# 阿里云
# 视觉智能开放平台

## 签名机制

文档版本：20200513

# 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. 阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，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 |
| --- | --- | --- |
|  | 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。 |  **禁止：**<br>重置操作将丢失用户配置数据。 |
|  | 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：**<br>重启操作将导致业务中断，恢复业务时间约十分钟。 |
|  | 用于警示信息、补充说明等，是用户必须了解的内容。 |  **注意：**<br>权重设置为0，该服务器不会再接受新请求。 |
|  | 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：**<br>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。 | 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<br>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 请求签名

对于每一次HTTP或者HTTPS协议请求，我们会根据访问中的签名信息验证访问请求者身份。具体由使用AccessKeyID和AccessKeySecret对称加密验证实现。其中AccessKeyID是访问者身份，AccessKeySecret是加密签名字符串和服务器端验证签名字符串的密钥，必须严格保密谨防泄露。

**签名流程**

1. 指定请求参数。

   在代码中指定请求参数，参数中需要包含公共请求头和接口必备的参数信息。

   > **说明:**
   > 请求参数中不允许出现以Signature为key的参数。

   示例代码如下：

```
String accessKeyId = "yourAccessId";
String accessSecret = "yourAccessSecret";
java.text.SimpleDateFormat df = new java.text.SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
df.setTimeZone(new java.util.SimpleTimeZone(0, "GMT"));// 这里一定要设置GMT时区
java.util.Map<String, String> paras = new java.util.HashMap<String, String>();
```

   指定参数：

```
paras.put("SignatureMethod", "HMAC-SHA1");
paras.put("SignatureNonce", java.util.UUID.randomUUID().toString());
paras.put("AccessKeyId", accessKeyId);
paras.put("SignatureVersion", "1.0");
paras.put("Timestamp", df.format(new java.util.Date()));
paras.put("Format", "JSON");
paras.put("RegionId", "cn-shanghai");
# API 版本
paras.put("Version", "2019-09-30");
# API ACTION
paras.put("Action", "MakeSuperResolutionImage");
# 下面是业务参数
parasput("Url", "http://viapi-demo.oss-cn-shanghai.aliyuncs.com/viapi-demo/images/MakeSuperResolution/sup-dog.png");
```

   去除签名关键字Key：

```
if (paras.containsKey("Signature")) {
paras.remove("Signature");
```

```
}
```

2. 根据参数Key排序（顺序）。

   参考代码如下：

```
java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<String, String>();
sortParas.putAll(paras);
```

3. 构造待签名的请求串。

   首先介绍下面会用到的特殊URL编码，这个是POP特殊的一种规则，即在一般的URLEncode后再增加如下三种字符替换。

   - 加号 （+）替换成 %20
   - 星号 （*）替换成 %2A
   - %7E 替换回波浪号 （~）

   参考代码如下：

```
public static String specialUrlEncode(String value) throws Exception {
    return java.net.URLEncoder.encode(value, "UTF-8").replace("+", "%20").replace("*",
 "%2A").replace("%7E", "~");
}
```

   构造待签名的请求串，把排序后的参数顺序拼接成如下格式：

```
"&" + specialUrlEncode(参数Key) + "=" + specialUrlEncode(参数值)
```

   参考代码如下：

```
java.util.Iterator<String> it = sortParas.keySet().iterator();
StringBuilder sortQueryStringTmp = new StringBuilder();
while (it.hasNext()) {
    String key = it.next();
    sortQueryStringTmp.append("&").append(specialUrlEncode(key)).append("=").
append(specialUrlEncode(paras.get(key)));
}
String sortedQueryString = sortQueryStringTmp.substring(1);// 去除第一个多余的&符号
```

   打印上面的sortQueryString。结果如下：

```
AccessKeyId=yourAccessId&Action=MakeSuperResolutionImage&Format=JSON&
RegionId=cn-shanghai&SignatureMethod=HMAC-SHA1&SignatureNonce=4a816d44
-6186-4f7e-a45f-ba1b3ed73aed&SignatureVersion=1.0&Timestamp=2019-12-07T13
%3A28%3A52Z&Url=http%3A%2F%2Fviapi-demo.oss-cn-shanghai.aliyuncs.com%
2Fviapi-demo%2Fimages%2FMakeSuperResolution%2Fsup-dog.png&Version=2019-
09-30
```

