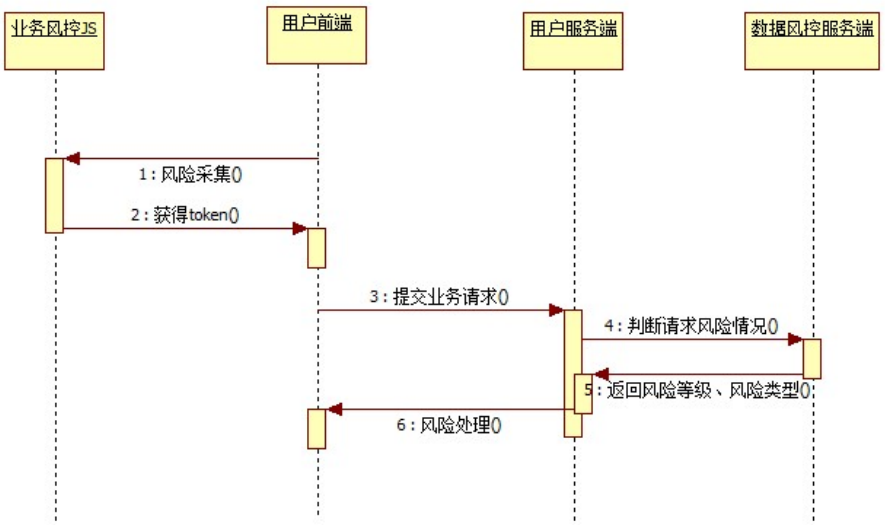
数据风控

用户指南

用户指南

业务风险防控，包含：注册防控、登陆防控、活动防控、消息防控和其他风险防控，通过用户行为、软硬件环境信息、设备指纹、业务基础信息综合判定用户请求的风险程度：
注册防控：在注册场景提供安全防护，防止机器注册、人工恶意注册、注册短信被攻击。
登陆防控：在登陆场景提供安全防护，防止刷库撞库、暴力破解、可疑登陆。
活动防控：在活动场景提供安全防护，防止刷红包、抢优惠券、黄牛抢号、黄牛刷单。
消息防控：在发帖、评论场景提供安全防护，防止批量发帖、垃圾评论。
其他风险防控：其他场景下的风险防控，防止如频繁查机票、机器点赞、批量送礼物、虚假投票等。

系统间交互流程



1.风险采集：采集用户行为、软硬件环境信息、设备指纹信息。此步骤由JS脚本自动完成，无需用户处理；
2.提交业务请求：用户提交业务请求如注册、登陆请求时，需要将前端风控参数传递给服务端，风控参数包含：

- token：请求唯一id
- scene：场景

3.判断请求风险情况：用户服务端调用数据风控服务api，获得请求风险结果。该步骤需要放在业务请求处理之前。
4.风险处理：根据返回的风险结果，进行风险处理。对于不同风险结果，建议处理方案：

- 无风险：继续做业务请求处理，如注册、登陆处理；
- 中风险：建议在做业务请求处理之前，让操作者进行一定的验证，如验证码、语音、短信等验证；

- 高风险：建议直接返回业务请求失败，让操作者重新再来一次。

接入流程

1.第一步，进入数据风控控制台：

- 未开通服务，开通服务；
- 已开通服务，进入服务管理，选择需要的业务风控服务，点击【创建】：

2.第二步，选择应用类型，输入高峰期PV，点击【下一步】：

3.第三步，根据系统集成代码的操作步骤，将示例代码拷贝出，集成到业务系统中，详细操作参见下方：系统集成。集成完成，点击【下一步】：

4.第四步，系统集成完毕，发布对应的服务：

系统集成

前端页面引入

1. 找到需要使用风险识别服务的页面，将示例代码复制进去，注意将form表单替换成自己的业务表单；
2. 运行环境：Tomcat、Apache等；
3. 接入成功校验：打开浏览器控制台（F12开发者工具），在页面上移动鼠标、点击键盘：

- a. 控制台收到analyze.jsonp请求(多个)；
- b. 在Headers - Query String Paraments 里看到：n、a、t、asyn、scene等参数；
- c. 双击analyze.jsonp请求，获得onJSONPCallback({“result”:0,” success”:true})。

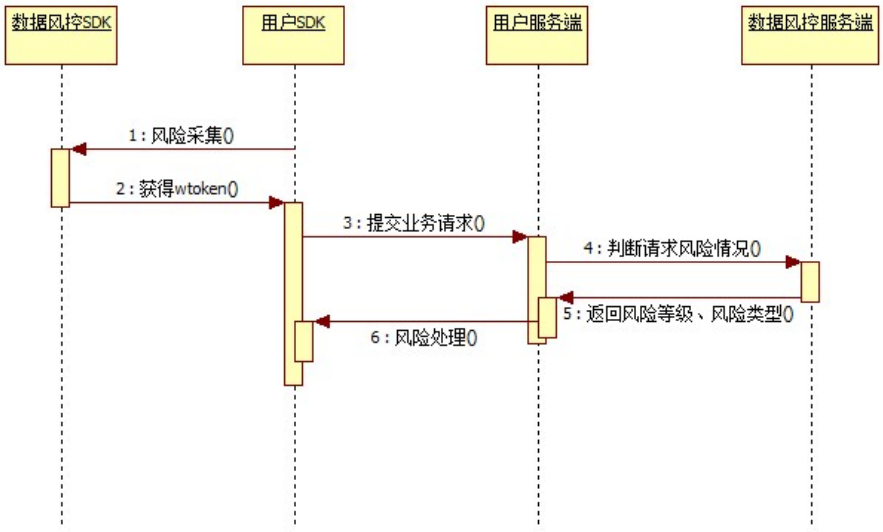
服务端API调用

1. 下载对应语言的SDK，SDK下载请进入数据风控控制台；
2. 将SDK加载到工程中；
3. 参考示例代码，开发第一步页面请求的处理类（如java的Action,Controller,Servlet等）;注意填入自己的阿里云accesskey和secret；
4. 参考示例代码，对运行结果进行处理；请注意对服务端出现的错误进行兼容处理。
5. 业务风险防控API列表：

API	描述
SpamRegisterPrevention	注册防控接口
LoginPrevention	登陆防控接口
CampaignPrevention	活动防控接口
BbsPrevention	消息防控接口
OtherPrevention	其他风险防控接口

业务风控，包含：注册防控、登陆防控、活动防控、消息防控和其他风险防控，通过用户行为、软硬件环境信息、设备指纹、业务基础信息综合判定用户请求的风险程度：
注册防控：在注册场景提供安全防护，防止机器注册、人工恶意注册、注册短信被攻击。
登陆防控：在登陆场景提供安全防护，防止刷库撞库、暴力破解、可疑登陆。
活动防控：在活动场景提供安全防护，防止刷红包、抢优惠券、黄牛抢号、黄牛刷单。
消息防控：在发帖、评论场景提供安全防护，防止批量发帖、垃圾评论。
其他风险防控：其他场景下的风险防控，防止如频繁查机票、机器点赞、批量送礼物、虚假投票等。

系统间交互流程



1.风险采集：采集用户行为、软硬件环境信息、设备指纹信息。用户在提交业务请求之前，需要调用数据风控sdk接口，完成风险采集；
2.提交业务请求：用户提交业务请求如注册、登陆请求时，需要将sdk端风控参数传递给服务端，风控参数包含：

