升级 SDK 好处

为了能够长久持续的为用户提供优质服务，新的 SDK 提升了接口调用性能，保障了数据传输安全。为后续提供更多的优质服务带来更多的扩展服务。

老用户升级 SDK 注意事项

老的访问地址：

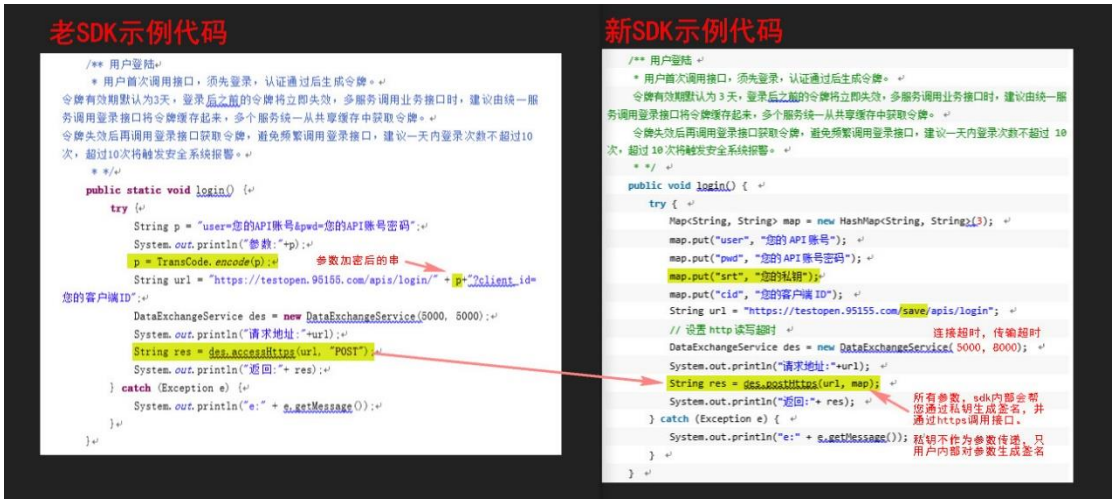
https://域名/apis/接口路径/DES 加密参数?client_id=客户端 ID

新的访问地址：

https://域名/save/apis/接口路径

注：URL 中无需再拼接加密参数串，通过 POST 方式，提交表单参数数据，使用新 SDK 时接口 url 中增加 save 路径。

私钥的作用：通过私钥将传递的参数数据生成签名字符串，作为参数发送到服务端，防止参数被篡改(Java 版本 SDK 已封装，按下图使用，用户无需关心，其他语言签名算法详见下文)。



