**渠道安全对接文档**

**互联网平台项目组**

**2017-11-17**

**版本历史**

| 版本编号 | 版本日期 | 修改人 | 说明 |  |
| --- | --- | --- | --- | --- |
| V1.0 | 2017-11-17 | 阳剑 | 创建基准文档 |  |
| V1.1 | 2017-12-15 | 汪彬彬 | 更新 | 1. 更新错误编码 2. 增加SDK说明 |
| V1.2 | 2018-01-15 | 汪彬彬 | 更新 | 1. 完善调用验签服务流程相关的说明 |

# 名词解释

HTTPS： 安全的http传输协议，即通道加密；

加签验签：请求方对报文进行某种算法的加密，将加密后的签文按照约定追加到传输报文中，主要用于通信唯一性，为了防止报文篡改；服务方收到报文进行验签；

报文加密： 在通道内传输的数据进行加密；

SDK：Software Development Kit ，软件开发工具包，辅助软件开发的工具类集合。

# 总体说明

验签服务器来实现渠道的安全接入：HTTPS协议+ 签名机制。

通过HTTPS协议，保证网络上传输的数据是加密的，不被窃取；

通过签名机制，进行双重防护，即便HTTPS协议被破解，也能防止数据被篡改，同时还能对客户端身份进行识别和授权。

# 接口地址

## 服务访问地址

请求验签服务的地址，HTTPS协议访问时须通过域名访问。

测试环境：

https://111.231.176.126:6666/checksign/v1（内网，IP段需改成对应的HTTPS证书的域名）

## 服务接口清单页面地址

测试环境：

https://111.231.176.126:6666

# 调用验签服务流程

验签系统会对访问的每个服务请求进行身份验证，即每个请求都需要在原有的请求参数中增加一个签名字符串参数（Signature），以验证用户身份。签名串由用户所执有的安全凭证密钥对生成，密钥对包括密钥ID（SecretId）和密钥KEY（SecretKey），若用户还没有该安全凭证，则需要向服务方申请。

## 获取安全凭证

安全凭证密钥对包括 SecretId 和 SecretKey，其中：

* **SecretId：**密钥ID，用于标识 API 调用者身份；
* **SecretKey：**密钥KEY，用于加密签名字符串和服务器端验证签名字符串的密钥。

**注意：**  
API 密钥是业务系统请求的重要凭证，为了您的财产和服务安全，请妥善保存并定期更换密钥，当您更换密钥后，请及时删除旧密钥。

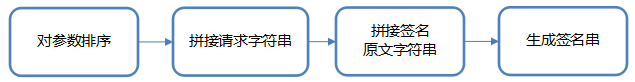
## 准备服务的请求参数

调用服务接口时，需要以HTTPS协议的POST或GET方法发送约定的请求参数，除了以下约定的参数，不同的服务接口也可约定特定的参数，详见具体的服务接口文档，如未特别约定参数，便是使用如下的默认服务参数。这些请求参数和请求协议、请求地址都是生成密钥串的前提。详细的服务请求参数见如下表：

| 参数名称 | 描述 | 是否必须 | 示例 |
| --- | --- | --- | --- |
| Action | 服务名.方法名 | 是 | ProdQueryService.qryProdKindList |
| SecretId | 密钥 ID | 是 | 2342343423 |
| Timestamp | 当前时间戳 | 是 | 1465185768 |
| Nonce | 随机正整数(至少8位，保证全局唯一) | 是 | 11886888 |
| SignatureMethod | 签名方式，确定具体的签名算法。如果不为HmacSHA256，否则为HmacSHA1 | 是 | HmacSHA256或HmacSHA1 |
| ZData | 服务默认的参数名，参数值一般为Json字符串，视具体的服务，有的服务可能不需要服务参数，详见具体的服务接口文档。 | 否 | {"CProdNo":"1111111111"} |
| Signature | 签名串，根据签名规则生成 | 是 | VKtgCxOzi9sQ/iM3L/BYD3JburNCKcCchKu11SKAOC8= |

## 生成签名串

有了安全凭证 SecretId 和 SecretKey 后，就可以生成签名串了。生成签名串的详细过程如下：



假设用户的 SecretId 和 SecretKey 分别是：  
SecretId： AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA  
SecretKey： Gu5t9xGARNpq86cd98joQYCN3Cozk1qA

**注意：**  
这里只是示例，请用户根据自己实际的 SecretId 和 SecretKey 和请求参数进行后续操作。

以洁牙服务为例，当用户调用绑卡接口时，其请求参数为：

| 参数名称 | 描述 | 参数值 |
| --- | --- | --- |
| Action | 服务名.方法名 | HealthCleanTeethService.manualBindCards |
| SecretId | 密钥 ID | AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA |
| Timestamp | 当前时间戳 | 1465185768 |
| Nonce | 随机正整数(至少8位，保证全局唯一) | 11886888 |
| SignatureMethod | 签名方式 | HmacSHA256 |
| ZData | 服务默认参数名，json字符串 | 无 |

