ALIBABA CLOUD

阿里云

号码认证服务

开发指南

文档版本：20220307

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.服务端API返回码

本文为您介绍号码认证服务的服务端API返回码。

返回码	说明	解决方案
OK	请求成功	无。
isv.VERIFY_SCHEME_NOT_EXIST	认证方案不存在	请确认是否已经在控制台创建方案，并核对创建时的包名包签名或BundleID与实际使用是否一致。
isv.VERIFY_SCHEME_CONFLICT	认证方案冲突	请确认同账号下是否有相同包名包签名或BundleID的方案。
isv.PACK_SIGN_CONFLICT	包签名冲突	已存在相同包名和签名的认证方案。
isv.BUNDLE_ID_CONFLICT	BundleID冲突	已存在相同BundleID的认证方案。
isv.RAM_PERMISSION_DENY	没有访问权限	请确认阿里云RAM账号是否授权调用号码认证服务。
isv.SCENE_QUERY_FAIL	场景查询失败	创建工单并提供详细的错误日志信息。
isv.TOKEN_INVALID	令牌无效	AccessToken或AccessCode已过期，请重新获取。
isv.TOKEN_UNAUTHORIZED_USED	越权使用的Token	使用本机号码校验的Token去调用了一键登录的接口。
isv.PRODUCT_UNSUBSCRIBE	产品未开通	请进入号码认证控制台开通服务。
isv.PRODUCT_UN_SUBSCRIPT	未开通云通信产品的阿里云客户	请进入号码认证控制台开通服务。
isv.PARAMETERS_MISMATCH	参数不匹配	请检查参数，输入手机号码和当前流量卡运营商归属不一致。
isv.OUT_OF_SERVICE	业务停机	请检查阿里云账户余额。
isv.MOBILE_NUMBER_ILLEGAL	号码格式非法	请检查号码格式。
isv.INVALID_PARAMETERS	非法参数	请求参数不合法。
isv.INVALID_PARAMETERS	请求的参数VendorKey不合法	传入的参数中的VendorKey有误，请检查参数是否正确。
isv.INVALID_PARAMETERS	无法识别运营商	查询场景详情时传入的参数有误，请检查Accesscode是否正确。
isv.INVALID_PARAMETERS	Accesscode参数不能为空	传入的参数中的Accesscode为空或错误，请检查Accesscode是否正确。

返回码	说明	解决方案
isv.INVALID_PARAMETERS	os_type can not be blank	传入的参数中的操作系统类型为空，请检查os_type是否正确。
isv.INVALID_PARAMETERS	os_type is illegal	传入的参数中的操作系统类型非法，请检查os_type是否正确。
isv.INVALID_PARAMETERS	AccessKey can not be blank	传入的参数中的AccessKey为空，请检查AccessKey是否正确。
isv.INVALID_PARAMETERS	AccessParam can not be blank	传入的参数中的AccessParam为空，请检查AccessParam是否正确。
isv.INVALID_APP	此次App的请求非法	请确认创建方案号的阿里云账号和服务端调用的阿里云账号是否一致。
isv.FORBIDDEN_ACTION	无权操作	创建工单。
isv.CSRF_CHECK_FAILED	csrf Token 检查失败	失效的或者非法的Token参数。
isv.ACCOUNT_NOT_EXISTS	账户不存在	请确认账号是否正确。
isv.ACCOUNT_ABNORMAL	账户异常	请确认账号状态是否正常。
isv.ACCESS_CODE_ILLEGAL	AccessCode参数非法	请确认Accesscode参数是否与客户端返回的一致。
isp.UNKNOWN	未知错误	创建工单，提供阿里云返回的RequestID。
isp.SYSTEM_ERROR	系统错误	创建工单，提供阿里云返回的RequestID。
isp.RES_OWNER_ID_UNKNOWN	找不到资源归属的阿里云ID	请检查阿里云账号是否正确。
isp.RAM_PERMISSION_DENY	RAM权限DENY	请确认阿里云RAM账号是否已授权调用号码认证服务。
isp.QPS_LIMIT	QPS受限	调用该接口的请求过多。
isp.OPERATOR_LIMIT	该SDK版本受运营商限制	目前只支持电信移动和联通三大供应商。
InvalidParameter.MissingCustomerId	参数CustomerId缺失	账号有误，请检查账号是否正确。
InvalidParameter.MissingPackageName	参数PackageName缺失	当os_type=Android时，包名不能为空。
InvalidParameter.MissingSignName	参数SignName缺失	当os_type=Android时，签名不能为空。

返回码	说明	解决方案
InvalidParameter.MissingBundleId	参数BundleID缺失	当os_type=BundleID时，BundleID不能为空。
InvalidParameter.SmsTemplateCode	参数SmsTemplateCode不合法	请确认同账号下，是否有对应的短信模板Code。
InvalidParameter.OsType	参数os_type不合法	请确认os_type是否为Android或iOS。
InvalidParameter.MisMatchWithSceneCode	参数SceneCode不匹配	请确认同账号下方案号对应的包名、签名或者BundleID是否一致。
InvalidPhoneNumber.Check	校验PhoneNumber错误	请检查手机号格式是否正确。
InvalidBizToken.Expired	BizToken过期	重新获取BizToken。
LimitExceeded.SmsCode	SmsCode超出限制	同一个号码对同一个签名和模板生成验证码超出限制，可在控制台**通用设置**中进行多次修改。
InvalidBizToken.Check	校验BizToken失败	请检查BizToken是否正确。
EntityNotExist.SceneCode	请检查同账号下是否存在方案号	请检查同账号下是否存在方案号。
EntityNotExist.SmsTemplateCode	数据SmsTemplateCode不存在	请检查同账号下是否存在审核通过的短信模板Code。
EntityNotExist.SmsSignName	数据SmsSignName不存在	请检查同账号下是否存在审核通过的短信签名，并且已绑定到此方案号上。
EncryptError.BizToken	加密BizToken错误	生成BizToken失败。
QueryFail.SceneCode	查询SceneCode失败	检查方案号是否正确。
CreateFail.StsToken	创建StsToken错误	创建临时StsToken失败。
CreateFail.SmsCode	创建SmsCode错误	生成短信验证码失败。
CreateFail.VerifyKey	创建VerifyKey错误	验证短信验证码失败。

2.SDK参考

2.1. SDK发布历史

本文为您介绍号码认证服务客户端SDK发布历史。

目前号码认证服务中各功能使用的SDK最新版本如下所示：

- 一键登录和本机号码校验：
  - Android：V2.12.3.4
  - iOS：V2.12.3.3
- 短信认证：

  Android或iOS：V1.0.1
- H5本机号码校验：V1.1.2

一键登录和本机号码校验SDK发布历史

- Android：

版本号	发布时间	更新内容
V2.12.3.4	2021.11.16	◦ 升级中国电信SDK版本。 ◦ 优化NetworkCallback泄漏问题以及线程泄漏问题。
V2.12.3	2021.09.06	◦ 修复没有设置authuiconfig导致的动画空指针。 ◦ 去掉对蜂窝网络开启的拦截判断。 ◦ 优化接口超时时间设置，提高取号成功率。 ◦ 运营商版本升级。 ◦ 优化网络问题的错误码返回。 ◦ 增加接口setHiddenLoading控制SDK自带的loading可以隐藏。 ◦ 增加setLoadingBackgroundPath和setLoadingBackgroundDrawable设置loading背景。
V2.12.1.4	2021.08.07	优化IP信息采集频率。

版本号	发布时间	更新内容
V2.12.1.2	2021.06.07	◦ 优化checkEnvAvailable接口逻辑，降低误拦截，提高取号成功率。 ◦ 对于包名和签名部分逻辑进行加固，提高SDK安全性。 ◦ 优化超时和网络异常的错误码处理。
V2.12.1	2021.04.06	◦ 新增接口支持授权页底部各协议之间可以修改文案；使用setPrivacyOperatorIndex方法调整运营商协议和客户自定义协议之间的顺序，运营商协议指定显示顺序，默认值为0，在第1个协议位置显示，最大值为3，即在第4个协议的位置显示。 ◦ 新增接口支持设置字体大小单位为dp，解决改变系统字体导致授权页错乱问题。 ◦ 新增接口支持设置授权页相关页面图片drawable方式，从而支持背景图片的SDK更新。 ◦ 新增接口设置协议action，可以通过action跳转客户自定义协议展示页。 ◦ 修复底部导航栏沉浸式问题。 ◦ 修复获取networktype出现异常问题。 ◦ 修复Demo某些界面拉起登录页之后，切换系统字体、深色模式出现的页面重复问题。 ◦ 优化移动获取Token的逻辑，避免103505报错。 ◦ 修复setWebViewStatusBarColor设置协议页状态栏颜色方法无效问题。 ◦ 去除部分废弃接口，去除SDK内部的write_external_storage权限。

版本号	发布时间	更新内容
V2.12.0.1	2021.02.02	○ 增加协议页面中JS支持开关，可选择关闭或开启。 ○ 对线上crash增加保护功能。 ○ 增加crash组件开关，修复crash组件版本冲突问题。 ○ 修复appname获取失败时，出现解析密钥失败的问题。
V2.12.0	2021.01.20	○ 更新移动、电信、联通运营商的SDK。 ○ SDK组件化，包含4个aar组件库。 ○ 解决域名中间人劫持安全漏洞。 ○ 自定义协议H5页面禁用js，避免绕过file协议的同源检查导致敏感信息泄露的风险。
V2.11.1.1	2020.11.06	○ 密钥规则更新，建议升级Android SDK并使用最新密钥。 ○ 少量bug修复。
V2.11.0	2020.10.22	○ 新增活体认证功能。 ○ 版本： ■ 增强版：包含活体认证和号码认证功能。 ■ 标准版：仅包含号码认证功能。 ○ 去掉对fastjson的依赖。
V2.10.1	2020.08.28	○ 新增ActivityResultListener回调，将授权页onActivityResult的数据全部回抛给客户，支持第三方登录。 ○ 客户不再需要配置SDK混淆也能够正常使用。 ○ 对于神策、aspectjx等框架不再需要exclude。 ○ 内部逻辑优化，提高稳定性。

版本号	发布时间	更新内容
V2.10.0	2020.08.05	○ 增加掩码和Token缓存优化逻辑。 ○ 提高接口调用成功率和速度。新增认证加速功能。 ○ 新增accelerateVerify加速认证接口。 ○ 更新联通运营商SDK。 ○ 修改接口回调都回调到主线程。 ○ 调整资源混淆配置。
V2.8.4.1	2020.07.07	○ 修复空指针异常问题。 ○ 修复setAuthSDKInfo不兼容老用户问题。
V2.8.3	2020.06.08	○ 升级移动SDK。 ○ 增加一键登录按钮多次点击防护功能。 ○ 增加设置底部导航栏颜色接口。 ○ 兼容娜迦加固。
V2.8.0	2020.03.26	○ 优化checkEnvAvailable接口。 ○ 优化业务逻辑流程。
V2.7.1	2019.12.20	○ 增加弹层蒙层透明度设置接口setDialogAlpha。 ○ 增加设置第三条隐私条款接口setAppPrivacyThree。 ○ 修改设置颜色白色不生效的问题。 ○ 增强SDK稳定性。
V2.7.0.2	2019.12.05	○ 适配AndroidX工具包。 ○ 解决某些用户在mutidex配置开启时，唤起移动授权页会crash的问题。
V2.7.0.1	2019.11.26	修复协议链接重定向时会跳转到浏览器的问题。

版本号	发布时间	更新内容
V2.7.0	2019.11.19	○ 授权页支持横屏模式。 ○ 授权页支持横竖屏弹窗模式。 ○ 适配Android Q。 ○ READ_PHONE_STATE改为可选权限，无需强制声明。 ○ 更多UI特性支持。 ○ 双卡切换、插拔卡、弱网等极限环境下的功能优化。 ○ 错误码统一。

- iOS：

版本号	发布时间	更新内容
V2.12.3.3	2022.01.20	修复联通SDK多线程crash问题。
V2.12.3	2021.09.06	○ 适配iOS 15。 ○ 运营商SDK版本升级。您可咨询号码认证服务的技术支持，只需提供您当前集成的一键登录和本机号码校验SDK的版本号，即可查询集成的运营商SDK版本。 ○ 新增设置授权页背景色、背景图片、背景填充模式属性。 ○ 修复CTTelephonyNetworkInfo初始化频繁造成偶发crash问题。 ○ 优化接口超时设置。
V2.12.1.3	2021.06.07	○ check接口内部环境检查优化，降低误拦截，提高取号成功率。 ○ 功能接口内部的环境检查优化。 ○ 替换SDK内部部分关键字，避免跟苹果私有API名字一样，导致审核被拒问题。

版本号	发布时间	更新内容
V2.12.1	2021.04.06	○ 新增授权页push到其他页面后导航栏是否显示。 ○ 新增运营商协议可指定显示顺序。 ○ 新增协议之间字符可自定义。 ○ 新增授权页动画自定义。 ○ 新增协议详情页展示容器自定义。 ○ 新增弹窗模式下支持全屏大小且全透明背景效果。 ○ 去掉业务废弃接口。 ○ 修复授权页自定义添加的CustomView不能正常释放问题。 ○ 适配xcode 12打包。
V2.12.0	2021.01.20	○ 升级运营商SDK，修复联通5G下网络双开取号失败问题。 ○ 动态库去除模拟器架构，解决集成时打包失败问题。 ○ 关闭bitcode支持。
V2.11.2	2020.12.21	○ 修复从V2.10.x版本升级到V2.11.x，会出现在iOS 10及以下系统的crash问题。 ○ 最低支持系统，从iOS 8.0改为iOS 9.0。
V2.11.1	2020.11.06	○ 新增的设置授权页展示和动画消失时间配置项。 ○ 新增协议栏check box图片填充大小，扩展check box响应区域。 ○ 更改登录按钮高度最小限制调整为20 pt。
V2.11.0	2020.10.22	○ 新增活体认证功能。 ○ 版本： ■ 增强版：包含活体认证和号码认证功能。 ■ 标准版：仅包含号码认证功能。 ○ 一键登录和本机号码校验缓存逻辑优化。

版本号	发布时间	更新内容
V2.10.1	2020.09.02	◦ 新增crash防护组件，提高SDK稳定性。 ◦ 修复iOS 13上状态栏颜色设置不生效问题。 ◦ 修复设置授权页动画方向不生效问题。 ◦ 适配iOS 14。 ◦ 更新移动运营商SDK，修复ca证书解析错误的问题。 ◦ 新加debugLoginUI接口，用于在模拟器或真机上拉起调试授权页，方便开发进行UI适配。 ◦ 修复点击授权页协议富文本时，回调结果里面协议对应key写错问题。
V2.10.0	2020.08.05	◦ 更新联通SDK，接口调整优化。 ◦ 增加线程安全代码，优化接口耗时。 ◦ 优化取号、拉起授权页面、获取Token速度。 ◦ 新增accelerateVerifyWithTimeout接口，提高获取本机号码校验Token速度。
V2.8.4	2020.07.01	◦ 修复TXCustomModel内存泄漏问题。 ◦ 修复弹框消失时，有白条闪过问题。
V2.8.1	2020.04.16	◦ 更新移动SDK，修复keychain卡顿、越狱检测crash和IPv6网络解析问题。 ◦ 增加异步线程同步队列管理，规避移动SDK多线程调用crash问题。 ◦ 更新电信SDK，修复firebase库冲突问题。 ◦ 优化业务逻辑流程。
V2.7.2	2020.03.04	◦ 修复移动授权页无生命周期和导航控制器问题。 ◦ 增加联通电信授权页生命周期和导航控制器。

版本号	发布时间	更新内容
V2.7.1	2019.12.20	◦ 修复因工程无ATAuthSDK.bundle造成初始化的闪退问题。 ◦ 修复移动授权页面点击checkbox或协议名称未返回一些字段信息问题。 ◦ 修复授权页未释放导致内存泄漏问题。 ◦ 修复登录按钮不可点击时背景会变色问题。 ◦ 修复logo图片被其他自定义控件可遮挡问题。 ◦ 授权页隐私兰支持三个自定义协议。 ◦ 授权页导航栏右侧返回及左侧更多按钮限制约束区域放大。 ◦ 获取登录Token之后统一清除掩码信息。 ◦ 全面对容易因异常数据造成闪退的代码进行防护优化，提升SDK健壮性及稳定性。 ◦ check接口增加了App网络权限是否开通检测。 ◦ 修复偶现iOS 13.x beta系统上因找不到dataServiceIdentifier方法闪退问题。
V2.7.0	2019.11.19	◦ 授权页竖屏全屏模式UI属性扩展，新增点击事件回抛。 ◦ 授权页支持横竖屏弹窗模式。 ◦ 授权页支持横屏全屏模式。 ◦ 支持横竖屏切换。 ◦ 支持跨运营商SIM卡切换检测。 ◦ 联通SDK更新，修复RSA为空闪退、内存泄漏、socket关闭闪退容错、一定条件下必现超时等问题。 ◦ 双卡切换、插拔卡、弱网等极限环境下的功能优化。 ◦ 修复部分iOS 12.x.x系统上无SIM卡问题。 ◦ 修复电信卡偶现-10002异常。 ◦ 错误码统一。

短信认证SDK发布历史

Android SDK和iOS SDK发布历史均如下所示：

版本号	发布时间	更新内容
V1.0.1	2020.11.16	修复V1.0.0中一些日志错误的问题和加密逻辑。
V1.0.0	2021.09.16	发布短信验证码功能。

H5端JSSDK发布历史

版本号	发布时间	更新内容
V1.1.2	2021.08.07	优化鉴权接口，手机号码参数修改为非必选。
V1.1.1	2021.06.07	• 运营商SDK改为本地发布，避免因为运营商在线修改代码导致影响客户业务。 • 修复iOS系统抖音的兼容性bug（部分安卓5.1机型不支持抖音快手移动手机号）。
V1.1.0	2021.04.19	• 移动SDK接入新版本（接口改为HTTPS接口）。 • 鉴权接口错误码细化。 • 代码规范及优化。
V1.0.3	2021.02.07	• 日志埋点完善。 • 增加超时接口。
V1.0.2	2020.09.26	• 优化H5体验Demo。 • 提高H5本机号码校验速度。 • 更新联通JSSDK版本。
V1.0.1	2020.08.01	• 优化H5本机号码流程。 • 增加H5体验Demo。
V1.0.0	2020.05.01	全新发布H5本机号码校验功能。

2.2. SDK与密钥说明

本文为您介绍了新旧密钥及版本的划分、密钥逻辑等信息。

新旧密钥及版本的划分

新旧密钥和新旧SDK版本的划分如下：

- 新密钥与旧密钥的划分：新密钥是指采用新结构的密钥，同理，旧密钥是指采用旧结构的密钥。新旧密钥划分的时间界限为2020-10-20，在该时间点之前创建的方案号称为旧方案号，该时间点之后创建的方案号称为新方案号。
- 新版SDK与旧版SDK的划分：只有Android端SDK区分新旧版本。一键登录和本机号码校验的新版SDK是指版本在v2.8.4和v9.4.0及以上的SDK，其余功能的SDK版本不涉及此问题。

密钥逻辑

若您开启一个新功能，旧认证方案的密钥会自动更新一次，SDK也需要更换成最新的密钥，否则无法使用此功能。密钥的处理逻辑为：

- 对于旧方案号，旧方案的密钥信息没有携带认证相关的信息，例如您想使用活体认证功能，则需要更换为最新版本的SDK，并且要更换为最新的密钥。
- 对于新方案号，如果想用号码认证服务的功能，iOS可正常使用，Android SDK需要保证为新版本，否则会报密钥解析错误600017。

> **注意** 由于低版本的Android SDK不兼容新方案号，若客户的方案号存在于服务端，需要由App主动拉取，在服务端更新密钥时需要对App或者SDK版本号做判断区分。如果密钥信息是保存在App客户端，则无需考虑该问题。

密钥配对

SDK	配对详情	功能是否可正常使用
Android SDK	旧版SDK与旧密钥进行配对	可正常使用。
	新版SDK与新密钥进行配对	
	旧版SDK与新密钥进行配对	功能不可使用。需保证SDK版本为新版，否则会报密钥解析错误600017，建议升级到最新版本的SDK。
	新版SDK与旧密钥进行配对	不可使用。
iOS SDK（不区分新旧版）	SDK与新密钥进行配对	可正常使用。
	SDK与旧密钥进行配对	不可使用。

2.3. 一键登录和本机号码校验

2.3.1. Demo体验和样例

本文提供号码认证服务（标准版）的Demo体验和一键登录授权页样例。

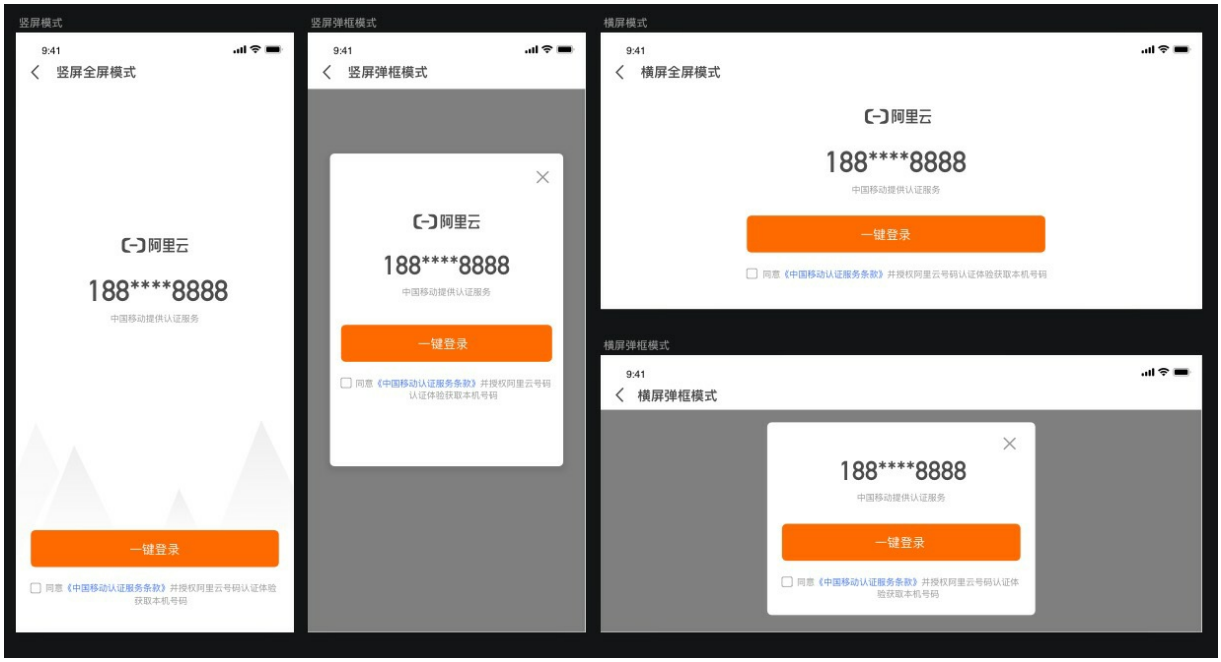
号码认证（标准版）Demo体验

请扫描下方二维码，下载标准版号码认证Demo压缩包（V1.0.0），体验一键登录/注册和本机号码校验功能。

最新版Demo压缩包请登录号码认证服务控制台下载。

一键登录授权页样例

一键登录授权页面支持竖屏全屏、竖屏弹窗、横屏全屏、横屏弹窗。样例仅供示意，您可以根据自己的需求进行UI设计。授权页设计和配置详情，请根据您的客户端参见Android客户端接入、iOS客户端接入或UniApp客户端接入。



