直播 SDK

常见问题

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

常见问题
平台编译

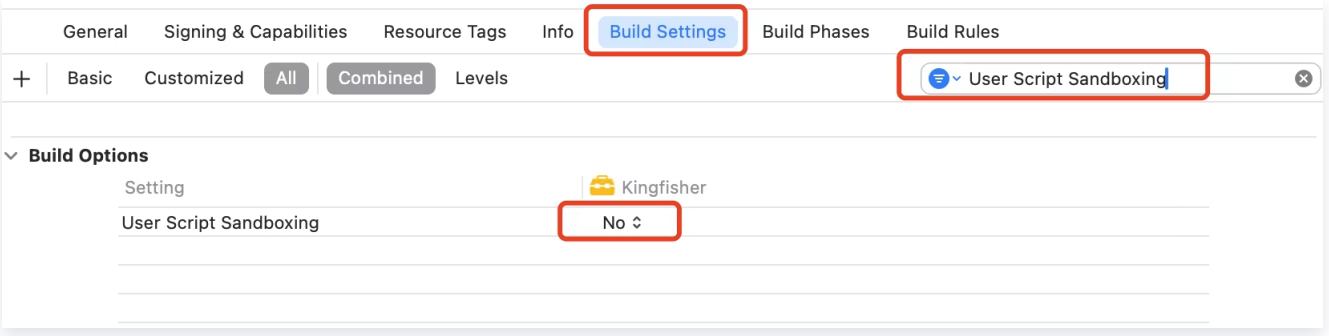
最近更新时间：2025-11-21 14:15:36

iOS

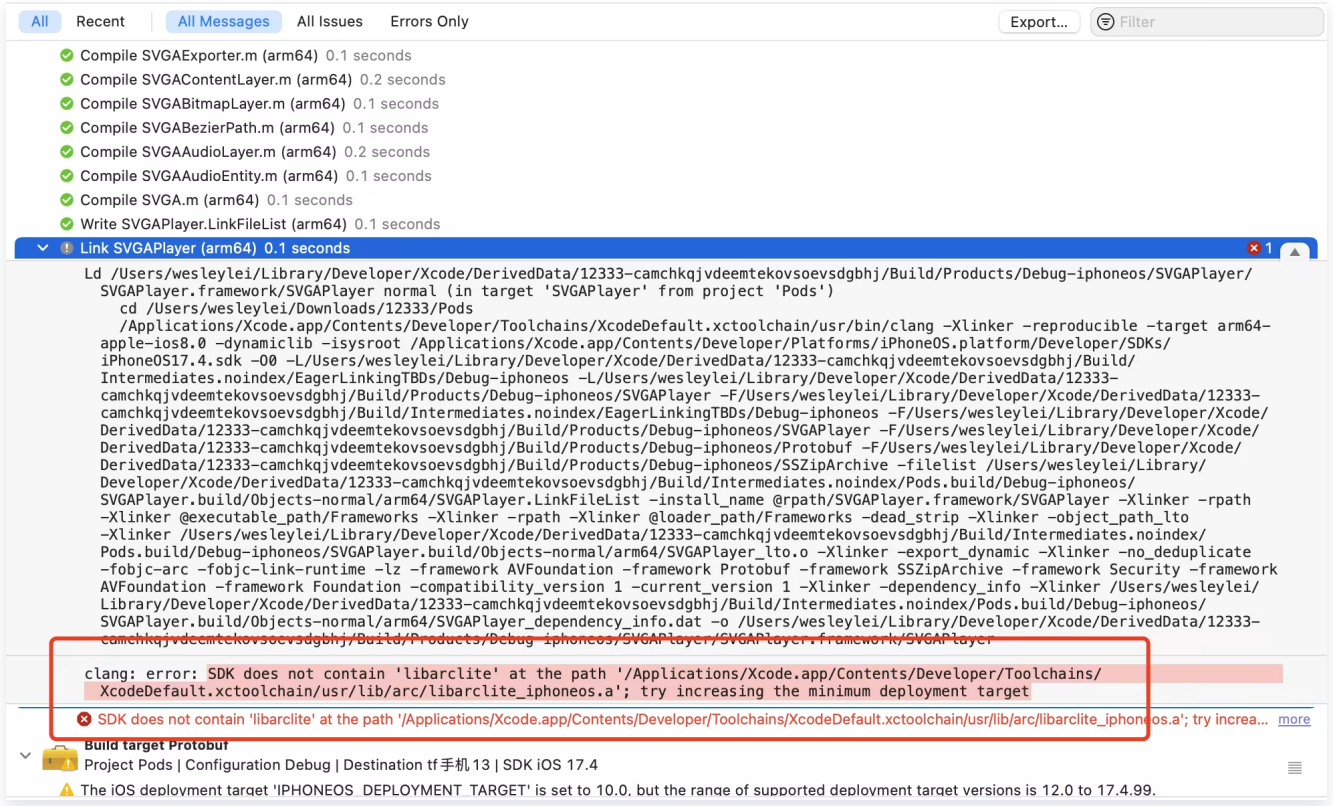
Xcode 15 编译报错?

