工具用于黑盒的自动化测试。可以在有源码或者只有APK的情况下对目标应用进行测试。  Robotimu 提供了模仿用户操作行为的API，比如在某个控件上点击，输入 Text 等等。Robotium 模仿普通用户行为，可以试着把一些原来由测试工程师做的测试变成Robotium自动化实现。  **Robotium 提供下列好处:**   1. 以最小的应用程序知识，开发功能强大的测试案例。 2. 框架支持多个activities自动活动。 3. 最短的时间需求写出测试用例。 4. 测试案例的可读性比标准的仪器测试大大提高。 5. 通过运行时绑定GUI组件使测试用例更强大。 6. 执行测试用例速度快。 7. 顺利整合了Maven或Ant来运行测试，持续集成的一部分。   **Robotium缺点：**   1. 无法捕获Toast。 2. robotium无法操作webview控件。 3. Robotium提供的API是面向过程的，测试代码的可扩展性差。 4. testapp和app运行在两个不同的进程中。 | | |
| **详细技术说明** | Android的测试框架robotium框架，使用的类是Instrumentation，其原理：为某工程新建了一个测试工程，在安装原有工程项目时，也将测试的工程项目安装到AVD（模拟器）上面，同时通过测试的工程项目来与运行时的项目交互，触发其组件的动作等。这种方法的缺点是要求两个项目同时安装到AVD上面。Google采取Remote Control的方式实现自动化测试框架，类似于selenium的Remote Control，即在AVD上运行一个监控程序，而测试项目只需要连接这个监控程序，并发送相应的指令即可与程序进行交互，进行测试。  Android的Instrumentation对某个监控程序进行交互时，其大致采用如下步骤：  1）启动时将项目配置文件AndroidManifest.xml文件中的instrumentation标签中的内容进行初始化，其中标明了所使用的测试运行类，目标项目的包名等  2）执行测试时（可用adb命令触发），将启动目标应用的Activity，同时将待测试ActivityThread作为一个引用进行初始化，如果找不到目标应用则会报错  3）在测试时测试项目的任何对目标项目进行的操作，都会用异步的方式，将消息体放在目标程序的MessageQueue里面，这样目标程序在看到自己的MessageQueue里有内容时，就会执行之。  使用Robotium编写测试脚本需要对Android开发基础知识有一定了解。 | | |
| **使用**  **说明** | 1. 安装包文件重签名 2. 把重签名的文件安装到模拟器或者被测机器上。 3. 新建测试工程编写好测试用例代码。 4. 执行代码，选则在安装好重签名应用的模拟器或者真机上。 5. 查看测试结果。 | | |
| **工具库地址** | 测试工具安装包和案例可以在测试工具库获取 | | |

#### 3.3UI Automation

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | UI Automation | **整理人员** | 唐晓文 |
| **功能**  **简介** | IOS应用的自动化测试工具就是Instrument自带的UI Automation | | |
| **详细技术说明** | **工具运行环境：**  UI Automation是集成在Instruments工具中的一个子工具。而Instruments又是集成在IOS开发工具Xcode中，所以要使用这个必须安装IOS开发环境。  最好使用苹果电脑真机安装，虚拟机上安装IOS操作系统会有很多问题很难解决。  **使用UI Automation开发脚本的前提条件：**  1）系统运行权限限制，所以对程序的签名有一定的要求（必须是开发者签名的测试程序）。  2）应用程序控件识别能力，必须是大部分控件都能识别。（开发程序的时候就要处理UI控件的属性为Accessibility，这样才能识别控件）  3）因为测试脚本要针对控件操作，所以在编写脚本前，必须要了解被测应用界面控件信息。（可以在详细日志区通过查看相关控件的截图来定位控件，获取控件名称等信息）  4）脚本语言使用的是js，因此需要有一定的js语言基础。 | | |
| **使用**  **说明** | **UI Automation开发脚本步骤：**  1、连接手机、打开Instrument，选择Automation，在Target中选择被测应用程序。  2、在左下角的New Script点击“Add“下拉框的”Create“按钮新建脚本。  3、在脚本编辑区域，编写脚本。（获取相关控件，对相关控件做操作）  4、脚本编写完成后，点击”Run”按钮进行回放测试。  关于编写脚本有很多技巧和语言上的问题需要学习解决（例如：模拟滑动效果、输出自动判断的测试结果、添加延迟等等），在此不详细讲述。感兴趣的同学可以看书《IOS测试指南》  **UI Automation录制脚本步骤：**  如果对于写脚本感到很吃力也可以使用Automation的录制脚本功能进行脚本创建。步骤如下：  1、连接手机、打开Instrument，选择Automation，在Target中选择被测应用程序。  2、点击正下方的红色录制按钮，开始录制脚本。  3、录制开始后，在手机上打开应用做一些功能操作。手动操作的同时会自动生成自动化脚本。（录制功能支持真机也支持模拟器录制）  4、录制生成的脚本会显示在脚本编辑区域。  5、脚本录制完成后，根据测试需要再修改加工下脚本、删除一些不必要的多余操作，使其更符合我们的测试目的，回放起来更流畅准确。  6、脚本完成后即可点击左上角的回放功能执行测试脚本了。 | | |
| **工具库地址** | 详细的脚本编写技巧需要在看书学习，推荐看《IOS测试指南》 | | |

