阿里云号码认证iOS v2.12.1（标准版）接入文档

1. 概述

2. 准备工作

3. 运行demo工程

4. 开发环境搭建

   4.1 添加主库

   4.2 Targets->BuildSettings设置

   4.4 添加资源文件

   4.5 Crash收集（建议开启）

      功能介绍

      注意事项

5. SDK 使用说明

   5.1 SDK功能介绍

   5.2 一键登录接口调用说明

   5.3 本机号码校验接口调用说明

6. SDK方法说明

   6.1 主类TXCommonHandler接口简介

      6.1.1 获取认证实例(sharedInstance)

      6.1.2 获取sdk版本号（getVersion）

      6.1.3 设置SDK秘钥(setAuthSDKInfo)

      6.1.4 检查认证环境(checkEnvAvailableWithAuthType)

      6.1.5 一键登录预取号(accelerateLoginPageWithTimeout)

      6.1.6 一键登录获取token(getLoginTokenWithTimeout)

      6.1.7 隐藏授权时关闭loading(hideLoginLoading)

      6.1.8 一键登录注销登录页面(cancelLoginVCAnimated)

      6.1.9 获取日志埋点相关控制对象(getReporter)

      6.1.10 加速获取本机号码校验token(accelerateVerifyWithTimeout)

      6.1.11 本机号码校验获取token(getVerifyTokenWithTimeout)

   6.2 日志埋点设置类PNSReporter接口简介

6.2.1 设置控制台日志输出开关(setConsolePrintLoggerEnable)
6.2.2 设置日志及埋点上传开关(setUploadEnable)
6.3 工具类TXCommonUtils接口简介
6.3.1 判断设备蜂窝网络是否开启(checkDeviceCellularDataEnable)
6.3.2 判断当前上网卡是否是中国联通(isChinaUnicom)
6.3.3 判断当前上网卡是否是中国移动(isChinaMobile)
6.3.4 判断当前上网卡是否是中国电信(isChinaTelecom)
6.3.5 获取当前上网卡网络名称(getCurrentMobileNetworkName)
6.3.6 获取当前上网卡运营商名称(getCurrentCarrierName)
6.3.7 获取当前上网网络类型(getNetworktype)
6.3.8 判断设备是否有SIM卡(simSupportedIsOK)
6.3.9 判断wwan是否开启(isWWANOpen)
6.3.10 判断无wifi下wwan是否开启(reachableViaWWAN)
6.3.11 获取设备当前网络私网IP地址(getMobilePrivateIPAddress)
6.4 一键登录获取手机号
6.5 本机号码校验结果
6.6 SDK返回码
6.7 授权页点击事件响应码
7. SDK接入demo用例
7.1 一键登录唤起授权页
7.2 授权页全屏模式，横竖屏切换
7.3 授权页弹窗模式，支持横竖屏切换
7.4 本机号码校验用例
8. 一键登录授权页面说明
8.1 全屏授权页面设计规范（支持横竖屏，以竖屏示意）
8.2 弹窗授权页面设计规范（支持横竖屏，以竖屏示意）
9. iOS常见问题

1. 概述

官网下载SDK后进行解压，解压后包含五个文件/目录是

a） iOS SDK开发接入文档
b） SceneDemo工程（场景化Demo工程，通过在控制台下载DEMO压缩包）
c） ATAuthSDK.framework（号码认证静态库）
d） ATAuthSDK_D（号码认证动态库）
e） AliComCrash（号码认证Crash收集库）
f）YTXOperators.framework（运营商相关组件）
g) YTXMonitor.framework（埋点组件）

号码认证服务包含本机号码校验和一键登录两个不同的功能，使用场景不一样，无需一起使用。
本机号码校验使用场景：用户输入手机号码，通过SDK获取token（**通过`getVerifyTokenWithTimeout`获得**），服务端携带输入的手机号码和token去运营商网关进行校验（**服务端调用`VerifyMobile`接口**），返回的结果是用户当前上网使用的号码与输入的号码是否一致。
一键登录使用场景：用户无需输入手机号码，SDK会拉起授权页，用户确认授权后，SDK会获取token（**通过`getLoginTokenWithTimeout`获得**），服务端携带token到运营商网关获取用户当前上网使用的号码（**服务端调用`GetMobile`接口**），并返回给App服务端。

2. 准备工作

a） 请确保您的终端设备已经开启了4G网络（联通、移动支持3G网络，但接口耗时会增加）
b） 请确保已经在阿里云控制台开通了号码认证服务并创建了对应的方案，点击进入阿里云控制台。方案创建成功后，联通电信可立即使用，移动需等待10分钟后使用。
c） 您也可以登录阿里云官网查看接入流程，点击查看完整使用流程。

3. 运行demo工程

解压后包含SceneDemo工程，您需要将bundleID和开发者证书修改为您自己的。

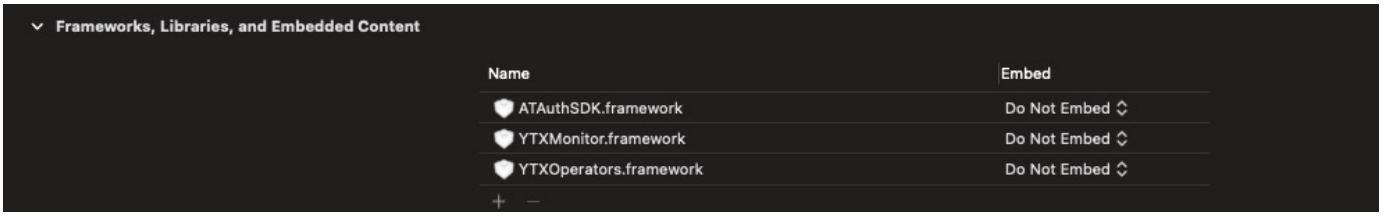
4. 开发环境搭建

前提条件：应用必须运行在iOS 9.0+平台上。

4.1 添加主库

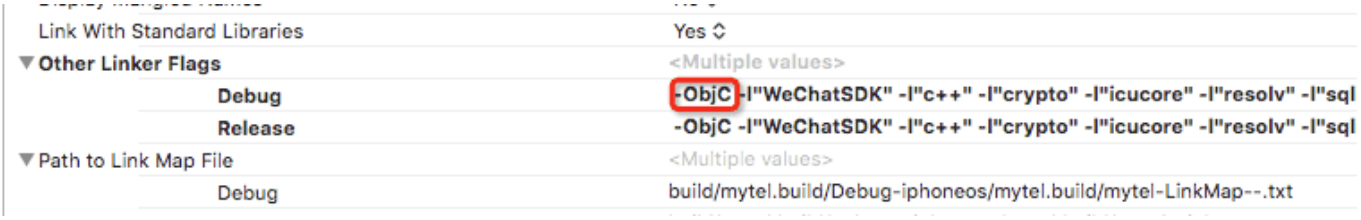
Targets->General->Linked Frameworks and Libraries 中添加主库ATAuthSDK.framework, YTXMonitor.framework, YTXOperators.framework；