   对应的未进行URL编码的值（方便进行对比）：

```
AccessKeyId=yourAccessId&Action=MakeSuperResolutionImage&Format=JSON&
RegionId=cn-shanghai&SignatureMethod=HMAC-SHA1&SignatureNonce=4a816d44-
6186-4f7e-a45f-ba1b3ed73aed&SignatureVersion=1.0&Timestamp=2019-12-07T13:
```

```
28:52Z&Url=http://viapi-demo.oss-cn-shanghai.aliyuncs.com/viapi-demo/images/MakeSuperResolution/sup-dog.png&Version=2019-09-30
```

按POP的签名规则拼接成最终的待签名串。规则如下：

```
HTTPMethod + “&” + specialUrlEncode(“/”) + ”&” + specialUrlEncode(sortedQueryString)
```

参考代码如下：

```
StringBuilder stringToSign = new StringBuilder();
stringToSign.append("POST").append("&");
stringToSign.append(specialUrlEncode("/")).append("&");
stringToSign.append(specialUrlEncode(sortedQueryString));
```

这就完成了待签名的请求字符串。结果如下：

```
POST&%2F&AccessKeyId%3DyourAccessId&Action%3DMakeSuperResolutionImage&Format%3DJSON&RegionId%3Dcn-shanghai&SignatureMethod%3DHMAC-SHA1&SignatureNonce%3D4a816d44-6186-4f7e-a45f-ba1b3ed73aed&SignatureVersion%3D1.0&Timestamp%3D2019-12-07T13%253A28%253A52Z&Url%3Dhttp%253A%252F%252Fviapi-demo.oss-cn-shanghai.aliyuncs.com%252Fviapi-demo%252Fimages%252FMakeSuperResolution%252Fsup-dog.png&Version%3D2019-09-30
```

4. 进行签名。

签名采用HmacSHA1算法 + Base64，编码采用UTF-8。参考代码如下：

```
String sign = sign(accessSecret + "&", stringToSign.toString());
public static String sign(String accessSecret, String stringToSign) throws Exception {
    javax.crypto.Mac mac = javax.crypto.Mac.getInstance("HmacSHA1");
    mac.init(new javax.crypto.spec.SecretKeySpec(accessSecret.getBytes("UTF-8"), "HmacSHA1"));
    byte[] signData = mac.doFinal(stringToSign.getBytes("UTF-8"));
    return new sun.misc.BASE64Encoder().encode(signData);
}
```

**表 1-1: 参数说明**

| 参数 | 说明 |
| --- | --- |
| accessSecret | 您的AccessKeyID对应的密钥AccessKeySecret。<br>**说明:**<br>POP要求后面多加一个and（&）字符，即accessSecret + “&” 。 |

| 参数 | 说明 |
| --- | --- |
| stringToSign | 第三步生成的待签名请求串。 |

签名后的结果如下：

```
poMnQhB2W5xndjcsW5VZjSdkvnU=
```

5. 增加签名结果到请求参数中，发送请求。

```
String Signature = specialUrlEncode(sign);// poMnQhB2W5xndjcsW5VZjSdkvnU%3D
```

**说明：**

签名也要做特殊URL编码。

最终完整的POST请求HTTP为：

```
http://imageenhan.cn-shanghai.aliyuncs.com/?Signature=poMnQhB2W5xndjcsW5VZjSdkvnU%3D&AccessKeyId=yourAccessId&Action=MakeSuperResolutionImage&Format=JSON&RegionId=cn-shanghai&SignatureMethod=HMAC-SHA1&SignatureNonce=4a816d44-6186-4f7e-a45f-ba1b3ed73aed&SignatureVersion=1.0&Timestamp=2019-12-07T13%3A28%3A52Z&Url=http%3A%2F%2Fviapi-demo.oss-cn-shanghai.aliyuncs.com%2Fviapi-demo%2Fimages%2FMakeSuperResolution%2Fsup-dog.png&Version=2019-09-30
```