2.3.2. 接入概述

本文为您介绍了一键登录和本机号码校验功能的交互流程。开发者需要在App中集成号码认证服务客户端SDK，并在服务端完成API对接。

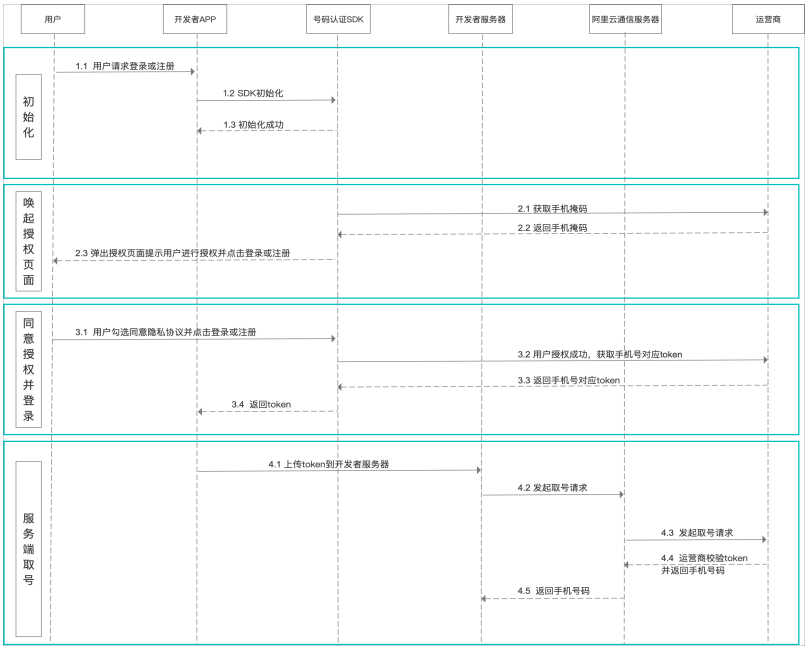
使用场景

号码认证服务包含一键登录和本机号码校验两个不同的功能，使用场景不一样，无需一起使用。

- 本机号码校验使用场景：请您输入手机号码，通过SDK获取Token，服务端携带输入的手机号码和Token到运营商网关进行校验，返回的结果显示您当前上网使用的号码与校验的号码是否一致。
- 一键登录使用场景：您无需输入手机号码，SDK会弹出授权页，确认授权后，SDK获取Token，服务端携带Token到运营商网关获取您当前上网使用的号码，并返回给App服务端。

一键登录的交互流程

一键登录的系统交互流程主要分为四个步骤：初始化、唤起授权页面、同意授权并登录、服务端取号。



1. 初始化
   i. 访问App页面。
   ii. 调用getVersion接口获取SDK版本号，并进行终端网络环境判断。SDK详情请参见：Android客户端接入或iOS客户端接入。
2. 唤起授权页面
   i. 初始化成功后，调用getLoginToken接口唤起授权页面。
   ii. SDK会先请求脱敏号码。
   iii. 请求成功后会在授权页面展示脱敏号码及运营商协议供终端用户确认。

> 注意
> - 一键登录或注册需用户确认授权方可使用，且登录按钮文字描述必须包含“登录”、注册按钮文字描述必须包含“注册”等文字，不得诱导用户授权，开发者不得通过任何技术手段跳过或模拟此步骤，否则我方有权停止服务并追究相关法律责任。
> - 对于接入移动认证SDK并上线的应用，阿里云对上线的应用授权页面做审查，若出现未按要求弹出或设计授权页面的，将停止应用的一键登录或注册服务。
> - 为减少授权页唤起的等待时间，可预先判断用户是否需要进行登录或注册，如果需要可调用预取号接口，调用后会在终端侧缓存预取号信息，供后续流程使用。
> - 请注意预取号（不收费）的频率，阿里云会在预取号与实际取号的比例异常时，停止提供相应的服务。

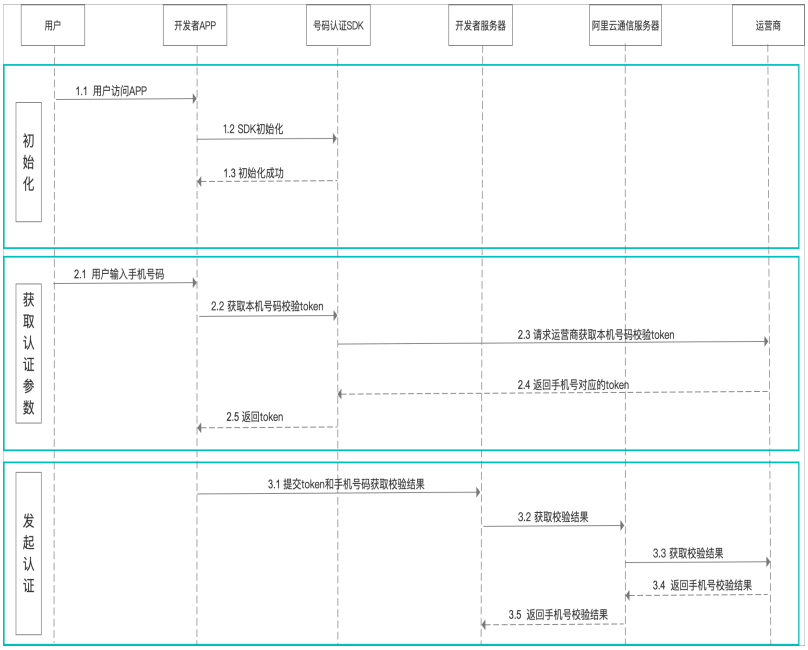
3. 同意授权并登录
   i. 确认授权页面的内容，并同意相关协议。
   ii. 单击授权页面的登录或注册按钮，SDK会发起本次取号的Token获取。
   iii. 获取成功后将Token返回给开发者App。

4. 服务端取号
   i. 开发者App将获得的取号Token传递至开发者服务器端。
   ii. 开发者服务端携带Token调用号码认证服务端GetMobile接口。
   iii. 号码认证服务端取得号码后将号码返回给开发者服务端。

> 注意 由于运营商限制，客户端无法获取到手机号、手机掩码。

本机号码校验

号码认证的系统交互流程主要分为三个步骤：初始化、获取认证参数、发起认证。



1. 初始化
   i. 用户访问App页面。
   ii. 调用getVersion接口获取SDK版本号、初始化。SDK详情请参见：Android客户端接入或iOS客户端接入。

> 注意 对于Android系统，当用户授权允许读取SIM卡数据时，`public InitResult init()`方法会同时返回从SIM卡读取到的手机号码，帮助用户提前填写手机号码。如果用户未授权或其它原因，则该函数仅返回是否支持号码认证。iOS系统不支持从SIM卡读取手机号码进行助填。

2. 获取认证参数
   i. 用户输入认证手机号码。
   ii. 调用getInstance接口获取认证相关参数。
   iii. 返回认证参数。
3. 发起认证
   i. 开发者App向其服务端发起认证请求。
   ii. 调用认证接口。开发者服务端调用本机号码校验认证VerifyMobile接口获取认证结果，判断用户输入的手机号码与用户终端当前访问网络的手机号码是否一致。

2.3.3. 客户端接入

2.3.3.1. Android客户端接入

本文为您介绍了Android客户端如何接入一键登录和本机号码校验功能。

前提条件

- 确保您已开通了号码认证服务，并成功创建了对应的认证方案。详情请参见一键登录和本机号码校验使用流程。
- 确保您的终端设备已开启SIM卡的移动数据（支持中国联通、中国移动的3G网络，但接口耗时会增加）。
- 支持版本：Android 9.0版本以上支持HTTP配置、com.android.support:support-v4版本高于25.4.0或者vcom.android.support:appcompat-v7版本高于25.4.0。

步骤一：搭建开发环境

1. 下载并解压Android SDK。登录号码认证产品控制台，在**标准版**选项卡，下载并解压Android SDK。
2. 将已解压的SDK包中后缀为aar的文件复制至工程的libs目录下。
3. 在App工程src包中main AndroidManifest.xml增加Activity声明 。

```
 <!--联通电信授权页-->
 <!--如果不需要使用窗口模式，不要使用authsdk_activity_dialog主题，会出现异常动画-->
 <!--如果需要使用authsdk_activity_dialog主题，则screenOrientation一定不能指定明确的方向，
        比如portrait、sensorPortrait，在8.0的系统上不允许窗口模式指定orientation，会发生crash
，需要指定为behind，
        然后在授权页的前一个页面指定具体的orientation-->
<activity
    android:name="com.mobile.auth.gatewayauth.LoginAuthActivity"
    android:configChanges="orientation|keyboardHidden|screenSize"
    android:exported="false"
    android:theme="@style/authsdk_activity_dialog"          使用弹窗模式必须添加！！！
    android:launchMode="singleTop" />
<!--协议页面webview-->
<activity
    android:name="com.mobile.auth.gatewayauth.activity.AuthWebVeiwActivity"
    android:configChanges="orientation|keyboardHidden|screenSize"
    android:exported="false"
    android:launchMode="singleTop"
    android:screenOrientation="behind" />
 <!--移动授权页-->
 <activity
    android:name="com.cmic.sso.sdk.activity.LoginAuthActivity"
    android:configChanges="orientation|keyboardHidden|screenSize"
    android:exported="false"
    android:launchMode="singleTop" />
```

4. 集成组件。
   - crashshiled-xx-release.aar：SDK内部的crash防护和收集库，可以减少SDK的崩溃率，在SDK发生问题时不会影响接入者的App。

> **注意** 由于uc的crash收集原理与市面常见crash收集库的原理类似，如果App自身原本就有crash收集能力，建议将自身的crash库放在前面加载，uc的crash不会替换原有的crash收集能力，而是会形成链式传递，且只会收集跟号码认证SDK相关的crash，其他的crash信息不会收集。

- logger-xx-release.aar：SDK内部用于收集日志信息和监控告警的。
- main-xx-release.aar：SDK内部的一些核心工具类，比如线程池、json解析等。

以上三个组件都需要集成，具体的集成方法是：将3个arr包放在App工程下的libs目录下，在bulid.gradle中写入以下代码。

```
implementation(name:'auth_number_product-2.12.3-log-online-standard-release', ext:'aar'
)
implementation(name:'crashshield-release', ext:'aar')
implementation(name:'main-release', ext:'aar')
implementation(name:'logger-release', ext:'aar')
```

5. （可选）若您需要开启资源混淆，则需要在相应工程的proguard-rules.pro文件中写入以下代码。

```
"R.drawable.authsdk*",
"R.layout.authsdk*",
"R.anim.authsdk*",
"R.id.authsdk*",
"R.string.authsdk*",
"R.style.authsdk*",
```

6. （可选）域名配置。

   若您的终端设备使用的是中国联通SIM卡的5G移动数据，可能会导致使用一键登录功能获取本机号码失败。您可在networkSecurityConfig清单文件中配置联通域名 `enrichgw.10010.com` 解决此问题。

步骤二：运行Demo

登录号码认证产品控制台，在**概览**页面下载**开发者接入Demo**，再打开开发工具直接运行Demo工程。

步骤三：完成AndroidManifest.xml设置

在App AndroidManifest.xml文件中添加必要的权限支持：

```
<uses-permission android:name="android.permission.INTERNET" /> <!-- 网络访问 -->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" /> <!-- 检查wifi网络状态
-->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" /> <!-- 检查网络状态
-->
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" /> <!-- 切换网络通道
-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/> <!-- 本地信息缓存
-->
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" /> <!-- 开关Wi-Fi状态，
解决国内机型移动网络权限问题需要 -->
```

HTTP配置。在App AndroidManifest.xml里，给Application节点增加usesCleartextTraffic配置。

```
<application
       android:name=".DemoApplication"
       android:icon="@drawable/ic_launcher"
       android:label="@string/app_name"
       android:supportsRtl="true"
       android:theme="@style/AppTheme"
       android:usesCleartextTraffic="true">
```

目前中国移动提供的个别接口为HTTP请求，对于全局禁用HTTP的项目，需要设置HTTP白名单。以下为运营商HTTP接口域名： `onekey.cmpassport.com` ， `enrichgw.10010.com` 。详情可参见Demo代码。

功能示例

- 获取认证实例（getInstance）

```
/**
 * 获取号码认证服务示例，此实例为单例，获取多次为同一对象
 * @param context Android上下文
 * @param tokenListener需要实现的获取Token回调
 * @return PhoneNumberAuthHelper
 */
public static PhoneNumberAuthHelper getInstance(Context context, TokenResultListener toke
nListener)
```

- 检查认证环境（checkAuthEnvEnable）

```
/**
 * SDK环境检查函数，检查终端是否支持号码认证，通过TokenResultListener返回code
 * type 1：本机号码校验 2：一键登录
 * 600024 终端支持认证
 * 600013 系统维护，功能不可用
 */
 public void checkEnvAvailable(@IntRange(from = 1, to = 2) int type)
```

- 加速本机号码校验（accelerateVerify）

```
/**
 public void accelerateVerify(int overdueTime, final PreLoginResultListener listener);
```

- 本机号码校验Token（getVerifyToken）

```
/**
* 获取认证Token
*
* @param totalTimeout超时时间（单位：ms）
*/
 public void getVerifyToken(final int totalTimeout)
```

- 加速授权页弹出（accelerateLoginPage）

```
/**
* 加速授权页唤起
*
* @param overdueTime预取号有效期
* @param listener预取号回调
*/
public void accelerateLoginPage(final int overdueTime, final PreLoginResultListener liste
ner)
```

- 一键登录唤起授权页（getLoginToken）

```
/**
* 获取登录Token调起一键登录授权页面，在用户授权后获取一键登录的Token
*
* @param totalTimeout超时时间（单位：ms）
*/
public void getLoginToken(final Context context, final int totalTimeout)
```

- 退出授权页（quitLoginPage）

```
/**
* 退出授权认证页
* SDK完成回调之后不会关闭授权页，需要开发者主动调用quitLoginPage退出授权页
*/
public void quitLoginPage()
```

- 关闭授权页loading（hideLoginLoading）

```
/**
* 关闭授权页loading
* SDK完成回调之后不会关闭loading，需要开发者主动调用hideLoginLoading关闭loading
*/
public void hideLoginLoading()
```

其他功能示例

- 返回默认上网卡运营商（getCurrentCarrierName）

```
/**
* 返回默认上网卡运营商
*
* @return CMCC、CUCC、CTCC
*/
public String getCurrentCarrierName()
```

- 使用XML添加自定义控件至一键登录授权页（addAuthRegisterXmlConfig()）

  调用一次addAuthRegisterXmlConfig()方法，XML内绘制的自定义控件全部添加完成。

```
/**
* 添加自定义View
*
* @param xmlConfig
*/
public void addAuthRegisterXmlConfig(AuthRegisterXmlConfig xmlConfig)
```

初始化addAuthRegisterXmlConfig类时需要先调静态内部类Builder()里面的2个方法。

- setLayout()：开发者传入自定义的控件的XML资源ID。
- AbstractPnsViewDelegate()：授权页使用XML添加自定义布局时，可以配合该Delegate类实现XML中相关View的操作，比如事件监听以及动态UI改动等等，当XML对应的View加载后SDK将调用onViewCreated(View)方法通知View已经创建OK，此时可以获取XML中的View并进行相关事件绑定等操作。

> **注意** `onViewCreated(View)` 中返回的View不要用强引用，或者用完要及时释放，否则容易造成内存泄漏。

调用示例：

```
mAlicomAuthHelper.addAuthRegisterXmlConfig(new AuthRegisterXmlConfig.Builder()
 .setLayout(R.layout.xxxxxx, new AbstractPnsViewDelegate() {
    @Override public void onViewCreated(View view) {
        //这里返回的View，不建议用强引用，如果要用，请及时释放，否则容易造成内存泄漏
        findViewById(R.id.xxxx).setOnClickListener(new View.OnClickListener() {
            @Override public void onClick(View v) {
                //do something
            }
        });
    }
})
.build());
```

- 添加代码编写的自定义控件至登录授权页（addAuthRegistViewConfig）

```
/**
* 动态添加控件
*
* @param viewID开发者自定义控件名称
* @param viewConfig配置开发者自定义控件的控件来源、位置和处理逻辑
*/
public void addAuthRegistViewConfig(String viewID, AuthRegisterViewConfig viewConfig)
```

> **注意** 每次调用getVerifyToken授权请求之前，都需初始化一次AuthRegisterViewConfig，因为在授权页关闭时都会清空注入进去的AuthRegisterViewConfig，具体实现请见Demo工程。

初始化AuthRegisterViewConfig类时需要先调静态内部类Builder()里面的3个方法。

- setView()：开发者传入自定义的控件，开发者需要提前设置好控件的布局属性，SDK只支持RelativeLayout布局。
- setRootViewId()：设置控件的位置，目前SDK授权页允许在标题栏RootViewId.ROOT_VIEW_ID_TITLE_BAR、授权页空白处RootViewId.ROOT_VIEW_ID_BODY、授权页号码掩码区域RootViewId.ROOT_VIEW_ID_NUMBER 3个位置插入开发者控件。

- setCustomInterface()：设置控件事件。

  ```
  public Builder setCustomInterface(CustomInterface customInterface)
  ```

  调用示例：

  ```
  mAlicomAuthHelper.addAuthRegistViewConfig("switch_acc_tv", new AuthRegisterViewConfig.B
  uilder()
  .setView(mRL)
  .setRootViewId(AuthRegisterViewConfig.RootViewId.ROOT_VIEW_ID_BODY)
  .setCustomInterface(new CustomInterface(){
     @Override
      public void onClick(Context context){
      startActivityForResult(new Intent(context, SecondActivity.class), 1234);
      }
   }).build());
  ```

  > 注意 成功获取Token后，需把通过setView()方法注入进去的View设置为null。

- 一键登录修改授权页主题（setAuthUIConfig）

  ```
   /**
  * 修改授权页面主题，开发者可以通过此方法修改授权页面主题，需在getLoginToken接口之前调用
  *
  * @param authUIConfig登录授权页UI自定义配配置
  */
  public void setAuthUIConfig(AuthUIConfig authUIConfig)
  ```

  调用示例：

  ```
  setAuthUIConfig(new AuthUIConfig.Builder()
  .setLogBtnText("一键登录")
  .setLogBtnClickableColor(Color.BLACK)
  .setLogBtnUnClickableColor(Color.BLUE)
  .setLogBtnTextColor(Color.WHITE).setLogoHidden(false)
  .setNavColor(0xff026ED2)
  .setNavText("免密登录")
  .setNavTextColor(Color.WHITE)
  .setNumberColor(Color.WHITE)
  .setNumberSize(28)
  .setNumberColor(0xff000000).create());
  ```

内存泄露说明

当您使用自定义控件后，存在内存泄露的风险。您可在获取Token和关闭授权页之前，调用以下代码即可解决。详情可参见Demo工程。

```
mAuthHelper.removeAuthRegisterXmlConfig();
mAuthHelper.removeAuthRegisterViewConfig();
```

SDK回调说明

- 获取Token回调

- 回调返回的ret都通过TokenRet tokenRet = JSON.parseObject(ret, TokenRet.class) 解析。
- 授权页唤起成功、获取Token成功都会回调onTokenSuccess方法（可以通过返回码来区分）。
- 获取Token失败会回调onTokenFailed。

获取Token回调示例代码：

```
mTokenListener = new TokenResultListener() {
 @Override
 public void onTokenSuccess(final String ret) {
 MainActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    /*
                     *  setText just show the result for get token
                     *  use ret to verify number
                     */
                    //ResultCode#CODE_START_AUTHPAGE_SUCCESS是授权页唤起成功码，若不需要处理
，则过滤
                     if (ResultCode.CODE_START_AUTHPAGE_SUCCESS.equals(tokenRet.getCode()
)) {
                           return;
                    }
                    TokenRet tokenRet = JSON.parseObject(ret, TokenRet.class);
                    if (tokenRet != null) {
                        token = tokenRet.getToken();
                    }
                    mAlicomAuthHelper.quitLoginPage();
                }
            });
        }
        @Override
        public void onTokenFailed(final String ret) {
            MainActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    /*
                     *  setText just show the result for get token
                     *  do something when getToken failed, such as use sms verify code.
                     */
                    TokenRet tokenRet = JSON.parseObject(ret, TokenRet.class);
                    mAlicomAuthHelper.quitLoginPage();
                }
            });
        }
    };
```

- 加速唤起授权页或加速本机号码校验回调

```
public interface PreLoginResultListener {
/**
* @param vendor返回预取成功运营商
*/
void onTokenSuccess(String vendor);
/**
* @param vendor返回预取失败运营商
* @param ret返回失败原因
*/
void onTokenFailed(String vendor, String ret);
}
```

预取号回调示例代码：

```
mPreLoginResultListener = new PreLoginResultListener() {
            @Override
            public void onTokenSuccess(final String s) {
                MainActivity.this.runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                       /*
                        *  推荐在登录页初始化的时候调用
                        *  如果没有合适的调用时机
                        *  不调用此接口也没关系
                        *  千万不要App冷启动初始化就调用
                        *  不要调用完预取号后马上调用getLoginToken
                        *  最好判断用户是否登录，若已登录不要使用此接口
                        */
                        mRetTV.setText("预取号成功:" + s);
                    }
                });
            }
            @Override
            public void onTokenFailed(final String s, final String s1) {
                /*
                 *  预取号调用失败
                 *  不用太关注，还是可以直接在用户点击"登录"时，调用getLoginToken
                 */
                mRetTV.setText("预取号失败:" + s + s1);
            }
        });
     }
}
```

- 控件点击事件回调

  授权页控件点击事件通过此回调返回

```
public interface AuthUIControlClickListener{
/**
*
* @param code控件点击事件code
* @param context Android上下文
* @param jsonObj点击事件返回的具体内容，不同控件返回的事件内容有所不同
*/
void onClick(String code, Context context, JSONObject jsonObj);
}
```

回调示例代码

```
mAlicomAuthHelper.setUIClickListener(new AuthUIControlClickListener( ) {
            @Override
            public void onClick(String code, Context context, JSONObject jsonObj) {
                Log.e("xxxxxx", "OnUIControlClick:code=" + code + ", jsonObj=" + (jsonObj
== null ? "" : jsonObj.toJSONString()));
}
});
```

代码调用顺序

```
/*
* 1.初始化获取Token实例
*/
mTokenListener = new TokenResultListener() {}
/*
* 2.初始化SDK实例
*/
mAlicomAuthHelper = PhoneNumberAuthHelper.getInstance(context, mTokenListener);
/*
* 3.设置SDK密钥
*/
mAlicomAuthHelper.setAuthSDKInfo();
/*
* 4.检测终端网络环境是否支持一键登录或者号码认证，根据回调结果确定是否可以使用一键登录功能
*/
mAlicomAuthHelper.checkEnvAvailable(PhoneNumberAuthHelper#SERVICE_TYPE_LOGIN);
/*
* 5.若步骤4返回true，则根据业务情况，调用预取号或者一键登录接口
*   详见Demo接入工程
*/
mAlicomAuthHelper.getLoginToken(context, 5000);
```

进行取号

- 一键登录获取手机号：当您成功获取到getLoginToken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。
- 本机号码校验结果：当您成功获取到GetVerifyToken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VerifyMobile接口，即可进行最终的取号操作。

SDK返回码

返回码	返回码描述	解决方案
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。	升级SDK版本。您可前往号码认证服务控制台下载最新版SDK。
600005	手机终端不安全。	切换到其他登录方式。
600007	未检测到SIM卡。	提示用户检查SIM卡后重试。
600008	移动数据网络未开启。	提示用户开启移动数据网络后重试。
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。	检查数据网络环境后重试，若还未解决问题，您可通过切换飞行模式、重启手机或切换到其他登录方式的操作解决问题。
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，该功能调用次数已达上限。	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。	密钥未设置或者设置错误，请先检查密钥信息，如密钥无问题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。
600023	加载自定义控件异常。	检查自定义控件添加是否正确。
600024	终端环境检查支持认证。	无。
600025	终端检测参数错误。	检查传入参数类型与范围是否正确。
600026	授权页已加载时不允许调用加速或预取号接口。	检查是否有授权页拉起后，调用preLogin或者accelerateAuthPage接口的行为不被允许。