- wtoken：风险采集id

3.判断请求风险情况：用户服务端调用数据风控服务api，获得请求风险结果。该步骤需要放在业务请求处理之前。
4.风险处理：根据返回的风险结果，进行风险处理。对于不同风险结果，建议处理方案：

- 无风险：继续做业务请求处理，如注册、登陆处理；
- 中风险：建议在做业务请求处理之前，让操作者进行一定的验证，如验证码、语音、短信等验证；
- 高风险：建议直接返回业务请求失败，让操作者重新再来一次。

接入流程

1.第一步，进入数据风控控制台：

- 未开通服务，开通服务；
- 已开通服务，进入服务管理，选择需要的业务风控服务，点击【创建】：

2.第二步，选择应用类型，输入高峰期PV，点击【下一步】：

3.第三步，根据系统集成代码的操作步骤，下载SDK，集成到APP中；服务端调用对应api，详细操作参见下方：系统集成。集成完成，点击【下一步】：

4.第四步，系统集成完毕，发布对应的服务：

系统集成

Android SDK集成配置

1.生成SDK

出于对应用数据安全考虑，数据风控生成的SDK会与应用强绑定。如果此前使用Debug版应用 生成SDK需要在应用发布前使用Release版应用重新上传，并替换原有SDK。

- 上传Release版APK；
- 点击“生成SDK”按钮，生成SDK；
- 生成完SDK后，SDK自动下载到本地；

2.导入SDK

2.1 导入前准备

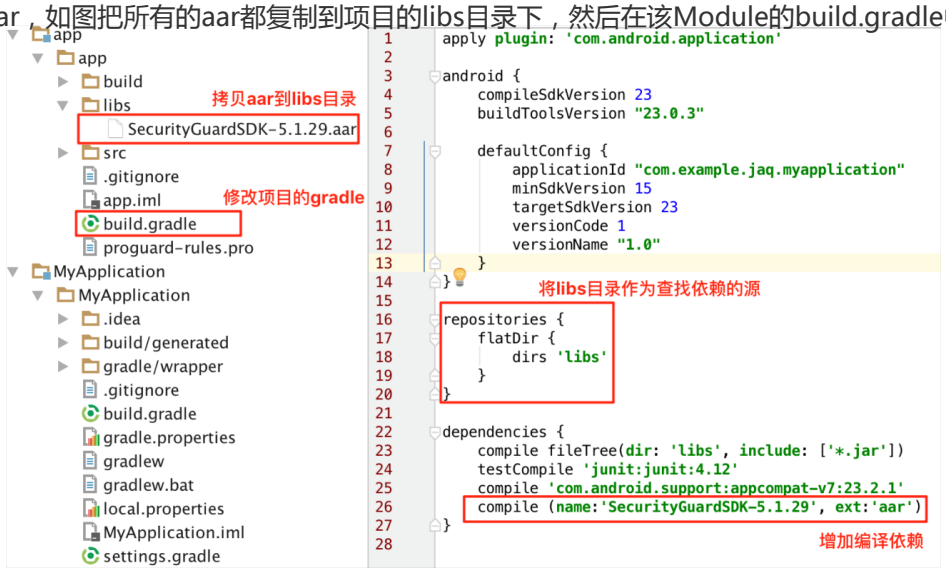
数据风控SDK在阿里巴巴的很多开放的SDK中也有包含，如果你的APK中同时引入了这些SDK，则在集成聚安全SDK之前需要将这些SDK（TAE，支付宝等)中包含的安全组件（.jar、.aar文件）、图片文件（YW_1222.*JPG*

）全部删除掉。

2.2 导入SDK

根据使用方式导入SDK，AndroidStudio使用aar方式导入，Eclipse使用jar和so方式导入；

- 导入aar，如图把所有的aar都复制到项目的libs目录下，然后在该Module的build.gradle中增加如图配置：




```
dependencies {
compile fileTree(dir: 'libs', include: ['*.jar'])
testCompile 'junit:junit:4.12'
compile 'com.android.support:appcompat-v7:23.+'
compile name: 'NoCaptchaSDK-external-release-5.1.17', ext: 'aar'
compile name: 'SecurityBodySDK-external-release-5.1.25', ext: 'aar'
compile name: 'SecurityGuardSDK-external-release-5.1.81', ext: 'aar'
compile name: 'verificationsdklib', ext: 'aar'
}
```

- 导入jar包和so，如图把SDK中的所有的jar包和so文件都复制到工程的libs目录下：

注意：

a.目前只提供两种架构下的so文件，armeabi是针对arm架构编译的包，x86是针对x86架构编译的包。应用程序在不同cpu架构的机型里会选择相应的so文件加载。

b.如果libs下有armeabi-v7a文件夹的话，需要将armeabi中对应的so复制一份到armeabi-v7a文件夹下。

c.如果libs下有arm64-v8a或x86_64文件夹的话，需要将arm64-v8a文件夹删除掉。

d.如果在想在x86或者模拟器上运行你的程序，必须导入x86架构的so，但是可以在应用发布时去掉x86目录下的so。

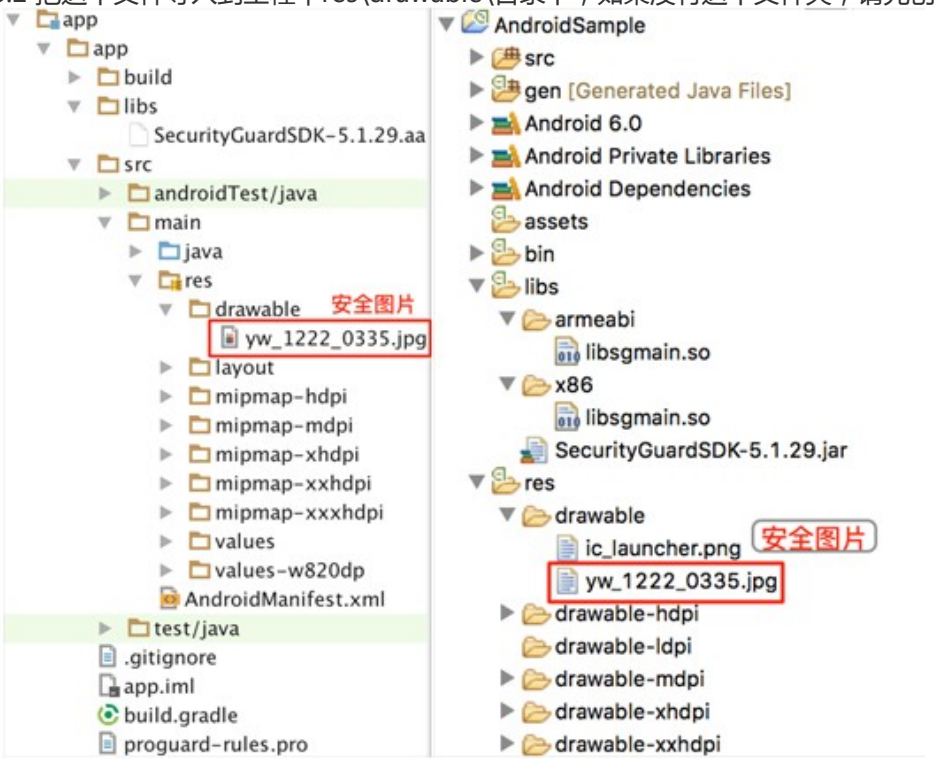
e.最后目录显示如下：

```
NoCaptchaSDK-5.1.16.jar
armeabi
SecurityBodySDK-5.1.19.jar
verificationsdklib.aar
SecurityGuardSDK-5.1.58.jar
x86
```

