SDK自动化测试实用技术分享

姓　名：唐继玉
部　门：360　QA中心
联　系：tangjiyu1@360.cn

目录

一、简述 SDK测试

- SDK：软件开发工具包

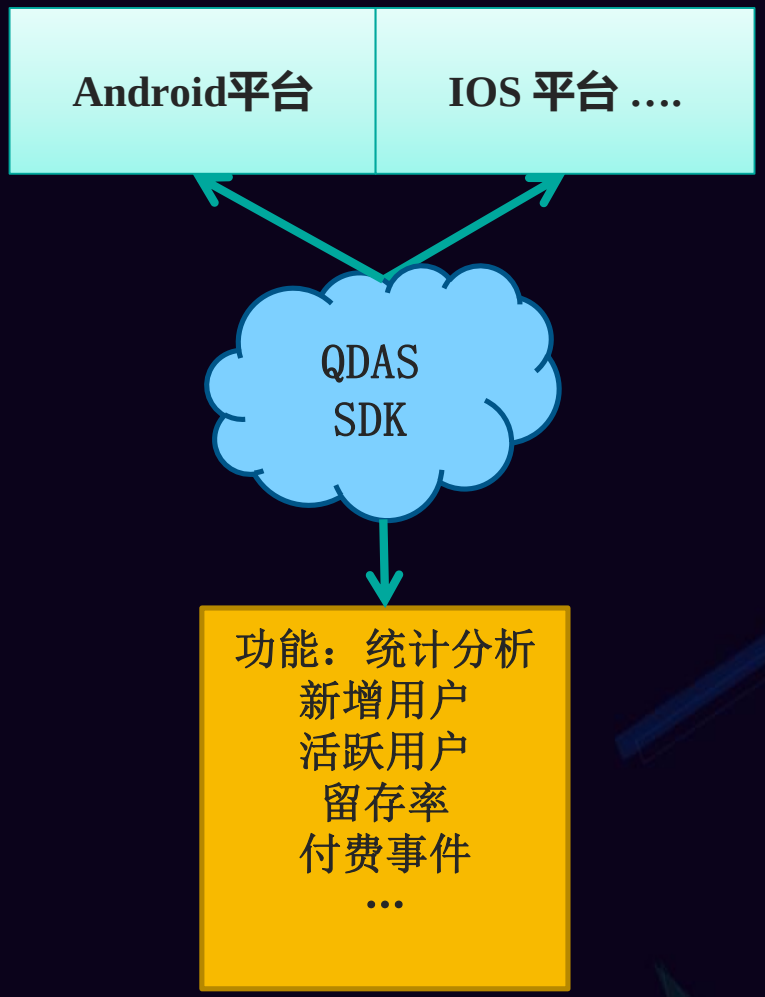
  （Software　Development Kit）

- 多平台覆盖

  （Android、IOS、windows PC端、 web）

- QDAS SDK

  功能：统计分析

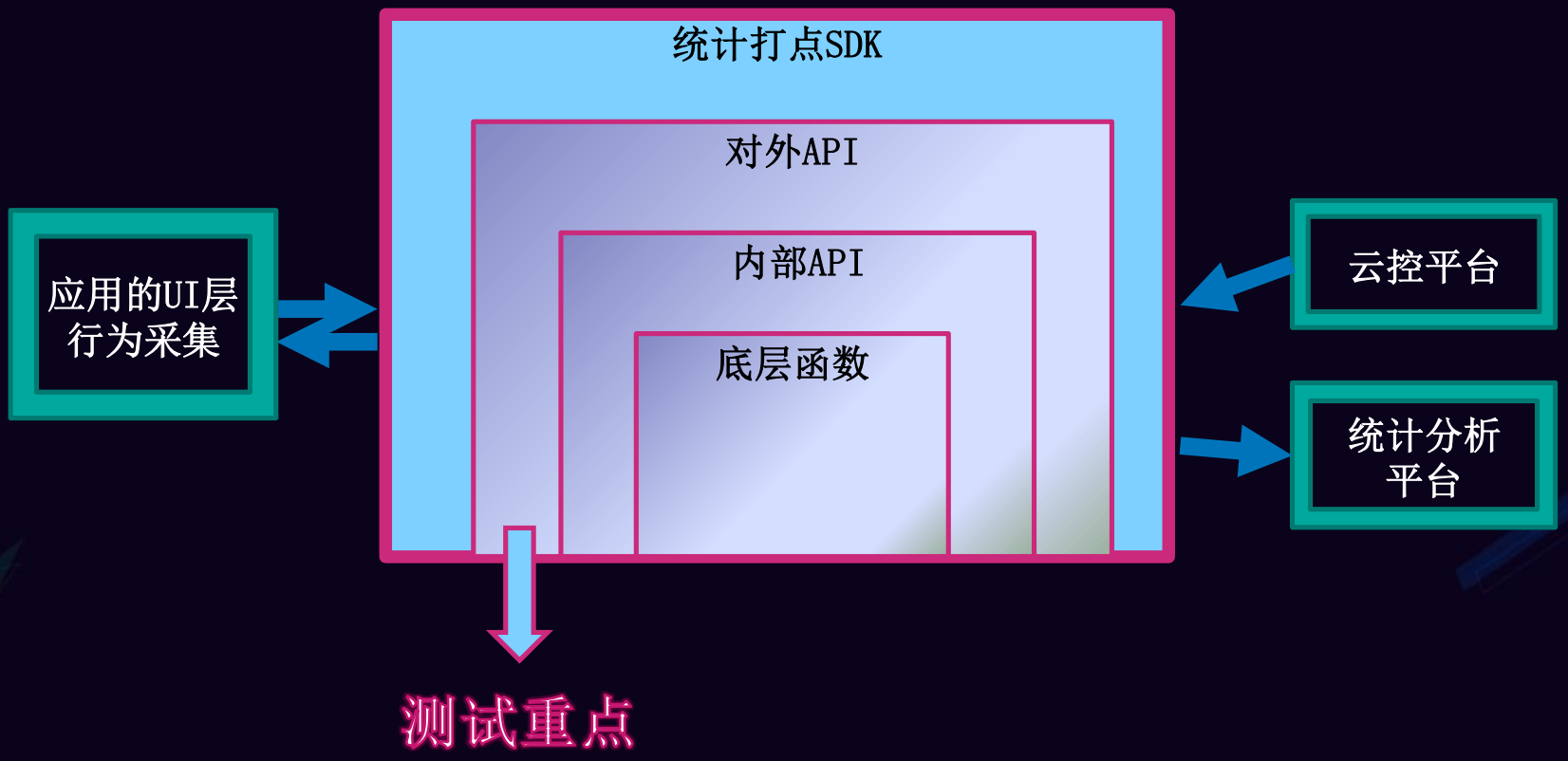


一、简述 SDK测试

- SDK 发布文件组成
  - SDK 文件（jar, js, arr, so, ipa等）
  - 集成SDK的示例工程
  - 集成文档（集成方法说明，接口文档）

一、简述 SDK测试

- SDK 逻辑架构



二、Android端实用技术之一

- 接口集成方法
  - 单次调用(按键、广播、service、provider等)
  - 多次并发调用(循环调用、多线程并发调用)
  - 跨进程调用（如 service、AIDL调用）

二、Android端实用技术之一

- 接口集成方法
  - 单次主动调用---按键

```
public class MainActivity extends AppCompatActivity {
    private TextView mTvInfo;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        findViewById(R.id.btnUILoadOAID).setOnClickListener((v) -> {
                mTvInfo.setText("");
                QosSdk.getDeviceIdsAsync( context: MainActivity.this, EnumSet.of(IdFeature.OAID),
                        new DeviceIdCallback() {
                    @Override
                    public void onValue(final DeviceIdInfo deviceIdInfo) {
                        runOnUiThread(() -> {
                                mTvInfo.setText("oaid:" + deviceIdInfo.getOAID());
                        });
                    }
                });
        });
```

调用API

二、Android端实用技术之一

- 接口集成方法

  - 单次主动调用---广播

```
<receiver
    android:name=".MyBroadcastReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter >
        <action android:name="AdbOnException"/>
        <action android:name="AdbOnEventL5"/>
    </intent-filter>
</receiver>
```

```
public class MyBroadcastReceiver extends BroadcastReceiver {

    private String AdbCmd = "";

    @Override
    public void onReceive(Context context, Intent intent) {
        AdbCmd = intent.getAction();
        Toast.makeText(context, AdbCmd, Toast.LENGTH_SHORT).show();
        adb shell am broadcast -a AdbOnException -e exception exceptionString
        if (AdbCmd.equals("AdbOnException")) {
            String exString = intent.getStringExtra( name: "exception");
            QHStatAgent.onError(context, exString);
        }
//          adb shell am broadcast -a AdbOnEventL5
        if (AdbCmd.equals("AdbOnEventL5")) {
            QHStatAgent.onEvent(context, s: "evntid", s1: "AdbOnEventL5", i: 123,
                    QHStatAgent.DataUploadLevel.L5, QHStatAgent.SamplingPlan.A);
        }
```