1. 出现Sandbox: rsync，编译报错截图：

可以在“Build Settings”中把"User Script Sandboxing"设置为NO



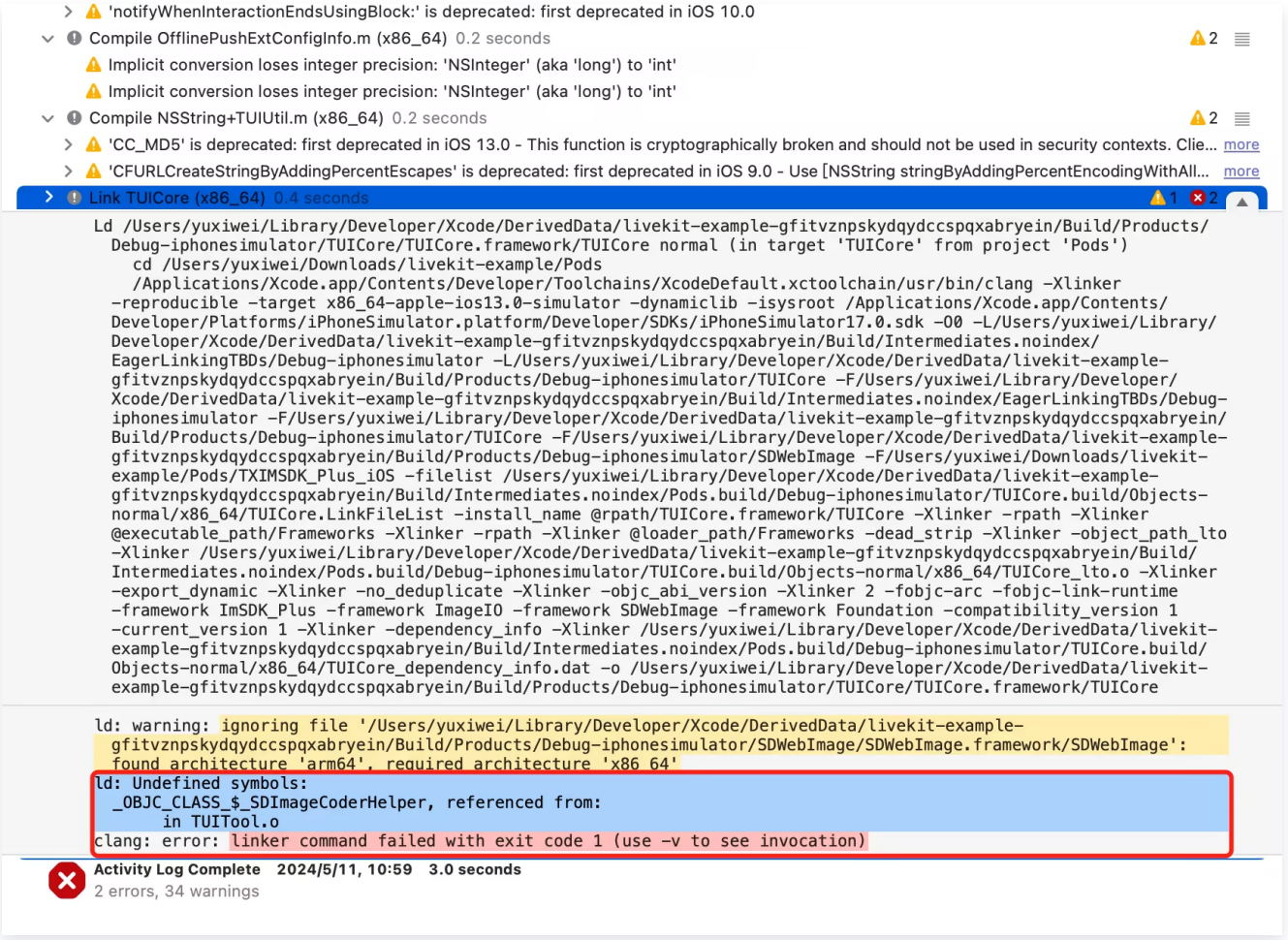
2. 如果出现 SDK does not contain，编译报错截图：



请在 Podfile 添加如下代码：

```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    target.build_configurations.each do |config|
      config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = '13.0'
    end
  end
end
```

3. **如果在 M 系列电脑上运行模拟器，可能会出现 Linker command failed with exit code 1 (use -v to see invocation)，编译报错截图：**



可以在 Podfile 添加如下代码

```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    target.build_configurations.each do |config|
      config.build_settings['EXCLUDED_ARCHS[sdk=iphonesimulator*]'] =
"arm64"
    end
  end
end
```

TUILiveKit 和自己集成的音视频库冲突了?

腾讯云的 音视频库 不能同时集成，可能存在符号冲突，可以按照下面的场景处理。

1. 如果您使用了 `TXLiteAVSDK_TRTC` 库，不会发生符号冲突。可直接在 `Podfile` 文件中添加依赖，

```
pod 'TUILiveKit'
```

2. 如果您使用了 `TXLiteAVSDK_Professional` 库，会产生符号冲突。您可在 `Podfile` 文件中添加依赖，

```
pod 'TUILiveKit/Professional'
```

3. 如果您使用了 `TXLiteAVSDK_Enterprise` 库，会产生符号冲突。建议升级到 `TXLiteAVSDK_Professional` 后使用 `TUILiveKit/Professional` 。

如何查看 TRTC 日志?

TRTC 的日志默认压缩加密，后缀为 .xlog。日志是否加密是可以通过 setLogCompressEnabled 来控制，生成的文件名里面含 C(compressed) 的就是加密压缩的，含 R(raw) 的就是明文的。
iOS： `sandbox的Documents/log` 。

> **说明：**