3.导入图片

3.1 解压第1点中生成的SDK，获得文件：yw*1222*.jpg；

3.2 把这个文件导入到工程中res\drawable\目录下，如果没有这个文件夹，请先创建，如下图：



左图为AndroidStudio中安全加密图片位置，右图为Eclipse中安全加密图片位置；

3.3 如果开启混淆要检查发布包335大小不为0， shrinkResources true会导致yw_1222_0335以及yw_1222图片为0

```
release {
minifyEnabled true // 是否混淆
shrinkResources true // <<<会导致335或者122图片为0
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
```

解决方案一

放弃资源压缩

解决方案二

参考google 关于shrink resource

新建 res/raw/keep.xml后加入如下内容:

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools"
tools:keep="@drawable/yw_1222_0335, @drawable/yw_1222"/>
```

- 保留文件规则简单介绍，资源文件相对路径加上图片文件名(不需要扩展名)；
- 执行 ./gradlew clean assembleRelease –info|grep “Skipped unused resource” 观察是否安全图片给压缩,同时检查解压缩后文件是否为0。

4.android studio修改项目文件

4.1 修改应用的工程根目录build.gradle文件：

```
allprojects {
repositories {
jcenter()
flatDir { //<------添加三行
dirs 'libs' //<------
} //<------
}
}
```

4.2 修改application子工程的build.gradle文件

```
dependencies {
compile fileTree(include: ['*.jar'], dir: 'libs')
compile(name:'verificationsdklib', ext:'aar')
}
```

4.3 修改application子工程的build.gradle文件，增加jnilib

```
sourceSets {
main {
jniLibs.srcDirs = ['libs']
}
}
```

4.4 AndroidManifest.xml 加入验证SDK入口Activity声明

```
<activity android:name="com.alibaba.verificationsdk.ui.VerifyActivity"
android:screenOrientation="portrait"
android:theme="@android:style/Theme.NoTitleBar"
android:windowSoftInputMode="adjustResize" >
</activity>
```

5.Eclispe修改项目文件5.1 添加权限信息

- 如果是AndroidStudio项目，则不需要在项目中额外配置权限，因为在aar中我们自己已经声明了权限；
- 如果是Eclipse项目，需要在AndroidManifest.xml文件中添加下列权限配置：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
```

5.2 AndroidManifest.xml 加入验证SDK入口Activity声明

```
<activity android:name="com.alibaba.verificationsdk.ui.VerifyActivity"
android:screenOrientation="portrait"
android:theme="@android:style/Theme.NoTitleBar"
android:windowSoftInputMode="adjustResize" >
</activity>
```

6.关于混淆

如果设置资源压缩 shrinkResources true，参考导入图片的处理，防止安全图片被压缩为0字节，配置 proguard-rules.pro：

```
-keep class com.taobao.securityjni.**{*;}
-keep class com.taobao.wireless.security.**{*;}
-keep class com.ut.secbody.**{*;}
-keep class com.taobao.dp.**{*;}
-keep class com.alibaba.wireless.security.**{*;}
-keep class com.alibaba.verificationsdk.**{*;}
-keep interface com.alibaba.verificationsdk.ui.IActivityCallback
```

7.SDK API

7.1 SDK初始化：

- 初始化负责完成整个数据风控安全组件的全局初始化。初始化是线程安全的，初始化调用只需要进行一次，无需重复调用；
- 查看接口详情

7.2 获取数据风控wtoken

- 获取wtoken后，服务端用wtoken为入参调用风险识别；
- 使用场景：在需要使用数据风控的场景，如注册、登陆、活动页面，可以在用户点击“注册”、“登陆”等业务按钮、业务逻辑处理前获得该token；
- 查看接口详情

iOS SDK集成配置

1.生成SDK

出于对应用数据安全考虑，数据风控生成的SDK会与应用强绑定。如果此前使用Debug版应用 生成SDK需要在应用发布前使用Release版应用重新上传，并替换原有SDK。

- 上传Release版ipa；
- 点击“生成SDK”按钮，生成SDK；

- 生成完SDK后，SDK自动下载到本地；

2.导入SDK

2.1 导入Framework

a. 把SDK中的framework文件添加到项目目录中

```
MSAuthSDK.framework SecurityGuardSDK.framework
SGMain.framework``SGNoCaptcha.framework``SGSecurityBody.framework
```

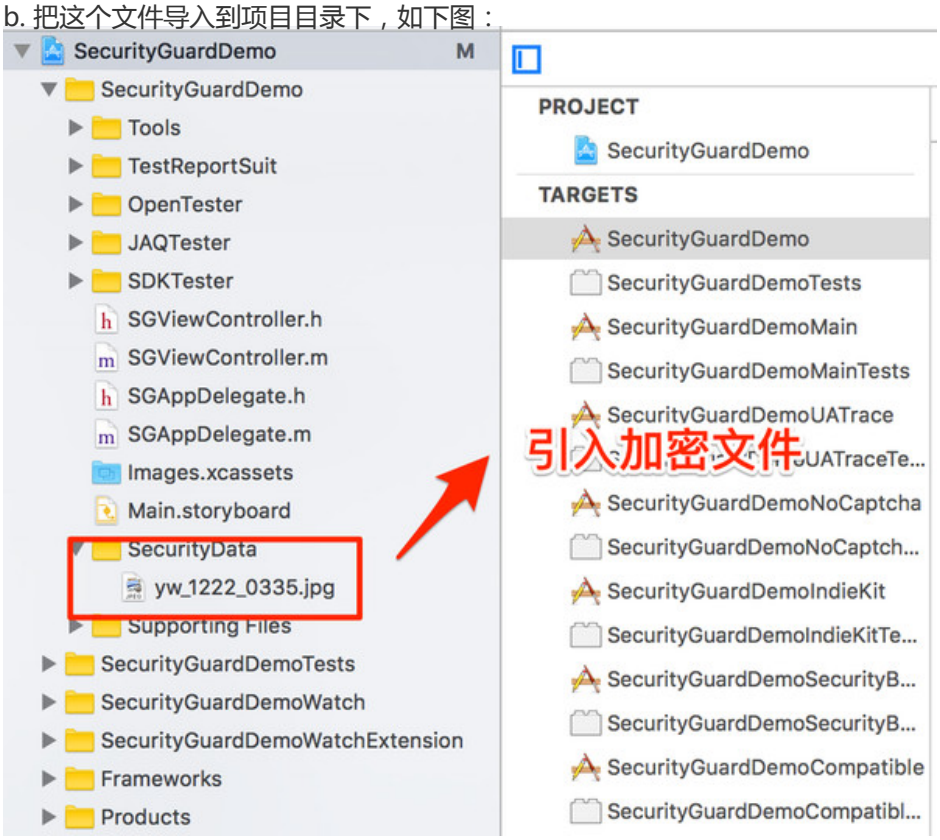
b. 将MSADefaultImages.bundle和MSADefaultLocale.bundle加入资源中