除阿里云SDK返回码外，运营商错误码详情请参见运营商SDK错误码。

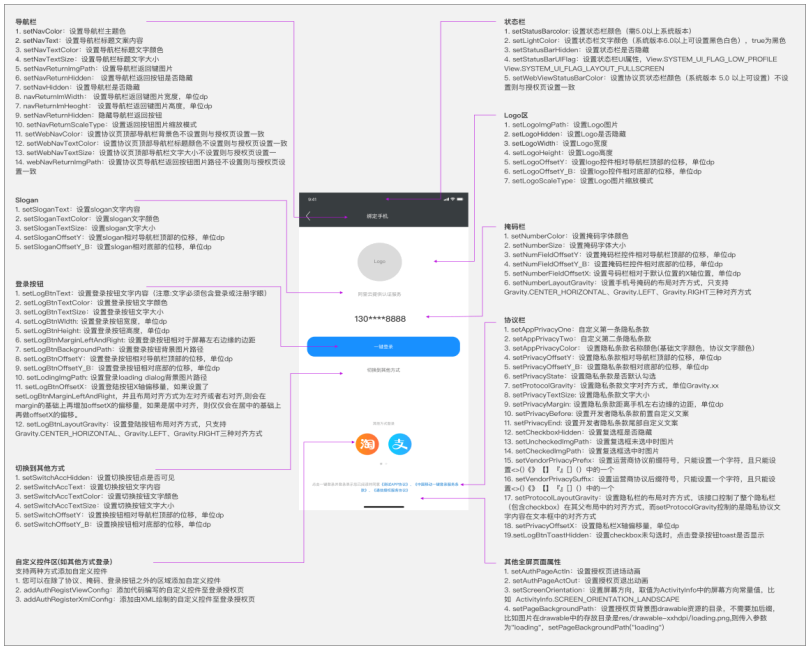
授权页点击事件响应码

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。

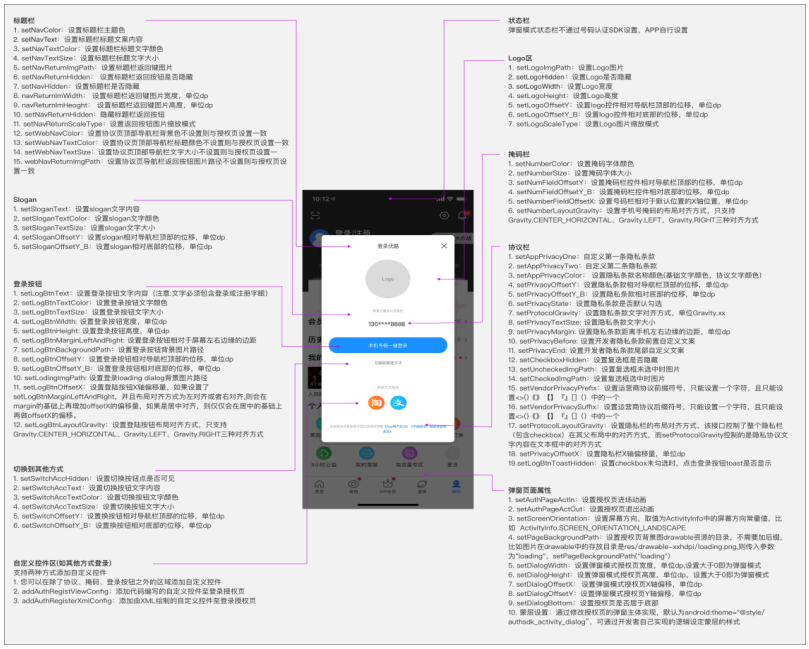
授权页面设计规范与配置说明

一键登录的授权页面：

> **注意** 涉及图片路径的参数，仅仅为图片名称（不带路径或后缀名），并且图片需要放置在drawable、drawable-xxhdpi等目录下。



弹窗授权页面设计规范：支持横屏，不支持横竖屏切换，以竖屏举例。



授权页配置说明如下所示：

- 授权页导航栏

方法	方法类型	说明
setStatusBarColor	int	设置状态栏颜色（系统版本5.0以上可设置）。
setLightColor	int	设置状态栏字体颜色（系统版本6.0以上可设置黑色、白色，默认为黑色）。
setNavColor	int	设置导航栏颜色。
setNavText	String	设置导航栏标题文字。
setNavTextColor	int	设置导航栏标题文字颜色。
setNavTextSize	int	设置导航栏标题文字大小。
setNavReturnImgPath	String	设置导航栏返回键图片。
setNavReturnHidden	boolean	设置导航栏返回按钮隐藏。
setNavHidden	boolean	设置默认导航栏是否隐藏。
setStatusBarHidden	boolean	设置状态栏是否隐藏。
setStatusBarUIFlag	int	设置状态栏UI属性，View.SYSTEM_UI_FLAG_LOW_PROFILE、View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN。

方法	方法类型	说明
setWebViewStatusBarColor	int	设置协议页状态栏颜色（系统版本5.0以上可设置），不设置则与授权页设置一致。
setWebNavColor	int	设置协议页顶部导航栏背景色，不设置则与授权页设置一致。
setWebNavTextColor	int	设置协议页顶部导航栏标题颜色，不设置则与授权页设置一致。
setWebNavTextSize	int	设置协议页顶部导航栏文字大小，不设置则与授权页设置一致。
webNavReturnImgPath	String	设置协议页导航栏返回按钮图片路径，不设置则与授权页设置一致。
setBottomNavColor	int	设置底部虚拟按键背景色（系统版本5.0以上可设置）。
setNavTextSizeDp	int	设置导航栏标题文字大小（单位：db，字体大小不随系统变化）。
setNavReturnImgDrawable	Drawable	设置导航栏返回键图片。

- 授权页Logo

方法	方法类型	说明
setLogoHidden	boolean	隐藏logo。
setLogoImgPath	String	设置logo图片。
setLogoWidth	int	设置logo控件宽度。
setLogoHeight	int	设置logo控件高度。
setLogoOffsetY	int	设置logo控件相对导航栏顶部的位移（单位：dp）。
setLogoOffsetY_B	int	设置logo控件相对底部的位移（单位：dp）。
setLogoScaleType	ImageView.ScaleType	设置logo图片缩放模式。

- 授权页Slogan

方法	方法类型	说明
setSloganText	String	设置Slogan文字内容。
setSloganTextColor	int	设置Slogan文字颜色。

方法	方法类型	说明
setSloganTextSize	int	设置Slogan文字大小。
setSloganOffsetY	int	设置Slogan相对导航栏顶部的位移（单位：dp）。
setSloganOffsetY_B	int	设置Slogan相对底部的位移（单位：dp）。
setSloganTextSizeDp	int	设置Slogan文字大小（单位：db，字体大小不随系统变化）。

- 授权页号码栏

方法	参数类型	说明
setNumberColor	int	设置手机号码字体颜色。
setNumberSize	int	设置手机号码字体大小。
setNumFieldOffsetY	int	设置号码栏控件相对导航栏顶部的位移（单位：dp）。
setNumFieldOffsetY_B	int	设置号码栏控件相对底部的位移（单位：dp）。
setNumberFieldOffsetX	int	设置号码栏相对于默认位置的X 轴偏移量（单位：dp）。
setNumberLayoutGravity	int	设置手机号掩码的布局对齐方式，只支持Gravity.CENTER_HORIZONTAL、Gravity.LEFT、Gravity.RIGHT三种对齐方式。
setNumberSizeDp	int	设置手机号码字体大小（单位：db，字体大小不随系统变化）。

- 授权页登录按钮

方法	参数类型	说明
setLogBtnText	String	设置登录按钮文字。
setLogBtnTextColor	int	设置登录按钮文字颜色。
setLogBtnTextSize	int	设置登录按钮文字大小。
setLogBtnWidth	int	设置登录按钮宽度（单位：dp）。
setLogBtnHeight	int	设置登录按钮高度（单位：dp）。

方法	参数类型	说明
setLogBtnMarginLeftAndRight	int	设置登录按钮相对于屏幕左右边缘边距。
setLogBtnBackgroundPath	String	设置登录按钮背景图片路径。
setLogBtnOffsetY	int	设置登录按钮相对导航栏顶部的位移（单位：dp）。
setLogBtnOffsetY_B	int	设置登录按钮相对底部的位移（单位：dp）。
setLoadingImgPath	String	设置登录loading dialog背景图片路径。
setLogBtnOffsetX	int	设置登录按钮X轴偏移量，如果设置了setLogBtnMarginLeftAndRight，并且布局对齐方式为左对齐或者右对齐,则会在margin的基础上再增加offsetX的偏移量，如果是居中对齐，则仅仅会在居中的基础上再做offsetX的偏移。
setLogBtnLayoutGravity	int	设置登录按钮布局对齐方式，只支持Gravity.CENTER_HORIZONTAL、Gravity.LEFT、Gravity.RIGHT三种对齐方式。
setLogBtnTextSizeDp	int	设置登录按钮文字大小（单位：db，字体大小不随系统变化）。
setLogBtnBackgroundDrawable	Drawable	设置登录按钮背景图片路径。
setLoadingImgDrawable	Drawable	设置登录loading dialog背景图片路径。

- 授权页隐私栏

方法	参数类型	说明
setAppPrivacyOne	String, String	设置开发者隐私条款1名称和URL（名称，URL）。
setAppPrivacyTwo	String, String	设置开发者隐私条款2名称和URL（名称，URL）。
setAppPrivacyColor	int, int	设置隐私条款名称颜色（基础文字颜色，协议文字颜色）。
setPrivacyOffsetY	int	设置隐私条款相对导航栏顶部的位移（单位：dp）。

方法	参数类型	说明
setPrivacyOffsetY_B	int	设置隐私条款相对底部的位移（单位：dp）。
setPrivacyState	boolean	设置隐私条款是否默认勾选。
setProtocolGravity	int	设置隐私条款文字对齐方式（单位：Gravity.xxx）。
setPrivacyTextSize	int	设置隐私条款文字大小（单位：sp）。
setPrivacyMargin	int	设置隐私条款距离手机左右边缘的边距（单位：dp）。
setPrivacyBefore	String	设置开发者隐私条款前置自定义文案。
setPrivacyEnd	String	设置开发者隐私条款尾部自定义文案。
setCheckboxHidden	boolean	设置复选框是否隐藏。
setUncheckedImgPath	String	设置复选框未选中时图片。
setCheckedImgPath	String	设置复选框选中时图片。
setVendorPrivacyPrefix	String	设置运营商协议前缀符号，只能设置一个字符，且只能设置<>、()、《》、【】、『』、[]、（）中的一个。
setVendorPrivacySuffix	String	设置运营商协议后缀符号，只能设置一个字符，且只能设置<>、()、《》、【】、『』、[]、（）中的一个。
setProtocolLayoutGravity	int	设置隐私栏的布局对齐方式，该接口控制了整个隐私栏（包含check box）在其父布局中的对齐方式，而setProtocolGravity控制的是隐私协议文字内容在文本框中的对齐方式。
setPrivacyOffsetX	int	设置隐私栏X轴偏移量（单位：dp）。
setLogBtnToastHidden	boolean	设置check box未勾选时，点击登录按钮toast是否显示。
setPrivacyTextSizeDp	int	设置隐私条款文字大小（单位：dp，字体大小不随系统变化）。
setUncheckedImgDrawable	Drawable	设置复选框未选中时图片。

方法	参数类型	说明
setCheckedImgDrawable	Drawable	设置复选框选中时图片。

- 切换控件方式

方法	参数类型	说明
setSwitchAccHidden	boolean	设置切换按钮点是否可见。
setSwitchAccText	String	设置切换按钮文字内容。
setSwitchAccTextColor	int	设置切换按钮文字颜。
setSwitchAccTextSize	int	设置切换按钮文字大小。
setSwitchOffsetY	int	设置换按钮相对导航栏顶部的位移（单位：dp）。
setSwitchOffsetY_B	int	设置换按钮相对底部的位移（单位：dp）。
setSwitchAccTextSizeDp	int	设置切换按钮文字大小（单位：dp，字体大小不随系统变化）。

- 页面相关函数

方法	参数类型	说明
setAuthPageActIn	String	设置授权页进场动画。
setAuthPageActOut	String	设置授权页退出动画。
setScreenOrientation	int	设置屏幕方向，取值为ActivityInfo中的屏幕方向常量值，比如：ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE。
setPageBackgroundPath	String	设置授权页背景图，drawable资源的目录，不需要加后缀，比如图片在drawable中的存放目录是res/drawable-xxhdpi/loading.png,则传入参数为"loading"，setPageBackgroundPath("loading")。
setDialogWidth	int	设置弹窗模式授权页宽度（单位：dp）,设置大于0，即为弹窗模式。
setDialogHeight	int	设置弹窗模式授权页高度（单位：dp），设置大于0，即为弹窗模式。

方法	参数类型	说明
setDialogOffsetX	int	设置弹窗模式授权页X轴偏移（单位：dp）。
setDialogOffsetY	int	设置弹窗模式授权页Y轴偏移（单位：dp）。
setDialogBottom	boolean	设置授权页是否居于底部。
setPageBackgroundDrawable	Drawable	设置授权页背景图。
setProtocolAction	String	自定义协议页跳转Action。
setPackageName	String	配置自定义协议页跳转Action必须配置这个属性，值为App的包名。

2.3.3.2. iOS客户端接入

本文为您介绍了iOS客户端如何接入一键登录和本机号码校验功能。

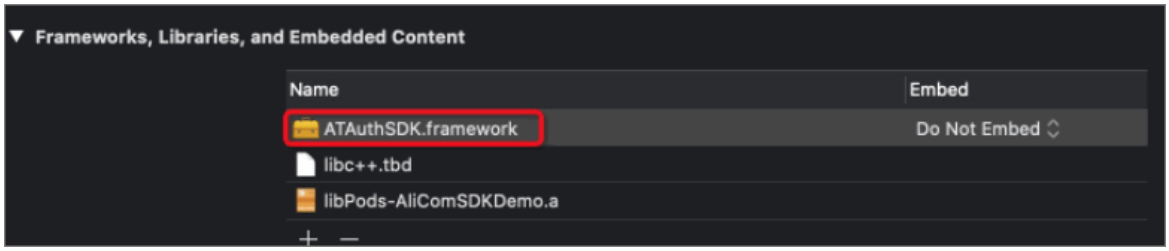
前提条件

- 确保您已开通了号码认证服务，并成功创建了对应的认证方案。详情请参见一键登录和本机号码校验使用流程。
- 确保您的终端设备已开启SIM卡的移动数据（支持中国联通、中国移动的3G网络，但接口耗时会增加）。
- 开发工具建议使用Xcode 11及以上。
- 支持iOS 10及以上系统。

步骤一：搭建开发环境

1. 登录号码认证服务控制台，在**标准版**选项卡，下载并解压iOS SDK。
2. 下载并安装Xcode 11。
3. 添加主库。

   在菜单栏选择**TARGETS > General > Frameworks, Libraries, and Embedded Content**，添加主库**ATAuthSDK.framework**。

4. BuildSettings设置。

   在菜单栏选择**TARGETS > BuildSettings > Other Linker Flags**，**Other Linker Flags**增加-ObjC。

5. 添加ATAuthSDK.bundle资源文件。

   打开*PhoneNumberAuthSDK*文件，添加**ATAuthSDK.framework > ATAuthSDK.bundle**资源文件，否则一键登录授权页面默认的图片或icon不可显示。

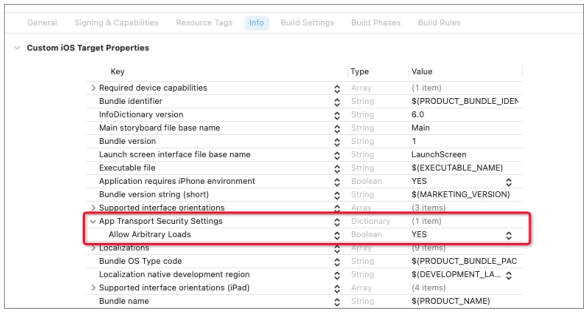
6. Crash收集。

   - Crash组件库只收集号码认证SDK的Crash，不收集工程里其他Crash。
   - ATAuthSDK_D.framework和ATAuthSDK.framework不能同时集成到工程里，两者是同一个SDK，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），静态库则不去收集，且同时要在工程中加入AliComCrash目录中的AliComCrash.framework（Crash收集组件库）。

   > **说明** 从v2.10.1版本之后，号码认证SDK支持收集SDK内部的Crash问题，方便线上问题排查。您可以选择是否需要SDK支持该功能，如需支持，请接入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）。

7. （可选）域名配置。

   若您的终端设备使用的是中国联通SIM卡的5G移动数据，可能会导致使用一键登录功能获取本机号码失败。您可在菜单栏选择**TARGETS > Info > Custom iOS Target Properties > App Transport Security Settings > Allow Arbitrary Loads**，并将其值设置为YES可解决此问题。



步骤二：运行Demo工程

登录号码认证产品控制台，在**概览**页面下载**开发者接入Demo**，再打开开发工具直接运行Demo工程。

功能示例

- 获取认证实例（sharedInstance）

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

- 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;
```

- 设置SDK密钥（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方案对应的密钥，请登录阿里云控制台，进入认证方案管理，点击复制密钥，建议维护在业务服务端
* @param complete结果同步回调，成功时resultDic=@{resuFltCode:600000，msg:...}，其他情况时"res
ultCode"值请参考PNSReturnCode
*/
- (void)setAuthSDKInfo:(NSString * _Nonnull)info complete:(void(^_Nullable)(NSDictionary
* _Nonnull resultDic))complete;
```

- 检查认证环境（checkEnvAvailableWithComplete）

```
/**
* 函数名：checkEnvAvailableWithComplete
* @brief：检查及准备调用环境，resultDic返回PNSCodeSuccess才能调用下面的功能接口，在初次或切换移动
数据网络之后需要重新调用，一般在一次登录认证流程开始前调一次即可
*@param complete：异步结果回调，成功时resultDic=@{resultCode:600000, msg:...}，其他情况时"res
ultCode"值请参考PNSReturnCode，只有成功回调才能保障后续接口调用
*/
- (void)checkEnvAvailableWithComplete:(void (^_Nullable)(NSDictionary * Nullable resultDi
c))complete;
```

- 一键登录预取号（accelerateLoginPageWithTimeout）

```
/**
* 函数名：accelerateLoginPageWithTimeout
* @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
* @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，成功时resultDic=@{resultCode:600000,msg:...}，其他情况时"res
ultCode"值请参考PNSReturnCode
*/
- (void)accelerateLoginPageWithTimeout:(NSTimeInterval)timeout complete:(void (^_Nullable
)(NSDictionary * Nonnull resultDic))complete;
```

- 一键登录获取Token（getLoginTokenWithTimeout）

```
/**
* 函数名：getLoginNumberWithTimeout
* @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
* @warning：注意的是，如果前面没有调用accelerateLoginPageWithTimeout:complete接口，成功后弹起授权页
* @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 controller：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
* @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
* @param complete：结果异步回调，"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）.....
*/
- (void)getLoginTokenWithTimeout:(NSTimeInterval)timeout controller:(UIViewController *_Nonnull)controller model:(TXCustomModel *_Nullable)model complete:(void (^_Nullable)(NSDictionary * _Nonnull resultDic))complete;
```

- 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;
```

- 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;
```

- 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;
```

- 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)(N
SDictionary
* _Nonnull resultDic))complete;
```

- 本机号码校验获取Token（getVerifyTokenWithTimeout）

```
/**
* 函数名：getVerifyTokenWithTimeout
* @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
情时"resultCode"值请参考PNSReturnCode
*/
 - (void)getVerifyTokenWithTimeout:(NSTimeInterval)timeout complete:(void (^Nullable)(NSD
ictionary *
_Nonnull resultDic))complete;
```

其他功能示例

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

```
/**
* 函数名：setConsolePrintLoggerEnable
* @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
* @param enable：开关参数，默认为NO
*/
 - (void)setConsolePrintLoggerEnable:(BOOL)enable;
```

- 设置日志及埋点上传开关（setUploadEnable）

```
/**
* 函数名：setUploadEnable
* @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
* @param enable开关设置BOOL值，默认为YES
*/
- (void)setUploadEnable:(BOOL)enable;
```

- 判断设备移动数据网络是否开启（checkDeviceCellularDataEnable）

```
/**
* 函数名：checkDeviceCellularDataEnable
*/
+ (BOOL)checkDeviceCellularDataEnable;
```

- 判断当前上网卡是否为中国联通（isChinaUnicom）

```
/**
* 函数名：isChinaUnicom
*/
+ (BOOL)isChinaUnicom;
```

- 判断当前上网卡是否为中国移动（isChinaMobile）

```
/**
* 函数名：isChinaMobile
*/
+ (BOOL)isChinaMobile;
```

- 判断当前上网卡是否为中国电信（isChinaTelecom）

```
/**
* 函数名：isChinaTelecom
*/
+ (BOOL)isChinaTelecom;
```

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

```
/**
* 函数名：getCurrentMobileNetworkName
* @return ChinaMobile,ChinaUnicom,ChinaTelecom,OtherChinaMobileNetwork,NoChinaMobileNetwo
rk
*/
+ (NSString *)getCurrentMobileNetworkNam
```

- 获取当前上网卡运营商名称（getCurrentCarrierName）

```
/**
* 函数名：getCurrentCarrierName
* @return：中国移动，中国联通，中国电信等
*/
+ (NSString *)getCurrentCarrierName;
```

- 获取当前上网网络类型（getNetworktype）

```
/**
* 函数名：getNetworktype
* @return：Wi-Fi，4G，3G，2G，NoInternet等
*/
+ (NSString *)getNetworktype;
```

- 判断设备是否有SIM卡（simSupportedIsOK）

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

- 判断WWAN是否开启（isWWANOpen）

```
/**
* 函数名：isWWANOpen
* @breif：判断WWAN是否开启（通过pO网卡判断，无Wi-Fi或有Wi-Fi情况下都能检测到）
*/
+ (BOOL)isWWANOpen;
```

- 判断无Wi-Fi下WWAN是否开启（reachableViaWWAN）

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

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

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

SDK返回码

返回码	返回码描述	解决方案
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。	升级SDK版本。您可前往号码认证服务控制台下载最新版SDK。
600005	手机终端不安全。	切换到其他登录方式。
600007	未检测到SIM卡。	提示用户检查SIM卡后重试。
600008	移动数据网络未开启。	提示用户开启移动数据网络后重试。
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。	检查数据网络环境后重试，若还未解决问题，您可通过切换飞行模式、重启手机或切换到其他登录方式的操作解决问题。
600013	运营商维护升级，该功能不可用。	创建工单联系工程师。

返回码	返回码描述	解决方案
600014	运营商维护升级，该功能调用次数已达上限。	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。	密钥未设置或者设置错误，请先检查密钥信息，如密钥无问题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。
600023	加载自定义控件异常。	检查自定义控件添加是否正确。
600024	终端环境检查支持认证。	无。
600025	终端检测参数错误。	检查传入参数类型与范围是否正确。
600026	授权页已加载时不允许调用加速或预取号接口。	检查是否有授权页拉起后，调用preLogin或者accelerateAuthPage接口的行为不被允许。

除阿里云SDK返回码外，运营商错误码详情请参见运营商SDK错误码。

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。

一键登录唤起授权页

```
//1.设置SDK参数，App生命周期内调用一次即可
 NSString * info = @"客户的密钥串";
 __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 r
esultDic) {
    [weakSelf showResult:resultDic];
 }];
 //2.检测当前环境是否支持一键登录
 __block BOOL support = YES;
 [[TXCommonHandler sharedInstance] checkEnvAvailableWithAuthType:PNSAuthTypeLoginToken comp
lete:^(NSDictionary * _Nullable resultDic) {
    support = [PNSCodeSuccess isEqualToString:[resultDic objectForKe y:@"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:^(NSDict
ionary * _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，可以立刻弹起授权页，model的创建需要放在主线程
    [ProgressHUD dismiss];
    [[TXCommonHandler sharedInstance] getLoginTokenWithTimeout:timeout controller:weakSelf
model:model complete:^(NSDictionary * _Nonnul l resultDic)     {
        NSString *code = [resultDic objectForKey:@"resultCode"];
        if ([PNSCodeLoginControllerPresentSuccess isEqualToString:co de]) {
            [ProgressHUD showSuccess:@"弹起授权页成功"];
        } else if ([PNSCodeLoginControllerClickCancel isEqualToStrin g:code]) {
            [ProgressHUD showSuccess:@"点击了授权页的返回"];
        } else if ([PNSCodeLoginControllerClickChangeBtn isEqualToSt ring:code]) {
            [ProgressHUD showSuccess:@"点击切换其他登录方式按钮"];
        } else if ([PNSCodeLoginControllerClickLoginBtn isEqualToStr ing: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
录Token"];
            } else {
                [ProgressHUD showSuccess:@"点击了登录按钮，check box选中，SDK内部不会去获取登录T
oken"];
            }
        } else if ([PNSCodeLoginControllerClickCheckBoxBtn isEqualTo String:code]) {
            [ProgressHUD showSuccess:@"点击check box"];
        } else if ([PNSCodeLoginControllerClickProtocol isEqualToStr ing: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"];
            //用Token去服务器换手机号，以下操作仅做参考
            [PNSVerifyTopRequest requestLoginWithToken:token complet e:^(BOOL isSuccess, NS
String * _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] cancelLoginVCA nimated:YES complete:n
il];
                });
                [weakSelf showResult:data];
            }];
```

```
        } else {
            [ProgressHUD showError:@"获取登录Token失败"];
        }
        [weakSelf showResult:resultDic];
    }];
 }];
```

授权页全屏模式

```
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 system
FontOfSize: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"attrib
utes:@{NSForegroundColorAttributeName : UIColor.orangeColor,NSFontAttributeName : [UIFont s
ystemFontOfSize: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 systemFon
tOfSize: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"attrib
utes:@{NSForegroundColorAttributeName : UIColor.orangeColor,NSFontAttributeName : [UIFont s
ystemFontOfSize: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 fra
me) {
        CGFloat width = superViewSize.height;
        CGFloat height = width;
        return CGRectMake(superViewSize.width - 15 - width, 0, width, height);
    };
    model.loginBtnFrameBlock = ^CGRect(CGSize screenSize,CGSize superViewSize,CGRect frame)
{
        if ([self isHorizontal:screenSize]) {
            frame.origin.y = 20;
            return frame;
        }
        return frame;
    };
    model.sloganFrameBlock = ^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(0,140,superViewSize.width,frame.size.height);
        }
    };
    model.numberFrameBlock = ^CGRect(CGSize screenSize,CGSize superViewSize,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 navFra
me,CGRect titleBarFrame,CGRect logoFrame,CGRect sloganFrame,CGRect numberFrame,CGRect lo
```

```
me,CGRect titleBarFrame,CGRect logoFrame, CGRect sloganFrame, CGRect numberFrame, CGRect lo
ginFrame, 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;
```