> - 查看 .xlog 文件需要下载 解密工具，在 Python 2.7环境中放到 xlog 文件同目录下直接使用 `python decode_mars_log_file.py` 运行即可。
> - 查看 .clog 文件（9.6 版本以后新的日志格式）需要下载 解密工具，在 Python 2.7 环境中放到 clog 文件同目录下直接使用 `python decompress_clog.py` 运行即可。
> - 更多日志相关设置参考： 日志输出配置 。

Android

TUILiveKit 是否可以不引入 IM SDK，只使用 TRTC?

不可以，TUIKit 全系组件都使用了腾讯云 IM SDK 作为通信的基础服务，例如创建房间信令、连麦信令等核心逻辑都使用 IM服务，如果您已经购买有其他 IM 产品，也可以参照 TUILiveKit 逻辑进行适配。

allowBackup 异常，如何处理?

```
Manifest merger failed : Attribute application@allowBackup value=(false) from AndroidManifest.xml:7:9-36
    is also present at [com.github.yyued:SVGAPlayer-Android:2.6.1] AndroidManifest.xml:12:9-35 value=(true).
    Suggestion: add 'tools:replace="android:allowBackup"' to <application> element at AndroidManifest.xml:5:5-53:19 to override.
```

- **问题原因**：多个模块的 `AndroidManifest.xml` 中都配置了 `allowBackup` 属性，造成冲突。
- **解决方法**：您可以在您工程的 `AndroidManifest.xml` 文件中删除 allowBackup 属性或将该属性改为 false，表示关闭备份和恢复功能；并在 `AndroidManifest.xml` 文件的 application 节点中添加 `tools:replace="android:allowBackup"` 表示覆盖其他模块的设置，使用您自己的设置。修复示例如图所示：

```
oginActivity.java | AndroidManifest.xml | VideoViewFactory.java | Ar
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        tools:replace="android:allowBackup"
        android:allowBackup="false"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
```