### **4、云测试平台**

|  |  |  |  |
| --- | --- | --- | --- |
| **工具** | ITestin | **整理人员** | 黄万冰 |
| **功能**  **简介** | Testin是免费服务移动App开发者的真机自动化云测试客户端工具。基于真实的智能终端设备录制一个测试脚本然后运行，并输出运行结果。覆盖 Android和iOS两大设备平台，支持Pad/Phone/Smart TV等智能终端设备。支持功能测试、UI测试、性能测试、稳定性测试、压力测试，返回包括日志和截图的详细测试报告，可结合Testin云测平台300多 款3000多部终端批量验证功能点，帮助开发者快速找到错误原因，提升应用质量，并大幅度降低测试成本，缩减测试周期。 | | |
| **详细技术说明** | **i**[**Testin**](http://www.testin.cn)**基本功能和使用步骤：**  （iTestin是Testin平台的一个用于录制回放脚本、上传脚本的客户端，需要安装）  1.用户通过客户端可将手机上的操作录制成脚本，录制过程中客户端提供多种截图方式供用户进行记录。 1.jpg  2.录制的脚本可以在相同分辨率的不同手机下重复执行成百上千次的操作。 3.录制的脚本可以上传至Testin平台进行适配测试。 4.录制的脚本可以提交到社群测试平台（<http://m.testin.cn/> ）作为稿件提交到平台，获取佣金。  **Testin平台的特点和功能：**  #1 真机自动化测试  可提供业界最全的自动化测试类型：兼容测试、性能测试、功能测试、压力测试。返回测试结果，包括：CPU消耗、内存消耗、启动时间，界面截图、Log日志、错误原因及分析等。  #2 支持两大智能手机平台  全球首款同时支持Android和iOS设备的黑盒测试的自动化客户端工具，结合Testin云测试平台300多款1000多部终端为开发者提供基于批量终端的脚本录制回放以及测试记录的功能。  #3 快速交付测试结果  测试结果自动记录，自动输出，支持在终端设备上记录操作行为、生成脚本文件，在终端设备上反复回放并记录回放结果，测试团队Check测试结论只需使用iTestin独有的专利技术回放测试期录制的脚本及截图及日志信息。  #4 专业测试报告  支持在终端设备上记录操作行为、生成脚本文件，在终端设备上反复回放并记录回放结果：错误屏幕截图、Log日志，为开发者提供最专业、最详细的测试报告，帮助开发者快速找到错误原因。  **目前Testin平台对我们的主要用途：**   1. 测试应用在各款不同机器上的安装、启动、卸载兼容性测试。 2. 测试应用的能耗（CPU、内存、启动时间） 3. 做简短的MonkeyTest（测试在不同机器上运行是否会崩溃） | | |
| **使用**  **说明** | **创建测试简单三步走起：1.创建测试  2. 管理测试任务 3.查看测试报告**  arrow_right.png**创建测试** 2bdec7521c263b1b43a75b4a.jpg  1.1 首先选择:创建测试==>点击"云测试"，根据你的测试需求可选择兼容测试、性能测试、功能测试。  **未标题-1.jpg**  兼容测试等.jpg    1.2 上传需要测试的App(支持：apk/ipa格式)，支持2种上传方式：（上传App/App链接下载)。  上传APP.jpg  1.3 选择免费测试终端：上传成功的App、版本名称可以；[APP测试](http://www.testin.cn)需要账号登陆，需填写账号和密码；选择需要测试的终端，可以根据操作系统、品牌、屏幕分辨率选择，确认提交测试。（认证权限不同，使用的测试服务等级也不一样，企业认证用户可以使用终端数为：200+......） 详情可以查看：[**http://sso.testin.cn/uc.action?op=Profile.comCertificate**](http://sso.testin.cn/uc.action?op=Profile.comCertificate)  **app提交3.jpg**  **终端5.jpg**  **终端6.jpg**  **提交成功1.jpg**  **云测.jpg  开始真机**[**自动化测试**](http://www.testin.cn)**(后台运行)**  arrow_right.png**管理测试任务**  **2bdec7521c263b1b43a75b4a.jpg**  进入个人“管理中心”，查看测试。 任务的执行情况，可进行中止测试，查看报告等管理，也可以点击“创建测试” 进入创建测试流程。  管理中心xin.jpg  arrow_right.png**查看测试报告**  **2bdec7521c263b1b43a75b4a.jpg**  进入“管理中心”，在测试列表页，点击要查看的App测试报告，进入测试报告页面。测试报告分为5部份，包含重点报告、 综合性能报告、 终端详情 、屏幕截图 、应用详情。  **3.1测试报告--重点报告** 重点报告：包含终端兼容适配率，错误定位（认证用户可查看）通过、待优化、未通过的全部终端。点击单款终端可以查看测试结果报告（启动时间 、CPU占有率 、内存使用 、 流量耗用等）  **错误定位.jpg**  **报告1.jpg**  **详情1xin.jpg**  **详情2xin.jpg**  **详情3xin.jpg**  **详情4xin.jpg**  **详情5xin.jpg**  **3.2测试报告-- 综合性能报告**  综合性能报告：性能概况 ：启动耗时 CPU占用 内存占用 均值、终端数、占比、测试高峰的手机型号。  **报告2xin.jpg**  **3.3 测试报告-- 终端详情**  终端详情：可查看单款终端安装、卸载是否通过情况；错误日志、问题日志、调试日志等，可以查看、下载详细报告，企业服务可查看、下载（安装日志 运行日志 卸载日志）  **终端详情xin.jpg**  **3.4 测试报告-- 屏幕截图**  屏幕截图：展现测试过程的截图，支持在线浏览、下载：Excel报告、 PDF报告（企业服务）。 屏幕截图xin.jpg  **3.5测试报告-- 应用详情**  应用详情：包含应用程序名称、文件大小 、平均流量耗用、  版本号、适用系统 、启动路径 。 应用详情xin.jpg | | |
| **工具库地址** | 工具安装包可在测试工具库获取。 | | |