授权页弹窗模式

```
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）"attribut
es:@{NSForegroundColorAttributeName : UIColor.whiteColor, NSFontAttributeName : [UIFont sys
temFontOfSize: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"attrib
utes:@{NSForegroundColorAttributeName : UIColor.orangeColor,NSFontAttributeName : [UIFont s
ystemFontOfSize: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 systemFo
ntOfSize: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"attrib
utes:@{NSForegroundColorAttributeName : UIColor.orangeColor,NSFontAttributeName : [UIFont s
ystemFontOfSize: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_Padding;
            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 superViewSize,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 superViewSize, 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 superViewSize,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 fra
me) {
        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 navF
rame, CGRect titleBarFrame, CGRect logoFrame, CGRect sloganFrame, CGRect numberFrame, CGRec
t 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
```

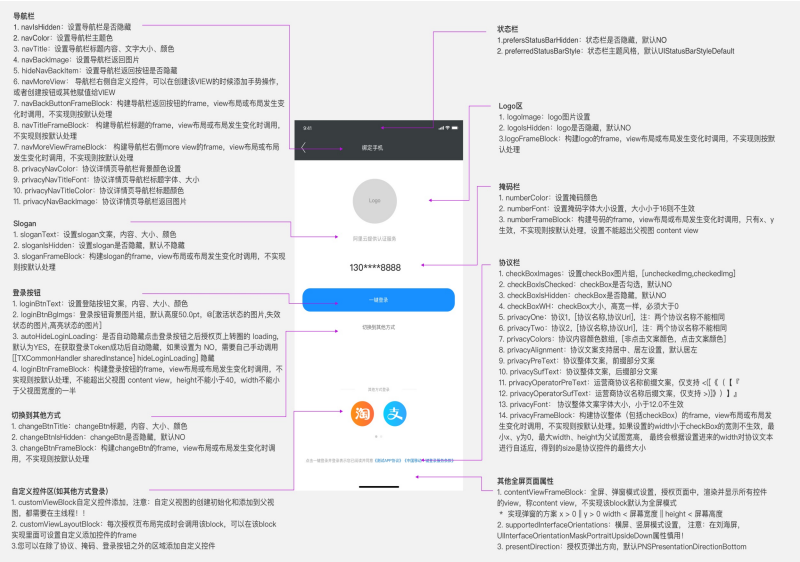
```
model.supportedInterfaceOrientations = UIInterfaceOrientationMaskLandscape;
```

本机号码校验用例

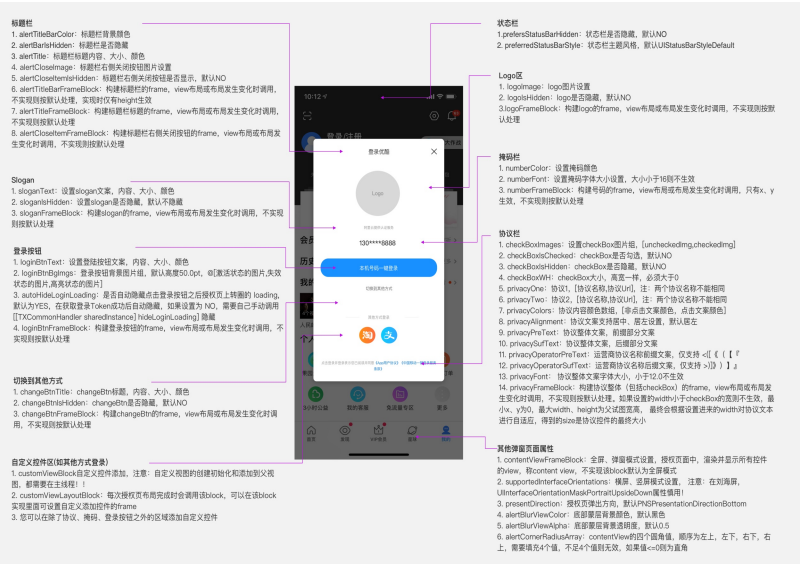
```
//1.设置SDK参数，App生命周期内调用一次即可
NSString *info = @"客户的密钥串";
__weak typeof(self) weakSelf = self;
//设置SDK参数，App生命周期内调用一次即可
[[TXCommonHandler sharedInstance] setAuthSDKInfo:info complete:^(NSDictionary * _Nonnull re
sultDic) {
[weakSelf showResult:resultDic];
}];
//2.检测当前环境是否支持本机号码校验
 __block BOOL support = YES;
 [[TXCommonHandler sharedInstance] checkEnvAvailableWithAuthType:PNSAuthTypeVerifyToken com
plete:^(NSDictionary * _Nullable resultDic) {
    support = [PNSCodeSuccess isEqualToString:[resultDic objectForKe y:@"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:^(NSDictionar
y * _Nonnull resultDic) {
    if ([PNSCodeSuccess isEqualToString:[resultDic objectForKey:@"re sultCode"]] == NO) {
        [ProgressHUD showError:@"获取VerifyToken失败"];
        [weakSelf showResult:resultDic];
        return ;
    }
 //4.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
    [PNSVerifyTopRequest requstVerifyWithNumber:weakSelf.tf_phoneNum ber.text token:token c
omplete:^(BOOL isSuccess, NSString * _Nonnull msg, NSDictionary * _Nonnull data) {
        NSDictionary *module = data[@"module"];
        NSString *verify_result = [module objectForKey:@"verify_resu lt"];
        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];
    }];
}];
```

授权页面设计规范

全屏授权页面设计规范：支持横屏，以竖屏示意。



弹窗授权页面设计规范：支持横屏，以竖屏示意。



> **注意** 一键登录按钮形状不支持设置圆角，但您可使用loginBtnBgimgs方法配置圆角形状的登录按钮背景图。

2.3.3.3. UniApp客户端接入

本文为您介绍了UniApp插件接入流程。

前提条件

您已完成应用创建并获取相应的访问密钥，详情请参见一键登录和本机号码校验使用流程。

背景信息

阿里云号码认证服务基于HBuilder提供的UniApp原生插件扩展开发出的认证插件，适用于Android和iOS的Hbuilder插件，开发者可以轻松将阿里云号码认证能力集成到自己的项目中，从而在js层实现对认证的控

制。

SDK初始化API使用说明

除本文档的描述外，您还可参见DCloud官网文档，完成UniApp插件接入。

1. 引入方式。

   调用示例：

   ```
   const aLiSDKModule = uni.requireNativePlugin('AliCloud-NirvanaPns')
   ```

2. 设置密钥（必调接口）。

   在您使用一键登录功能前，需要先设置密钥，并确保密钥与Bundle ID或包名匹配。

   Object参数说明：

参数	类型	描述
info	String	阿里云号码认证产品控制台生成的密钥。

   调用示例：

   ```
   aLiSDKModule.setAuthSDKInfo("从阿里云控制台获取的密钥")
   ```

一键登录API使用说明

1. （可选）加速拉起授权页

   加速拉起授权页，防止由于调用getLoginToken接口，导致拉起授权页时间过长。

   > 注意
   > - 建议在当前用户处于未登录状态时调用该方法，已登录用户无需调用。
   > - 由于加速方法需要1~3秒的时间取得临时凭证，建议在调用拉取授权页方法2~3秒前调用预取号方法。
   > - 请勿频繁地调用加速拉起授权页方法，不得与拉起授权登录页同时调用或在其后调用。
   > - App直接登录的场景无需调用此接口。

   Object参数说明：

参数	类型	描述
timeout	Number	设置接口超时时间（单位：毫秒）。

   Callback参数说明：

参数	类型	描述
resultCode	String	Code值。600000表示接口调用成功，更多Code值请参见SDK返回码。

参数	类型	描述
msg	String	相关提示信息。

调用示例：

```
aLiSDKModule.accelerateLoginPage(5000, result => {
    if ("600000" == result.resultCode) {
        console.log("加速成功")
    }
})
```

2. 唤起一键登录授权页

获取一键登录Token，调用该接口会先弹出授权页，在该页面单击登录按钮获取Token。

Object参数说明：

参数	类型	描述
timeout	Number	设置接口超时时间（单位：毫秒）。
authUiConfig	Map	授权页UI配置，具体配置，请参见下文一键登录修改授权页主题。

TokenCallback参数说明：

参数	类型	描述
resultCode	String	Code值。600000表示接口调用成功，更多Code值请参见SDK返回码。
msg	String	相关提示信息。
token	String	最终换取手机号码的Token，只有在resultCode值为600000的时候该字段才会有值。

UiCallback（单击自带控件回调）参数说明：

参数	类型	描述
resultCode	String	Code值，具体请参见下文授权页单击事件响应码。

授权页单击事件响应码

resultCode	描述
700000	单击返回，用户取消免密登录。
700001	单击切换按钮，用户取消免密登录。

resultCode	描述
700002	单击登录按钮事件。
700003	单击check box事件。
700004	单击协议富文本文字事件。

CustomUiCallBack（单击自定义控件回调）参数说明：

参数	类型	描述
widgetId	String	自定义控件的widgetId。

调用示例：

```
aLiSDKModule.getLoginToken(
    5000,
    this.authUiConfig,
    tokenResult => {
        if ("600001" == tokenResult.resultCode) {
            console.log("授权页拉起成功")
        } else if ("600000" == tokenResult.resultCode) {
            console.log("获取Token成功，接下来拿着结果里面的Token去服务端换取手机号码，SDK服务到此结束")
            //手动关闭授权页
            aLiSDKModule.quitLoginPage()
        } else {
            //其他失败情况，手动关闭授权页
            aLiSDKModule.quitLoginPage()
        }
    },
    clickResult => {
        switch (clickResult.resultCode) {
            case "700000":
                console.log("用户点击返回按钮")
                break
            case "700001":
                console.log("用户切换其他登录方式")
                break
            case "700002":
                console.log("用户点击登录按钮")
                break
            case "700003":
                console.log("用户点击checkBox")
                break
            case "700004":
                console.log("用户点击协议")
                break
        }
    },
    customUiResult => {
        //这里回调的是自定义控件的单击事件，通过customUiResult.widgetId来识别自定义控件，然后做一些自己的处理
    }
)
```

3. 退出授权页

   获取登录Token后需开发者主动调用改接口退出授权页。

   调用示例：

   ```
   aLiSDKModule.quitLoginPage()
   ```

4. 关闭授权页loading

   Android客户端与iOS客户端获取Token之后，关闭授权页面上的loading动画方法如下：

   - Android：获取Token结果回调后不会关闭loading动画，需要主动调用接口关闭loading动画。具体操作请参见Step 5的调用示例。

- iOS：获取Token结果回调后会主动关闭loading动画。若要设置SDK不主动关闭loading动画，即手动控制loading消失事件，需在uiConfig配置项中加上 `autoHideLoginLoading: "false"` 。具体操作请参见步骤5的调用示例。

调用示例

```
aLiSDKModule.hideLoginLoading()
```

5. 一键登录修改授权页主题
   - Android配置项说明。调用示例：

```
{
    uiConfig: {
        setNavHidden: "false",
        setLogoHidden: "false",
        setSloganHidden: "false",
        setCheckboxHidden: "false",
        setSwitchHidden: "true",
        //授权页Loading图
        setLoadingImgPath: "",
        //授权页背景图
        setAuthBgImgPath: "",
        setNavUi: {
            bgColor:"#1190ff",
            text: "一键登录",
            textColor: "#fbfbfb",
            textSize: "17",
            returnImgHidden: "false",
            returnImgPath: "static/close_black.png",
            returnImgWidth: "",
            returnImgHeight: "",
        },
        setWebNavUi: {
            bgColor:"",//协议页面导航栏背景色
            textColor: "",//协议页面导航栏Title字体颜色
            textSize: "",//协议页面导航栏Title字体大小（单位：sp）
            returnImgPath: ""//协议页面导航栏返回按钮图标地址，必须是assets目录中的
        },
        setLogoUi: {
            imgPath: "static/mytel_app_launcher.png",
            width: "",
            height: "",
            offsetY: logoOffsetY,
            offsetY_B: "",
        },
        setSloganUi: {
            text: "xxxx",
            textColor: "#000000",
            textSize: "20",
            offsetY: sloganOffsetY,
            offsetY_B: "",
        },
        setNumberUi: {
            textColor: "#000000",
            textSize: "30",
```

```
            offsetY: "",
            offsetY_B: "",
            offsetX: "",
            //3 左对齐
            //5 右对齐
            //默认居中
            layoutGravity: "",
        },
        setLoginBtnUi: {
            text: "",
            textColor: "",
            textSize: "",
            imgPath: "",
            offsetX: "",
            offsetY: "",
            offsetY_B: "",
            width: "",
            height: "",
            marginLeftAndRight: "",
            layoutGravity: "",
            toastHidden: "false",
        },
        setSwitchUi: {
            text: "",
            textColor: "",
            textSize: "20",
            offsetY: "",
            offsetY_B: "",
        },
        setCheckBoxUi: {
            defaultChecked: "true",
            width: "",
            height: "",
            //checkBox未勾选图片
            unCheckedImgPath: "",
            //checkBox已勾选图片
            checkedImgPath: "",
        },
        setAppPrivacyOne: {
            title: "用户协议",
            url: "www.example.com",
        },
        setAppPrivacyTwo: {
            title: "隐私政策",
            url: "www.example.com",
        },
        setAppPrivacyThree: {
            title: "服务协议",
            url: "www.example.com",
        },
        setPrivacyUi: {
            beforeText: "",
            endText: "",
            baseColor: "",
```

```
            protocolColor: "",
            vendorPrivacyPrefix: "",
            vendorPrivacySuffix: "",
            offsetX: "",
            offsetY: "",
            offsetY_B: "",
            textSize: "",
            marginLeftAndRight: "",
            gravity: "",
            layoutGravity: ""
        },
        setDialogTheme: {
            alpha: "",
            width: "300",
            height: "400",
            offsetX: "0",
            offsetY: "0",
            isBottom: "false"
        }
    }, //授权页添加自定义控件元素
    widgets: {
        widget1: {
            widgetId: "one",
            type: "TextView",
            left: "",
            top: widgetOffsetY,
            right: "",
            bottom: "",
            width: "",
            height: "30",
            textContent: "切换其他登录方式",
            textSize: "15",
            textColor: "#cc0000",
            // 0位置是在导航栏下面的区域
            // 1位置是在导航栏上
            // 2位置是在号码栏水平线上
            locate: "0"
        },
        widget2: {
            widgetId: "two",
            type: "ImageView",
            left: "30",
            top: widgetOffsetY,
            right: "",
            bottom: "",
            width: "",
            height: "30",
            backgroundImgPath: "static/weixin.png",
            locate: "1"
        }
    }
}
```

- iOS配置项说明。调用示例：

```
{
    uiConfig: {
        // UIInterfaceOrientationMaskPortrait: 2
    // UIInterfaceOrientationMaskLandscapeLeft: 16
    // UIInterfaceOrientationMaskLandscapeRight: 8
    // UIInterfaceOrientationMaskPortraitUpsideDown: 4
    supportedInterfaceOrientations: "2", //设置屏幕方向，默认为竖屏，其他方向请参考上面列举
        autoHideLoginLoading: "false", //在获取登录Token成功后，是否自动隐藏loading动画，
默认隐藏，如果设置为不隐藏，需要手动调用 aLiSDKModule.hideLoginLoading() 来隐藏loading动画
        setNavHidden: "true", //是否隐藏导航栏，默认不隐藏
        setLogoHidden: "true", //是否隐藏中间的Logo图片，默认不隐藏
        setSloganHidden: "true", //是否隐藏slogan，默认不隐藏
        setSwitchHidden: "true", //是否隐藏切换其他方式按钮，默认不隐藏
        setCheckboxHidden: "true", //是否隐藏check box，默认不隐藏
        presentDirection: "2", //授权页动画方向，0:从底部弹出，1:从右边弹出，2:从上面弹出，3
:从左边弹出
        animationDuration: "2.5", //授权页动画时间，设置为0则为关闭动画
        prefersStatusBarHidden: "true", //是否隐藏状态栏，默认不隐藏
        // UIStatusBarStyleDefault: 0
    // UIStatusBarStyleLightContent: 1
    // UIStatusBarStyleDarkContent: 3
        preferredStatusBarStyle: "0", //状态栏样式，参考上面枚列举
        //导航栏相关设置
        setNavUi: {
            bgColor: "#FF8247",
            text: "一键登录^-^",
            textColor: "#FFFFFF",
            textSize: "17",
            returnImgHidden: "true", //是否隐藏返回按钮，默认不隐藏
            returnImgPath: "static/close_black.png", //自定义返回按钮图片
            returnImgX: "15",
            returnImgY: "10",
            returnImgWidth: "44",
            returnImgHeight: "44"
        },
        //logo相关设置
        setLogoUi: {
            imgPath: "static/mytel_app_launcher.png",
            x: "100",
            y: "30",
            width: "30",
            height: "60"
        },
        //slogan相关设置
        setSloganUi: {
            text: "由阿里云通信提供服务",
            textColor: "#FF8247", //必须在text不为空的情况下设置的该属性才会生效
            textSize: "12", //必须在text不为空的情况下设置的该属性才会生效
            x: "10",
            y: "200",
            width: "300",
            height: "20"
        },
        //掩码相关设置
```

```
        setNumberUi: {
            textColor: "#FF8247",
            textSize: "17",
            x: "10",
            y: "320"
        },
        //登录按钮相关设置
        setLoginBtnUi: {
            text: "一键登录^-^",
            textColor: "#551A8B",
            textSize: "15",
            activeImgPath: "static/loginBtn_active.png", //check box勾选时按钮的颜色，注：只有activeImgPath、invalidImgPath、hightedImgPath全都设置，并且有效情况下才会生效
            invalidImgPath: "static/loginBtn_invalid.png", //check box未勾选时按钮的颜色，注：只有activeImgPath、invalidImgPath、hightedImgPath全都设置，并且有效情况下才会生效
            hightedImgPath: "static/loginBtn_highted.png", //check box勾选时，按压按钮时的颜色，注：只有activeImgPath、invalidImgPath、hightedImgPath全都设置，并且有效情况下才会生效
            x: "10",
            y: "350",
            width: "300", //宽度必须大于屏幕的一半
            height: "30" //高度不能小于20
        },
        //切换其他登录按钮相关设置
        setSwitchUi: {
            text: "切换其他登录方式^-^",
            textColor: "#551A8B",
            textSize: "12",
            x: "10",
            y: "420",
            width: "300",
            height: "20"
        },
        //check box相关设置
        setCheckBoxUi: {
            defaultChecked: "true", //check box默认是否勾选
            unCheckedImgPath: "static/checkbox0", //必须同时设置checkedImgPath有效，该属性才会生效
            checkedImgPath: "static/checkbox1", //必须同时设置unCheckedImgPath有效，该属性才会生效
            width: "30" //默认宽度为17 pt
        },
        //授权页底部自加协议
        setAppPrivacyOne: {
            title: "《协议1》",
            url: "www.example.com"
        },
        //只有设置了setAppPrivacyOne生效，setAppPrivacyTwo才会生效
        setAppPrivacyTwo: {
            title: "《协议2》",
            url: "www.example.com"
        },
        //只有设置了setAppPrivacyOne、setAppPrivacyTwo生效，setAppPrivacyTwo才会生效
        setAppPrivacyThree: {
```

```
            title: "《协议3》",
            url: "www.example.com"
        },
        //设置协议相关
        setPrivacyUi: {
            beforeText: "协议整体前缀",
            endText: "协议整体后缀",
            baseColor: "#8B8878",
            protocolColor: "#FFB5C5",
            textSize: "13", //协议文字大小，小于12不生效
            vendorPrivacyPrefix: "《", //供应商协议前缀
            vendorPrivacySuffix: "》", //供应商协议后缀
            alignment: "1", //协议文字对齐方式，0：左对齐，1：中间对齐，2：右对齐
            x: "20",
            y: "800",
            width: "200" //宽度小于check box的宽度时不生效（该宽度为check box宽度 + 协议文本宽度），高度SDK内部自适应，外面设置无效
        },
        //设置协议详情页导航栏相关
        setWebNavUi: {
            bgColor: "#CD1076",
            textColor: "#EECFA1",
            textSize: "30",
            returnImgPath: "/static/close_black.png"
        },
        //设置弹窗样式相关
        setDialogTheme: {
            alpha: "0.2", //弹框背部蒙层透明度
            radius: ["10", "10", "10", "10"], //设置弹窗四个角的弧度
            x: "10",
            y: "160",
            width: "300",
            height: "650"
        }
    },
    //自定义控件
    widgets: [{
            widgetId: "widgetId-001", //控件id
            type: "ImageView", //控件类型，ImageView（图片）
            imgPath: "static/qq.png",
            x: "10",
            y: "60",
            width: "100",
            height: "100",
            // UIViewContentModeScaleToFill: 0
            // UIViewContentModeScaleAspectFit: 1
            // UIViewContentModeScaleAspectFill: 2
            // UIViewContentModeRedraw: 3
            // UIViewContentModeCenter: 4
            // UIViewContentModeTop: 5
            // UIViewContentModeBottom: 6
            // UIViewContentModeLeft: 7
            // UIViewContentModeRight: 8
            // UIViewContentModeTopLeft: 9
```

```
            // UIViewContentModeTopRight: 10
            // UIViewContentModeBottomLeft: 11
            // UIViewContentModeBottomRight: 12
            contentMode: "1", //内容填充方式，参考上面列举
            backgroundColor: "#EE6A50",
            masksToBounds: "true",
            cornerRadius: "5",
            borderColor: "#8B7D6B",
            borderWidth: "1.0"
        },
        {
            widgetId: "widgetId-002",
            type: "Button", //Button（按钮）
            backgroundImage: "static/qq.png",
            // UIControlContentVerticalAlignmentCenter: 0
            // UIControlContentVerticalAlignmentTop: 1
            // UIControlContentVerticalAlignmentBottom: 2
            // UIControlContentVerticalAlignmentFill: 3
            verticalAlignment: "0", //标题竖直对齐方式，参考上面列举
            // UIControlContentHorizontalAlignmentCenter: 0
            // UIControlContentHorizontalAlignmentLeft: 1
            // UIControlContentHorizontalAlignmentRight: 2
            // UIControlContentHorizontalAlignmentFill: 3
            // UIControlContentHorizontalAlignmentLeading  API_AVAILABLE(ios(11.0), t
vos(11.0)): 4
            // UIControlContentHorizontalAlignmentTrailing API_AVAILABLE(ios(11.0), t
vos(11.0)): 5
            horizontalAlignment: "0", //标题水平对齐方式，参考上面列举
            textContent: "这是一个按钮",
            textSize: "13",
            textColor: "#FFFFFF",
            numberOfLines: "0",
            x: "10",
            y: "200",
            width: "100",
            height: "100",
            backgroundColor: "#EE6A50",
            masksToBounds: "true",
            cornerRadius: "5",
            borderColor: "#8B7D6B",
            borderWidth: "1.0"
        },
        {
            widgetId: "widgetId-003",
            type: "Label", //Label（文本）
            // NSTextAlignmentLeft: 0
            // NSTextAlignmentCenter: 1
            // NSTextAlignmentRight: 2
            alignment: "1", //文字对齐方式，参考上面枚举
            textContent: "这是一串文本信息xxxx这是一串文本信息xxxx这是一串文本信息xxxx",
            textSize: "13",
            textColor: "#FFFFFF",
            numberOfLines: "0",
            x: "10",
            y: "450"
```

```
            y: "450",
            width: "100",
            height: "100",
            backgroundColor: "#EE6A50",
            masksToBounds: "true",
            cornerRadius: "5",
            borderColor: "#8B7D6B",
            borderWidth: "1.0"
        }
    ]
}
```

本机号码校验方法说明

1. （可选）加速校验接口

   加速本机号码校验Token的获取。

   > **说明**
   > - 由于加速方法需要1~3秒的时间取得临时凭证，建议在调用获取本机号码校验Token方法2~3秒前调用预取号方法。
   > - 请勿频繁的调用加速拉起授权页方法，不得与拉起授权登录页同时调用或在其后调用。
   > - 进入App仅获取本机号码校验Token的场景无需调用此接口。