c. 如果sdk中带有xib，需要将所有xib加入资源中

2.2 导入图片

a. 解压第1点中生成的SDK，获得文件：yw*1222*.jpg；

b. 把这个文件导入到项目目录下，如下图：



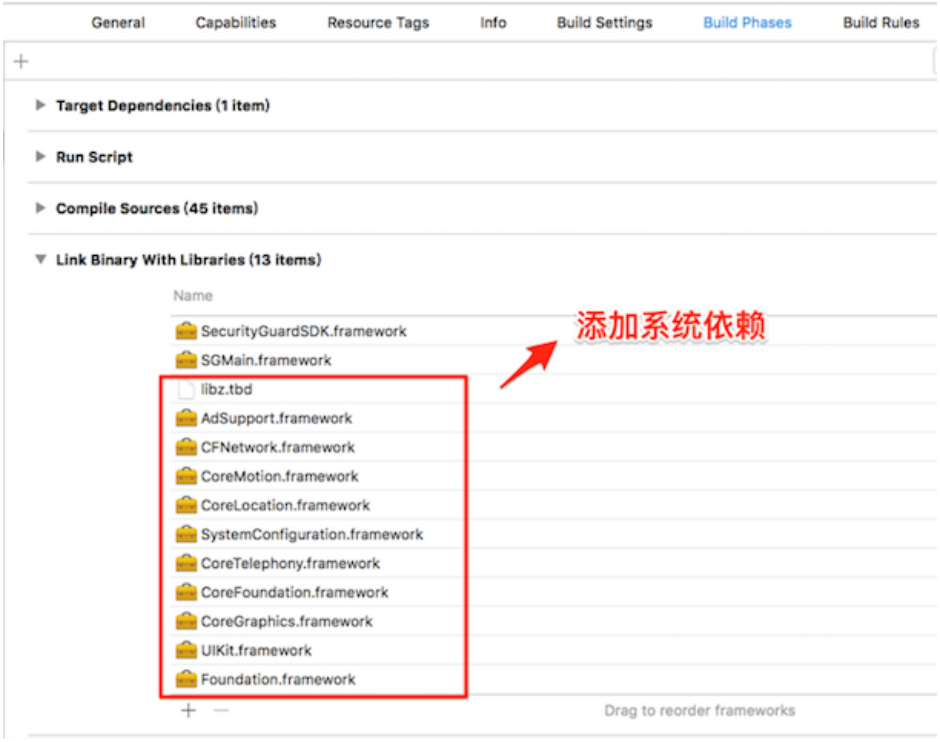
2.3 依赖 CocoaPods:

不使用pod可以直接使用sdk内压缩包：

```
pod 'SVProgressHUD', '~> 1.1'
pod 'SSZipArchive', '~> 1.1'
```