Activity 主题问题：Activity need to use a Theme.AppCompat theme，如何处理?

```
FATAL EXCEPTION: main
Process: com.trtc.uikit.livekit.example, PID: 15190
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.trtc.uikit.livekit.example/com.trtc.uikit.livekit.example.login
.LoginActivity}: java.lang.IllegalStateException: You need to use a Theme.AppCompat theme (or descendant) with this activity.
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3730)
    at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3885)
    at android.app.servertransaction.LaunchActivityItem.execute(LaunchActivityItem.java:101)
    at android.app.servertransaction.TransactionExecutor.executeCallbacks(TransactionExecutor.java:135)
    at android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.java:95)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2332)
    at android.os.Handler.dispatchMessage(Handler.java:107)
    at android.os.Looper.loop(Looper.java:230)
    at android.app.ActivityThread.main(ActivityThread.java:8115) <1 internal line>
    at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:526)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1034)
```

- **问题原因：由于 `LoginActivity` 继承自 `AppCompatActivity` ，所以要给 `LoginActivity` 设置一个 `Theme.AppCompat` 主题。**
- **解决方法：您可以在您工程的 `AndroidManifest.xml` 文件中 `LoginActivity` 的配置里，增加一个 `Theme.AppCompat` 主题。您也可以使用自己的 `Theme.AppCompat` 主题。修复示例如图所示：**

```
<activity
    android:name=".login.LoginActivity"
    android:theme="@style/Theme.AppCompat.DayNight.NoActionBar">
</activity>
```

跳转浏览器打开网页地址失败?

解决办法：您可以在您工程的 `AndroidManifest.xml` 文件中，新增如下配置：

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <queries>
        <intent>
            <action android:name="android.intent.action.VIEW" />
            <data android:scheme="https" />
        </intent>
    </queries>
```

用户鉴权

最近更新时间：2026-06-29 11:00:02

概览

本文档主要介绍腾讯云实时音视频（TRTC）服务的两种鉴权方式，目前，腾讯云实时音视频（TRTC）、即时通信（IM）等服务都采用了 UserSig 的鉴权方式。UserSig 是腾讯云设计的一种安全保护签名，目的是为了阻止恶意攻击者盗用您的云服务使用权。如果您要使用这些基础云服务，就需要在 SDK 初始化或登录函数中提供 SDKAppID，UserID 和 UserSig 三个关键信息，具体含义如下：

- SDKAppID 用于标识您的应用。
- UserID 用于标识您的用户。
- UserSig 则是基于前两者计算出的安全签名，它由 **HMAC SHA256** 加密算法计算得出。只要攻击者不能伪造 UserSig，就无法盗用您的云服务流量。

在调试阶段如何计算 UserSig?

如果您希望快速跑通 Demo，了解 TRTC SDK 相关能力，您可以通过 客户端示例代码 和 控制台 两种方法计算获取 UserSig，具体请参见以下介绍。

> ⊘ **不安全：**
> 注意，如下两种 UserSig 获取计算方案仅适用于调试，如果产品要正式上线，**不推荐**采用这种方案，因为客户端代码（尤其是 Web 端）中的 SECRETKEY 很容易被反编译逆向破解。一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量。

客户端计算 UserSig

1. 获取 SDKAPPID 和密钥：

- 登录**实时音视频控制台** > 应用管理 。
- 单击您需查看的 SDKAppID 对应的**应用信息**，单击进入**应用概览**。
- **点击 SDK 密钥查看**密钥，即可获取用于计算 UserSig 的加密密钥。
- 单击**复制密钥**，可将密钥拷贝到剪贴板中。

2. 计算 UserSig：

为了方便客户端使用，我们提供各平台计算 UserSig 的源码文件，您可直接下载使用：

Android	iOS	Web	微信小程序	Windows（C++）	Flutter	Mac
GitHub	GitHub	GitHub	GitHub	GitHub	GitHub	GitHub

示例代码如下（当然您也可以参考我们各产品的 Demo 工程，详见各产品的开发文档）：

Android

```
// Step1: 导入源码文件
import com.xxx.xxx.GenerateTestUserSig;

// Step2: 填写上一步骤中获取到的 SDKAppID，SDK 密钥
GenerateTestUserSig.SDKAPPID = xxxxxx;
GenerateTestUserSig.SECRETKEY = "xxxxxx";