   Object参数说明：

参数	类型	描述
timeout	Number	设置接口超时时间（单位：毫秒）。

   Callback参数说明：

参数	类型	描述
resultCode	String	Code值，600000表示接口调用成功，更多Code值请参见SDK返回码。
msg	String	相关提示信息。

   调用示例：

```
aLiSDKModule.accelerateLoginPage(5000, result => {
    if ("600000" == result.resultCode) {
        console.log("加速成功")
    }
})
```

2. 获取本机号码校验Token

   Object参数说明：

参数	类型	描述
timeout	Number	设置接口超时时间（单位：毫秒）。

Callback参数说明：

参数	类型	描述
resultCode	String	Code值，600000表示接口调用成功，更多Code值请参见SDK返回码。
msg	String	相关提示信息。
token	String	最终进行号码校验的Token，只有在resultCode值为600000的时候该字段才会有值。

调用示例

```
aLiSDKModule.getVerifyToken(5000, result => {
    if ("600000" == result.resultCode) {
        console.log("获取本机号码校验Token成功，接下面需要拿手机号和Token去服务端进行校验，SDK服务到此结束")
    }
})
```

更多方法说明

- 获取号码认证SDK版本号

  Callback参数说明：

返回值	类型	描述
version	String	号码认证SDK版本号

  调用示例：

  ```
  aLiSDKModule.getVersion(version => {
      console.log("当前SDK版版本号为：" + version)
  })
  ```

- 环境检查

  Object参数说明：

参数	类型	描述
authType	Number	检查类型。取值： ○ 1：检查号码认证环境。 ○ 2：检查一键登录环境。

Callback参数说明：

参数	类型	描述
resultCode	String	Code值，600000表示接口调用成功。环境支持一键登录或本机号码校验，更多Code值请参见SDK返回码。
msg	String	相关提示信息。

调用示例：

```
aLiSDKModule.checkEnvAvailable(2, result => {
    console.log(JSON.stringify(result))
})
```

- 获取默认上网卡运营商

  Callback参数说明：

返回值	类型	描述
carrierName	String	当前运营商。取值： ○ 中国移动：CMCC。 ○ 中国联通：CUCC。 ○ 中国电信：CTCC。

  调用示例：

```
aLiSDKModule.getCurrentCarrierName(carrierName => {
    console.log("当前运营商为：" + carrierName)
})
```

SDK返回码

返回码	返回码描述	解决方案
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。	升级SDK版本。您可前往号码认证服务控制台下载最新版SDK。
600005	手机终端不安全。	切换到其他登录方式。
600007	未检测到SIM卡。	提示用户检查SIM卡后重试。
600008	移动数据网络未开启。	提示用户开启移动数据网络后重试。

返回码	返回码描述	解决方案
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。	检查数据网络环境后重试，若还未解决问题，您可通过切换飞行模式、重启手机或切换到其他登录方式的操作解决问题。
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，该功能调用次数已达上限。	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。	密钥未设置或者设置错误，请先检查密钥信息，如密钥无问题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。
600023	加载自定义控件异常。	检查自定义控件添加是否正确。
600024	终端环境检查支持认证。	无。
600025	终端检测参数错误。	检查传入参数类型与范围是否正确。
600026	授权页已加载时不允许调用加速或预取号接口。	检查是否有授权页拉起后，调用preLogin或者accelerateAuthPage接口的行为不被允许。

除阿里云SDK返回码外，运营商错误码详情请参见运营商SDK错误码。

2.3.4. 运营商SDK错误码

本文为您介绍了在调用号码认证服务一键登录和本机号码校验功能的接口时，运营商返回的SDK错误码。

运营商	错误码	描述
	103000	成功。
	102507	登录超时（授权页点登录按钮时）。
	103101	请求签名错误。
	103102	包签名或Bundle ID错误。
	103111	网关IP错误或错误的运营商请求。

运营商	错误码	描述
中国移动	103119	AppID不存在。
	103211	其他错误，请提工单联系工程师。
	103412	无效的请求有加密方式错误、非JSON格式和空请求等。
	103414	参数校验异常。
	103511	服务器IP白名单校验失败。
	103811	Token为空。
	103902	短时间内重复登录，造成script失效。
	103911	Token请求过于频繁，10分钟内获取Token且未使用的数量不超过30个。
	104201	Token重复校验失败、失效或不存在。
	105018	使用了本机号码校验的Token获取号码，导致Token权限不足。
	105019	应用未授权。
	105021	已达当天取号限额。
	105302	AppID不在白名单。
	105313	非法请求。
	200005	用户未授权（READ_PHONE_STATE）。
	200010	无法识别SIM卡或没有SIM卡（Android）。
	200020	用户取消登录。
	200021	数据解析异常。
	200022	无网络。
	200023	请求超时。
	200024	数据网络切换失败。
	200025	位置错误（一般是线程捕获异常、socket、系统未授权移动数据网络权限等，请提工单联系工程师）。
	200026	输入参数错误。
	200027	未开启移动数据网络或网络不稳定。
	200028	网络请求出错。

运营商	错误码	描述
	200038	异网取号网络请求失败。
	200039	异网取号网关取号失败。
	200048	用户未安装SIM卡。
	200050	EOF异常。
	200061	授权页面异常。
	200064	服务端返回数据异常。
	200072	CA根证书校验失败。
	200082	服务器繁忙。
	200086	ppLocation为空。
	200087	仅用于监听授权页成功拉起。
	200096	当前网络不支持取号。

运营商	错误码，msg	描述
	0	表示请求成功。
	-10008	JSON转换失败。
	1，请求超时	请求超时。
	1，私网IP查找号码失败	私网IP查找号码失败。
	1，私网IP校验错误	私网IP校验错误。
	1，源IP鉴权失败	源IP鉴权失败。
	1，获取鉴权信息失败	获取鉴权信息失败。
	1，获得的手机授权码失败	一般是由于请求SDK超时导致的失败。
	1，网关取号失败	网关取号失败。
	1，网络请求失败	网络请求失败。
	1，验签失败	签名校验失败。
	1，传入code和AppID不匹配	传入code和AppID不匹配。
	1，似乎已断开与互联网的链接	网络不稳定导致连接断开。

运营商	错误码，msg	描述
中国联通	1，此服务器的证书无效	连接到不安全的服务器，提工单联系工程师。
	1，PIP校验不通过	提工单联系工程师。
	1，网关并发连接数受限	提工单联系工程师。
	1，目前不允许数据链接	提工单联系工程师。
	1，无法连接到服务器	提工单联系工程师。
	1，系统内部错误	提工单联系工程师。
	1，取号网关内部错误	取号网关内部错误，提工单联系工程师。
	1，connect address error	连接地址错误，一般是由于超时导致失败。
	1，select socket error	选择socket错误。
	1，handshake failed	握手失败。
	1，decode ret_url fail	URL解码失败。
	1，connect error	连接错误。
	1，read failed	提工单联系工程师。
	1，response null	无响应，提工单联系工程师。
	1，未知错误	提工单联系工程师。
	2，请求超时	接口请求耗时超过timeout设定的值。

运营商	错误码	描述
	-64	Permission-denied（无权限访问）。
	-65	API-request-rates-Exceed-Limitations（调用接口超限）。
	-10001	取号失败，mdn为空。
	-10002	参数错误。
	-10003	解密失败。
	-10004	IP受限。
	-10005	异网取号回调参数异常。
	-10006	mdn取号失败，且属于电信网络。

运营商	错误码	描述
中国电信	-10007	重定向到异网取号。
	-10008	超过预设取号阈值。
	-10009	时间戳过期。
	-10013	Perator_unsupported，提工单联系工程师。
	-20005	Sign-invalid（签名错误）。
	-8001	网络异常，请求失败。
	-8002	请求参数错误。
	-8003	请求超时。
	-8004	移动数据网络未开启。
	-8010	无网络连接（网络错误）。
	-720001	Wi-Fi切换4G请求异常。
	-720002	Wi-Fi切换4G超时。
	80000	请求超时。
	80001	网络连接失败、网络链接已中断、Invalid argument、目前不允许数据连接。
	80002	响应码错误404。
	80003	网络无连接。
	80005	socket超时异常。
	80007	IO异常。
	80008	No route to host.
	80009	Nodename nor servname provided, or not known.
	80010	Socket closed by remote peer.
	80800	Wi-Fi切换超时。
	80801	Wi-Fi切换异常。

2.4. H5本机号码校验

2.4.1. Demo体验

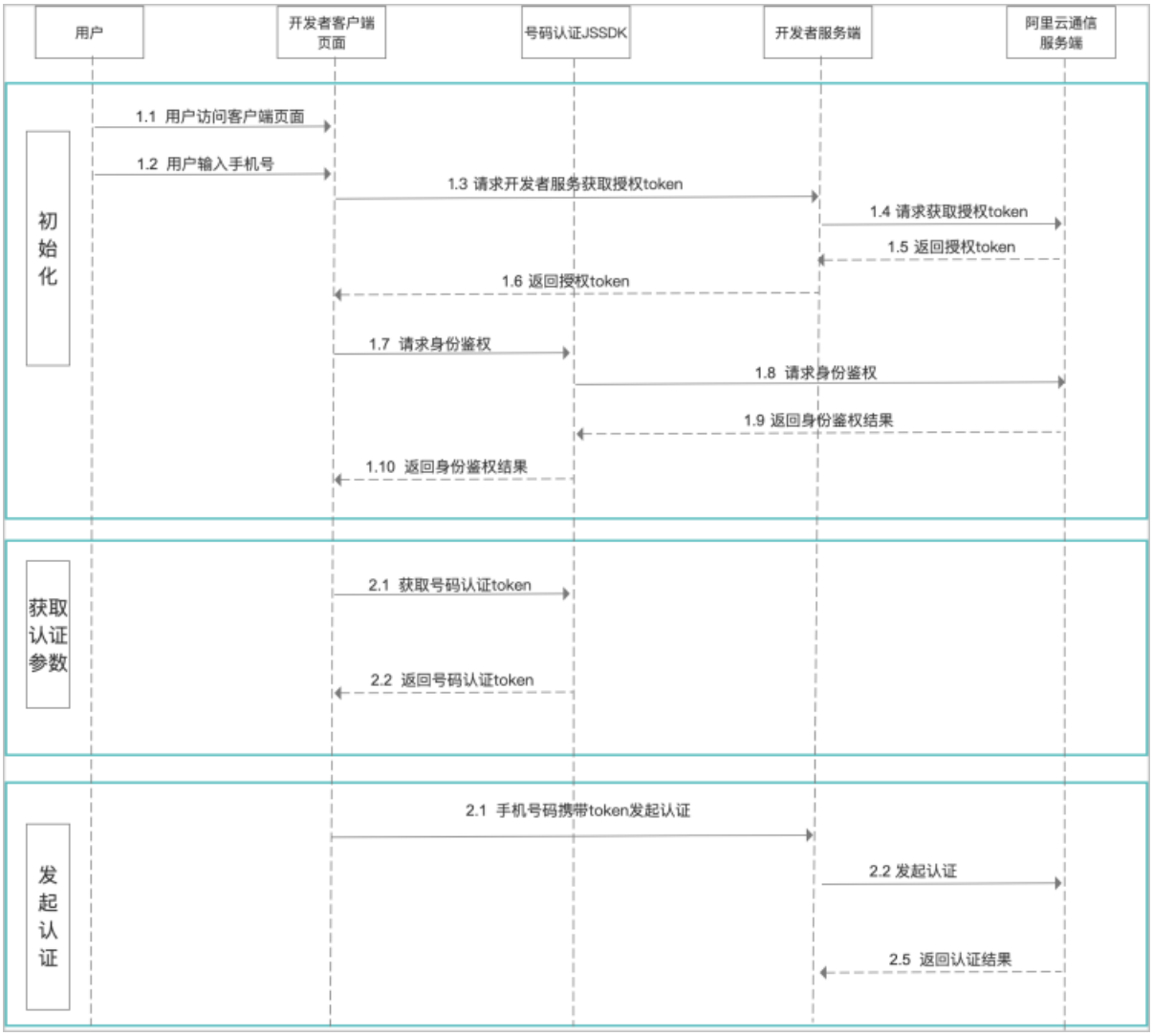
本文提供H5本机号码校验的Demo体验。

请扫描下方二维码，体验H5本机号码校验功能。

2.4.2. 接入概述

本文为您介绍了H5本机号码校验的系统交互流程。开发者需要在H5页面集成JSSDK，并在服务端完成API对接。

H5本机号码认证的系统交互流程主要分为三个步骤：初始化、获取认证参数、发起认证。



1. 初始化。
   i. 用户访问客户端页面并输入手机号。
   ii. 开发者客户端通过开发者服务端向阿里云通信服务端请求获取授权Token（包括业务鉴权accessToken和API鉴权jwtToken两个参数）。
   iii. H5页面请求发起身份鉴权。
2. 获取认证参数。
   i. 身份鉴权通过，请求号码认证JSSDK中的获取号码认证Token方法，获取号码认证Token。
   ii. 返回号码认证Token。
3. 发起认证。
   i. H5向其服务端发起认证请求。
   ii. 开发者服务端调用VerifyPhoneWithToken接口获取认证结果，判断用户输入的手机号码与用户终端当前访问网络的手机号码是否一致。

2.4.3. H5端JSSDK集成

本文为您介绍H5端JSSDK的集成方法及接口的功能示例。

前提条件

- 确保您已开通了号码认证服务，并成功创建了对应的认证方案。详情请参见H5本机号码校验使用流程。
- 确保您的终端设备已关闭Wi-Fi连接且开启了SIM卡的4G移动数据网络（支持中国联通、中国移动的3G网络，但接口耗时会增加）。

集成JSSDK

您需要在H5页面中集成号码认证服务的JSSDK，并在服务端完成API对接。有两种集成JSSDK的方法，您可任选其中一种：

- 静态资源引入。登录号码认证产品控制台，在**标准版**选项卡下载并解压**H5端JSSDK**，将解压后文件中的js文件引入项目目录。

```
<script type="text/javascript" charset="utf-8" src="xxx/numberAuth-web-sdk-1.0.3.js"></sc
ript>
```

- npm引入。下载npm资源：npm资源。

```
//加载npm资源
npm i aliyun_numberauthsdk_web
```

为保证服务正常使用，在您的HTML代码head标签中增加以下代码：

```
 <meta name="referrer" content="origin">
```

SDK接口功能示例及参数说明

- 初始化实例

```
// 初始化实例
var PhoneNumberServer = window.PhoneNumberServer; //引用静态资源包
//引用npm包
import { PhoneNumberServer } from 'aliyun_numberauthsdk_web';  //ES6
var PhoneNumberServer = require('aliyun_numberauthsdk_web'); //ES5
var phoneNumberServer = new PhoneNumberServer();
```

设置SDK是否开启日志。开启后会在控制台打印更多内容便于排查问题。

```
phoneNumberServer.setLoggerEnable();
```

参数说明：

参数	说明
isEnable	默认**true**（开启日志）。

- 获取号码认证SDK版本号

```
var sdkVersion = phoneNumberServer.getVersion();  //返回SDK版本号，例如'1.0.0'
```

- 身份鉴权

```
//调用之前先去用户服务端获取accessToken和jwtToken
phoneNumberServer.checkAuthAvailable({
    phoneNumber: '151********',
    accessToken: 'XXXXXXXXxx',
    jwtToken:'******',
    success:function(res){
      console.log(res);
    }
    error: function(res){
    }
});
```

参数名称及说明：

参数名称	说明
phoneNumber	本机号码。
AccessToken	号码认证业务鉴权Token由阿里云对外暴露的getAuthToken接口生成。
JwtToken	API鉴权Token由阿里云对外暴露的getAuthToken接口生成。
timeout	可选参数（默认为10s）。
success	成功回调。
error	失败回调。

- 获取本机号码校验Token

> 注意 身份鉴权成功后才可调用获取Token接口。

```
phoneNumberServer.getVerifyToken({
    success:function(res){
    console.log(res)
    }
    error: function(res){
    }
});
```

参数名称及说明：

参数名称	说明
success	成功回调。
error	失败回调。

返回参数及说明：

返回参数	说明
code	成功600000。
spToken	运营商Token。
content	失败时运营商返回的内容。

返回码及说明

返回码	说明
600000	成功返回code。
600004	方案号不存在。
600009	无法判断运营商。
600010	未知异常。
600011	获取Token失败。
600013	运营商维护升级，该功能不可用。
600014	运营商维护升级，该功能调用次数已达上限。
600015	调用接口超时。
600025	接入方身份信息校验失败。
600008	环境错误（未在移动数据网络下使用或未使用手机浏览器）。
600028	入参错误（未传入手机号码，accessToken，jwtToken）。

除阿里云SDK返回码外，运营商错误码详情请参见运营商SDK错误码。

2.4.4. 运营商SDK错误码

本文为您介绍了在调用号码认证服务H5本机号码校验功能的接口时，运营商返回的SDK错误码。

运营商	错误码	描述
	500	取号失败，可能原因有以下几种： • 传入的Referrer和创建方案号的URL不一致。 • 当前联网的可能不是中国移动的移动数据。
	0	请求异常。

运营商	错误码	描述
中国移动	999999	系统错误。
	130032	参数解析错误。
	130018	签名验证失败。
	130030	参数无效。
	110028	AppID不存在。

运营商	错误码	描述
中国联通	100001	应用鉴权错误或获取Token失败。
	100002	未进行初始化操作。
	100003	请求超时。
	100009	未知错误。
	200001	初始化失败。

错误详细码表

运营商	错误详细码（RespCode）	描述
中国联通	102	客户端类型错误。
	104	clientId错误。
	106	请求时间超时。
	107	鉴权信息错误。
	108	应用签名错误。
	110	Referrer未报备。
	111	网络环境错误（非移动数据环境，例如Wi-Fi）。
	113	客户端无权限。
	1002	网关错误。
	1003	预取号错误。
	1004	AccessCode错误。
	1011	数据解析错误。

运营商	错误详细码（RespCode）	描述
	1012	网络环境错误（1012是公网IP错误）。
	1013	网络环境错误（私网IP错误）。

运营商	错误码	描述
中国电信	30002	无法识别用户网络，返回两个重定向异网取号地址。
	30901	Code换Tokenfail（Code已经使用或者Code超过10分钟未使用）。
	-64	没有权限（天翼账号平台未授权应用访问权限）。
	-20005	签名非法。
	-10001	取号失败。
	-10002	参数错误。
	-10003	解密失败。
	-10004	无效的IP。
	-10005	异网授权回调参数异常。
	-10006	授权失败且属于中国电信网络。
	-10007	重定向到异网取号。
	-10008	超过预设取号阈值。
	-10009	时间戳过期。
	-20005	签名非法。
	-20006	应用不存在。
	-20007	公钥数据不存在。
	-20100	内部解析错误。
	-20102	加密参数解析失败。
	-30001	非法的时间戳。
	51002	参数为空。
	51114	无法获取手机号数据。

2.5. 活体认证

2.5.1. 接入概述

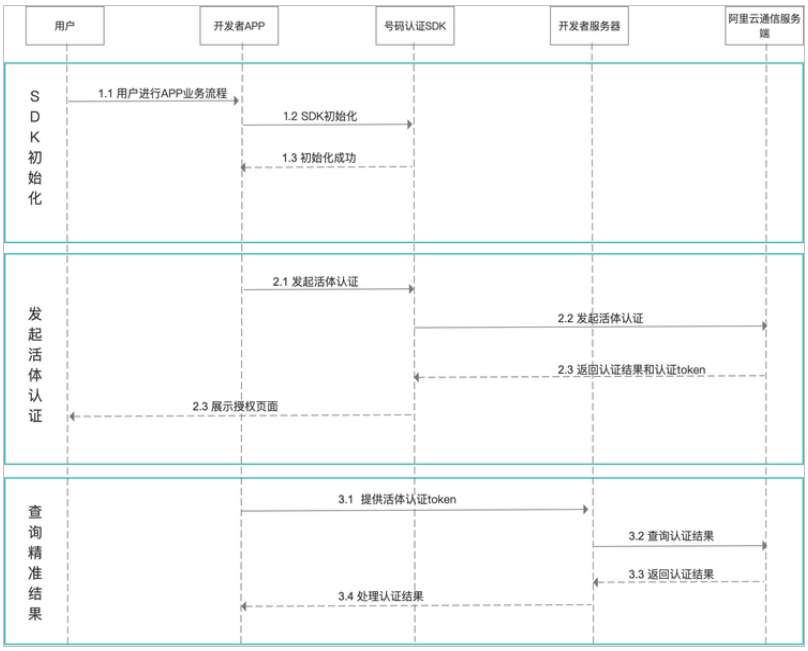
本文为您介绍了活体认证的系统交互流程。

前提条件

- 您已开启活体认证功能。详情请参考活体认证使用流程。
- 您已成功创建了认证方案。详情请参见步骤二：添加认证方案。

活体认证的交互流程

活体认证的交互流程主要分为三个步骤：SDK初始化、发起活体认证、查询精准结果。



1. SDK初始化。为SDK设置密钥。
2. 调用活体认证接口。您可提前调用加速活体认证accelerateLifeBodyVerify接口，进行初始化加速（加速缓存有效期30分钟）。
    i. 开发者App通过开发者服务器向阿里云通信服务端发起活体认证。
    ii. 无论认证成功或失败，阿里云通信服务端都会返回认证结果和认证Token。
3. 查询精准结果。
    i. 开发者App提供活体认证Token至开发者服务端。
    ii. 阿里云通信服务端查询并返回认证结果。
    iii. 开发者服务端将返回的认证结果进行逻辑处理，再返回给开发者App。

2.5.2. 客户端接入

2.5.2.1. iOS客户端接入

本文为您介绍了iOS客户端如何接入活体认证功能。

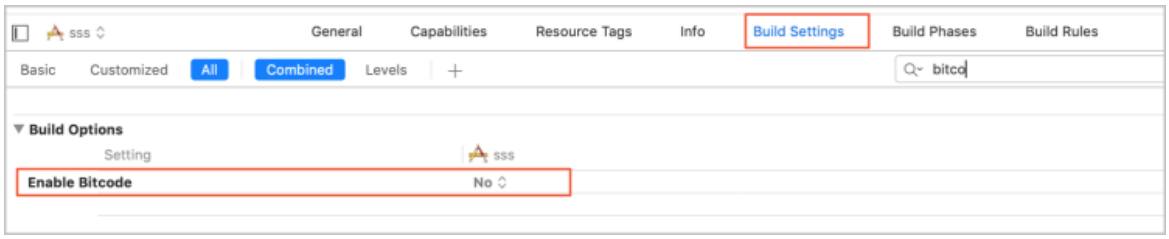
前提条件

- 活体认证应用必须运行在iOS 9.0及以上版本的平台上。
- 您的应用已集成支付宝mPaaS SDK。

步骤一：开发环境配置

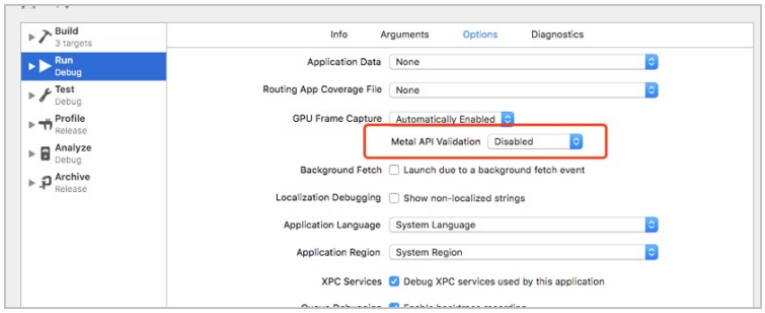
1. 下载并解压iOS SDK。登录号码认证产品控制台，在**概览**页面右侧，选择**增强版**选项卡，下载并解压iOS SDK。
2. 在*info.plist*文件中配置摄像头权限请求。

3. 将Xcode编译设置的**Build Settings > Build Options > Enable Bitcode**设置为No。



4. 在Xcode编译设置的**Linking > Other Linker Flags**中，添加设置 **-ObjC -framework "BioAuthAPI" -lxml2**。如果您的工程已设置了**-force_load**选项，则需要加入**-force_load <framework path>/AliyunOSSiOS**。

5. 将Xcode调试设置的**Edit Scheme > Run > Options**的**Metal API Validation**设置为Disabled。