2.4 添加系统依赖库

在项目中添加其他依赖的framework，如下图：



2.5 其他项目配置

- 在Build Setting中的Other Linker Flags中添加“-ObjC”选项，如下图：

- 在info.plist中设置，开放http请求：

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSAllowsArbitraryLoads</key><true/>
</dict>
```

3.SDK API

3.1 获取数据风控wtoken

- 获取wtoken后，服务端用wtoken为入参调用风险识别；
- 使用场景：在需要使用数据风控的场景，如注册、登陆、活动页面，可以在用户点击“注册”、“登陆”等业务按钮、业务逻辑处理前获得该token；
- 接口详情:查看接口详情

服务端API调用

1. 下载对应语言的SDK，SDK下载请进入数据风控控制台
2. 将SDK加载到工程中；
3. 参考下方的代码，开发第一步页面请求的处理类（如java的Action,Controller,Servlet等）;注意填入自己的阿里云accesskey和secret；
4. 参考示例代码，对运行结果进行处理；请注意对服务端出现的错误进行兼容处理。
5. 业务风险防控API列表：

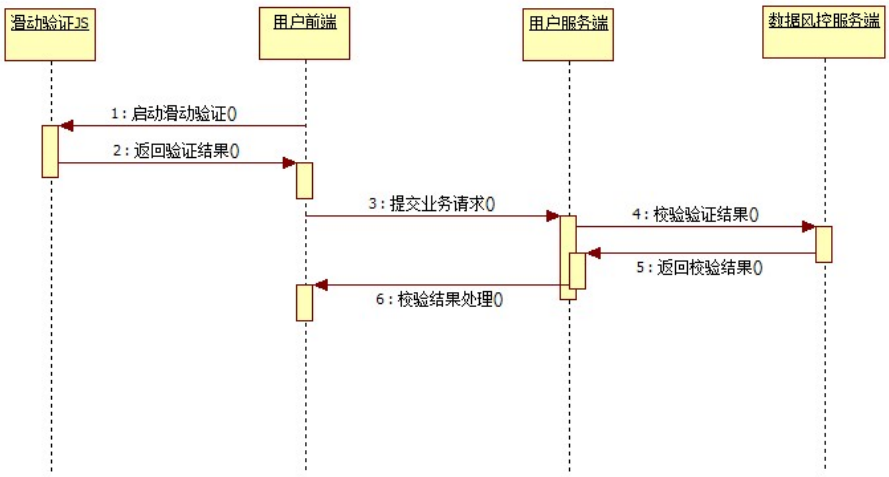
API	描述
SpamRegisterPrevention	注册防控接口
LoginPrevention	登陆防控接口
CampaignPrevention	活动防控接口
BbsPrevention	消息防控接口
OtherPrevention	其他风险防控接口

滑动验证，通过生物特征判定操作计算机的是人还是机器，从而取代传统验证方式。WEB网页、HTML5页面滑动验证组件展现形式：

WEB网页：

移动端HTML5页面：

系统间交互流程



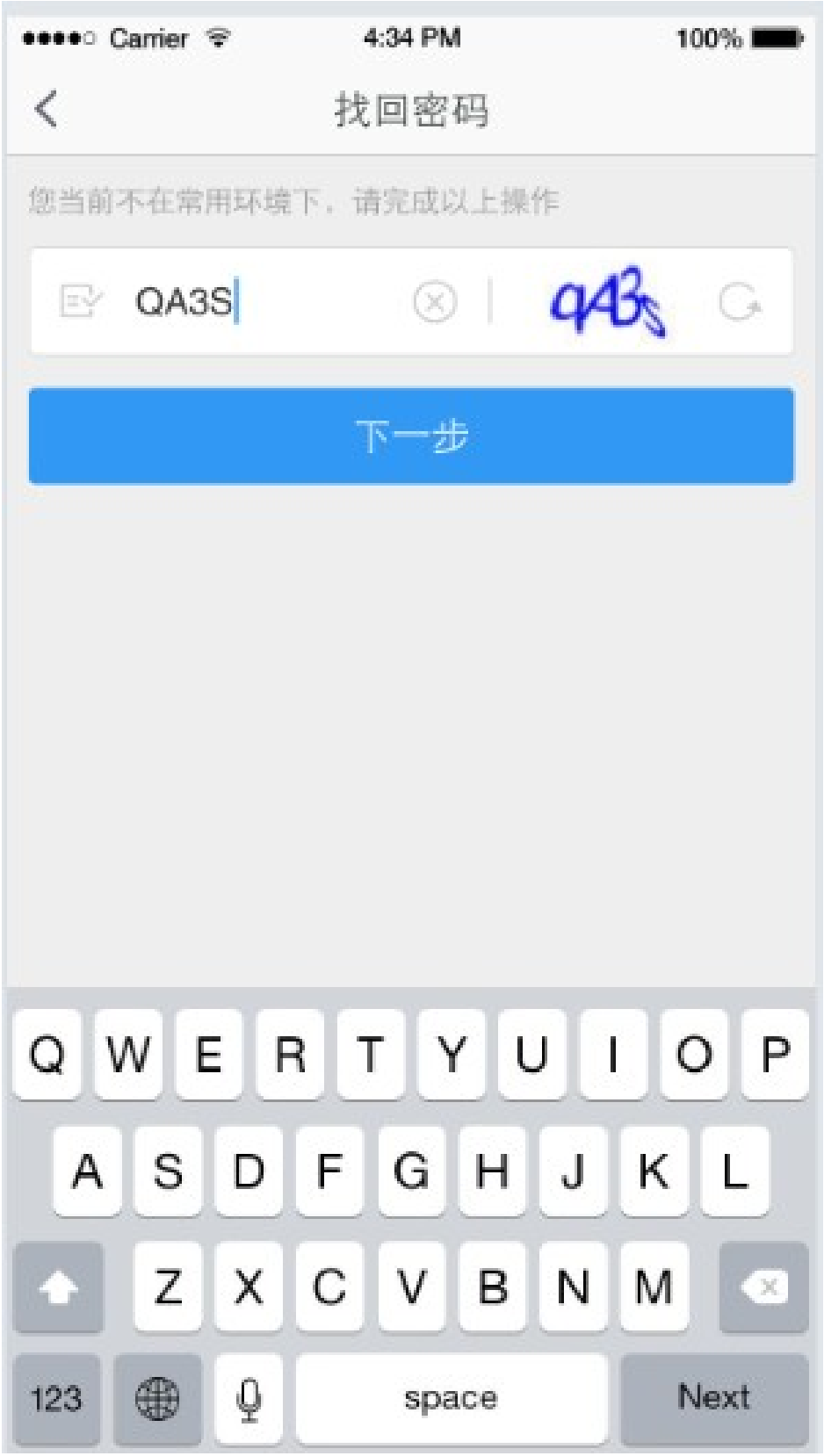
1.启动滑动验证：引入并展示滑动验证组件。用户在页面操作滑动验证：

- 正常用户：滑动直接通过；

- 中风险用户：滑动后，展示点击、输入验证码（点击验证码只在WEB端展现）；用户点击、输入成功，验证通过；

web网页识别到风险后出现点击验证：

移动端HTML5页面识别到风险后出现输入验证：



- 高风险用户：滑动后，直接拦截；

2.提交业务请求：用户提交业务请求如注册、登陆请求时，需要将验证码参数传递给服务端，验证码参数包含：

- token：请求唯一id
- scene：场景
- csessionid：session
- sig：验证成功签名

3.校验验证结果：用户服务端调用验证码服务api，获得校验结果。该步骤需要放在业务请求处理之前。
4.校验结果处理：根据返回的校验结果，进行处理。对于不同校验结果，建议处理方案：

- 校验成功：继续做业务请求处理，如注册、登陆处理；
- 校验失败：建议直接返回业务请求失败，让操作者重新再来一次。

接入流程

1.第一步，进入数据风控控制台：

- 未开通服务，开通服务；
- 已开通服务，进入服务管理，选择滑动验证服务，点击【创建】：

2.第二步，选择应用类型（网页、移动端WAP/HTML5）、使用场景、输入高峰期PV，点击【下一步】：

3.第三步，根据系统集成代码的操作步骤，将示例代码拷贝出，集成到业务系统中，详细操作参见下方：系统集成。集成完成，点击【下一步】：

4.第四步，系统集成完毕，发布对应的服务：

系统集成

前端页面引入

1. 找到需要使用风险拦截服务的页面，将下方示例代码复制进去，注意将form表单替换成自己的业务表单；
2. 运行环境：Tomcat、Apache等；
3. 接入成功校验：滑块正常显示；拖动滑块，正常滑动。打开浏览器控制台（F12开发者工具）：

- 控制台收到analyze.jsonp请求；
- 在Headers - Query String Paraments 里看到：a、t、n等参数。

4.测试不同的滑动结果：正常通过、出点击验证码通过(H5端不出现)、出图形验证码通过、直接拦截；测试方法如下：

- 设置特定的appkey值： nc_appkey = ‘CF_APP_1’ ；上线前务必调整回默认值，否则服务端校验都会不通过；

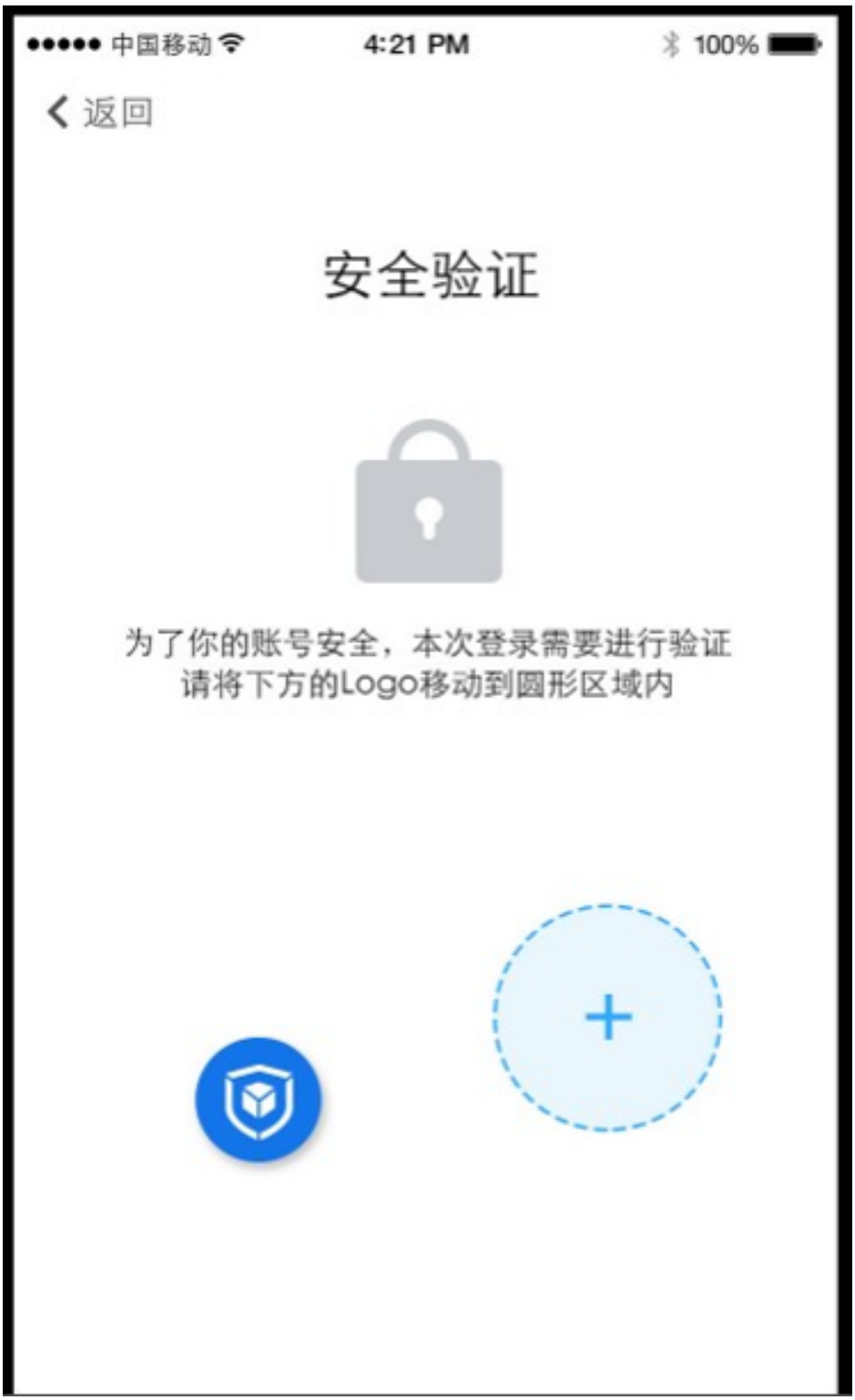
- 调整 trans里的code值。

服务端API调用

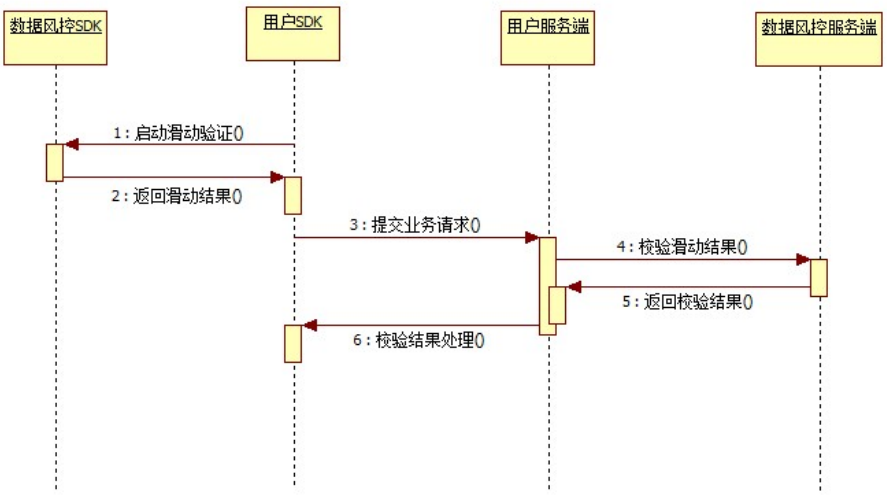
1. 下载对应语言的SDK，SDK下载请进入数据风控控制台；
2. 将SDK加载到工程中；
3. 参考示例代码，开发第一步页面请求的处理类（如java的Action,Controller,Servlet等）;注意填入自己的阿里云accesskey和secret；
4. 参考示例代码，对运行结果进行处理；请注意对服务端出现的错误进行兼容处理。
5. API详情：验证码服务web/html5接口

滑动验证，通过生物特征判定操作计算机的是人还是机器，从而取代传统验证方式。Android、iOS端滑动验证组件展现形式：

Android/iOS：


中国移动
4:21 PM
100%
返回
安全验证
为了你的账号安全，本次登录需要进行验证
请将下方的Logo移动到圆形区域内

系统间交互流程



1.启动滑动验证：用户APP调用风控SDK启动风险验证。用户在页面操作滑动验证：

- 正常用户：滑动直接通过；
- 有风险用户：滑动后失败，直接返回初始页面，提示重新滑动；

2.提交业务请求：用户提交业务请求如注册、登陆请求时，需要将验证码参数传递给服务端，验证码参数包含：

- session_id：验证会话id

3.校验验证结果：用户服务端调用验证码服务api，获得校验结果。该步骤需要放在业务请求处理之前。
4.校验结果处理：根据返回的校验结果，进行处理。对于不同校验结果，建议处理方案：

- 校验成功：继续做业务请求处理，如注册、登陆处理；
- 校验失败：建议直接返回业务请求失败，让操作者重新再来一次。

接入流程

1.第一步，进入数据风控控制台：

- 未开通服务，开通服务；
- 已开通服务，进入服务管理，选择滑动验证服务，点击【创建】：

2.第二步，选择应用类型（Android、iOS）、使用场景、输入高峰期PV，点击【下一步】：

3.第三步，根据系统集成代码的操作步骤，下载SDK，集成到APP中；服务端调用对应api，详细操作参见下方：系统集成。集成完成，点击【下一步】：

4.第四步，系统集成完毕，发布对应的服务：

系统集成

Android SDK集成配置

1.生成SDK

出于对应用数据安全考虑，数据风控生成的SDK会与应用强绑定。如果此前使用Debug版应用 生成SDK需要在应用发布前使用Release版应用重新上传，并替换原有SDK。

- 上传Release版APK；
- 点击“生成SDK”按钮，生成SDK；
- 生成完SDK后，SDK自动下载到本地；

2.导入SDK

2.1 导入前准备

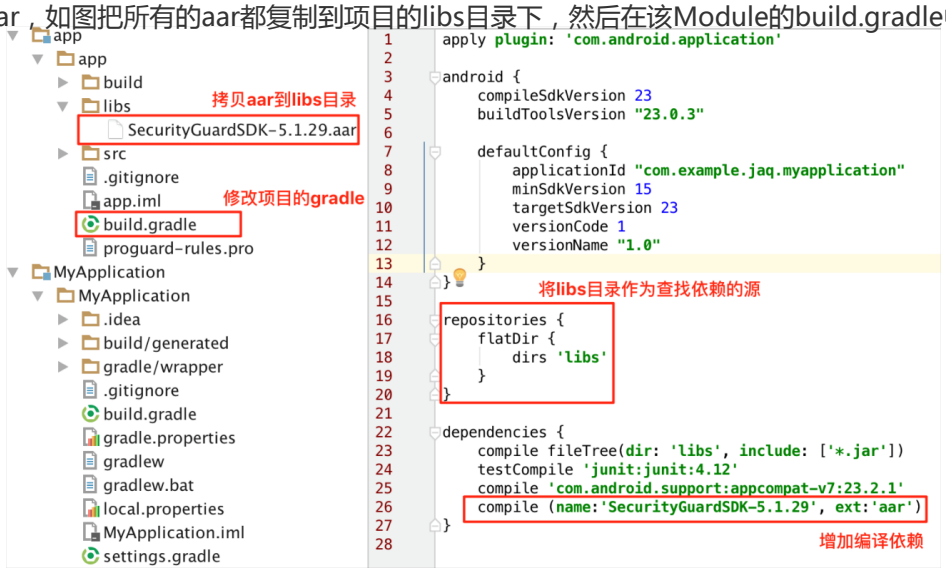
数据风控SDK在阿里巴巴的很多开放的SDK中也有包含，如果你的APK中同时引入了这些SDK，则在集成聚安

全SDK之前需要将这些SDK（TAE，支付宝等)中包含的安全组件（.jar、.aar文件）、图片文件（YW_1222.*JPG*）全部删除掉。

2.2 导入SDK

根据使用方式导入SDK，AndroidStudio使用aar方式导入，Eclipse使用jar和so方式导入；

- 导入aar，如图把所有的aar都复制到项目的libs目录下，然后在该Module的build.gradle中增加如图配置：




