

# 上上签 API 接口文档

## BestSign OpenAPI 混合云 V2.26.1

### 目录

|                                |    |
|--------------------------------|----|
| 目录.....                        | 1  |
| 一、API 架构介绍 .....               | 3  |
| 1、混合云架构介绍.....                 | 3  |
| 2、建议配置.....                    | 3  |
| 3、混合云接入方法（必读） .....            | 4  |
| 4、注意事项（必读） .....               | 8  |
| 二、API 协议须知 .....               | 9  |
| 1、    调用地址（host） .....         | 9  |
| 1.1、sdk-host.....              | 9  |
| 1.2、测试与生产的切换方法.....            | 9  |
| 2、调用流程.....                    | 10 |
| 3、请求类型.....                    | 10 |
| 1）、POST/GET.....               | 10 |
| 2）、字符集.....                    | 10 |
| 3）、JAVA 的特殊处理 .....            | 10 |
| 4、请求格式.....                    | 10 |
| 5、生成签名（计算 sign 参数） .....       | 11 |
| 1）、先得到签名原文字符串.....             | 11 |
| A）、POST 方法原文字符串示例 .....        | 12 |
| B）、GET 方法原文字符串示例 .....         | 12 |
| 2）、签名计算方法（rsa 算法） .....        | 12 |
| 6、接口响应.....                    | 13 |
| 7、异步通知.....                    | 14 |
| 8、返回消息体（response）加密说明.....     | 16 |
| 使用方法.....                      | 16 |
| 加密原理.....                      | 16 |
| 三、API 调用流程 .....               | 18 |
| 1、注册用户.....                    | 20 |
| 接口调用流程.....                    | 20 |
| 2、快捷签署（自动签） .....              | 20 |
| 接口调用流程.....                    | 20 |
| 3、手动签署.....                    | 21 |
| 接口调用流程.....                    | 21 |
| 4、PDF 签名图片尺寸、位置的计算方法（必读） ..... | 21 |
| 4.1、图片尺寸的计算方法.....             | 22 |
| 4.2、位置坐标的计算方法.....             | 22 |
| 四、标配 API 功能列表 .....            | 23 |
| 1、用户服务.....                    | 23 |
| 1.1、注册个人用户并申请证书.....           | 23 |
| 1.2、注册企业用户并申请证书.....           | 26 |

|                               |    |
|-------------------------------|----|
| 1.3、查询证书编号.....               | 30 |
| 1.4、查询个人用户证件信息.....           | 31 |
| 1.5、查询企业用户证件信息.....           | 33 |
| 1.11、异步申请状态查询.....            | 34 |
| 1.12、获取证书详细信息.....            | 35 |
| 2、签名/印章图片服务 .....             | 37 |
| 2.1、生成用户签名/印章图片 .....         | 37 |
| 2.2、上传用户签名/印章图片 .....         | 38 |
| 2.3、下载用户签名/印章图片 .....         | 40 |
| 3、文件存储服务.....                 | 41 |
| 3.1、上传合同文件.....               | 41 |
| 3.2、PDF 文件添加元素 .....          | 42 |
| 3.3、下载文件.....                 | 44 |
| 3.4、PDF 文件验签 .....            | 44 |
| 4、单文档合同服务.....                | 46 |
| 4.1、上传并创建合同.....              | 46 |
| 4.2、签署合同（即自动签） .....          | 49 |
| 4.3、发送合同（即手动签，指定图片大小） .....   | 52 |
| 4.4、锁定并结束合同.....              | 56 |
| 4.5、撤销合同.....                 | 57 |
| 4.6、查询合同信息.....               | 58 |
| 4.7、查询合同签署参数.....             | 60 |
| 4.8、查询合同签署者状态.....            | 61 |
| 4.9、下载合同文件.....               | 62 |
| 4.10、获取预览页 URL.....           | 63 |
| 4.11、创建合同.....                | 64 |
| 4.20、模版签署功能.....              | 66 |
| 4.20.0、获取模版变量.....            | 69 |
| 4.20.1、通过模版生成合同文件.....        | 71 |
| 4.20.2、通过模版创建合同.....          | 73 |
| 4.20.3、用模版变量签署合同.....         | 74 |
| 4.20.4、获取模版信息.....            | 76 |
| 4.20.5、用模版变量的手动签.....         | 80 |
| 4.20.6、获取创建模版的地址.....         | 82 |
| 4.20.7、获取编辑模版的地址.....         | 84 |
| 4.20.8、获取开发者模版列表.....         | 85 |
| 4.20.9、预览模版.....              | 87 |
| 4.21、获取合同文件列表.....            | 88 |
| 4.22、生成合同附页.....              | 89 |
| 4.23、下载合同附页文件.....            | 90 |
| 4.24、关键字定位签署合同.....           | 91 |
| 4.25、在线验签（通过合同 ID 和哈希值） ..... | 94 |
| 4.26、获取存证页 URL.....           | 95 |
| 12、支付服务.....                  | 96 |
| 12.1、微信支付.....                | 96 |
| 12.1、查询支付订单.....              | 97 |