Java 示例代码（使用 SDK）

```java
package com.sinoiov.test;

import com.openapi.sdk.service.DataExchangeService;
import java.util.HashMap;
import java.util.Map;

/***
 * 使用 SDK 接口客户端调用示例（如不使用 SDK，可通过标准 HTTPS 方式调用接口，忽略正式信任。）
 * 该文件使用 UTF-8 编码格式
 */
public class DemoMain{
    /** 用户登陆
     * 用户首次调用接口，须先登录，认证通过后生成令牌。
     令牌有效期默认为 3 天，登录后之前的令牌将立即失效，多服务调用业务接口时，建议由统一服务调用登录接口将令牌缓存起来，多个服务统一从共享缓存中获取令牌。
     令牌失效后再调用登录接口获取令牌，避免频繁调用登录接口，建议一天内登录次数不超过 10 次，超过 10 次将触发安全系统报警。
     * */
    public void login() {
        try {
            Map<String, String> map = new HashMap<String, String>(3);
            map.put("user", "您的 API 账号");
            map.put("pwd", "您的 API 账号密码");
            map.put("srt", "您的私钥");
            map.put("cid", "您的客户端 ID");
            String url = "https://testopen.95155.com/save/apis/login";
```

```
            // 设置 http 读写超时
            DataExchangeService des = new DataExchangeService(5000, 8000);
            System.out.println("请求地址:"+url);
            String res = des.postHttps(url, map);
            System.out.println("返回:"+ res);
        } catch (Exception e) {
            System.out.println("e:" + e.getMessage());
        }
    }

    /** 一、  车辆最新位置查询（车牌号）接口
     * 本接口提供指定车牌号的车辆最新位置查询。
     *  */
    public static void vLastLocation() {
        try {
            Map<String, String> map = new HashMap<String, String>(4);
            map.put("token", "您的令牌");
            map.put("cid", "您的客户端 ID");
            map.put("srt", "您的私钥");
            map.put("vclN", "陕 YH0009");
            map.put("timeNearby", "30");
            String url = "https://testopen.95155.com/save/apis/vLastLocationV3";
            DataExchangeService des = new DataExchangeService(5000, 8000);
            System.out.println("请求地址:"+url);
            String res = des.postHttps(url, map);
            System.out.println("返回:"+ res);
        } catch (Exception e) {
            System.out.println("e:" + e.getMessage());
        }
    }

    /** 二、  行驶证识别信息查询接口
     * 本接口提供行驶证识别信息查询。
     *  */
    public void vehicleLicense() {
        try {
            Map<String, String> map = new HashMap<String, String>(4);
            map.put("token", "您的令牌");
            map.put("cid", "您的客户端 ID");
            map.put("srt", "您的私钥");
            // 文件路径，包含文件的扩展名
            map.put("path", "E:\\图片\\行驶证\\20190509140734.jpg");
            String url = "https://testopen.95155.com/save/apis/vehicleLicense";
            DataExchangeService des = new DataExchangeService(5000, 8000);
```

```
            System.out.println("请求地址:"+url);
            String res = des.postHttps(url, map);
            System.out.println("返回:"+ res);
        } catch (Exception e) {
            System.out.println("e:" + e.getMessage());
        }
    }

    /** 三、  身份证识别信息查询（正、反面）
     * 本接口提供身份证识别信息查询。
     *  */
    public void idCardLicense() {
        try {
            Map<String, String> map = new HashMap<String, String>(4);
            map.put("token", "您的令牌");
            map.put("cid", "您的客户端 ID");
            map.put("srt", "您的私钥");
            // 文件路径，包含文件的扩展名
            map.put("path", "E:\\图片\\身份证\\正面.jpg");
            map.put("path2", "E:\\图片\\身份证\\背面.jpg");

            String url = "https://testopen.95155.com/save/apis/idCardLicense";
            DataExchangeService des = new DataExchangeService(5000, 8000);
            System.out.println("请求地址:"+url);
            String res = des.postHttps(url, map);
            System.out.println("返回:"+ res);
        } catch (Exception e) {
            System.out.println("e:" + e.getMessage());
        }
    }

    public static void main(String[] args)throws Exception {
        DemoMain demo = new DemoMain();
        demo.login();
//      demo.vLastLocation();
//      demo.vehicleLicense();
//      demo.idCardLicense();
    }

}
```

非 JAVA 语言获取签名算法（无 SDK）

签名原理以及步骤

签名所需参数:
Map<String, String> params ， 调用 API 的所有请求参数，封装成 Map
String srtKey,用户私钥，开放用户在开通用户或者在重置私钥时产生，以邮件方式发送给用户。

1. 调用用户封装请求参数
2. 调用签名方法获取签名
3. 使用签名和 API 入参调用开放平台 API 接口

签名流程（由开放平台提供的 SDK 或工具包提供）

1. 遍历 Map<String, String>，拼装 key+value，放入到 List 集合中，转成 List<String>
2. List<String>集合排序，排序算法集合自带
3. 遍历 List<String>集合，拼装排序后的各个字符串
4. 对拼装后的字符串使用私钥进行 hmac_sha1(通用的 hmac_sha1 算法，各语言都有相应的类库)
5. 对 hmac_sha1 结果进行十六进制转换
6. 对十六进制结果集转为大写字符串，得到签名