或者Xcode 11中Targets->General->Frameworks,Libraries, and Embedded Content 中添加主库ATAuthSDK.framework, YTXMonitor.framework, YTXOperators.framework；



4.2 Targets->BuildSettings设置

Targets->BuildSettings 中 ， Other Linker Flags增加-ObjC，一定要添加此选项，**注意是大写C，不是小写c**，否则工程运行会crash!



4.4 添加资源文件

主项目右键添加下ATAuthSDK.framework中ATAuthSDK.bundle ， 否则一键登录授权页面默认的图片或icon显示不出来；

4.5 Crash收集（建议开启）

功能介绍

从v2.10.1版本之后，号码认证SDK支持自己收集SDK内部的Crash问题，方便线上问题排查。客户可以自行选择是否让SDK支持该能力，如果需要支持，则要接入SDK包中提供的 ATAuthSDK_D.framework（号码认证动态库）和 AliComCrash.framework（Crash收集组件库）。

注意事项

- 该Crash组件库只会收集号码认证SDK的Crash，工程里面的其他Crash不会收集
- ATAuthSDK_D.framework 和 ATAuthSDK.framework 不能同时集成到工程里面，他们两是同一个SDK，只是前者是动态库，后者是静态库
- 集成号码认证Crash收集能力，工程中必须使用ATAuthSDK_D目录中ATAuthSDK_D.framework（动态库），静态库则不会去收集，而且同时要在工程中加入AliComCrash目录中的 AliComCrash.framework（Crash收集组件库）

5. SDK 使用说明

5.1 SDK功能介绍

阿里云号码认证iOS SDK目前主要提供功能，*目前仅提供静态库ATAuthSDK.framwork形式。*

- 获取SDK版本号。
- SDK接口调用环境检查及准备，检测设备是否开启蜂窝数据网络，且配置SDK相关数据，该步骤是本机号码校验及一键登录功能可以使用的前提条件。
- 本机号码校验`getVerifyTokenWithTimeout:complete:`接口（支持移动，电信，联通三大运营商）请求本机号码校验的凭证，可自定义超时时间，默认3.0s，单位s。

一键登录`accelerateLoginPageWithTimeout:complete:`接口（支持移动，电信，联通三大运营商），是预取号缓存接口，为`getLoginTokenWithTimeout:controller:model:complete:`接口缓存前置条件，加快授权页的唤起速度。

- 一键登录`getLoginTokenWithTimeout:controller:model:complete:`接口（支持移动，电信，联通三大运营商），请求一键登录的凭证，可自定义超时时间，默认3.0s，单位s。

5.2 一键登录接口调用说明

1） 调用接口`setAuthSDKInfo:complete:`，**初始化**`SDK`**秘钥参数**

2）(可选)调用接口`checkEnvAvailableWithAuthType:complete:`，**检查当前环境是否支持一键登录**

3）(可选，唤起授权页更快)调用接口`accelerateLoginPageWithTimeout:complete:`，`SDK`**预取号缓存，主要获取手机掩码及运营商相关协议信息**，调用该接口可以加快授权页唤起的速度，**请先进行登录态判断后，确保用户需要登录再调用此接口，否则您的功能可能会被限流**

4） 调用SDK一键登录token接口`getLoginTokenWithTimeout:controller:model:complete:`弹起授权页，点击登陆按钮获取一键登录token

5） 调用服务端`getMobile`接口获取最终号码

6）(可选)调用接口`cancelLoginVCAnimated`，注销授权页面

5.3 本机号码校验接口调用说明

1) 调用接口`setAuthSDKInfo:complete:`，初始化SDK秘钥参数

2）(可选)调用接口`checkEnvAvailableWithAuthType:complete:`，**检查当前环境是否支持本季号码检验**

3) 调用SDK一键登录token接口`getVerifyTokenWithTimeout:complete:`获取本机号码校验token

4）调用服务端`VerifyMobile`接口进行本机号码校验

6. SDK方法说明

6.1 主类TXCommonHandler接口简介

6.1.1 获取认证实例(sharedInstance)

```
/**
 * 函数名：sharedInstance
 * @param 无
 * 返回：获取该类的单例实例对象
 */
+ (instancetype _Nonnull )sharedInstance;
```

6.1.2 获取sdk版本号（getVersion）

```
/**
 * 函数名：getVersion
 * @param 无
 * 返回：字符串，sdk版本号
 */
- (NSString *_Nonnull)getVersion;
```

6.1.3 设置SDK秘钥(setAuthSDKInfo)

```
/**
 * 函数名：setAuthSDKInfo
 * @brief 初始化SDK调用参数，app生命周期内调用一次
 * @param  方案对应的秘钥，请登录阿里云控制台后，进入认证方案管理，点击秘钥后复制秘钥，建议维护在业务服务端
 * @param complete 结果异步回调到主线程，成功时resultDic=@{resultCode:600000, msg:...}，其他情况时"resultCode"值请参考PNSReturnCode
 */
- (void)setAuthSDKInfo:(NSString * _Nonnull)info complete:(void(^_Nullable)(NSDictionary * _Nonnull resultDic))complete;
```

6.1.4 检查认证环境(checkEnvAvailableWithAuthType)

```
/**
 * 函数名：checkEnvAvailableWithAuthType
 * @brief检查当前环境是否支持一键登录或号码认证，resultDic 返回 PNSCodeSuccess 说明当前环境支持
 * @param  authType 服务类型 PNSAuthTypeVerifyToken 本机号码校验流程，PNSAuthTypeLoginToken 一键登录流程
 * @param  complete 结果异步回调到主线程，成功时resultDic=@{resultCode:600000, msg:...}，其他情况时"resultCode"值请参考PNSReturnCode
 */
```