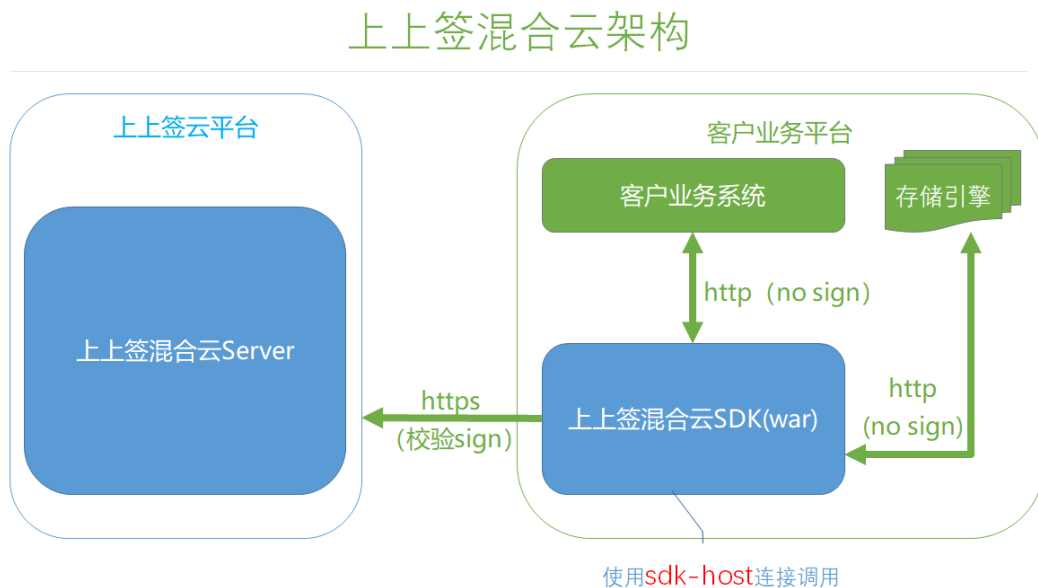
## 一、API 架构介绍

上上签 OpenAPI 同时支持公有云模式、混合云模式，两种模式有各自不同的特点，适用于不同的场景。本文档内容适用于混合云模式。

### 1、混合云架构介绍

上上签混合云提供了一个**混合云 SDK**，上上签的所有服务都通过调用这个 SDK 使用，功能包括：提供合同签署的源文件、过程文件、结果文件、签名图片 Data 信息（data-information，数据的实体信息，包括文件流，图片流）存储、计算等功能，实现“文档不出门”。该混合云 SDK，采用 war 包的形式部署在开发者自己的服务器，提供 Http 接口供开发者使用。调用这部分 API 使用的 host 是 **sdk-host**，即开发者向自己部署了混合云 SDK 的服务器地址发起调用请求。

这个架构如下图所示：



### 2、建议配置

SDK 包的运行环境如下，请按照以下要求独立部署：

| 类别   | 最低配置  | 建议配置   |
|------|---|--|
| 硬件要求 | CPU 2 核 2.0GHz<br>内存 4GB DDR2<br>硬盘 100GB   | CPU 四核至强处理器 2.0GHz 或以上<br>内存 8GB DDR2 或以上<br>硬盘 500GB SATA 硬盘或以上 |
| 软件要求 | 操作系统: CentOS 7.x/ubuntu 16.x/windows 7/8/10/server2008/server2012<br>服务容器: Tomcat8.0 release 稳定版<br>运行环境: Oracle JDK 8.0  |  |
| 网络要求 | SDK 所在的服务器需要开通互联网访问权限, 或者允许通过代理访问公网, 带宽 2M 以上。<br>SDK 所在的服务器测试环境可支持 HTTP 和 HTTPS 访问, 正式环境必须支持 HTTPS 访问。<br>业务平台内部访问 war 包直接通过 <a href="http://ip:port">http://ip:port</a> 访问, 请勿使用 https. |  |

### 3、混合云接入方法（必读）

1. 在实施指导下注册上上签账号, 获取开发者 ID (developerId), 生成公钥, 私钥 (可以使用 OpenSSL 生成, 到 OpenSSL 官网下载工具; 也可以使用 RSA 在线生成工具生成), 准备混合云的运行环境 (建议 linux+tomcat8+jdk8)。

war 包的集成运行环境 (已经集成了 tomcat8+jdk8) :

- ① linux 环境: <https://demotest.bestsign.cn/download/distributed-server-env-linux.zip>
- ② windows 环境: <https://demotest.bestsign.cn/download/distributed-server-env-windows.zip>

使用方法: 下载对应环境的 zip 包, 解压缩到系统目录 (目录可以自行指定) 下。

启停服务:

#### 1)、Linux 启停

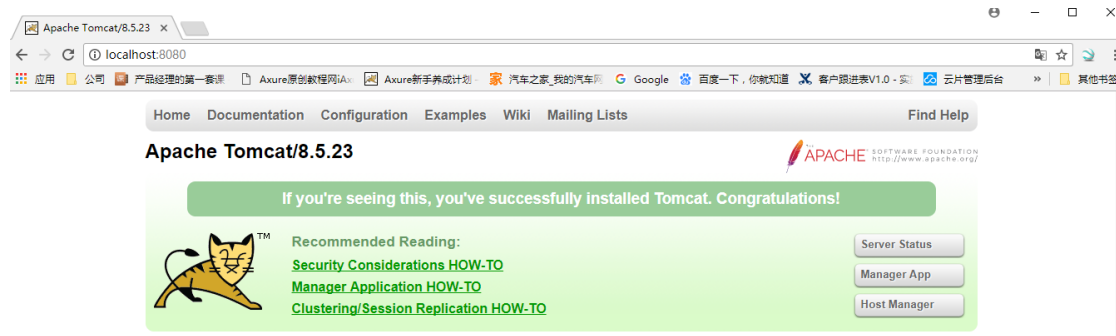
以上工作准备就绪后, 就可以使用 /usr/local/distributed-server-env-linux 下的 shell 脚本来启停 SDK 服务了, 启停命令如下:

```
首次运行脚本, 需要先对 tomcat 的 shell 脚本授权: chmod 777 /usr/local/distributed-server-env-linux/*.sh
```

```
启动 Tomcat: sh /usr/local/distributed-server-env-linux/startup.sh
```

```
停止 Tomcat: sh /usr/local/distributed-server-env-linux/shutdown.sh
```

启动成功后, 打开 <http://ip:port>, 显示下图则代表 tomcat 启动成功:



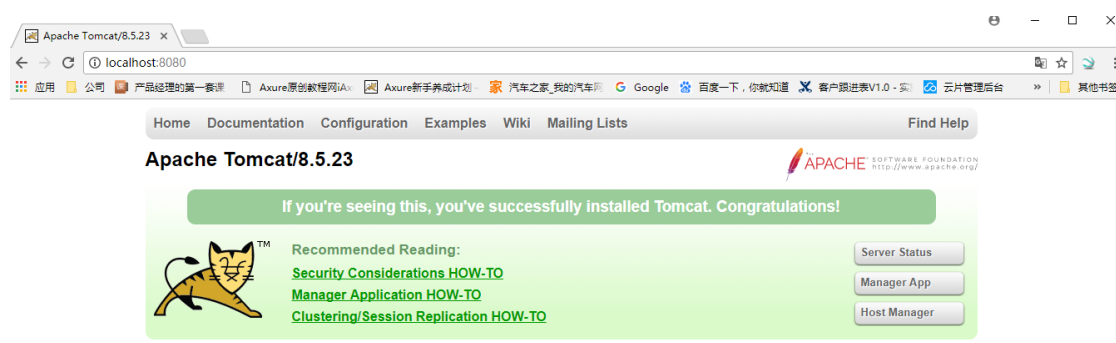
## 2)、Windows 启停

使用 D:\server\distributed-server-env-windows\下可以使用 D:\server\distributed-server-env-windows\下的 shell 脚本来启停 SDK 服务了，启停命令如下：