注意：发送广播的action尽量保持唯一性，以免有不同的APP都收到同 一个广播而对预期造成影响！

二、Android端实用技术之一

➢接口集成方法

- 被动触发调用

✓ 在activity生命周期中调用SDK接口

✓ 监听系统广播，如网络变化，声音改变

✓ ...

```
protected void onResume() {
    super.onResume();
    QHStatAgent.onResume( context: this);
    Log.d(TAG, msg: "调用onResume");
    Log.d(TAG, msg: "M1:" + getM1( context: this));
    MyApplication.myABTestList.add(listener);

}

@Override
protected void onPause() {
    super.onPause();
    Log.d(TAG, msg: "调用onPause");
    QHStatAgent.onPause( context: this);
    MyApplication.myABTestList.remove(listener);
}

@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    Log.d(TAG, msg: "调用onNewIntent");
    QHStatAgent.onNewIntent(intent);
}
```

二、Android端实用技术之一

- 接口集成方法

  - 多次并发调用（循环、多线程/多进程并发调用；压力测试）

```
<service
    android:name=".serviceForTestMultiprocess"
    android:enabled="true"
    android:process=":myserver"
    android:exported="false"></service>
```

```
public class serviceForTestMultiprocess extends Service {
    public static final HashMap<String, String> MAP;
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d( tag: "ABTest_QA",  msg: "后台进程0：开启多线程调用onevent打点");
        for (int i = 0; i < 2; i++) {
            OnEventRunnable R1 = new OnEventRunnable( context: this,  name: "Thread-" + i);
            R1.start();}
        return super.onStartCommand(intent, flags, startId);
    }
    class OnEventRunnable implements Runnable {
        ...
        public void run() {
            String key = "multiService-" + threadName;
            QHStatAgent.onEvent(rContext, key, MAP,  i: 1,
                    QHStatAgent.DataUploadLevel.L9, QHStatAgent.SamplingPlan.A);
        }
    }
}
```

二、Android端实用技术之二

➢复杂参数组合初始化

1、初始化前

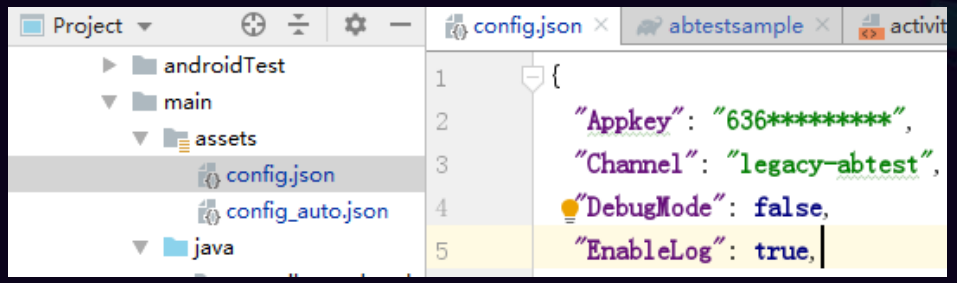
接口调用或者不调用?

接口调用参数多种方式?

不同渠道包集成方式不同?

