ALIBABA CLOUD

阿里云

号码隐私保护
开发指南（旧版）

文档版本：20211130

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.SDK&DEMO下载 05
2.回执消息-HTTP回调方式 06
3.HTTP协议及签名 11

1.SDK&DEMO下载

本文为您介绍号码隐私保护产品相关的SDK&DEMO下载。

隐私保护产品相关的SDK及DEMO程序

文件清单：

1. api_demo（隐私保护服务API接口调用DEMO工程）。
2. api_sdk（隐私保护服务API接口依赖的SDK）。
3. msg_demo（隐私保护回执消息的DEMO）。
4. msg_sdk（隐私保护回执消息的SDK）。

SDK&DEMO下载

如无法下载，建议使用：IE 10 +、Edge、Chrome或Firefox浏览器版本进行下载，目前号码隐私保护控制台产品不支持在Pad、手机等移动设备上使用。如果还是不能下载请确认电脑中是否安装P2P下载工具，当浏览器有下载任务的时候，不能使用P2P下载工具下载，浏览器默认的文件另存；建议关闭P2P下载工具。

1. Java下载地址，请参见Java下载。
2. PYTHON下载地址，请参见PYTHON下载。
3. .NET下载地址，请参见.NET下载。
4. PHP和SDK轻量版下载地址，请参见PHP和SDK轻量版。

   > 说明
   >
   > - PHP仅支持5.5以上版本.
   > - SDK轻量版支持php>=5.3。使用方法：先运行Test.php测试PHP环境，测试成功后再运行Demo.php。

5. Node.js下载地址，请参见Node.js下载。

   > 说明
   >
   > Node.js版本≥v4.6.0。

2.回执消息-HTTP回调方式

本文为您详细介绍隐私号码服务回执消息的HTTP回调方式。

隐私号码通话记录

1.协议说明

参数	说明
协议	HTTP + JSON
编码	UTF-8

2.请求说明

请求内容为JSON Array格式，单次请求可能会包含多个通话记录。

2.1 请求样例

```
[
 {
  "id" : 12345678,
  "partner_key" : "AB_CD",
  "out_id" : "12345678",
  "sub_id" : 12345678,
  "call_id" : "12345678",
  "call_type" : 1,
  "origin_no" : "12345678",
  "phone_no" : "13800000000",
  "secret_no" : "13800000001",
  "peer_no" : "13800000002",
  "release_dir" : 1,
  "release_cause" : 1,
  "start_time" : "2017-01-01 00:00:00",
  "release_time" : "2017-01-01 00:00:00",
  "call_time" : "2017-01-01 00:00:00",
  "ring_time" : "2017-01-01 00:00:00"
 }
]
```

2.2 字段说明

名称	类型	描述	示例值	是否必须
id	Number	id。	12345678	必须
partner_key	String	商户Key。	AB_CD	必须
out_id	String	用户序列号。	12345678	可选
sub_id	Long	订单ID。	12345678	可选
call_id	String	通话ID。	12345678	可选
call_type	Number	呼叫类型。取值： • **0**：主叫。 • **1**：被叫。 • **2**：短信发送。 • **3**：短信接收。	1	可选
origin_no	String	通话ID。	12345678	可选
phone_no	String	用户号码。	138****0000	必须
secret_no	String	隐私号码。	138****0000	必须
peer_no	String	对端号码。	138****0000	必须
release_dir	Number	释放方向。取值： • **0**：平台释放。 • **1**：主叫释放。 • **2**：被叫释放。	1	可选
release_cause	Number	释放原因。详细信息，请参见下表。	1	可选
start_time	String	通话开始时间。	2017-01-01 00:00:00	可选

名称	类型	描述	示例值	是否必须
release_time	String	通话结束时间。	2017-01-01 00:00:00	可选
call_time	String	拨打时间。	2017-01-01 00:00:00	可选
ring_time	String	振铃时间。	2017-01-01 00:00:00	可选

释放原因

编号	说明
1	未分配的号码。
2	无路由到指定的转接网。
3	无路由到目的地。
4	发送专用信息音。
16	正常的呼叫拆线。
17	用户忙。
18	用户未响应。
19	用户未应答。
20	用户缺席。
21	呼叫拒收。
22	号码改变。
27	目的地不可达。