启动 Tomcat：双击 D:\server\distributed-server-env-windows\startup.sh

停止 Tomcat：双击 D:\server\distributed-server-env-windows\shutdown.sh

启动成功后，打开 <http://ip:port>，显示下图即代表 tomcat 启动成功：



2. 获取 sign-api-v2-distributed.war 包和本文档（由实施人员下发），并向实施申请开通上上签开发者中心权限。

3. 解压缩 sign-api-v2-distributed.war 到 tomcat/webapps/ROOT 目录下，如下图所示：



4. 在 SDK 目录（

Linux: [usr/local/distributed-server-env-linux/tomcat8/webapps/ROOT/WEB-INF/classes/conf](#),

Windows: `D:\server\distributed-server-env-windows\tomcat8\webapps\ROOT\WEB-INF\classes\conf`)下找到 `common.properties` 配置文件，修改开发者 ID (`developerId`)，公钥，私钥，存储引擎地址配置。

并在 `common.properties` 配置文件中新增 `app.bestsign.revert.server=1`

另注：

以前 `common.properties` 配置文件有个 `http` 的超时配置搞错了，会导致 `http-client` 的超时时间设置无效。

错误配置：

`app.http.default.connection.timeout=800`

`app.http.default.read.timeout=5000`

正确配置：

`app.http.default.connectTimeout=800`

`app.http.default.readTimeout=5000`

具体修改参考文档：【混合云开发者接入流程】

查看地址：<https://www.fangcloud.com/share/b065ebee69f41e6550a48e1403>

查看密码：bestsign

5. 自定义实现一个存储引擎 `http` 服务，具体可参考文档【混合云存储引擎实现方法】

查看地址：<https://www.fangcloud.com/share/c136527585313824fab6490c00>

查看密码：bestsign

| 请求方法              | 请求报文   | 响应类型   | 响应报文（正常）   | 响应报文（异常）                        |
|-------------------|--|--------|--|---------------------------------|
| 上传文件 (set)        | <pre>{   "method": "set",   "data": {     "dhash": "10000000001",     "fdata": "BASE64字符串"   } }</pre> | String | Http-status: 200<br>(串)                                  | Http-status: 400<br>empty fdata |
| 下载文件 (get)        | <pre>{   "method": "get",   "data": {     "dhash": "10000000001"   } }</pre>                           | byte[] | Http-status: 200<br>byte[]流                              | 返回文件                            |
| 查询文件是否存在 (exists) | <pre>{   "method": "exists",   "data": {     "dhash": "10000000001"   } }</pre>                        | String | Http-status: 200<br>串: "0" 或者 "1"。 "0"--文件不存在, "1"--文件存在 | 返回字符                            |
| 删除文件 (del)        | <pre>{   "method": "del",   "data": {     "dhash": "10000000001"   } }</pre>                           | String | Http-status: 200<br>(串)                                  | (返回空字符)                         |

```
#####
# 存储引擎配置
#####
# 存储引擎，支持两种：

# storage.default.driver 是默认存储引擎，必须配置。如果其他引擎未配置，则默认使用该引擎
存储

# storage.signatureimage.driver 签名图片存储引擎，可选配置，所有的签名图片都使用该引
擎存储，如果不配置，则使用默认存储引擎

# storage.persistence.driver 永久存储引擎，可选配置，签署完成的最终合同，会转移到该引
擎存储，如果不配置，则使用默认存储引擎

# storage.temp.driver 临时存储引擎，可选配置，合同签署产生的中间文件存储，该引擎中的数
据可以由客户自行定期清理，如果不配置，则使用默认存储引擎

# 配置格式是类名，参数列表，用竖线分隔 如 HttpStorage|
http://127.0.0.1:8080/storage

# 注：HttpStorage 是固定值，无需修改，修改红色标记的部分为您的本地 http 存储地址

storage.default.driver=HttpStorage|http://127.0.0.1:8080/storage

storage.persistence.driver=HttpStorage|http://127.0.0.1:8080/storage

storage.temp.driver=HttpStorage|http://127.0.0.1:8080/storage

storage.signatureimage.driver=HttpStorage|http://127.0.0.1:8080/storage
```

存储引擎实现以后，需要在 **common.properties** 配置文件中将

“storage.default.driver”和“storage.signatureimage.driver”的路径

“http://127.0.0.1:8080/storage”替换成如

http://www.yourdomain.com:8088/storage 这样完整的实际地址。其中，

“storage.default.driver”必须要配置，其他存储引擎如

“storage.signatureimage.driver”路径如果为空，即

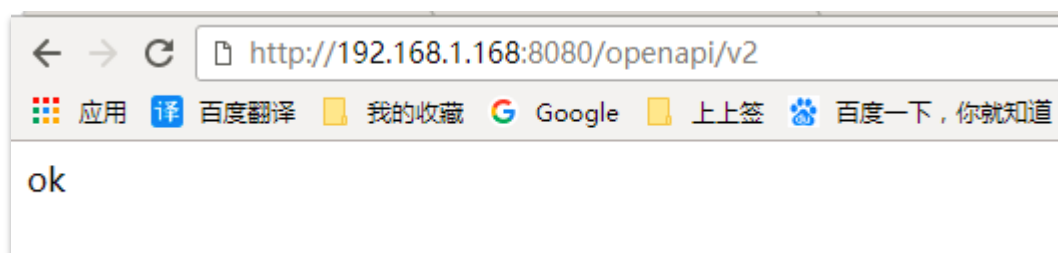
“storage.signatureimage.driver=”这样子，表示图片存储与文件存储共用同一个存  
储。各存储路径的使用优先级说明如下：

| 配置路径                       | 使用优先级   |
|----------------------------|---|
| storage.default.driver     | 当下列 3 个路径都没有配置时此配置才会生效，必须配置。  |
| storage.persistence.driver | 永久存储，用于存合同的最终文件（指签署流程已结束，即<br>调用“锁定并结束合同”或“结束合同”接口成功之后合同文<br>件）。此配置比 default 优先使用。如此项未配置就会使 |

|                               |   |
|-------------------------------|---|
|                               | 用 default 项。  |
| storage.temp.driver           | 临时存储，用于存储签署过程中产生的临时文件，此配置比 default 优先使用。如此项未配置就会使用 default 项。 |
| storage.signatureimage.driver | 用于存储签名图片，此配置比 default 优先使用。如此项未配置就会使用 default 项。              |

