

APP自动化测试规范

质量中心

文档修订记录

（状态：C―创建，A—增加，M—修改，D—删除，R—发布）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **版本** | **状态** | **简要说明（内容和范围）** | **更改人** | **更改日期** | **审批人** | **审批日期** |
| V1.0 | C | APP自动化规范 | 张文俊 | 2020-2-6 | 樊兆妮 |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

目录

[1 、概述 4](#_Toc189992337)

[1.1 文档目的 4](#_Toc2040508517)

[1.2 文档范围 4](#_Toc1660286276)

[2 、APP自动化测试 4](#_Toc28931614)

[2.1 测试需求 4](#_Toc922332276)

[2.2 测试方式 4](#_Toc1101598686)

[2.3 测试工具 4](#_Toc1112594815)

[2.4 脚本编写规范 5](#_Toc1240941276)

[2.4.1 基础文件、基础函数和case分开保存 5](#_Toc138846068)

[2.4.2 case命名（根据用例设计来规划） 5](#_Toc1418624234)

[2.4.3 函数使用方式 5](#_Toc1454051844)

[2.4.4 基础函数封装注释 6](#_Toc2032922895)

[2.4.5 变量、函数、类名、模块名等命名方式 6](#_Toc870272495)

[2.4.6 case注释 7](#_Toc158703748)

# 、概述

## 文档目的

本文主要是为了统一规范APP自动化测试脚本，提高脚本可读性，降低脚本维护成本，同时也为测试人员提供APP自动化测试方法编写参考。

## 文档范围

本规范适用于测试人员。

# 、APP自动化测试

## 测试需求

随着公司的转型，APP软件逐渐增多，迭代的频繁以及各种机型的兼容给测试人员带来大量的重复劳动工作，因此引入相应技术实现APP自动化。

APP自动化测试与普通的web自动化类似，都需要进行UI元素的定位，然而由于市面上工具也繁多，根据入手难度以及学习成本而言，我们采用网易的AirTest Project作为质量中心APP自动化测试工具。

为了更好的进行测试，以及后期维护成本，因此制订相应规范，见后续内容。

## 测试方式

根据迭代需求，以及成品APP软件进行相应脚本开发，主要兼顾主流程全覆盖+次流程主要功能点，运行时着重考虑兼容性。

每个迭代测试前以及上线前通过测试环境进行最终验证，预发布环境以及产线环境根据实际情况进行验证

## 测试工具

Python3+AirTest+Poco

## 脚本编写规范

### 基础文件、基础函数和case分开保存

根据实际项目搭建相应测试框架（搭建需要进行组内评审，待评审通过后方可执行后面脚本设计）

### case命名（根据用例设计来规划）

如：test\_0\*\*\*\_bus

第一位数：代表模块，商机

第二位数：代表子模块，自开发商机

第三位数：代表子-子模块，依次类推

字符串：代表用例名称，名称小写字母，单词之间用\_分割

如：test\_0001\_bus\_add\_normal（商机新增模块-正常新增商机）

### 函数使用方式

#### AirTest：

主要用于图形定位，由于图形定位可能存在失败的情况，仅在Poco无法进行定位时再 使用

简单用法如下：

1、初始化：auto\_setup(basedir=None, devices=None, logdir=None, project\_root=None, compress=0)

2、连接设备：connect\_device(uri)；set\_current(idx)

3、获取设备：device()

4、执行ADB的shell命令：shell(cmd)

5、操作应用：start\_app(package, activity=None)；stop\_app(package)；clear\_app(package)

6、应用安装与卸载：install(filepath, \*\*kwargs)；uninstall(package)

7、截图到文件：snapshot(filename=None, msg="", quality=ST.SNAPSHOT\_QUALITY)

8、唤醒解锁回到首页：wake()；home()

9、单双机：touch(v, times=1, \*\*kwargs)；double\_click(v) ；v表示模板实例Template(...) 或者绝对坐标(x, y)

10、滑动：swipe(v1, v2=None, vector=None, \*\*kwargs) ；v1起始位置，v2结束位置， vector

11、文本输入：text(text, enter=True, \*\*kwargs)

12、等待睡眠：wait(v, timeout=None, interval=0.5, intervalfunc=None)；sleep(secs=1.0) ； timeout默认20s

13、判断是否存在：exists(v)；找到返回坐标pos，没有找到返回false

14、查询模板实例坐标：find\_all(v)

15、断言：assert\_exists(v, msg="")；assert\_not\_exists(v, msg="")；assert\_equal(first, second, msg="")；assert\_not\_equal(first, second, msg="")

#### Poco：

根据界面元素定位，调用的是原生ADB命令，如果出现无法定位情况再使用AirTest

简单用法：

poco=AndroidUiautomationPoco(...) #申明Poco子类对象实例

poco(“表达式”).操作 #选取元素并执行某个操作

表达式：

name表达式： poco('com.tencent.mobileqq:id/password')

普通关键字表达式 ：poco(desc="密码 安全")

正则表达式： poco(descMatches='^密码.\*$')

……更多表达式编写方式可以参考源码Uiautomation相关类

#### Python3：

（略）

### 基础函数封装注释

在每个函数前，增加必要的注释。如：

’’’  
 [Name](https://my.oschina.net/appl)：函数名称  
 Content：函数具体说明，包括如何使用，入参，反参等  
 ’’’

### 变量、函数、类名、模块名等命名方式

命名规则可以被视为一种惯例，并无绝对与强制

目的是为了增加代码的识别和可读性

1、模块名、包名:   
 小写字母，单词之间用\_分割 ，如ad\_stats.py

2、类名:   
 单词首字母大写 ，如AdStats 、ConfigUtil

3、全局变量名(类变量，在java中相当于static变量):   
 大写字母，单词之间用\_分割 ，如NUMBER 、COLOR\_WRITE

4、普通变量:   
 小写字母，单词之间用\_分割 ，如this\_is\_a\_var

5、私有变量:   
 以\_\_开头(2个下划线)，其他和普通变量一样 ，如\_\_private\_var

6、普通函数:   
 和普通变量一样:，如get\_name() 、count\_number()

7、私有函数:   
 以\_\_开头(2个下划线)，其他和普通函数一样 ，如\_\_get\_name()

### case注释

在每条case前，增加必要的case注释。如：

’’’  
 [Name](https://my.oschina.net/appl)：用例编号+名称  
 Content：用例简单说明，前置条件，预期结果  
 ’’’