2、初始化后

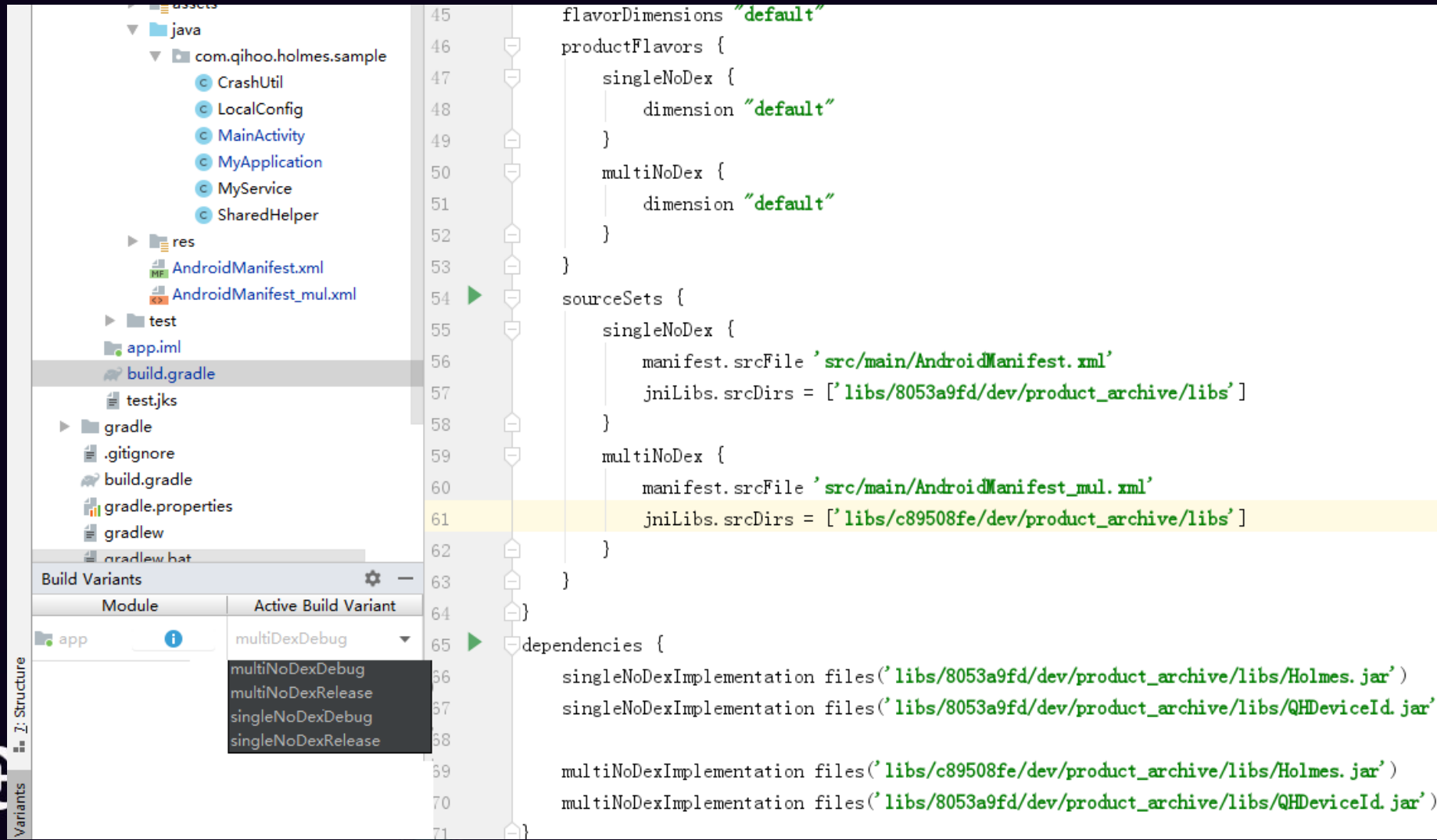
可手动触发接口调用




```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        if (!sharedHelper.readString( key: "Appkey").equals("")) {
            configJson = sharedHelper.read();
            Log.d(TAG, msg: "SP读取配置: " + sharedHelper.read());
        } else {
            configJson = sharedHelper.getJsonFromFile( context: this, fileName: "config.json");
            Log.d(TAG, msg: "assets/config.json中读取配置: " + configJson);
            sharedHelper.save(configJson);
        }
        try {
            QHConfig.setAppkey( context: this, configJson.optString( name: "Appkey"));
            QHStatAgent.setChannel( context: this, configJson.optString( name: "Channel"));
            QHStatAgent.setLoggingEnabled(configJson.getBoolean( name: "EnableLog"));
            QHConfig.setManualMode( context: this, configJson.getBoolean( name: "ManualMode"));
        } catch (JSONException e) {}
        QHStatAgent.init( context: this);
        QHStatAgent.setListener(abTestListener); //初始化之后设置
    }
```