7. - (void)checkEnvAvailableWithAuthType:(PNSAuthType)authType complete:(void (^_Nullable)(NSDictionary * _Nullable resultDic))complete;

6.1.5 一键登录预取号(accelerateLoginPageWithTimeout)

1. /**
2. * 函数名: accelerateLoginPageWithTimeout
3. * @brief 加速一键登录授权页弹起，防止调用 getLoginTokenWithTimeout:controller:model:complete: 等待弹起授权页时间过长
4. * @param timeout：接口超时时间，单位s，默认3.0s，值为0.0时采用默认超时时间
5. * @param complete 结果异步回调到主线程，成功时resultDic=@{resultCode:600000, msg:...}，其他情况时"resultCode"值请参考PNSReturnCode
6. */
7. - (void)accelerateLoginPageWithTimeout:(NSTimeInterval)timeout complete:(void (^_Nullable)(NSDictionary * _Nonnull resultDic))complete;

6.1.6 一键登录获取token(getLoginTokenWithTimeout)

1. /**
2. * 函数名: getLoginNumberWithTimeout
3. * @brief 获取一键登录Token，调用该接口首先会弹起授权页，点击授权页的登录按钮获取Token
4. * @warning 注意的是，如果前面没有调用 accelerateLoginPageWithTimeout:complete: 接口，该接口内部会自动先帮我们调用，成功后才会弹起授权页，所以有一个明显的等待过程
5. * @param timeout: 接口超时时间，单位s，默认3.0s，值为0.0时采用默认超时时间
6. * @paramcontroller 唤起自定义授权页的容器，内部会对其进行验证，检查是否符合条件
7. * @param model 自定义授权页面选项，可为nil，采用默认的授权页面，具体请参考TXCustomModel.h文件

8. * @paramcomplete 结果异步回调到主线程，"resultDic"里面的"resultCode"值请参考PNSReturnCode，如下：
    * 授权页控件点击事件：700000（点击授权页返回按钮）、700001（点击切换其他登录方式）、
    * 700002（点击登录按钮事件，根据返回字典里面的 "isChecked"字段来区分check box是否被选中，只有被选中的时候 * 内部才会去获取Token）、700003（点击check box事件）、700004（点击协议富文本文字）
    * 接口回调其他事件：600001（授权页唤起成功）、600002（授权页唤起失败）、600000（成功获取Token）、 * 600011（获取Token失败）、600015（获取Token超时）、600013（运营商维护升级，该功能不可用）、600014（运营商维护升级，该功能已达最大调用次数）.....
8. */
10. - (void)getLoginTokenWithTimeout:(NSTimeInterval)timeout controller:(UIViewController *_Nonnull)controller model:(TXCustomModel *_Nullable)model complete:(void (^_Nullable)(NSDictionary * _Nonnull resultDic))complete;

6.1.7 隐藏授权时关闭loading(hideLoginLoading)

```
/**
 * 函数名：hideLoginLoading
 * @brief 手动隐藏一键登录获取登录Token之后的等待动画，默认为自动隐藏，当设置 TXCustomModel 实例 autoHideLoginLoading = NO 时，可调用该方法手动隐藏
 */
- (void)hideLoginLoading;
```

6.1.8 一键登录注销登录页面(cancelLoginVCAnimated)

```
/**
 * 函数名：cancelLoginVCAnimated
 * @param flag：是否添加动画
 * @param complete成功返回
 */
-(void)cancelLoginVCAnimated:(BOOL)flag complete:(void (^_Nullable) (void))complete;
```

6.1.9 获取日志埋点相关控制对象(getReporter)

```
/**
 * 函数名：getReporter
 * @brief 获取日志埋点相关控制对象
 */
- (PNSReporter * _Nonnull)getReporter;
```

6.1.10 加速获取本机号码校验token(accelerateVerifyWithTimeout)

```
/**
 * 函数名：accelerateVerifyWithTimeout
 * @param timeout：接口超时时间，单位s，默认3.0s，值为0.0时采用默认超时时间
 * @param complete 结果异步回调到主线程，成功时resultDic=@{resultCode:600000, token:..., msg:...}，其他情况时"resultCode"值请参考PNSReturnCode
 */
- (void)accelerateVerifyWithTimeout:(NSTimeInterval)timeout complete:(void (^_Nullable)(NSDictionary * _Nonnull resultDic))complete;
```

6.1.11 本机号码校验获取token(getVerifyTokenWithTimeout)

```
/**
```

```
2.  * 函数名：getVerifyTokenWithTimeout
3.  * @param timeout：接口超时时间，单位s，默认3.0s，值为0.0时采用默认超时时间
4.  * @param complete 结果异步回调到主线程，成功时resultDic=@{resultCode:600000, token:..., msg:...}，其他情时"resultCode"值请参考PNSReturnCode
5.  */
6.  - (void)getVerifyTokenWithTimeout:(NSTimeInterval)timeout complete:(void (^_Nullable)(NSDictionary *_Nonnull resultDic))complete;
```

6.2 日志埋点设置类PNSReporter接口简介

6.2.1 设置控制台日志输出开关(setConsolePrintLoggerEnable)

```
1. /**
2.  * 函数名：setConsolePrintLoggerEnable
3.  * @brief 控制台日志输出开关，若开启会以PNS_LOGGER为开始标记对日志进行输出
4.  * @param enable 开关参数，默认为NO
5.  */
6.  - (void)setConsolePrintLoggerEnable:(BOOL)enable;
```

6.2.2 设置日志及埋点上传开关(setUploadEnable)

```
1. /**
2.  * 函数名：setUploadEnable
3.  * @brief 设置日志及埋点上传开关，但不会对通过 setupUploader: 接口实现的自定义上传方法起作用
4.  * @param enable 开关设置BOOL值，默认为YES
5.  */
6.  - (void)setUploadEnable:(BOOL)enable;
```

6.3 工具类TXCommonUtils接口简介

6.3.1 判断设备蜂窝网络是否开启(checkDeviceCellularDataEnable)

```
1. /**
2.  * 函数名：checkDeviceCellularDataEnable
3.  */
4. + (BOOL)checkDeviceCellularDataEnable;
```