步骤二：配置依赖

在Xcode添加Link Binary With Libraries、1个号码认证基础包、12个活体认证依赖包和额外系统库依赖。具体如下表：

号码认证基础包	活体认证依赖包	系统库依赖

号码认证基础包	活体认证依赖包	系统库依赖
ATAuthSDK	• APPSecuritySDK • MPRemoteLogging • AliyunOSSiOS • BioAuthEngine • ZolozUtility • AliyunIdentityManager • APBToygerFacade • BioAuthAPI • ToygerService • ZolozSensorServices • ZolozOpenPlatformBuild • YunCeng	• CoreGraphics.framework • Accelerate.framework • SystemConfiguration.framework • AssetsLibrary.framework • CoreTelephony.framework • QuartzCore.framework • CoreFoundation.framework • CoreLocation.framework • ImageIO.framework • CoreMedia.framework • CoreMotion.framework • AVFoundation.framework • WebKit.framework • libresolv.tbd • libz.tbd • libc++.tbd • libc++.1.tbd • libc++abi.tbd • AudioToolbox.framework • CFNetwork.framework • MobileCoreServices.framework • libz.1.2.8.tbd • AdSupport.framework

步骤三：拷贝资源文件

选择**TARGETS**，单击**Build Phases > Copy Bundle Resources**，添加1个号码认证Bundle和4个活体认证相关Bundle。

- 号码认证Bundle。ATAuthSDK.bundle所在位置如下图所示：

- 活体认证相关Bundle。
  - APBToygerFacade.bundle：所在位置在APBToygerFacade.framework中。
  - ToygerService.bundle：所在位置在ToygerService.framework中。
  - BioAuthEngine.bundle：所在位置在BioAuthEngine.framework中。
  - OCRXMedia.bundle：所在位置在AliyunIdentityManager.framework中。

功能示例

- 活体认证加速

```
/**
 * 函数名：accelerateLifeBodyVerify
 * @brief活体认证加速接口，提前调用可为lifeBodyVerify拉起活体认证页面缩短时间
 */
- (void)accelerateLifeBodyVerify;
```

- 开始活体认证

```
/**
 * 函数名：lifeBodyVerifyWithTimeout:controller:complete:
 * @brief开始活体认证
 * @param timeout拉起认证页超时时间（单位：s），默认为20.0s
 * @param controller唤起活体认证页的容器，内部会对其进行验证，检查是否符合条件
 * @param complete认证过程中的失败及认证有结果会调用该block，"resultDic"里面的"resultCode"值请参见PNSReturnCode，如下：
 *          700005（活体认证页准备启动，可结束loading）、700000（用户主动取消操作）
 *          600017（密钥解析错误）、600034（不合法的密钥）、600035（状态繁忙）、600033（功能不可用）、
 *          600036（业务停机）、600000（认证成功）、600015（唤起认证页超时）、600030（认证失败）、
 *          600031（网络错误）、600032（客户端设备时间错误）
 */
- (void)lifeBodyVerifyWithTimeout:(NSTimeInterval)timeout
                       controller:(UIViewController *_Nonnull)controller
                         complete:(void (^_Nullable)(NSDictionary * _Nonnull resultDic))complete;
```

SDK使用示例

- 引入头文件

```
#import <ATAuthSDK/ATAuthSDK.h>
```

- 初始化SDK

```
[[TXCommonHandler sharedInstance] setAuthSDKInfo:<#请填写您的密钥，从控制台上可复制#>
                                        complete:^(NSDictionary * _Nonnull resultDic) {
    NSLog(@"设置密钥结果：%@", resultDic);
}];
```

- 拉起活体认证页面开始认证

```
//这里开始等待动画
[[TXCommonHandler sharedInstance] lifeBodyVerifyWithTimeout:20.0
                                                 controller:self
                                                   complete:^(NSDictionary * _Nonnull res
ultDic) {
    NSString *resultCode = [resultDic objectForKey:@"resultCode"];
    if ([PNSCodeLiftBodyVerifyReadyStating isEqualToString:resultCode]) {
        //开始拉起授权页，暂停等待动画
    } else {
        if ([PNSCodeSuccess isEqualToString:[resultDic objectForKey:@"resultCode"]]) {
            NSString *token = [resultDic objectForKey:@"token"];
            //将Token带去服务端校验
        } else {
            //其他回调时也需暂停您的等待动画。比如请求失败的情况
        }
    }
}];
```

SDK返回码

返回码	描述	解决方案
600000	认证成功	无。
600030	认证失败	根据返回msg排查对应错误。
600031	网络错误	检查手机网络后重试。
600032	客户端设备时间错误	检查手机时间后重试。
600033	功能不可用	前往控制台开通对应功能。
600034	不合法的SDK密钥	开通相应功能后，前往控制台复制新密钥。
600035	状态繁忙	不能连续调用认证接口。
600036	业务停机	开通控制台或续费。
700000	取消认证	用户主动取消操作UI事件，用户取消操作。
700005	活体认证页面准备启动	接收到该code可关闭等待动画。

2.5.2.2. Android客户端接入

本文为您介绍了Android客户端如何接入活体认证功能。

前提条件

- 确保您已开通了号码认证服务，并成功创建了对应的认证方案。详情请参见活体认证使用流程。
- 确保您的终端设备已开启SIM卡的4G移动数据（支持中国联通、中国移动的3G网络，但接口耗时会增

加）。

- 支持版本：Android 9.0版本以上支持HTTP配置、com.android.support:support-v4版本高于25.4.0或者vcom.android.support:appcompat-v7版本高于25.4.0。

步骤一：搭建开发环境

1. 下载并解压Android SDK。登录号码认证产品控制台，在**标准版**选项卡，下载并解压Android SDK。
2. 将已解压的SDK包中后缀为aar的文件复制至工程的libs目录下。
3. 在App工程src包中main AndroidManifest.xml增加Activity声明 。

```
 <!--联通电信授权页-->
 <!--如果不需要使用窗口模式，不要使用authsdk_activity_dialog主题，会出现异常动画-->
 <!--如果需要使用authsdk_activity_dialog主题，则screenOrientation一定不能指定明确的方向，
        比如portrait、sensorPortrait，在8.0的系统上不允许窗口模式指定orientation，会发生crash
，需要指定为behind，
        然后在授权页的前一个页面指定具体的orientation-->
<activity
    android:name="com.mobile.auth.gatewayauth.LoginAuthActivity"
    android:configChanges="orientation|keyboardHidden|screenSize"
    android:exported="false"
    android:theme="@style/authsdk_activity_dialog"使用弹窗模式必须添加！！！
    android:launchMode="singleTop" />
<!--协议页面webview-->
<activity
    android:name="com.mobile.auth.gatewayauth.activity.AuthWebVeiwActivity"
    android:configChanges="orientation|keyboardHidden|screenSize"
    android:exported="false"
    android:launchMode="singleTop"
    android:screenOrientation="behind" />
 <!--移动授权页-->
 <activity
    android:name="com.cmic.sso.sdk.activity.LoginAuthActivity"
    android:configChanges="orientation|keyboardHidden|screenSize"
    android:exported="false"
    android:launchMode="singleTop" />
```

4. 集成组件。

   集成以下三个组件：

   - crashshiled-xx-release.aar：自V2.10.0版本之后，SDK内部的crash防护和收集库可以减少SDK的崩溃率，在SDK发生问题时不会影响接入者的App。

     > **注意** 由于uc的crash收集原理与市面常见crash收集库的原理类似，如果App自身原本就有crash收集能力，建议将自身的crash库放在前面加载，uc的crash不会替换原有的crash收集能力，而是会形成链式传递，且只会收集跟号码认证SDK相关的crash，其他的crash信息不会收集。

   - logger-xx-release.aar：SDK内部用于收集日志信息和监控告警的。
   - main-xx-release.aar：SDK内部的一些核心工具类，比如线程池、json解析等。

   具体的集成方法是：将3个arr包放在App下的libs目录下，在bulid.gradle中写入以下代码。

```
implementation 'com.ucweb.wpk:crashsdk-java:3.2.0.1'
implementation(name:'auth_number_product-2.12.3-log-online-standard-release', ext:'aar'
)
implementation(name:'crashshield-release', ext:'aar')
implementation(name:'main-release', ext:'aar')
implementation(name:'logger-release', ext:'aar')
```

5. AndroidX适配问题。

   由于SDK使用的是support包，所以使用AndroidX需要在gradle.properties中配置。

```
android.useAndroidX=true
android.enableJetifier=true
```

步骤二：运行Demo

登录号码认证产品控制台，在**概览**页面下载**开发者接入Demo**，再打开开发工具直接运行Demo工程。

步骤三：完成AndroidManifest.xml设置

1. 在App AndroidManifest.xml文件中添加必要的权限支持：

```
<uses-permission android:name="android.permission.INTERNET" /> <!-- 网络访问 -->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" /> <!-- 检查Wi-Fi网络状态 -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" /> <!-- 检查网络状态 -->
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" /> <!-- 切换网络通道 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/> <!-- 本地信息缓存 -->
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" /> <!-- 开关Wi-Fi状态，解决国内机型移动网络权限问题需要 -->
```

2. HTTP配置。在App AndroidManifest.xml里，给Application节点增加usesCleartextTraffic配置。

```
<application
       android:name=".DemoApplication"
       android:icon="@drawable/ic_launcher"
       android:label="@string/app_name"
       android:supportsRtl="true"
       android:theme="@style/AppTheme"
       android:usesCleartextTraffic="true">
```

> **说明** 目前中国移动提供的个别接口为HTTP请求，对于全局禁用HTTP的项目，需要设置HTTP白名单。以下为运营商HTTP接口域名：`demo.aliyundoc.com`，`example.aliyundoc.com`。

功能示例

- 获取认证实例（getInstance）

```
/**
 * 获取号码认证服务示例，此实例为单例，获取多次为同一对象
 * @param context Android上下文
 * @param tokenListener需要实现的获取Token回调
 * @return PhoneNumberAuthHelper
 */
public static PhoneNumberAuthHelper getInstance(Context context,  TokenResultListener tok
enListener)
```

- 检查认证环境（checkAuthEnvEnable）

```
/**
 * SDK环境检查函数，检查终端是否支持号码认证，通过TokenResultListener返回Code
 * type 1：本机号码校验 2：一键登录
 * 600024 终端支持认证
 * 600013 系统维护，功能不可用
 */
 public void checkEnvAvailable(@IntRange(from = 1, to = 2) int type)
```

- 加速活体认证（accelerateLifeBodyVerify)

```
/**
 * 加速活体认证
 */
public void accelerateLifeBodyVerify()
```

- 开始活体认证（lifeBodyVerify）

```
/**
 * 发起活体认证
 * 由于认证启动是个耗时的的操作，因此在认证信息准备完毕后，
 * 启动认证页面前会有一个UI事件回调，开发者可以根据这个事件进行一些交互操作，避免用户等待。
 * 对应的事件Code是{@link ResultCode#CODE_BI_LIFE_BODY_VERIFY_READY_STARTING}
 * @see AuthUIControlClickListener
 * @param timeoutMills超时时间
 * @param context认证页面启动的上下文，要求必须是{@link Activity}，不能是applicationContext
 * @param uiEventListener ui事件回调
 * @param resultListener认证结果回调
 */
public void lifeBodyVerify(final long timeoutMills,
                           final Context context,
                           final AuthUIControlClickListener uiEventListener,
                           final TokenResultListener resultListener)
```

- 加速本机号码校验（accelerateVerify）

```
/**
 * 加速本机号码校验
 *
 * @param overdueTimeMills超时时间
 * @param listener结果回调
 */
public void accelerateVerify(int overdueTime, final PreLoginResultListener listener);
```

- 加速授权页弹出（accelerateLoginPage）

```
/**
* 加速授权页唤起
*
* @param overdueTime预取号有效期
* @param listener预取号回调
*/
public void accelerateLoginPage(final int overdueTime, final PreLogin ResultListener list
ener)
```

- 一键登录唤起授权页（getLoginToken）

```
/**
* 获取登录Token调起一键登录授权页面，在用户授权后获取一键登录的Token
*
* @param totalTimeout超时时间（单位：ms）
*/
public void getLoginToken(final Context context, final int totalTimeout)
```

- 退出授权页（quitLoginPage）

```
/**
* 退出授权认证页
* SDK完成回调之后不会关闭授权页，需要开发者主动调用quitLoginPage退出授权页
*/
public void quitLoginPage()
```

- 关闭授权页loading（hideLoginLoading）

```
/**
* 关闭授权页loading
* SDK完成回调之后不会关闭loading，需要开发者主动调用hideLoginLoading关闭loading
*/
public void hideLoginLoading()
```

其他功能示例

- 返回默认上网卡运营商（getCurrentCarrierName）

```
/**
* 返回默认上网卡运营商
*
* @return CMCC、CUCC、CTCC
*/
public String getCurrentCarrierName()
```

- 使用XML添加自定义控件至一键登录授权页（addAuthRegisterXmlConfig()）

  调用一次addAuthRegisterXmlConfig()方法，XML内绘制的自定义控件全部添加完成。

```
/**
* 添加自定义View
*
* @param xmlConfig
*/
public void addAuthRegisterXmlConfig(AuthRegisterXmlConfig xmlConfig)
```

初始化addAuthRegisterXmlConfig类时需要先调静态内部类Builder()里面的2个方法。

- setLayout()：开发者传入自定义的控件的XML资源ID。
- AbstractPnsViewDelegate()：授权页使用XML添加自定义布局时，可以配合该Delegate类实现XML中相关View的操作，例如事件监听以及动态UI改动等等，当XML对应的View加载后SDK将调用onViewCreated(View)方法通知View已经创建OK，此时可以获取XML中的View并进行相关事件绑定等操作。

> 注意 `onViewCreated(View)` 中返回的View不使用强引用，或用完要及时释放，否则容易造成内存泄漏。

调用示例：

```
mAlicomAuthHelper.addAuthRegisterXmlConfig(new AuthRegisterXmlConfig.Builder()
 .setLayout(R.layout.xxxxxx, new AbstractPnsViewDelegate() {
    @Override public void onViewCreated(View view) {
        //这里返回的View，不建议用强引用，如果要用，请及时释放，否则容易造成内存泄漏
        findViewById(R.id.xxxx).setOnClickListener(new View.OnClickListener() {
            @Override public void onClick(View v) {
                //do something
            }
        });
    }
})
.build());
```

- 添加代码编写的自定义控件至登录授权页（addAuthRegistViewConfig）

```
/**
* 动态添加控件
*
* @param viewID开发者自定义控件名称
* @param viewConfig配置开发者自定义控件的控件来源、位置和处理逻辑
*/
public void addAuthRegistViewConfig(String viewID, AuthRegisterViewConfig viewConfig)
```

> 注意 由于授权页关闭时会清空已注入的AuthRegisterViewConfig，所以每次调用getVerifyToken接口请求授权前，都需对AuthRegisterViewConfig进行一次初始化，具体实现请参见Demo工程。

初始化AuthRegisterViewConfig类时需要先调静态内部类Builder()里面的3个方法。

- setView()：开发者传入自定义的控件，开发者需要提前设置好控件的布局属性，SDK只支持RelativeLayout布局。
- setRootViewId()：设置控件的位置，目前SDK授权页允许在标题栏RootViewId.ROOT_VIEW_ID_TITLE_BAR、授权页空白处RootViewId.ROOT_VIEW_ID_BODY、授权页号码掩码区域RootViewId.ROOT_VIEW_ID_NUMBER 3个位置插入开发者控件。

- setCustomInterface()：设置控件事件。

```
public Builder setCustomInterface(CustomInterface customInterface)
```

调用示例：

```
mAlicomAuthHelper.addAuthRegistViewConfig("switch_acc_tv", new AuthRegisterViewConfig.B
uilder()
.setView(mRL)
.setRootViewId(AuthRegisterViewConfig.RootViewId.ROOT_VIEW_ID_BODY)
.setCustomInterface(new CustomInterface(){
   @Override
    public void onClick(Context context){
    startActivityForResult(new Intent(context, SecondActivity.class), 1234);
    }
 }).build());
```

> **注意** 成功获取Token后，需把通过setView()方法注入进去的View设置为null。

- 一键登录修改授权页主题（setAuthUIConfig）

```
 /**
* 修改授权页面主题，开发者可以通过此方法修改授权页面主题，需在getLoginToken接口之前调用
*
* @param authUIConfig登录授权页UI自定义配配置
*/
public void setAuthUIConfig(AuthUIConfig authUIConfig)
```

调用示例：

```
setAuthUIConfig(new AuthUIConfig.Builder()
.setLogBtnText("一键登录")
.setLogBtnClickableColor(Color.BLACK)
.setLogBtnUnClickableColor(Color.BLUE)
.setLogBtnTextColor(Color.WHITE).setLogoHidden(false)
.setNavColor(0xff026ED2)
.setNavText("免密登录")
.setNavTextColor(Color.WHITE)
.setNumberColor(Color.WHITE)
.setNumberSize(28)
.setNumberColor(0xff000000).create());
```

SDK回调说明

- 获取Token回调
  - 回调返回的ret都通过TokenRet tokenRet = JSON.parseObject(ret, TokenRet.class) 解析。
  - 授权页唤起成功、获取Token成功都会回调onTokenSuccess方法（可以通过返回码来区分）。
  - 获取Token失败会回调onTokenFailed。

获取Token回调示例代码：

```
mTokenListener = new TokenResultListener() {
 @Override
 public void onTokenSuccess(final String ret) {
 MainActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    /*
                     *  setText just show the result for get token
                     *  use ret to verify number
                     */
                    //ResultCode#CODE_START_AUTHPAGE_SUCCESS是授权页唤起成功码，若不需要处理
，则过滤
                     if (ResultCode.CODE_START_AUTHPAGE_SUCCESS.equals(tokenRet.getCode()
)) {
                           return;
                    }
                    TokenRet tokenRet = JSON.parseObject(ret, TokenRet.class);
                    if (tokenRet != null) {
                        token = tokenRet.getToken();
                    }
                    mAlicomAuthHelper.quitLoginPage();
                }
            });
        }
        @Override
        public void onTokenFailed(final String ret) {
            MainActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    /*
                     *  setText just show the result for get token
                     *  do something when getToken failed, such as use sms verify code.
                     */
                    TokenRet tokenRet = JSON.parseObject(ret, TokenRet.class);
                    mAlicomAuthHelper.quitLoginPage();
                }
            });
        }
    };
```

- 加速唤起授权页或加速本机号码校验回调

```
public interface PreLoginResultListener {
/**
* @param vendor返回预取成功运营商
*/
void onTokenSuccess(String vendor);
/**
* @param vendor返回预取失败运营商
* @param ret返回失败原因
*/
void onTokenFailed(String vendor, String ret);
}
```

预取号回调示例代码：

```
mPreLoginResultListener = new PreLoginResultListener() {
            @Override
            public void onTokenSuccess(final String s) {
                MainActivity.this.runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                       /*
                        *  推荐在登录页初始化的时候调用
                        *  如果没有合适的调用时机
                        *  不调用此接口也没关系
                        *  千万不要App冷启动初始化就调用
                        *  不要调用完预取号后马上调用getLoginToken
                        *  最好判断用户是否登录，若已登录不要使用此接口
                        */
                        mRetTV.setText("预取号成功:" + s);
                    }
                });
            }
            @Override
            public void onTokenFailed(final String s, final String s1) {
                /*
                 *  预取号调用失败
                 *  不用太关注，还是可以直接在用户点击"登录"时，调用getLoginToken
                 */
                mRetTV.setText("预取号失败:" + s + s1);
            }
        });
    }
}
```

- 控件点击事件回调

  授权页控件点击事件通过此回调返回

```
public interface AuthUIControlClickListener{
/**
*
* @param code控件点击事件code
* @param context Android上下文
* @param jsonObj点击事件返回的具体内容，不同控件返回的事件内容有所不同
*/
void onClick(String code, Context context, JSONObject jsonObj);
}
```

  回调示例代码

```
mAlicomAuthHelper.setUIClickListener(new AuthUIControlClickListener( ) {
            @Override
            public void onClick(String code, Context context, JSONObject jsonObj) {
                Log.e("xxxxxx", "OnUIControlClick:code=" + code + ", jsonObj=" + (jsonObj
== null ? "" : jsonObj.toJSONString()));
}
});
```

代码调用顺序

```
/*
* 1.初始化获取Token实例
*/
mTokenListener = new TokenResultListener() {}
/*
* 2.初始化SDK实例
*/
mAlicomAuthHelper = PhoneNumberAuthHelper.getInstance(context, mTokenListener);
/*
* 3.设置SDK密钥
*/
mAlicomAuthHelper.setAuthSDKInfo();
/*
* 4.检测终端网络环境是否支持一键登录或者号码认证，根据回调结果确定是否可以使用一键登录功能
*/
mAlicomAuthHelper.checkEnvAvailable(PhoneNumberAuthHelper#SERVICE_TYPE_LOGIN);
/*
* 5.若步骤4返回true，则根据业务情况，调用预取号或者一键登录接口
*   详见Demo接入工程
*/
mAlicomAuthHelper.getLoginToken(context, 5000);
```

进行取号

- 一键登录获取手机号：当您调用getLoginToken接口成功获取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。
- 本机号码校验结果：当您调用GetVerifyToken接口成功获取Token后，将Token传递至您的服务端，服务端携带Token调用阿里云的VerifyMobile接口，即可进行最终的取号操作。

SDK返回码

返回码	返回码描述	解决方案
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。	升级SDK版本。您可前往号码认证服务控制台下载最新版SDK。

返回码	返回码描述	解决方案
600005	手机终端不安全。	切换到其他登录方式。
600007	未检测到SIM卡。	提示用户检查SIM卡后重试。
600008	移动数据网络未开启。	提示用户开启移动数据网络后重试。
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。	检查数据网络环境后重试，若还未解决问题，您可通过切换飞行模式、重启手机或切换到其他登录方式的操作解决问题。
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，该功能调用次数已达上限。	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。	密钥未设置或者设置错误，请先检查密钥信息，如密钥无问题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。
600023	加载自定义控件异常。	检查自定义控件添加是否正确。
600024	终端环境检查支持认证。	无。
600025	终端检测参数错误。	检查传入参数类型与范围是否正确。
600026	授权页已加载时不允许调用加速或预取号接口。	检查是否有授权页拉起后，调用preLogin或者accelerateAuthPage接口的行为不被允许。

除阿里云SDK返回码外，运营商错误码详情请参见运营商SDK错误码。

授权页点击事件响应码

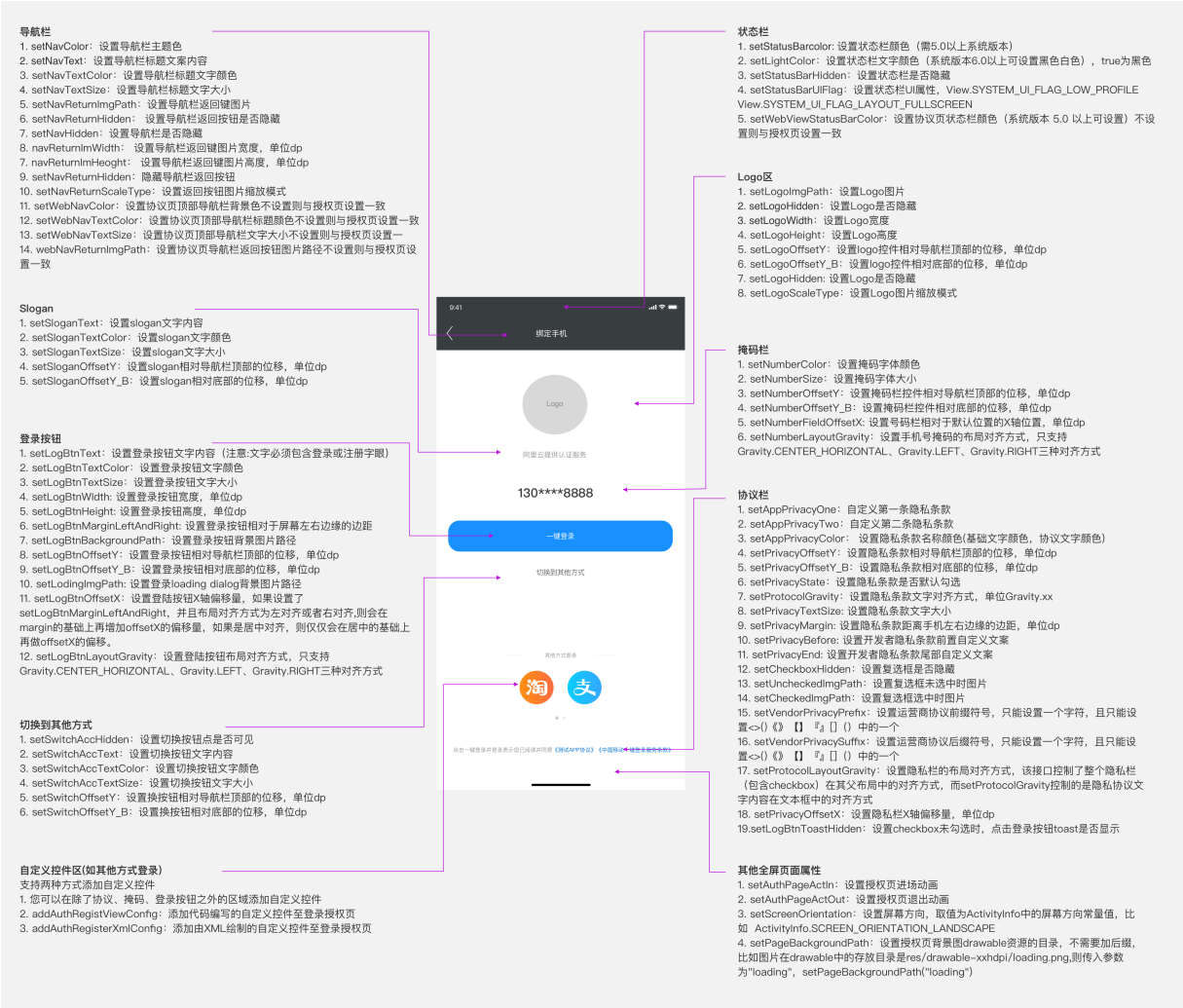
响应码	响应码描述
700000	点击返回，用户取消免密登录。
700001	点击切换按钮，用户取消免密登录。
700002	点击登录按钮事件。