**JAVA示例**

完整的Java签名代码示例。

```
package com.aliyun.imageenhan.demo;
import java.net.URI;
import java.util.HashMap;
import java.util.Map;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URIBuilder;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class SignDemo {
    static final String API_HTTP_METHOD = "POST";
    static final String API_VERSION = "2019-09-30";
    static final java.text.SimpleDateFormat DF = new java.text.SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
    public static void main(String[] args) throws Exception {
        String accessKeyId = "yourAccessId";
        String accessSecret = "yourAccessSecret";
        DF.setTimeZone(new java.util.SimpleTimeZone(0, "GMT"));// 这里一定要设置GMT时区
        Map<String, String> params = new HashMap<>();//业务参数名字是大驼峰
        params.put("Url", "http://viapi-demo.oss-cn-shanghai.aliyuncs.com/viapi-demo/images/MakeSuperResolution/sup-dog.png");
        String action = "MakeSuperResolutionImage";
        execute(action, accessKeyId, accessSecret, params);
```

```
    }
    public static void execute(String action, String accessKeyId, String accessSecret, Map<
String, String> bizParams) throws Exception {
        java.util.Map<String, String> paras = new java.util.HashMap<String, String>();
        // 1. 系统参数
        paras.put("SignatureMethod", "HMAC-SHA1");
        paras.put("SignatureNonce", java.util.UUID.randomUUID().toString());//防止重放攻击
        paras.put("AccessKeyId", accessKeyId);
        paras.put("SignatureVersion", "1.0");
        paras.put("Timestamp", DF.format(new java.util.Date()));
        paras.put("Format", "JSON");
        // 2. 业务API参数
        paras.put("RegionId", "cn-shanghai");
        paras.put("Version", API_VERSION);
        paras.put("Action", action);
        if (bizParams != null && !bizParams.isEmpty()) {
            paras.putAll(bizParams);
        }
        // 3. 去除签名关键字Key
        if (paras.containsKey("Signature")) {
            paras.remove("Signature");
        }
        // 4. 参数KEY排序
        java.util.TreeMap<String, String> sortParas = new java.util.TreeMap<String, String>();
        sortParas.putAll(paras);
        // 5. 构造待签名的字符串
        java.util.Iterator<String> it = sortParas.keySet().iterator();
        StringBuilder sortQueryStringTmp = new StringBuilder();
        StringBuilder sortQueryStringTmp1 = new StringBuilder();
        while (it.hasNext()) {
            String key = it.next();
            sortQueryStringTmp.append("&").append(specialUrlEncode(key)).append("=").
append(specialUrlEncode(paras.get(key)));
            sortQueryStringTmp1.append("&").append(key).append("=").append(paras.get(
key));
        }
        String sortedQueryString = sortQueryStringTmp.substring(1);// 去除第一个多余的&符
号
        StringBuilder stringToSign = new StringBuilder();
        stringToSign.append(API_HTTP_METHOD).append("&");
        stringToSign.append(specialUrlEncode("/")).append("&");
        stringToSign.append(specialUrlEncode(sortedQueryString));
        String sign = sign(accessSecret + "&", stringToSign.toString());
        // 6. 签名最后也要做特殊URL编码
        String signature = specialUrlEncode(sign);
        System.out.println(paras.get("SignatureNonce"));
        System.out.println("\r\n=========\r\n");
        System.out.println(paras.get("Timestamp"));
        System.out.println("\r\n=========\r\n");
        System.out.println(sortedQueryString);
        System.out.println("\r\n=========\r\n");
        System.out.println(stringToSign.toString());
        System.out.println("\r\n=========\r\n");
        System.out.println(sign);
        System.out.println("\r\n=========\r\n");
        System.out.println(signature);
        System.out.println("\r\n=========\r\n");
        // 最终生成出合法请求的URL
        System.out.println("http://imageenhan.cn-shanghai.aliyuncs.com/?Signature=" +
signature + sortQueryStringTmp);

        // 添加直接做post请求的方法
        try {
            // 使用生成的 URL 创建POST请求
```

```
            URIBuilder builder = new URIBuilder("http://imageenhan.cn-shanghai.aliyuncs.
com/?Signature=" + signature + sortQueryStringTmp);
            URI uri = builder.build();
            HttpPost request = new HttpPost(uri);
            HttpClient httpclient = HttpClients.createDefault();
            HttpResponse response = httpclient.execute(request);
            HttpEntity entity = response.getEntity();
            if (entity != null) {
                System.out.println(EntityUtils.toString(entity));
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
    public static String specialUrlEncode(String value) throws Exception {
        return java.net.URLEncoder.encode(value, "UTF-8").replace("+", "%20").replace("*",
 "%2A").replace("%7E", "~");
    }
    public static String sign(String accessSecret, String stringToSign) throws Exception {
        javax.crypto.Mac mac = javax.crypto.Mac.getInstance("HmacSHA1");
        mac.init(new javax.crypto.spec.SecretKeySpec(accessSecret.getBytes("UTF-8"), "
HmacSHA1"));
        byte[] signData = mac.doFinal(stringToSign.getBytes("UTF-8"));
        return new sun.misc.BASE64Encoder().encode(signData);
    }
}
```