6.3.2 判断当前上网卡是否是中国联通(isChinaUnicom)

```
1. /**
2.  * 函数名：isChinaUnicom
```

```
3.  */
4. + (BOOL)isChinaUnicom;
```

6.3.3 判断当前上网卡是否是中国移动(isChinaMobile)

```
1. /**
2.  * 函数名：isChinaMobile
3.  */
4. + (BOOL)isChinaMobile;
```

6.3.4 判断当前上网卡是否是中国电信(isChinaTelecom)

```
1. /**
2.  * 函数名：isChinaTelecom
3.  */
4. + (BOOL)isChinaTelecom;
```

6.3.5 获取当前上网卡网络名称(getCurrentMobileNetworkName)

```
1. /**
2.  * 函数名：getCurrentMobileNetworkName
3.  * @return：ChinaMobile,ChinaUnicom,ChinaTelecom,OtherChinaMobileNetwork,NoChinaMobileNetwork
4.  */
5. + (NSString *)getCurrentMobileNetworkName;
```

6.3.6 获取当前上网卡运营商名称(getCurrentCarrierName)

```
1. /**
2.  * 函数名：getCurrentCarrierName
3.  * @return：中国移动，中国联通，中国电信等
4.  */
5. + (NSString *)getCurrentCarrierName;
```

6.3.7 获取当前上网网络类型(getNetworktype)

```
1. /**
2.  * 函数名：getNetworktype
3.  * @return：WiFi，4G，3G，2G，NoInternet等
4.  */
5. + (NSString *)getNetworktype;
```

6.3.8 判断设备是否有SIM卡(simSupportedIsOK)

```
/**
 * 函数名：simSupportedIsOK
 * @return：
 */
+ (BOOL)simSupportedIsOK;
```

6.3.9 判断wwan是否开启(isWWANOpen)

```
/**
 * 函数名：isWWANOpen
 * @breif：判断wwan是否开启（通过p0网卡判断，无wifi或有wifi情况下都能检测到）
 */
+ (BOOL)isWWANOpen;
```

6.3.10 判断无wifi下wwan是否开启(reachableViaWWAN)

```
/**
 * 函数名：reachableViaWWAN
 * @return：
 */
+ (BOOL)reachableViaWWAN;
```

6.3.11 获取设备当前网络私网IP地址(getMobilePrivateIPAddress)

```
/**
 * 函数名：getMobilePrivateIPAddress
 * @return：
 */
+ (NSString *)getMobilePrivateIPAddress;
```

6.4 一键登录获取手机号

当您成功获取到getLoginTokenWithTimeout成功获取token后，将token传递至您的服务端，服务端携带token调用阿里云的getMobile接口即可进行最终的取号操作。

6.5 本机号码校验结果

当您成功获取到getVerifyTokenWithTimeout成功获取token后，将token与输入的手机号码传递至您的服务端，服务端携带token调用阿里云的verifyMobile接口即可进行最终的校验操作。

6.6 SDK返回码

该返回码为阿里云号码认证SDK自身的返回码，请注意600011及600012错误内均含有运营商返回码，具体错误在碰到之后查阅阿里云官网，

返回码	返回码描述	建议
600000	获取token成功	无
600001	唤起授权页成功	无
600002	唤起授权页失败	建议切换到其他登录方式
600004	获取运营商配置信息失败	创建工单联系工程师
600007	未检测到sim卡	提示用户检查 SIM 卡后重试
600008	蜂窝网络未开启	提示用户开启移动网络后重试
600009	无法判断运营商	创建工单联系工程师
600010	未知异常	创建工单联系工程师
600011	获取token失败	切换到其他登录方式
600012	预取号失败	切换到其他登录方式
600013	运营商维护升级，该功能不可用	创建工单联系工程师
600014	运营商维护升级，该功能已达最大调用次数	创建工单联系工程师
600015	接口超时	切换到其他登录方式
600017	AppID、Appkey解析失败	创建工单联系工程师
600021	点击登录时检测到运营商已切换	切换到其他登录方式
600025	终端环境检测失败（终端不支持认证 / 终端检测参数错误）	检查接口调用传参是否正确
600026	授权页已加载时不允许调用加速或预取号接口	授权页显示期间，不允许调用预取号接口

6.7 授权页点击事件响应码

响应码	响应码描述

700000	点击返回，用户取消免密登录
700001	点击切换按钮，用户取消免密登录
700002	点击登录按钮事件
700003	点击check box事件
700004	点击协议富文本文字事件