C#

签名生成方法（建议封装到您的 util 类中）

```
    /**
     * 生成签名
     * @param paramDic     API请求参数，以Map<String,String>类型
     * @param appSecret    用户私钥
     * @return 签名
     */
    private String sign(Dictionary<String, String> paramDic, String
appSecret)
    {
        byte[] signatureKey = Encoding.UTF8.GetBytes(appSecret);
        //第一步：拼装key+value
        List<String> list = new List<String>();
        foreach (KeyValuePair<String, String> kv in paramDic)
        {
            list.Add(kv.Key + kv.Value);
        }
        //第二步：排序
        list.Sort();
        //第三步：拼装排序后的各个字符串
```

```
        String tmp = "";
        foreach (String kvstr in list)
        {
            tmp = tmp + kvstr;
        }
        //第四步：将拼装后的字符串和app密钥一起计算签名
        //HMAC-SHA1
        HMACSHA1 hmacsha1 = new HMACSHA1(signatureKey);
        hmacsha1.ComputeHash(Encoding.UTF8.GetBytes(tmp));
        byte[] hash = hmacsha1.Hash;
        //TO HEX
        return BitConverter.ToString(hash).Replace("-",
String.Empty).ToUpper();
    }
```

示例（伪代码）

登录请求

```
    /**
     * 现API登录调用方式
     */
    public void loginDemo() {
        try {
            // 请求表单数据
            Map<String, String> map = new HashMap<String, String>(4);
            // 账号
            map.put("user", "此处输入名称");
            // 密码
            map.put("pwd", "此处输入密码");
            // 客户端id
            map.put("cid", "此处输入clientId");

            // url地址 //登录接口为login
            String url =
"https://zhiyunopenapi.95155.com/save/apis/login";
            // 用户私钥
            String appSecret = "此处输入私钥";
            // 调用工具类获取签名
            String sign = XXXUtils.sign(map, appSecret);
            map.put("sign", sign);
            // 以POST表单方式请求开放平台获取token
            String result = XXXXXHttpUtil.post(url, map);
            System.out.println("返回token:"+ result);
        } catch (Exception e) {
```

```
            System.out.println("e:" + e.getMessage());
        }
    }
```

业务 API 请求，以获取车辆信息为例

```
    /**
     * 现API请求业务数据调用方式
     */
    public void queryServiceDataDemo() {
        try {
            // 请求表单
            Map<String, String> map = new HashMap<String, String>(4);

            // 请求表单参数
            // 客户端id
            map.put("cid", "此处输入clientId");
            // token
            map.put("token", "此处输入token");
            // 车牌号
            map.put("vclN", "此处输入车牌号");
            // 车辆颜色 2代表黄色车牌
            map.put("vco", "车牌颜色");

            // url地址,以获取车辆信息为例url
            String url = "https://zhiyunopenapi.95155.com/save/apis/此处输入接口名称";
            // 用户私钥
            String appSecret = "此处输入私钥";
            // 调用工具类获取获取签名
            String sign = XXXUtils.sign(map, appSecret);
            map.put("sign", sign);
            // 以POST表单方式请求开放平台获取车辆信息
            String result = XXXXHttpUtil.post(url, map);
            System.out.println("车辆信息:"+ result);
        } catch (Exception e) {
            System.out.println("e:" + e.getMessage());
        }
    }
```