### 对参数排序

首先对所有请求参数按参数名做字典序升序排列。（所谓字典序升序排列，直观上就如同在字典中排列单词一样排序，按照字母表或数字表里递增顺序的排列次序，即先考虑第一个“字母”，在相同的情况下考虑第二个“字母”，依此类推。）您可以借助编程语言中的相关排序函数来实现这一功能，如 java 中的 sort 函数。上述示例参数的排序结果如下：

{

"Action" : "HealthCleanTeethService.manualBindCards",

"Nonce" : 11886888,

"SecretId" : "AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA",

"SignatureMethod" : "HmacSHA256",

"Timestamp" : 1465185768

}

### 拼接请求字符串

此步骤将生成请求字符串。  
将把上一步排序好的请求参数格式化成“参数名称”=“参数值”的形式，如对 Action 参数，其参数名称为" Action "，参数值为"HealthCleanTeethService.manualBindCards"，因此格式化后就为 Action= HealthCleanTeethService.manualBindCards。

**注意：**

“参数值” 为原始值而非 url 编码后的值。

若输入参数的 Key 中包含下划线，则需要将其转换为“.”，但是 Value 中的下划线则不用转换。

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为（请忽略文中的换行）：

Action=HealthCleanTeethService.manualBindCards

&Nonce=11886888

&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA

&SignatureMethod=HmacSHA256

&Timestamp=1465185768

### 拼接签名原文字符串

此步骤将生成签名原文字符串。  
签名原文字符串的拼接规则为：

**请求方法 + 请求主机 +请求路径 + ? + 请求字符串**

参数构成说明：

* **请求方法：**支持 POST 和 GET 方式，这里使用 GET 请求， 注意方法为全大写。
* **请求主机：**即主机域名，举例如：api.rhassurance.com
* **请求路径：** /v1/api.do
* **请求字符串：** 即上一步生成的请求字符串。

因此，上述示例的拼接签名原文字符串结果为（请忽略文中的换行）：

GETapi.rhassurance.com/v1/api.do?Action=HealthCleanTeethService.manualBindCards

&Nonce=11886888

&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA

&SignatureMethod=HmacSHA256

&Timestamp=1465185768

**4. 生成签名串**

此步骤生成签名串。

**注意：**  
计算签名的方法有两种：HmacSHA256 和 HmacSHA1 这里要根据您指定的签名算法（即 SignatureMethod 参数）生成签名串。当指定 SignatureMethod 为 HmacSHA256 时，需要使用 HmacSHA256 计算签名，其他情况请使用 HmacSHA1 计算签名。

首先使用签名算法（HmacSHA256 或 HmacSHA1）对上一步中获得的 **签名原文字符串** 进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。

具体代码如下，以 java 语言为例，由于本例中所用的签名算法为 **HmacSHA256**，因此生成签名串的代码如下（使用其它程序设计语言开发时，可用上述示例中的原文字符串进行签名验证，得到的签名串与例子中的一致即可）：

String secretKey="Gu5t9xGARNpq86cd98joQYCN3Cozk1qA";

String srcStr="GETapi.rhassurance.com/v1/api.do?Action=HealthCleanTeethService.manualBindCards" +

"&Nonce=11886888" +

"&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA" +

"&SignatureMethod=HmacSHA256" +

"&Timestamp=1465185768";

String signStr=null;

try {

signStr = sign(srcStr, secretKey, "sha256");

} catch (InvalidKeyException | NoSuchAlgorithmException | UnsupportedEncodingException e) {

e.printStackTrace();

}

System.out.println(signStr);

最终得到的签名串为:

VKtgCxOzi9sQ/iM3L/BYD3JburNCKcCchKu11SKAOC8=

同理，当您指定签名算法为 **HmacSHA1** 时，生成签名串的代码如下：

String secretKey="Gu5t9xGARNpq86cd98joQYCN3Cozk1qA";

String srcStr="GETapi.rhassurance.com/v1/api.do?Action=HealthCleanTeethService.manualBindCards" +

"&Nonce=11886888" +

"&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA" +

"&SignatureMethod=HmacSHA256" +

"&Timestamp=1465185768";

String signStr=null;

try {

signStr = sign(srcStr, secretKey, "sha1");

} catch (InvalidKeyException | NoSuchAlgorithmException | UnsupportedEncodingException e) {

e.printStackTrace();

}

System.out.println(signStr);

最终得到的签名串为：

6L6/Ngc2gRVCepFvr3+ZPEMNrYw=

## 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。  
如上一步生成的签名串为VKtgCxOzi9sQ/iM3L/BYD3JburNCKcCchKu11SKAOC8=，则其编码后为VKtgCxOzi9sQ%2FiM3L%2FBYD3JburNCKcCchKu11SKAOC8%3D。因此，最终得到的签名串请求参数 (Signature) 为：VKtgCxOzi9sQ%2FiM3L%2FBYD3JburNCKcCchKu11SKAOC8%3D，它将用于生成最终的请求URL。