6. 以上步骤全部完成后，启动 Tomcat，SDK 部署结束。

启动成功后，打开 <http://ip:port/openapi/v2>，显示“ok”即代表 war 部署成功，如下图所示：



7. 如果需要使用手动签功能，需要把 sign-api-v2-distributed.war 部署的 host 地址通知上上签实施进行 Server 端配置。

准备工作结束，接下来就可以进行相应的接口调用了。

## 4、注意事项（必读）

使用 war 包 host 代理访问 server 有几个前提条件：

1. common.properties 配置新增 app.bestsign.revert.server=1
2. 业务平台内部访问 war 包直接通过 <http://ip:port> 访问，请勿使用 https
3. war 包必须部署在 tomcat 的 ROOT 目录下，通过根目录访问 url
4. war 包 common.properties 的配置请勿修改 app.service.path=openapi, app.service.version=v2
5. 混合云支持代理模式，需要代理时在配置文件 common.properties 中添加以下配置项：



```
# http proxy

http.proxy.enable=false

#enable=false 表示无需代理, enable=true 表示需要代理

http.proxy.host=192.168.30.183

#host 表示代理服务器的地址

http.proxy.port=10001

#port 表示代理服务器的端口号
```

## 二、API 协议须知

上上签 OpenAPI 基于 HTTPS 协议，您可以自行封装请求进行调用，以下对自行封装请求进行说明。

### 1、调用地址（host）

#### 1.1、sdk-host

**sdk-host:** 混合云模式下用于调用部署在开发者本地的混合云 war 的接口，是开发者自行配置的 host 地址，例如：

<http://ip:port/openapi/v2>

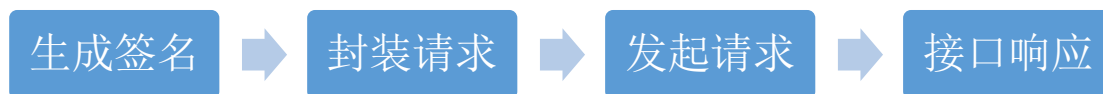
请注意：①必须是 <http://ip:port> 的格式；②在将 host 与接口方法拼接到一起时，v2 后面的斜杠“/”只需要 1 个，如果有 2 个“/”会出现 404 错误请求失败。

#### 1.2、测试与生产的切换方法

开发者切换测试环境与生产环境通过修改 `common.properties` 中的以下标红的几处配置实现：

```
1. # 开发者配置
2. # *****
3. app.developer.id=请填写您的开发者 ID
4. app.developer.authKey=请填写您的开发者私钥
5. app.developer.pubKey=请填写您的开发者公钥
6. # 上上签云服务环境配置，测试环境为：
   https://openapi.bestsign.info/openapi/v2, 正式环境为：
   https://openapi.bestsign.cn/openapi/v2
7. app.bestsign.server=https://openapi.bestsign.info/openapi/v2
```

## 2、调用流程



## 3、请求类型

### 1)、POST/GET

POST 方式提交的请求，BODY 一律使用 JSON 格式。请求 JSON 格式时，Headers 里需要设置 Content-Type 为 application/json。

GET 方式提交的请求，Headers 和 Request Body 都不需要。

### 2)、字符集

字符编码设置为 UTF-8。

### 3)、JAVA的特殊处理

HttpClientUtil.httpPostRequest 这个方法里面改一下

```
ByteArrayEntity byteArrayEntity = new ByteArrayEntity(JSON.toJSONString(params).getBytes("UTF-8"));
```

```
httpPost.setEntity(byteArrayEntity);
```

把 StringEntity 那里，改成 ByteArrayEntity

系统底层好像会把 string 转换成 charAt，所以 utf8 的中文就不知道它会怎么处理，开发人员在外面把它转成 utf8 的字节数组，不让系统自动转。

## 4、请求格式

GET/POST [host]/Service/Method/?developerId=开发者编号&rtick=unix 时间戳+4 位随机数&signType=签名串计算方法&sign=签名串&[其他 URL 参数.....]