// Step3: 根据 userID，生成 userSig
String userSig = GenerateTestUserSig.genTestUserSig("userID");
```

iOS

```
// Step1: 导入头文件
#import "GenerateTestUserSig.h"

// Step2: 填写上一步骤中获取到的 SDKAppID，SDK 密钥
[GenerateTestUserSig setSDKAPPID:xxxxxx];
```

```
[GenerateTestUserSig setSECRETKEY:@"xxxxxx"];

// Step3: 根据 userID，生成 userSig
NSString *userSig = [GenerateTestUserSig genTestUserSig:@"userID"];
```

Web

```
// Step1: 导入模块
<script src='js/libs/lib-generate-test-usersig.min.js'></script>
<script src='js/libs/generateTestUserSig.js'></script>

// Step2: 填写上一步骤中获取到的 SDKAppID，SDK 密钥、输入自定义的userID，生成
userSig
const {sdkAppId, userSig } = genTestUserSig({
    sdkAppId: xxxxxx,
    userId: 'xxxxxx',
    sdkSecretKey: 'xxxxxx',
}
```

微信小程序

```
// Step1: 导入模块
import { genTestUserSig } from 'GenerateTestUserSig';

// Step2: 填写上一步骤中获取到的 SDKAppID，SDK 密钥、输入自定义的userID，生成
userSig
const {userSig } = genTestUserSig({
    sdkAppId: xx,
    userId: 'xxxxxx',
    sdkSecretKey: 'xxxxxx'
 })
```

Window（C++）

```
// Step1: 导入头文件
#include "GenerateTestUserSig.h"
```

```
// Step2：填写上一步骤中获取到的 SDKAppID，SDK 密钥
const int SDKAPPID = xxxxxx;
const char* SECRETKEY = "xxxxxx";

// Step3：根据 userID，生成 userSig
const char* userSig = GenerateTestUserSig::genTestUserSig("userID",
SDKAPPID, SECRETKEY);
```

Window（C#）

```
// Step1：导入头文件
using GenerateTestUserSig;

// Step2：填写上一步骤中获取到的 SDKAppID 和 SDK 密钥
GenerateTestUserSig.SDKAPPID = xxxxxx;
GenerateTestUserSig.SECRETKEY = "xxxxxx";

// Step3：根据 userID，生成 userSig
string userSig =
GenerateTestUserSig.GetInstance().GenTestUserSig("userID");
```

Flutter

```
// Step1：导入源码文件
import 'package:xxx/GenerateTestUserSig.dart';

// Step2：填写上一步骤中获取到的 SDKAppID，SDK 密钥
GenerateTestUserSig.SDKAPPID = xxxxxx;
GenerateTestUserSig.SECRETKEY = "xxxxxx";

// Step3：根据 userID，生成 userSig
String userSig = GenerateTestUserSig.genTestUserSig("userID");
```

Mac