7. SDK接入demo用例

7.1 一键登录唤起授权页

用例代码详情可参考demo工程代码，如果不想调用accelerateLoginPageWithTimeout，也能正常拉起页面，也可提前调用拉取缓存。

```
//1. 设置SDK参数，app生命周期内调用一次即可
NSString *info = @"客户的秘钥串";
__weak typeof(self) weakSelf = self;
//设置SDK参数，app生命周期内调用一次即可
[[TXCommonHandler sharedInstance] setAuthSDKInfo:info complete:^(NSDictionary * _Nonnull resultDic) {
    [weakSelf showResult:resultDic];
}];


//2. 检测当前环境是否支持一键登录，支不支持提前知道
__block BOOL support = YES;
[[TXCommonHandler sharedInstance] checkEnvAvailableWithAuthType:PNSAuthTypeLoginToken complete:^(NSDictionary * _Nullable resultDic) {
    support = [PNSCodeSuccess isEqualToString:[resultDic objectForKey:@"resultCode"]];
}];

//3. 开始一键登录流程
//3.1 调用加速授权页弹起接口，提前获取必要参数，为后面弹起授权页加速
[[TXCommonHandler sharedInstance] accelerateLoginPageWithTimeout:timeout complete:^(NSDictionary * _Nonnull resultDic) {
```

```
    if ([PNSCodeSuccess isEqualToString:[resultDic objectForKey:@"resultCode"]] == NO) {
        [ProgressHUD showError:@"取号，加速授权页弹起失败"];
        [weakSelf showResult:resultDic];
        return ;
    }

    //3.2 调用获取登录Token接口，可以立马弹起授权页，model的创建需要放在主线程
    [ProgressHUD dismiss];
    [[TXCommonHandler sharedInstance] getLoginTokenWithTimeout:timeout controller:weakSelf model:model complete:^(NSDictionary * _Nonnull resultDic) {

        NSString *code = [resultDic objectForKey:@"resultCode"];
        if ([PNSCodeLoginControllerPresentSuccess isEqualToString:code]) {
            [ProgressHUD showSuccess:@"弹起授权页成功"];
        } else if ([PNSCodeLoginControllerClickCancel isEqualToString:code]) {
            [ProgressHUD showSuccess:@"点击了授权页的返回"];
        } else if ([PNSCodeLoginControllerClickChangeBtn isEqualToString:code]) {
            [ProgressHUD showSuccess:@"点击切换其他登录方式按钮"];
        } else if ([PNSCodeLoginControllerClickLoginBtn isEqualToString:code]) {
            if ([[resultDic objectForKey:@"isChecked"] boolValue] == YES) {
                [ProgressHUD showSuccess:@"点击了登录按钮，check box选中，SDK内部接着会去获取登陆Token"];
            } else {
                [ProgressHUD showSuccess:@"点击了登录按钮，check box选中，SDK内部不会去获取登陆Token"];
            }
        } else if ([PNSCodeLoginControllerClickCheckBoxBtn isEqualToString:code]) {
            [ProgressHUD showSuccess:@"点击check box"];
        } else if ([PNSCodeLoginControllerClickProtocol isEqualToString:code]) {
            [ProgressHUD showSuccess:@"点击了协议富文本"];
```

```
        } else if ([PNSCodeSuccess isEqualToString:code]) {
            //点击登录按钮获取登录Token成功回调
            NSString *token = [resultDic objectForKey:@"token"];
            //下面拿Token去服务器换手机号，下面仅做参考
            [PNSVerifyTopRequest requestLoginWithToken:token complete:^(BOOL isSuccess, NSString * _Nonnull msg, NSDictionary * _Nonnull data) {
                NSString *popCode = [data objectForKey:@"code"];
                NSDictionary *module = [data objectForKey:@"module"];
                NSString *mobile = module[@"mobile"];
                if ([popCode isEqualToString:@"OK"] && mobile.length > 0) {
                    [ProgressHUD showSuccess:@"一键登录成功"];
                } else {
                    [ProgressHUD showSuccess:@"一键登录失败"];
                }
                dispatch_async(dispatch_get_main_queue(), ^{
                    [[TXCommonHandler sharedInstance] cancelLoginVCAnimated:YES complete:nil];
                });
                [weakSelf showResult:data];
            }];
        } else {
            [ProgressHUD showError:@"获取登录Token失败"];
        }
        [weakSelf showResult:resultDic];
    }];
}];
```

7.2 授权页全屏模式，横竖屏切换

```
TXCustomModel *model = [[TXCustomModel alloc] init];

model.navColor = UIColor.orangeColor;
model.navTitle = [[NSAttributedString alloc] initWithString:@"一键登录（全屏）" attributes:@{NSForegroundColorAttributeName : UIColor.whiteColor,NSFontAttributeName : [UIFont systemFontOfSize:20.0]}];
```

```
//model.navIsHidden = NO;
model.navBackImage = [UIImage imageNamed:@"icon_nav_back_light"];
//model.hideNavBackItem = NO;
UIButton *rightBtn = [UIButton buttonWithType:UIButtonTypeSystem];
[rightBtn setTitle:@"更多" forState:UIControlStateNormal];
model.navMoreView = rightBtn;

model.privacyNavColor = UIColor.orangeColor;
model.privacyNavBackImage = [UIImage imageNamed:@"icon_nav_back_light"];