**注意：**  
如果用户的请求方法是 GET，则对所有请求参数的参数值均需要做 URL 编码；此外，部分语言库会自动对 URL 进行编码，重复编码会导致签名校验失败。

## 调用验签服务

生成签名串之后，在原有请求参数基础上，再带上签名串参数，向请求服务的访问地址发送HTTPS请求。由于开发测试环境的证书可能是自定义证书，而非权威授权机构的证书，这里可能存在需要信任服务端SSL证书的问题。比如JAVA开发需要在JVM中导入验签服务端提供的SSL证书，否则可能报错。

## 解析响应报文及错误处理

验签服务在接受请求后，会对附带的签名串进行验证，如果验签成功，将调用具体的服务，对于具体服务返回的报文格式，由于验签系统无法做约束，不管服务是调用成功或失败，将原样返回服务的报文字符串。因此特别注意，服务的返回报文中也可能是其服务调用出错的报文，需要开发者根据服务相关的文档进行错误处理。但如果是验签失败，验签系统将返回验签约定格式的错误报文。不管是服务调用出错还是验签出错，首先应该进行区别判断，然后进行错误处理。验签出错时的报文为JSON字符串，包含的参数说明如下：

| 参数名称 | 描述 | 是否必须 | 示例 |
| --- | --- | --- | --- |
| signCode | 返回结果的签名状态编码，具体参考下文的错误编码表 | 是 | 1001 |
| signMsg | 返回结果的消息 | 是 | 身份认证失败 |

返回报文中包含以上两个参数，即说明是验签失败。

验签错误报文格式示例如下：

{

"signCode":"1001",

"signMsg":"身份认证失败"

}

# 错误编码

当用SDK请求验签服务器时，可能出现如下表的错误：

| 错误代码 | 错误类型 | 错误描述 |
| --- | --- | --- |
| 1001 | 身份认证失败 | 身份验证失败，请确保您请求参数中的 Signature 按照上述步骤计算正确，特别注意 Signature 要做 url 编码后再发起请求。 |
| 1002 | 未被开发商授权访问本接口 | 未被授权调用此接口。 |
| 1003 | 未被开发商授权访问本接口中所操作的资源 | 请问的资源参数中，存在未被开发商授权授权访问的资源，请在 message 字段中查看无权查看的资源 ID。 |
| 1004 | SecretId 不存在 | 签名所用的 SecretId 不存在，也可能是密钥状态有误，请确保 API 密钥有效且未被禁用。 |
| 1005 | 鉴权失败 | 权限校验失败，请确保您有使用所访问资源的权限。 |
| 1006 | 重放攻击错误 | 请注意 Nonce 参数两次请求不能重复。 |
| 1007 | 重复发送错误 | Timestamp 与服务器相差不能超过 5分钟 |
| 2001 | 接口内部错误 | 验签通过，但调用接口报错 |
| 4001 | 请求URL错误 | URL参数格式错误 |
| 4002 | 服务未配置 | 服务接口参数错误 |
| 4003 | 密钥失效或过期 | 服务方提供的密钥已过期或者被禁用 |
| 4004 | 缺少Action参数 | 服务接口参数缺失 |
| 4005 | 缺少RequestMethod参数 | 缺少RequestMethod参数 |
| 4006 | 加签出错 | SDK客户端加签出错 |
| 5001 | 服务器请求超时 | 服务器请求超时 |

# SDK

为了方便渠道供应商更好地对接，提供不同开发语言的SDK，实现对上述签名规则的封装处理，提供简便的API进行开发。目前只提供基于java编程语言的SDK，其它语言可参考编写实现。

## Java SDK说明

向服务方获取的Java SDK为iescp-web-sign-sdk-<版本号>.zip，其中包括源码工程iescp-web-sign-sdk和已编译好的jar包。

如果您的项目非maven项目，可直接导入jar包。

如您的项目为maven项目，可在导入iescp-web-sign-sdk源码工程后，在您的项目中配置对应的maven依赖为：

<dependency>

<groupId>com.iss.iescp</groupId>

<artifactId>iescp-web-sign-sdk</artifactId>

<version>1.0.1</version>

</dependency>

注意，版本号按照您获取的SDK版本号修改。

### 关键API

PcisServiceClient类为所有服务的通用客户端类。为所有基础服务访问提供简便的请求方式，其方法callService会自动对请求数据进行加签并远程调用服务，并返回结果数据。示例如下：

具体可参考源码工程示例类SignClientSample.java

//从服务方得到的API密钥信息

String secretId="secret\_id\_111111";

String secretKey="secret\_key\_111111";

//服务方提供的验签服务访问地址

//String endpoint = "http://localhost:8080";

String endpoint = "https://api.rhassurance.com:6666";

String result=null;

try{

PcisServiceClient client = new PcisServiceClient(endpoint,secretId, secretKey);

//洁牙服务回调预约接口

//result=client.callService("HealthCleanTeethService.manualBindCards","");

//产品大类查询接口

result=client.callService("ProdQueryService.qryProdKindList","");

System.out.println(result);

}catch(SignServerException e1){//加签服务器抛出的异常

System.out.println("Sign Server Exception： " + e1.toString());

}catch(SignClientException e2){//加签SDK客户端抛出的异常

System.out.println("Sign Client Exception： " + e2.toString());

}catch (Exception e) {

System.out.println("error..." + e.toString());

}