```
// Step1：导入头文件
```

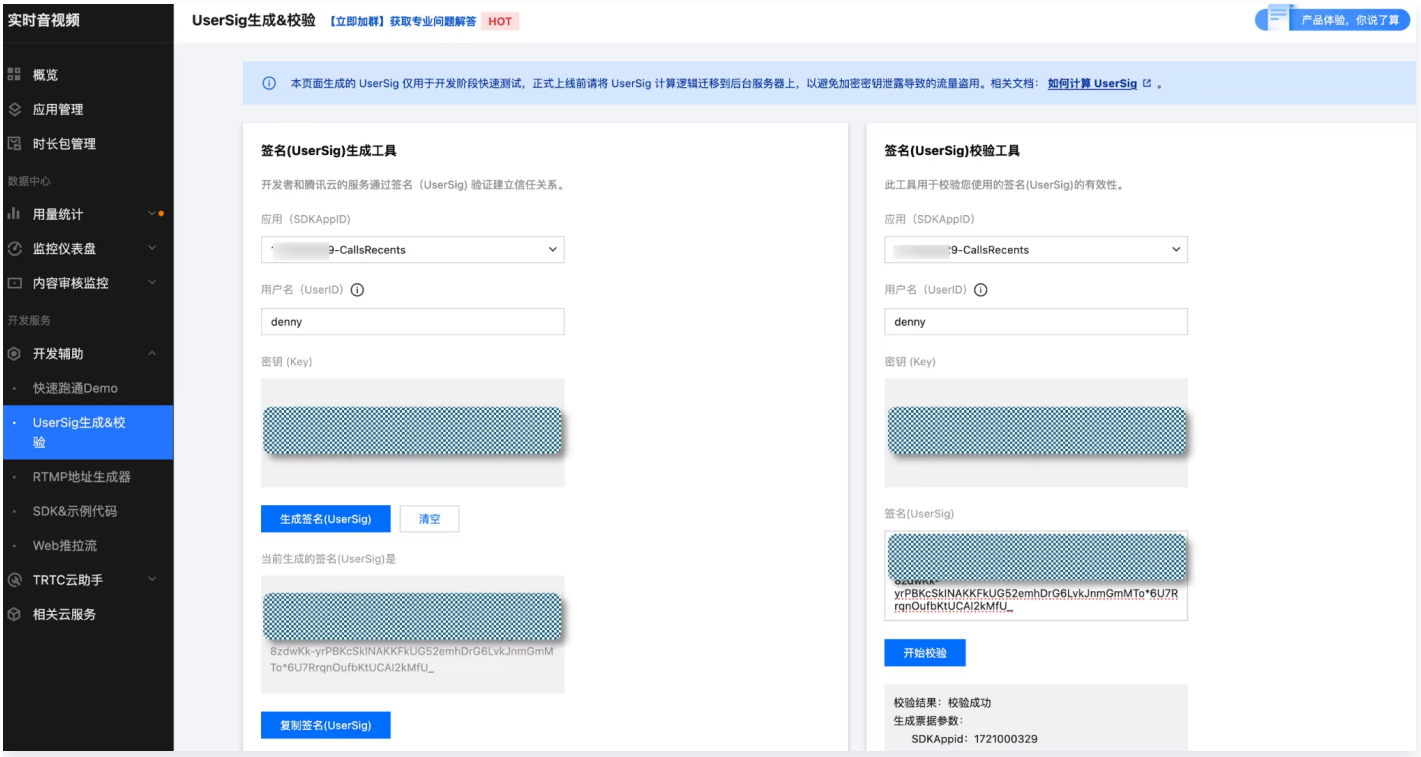
```
#import "GenerateTestUserSig.h"

// Step2: 填写上一步骤中获取到的 SDKAppID，SDK 密钥
[GenerateTestUserSig setSDKAPPID:xxxxxx];
[GenerateTestUserSig setSECRETKEY:@"xxxxxx"];