model.privacyNavTitleFont = [UIFont systemFontOfSize:20.0];
model.privacyNavTitleColor = UIColor.whiteColor;

model.logoImage = [UIImage imageNamed:@"taobao"];
//model.logoIsHidden = NO;
//model.sloganIsHidden = NO;
model.sloganText = [[NSAttributedString alloc] initWithString:@"一键登录slogan文案" attributes:@{NSForegroundColorAttributeName : UIColor.orangeColor,NSFontAttributeName : [UIFont systemFontOfSize:16.0]}];
model.numberColor = UIColor.orangeColor;
model.numberFont = [UIFont systemFontOfSize:30.0];
model.loginBtnText = [[NSAttributedString alloc] initWithString:@"一键登录" attributes:@{NSForegroundColorAttributeName : UIColor.whiteColor,NSFontAttributeName : [UIFont systemFontOfSize:20.0]}];
//model.autoHideLoginLoading = NO;
//model.privacyOne = @[@"《隐私1》",@"https://www.taobao.com/"];
//model.privacyTwo = @[@"《隐私2》",@"https://www.taobao.com/"];
model.privacyColors = @[UIColor.lightGrayColor, UIColor.orangeColor];
model.privacyAlignment = NSTextAlignmentCenter;
model.privacyFont = [UIFont fontWithName:@"PingFangSC-Regular" size:13.0];
model.privacyOperatorPreText = @"《";
model.privacyOperatorSufText = @"》";
//model.checkBoxIsHidden = NO;
model.checkBoxWH = 17.0;
```

```
model.changeBtnTitle = [[NSAttributedString alloc] initWithStrin
g:@"切换到其他方式" attributes:@{NSForegroundColorAttributeName : U
IColor.orangeColor,NSFontAttributeName : [UIFont systemFontOfSiz
e:18.0]}];
//model.changeBtnIsHidden = NO;
//model.prefersStatusBarHidden = NO;
model.preferredStatusBarStyle = UIStatusBarStyleLightContent;
//model.presentDirection = PNSPresentationDirectionBottom;

//授权页默认控件布局调整
//model.navBackButtonFrameBlock =
//model.navTitleFrameBlock =
model.navMoreViewFrameBlock = ^CGRect(CGSize screenSize, CGSize
superViewSize, CGRect frame) {
    CGFloat width = superViewSize.height;
    CGFloat height = width;
    return CGRectMake(superViewSize.width - 15 - width, 0, width
, height);
};
model.loginBtnFrameBlock = ^CGRect(CGSize screenSize, CGSize sup
erViewSize, CGRect frame) {
    if ([self isHorizontal:screenSize]) {
        frame.origin.y = 20;
        return frame;
    }
    return frame;
};
model.sloganFrameBlock = ^CGRect(CGSize screenSize, CGSize super
ViewSize, CGRect frame) {
    if ([self isHorizontal:screenSize]) {
        return CGRectZero; //横屏时模拟隐藏该控件
    } else {
        return CGRectMake(0, 140, superViewSize.width, frame.siz
e.height);
    }
};
model.numberFrameBlock = ^CGRect(CGSize screenSize, CGSize super
ViewSize, CGRect frame) {
    if ([self isHorizontal:screenSize]) {
        frame.origin.y = 140;
```

```
    }
    return frame;
};
model.loginBtnFrameBlock = ^CGRect(CGSize screenSize, CGSize superViewSize, CGRect frame) {
    if ([self isHorizontal:screenSize]) {
        frame.origin.y = 185;
    }
    return frame;
};
model.changeBtnFrameBlock = ^CGRect(CGSize screenSize, CGSize superViewSize, CGRect frame) {
    if ([self isHorizontal:screenSize]) {
        return CGRectZero; //横屏时模拟隐藏该控件
    } else {
        return CGRectMake(10, frame.origin.y, superViewSize.width - 20, 30);
    }
};
//model.privacyFrameBlock =

//添加自定义控件并对自定义控件进行布局
__block UIButton *customBtn = [UIButton buttonWithType:UIButtonTypeCustom];
[customBtn setTitle:@"这是一个自定义控件" forState:UIControlStateNormal];
[customBtn setBackgroundColor:UIColor.redColor];
customBtn.frame = CGRectMake(0, 0, 230, 40);
model.customViewBlock = ^(UIView * _Nonnull superCustomView) {
    [superCustomView addSubview:customBtn];
};
model.customViewLayoutBlock = ^(CGSize screenSize, CGRect contentViewFrame, CGRect navFrame, CGRect titleBarFrame, CGRect logoFrame, CGRect sloganFrame, CGRect numberFrame, CGRect loginFrame, CGRect changeBtnFrame, CGRect privacyFrame) {
    CGRect frame = customBtn.frame;
    frame.origin.x = (contentViewFrame.size.width - frame.size.width) * 0.5;
    frame.origin.y = CGRectGetMinY(privacyFrame) - frame.size.height - 20;
```

```
    frame.size.width = contentViewFrame.size.width - frame.origin.x * 2;
    customBtn.frame = frame;
};

// 横竖屏切换
model.supportedInterfaceOrientations = UIInterfaceOrientationMaskAllButUpsideDown;
// 仅支持竖屏
model.supportedInterfaceOrientations = UIInterfaceOrientationMaskPortrait;
// 仅支持横屏
model.supportedInterfaceOrientations = UIInterfaceOrientationMaskLandscape;
```