如果是 POST 请求，把请求参数，以 json 格式放 body 里，同时设置 Content-Type 为 application/json。

一般的请求 URL 格式如下：

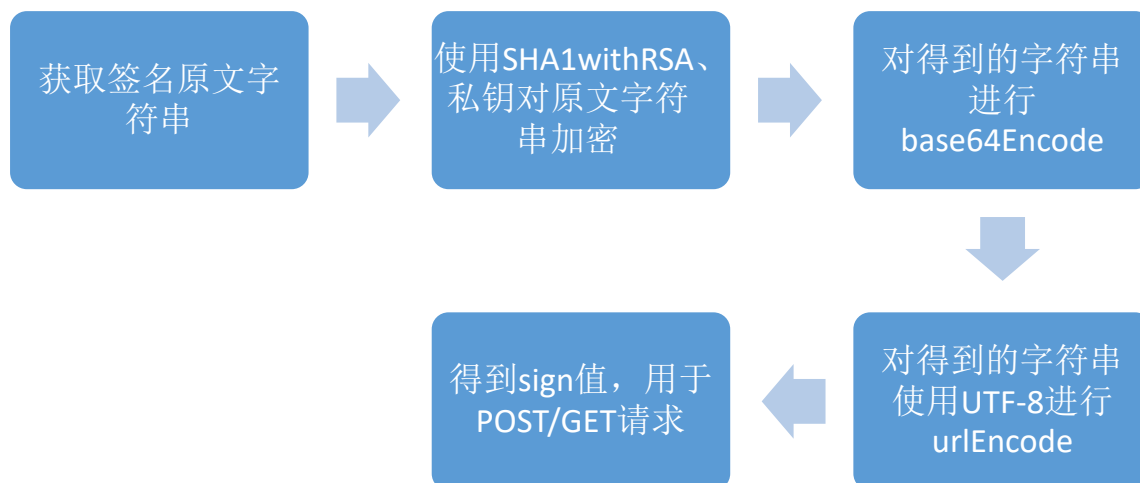
```
http://ip:port/openapi/v2/user/reg/?developerId=3333&rtick=15066659931690&signType=rsa&sign=OSIBr6v2u121r8QwE7mBRhC9iuRUWw24W8o1%2Fm87zqs6
```

```
s1fDEVwAjz8S7939p2eXAWpcnuOsPU13eD1Pk1LL7EiyQBIXAaIQBUXrhjqX%2BNpG3qd  
vuSkbY5g%2BRdlLHXS3sR%2FeXBtIvCJAjY6yM9iA%2Fm9gtY3mJyrGh83APb3m5Lw%3D
```

其中，**developerId**, **rtick**, **sign**, **signType** 这几个参数是每一个接口都必须有的参数。

url 里的其他参数，跟具体的接口相关。

## 5、生成签名（计算 sign 参数）



### 1）、先得到签名原字符串

(1) 查看您申请的开发者编号（developerId）。假设您的 developerId 为 aaaaaa，rtick 为 14840209310001。

(2) 将 url 中，需要参与签名计算的各 url 参数，按参数名的字典序进行排序，并将排序后的各参数按“**参数名 1=参数值 1** **参数名 2=参数值 2**..... **参数名 n=参数值 n**”的格式连接成一个字符串（各参数之间，不要用&分隔）。

(3) 将请求的 HTTP Request Path，如/openapi/v2/user/reg，附加到这个字符串后面。

(4) 如果是 POST 请求，有 Request Body，将 Request Body 的 md5 值，再附加到这个字符串后面得到最终的签名原字符串。

(5) 对签名原字符串进行签名计算，签名计算方法为 rsa 算法，通过 url 参数 signType 指定。

## A)、POST 方法原文字符串示例

假设调用注册用户接口, POST 请求/openapi/v2/user/reg, 使用 rsa 算法计算签名, 时间戳为 14840209310001, Request Body 为:

```
{"mail": "test@bestsign.cn", "type": "2"}
```

则请求的 URL 为:

```
/openapi/v2/user/reg/?developerId=aaaaaa&rtick=14840209310001&signType=e=rsa
```

按如下步骤得到签名原字符串:

- 先把需要签名计算的参数进行字典序排序, 得到一个字符串:

```
developerId=aaaaaartick=14840209310001signType=rsa
```

- 把请求 path 附加到字符串后面:

```
developerId=aaaaaartick=14840209310001signType=rsa/openapi/v2/user/reg/
```

- 计算 request body 的 md5 值(假设我们的 request body 的 md5 值是

fb5939fd51b1a929a6e2b75b187827f7), 附加到字符串后面, 得到最终的签名原字符串:

```
developerId=aaaaaartick=14840209310001signType=rsa/openapi/v2/user/reg/fb5939fd51b1a929a6e2b75b187827f7
```

## B)、GET 方法原文字符串示例

因为 GET 方法没有 Headers 和 Request Body, 因此原字符串与 POST 方法略有差别。以 /openapi/v2/signatureImage/user/download 为例:

参数按照参数名进行字典排序后得到签名:

- 先把需要签名计算的参数进行字典序排序, 得到一个字符串:

```
account=136developerId=1709264203795136512imageName=rtick=14928387557256980signType=e=rsa
```

- 把请求 path 附加到字符串后面, 得到签名原字符串:

```
account=136developerId=1709264203795136512imageName=rtick=14928387557256980signType=e=rsa/openapi/v2/signatureImage/user/download/
```

## 2)、签名计算方法 (rsa算法)

使用 rsa 算法对这个签名原文字符串进行签名计算。

URL 参数中的 signType 需要设为 rsa，然后使用私钥对这个签名原文字符串进行 rsa 加密。

用伪码表示这个过程：

```
rsa("developerId=aaaaaartick=14840209310001signType=rsa/openapi/v2/user/reg/fb5939fd51b1a929a6e2b75b187827f7")
```

得到 sign:

```
ECaLEwclWq+XzaGDO5I5luL35Ad+LRpUcunbVj5WGQzCAH7SBIH6EBoZaYLwo5Kicbv/Iy1t8PveILhlUDDJQL+AgB6sUfSdrT6j6Xk7U9Na/VQAsz3/DMmE8hxUK0jDlO196egh5WsLZdy5Lg9XJqVclCou4VTq4xWYvaMVTIw=
```

注：

- ①使用的 rsa 算法为 SHA1withRSA;
- ②rsa 之后需要进行一次 base64Encode，之后再使用 UTF-8 进行一次 urlEncode（因为 base64Encode 之后的字符串可能会带有“/”等特殊符号），得到的字符串才能用于 http 请求的 url 参数。
- ③特别注意：如果拼装到 http 请求 url 里的参数带有“/”等特殊符号的，则该参数需要先进行一次 urlEncode，得到的字符串才能拼到 http 请求 url 里。对参数的 urlEncode 只需做一遍即可，重复 urlEncode 之后使用会出错。

## 6、接口响应

业务正常返回 HTTP 200。

API 的响应包的 Body 部分采用 JSON 格式编码，因此需要对返回包进行 JSON 解析。

如果是下载类的请求（即响应的 Body 是数据流，不是 JSON 格式），无法使用 JSON 输出响应。业务正常时会正常返回数据流，当发生业务异常时，服务端会返回 5xx 类型的 HTTP 状态码，同时会将 errno 和 errmsg 放到 Header 中输出。