// Step3: 根据 userID，生成 userSig
NSString *userSig = [GenerateTestUserSig genTestUserSig:@"userID"];
```

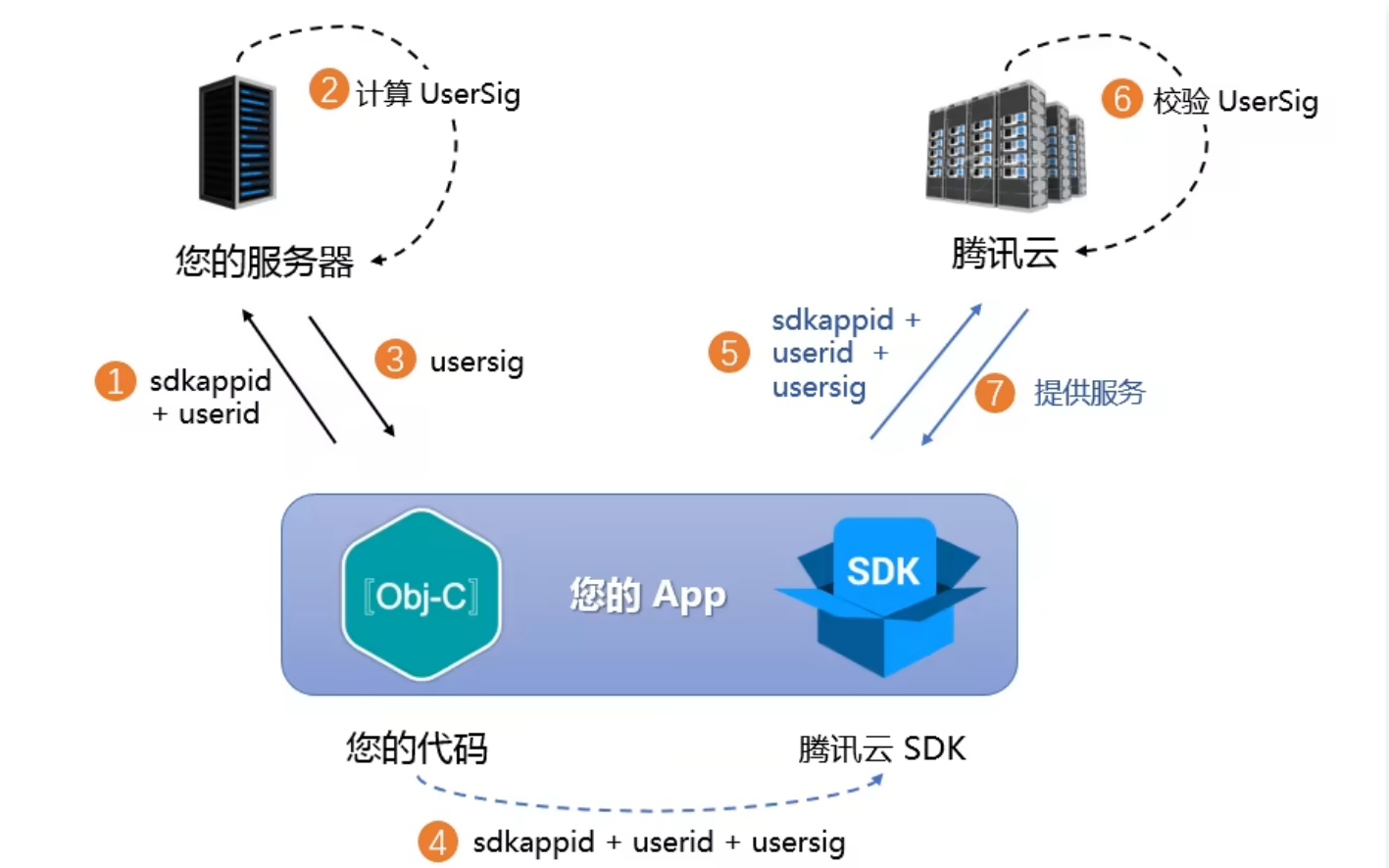
控制台获取 UserSig

- 登录**实时音视频控制台**，进入**开发辅助** > UserSig 生成&校验 。
- 在签名（UserSig）生成工具下，选择对应的 SDKAppID 和 UserID。
- 单击**生成签名(UserSig)**，即可计算得到对应的 UserSig。



在正式运行阶段如何计算 UserSig?

业务正式运行阶段，TRTC 提供安全等级更高的服务端计算 UserSig 的方案，可以最大限度地保障计算 UserSig 用的密钥不被泄露，因为攻破一台服务器的难度要高于逆向一款 App。具体的实现流程如下：



1. 您的 App 在调用 SDK 的初始化函数之前，首先要向您的服务器请求 UserSig。
2. 您的服务器根据 SDKAppID 和 UserID 计算 UserSig，计算源码见文档前半部分。
3. 服务器将计算好的 UserSig 返回给您的 App。
4. 您的 App 将获得的 UserSig 通过特定 API 传递给 SDK。
5. SDK 将 `SDKAppID + UserID + UserSig` 提交给腾讯云服务器进行校验。
6. 腾讯云校验 UserSig，确认合法性。
7. 校验通过后，会向 TRTC SDK 提供实时音视频服务。

为了简化您的实现过程，我们提供了多个语言版本的 UserSig 计算源代码及其示例：

语言版本	签名算法	源代码	使用示例
Java	HMAC-SHA256	genSig	GitHub
GO	HMAC-SHA256	GenSig	GitHub
PHP	HMAC-SHA256	genSig	GitHub
Node.js	HMAC-SHA256	genSig	GitHub
Python	HMAC-SHA256	genSig	GitHub

C#	HMAC-SHA256	GenSig	GitHub

常见问题

SDK 密钥如何切换为非对称密钥?

若查看密钥时只能获取公钥和私钥信息，可点击下图所示位置切换为最新的非对称密钥，请注意：切换密钥前请确保线上业务已停用。