7.3 授权页弹窗模式，支持横竖屏切换

```
TXCustomModel *model = [[TXCustomModel alloc] init];

model.alertCornerRadiusArray = @[@10, @10, @10, @10];
//model.alertCloseItemIsHidden = YES;
model.alertTitleBarColor = UIColor.orangeColor;
model.alertTitle = [[NSAttributedString alloc] initWithString:@"一键登录（弹窗）" attributes:@{NSForegroundColorAttributeName : UIColor.whiteColor, NSFontAttributeName : [UIFont systemFontOfSize:20.0]}];
model.alertCloseImage = [UIImage imageNamed:@"icon_close_light"];

model.privacyNavColor = UIColor.orangeColor;
model.privacyNavBackImage = [UIImage imageNamed:@"icon_nav_back_light"];
model.privacyNavTitleFont = [UIFont systemFontOfSize:20.0];
model.privacyNavTitleColor = UIColor.whiteColor;

model.logoImage = [UIImage imageNamed:@"taobao"];
```

```
//model.logoIsHidden = NO;
//model.sloganIsHidden = NO;
model.sloganText = [[NSAttributedString alloc] initWithString:@"一键登录slogan文案" attributes:@{NSForegroundColorAttributeName : UIColor.orangeColor,NSFontAttributeName : [UIFont systemFontOfSize:16.0]}];
model.numberColor = UIColor.orangeColor;
model.numberFont = [UIFont systemFontOfSize:30.0];
model.loginBtnText = [[NSAttributedString alloc] initWithString:@"一键登录" attributes:@{NSForegroundColorAttributeName : UIColor.whiteColor, NSFontAttributeName : [UIFont systemFontOfSize:20.0]}];
//model.autoHideLoginLoading = NO;
//model.privacyOne = @[@"《隐私1》",@"https://www.taobao.com/"];
//model.privacyTwo = @[@"《隐私2》",@"https://www.taobao.com/"];
model.privacyColors = @[UIColor.lightGrayColor, UIColor.orangeColor];
model.privacyAlignment = NSTextAlignmentCenter;
model.privacyFont = [UIFont fontWithName:@"PingFangSC-Regular" size:13.0];
model.privacyOperatorPreText = @"《";
model.privacyOperatorSufText = @"》";
//model.checkBoxIsHidden = NO;
model.checkBoxWH = 17.0;
model.changeBtnTitle = [[NSAttributedString alloc] initWithString:@"切换到其他方式" attributes:@{NSForegroundColorAttributeName : UIColor.orangeColor,NSFontAttributeName : [UIFont systemFontOfSize:18.0]}];
//model.changeBtnIsHidden = NO;
//model.prefersStatusBarHidden = NO;
//model.preferredStatusBarStyle = UIStatusBarStyleDefault;
//model.presentDirection = PNSPresentationDirectionBottom;

CGFloat ratio = MAX(TX_SCREEN_WIDTH, TX_SCREEN_HEIGHT) / 667.0;

//实现该block，并且返回的frame的x或y大于0，则认为是弹窗谈起授权页
model.contentViewFrameBlock = ^CGRect(CGSize screenSize, CGSize contentSize, CGRect frame) {
    CGFloat alertX = 0;
    CGFloat alertY = 0;
```

```
    CGFloat alertWidth = 0;
    CGFloat alertHeight = 0;
    if ([self isHorizontal:screenSize]) {
        alertX = ratio * TX_Alert_Horizontal_Default_Left_Paddin
g;
        alertWidth = screenSize.width - alertX * 2;
        alertY = (screenSize.height - alertWidth * 0.5) * 0.5;
        alertHeight = screenSize.height - 2 * alertY;
    } else {
        alertX = TX_Alert_Default_Left_Padding * ratio;
        alertWidth = screenSize.width - alertX * 2;
        alertY = TX_Alert_Default_Top_Padding * ratio;
        alertHeight = screenSize.height - alertY * 2;
    }
    return CGRectMake(alertX, alertY, alertWidth, alertHeight);
};

//授权页默认控件布局调整
//model.alertTitleBarFrameBlock =
//model.alertTitleFrameBlock =
//model.alertCloseItemFrameBlock =
model.logoFrameBlock = ^CGRect(CGSize screenSize, CGSize superVi
ewSize, CGRect frame) {
    if ([self isHorizontal:screenSize]) {
        return CGRectZero; //横屏时模拟隐藏该控件
    } else {
        frame.origin.y = 10;
        return frame;
    }
};
model.sloganFrameBlock = ^CGRect(CGSize screenSize, CGSize super
ViewSize, CGRect frame) {
    if ([self isHorizontal:screenSize]) {
        return CGRectZero; //横屏时模拟隐藏该控件
    } else {
        frame.origin.y = 110;
        return frame;
    }
};
model.numberFrameBlock = ^CGRect(CGSize screenSize, CGSize super
```

```
ViewSize, CGRect frame) {
    if ([self isHorizontal:screenSize]) {
        frame.origin.y = 20;
        frame.origin.x = (superViewSize.width * 0.5 - frame.size.width) * 0.5 + 18.0;
    } else {
        frame.origin.y = 140;
    }
    return frame;
};
model.loginBtnFrameBlock = ^CGRect(CGSize screenSize, CGSize superViewSize, CGRect frame) {
    if ([self isHorizontal:screenSize]) {
        frame.origin.y = 60;
        frame.size.width = superViewSize.width * 0.5; //登录按钮最小宽度是其父视图的一半，再小就不生效了
    } else {
        frame.origin.y = 180;
    }
    return frame;
};
model.changeBtnFrameBlock = ^CGRect(CGSize screenSize, CGSize superViewSize, CGRect frame) {
    if ([self isHorizontal:screenSize]) {
        return CGRectZero; //横屏时模拟隐藏该控件
    } else {
        return CGRectMake(10, 240, superViewSize.width - 20, 30);
    }
};
//model.privacyFrameBlock =

//添加自定义控件并对自定义控件进行布局
__block UIButton *customBtn = [UIButton buttonWithType:UIButtonTypeCustom];
[customBtn setTitle:@"这是一个自定义控件" forState:UIControlStateNormal];
[customBtn setBackgroundColor:UIColor.redColor];
model.customViewBlock = ^(UIView * _Nonnull superCustomView) {
    [superCustomView addSubview:customBtn];
```

```
};
model.customViewLayoutBlock = ^(CGSize screenSize, CGRect contentViewFrame, CGRect navFrame, CGRect titleBarFrame, CGRect logoFrame, CGRect sloganFrame, CGRect numberFrame, CGRect loginFrame, CGRect changeBtnFrame, CGRect privacyFrame) {
    CGFloat padding = 15;
    CGFloat x = 0;
    CGFloat y = 0;
    CGFloat width = 0;
    CGFloat height = 0;
    if ([self isHorizontal:screenSize]) {
        x = CGRectGetMaxX(loginFrame) + padding;
        y = padding;
        width = contentViewFrame.size.width - x - padding;
        height = CGRectGetMinY(privacyFrame) - y - padding;
    } else {
        x = padding;
        y = MAX(CGRectGetMaxY(changeBtnFrame), CGRectGetMaxY(loginFrame)) + padding;
        width = contentViewFrame.size.width - 2 * x;
        height = CGRectGetMinY(privacyFrame) - y - padding;
    }
    customBtn.frame = CGRectMake(x, y, width, height);
};

// 横竖屏切换
model.supportedInterfaceOrientations = UIInterfaceOrientationMaskAllButUpsideDown;
// 仅支持竖屏
model.supportedInterfaceOrientations = UIInterfaceOrientationMaskPortrait;
// 仅支持横屏
model.supportedInterfaceOrientations = UIInterfaceOrientationMaskLandscape;
```