其他类型的请求，将会以 JSON 格式输出数据。输出格式：

```
{
  "errno": 0,          // 错误码，如果为 0 表示成功没有错误
  "data": { },         // 输出数据（如果有的话，具体内容和每一个具体的接口相关）
  "cost": 533,         // 接口内部处理花费的时间，单位是 ms (毫秒)
  "errmsg": ""         // 错误描述，结合 errno 对照附录
}
```

其中：

- errno 返回 0 则为正常，errno 返回错误码则为异常，此时观察具体的 errmsg 具体分

析。errno 与 errmsg 请参阅本文《四、附录：返回信息编号列表》，返回信息编号列表持续更新。

- cost 是本接口的响应时间，表示上上签从收到请求到给出响应所经历的时间（上上签内部处理的时间），单位是毫秒，数字格式。
- data 是接口的具体返回参数，参见具体接口。

## 7、异步通知

有的开发者需要上上签主动将结果通知给开发者，因此我们需要开发者向上上签提供一个 pushUrl，开发者将这个 pushUrl 告知上上签，由上上签设置到开发者配置中，并且将推送消息格式及 accessKey 告知开发者。

此 pushUrl 的基本要求：

- （1）必须支持 http 或者 https 格式；
- （2）请求类型必须为 POST；
- （2）能够接收消息并返回 http 状态为 200；
- （3）必须是一个完整的 URL（如：https://ip:port/message/receiveBestsign.do），不能是一个相对路径（/message/receiveBestsign.do）；
- （4）安全要求：使用上上签实施提供的 accessKey 加解密。

1、异步通知针对下列接口有效：

- "1.1、注册个人用户并申请证书"—异步申请证书的申请结果会异步推送；
- "1.2、注册企业用户并申请证书"—异步申请证书的申请结果会异步推送；
- "4.3、发送合同"—将签署链接给用户进行手动签署的签署成功结果会异步推送；
- "4.20.6、获取创建模版的地址"—创建模版保存后的结果会异步推送；
- "4.20.7、获取编辑模版的地址"—编辑模版保存后的结果会异步推送；
- "12.1、微信支付"—通过微信扫码支付后的支付结果会异步推送；

2、推送消息格式（httpbody）：

//签署合同的推送消息格式如下：

```
{
  "action": "signContract",    //针对的事件，signContract 是签署合同
  "params": {
    "contractId": "1093274328532857",    //合同编号
    "account": "mlimd@163.com",    //签署者账号
    "signerStatus": 2,    //签署状态：（1：正在进行中；2：已完成；3：已拒绝并取消；）
    "errMsg": "success"    //签署成功为 success，签署失败为相应的错误信息
  }
}
```

//异步申请证书的推送消息格式如下

```
{
  "action": "certApply",      //针对的事件, certApply 是申请证书
  "params": {
    "certType": "ZJCA",
    "cert": "ZJCA-31-20171201174841514-03517",      //证书编号
    "message": "",
    "account": "ent1512121718841",      //申请证书的账号
    "taskId": "151212172101000002",      //异步申请队列的序号
    "status": "5"
  }
}

//创建模版的推送消息格式如下
{
  "action": "createTemplate", //针对的事件, createTemplate 是创建模版
  "params": {
    "tid": "2489738745798734",      //开发者模版编号
    "account": "13897423874",      //证书编号
    "sid": "47283u78347846"      //业务流水号
  }
}

//编辑模版的推送消息格式如下
{
  "action": "modifyTemplate", //针对的事件, modifyTemplate 是创建模版
  "params": {
    "tid": "2489738745798734",      //开发者模版编号
    "account": "13897423874",      //证书编号
    "sid": "47283u78347846"      //业务流水号
  }
}

//支付接口的推送消息格式如下
{
  "action": "onlinePay",
  "params": {
    "orderId": "2130985035547807747",
    "paymentMode": "WECHATPAY",
    "contractId": "2371924",
    "completeDateTime": "2018-11-22 14:28:19 ",
    "orderAmount": "0.4",
    "payStatus": "SUCCESS"
  }
}
```

```
}
```

### 3、推送消息的签名验签机制：

推送消息的 headers 中会含有本次推送的签名信息，用于校验。

```
"headers": {  
  "rtick": "15004551687129438",  
  "sign": "d517a52a9910f164ec42bf8c1a7803b2"  
}
```

**rtick** ----时间戳，unix 时间戳+4 位随机数

**sign** ----就是 sign,  $\text{sign} = \text{md5}(\text{md5}(\text{httpbody}) + \text{rtick} + \text{accessKey})$

## 8、返回消息体（response）加密说明

有些开发者对数据的保密性有更高的要求，因此上上签在返回消息时也进行加密加签。

### 使用方法

开发者若需要返回的消息加密，需要在 request 的 Headers 添加属性

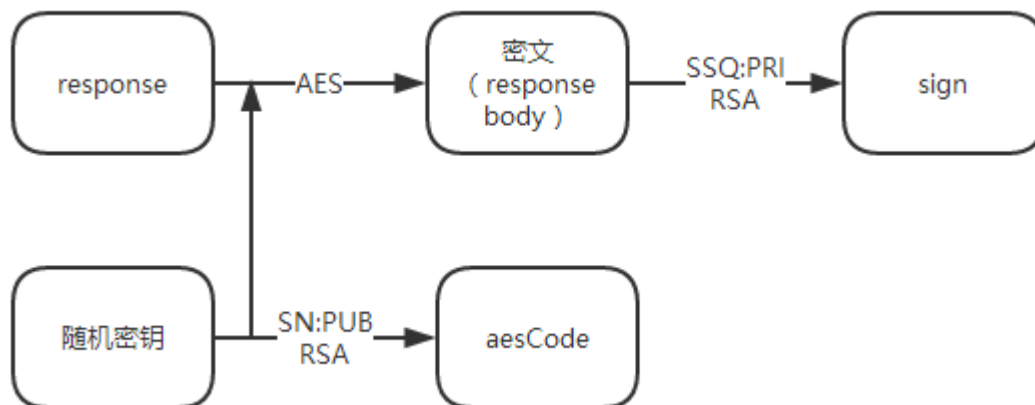
“encryptResponse”，仅在“encryptResponse”值为 1 时表示需要加密返回。目前支持加密的 response body 数据类型是 byte[]、String、JSONObject 这三种。