PHP

签名生成方法（建议封装到您的 util 类中）

示例中，包含三个 function，签名，登录和获取业务数据

```
<html>
 <body>
 <?php
    /**
    *获取签名，建议封装到公共类库
    **/
     function sign($params_arr, $appSecret)
     {
         // map 转 list
         $aliParams = array();
         foreach ($params_arr as $key => $val) {
             $aliParams[] = $key . $val;
         }
         // list 排序
         sort($aliParams);
         //  拼装字符串
         $sign_str = join('', $aliParams);
         //  用过私钥对字符串进行 hmac_sha1，得到签名
         $code_sign = strtoupper(bin2hex(hash_hmac("sha1", $sign_str, $appSecret, true)));

         return $code_sign;
     }

    /**

    * 登录示例

    **/
    function loginDemo()
     {
          //  用户私钥
         $appSecret = '此处输入私钥';
         // map 请求入参
         $code_arr = array(
             'cid' => '此处输入 clientId',
             'user' => '此处输入用户名',
             'pwd' => '此处输入用户密码'
```

```
        );
        // 获取签名
        $sign = sign($code_arr, $appSecret);
        // 把签名放入请求参数中
        $code_arr['sign'] = $sign;
        // 调用
        $url = 'https://zhiyunopenapi.95155.com/save/apis/此处输入接口名称';
        // 使用 https post 表单请求方式调用开放 API
        $result = XXXXXHttpsPost($url, $code_arr);
        echo '开放平台返回 token:' , $result;
        return $result;
    }

    /**
    * 业务请求示例，以获取车辆信息为例
    **/
    function queryServiceDemo()
    {
        // 用户私钥
        $appSecret = '此处输入用户私钥';

        // map 请求入参
        $code_arr = array(
            'cid' => '此处输入用户 clientId',
            'token' => '此处输入 token（通过登录获取的 token）',
            'vclN' => '此处输入车牌号',
            // 车辆颜色 2 代表黄色车牌
            'vco' => '此处输入车牌颜色'
        );
        // 获取签名
        $sign = sign($code_arr, $appSecret);
        // 把签名放入请求参数中
        $code_arr['sign'] = $sign;

        // 调用
        $url = 'https://zhiyunopenapi.95155.com/save/apis/此处输入接口名称;

        // 使用 https post 表单请求方式调用开放 API
        $result = XXXXXHttpsPost($url, $code_arr);
        echo '开放平台返回车辆信息:' , $result;

        return $result;
    }
?>
```

```
</body>
</html>
```

常见问题：

1、 为什么调用接口返回 1002 状态?

参数不正确，需要检查调用接口时传入的参数名称是否有字母写错，区分大小写。

可能您调用接口时没有传入参数，检查下必填项。

可能您只传了通过私钥生成的参数签名，并没有传具体参数。

非 Java 语言最新 SDK 用户，需将参数生成签名，并将生成的签名 `sign` 和业务参数一起 POST 表单方式提交请求。

2、 调用接口提示找不到证书文件怎么办?

Java 版本 SDK 内部已封装了 HTTPS 调用，其他语言使用标准 HTTPS 方式请求时需忽略证书信任方式调用。

已帮您百度搜索到一篇 C#语言的调用文章：

https://www.cnblogs.com/duanh/p/5781839.html
和
https://www.cnblogs.com/lzpong/p/5545464.html

SSL 通信-忽略证书认证错误

.NET 的 SSL 通信过程中，使用的证书可能存在各种问题，某种情况下可以忽略证书的错误继续访问。可以用下面的方式跳过服务器证书验证，完成正常通信。

1.设置回调属性 ServicePointManager.ServerCertificateValidationCallback

注：这个属性设置为要用于客户端的服务器证书的自定义验证方法

True：认证成功； False：认证失败。

C#代码

```
1 ServicePointManager.ServerCertificateValidationCallback =
2           new RemoteCertificateValidationCallback(
```

```
3                       OnRemoteCertificateValidationCallback);
```

VB.NET 代码

```
1 ServicePointManager.ServerCertificateValidationCallback = _
2           New RemoteCertificateValidationCallback( _
3                       AddressOf
OnRemoteCertificateValidationCallback)
```

2.把证书认证函数 OnRemoteCertificateValidationCallback 返回值 True

C#代码

```
1 // 忽略证书认证错误处理的函数
2 private bool OnRemoteCertificateValidationCallback(
3   Object sender,
4   X509Certificate certificate,
5   X509Chain chain,
6   SslPolicyErrors sslPolicyErrors)
7 {
8   return true;  // 认证正常，没有错误
9 }
```

VB.NET 代码

```
' 忽略证书认证错误处理的函数
Private Function OnRemoteCertificateValidationCallback( _
  ByVal sender As Object, _
  ByVal certificate As X509Certificate, _
  ByVal chain As X509Chain, _
  ByVal sslPolicyErrors As SslPolicyErrors _
) As Boolean
  Return True  ' 认证正常，没有错误
End Function
```