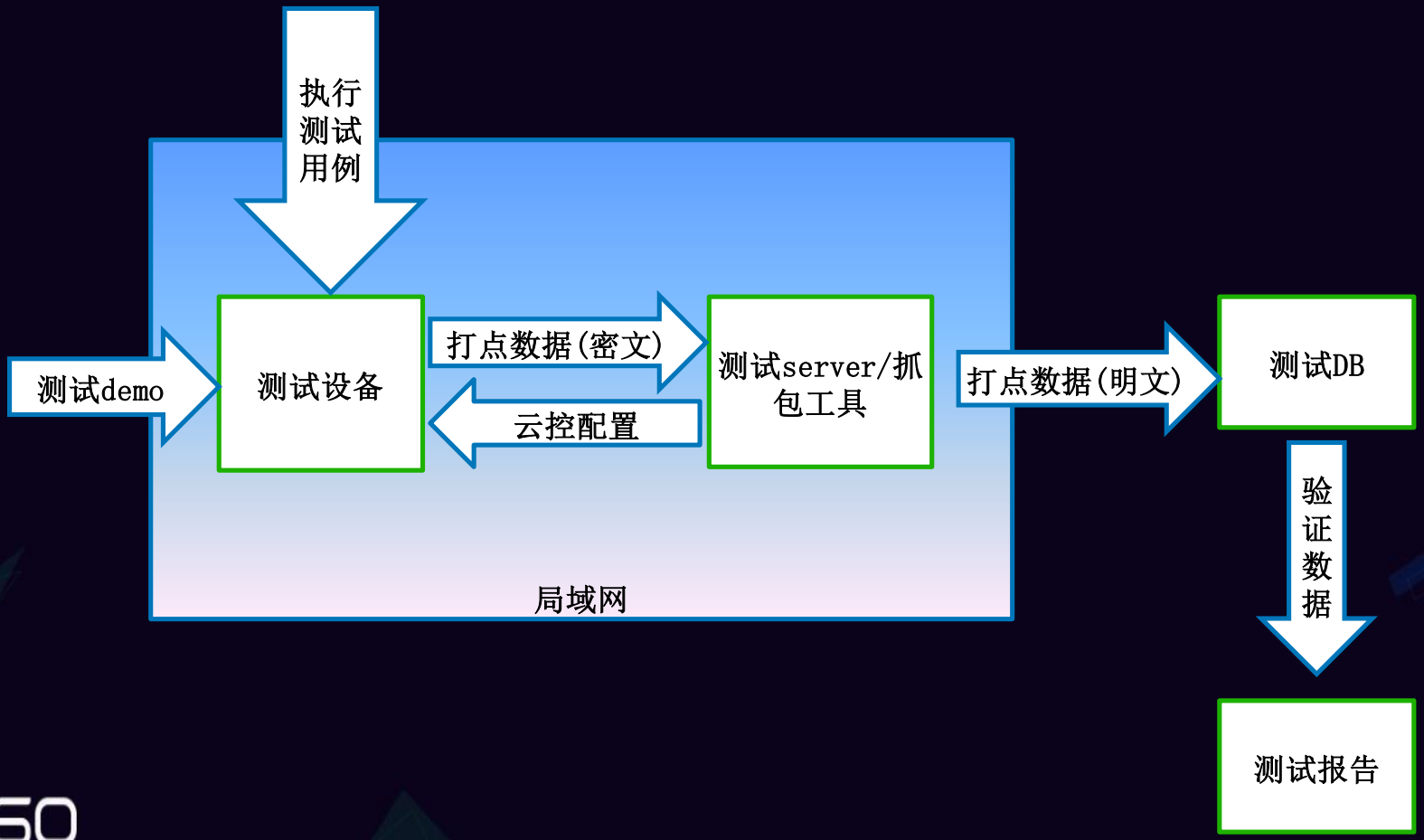
二、Android端实用技术之二

➢多渠道打包



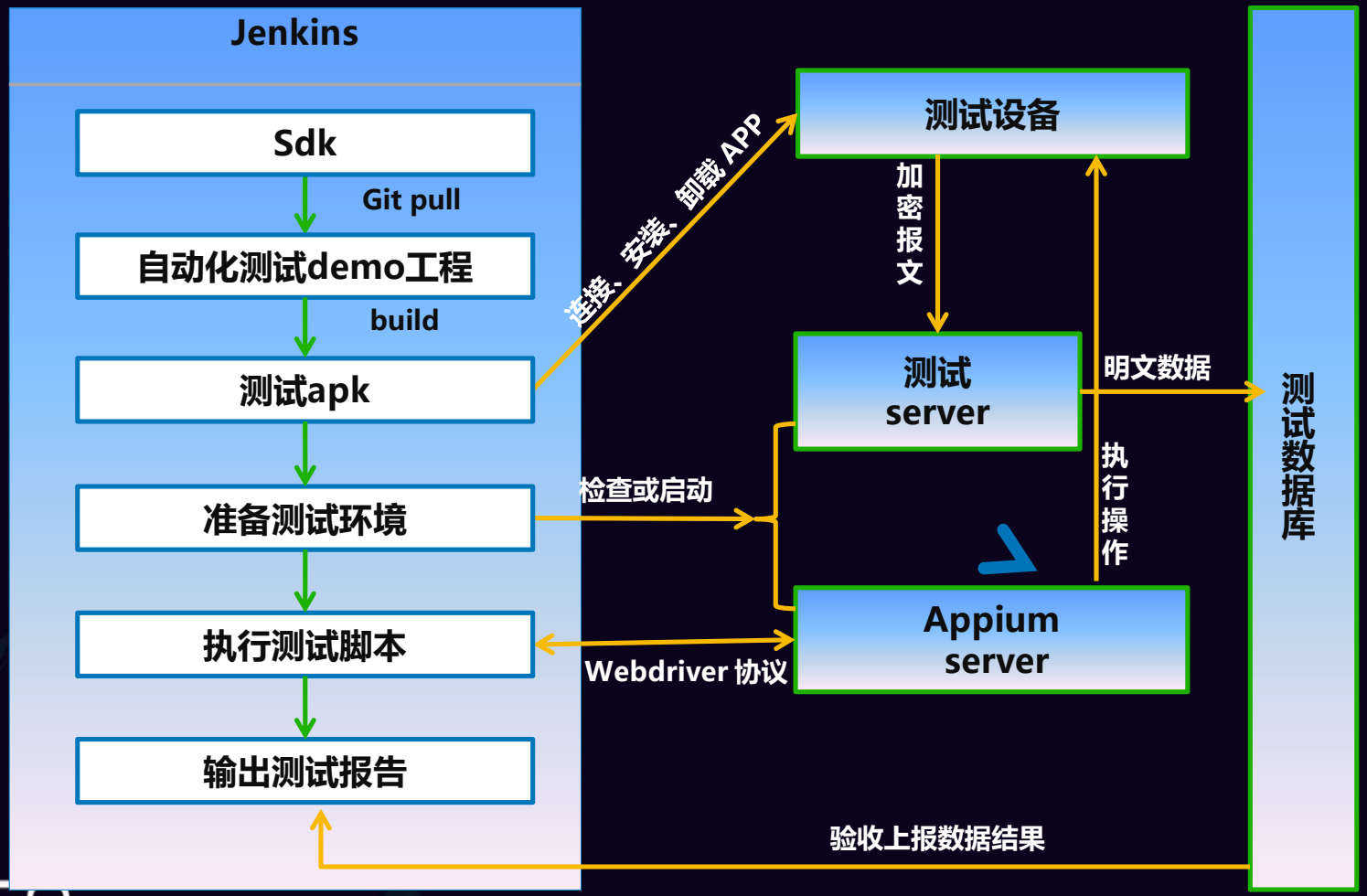
二、Android端实用技术之三

- 打点SDK 测试环境搭建以及验证



二、Android端实用技术之四

- 打点SDK自动测试方案和持续集成



二、Android端实用技术之四

其他的专项测试

SDK接口压力测试：多线程/进程并发调用

稳定性测试： 随机的接口测试（monkey，广播指令bat）

兼容性测试： 安卓的版本、品牌型号、真机以及模拟器