响应码	响应码描述
700003	点击check box事件。
700004	点击协议富文本文字事件。

授权页面设计规范

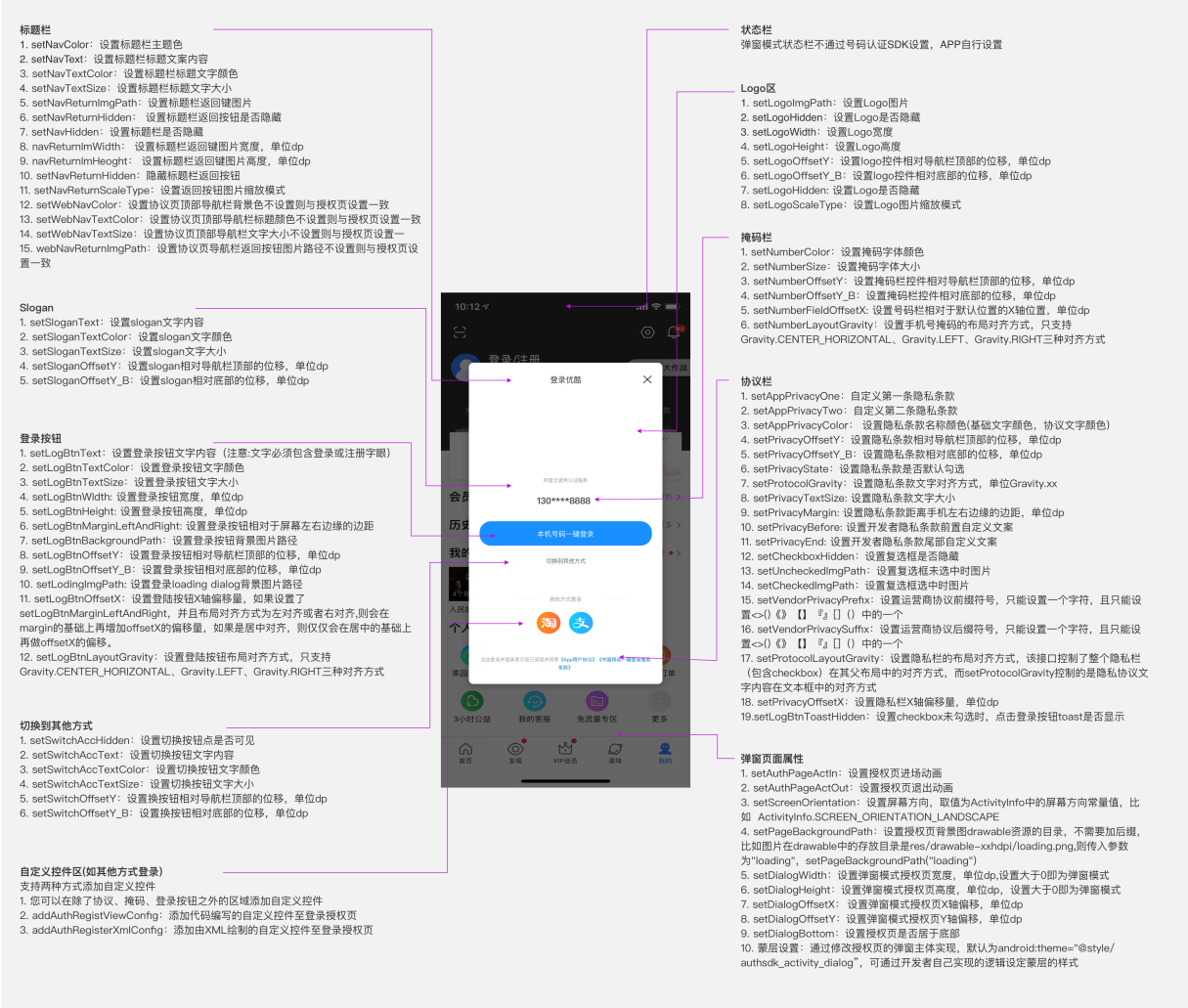
授权页面：

> **注意** 涉及图片路径的参数，仅仅为图片名称（不带路径或后缀名），并且图片需要放置在drawable、drawable-xxhdpi等目录下。



弹窗授权页面：

> **注意** 请勿遮挡协议栏、一键登录按钮、掩码或将字体颜色设置为透明，否则不能获取到一键登录Token。



授权页配置详情请参见授权页配置说明。

授权页配置说明

- 授权页导航栏

方法	参数类型	说明
setStatusBarColor	int	设置状态栏颜色（系统版本5.0以上可设置）。
setLightColor	int	设置状态栏字体颜色（系统版本6.0以上可设置，默认为黑色）。取值： ○ true：表示字体颜色为黑色。 ○ false：表示字体颜色为白色。
setNavColor	int	设置导航栏主题颜色。
setNavText	String	设置导航栏标题文字。
setNavTextColor	int	设置导航栏标题字体颜色。
setNavTextSize	int	设置导航栏标题字体大小。

方法	参数类型	说明
setNavReturnImgPath	String	设置导航栏返回键图片。
setNavReturnHidden	boolean	设置导航栏返回按钮是否隐藏。取值： ○ true：表示隐藏。 ○ false：表示显示。
setNavHidden	boolean	设置默认导航栏是否隐藏。取值： ○ true：表示隐藏。 ○ false：表示显示。
setStatusBarHidden	boolean	设置状态栏是否隐藏。取值： ○ true：表示隐藏。 ○ false：表示显示。
setStatusBarUIFlag	int	设置状态栏UI属性。取值： ○ View.SYSTEM_UI_FLAG_LOW_PROFILE：非全屏显示状态，状态栏的部分图标会被隐藏。 ○ View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN：全屏显示。
setWebViewStatusBarColor	int	设置协议页状态栏颜色（系统版本5.0以上可设置），不设置则与授权页设置一致。
setWebNavColor	int	设置协议页顶部导航栏背景色，不设置则与授权页设置一致。
setWebNavTextColor	int	设置协议页顶部导航栏标题颜色，不设置则与授权页设置一致。
setWebNavTextSize	int	设置协议页顶部导航栏字体大小，不设置则与授权页设置一致。
webNavReturnImgPath	String	设置协议页导航栏返回按钮图片路径，不设置则与授权页设置一致。
setBottomNavColor	int	设置底部虚拟按键背景色（系统版本5.0以上可设置。

- 授权页Logo

方法	参数类型	说明
setLogoHidden	boolean	设置Logo是否隐藏。取值： ○ true：表示隐藏。 ○ false：表示显示。
setLogoImgPath	String	设置Logo图片。

方法	参数类型	说明
setLogoWidth	int	设置Logo控件宽度。
setLogoHeight	int	设置Logo控件高度。
setLogoOffsetY	int	设置Logo控件相对导航栏顶部的位移（单位：dp）。
setLogoOffsetY_B	int	设置Logo控件相对底部的位移（单位：dp）。
setLogoScaleType	ImageView.ScaleType	设置Logo图片缩放模式。

- 授权页Slogan

方法	参数类型	说明
setSloganText	String	设置Slogan文字内容。
setSloganTextColor	int	设置Slogan字体颜色。
setSloganTextSize	int	设置Slogan字体大小。
setSloganOffsetY	int	设置Slogan相对导航栏顶部的位移（单位：dp）。
setSloganOffsetY_B	int	设置Slogan相对底部的位移（单位：dp）。

- 授权页掩码栏

方法	参数类型	说明
setNumberColor	int	设置手机号码字体颜色。
setNumberSize	int	设置手机号码字体大小。
setNumFieldOffsetY	int	设置号码栏控件相对导航栏顶部的位移（单位：dp）。
setNumFieldOffsetY_B	int	设置号码栏控件相对底部的位移（单位：dp）。
setNumberFieldOffsetX	int	设置号码栏相对于默认位置的X 轴偏移量（单位：dp）。

方法	参数类型	说明
setNumberLayoutGravity	int	设置手机号掩码的布局对齐方式，仅支持三种方式，取值： ○ Gravity.CENTER_HORIZONTAL：水平居中 ○ Gravity.LEFT：左对齐 ○ Gravity.RIGHT：右对齐

- 授权页登录按钮

方法	参数类型	说明
setLogBtnText	String	设置登录按钮文字（必须包含登录或注册关键字）。
setLogBtnTextColor	int	设置登录按钮字体颜色。
setLogBtnTextSize	int	设置登录按钮字体大小。
setLogBtnWidth	int	设置登录按钮宽度（单位：dp）。
setLogBtnHeight	int	设置登录按钮高度（单位：dp）。
setLogBtnMarginLeftAndRight	int	设置登录按钮相对于屏幕左右边缘边距。
setLogBtnBackgroundPath	String	设置登录按钮背景图片路径。
setLogBtnOffsetY	int	设置登录按钮相对导航栏顶部的位移（单位：dp）。
setLogBtnOffsetY_B	int	设置登录按钮相对底部的位移（单位：dp）。
setLoadingImgPath	String	设置登录loading dialog背景图片路径。
setLogBtnOffsetX	int	设置登录按钮X轴偏移量。如果设置了setLogBtnMarginLeftAndRight，对齐方式为左对齐或者右对齐，则会在margin的基础上再增加offsetX的偏移量，如果是居中对齐，则仅会在居中对齐的基础上再做offsetX的偏移。

方法	参数类型	说明
setLogBtnLayoutGravity	int	设置登录按钮布局对齐方式，仅支持三种方式，取值： ○ Gravity.CENTER_HORIZONTAL：水平居中 ○ Gravity.LEFT：左对齐 ○ Gravity.RIGHT：右对齐

- 授权页隐私栏

方法	参数类型	说明
setAppPrivacyOne	String, String	设置开发者隐私条款1名称和URL（名称，URL）。
setAppPrivacyTwo	String, String	设置开发者隐私条款2名称和URL（名称，URL）。
setAppPrivacyColor	int, int	设置隐私条款名称颜色（基础文字颜色，协议文字颜色）。
setPrivacyOffsetY	int	设置隐私条款相对导航栏顶部的位移（单位：dp）。
setPrivacyOffsetY_B	int	设置隐私条款相对底部的位移（单位：dp）。
setPrivacyState	boolean	设置隐私条款是否默认勾选。
setProtocolGravity	int	设置隐私条款文字对齐方式。
setPrivacyTextSize	int	设置隐私条款字体大小。
setPrivacyMargin	int	设置隐私条款距离手机左右边缘的边距，（单位：dp）。
setPrivacyBefore	String	设置开发者隐私条款前置自定义文案。
setPrivacyEnd	String	设置开发者隐私条款尾部自定义文案。
setCheckboxHidden	boolean	设置复选框是否隐藏。
setUncheckedImgPath	String	设置复选框未选中时显示的图片。
setCheckedImgPath	String	设置复选框选中时显示的图片。
setVendorPrivacyPrefix	String	设置运营商协议前缀符号，只能设置一个字符，且只能设置<>、()、《》、【】、『』、[]、（）中的一个。

方法	参数类型	说明
setVendorPrivacySuffix	String	设置运营商协议后缀符号，只能设置一个字符，且只能设置<>、()、《》、【】、『』、[]、（）中的一个。
setProtocolLayoutGravity	int	设置隐私栏的布局对齐方式，仅支持三种方式，取值： ○ **Gravity.CENTER_HORIZONTAL**：水平居中 ○ **Gravity.LEFT**：左对齐 ○ **Gravity.RIGHT**：右对齐 该接口控制了整个隐私栏（包含check box）在其父布局中的对齐方式，而setProtocolGravity控制的是隐私协议文字内容在文本框中的对齐方式。
setPrivacyOffsetX	int	设置隐私栏X轴偏移量（单位：dp）。
setLogBtnToastHidden	boolean	设置切换check box未勾选时，单击登录按钮toast是否显示。

请勿遮掩授权页面的隐私栏，否则会导致号码认证失败。

- 切换控件方式

方法	参数类型	说明
setSwitchAccHidden	boolean	设置切换按钮点是否可见。取值： ○ true：表示可见。 ○ false：表示不可见。
setSwitchAccText	String	设置切换按钮文字内容。
setSwitchAccTextColor	int	设置切换按钮字体颜色。
setSwitchAccTextSize	int	设置切换按钮字体大小。
setSwitchOffsetY	int	设置换按钮相对导航栏顶部的位移（单位：dp）。
setSwitchOffsetY_B	int	设置换按钮相对底部的位移（单位：dp）。

- 页面相关函数

方法	参数类型	说明
setAuthPageActIn	String	设置授权页进场动画。

方法	参数类型	说明
setAuthPageActOut	String	设置授权页退出动画。
setScreenOrientation	int	设置屏幕方向。
setPageBackgroundPath	String	设置授权页背景图。
setDialogWidth	int	设置弹窗模式授权页宽度（单位：dp），设置宽度大于0，即为弹窗模式。
setDialogHeight	int	设置弹窗模式授权页宽度（单位：dp），设置大于0，即为弹窗模式。
setDialogOffsetX	int	设置弹窗模式授权页X轴偏移（单位：dp）。
setDialogOffsetY	int	设置弹窗模式授权页Y轴偏移（单位：dp）。
setDialogBottom	boolean	设置授权页是否居于底部。

2.6. 短信认证

2.6.1. 接入概述

本文为您介绍了短信验证码功能的交互流程。开发者需要在App中集成短信验证码的客户端SDK，并在服务端完成API对接。

使用说明

短信验证码SDK可以与一键登录或本机号码校验SDK同时集成，提高用户登录或注册App时的认证覆盖率，快速实现用户认证。

短信验证码的交互流程

短信验证码的系统交互流程主要分为三个步骤：初始化、发送短信验证码、短信验证码校验。

1. 初始化
   i. 用户访问App。
   ii. 开发者App服务器请求调用GetSmsAuthTokens接口。
2. 发送短信验证码
   i. 获取授权Token后，调用短信验证码SDK中的发送验证码接口发送短信验证码。其中Android客户端的SDK接入请参考Android客户端接入，调用sendVerifyCode接口进行验证码发送。iOS客户端的SDK接入请参考iOS客户端接入，调用sendVerifyCodeWithTimeout接口进行验证码发送。
   ii. 终端用户会收到短信验证码，SDK侧会返回用于短信验证码校验的SmsToken。
3. 短信验证码校验
   i. 终端用户输入短信验证码，单击登录。
   ii. 开发者App侧提交SmsToken、手机号码、验证码至开发者服务器进行校验。
   iii. 开发者服务器请求调用VerifySmsCode接口。

2.6.2. Android客户端接入

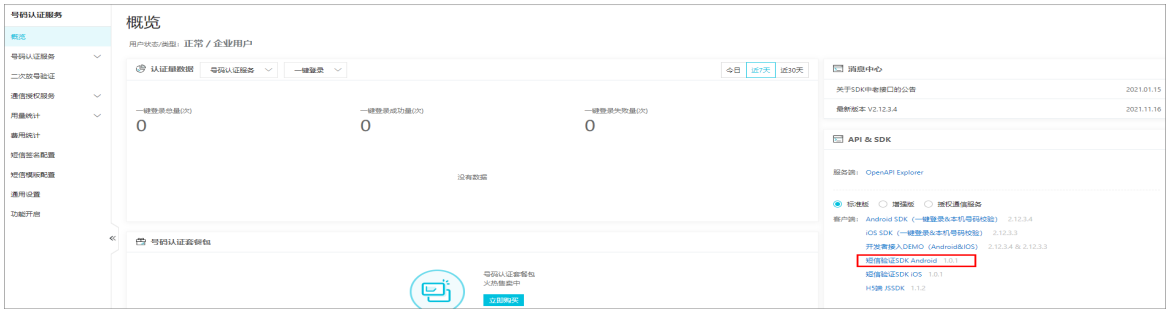
本文为您介绍了Android客户端如何接入短信验证码功能。

前提条件

- 确保您的终端设备已经开启了移动数据网络。
- 确保您已开通了号码认证服务，并成功创建了对应的短信验证码方案且短信签名、模板均已与该方案绑定，详情请参见短信认证使用流程。
- 应用须在iOS 9.0版本及以上平台上运行。

搭建开发环境

1. 下载并解压Android SDK。登录号码认证产品控制台，在**标准版**选项卡，下载并解压Android SDK（含Demo工程）。



2. 添加依赖。将已解压的SDK包中后缀为aar的文件复制至工程的libs目录下。
3. 添加权限支持。在App AndroidManifest.xml文件中添加必要的权限支持：

```
<uses-permission android:name="android.permission.INTERNET" />   <!-- 网络访问 -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" /> <!-- 检查网络状态 -->
```

SDK方法说明

- 获取实例（SmsAuthHelper）

```
SmsAuthHelper smsAuthHelper = new SmsAuthHelper(context, sceneCode);
```

- 设置Token更新处理器（setTokenUpdater）

```
/**
 * 设置Token更新器
 *
 * @param tokenUpdater
 */
public void setTokenUpdater(TokenUpdater tokenUpdater);
```

- 发送短信验证码（sendVerifyCode）

```
/**
 * 发送短信验证码
 * @param countryCode国际电话区号，目前仅作为保留字段，当前版本只支持86（即中国大陆的电话号码）
 * @param phoneNumber手机号
 * @param callback
 * @param timeoutMills接口超时时间（单位：ms）
 */
public void sendVerifyCode(int countryCode,
                           String phoneNumber,
                           SmsCallback callback, long timeoutMills)
```

- 销毁实例（destroy）

```
/**
 * 销毁实例
 */
public void destroy();
```

SDK回调说明

- TokenUpdater。SDK发送短信验证码时需要用到BizToken、StsToken（包含AccessKey信息）这两个Token。调用方法详情请参见GetSmsAuthTokens。

```
public interface TokenUpdater{
    /**
     * 更新Token1
     * @return
     */
    Tokens updateToken();
}
```

> 注意 SDK在检测到本地缓存中无有效Token（Token不存在或者已过期）时，则调用 `TokenUpdater#updateToken` 方法，开发者需要实现该接口，并且获取到Token后，封装成Tokens返回给SDK。Tokens中具体字段可参见Demo和GetSmsAuthTokens。

- SmsCallback。调用发送验证码接口时，无论成功或失败都会通过该接口给开发者发送通知，开发者可以根据返回的Ret对象中Code字段做出判断。

```
public interface SmsCallback {
    void onResult(Ret ret);
}
```

SDK事件返回码

处理错误码时，建议您直接参见SmsReturnCode.h文件中的常量字符进行比对处理，不建议直接使用数值。

返回码	描述	原因/解决方法
600000	成功。	无。
600001	网络不可用。	无可用的网络，建议您切换网络。
600002	手机号非法。	手机号格式错误或无效的手机号。
600003	Token获取失败。	`StsToken` 或 `BizToken` 获取失败、为空，建议您检查相应字段。
600004	接口状态异常。	检查接口状态是否正确，比如是否调用了destroy之后，又调用了业务接口。
600010	请求超时。	需要根据日志和环境信息具体排查，可联系阿里云客服。
600011	鉴权失败。	`StsToken` 鉴权失败或者其信息与 `BizToken` 不对应（账号关系），可检查账号信息、方案信息、包名、签名等。
600012	BizToken错误。	BizToken错误，建议重新申请。
600013	StsToken错误。	StsToken错误，建议重新申请StsToken。
600014	StsToken过期。	StsToken过期，建议重新申请StsToken。
600015	BizToken过期。	BizToken已过期，建议重新申请。
600016	验证码发送频次超出限制。	同一个号码每分钟、每小时或每日发送频次超过限制。具体限制次数，请登录号码认证产品控制台，单击**通用设置**查看详情。
600050	未知异常	联系阿里云客服进行排查。

2.6.3. iOS客户端接入

本文为您介绍了iOS客户端如何接入短信验证码功能。

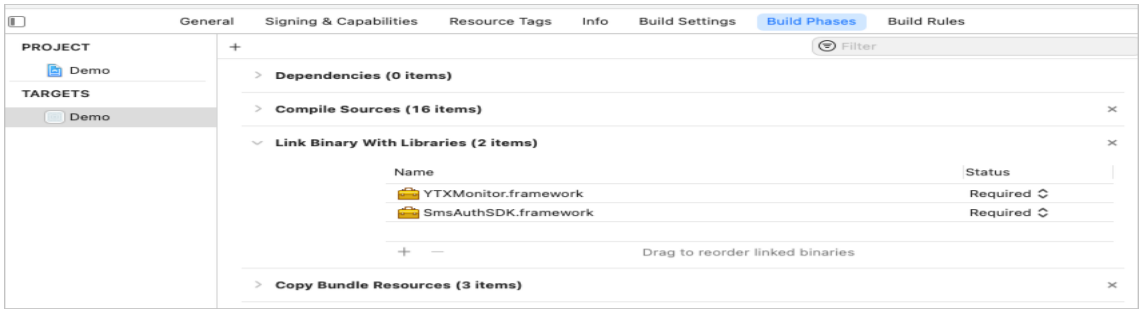
前提条件

- 确保您的终端设备已经开启了移动数据网络。
- 确保您已开通了号码认证服务，并成功创建了对应的短信验证码方案且短信签名、模板均已与该方案绑定，详情请参见短信认证使用流程。
- 应用须在iOS 9.0版本及以上平台上运行。

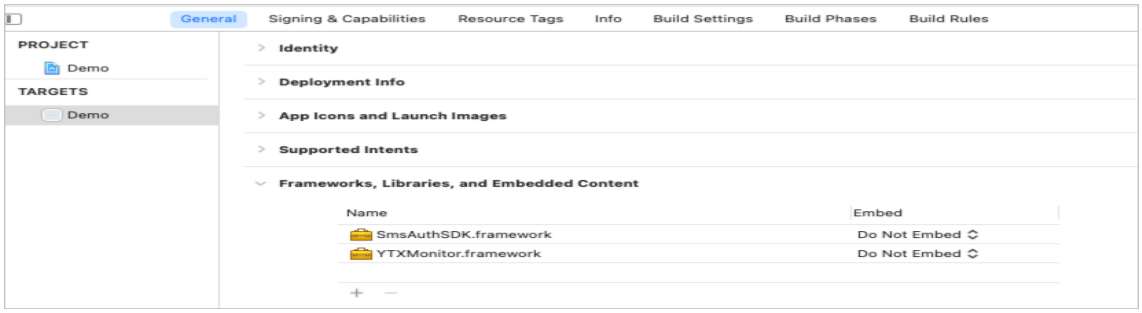
搭建开发环境

1. 下载并解压iOS SDK。登录号码认证产品控制台，在**标准版**选项卡，下载并解压iOS SDK（含Demo工程）。

2. 下载并安装Xcode 11。
3. 添加主库。添加主库的两种方法如下：
   - 选择并单击Xcode 11中**Targets > Build Phases > Link Binary With Libraries**，添加主库**YTXMonitor.framework**、**SmsAuthSDK.framework**，并将其**Status**设置为**Required**。



   - 选择并单击Xcode 11中**Targets > General > Frameworks, Libraries, and Embedded Content**，**SmsAuthSDK.framework**、**YTXMonitor.framework**两个依赖库都属于静态库，将**Embed**设置为**Do Not Embed**。



SDK方法说明

- 主类SmsVerifyCodeManager

- 获取实例

```
/**
 *  初始化SDK实例
 *  @param  sceneCode方案号，必传字段
 *  @return SDK操作实例
 */
- (instancetype)initWithSceneCode:(NSString *)sceneCode;
```

- 获取SDK版本号

```
/**
 *  获取SDK版本号
 *  @return SDK版本号
 */
- (NSString *)getVersion;
```