编号	说明
28	无效的号码格式（地址不全）。
29	性能拒绝。
31	正常—未指定。
34	无电路或通路可用。
42	交换设备拥塞。
50	所请求的性能未预定。
53	CUG中限制去呼叫。
55	CUG中限制来呼叫。
57	承载能力无权。
58	承载能力目前不可用。
65	承载能力未实现。
69	所请求的性能未实现。
87	被叫用户不是CUG的成员。
88	不兼容的目的地。
90	不存在的CUG。
91	无效的转接网选择。
95	无效的消息，未指定。

编号	说明
97	消息类型不存在或未实现。
99	参数不存在或未实现。
102	定时器终了时恢复。
103	参数不存在或未实现—传递。
110	消息带有未被识别的参数—舍弃。
111	协议错误，未指定。
127	互通，未指定。

3 应答说明

3.1 应答样例

```
{
 "code":0,
 "msg":"接收成功"
}
```

3.1 字段说明

参数	类型	描述	示例值	是否必须
code	Number	应答编码。	0	必须
msg	String	描述信息。	接收成功	可选

3.HTTP协议及签名

本文为您详细介绍隐私号码服务涉及的HTTP协议及签名。

为方便其它语言能快速实现，在此增加了HTTP的详解。

> **注意**
>
> 云通信下的四类产品（包括短信、语音、流量、隐私保护），接口协议一致。

一、前言说明

对应的协议是：发生通话行为后的话单消息回执采用的是阿里云的消息中间件MNS来实现。

二、协议说明

由于POP的签名算法比较复杂，下面这里结合AXB的绑定API进行详细说明。

2.1 号码隐私保护发送API的HTTP协议包示例

请求示例：

```
GET /?Signature=zJDF%2BLrzhj%2FThnlvIToysFRq6t4%3D&AccessKeyId=testId&Action=BindAxb&Format=X
ML&OutId=123&PhoneNumbers=15300000001&RegionId=cn-hangzhou&SignName=%E9%98%BF%E9%87%
8C%E4%BA%91%E7%9F%AD%E4%BF%A1%E6%B5%8B%E8%AF%95%E4%B8%93%E7%94%A8&SignatureM
ethod=HMAC-SHA1&SignatureNonce=45e25e9b-0a6f-4070-8c85-2956eda1b466&SignatureVersion=1.0&Pool
Key=FC123456&PhoneNoA=170000000&PhoneNoB=171000000&ExpireDate=2017-07-12T02%3A42%3A19Z&T
imestamp=2017-07-12T02%3A42%3A19Z&Version=2017-05-25
HTTP/1.1
Host: dyplsapi.aliyuncs.com
...
```

返回示例：

```
HTTP/1.1 200 OK
...
<?xml version='1.0' encoding='UTF-8'?><BindAxbResponse>
<Code>OK</Code>
<Message>OK</Message>
<RequestId>E8534574-7381-4810-8F70-65B37BBA8970</RequestId><SubsId>108374502347^1111325525761
</SubsId>
<SecretNo>1350000000</SecretNo>
</BindAxbResponse>
```

2.2 号码隐私保护Rest请求参数

如HTTP示例中，请求的参数可以分两大块：系统参数和业务参数。系统参数为POP协议的基本参数。

系统参数

参数	是否必填	说明
AccessKeyId	是	
Timestamp	是	格式为：yyyy-MM-dd'T'HH:mm:ss'Z'；时区为：GMT。
Format	否	没传默认为JSON，可选填值：XML。
SignatureMethod	是	建议固定值：HMAC-SHA1。
SignatureVersion	是	建议固定值：1.0。
SignatureNonce	是	用于请求的防重放攻击，每次请求唯一，JAVA语言建议用：java.util.UUID.randomUUID()生成即可。
Signature	是	最终生成的签名结果值。

业务参数

参数KEY	是否必填	说明
Action	是	API的命名，固定值，如发绑定AXB：BindAxb
Version	是	API的版本，固定值，如短信API的值为：2017-05-25。
RegionId	是	API支持的RegionID，如短信API的值为：cn-hangzhou。
PoolKey	是	具体请参见API文档。

参数KEY	是否必填	说明
PhoneNoA	是	具体请参见API文档。
PhoneNoB	是	具体请参见API文档。
Expiration	否	具体请参见API文档。
OutId	否	具体请参见API文档。