```
dependencies {
compile fileTree(dir: 'libs', include: ['*.jar'])
testCompile 'junit:junit:4.12'
compile 'com.android.support:appcompat-v7:23.+'
compile name: 'NoCaptchaSDK-external-release-5.1.17', ext: 'aar'
compile name: 'SecurityBodySDK-external-release-5.1.25', ext: 'aar'
compile name: 'SecurityGuardSDK-external-release-5.1.81', ext: 'aar'
compile name: 'verificationsdklib', ext: 'aar'
}
```

- 导入jar包和so，如图把SDK中的所有的jar包和so文件都复制到工程的libs目录下：

注意：

a.目前只提供两种架构下的so文件，armeabi是针对arm架构编译的包，x86是针对x86架构编译的包。应用程序在不同cpu架构的机型里会选择相应的so文件加载。

b.如果libs下有armeabi-v7a文件夹的话，需要将armeabi中对应的so复制一份到armeabi-v7a文件夹下。

c.如果libs下有arm64-v8a或x86_64文件夹的话，需要将arm64-v8a文件夹删除掉。

d.如果在想在x86或者模拟器上运行你的程序，必须导入x86架构的so，但是可以在应用发布时去掉x86目录下的so。

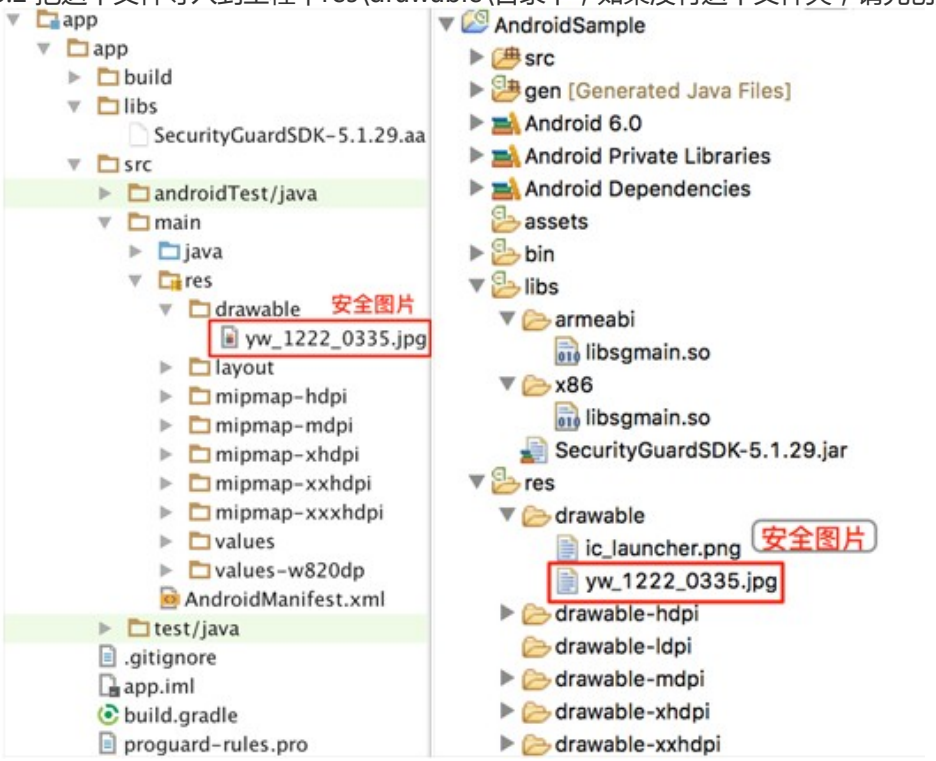
e.最后目录显示如下：