- 设置SmsTokens更新代理对象（ `id<SmsTokenUpadterDelegate>` ）

```
/**
 *  设置Token更新代理对象
 *  @param  tokenUpdateDelegate Token更新代理对象，注：SDK内部对该对象是弱引用，需要外部来维护其生命
 */
- (void)setTokenUpdateDelegate:(id<SmsTokenUpadterDelegate>)tokenUpdateDelegate;
```

- 发送短信验证码

```
/**
 *  发送短信验证码
 *  @param  timeout接口超时时间（单位：s）
 *  @param  countryCode国际电话区号，目前仅作为保留字段，当前版本只支持86（即中国大陆的电话号码）
 *  @param  phoneNumber手机号
 *  @param  complete接口调用结果回调
 */
- (void)sendVerifyCodeWithTimeout:(NSTimeInterval)timeout
                      countryCode:(int)countryCode
                      phoneNumber:(NSString *)phoneNumber
                         complete:(void(^)(SmsSendCodeResult *result))complete;
```

- 清空本地SmsTokens缓存

```
/**
 *  清空本地SmsTokens缓存
 */
- (void)clearTokenCache;
```

- **实体类SmsTokens**

```
/// stsToken
@property (nonatomic, copy) NSString *stsToken;
/// bizToken
@property (nonatomic, copy) NSString *bizToken;
/// Token过期时间（单位：ms）
@property (nonatomic, assign) long long expiredTimeMills;
/// 临时accessKeyId
@property (nonatomic, copy) NSString *accessKeyId;
/// 临时accessKeySecret
@property (nonatomic, copy)SmsSendCodeResult NSString *accessKeySecret;
```

- 实体类SmsSendCodeResult

```
/// 请求对应的requestId，记录下来方便后面问题的全链路排查
@property (nonatomic, copy) NSString *requestId;
/// 请求返回的code，具体请参见SmsReturnCode.h
@property (nonatomic, copy) NSString *code;
/// 请求返回的msg
@property (nonatomic, copy) NSString *msg;
/// 发送验证码请求返回的Token
@property (nonatomic, copy) NSString *smsVerifyToken;
/// 接口请求失败返回的详细内容
@property (nonatomic, strong) NSDictionary *failedResponseData;
```

- SmsTokens更新代理

SDK发送短信验证码时需要用到BizToken、StsToken（包含AccessKey信息）两个Token。调用方法详情请参见GetSmsAuthTokens。

```
@protocol SmsTokenUpadterDelegate <NSObject>
@required
/**
 *  返回最新的SmsTokens对象
 */
- (SmsTokens *)updateTokenWithVerifyCodeManager:(SmsVerifyCodeManager *)verifyCodeManager
;
@end
```

SDK在检测到本地缓存中没有有效 `SmsTokens` （不存在或者已过期）时，则会调用 `updateTokenWithVerifyCodeManager` 方法，开发者需要实现该代理方法，并且获取到Token后，封装成 `SmsTokens` 返回给SDK。

> 注意 使用该方法时需要发送网络请求获取相关参数，同时也会同步返回结果。在此过程中需要使用信号量做同步且属于私有线程，不会卡顿主线程，请放心同步。具体可以参见Demo工程。

SDK返回码

处理错误码时，建议您直接参见SmsReturnCode.h文件中的常量字符进行比对处理，不建议直接使用数值。

返回码	返回码描述	原因/解决方法
600000	成功。	无。

返回码	返回码描述	原因/解决方法
600001	网络不可用。	无可用的网络连接，建议您切换网络。
600002	手机号非法。	手机号格式错误或无效的手机号。
600003	Token获取失败。	`StsToken`或者`BizToken`获取失败、为空，建议您检查相应字段。
600010	请求超时。	具体需要根据日志和环境信息排查，请联系阿里云客服进行排查。
600011	鉴权失败。	`StsToken`鉴权失败或者其信息与`BizToken`不对应（账号关系），可检查账号信息、方案信息、包名、签名等。
600012	BizToken错误。	BizToken错误，建议重新申请。
600013	StsToken错误。	StsToken错误，建议重新申请StsToken。
600014	StsToken过期。	StsToken过期，建议重新申请StsToken。
600015	BizToken过期。	BizToken已过期，建议重新申请。
600016	验证码发送频次超出限制。	同一个号码每分钟、每小时或每日发送频次超过限制。具体限制次数，请登录号码认证产品控制台，单击**通用设置**查看详情。
600050	未知异常。	请联系阿里云客服进行排查。

2.7. 通信授权服务

2.7.1. Android客户端接入

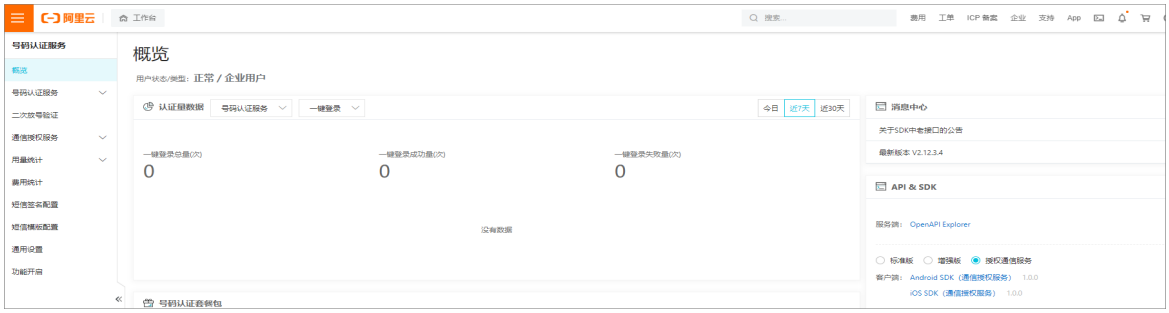
本文为您介绍了Android客户端如何接入通信授权服务功能。

前提条件

确保您已开通通信授权服务，并完成企业信息审核、授权场景审核和添加需要集成SDK的App信息，详情请参见通信授权服务使用流程。

搭建开发环境

1. 登录号码认证产品控制台，在**授权通信服务**选项卡，下载并解压Android SDK。



2. 将后缀为aar的文件复制至工程的libs目录下。
3. 在App工程AndroidManifest.xml增加Activity声明。

```
<activity   android:name="com.nirvana.communicationauth.AuthorizationActivity"
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:exported="false"
    android:launchMode="singleTop"
    android:theme="@style/TransparentTheme"
    android:windowSoftInputMode="adjustPan"/>
<activity   android:name="com.nirvana.communicationauth.ui.AuthWebViewActivity"
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:exported="false"
    android:launchMode="singleTop"
    android:theme="@style/TransparentTheme"/>
```

功能示例

- 获取认证实例

```
/*
  显示在授权页底部的自定义协议
  key是协议名，value是对应协议的URL
  不可为空
 */
HashMap<String, String> pMap = new HashMap<>();
pMap.put("《自定义隐私协议》", "https://www.taobao.com");
pMap.put("《阿里云》", "https://www.aliyun.com/");
CASResultListener pInitListener = new CASResultListener() {
    @Override
    public void onSuccess(@NonNull CASResult cassResult) {
        Log.e(TAG, cassResult.toString());
    }
    @Override
    public void onFailed(@NonNull CASResult cassResult) {
        Log.e(TAG, cassResult.toString());
    }
};
/*
    如果必要参数为空或者格式错误，build返回null
    对应错误信息从Listener抛出
 */
CASHelper helper = new CASBuilder
    //控制台获取的密钥
    .appKey(APP_KEY)
    //需要通信授权的手机号
    .phoneNumber("132xxxxxxxx")
    //控制台配置的方案ID分为合约型与商业型两种
    .schemeId(SCHEME_ID)
    .protocolMap(pMap)
    .context(getApplicationContext())
    .build(pInitListener);
```

- 开放实例接口

```
public interface CASHelper {
    /**
     * 提前获取SDK授权页相关授权信息（对应方案ID）以及当前手机号的授权状态，起到加速授权页拉起的作用
     * 如果不调用此接口在调startAuthPage接口唤起授权页的时候同样会去获取
     * @param timeout接口超时时间（单位：毫秒）
     * @param listener事件回调
     */
    void accelerateAuthInBackground(int timeout, @NonNull CASResultListener listener);
    /**
     * 唤起授权页
     * @param timeout接口超时时间（单位：毫秒）
     * @param uiConfig授权页UI配置项
     * @param listener授权事件回调
     * @param uiClickListener UI点击回调
     */
    void startAuthPage(int timeout, AuthUiConfig uiConfig, @NonNull CASResultListener lis
tener, UiClickListener uiClickListener);
    /**
     * 关闭授权页接口
     * 点击授权页上的关闭按钮会直接关闭授权页
     * 授权成功或失败不会主动关闭授权页 需要主动调用此接口关闭授权页
     * @param traceId流水号对应的是唤起授权页成功返回的traceId
     */
    void quitAuthorizationPage(String traceId);
    /**
     * 开启SDK内部日志打印
     * @param enable是否打印
     */
    void setLoggerEnable(boolean enable);
}
```

- 事件回调CASResultListener

```
public interface CASResultListener {
    /**
     * 成功回调
     * @param cassResult返回成功结果
     */
    void onSuccess(@NonNull CASResult cassResult);
    /**
     * 失败回调
     * @param cassResult返回失败原因
     */
    void onFailed(@NonNull CASResult cassResult);
}
public class CASResult {
    /**
     * 对应事件code ResultCode中定义
     */
    private String code;
    /**
     * 对应事件msg ResultCode中定义
     */
    private String msg;
    /**
     * 流水号，每次调用接口生成
     */
    private String traceId;
    /**
     * 手机号授权状态
     * accelerateAuthInBackground和startAuthPage接口成功时返回
     */
    private String authStatus;
    /**
     * SDK异常或服务端请求错误详细信息
     */
    private Map<String, String> innerFailedResultData;
}
```

- 单击授权页UI回调

```
public interface UiClickListener {
    /**
     * 回抛授权页控件点击事件，目前只有授权按钮和关闭授权页按钮有点击事件回抛
     * @param code UI点击事件code
     *             授权按钮点击code ResultCode.CODE_CLICK_AUTH_BTN
     *             授权页关闭按钮点击code ResultCode.CODE_CLICK_BACK_BTN
     * @param jsonData点击事件返回的具体内容不同，code对应不同内容可能为空
     *                 授权按钮点击{"isChecked":false,"verifyCode":""}isCheck对应协议是否勾选
，verifyCode对应输入的验证码
     */
    void onClick(@NonNull String code, String jsonData);
}
```

- 授权页UI配置AuthUiConfig

```
AuthUiConfig pAuthUiConfig =  new AuthUiConfig.Builder()
                //授权按钮文字大小
                .setAuthBtnTextSize(15)
                //授权按钮文字颜色
                .setAuthBtnTextColor(Color.BLACK)
                //授权按钮文字样式
                .setAuthBtnTextTypeface(Typeface.DEFAULT_BOLD, Typeface.NORMAL)
                //以此类推
                //底部文案文字大小、颜色、样式
                .setSloganXXX()
                //号码栏文字大小、颜色、样式
                .setNumberXXX()
                //顶部导航栏文字大小、颜色、样式
                .setNavXXX()
                //协议栏文字大小、颜色、样式
                .setPrivacyXXX()
                //弹窗蒙层颜色
                .setDialogMaskColor()
                //是否开启弹窗模式
                .setDialogMode(true)
                //默认loading隐藏
                .setLoadingHidden()
                //默认toast隐藏
                .setAuthBtnToastHidden()
                .create();
```

SDK事件返回码

返回码	描述
300000	成功。
300001	授权页唤起成功（Android页面以present方式展示）。
300002	授权页唤起失败。
300003	授权页面内容异常。
300004	参数错误。
300005	网络超时。
300006	用户取消授权。
300007	授权失败，具体原因见innerFailedResultData字段。
300008	AppKey解析失败。
300009	获取授权方案失败。
300010	终端不安全。
300011	发送短信失败。

SDK UI点击返回码

返回码	描述
100000	单击授权按钮回调。
100001	单击授权页退出按钮回调。

2.7.2. iOS客户端接入

本文为您介绍了iOS客户端如何接入通信授权服务功能。

前提条件

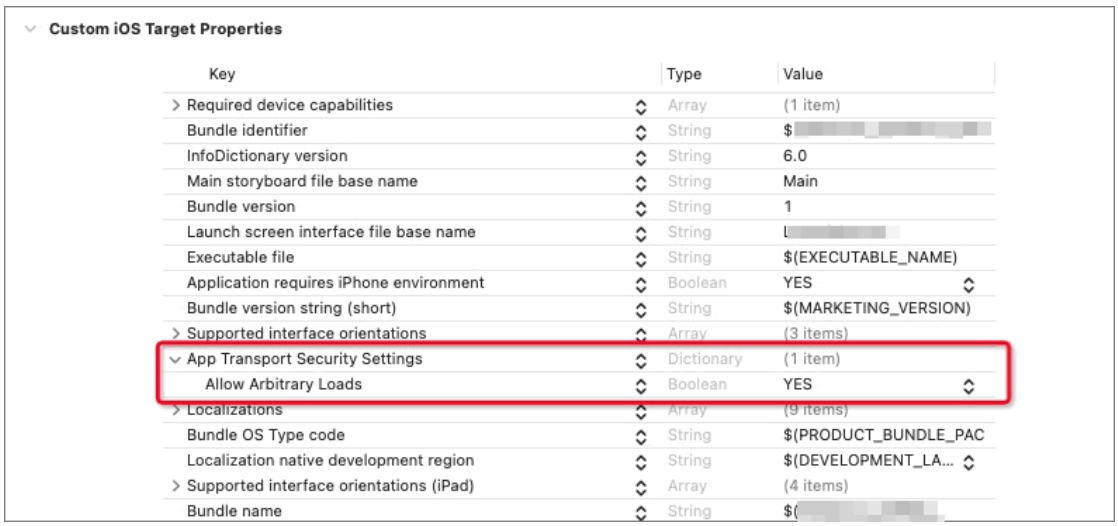
- 确保您已开通通信授权服务，并完成企业信息审核、授权场景审核和添加需要集成SDK的App信息，详情请参见通信授权服务使用流程。
- 设备及系统：
  - 支持iOS 10及以上系统。
  - 支持模拟器和arm架构。
- 开发工具建议使用Xcode 11及以上。

搭建开发环境

1. 登录号码认证产品控制台，在**概览**页面的右侧，选择**授权通信服务**选项卡，下载并解压iOS SDK。
2. 下载并安装Xcode 11。
3. 创建新工程。

   打开Xcode 11，在菜单栏选择**File > New > Project...**，创建新工程。
4. 导入SDK。
   i. 在创建的新工程上单击鼠标右键，选择Add Files To“*您创建的工程名称*”，单击添加CASAuth.framework。
   ii. 在菜单栏选择**TARGETS > General > Frameworks, Libraries, and Embedded Content**，将文件CASAuth.framework的**Embed**值设置为**Embed & Sign**。

iii. 在新建工程的plist文件，选择**Custom iOS Target Properties > App Transport Security Settings**，设置**Type**为**Dictionary**，在添加的子文件下增加Allow Arbitrary Loads且将其值设置为**YES**。



在下载的压缩包中，有三个文件分别是CASAuth.framework、CASAuth.framework.dSYM和ARM文件夹。其中CASAuth.framework包含了arm架构和模拟器架构，ARM文件夹中的CASAuth.framework包含了arm架构。

> **说明** CASAuth.framework是SDK的动态库，用户在最终上架App Store的包需要使用ARM文件夹下的framework。

创建实例

- `CASAuthSDK` 对象即授权通话实例对象，初始化方法为：

```
- (id)initWithConfiguration:(CASConfig *)configuration;
```

- `CASConfig` 为授权通话配置类，属性如下：

```
/// appKey由客户在控制台上生成得到，必填
@property(nonatomic, copy) NSString *appKey;
/// schemeId由客户在控制台生生成取得，必填
@property(nonatomic, copy) NSString *schemeId;
/// 针对合约型用户，传入的授权截止日志。例如：2030-03-30
@property(nonatomic, copy) NSString *expireDate;
/// 授权手机号码，必填
@property(nonatomic, copy) NSString *phoneNumber;
/// 隐私协议，字典形式，可有多个，必填
/// 示例：
///{
//    @"title1":@"url1",
//    @"title2":@"url2"
///}
@property(nonatomic, copy) NSDictionary<NSString *,NSString *>* privacyAgreements;
```

- `CASAuthSDK` 提供如下功能：

```
/// 预加载，主要用来加速获取授权配置信息
- (void)accelerateAuthInBackgroud:(void(^)(BOOL ret, CASResult *result))block;
/// 获取UI配置对象，可以用来进行UI的自定义设置
- (id<CASUIConfigureProtocol>)getUIConfigure;
/// 唤起授权页面，返回授权页面视图控制器给调用方使用
/// @param rootViewController根视图
/// @param timeout超时时间
- (void)startAuthPageWithRootViewController:(id)rootViewController timeout:(float)timeout
completeBlock:(void(^)(BOOL ret, CASResult *result))block;
/// 唤起授权弹框，需要用户传入指定的视图控制器
/// @param rootViewController弹起授权框的视图控制器
/// @param timeout超时时间
- (void)startAlertAuthViewWithRootViewController:(id)rootViewController timeout:(float)ti
meout completeBlock:(void(^)(BOOL ret, CASResult *result))block;
/// 关闭授权页面
- (void)closeAuthView;
```

授权页面弹出的模式有两种，分别是弹框模式（Alert）和页面弹出模式（Present），您可根据需要自行选择模式。 `accelerateAuthInBackgroud` 是加速获取授权信息接口，如果您对速度有要求可以在实例化AuthSDK对象后调用。

- 通过 `- (id<CASUIConfigureProtocol>)getUIConfigure` 接口可获取UI配置对象，提供了如下接口：

```
//title
/// 设置标题字体
/// @param font字体
- (void)setTitleLabelFont:(UIFont *)font;
/// 设置标题文字颜色
/// @param textColor文字颜色
- (void)setTtileLabelTextColor:(UIColor *)textColor;
//middle
/// 设置授权号码显示字体
/// @param font字体
- (void)setPhoneNumberLabelFont:(UIFont *)font;
/// 设置授权号码显示文字颜色
/// @param color颜色
- (void)setPhoneNumberLabelTextColor:(UIColor *)color;
/// 设置验证码发送按钮字体
/// @param font字体
- (void)setVerifyCodeSendButtonFont:(UIFont *)font;
/// 设置验证码发送按钮文字颜色
/// @param color颜色
- (void)setVerifyCodeSendButtonTextColor:(UIColor *)color;
/// 设置验证码发送按钮倒计时文字颜色
/// @param color文字颜色
- (void)setVerifyCodeSendButtonCountdownTextColor:(UIColor *)color;
/// 短信验证码输入框文字颜色
/// @param color颜色
- (void)setVerifyCodeInputViewTextColor:(UIColor *)color;
/// 短信验证码输入框文字字体
/// @param font字体
- (void)setVerifyCodeInputViewFont:(UIFont *)font;
/// 短信验证码输入框PlaceHolder文字颜色
/// @param color颜色
- (void)setVerifyCodeInputViewPlaceHolderTextColor:(UIColor *)color;
```

```
- (void)setVerifyCodeInputViewPlaceHolderTextColor:(UIColor *)color;
/// 短信验证码输入框PlaceHolder文字字体
/// @param font字体
- (void)setVerifyCodeInputViewPlaceHolderTextFont:(UIFont *)font;
//bottom
/// 协议勾选按钮颜色
/// @param color颜色
- (void)setTermsAgreeButtonColor:(UIColor *)color;
/// 一键授权按钮颜色
/// @param color颜色
- (void)setAuthAgreeButtonColor:(UIColor *)color;
/// 一键授权按钮文字字体
/// @param font字体
- (void)setAuthAgreeButtonTextFont:(UIFont *)font;
/// 一键授权按钮文字颜色
/// @param color颜色
- (void)setAuthAgreeButtonTextColor:(UIColor *)color;
/// 协议项前置文字
/// @param beforeText文字
- (void)setPrivacyBeforeText:(NSString *)beforeText;
/// 协议项后置文字
/// @param endText文字
- (void)setPrivacyEndText:(NSString *)endText;
/// 隐私文本颜色
/// @param textColor颜色
- (void)setPrivacyTextColor:(UIColor *)textColor;
/// 隐私文本字体
/// @param font字体
- (void)setPrivacyTextFont:(UIFont *)font;
/// 隐私文本描述文字颜色
/// @param textColor颜色
- (void)setPrivacyDescriptionTextColor:(UIColor *)textColor;
/// 隐私声明文本字体
/// @param font字体
- (void)setPrivacyNamingLabelFont:(UIFont *)font;
/// 隐私声明文本颜色
/// @param color颜色
- (void)setPrivacyNamingLabelTextColor:(UIColor *)color;
/// 是否禁用弹框
/// @param enable YES:能弹框，NO:不能弹框
- (void)toastEnable:(BOOL)enable;
```

- 事件回调。

  授权结果和授权中间状态以及UI点击事件均有回调，如下所示：

```
@required
/// 授权结果回调事件
/// @param ret YES:授权成功，NO:授权失败
/// @param result原因
- (void)onAuthReponseResult:(BOOL)ret reason:(CASResult *)result;
@optional
/// 发送短信验证码回调事件
/// @param ret YES:发送成功，NO:发送失败
/// @param result原因
- (void)onAuthSendVerifyCode:(BOOL)ret reason:(CASResult *)result;
/// 授权页面UI点击事件回调
/// @param code按钮事件code
/// @param msgDict扩展消息
- (void)onUIClickAction:(CASUIClickCode)code message:(NSDictionary *)msgDict;
```

示例代码

- 以Demo工程为例，在 `ViewDidLoad` 中进行初始化。

```
    CASConfig *config = [[CASConfig alloc] init];
    config.appKey = @"yourappKey";
    config.schemeId = @"100000****";
    config.phoneNumber = @"1898984****";
    config.expireDate = @"2021-10-10";
    config.privacyAgreements = @{@"《用户隐私政策》":@"https://example.aliyundoc.com"};
    self.authSDK = [[CASAuthSDK alloc] initWithConfiguration:config];
    self.authSDK.delegate = (id)self;
    [self.authSDK accelerateAuthInBackgroud:^(BOOL ret, CASResult * _Nonnull result) {
    }];
```

- 调用按钮弹出授权页面：

```
 [self.authSDK startAlertAuthViewWithRootViewController:self timeout:5 completeBlock:^(BO
OL ret, CASResult * _Nonnull result) {
        NSLog(@"start auth page:%d result:%@", ret, result);
    }];
```

- 若需要修改UI配置，则需要在**授权页面成功**弹出后，调用如下代码：

```
id<CASUIConfigureProtocol> conf = [self.authSDK getUIConfigure];
  [conf setTitleLabelFont:[UIFont systemFontOfSize:15]];
  [conf setTtileLabelTextColor:[UIColor yellowColor]];
  [conf setPhoneNumberLabelFont:[UIFont systemFontOfSize:10]];
  [conf setPhoneNumberLabelTextColor:[UIColor redColor]];
  [conf setVerifyCodeSendButtonFont:[UIFont systemFontOfSize:13]];
  [conf setVerifyCodeSendButtonTextColor:[UIColor greenColor]];
  [conf setVerifyCodeInputViewTextColor:[UIColor blueColor]];
  [conf setVerifyCodeInputViewFont:[UIFont systemFontOfSize:12]];
  [conf setTermsAgreeButtonColor:[UIColor blackColor]];
  [conf setAuthAgreeButtonColor:[UIColor redColor]];
  [conf setAuthAgreeButtonTextFont:[UIFont systemFontOfSize:12]];
  [conf setAuthAgreeButtonTextColor:[UIColor yellowColor]];
  [conf setPrivacyNamingLabelFont:[UIFont systemFontOfSize:16]];
  [conf setPrivacyNamingLabelTextColor:[UIColor redColor]];
  [conf setPrivacyTextColor:[UIColor redColor]];
  [conf setPrivacyDescriptionTextColor:[UIColor greenColor]];
  [conf setVerifyCodeInputViewPlaceHolderTextColor:[UIColor redColor]];
  [conf setVerifyCodeSendButtonCountdownTextColor:[UIColor greenColor]];
  [conf setPrivacyBeforeText:@"开头文字"];
  [conf setPrivacyEndText:@"结尾文字"];
```

SDK事件返回码

返回码	描述
300000	成功。
300001	授权页面唤起成功（iOS页面以present方式展示）。
300002	授权页唤起失败。
300003	授权页面内容异常。
300004	参数错误。
300005	网络超时。
300006	用户取消授权。
300007	授权失败，具体原因见innerFailedResultData字段。
300008	AppKey解析失败。
300009	获取授权方案失败。
300010	终端不安全。
300011	发送短信失败。

SDK UI点击返回码

返回码	描述
100000	单击授权按钮回调。
100001	单击授权页退出按钮回调。