### 加密原理

在此加密加签过程中，需要用到：

- 上上签的私钥（SSQ:PRI）——上上签生成，由上上签保管
- 上上签的公钥（SSQ:PUB）——上上签生成，提供给开发者，做验签用
- 随机密钥——即 accessKey，上上签生成（可登录开放平台获取），用作 AES 加密，并提供给开发者用作 AES 解密
- 开发者的公钥（SN:PUB）——开发者生成，提供给上上签，做验签用
- 开发者的私钥（SN:PRI）——开发者生成，由开发者保管

#### 1、加密原理

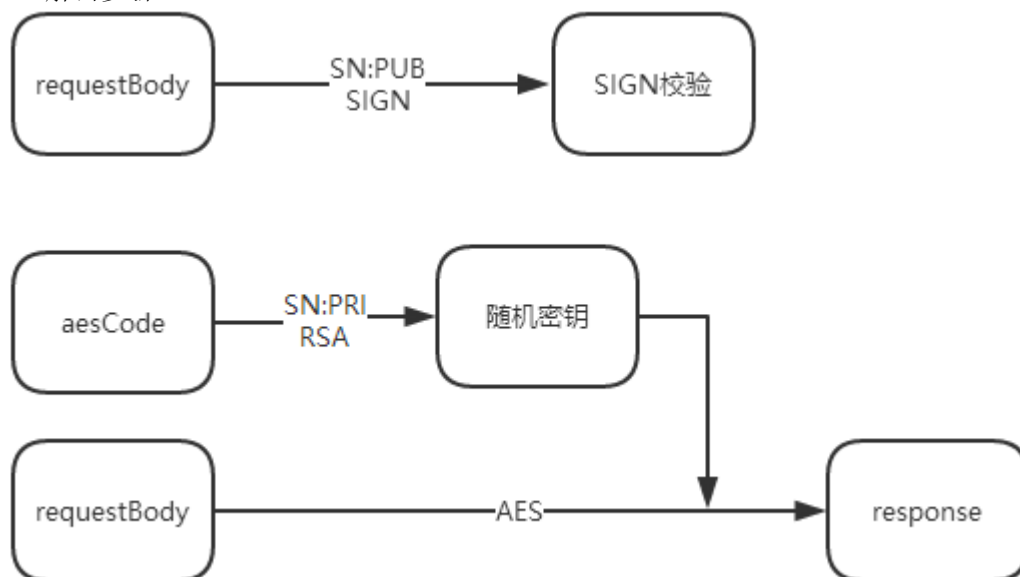


使用随机密钥对接口的 response 进行 AES 加密，然后将密文放在 response body 中，并将 body 使用上上签私钥进行 RSA 加签得到 sign 用于验签，同时将随机密钥使用开发者的公



钥 RSA 加密得到 aesCode，aesCode 与 sign 都放在 response header 中。

## 2、解密步骤



开发者接收到上上签的 response 之后，使用上上签公钥对 header 中的 sign 进行验签，验签通过，则使用开发者私钥对 aesCode 进行解密得到随机密钥，再用随机密钥对 body 中的密文进行解密得到真正的接口 response。

## 3、返回加密报文示例

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
aesCode:
fT27wCCS6GnDlAgiCeHiUPTDR0XSJ/uINdNMq+DdLFkz4YORObEK7wrsu8xNf0ECYM+WZ
3zhnEMd/IDQj0gcot7E+0b5Kjo7mIdkMNB9EFFF1OsBEbfjWlyFt2zJn738brkNDE25Kd
jeAvMnDS2O51tDm0XJPhDlH8QqaA8aSvg=
sign:
ZMLGaDhvNyerdCXHm/28SvvGF3N46VTchSCAFISa5Y+cZUFDLKQkWay5Y/GhIAMS9Dh/z
98NB5HMFdLPFTRNim46MKVZUNsEAHuZNoowSyX82R/SmSq0RkEn4bTQRru3nniKyMBkKU
/pXvGpGzn0a/g0KmRRrMrQ+2y3xdfnybU=
Expires: -1
Pragma: no-cache
CacheControl: no-cache
Content-Length: 88
Date: Thu, 02 Nov 2017 02:48:33 GMT
Content-Type: application/json; charset=UTF-8