7.4 本机号码校验用例

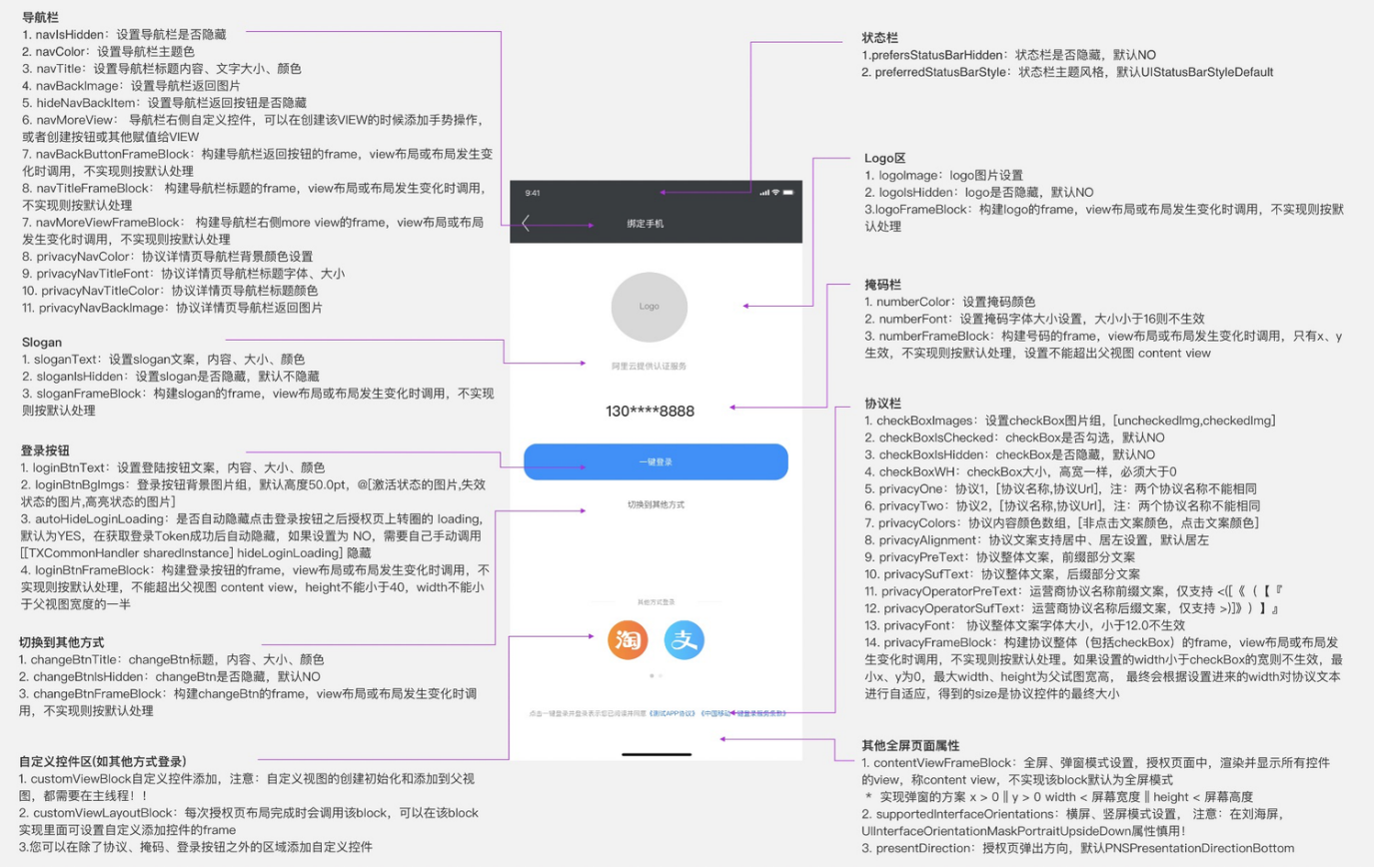
```
//1. 设置SDK参数，app生命周期内调用一次即可
```

```
NSString *info = @"客户的秘钥串";
__weak typeof(self) weakSelf = self;
//设置SDK参数，app生命周期内调用一次即可
[[TXCommonHandler sharedInstance] setAuthSDKInfo:info complete:^(NSDictionary * _Nonnull resultDic) {
    [weakSelf showResult:resultDic];
}];


//2. 检测当前环境是否支持本级号码校验，支不支持提前知道
__block BOOL support = YES;
[[TXCommonHandler sharedInstance] checkEnvAvailableWithAuthType:PNSAuthTypeVerifyToken complete:^(NSDictionary * _Nullable resultDic) {
    support = [PNSCodeSuccess isEqualToString:[resultDic objectForKey:@"resultCode"]];
}];

if (self.tf_phoneNumber.text.length == 0) {
    [ProgressHUD showError:@"请先输入手机号码"];
    return;
}

float timeout = self.tf_timeout.text.floatValue;
[ProgressHUD show:@"请稍等..." Interaction:YES];
__weak typeof(self) weakSelf = self;

//3. 获取 VerifyToken
[[TXCommonHandler sharedInstance] getVerifyTokenWithTimeout:timeout complete:^(NSDictionary * _Nonnull resultDic) {
    if ([PNSCodeSuccess isEqualToString:[resultDic objectForKey:@"resultCode"]] == NO) {
        [ProgressHUD showError:@"获取 VerifyToken失败"];
        [weakSelf showResult:resultDic];
        return ;
    }
    //3. 去服务器验证 VerifyToken
    [weakSelf showResult:resultDic];
    NSString *token = [resultDic objectForKey:@"token"];
    //注: 这里请求是通过自己服务器, 下面仅供参考
```

```objc
    [PNSVerifyTopRequest requstVerifyWithNumber:weakSelf.tf_phoneNumber.text token:token complete:^(BOOL isSuccess, NSString * _Nonnull msg, NSDictionary * _Nonnull data) {
        NSDictionary *module = data[@"module"];
        NSString *verify_result = [module objectForKey:@"verify_result"];
        if ([verify_result isEqualToString:@"PASS"]) {
            [ProgressHUD showSuccess:@"本机号码校验成功"];
        } else {
            [ProgressHUD showSuccess:@"本机号码校验失败"];
        }
        [weakSelf showResult:data];
    }];
}];
```

8. 一键登录授权页面说明

8.1 全屏授权页面设计规范（支持横竖屏，以竖屏示意）



8.2 弹窗授权页面设计规范（支持横竖屏，以竖屏示意）

9. iOS常见问题

- 初始化接口`checkEnvAvailableWithComplete`一直返回`NO`？
  - 排查顺序：1、手机sim卡是否被激活或欠费；2、手机设备蜂窝数据是否开启；3、App的网络权限是否开启；4、设备是否有代理；5、云控制台上是否创建了方案号；6、创建方案号中的`bundleId`一定要与项目中使用的保持一致；7，设备时间设置是否非标准，不能修改手机时间戳提前或延期；8，是否调用了`setAuthSDKInfo`接口；9，最后再提供`bundleId`给我们进行查询日志；
- 移动卡出现crash【[UAReachability reachableType]: unrecognized selector sent to instance】
  - 在主工程中Project -> Edit Active Target -> Build Setting -> Other Linker Flags中添加`-all_load`标记，前提也要添加`-ObjC`，即**两者都要添加**；
- 登录token存在失败吗?
  - 肯定存在的，如果是偶现，可以理解，因为比如网络波动导致网关断开、网络不可用、供应商服务端异常，业务方服务端异常等因素有关；如果是持续出现，一般需要运营商协助排查；
- 一键登录服务一般只4G或者4G+Wifi情况下，如果是3G、2G会怎么样?
  - 中国移动支持2G/3G/4G、中国联通支持3G/4G、中国电信支持4G，但2G和3G网络下接口请求失败或超时概率稍高。
- 经常超时?
  - 首先确认sim卡不欠费，再确认手机网络是否正常，通过Safari打开某个网址验证为准。

- 其次确认超时时间是否设置正确，单位是s。
- 2G和3G网络下接口请求失败或超时概率稍高。
- 切换网络时，设置网络不稳定，会较大概率出现超时。

- setSDKAuthSDKInfo的秘钥如何获取?
  - 请登录阿里云控制台，进入认证方案管理，点击”秘钥“进行复制，建议维护在APP服务端。此秘钥不是阿里云的AccessKey,AccessSecret!