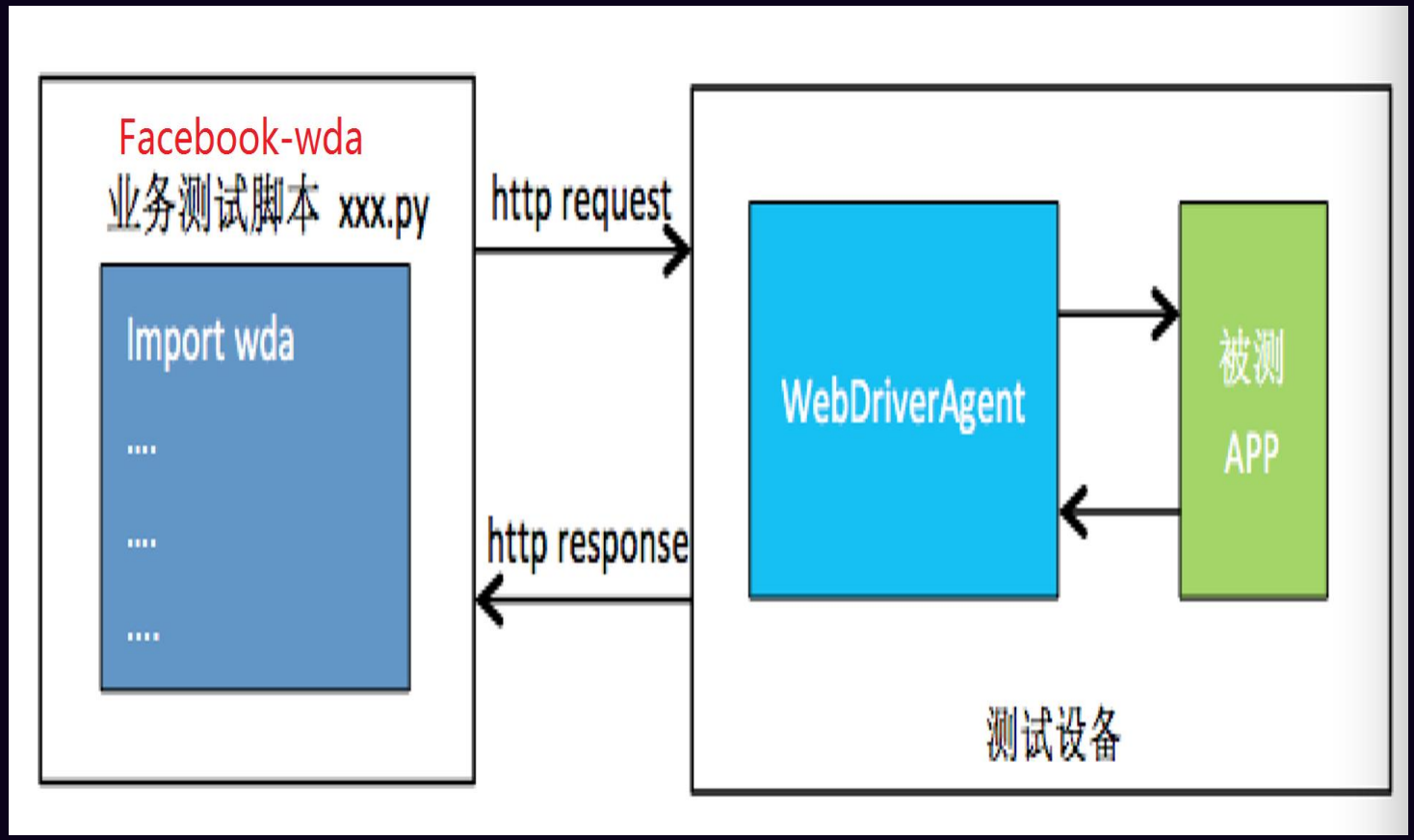
性能测试： CPU、内存、启动时间等等

隐私弹窗测试： 读取电话、短信、应用列表等权限

安全性测试：关闭日志、代码扫描、反编译查看关键API是否暴露

三、iOS端sdk自动化方案

- 自动化测试方案示意图



三、iOS端sdk自动化方案

➢测试用例脚本

- 安装和引入开源python库facebook-wda
  (https://github.com/openatx/facebook-wda)
  - 直接调用接口，通过构造HTTP请求直接跟WebDriverAgent通信
  - 接口丰富，易用，可定制修改

Create a client

```
import wda

# Enable debug will see http Request and Response
# wda.DEBUG = True
c = wda.Client('http://localhost:8100')
```

Client

```
# Press home button
c.home()
c.locked() # true of false
c.lock() # lock screen
c.unlock() # unlock
```