```
NoCaptchaSDK-5.1.16.jar
armeabi
SecurityBodySDK-5.1.19.jar
verificationsdklib.aar
SecurityGuardSDK-5.1.58.jar
x86
```

3.导入图片

3.1 解压第1点中生成的SDK，获得文件：yw*1222*.jpg；

3.2 把这个文件导入到工程中res\drawable\目录下，如果没有这个文件夹，请先创建，如下图：



左图为AndroidStudio中安全加密图片位置，右图为Eclipse中安全加密图片位置；

3.3 如果开启混淆要检查发布包335大小不为0， shrinkResources true会导致yw_1222_0335以及yw_1222图片为0

```
release {
minifyEnabled true // 是否混淆
shrinkResources true // <<<会导致335或者122图片为0
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
```

解决方案一

放弃资源压缩

解决方案二

参考google 关于shrink resource

新建 res/raw/keep.xml后加入如下内容:

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools"
tools:keep="@drawable/yw_1222_0335, @drawable/yw_1222"/>
```

- 保留文件规则简单介绍，资源文件相对路径加上图片文件名(不需要扩展名)；
- 执行 ./gradlew clean assembleRelease –info|grep “Skipped unused resource” 观察是否安全图片给压缩,同时检查解压缩后文件是否为0。

4.android studio修改项目文件

4.1 修改应用的工程根目录build.gradle文件：

```
allprojects {
repositories {
jcenter()
flatDir { //<------添加三行
dirs 'libs' //<------
} //<------
}
}
```

4.2 修改application子工程的build.gradle文件

```
dependencies {
compile fileTree(include: ['*.jar'], dir: 'libs')
compile(name:'verificationsdklib', ext:'aar')
}
```

4.3 修改application子工程的build.gradle文件，增加jnilib

```
sourceSets {
main {
jniLibs.srcDirs = ['libs']
}
}
```

4.4 AndroidManifest.xml 加入验证SDK入口Activity声明

```
<activity android:name="com.alibaba.verificationsdk.ui.VerifyActivity"
android:screenOrientation="portrait"
android:theme="@android:style/Theme.NoTitleBar"
android:windowSoftInputMode="adjustResize" >
</activity>
```

5.Eclispe修改项目文件5.1 添加权限信息

- 如果是AndroidStudio项目，则不需要在项目中额外配置权限，因为在aar中我们自己已经声明了权限；
- 如果是Eclipse项目，需要在AndroidManifest.xml文件中添加下列权限配置：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
```

5.2 AndroidManifest.xml 加入验证SDK入口Activity声明

```
<activity android:name="com.alibaba.verificationsdk.ui.VerifyActivity"
android:screenOrientation="portrait"
android:theme="@android:style/Theme.NoTitleBar"
android:windowSoftInputMode="adjustResize" >
</activity>
```

6.关于混淆

如果设置资源压缩 shrinkResources true，参考导入图片的处理，防止安全图片被压缩为0字节，配置 proguard-rules.pro：