2.3 生成签名

签名是为了让请求合法，共分为五步：

第一步：请求参数。

1. 请求参数包括系统参数和业务参数，不要遗漏。
2. 请求参数中不允许出现以Signature为key的参数。 参考代码如下。

```
String accessKeyId = "testId";
String accessSecret = "testSecret";
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>();
```

①系统参数

```
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", "XML");
```

②业务API参数

```
paras.put("Action", "BindAxb");
paras.put("Version", "2017-05-25");
paras.put("RegionId", "cn-hangzhou");
paras.put("PoolKey", "FC123456");
paras.put("PhoneNoA", "1700000000");
paras.put("PhoneNoB", "1700000000");
paras.put("Expiration", "2017-05-25 00:00:00");
paras.put("OutId", "123");
```

③去除签名关键字Key

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

第二步：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);
```

第三步：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这里有两步动作。

第1步，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(specialUrlEncod
e(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=testId&Action=BindAxb&Format=XML&OutId=123&RegionId=cn-hangzhou&SignatureMethod=
HMAC-SHA1&SignatureNonce=45e25e9b-0a6f-4070-8c85-2956eda1b466&SignatureVersion=1.0&PoolKey=FC
123456&PhoneNoA=170000000&PhoneNoB=17100000000&Expiration=2017-07-12T02%3A42%3A19Z&Timest
amp=2017-07-12T02%3A42%3A19Z&Version=2017-05-25`
```

第2步，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("GET").append("&");
stringToSign.append(specialUrlEncode("/")).append("&");
stringToSign.append(specialUrlEncode(sortedQueryString));
```

这就完成了待签名的请求字符串了。打印结果如下：

```
GET&%2F&AccessKeyId%3DtestId&Action%BindAxb&Format%3DXML&OutId%3D123&PhoneNoA%3D17000
000000&RegionId%3Dcn-hangzhou&PhoneNob%3D171000000000&SignatureMethod%3DHMAC-SHA1&Sign
atureNonce%3D45e25e9b-0a6f-4070-8c85-2956eda1b466&SignatureVersion%3D1.0&PoolKey%3DFC123456
&Expiration%3D2017-07-12T02%253A42%253A19Z&Timestamp%3D2017-07-12T02%253A42%253A19Z&Vers
ion%3D2017-05-25`
```

第四步：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);
}
```

参数说明： 1. `accessSecret` ：你的 `AccessKeyId` 对应的密钥 `AccessSecret` ，特别说明：POP要求需要后面多加一个“ `&` ”字符，即 `accessSecret + “&”` 2. `stringToSign` ：即第三步生成的待签名请求串

签名后的结果打印如下：

```
zJDF+Lrzhj/ThnlvIToysFRq6t4=
```

第五步：增加签名结果到请求参数中，发送请求 。

> **注意**
>
> 签名也要做特殊URL编码。

```
String Signature = specialUrlEncode(sign);// zJDF%2BLrzhj%2FThnlvIToysFRq6t4%3D
```

三、附加完整的Java签名Demo代码

以云通信短信产品-发送接口为列进行说明，业务参数自行替换成隐私保护绑定接口参数即可。

```
public class SignDemo {
    public static void main(String[] args) throws Exception {
        String accessKeyId = "testId";
        String accessSecret = "testSecret";
        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>();
        // 1. 系统参数
        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", "XML");
        // 2. 业务API参数
        paras.put("Action", "SendSms");
        paras.put("Version", "2017-05-25");
        paras.put("RegionId", "cn-hangzhou");
        paras.put("PhoneNumbers", "15300000001");
        paras.put("SignName", "阿里云短信测试专用");
        paras.put("TemplateParam", "{\"customer\":\"test\"}");
        paras.put("TemplateCode", "SMS_71390007");
        paras.put("OutId", "123");
        // 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();
        while (it.hasNext()) {
            String key = it.next();
            sortQueryStringTmp.append("&").append(specialUrlEncode(key)).append("=").append(specialU
rlEncode(paras.get(key)));
        }
        String sortedQueryString = sortQueryStringTmp.substring(1);// 去除第一个多余的&符号
        StringBuilder stringToSign = new StringBuilder();
        stringToSign.append("GET").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");
        // 最终打印出合法GET请求的URL
        System.out.println("http://dysmsapi.aliyuncs.com/?Signature=" + signature + sortQueryStringTmp);
    }
    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);
    }
}
```