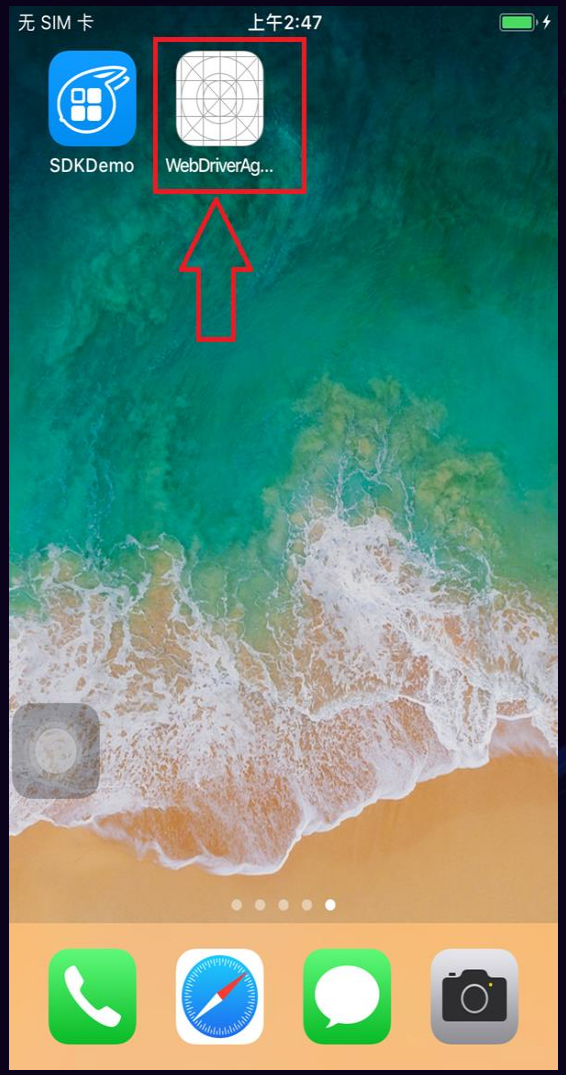
Session

```
s = c.session('com.apple.Health')
print(s.orientation)
s.tap(200, 200)
s.click(0.5, 200) # click center of x, and y(200)
s.double_tap(200, 200)
s.swipe(0.5, 0.5, 0.5, 1.0)  # swipe middle to bottom
```

三、iOS端sdk自动化方案

➢手机自动控制

- 使用开源的WebDriverAgent（https://github.com/facebookarchive/WebDriverAgent）
  - Facebook在 iOS 端实现的一个 WebDriver server，借助这个 server 我们可以远程控制 iOS 设备。
  - 通过调用私有api接口，可高效灵活操作手机



三、iOS端sdk自动化方案

主要优势：

- 使用facebook-wda，功能强大，短小精悍，编写测试用例方便快捷
- 与Android端自动化测试共用数据上报server及结果数据验证代码
- 均开源，可对原生的facebook-wda和WDA接口改进，定制操作，灵活稳定

三、iOS端sdk自动化方案

facebook-wda和webDriverAgent定制接口概览

操作	facebook-wda改进	WebDriverAgent改进	效果
点击/tap	不变	使用XCEventGenerator的私有接口tapAtTouchLocations实现	点击速度更快，接口可控制的参数更多
滑屏/swipe	原有scroll接口不可用，增加swipe接口	使用XCEventGenerator的私有接口pressAtPoint实现	滑动顺畅，可通过参数调节滑动速度
长按/long tap	不变	使用XCEventGenerator的私有接口pressAtPoint实现	可用，可通过参数设置长按时间
单击Home键	不变	注释掉同步代码	在ios11上不可用的问题解决

三、iOS端sdk自动化方案

swipe操作定制化修改：

```
s.swipe(x1, y1, x2, y2)
```

__init__.py文件中：

```
def swipe(self, x1, y1, x2, y2, duration=0):
    data = dict(fromX=x1, fromY=y1, toX=x2, toY=y2, duration=duration)
    return self.http.post('/wda/dragfromtoforduration3', data=data)
```

FBElementCommands.m文件中：

```
[[FBRoute POST:@"/wda/dragfromtoforduration3"] respondWithTarget:self
action:@selector(scrollFrom3:)]
```

```
+ (id<FBResponsePayload>)scrollFrom3:(FBRouteRequest *)request{
  CGPoint startPoint = CGPointMake((CGFloat)[request.arguments[@"fromX"] doubleValue], (CGFloat)[request.arguments[@
  CGPoint endPoint = CGPointMake((CGFloat)[request.arguments[@"toX"] doubleValue], (CGFloat)[request.arguments[@"toY
  double duration = [request.arguments[@"duration"] doubleValue];
  XCEventGenerator *t = [[XCEventGenerator alloc] init];

  [t pressAtPoint:startPoint forDuration:(double)0 liftAtPoint:endPoint velocity:duration orientation:0 name:@"drag'
      NSError *commandError) {}];
  return FBResponseWithOK();
}
```

谢谢