```
-keep class com.taobao.securityjni.**{*;}
-keep class com.taobao.wireless.security.**{*;}
-keep class com.ut.secbody.**{*;}
-keep class com.taobao.dp.**{*;}
-keep class com.alibaba.wireless.security.**{*;}
-keep class com.alibaba.verificationsdk.**{*;}
-keep interface com.alibaba.verificationsdk.ui.IActivityCallback
```

7.SDK API

7.1 SDK初始化：

- 初始化负责完成整个数据风控安全组件的全局初始化。初始化是线程安全的，初始化调用只需要进行一次，无需重复调用；
- 查看接口详情

7.2 启动风险验证

- 使用场景：在需要使用验证码的场景，如注册、登陆、活动页面，可以直接启动验证码进行验证；
- 查看接口详情

iOS SDK集成配置

1.生成SDK

出于对应用数据安全考虑，数据风控生成的SDK会与应用强绑定。如果此前使用Debug版应用 生成SDK需要在应用发布前使用Release版应用重新上传，并替换原有SDK。

- 上传Release版ipa；
- 点击“生成SDK”按钮，生成SDK；
- 生成完SDK后，SDK自动下载到本地；

2.导入SDK

2.1 导入Framework

a. 把SDK中的framework文件添加到项目目录中

```
MSAuthSDK.framework SecurityGuardSDK.framework
SGMain.framework``SGNoCaptcha.framework``SGSecurityBody.framework
```

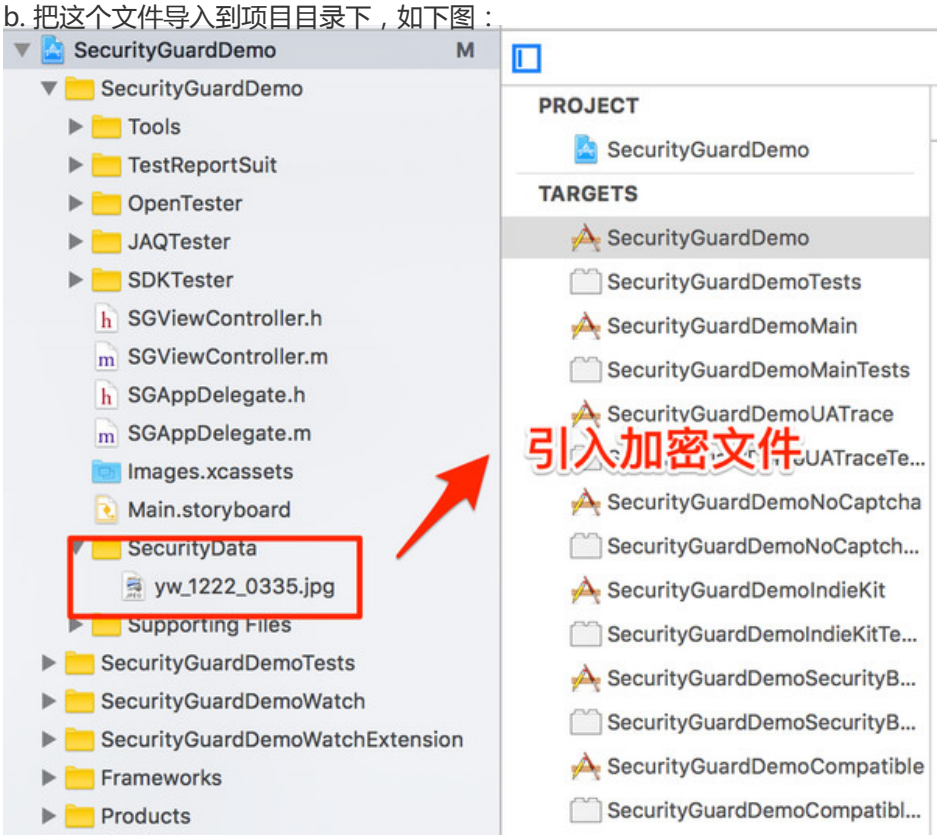
b. 将MSADefaultImages.bundle和MSADefaultLocale.bundle加入资源中

c. 如果sdk中带有xib，需要将所有xib加入资源中

2.2 导入图片

a. 解压第1点中生成的SDK，获得文件：yw*1222*.jpg；

b. 把这个文件导入到项目目录下，如下图：



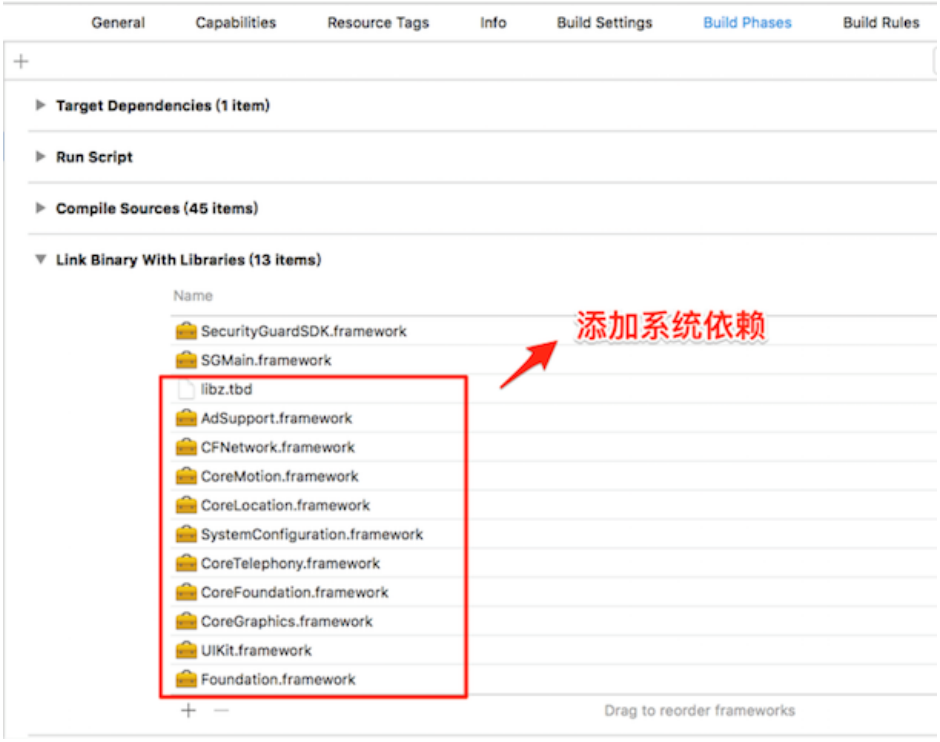
2.3 依赖 CocoaPods:

不使用pod可以直接使用sdk内压缩包：

```
pod 'SVProgressHUD', '~> 1.1'
pod 'SSZipArchive', '~> 1.1'
```

2.4 添加系统依赖库

在项目中添加其他依赖的framework，如下图：



2.5 其他项目配置

- 在Build Setting中的Other Linker Flags中添加“-ObjC”选项，如下图：

- 在info.plist中设置，开放http请求：

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSAllowsArbitraryLoads</key><true/>
</dict>
```

3.SDK API

3.1 启动风险验证

- 使用场景：在需要使用验证码的场景，如注册、登陆、活动页面，可以直接启动验证码进行验证；
- 查看接口详情

服务端API调用

1. 下载对应语言的SDK，SDK下载请进入数据风控控制台；

2. 将SDK加载到工程中；
3. 参考下方的代码，开发第一步页面请求的处理类（如java的Action,Controller,Servlet等）;注意填入自己的阿里云accesskey和secret；
4. 参考示例代码，对运行结果进行处理；请注意对服务端出现的错误进行兼容处理。
5. API详情：验证码服务Android/iOS接口

业务风控-使用说明