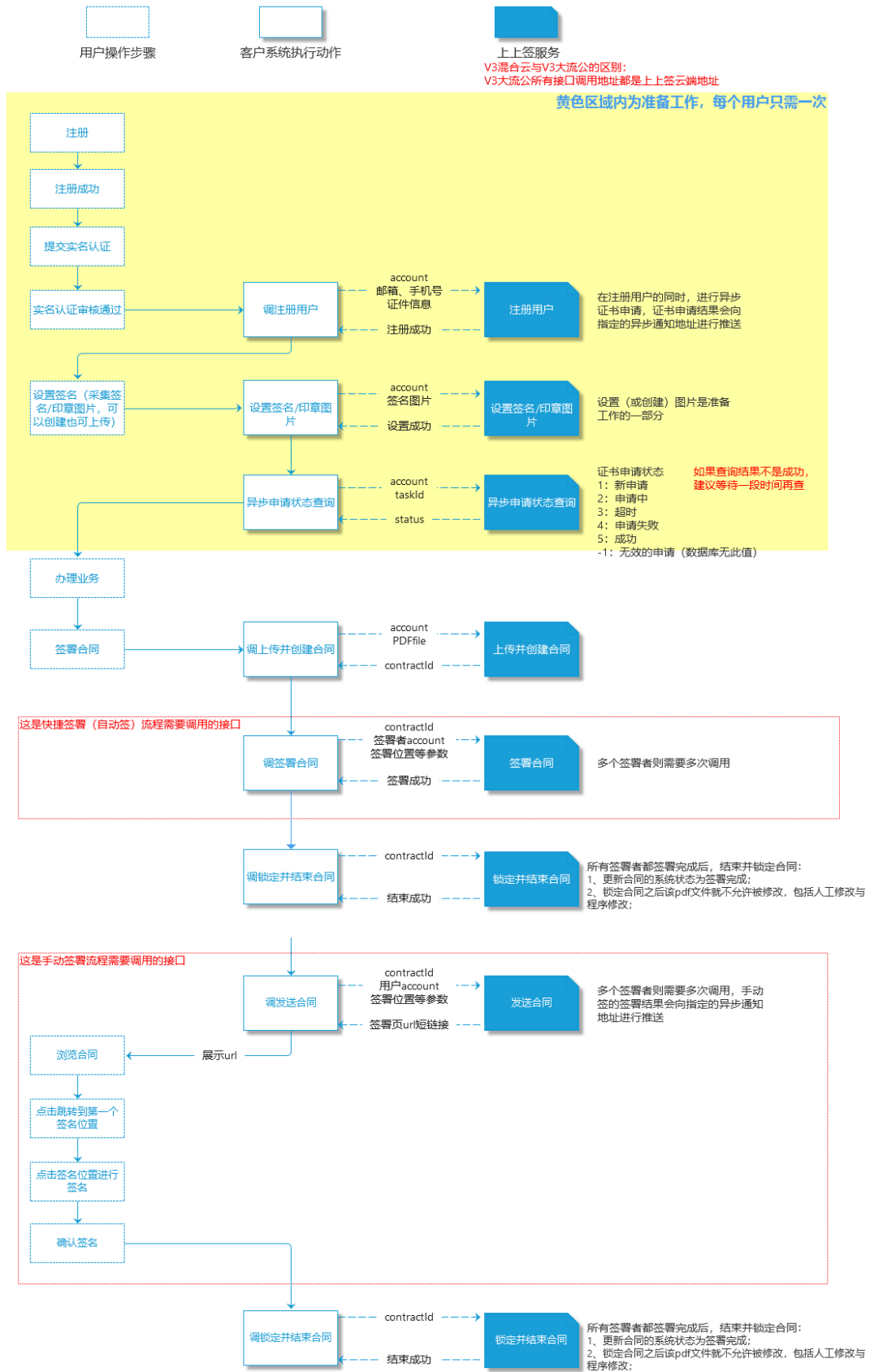
RESPONSE BODY:
FmCKgiFAuiasYrRiQolCtR3NbZZJGJ020esImdmg/TyzcGL7ftqR8GoOkRSSNUYMq6FH6
z1OVTb8C0uPqe1f0Q==
```

### 三、API 调用流程

根据不同的用户接入要求，上上签 OpenAPI 提供两种签署模式：快捷签署（又叫自动签，由系统自动签署，用户无感知）、手动签署（体现签署操作的仪式感）。“快捷签署”与“手动签署”可以混搭，比如同一份合同中，甲用“快捷签署”，乙用“手动签署”。总体流程如下图所示：

## V3混合云接口标准电子签约流程

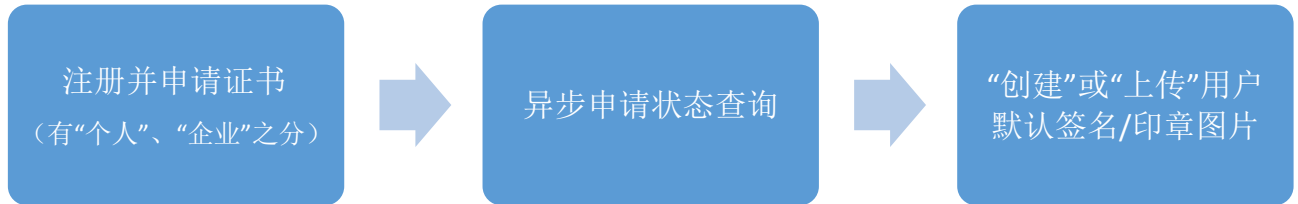
——既V3大流公接口标准电子签约流程标准



## 1、注册用户

使用上上签 OpenAPI 进行合同的发送、签署等服务的用户，必须要先注册一个上上签的用户，此后的所有操作都使用该用户 **account**（用户帐号）进行。

### 接口调用流程



相关接口如下：

|  |   |
|--|---|
| 注册个人用户并申请证书/user/reg                       | 包含：“注册用户、设置证件信息、异步申请证书”3个功能，证书是否申请成功需要通过“异步申请状态查询”来确认 |
| 注册企业用户并申请证书/user/reg                       | 包含：“注册用户、设置证件信息、异步申请证书”3个功能，证书是否申请成功需要通过“异步申请状态查询”来确认 |
| 创建用户签名或印章图片<br>/signatureImage/user/create | 二选一，要么创建要么上传，创建与上传会相互覆盖默认图片                           |
| 上传用户签名或印章图片<br>/signatureImage/user/upload | 二选一，要么创建要么上传，创建与上传会相互覆盖默认图片                           |
| 异步申请状态查询<br>/user/async/applyCert/status/  | 在申请证书之后、签署之前请先使用此接口确认证书是否申请成功，避免在签署时出错                |

## 2、快捷签署（自动签）

用户无感知的快捷签署，又称为“自动签”、“静默签”等。

### 接口调用流程



相关接口如下：

|         |                             |  |
|---------|-----------------------------|--|
| 上传并创建合同 | /storage/contract/upload    | 上传 PDF 文件并创建合同                                   |
| 签署合同    | /storage/contract/sign/cert | 自动签署   |
| 锁定并结束合同 | /storage/contract/lock/     | 当合同的所有签署人都签署完毕后，调用此接口结束合同的整个签署过程，结束后将不允许进行任何签署操作 |

### 3、手动签署

用户有感知的签署，有手写签名、校验短信验证码等交互动作，体现签署的仪式感。

#### 接口调用流程



相关接口如下：

|         |                          |   |
|---------|--------------------------|---|
| 上传并创建合同 | /storage/contract/upload | 上传 PDF 文件并创建合同  |
| 发送合同    | /contract/send           | 为某个手动签用户生成签署链接，可将此链接发送给用户，打开此链接进入手动签页面，在页面上查看合同并完成签署，体现签署的仪式感 |
| 锁定并结束合同 | /storage/contract/lock   | 当合同的所有签署人都签署完毕后，调用此接口结束合同的整个签署过程，结束后将不允许进行任何签署操作。             |

### 4、PDF 签名图片尺寸、位置的计算方法（必读）

尺寸单位换算（来源百度）：1 厘米≈28.35 磅

常用规格印章尺寸：

3.5 厘米≈99 磅

4 厘米≈113 磅

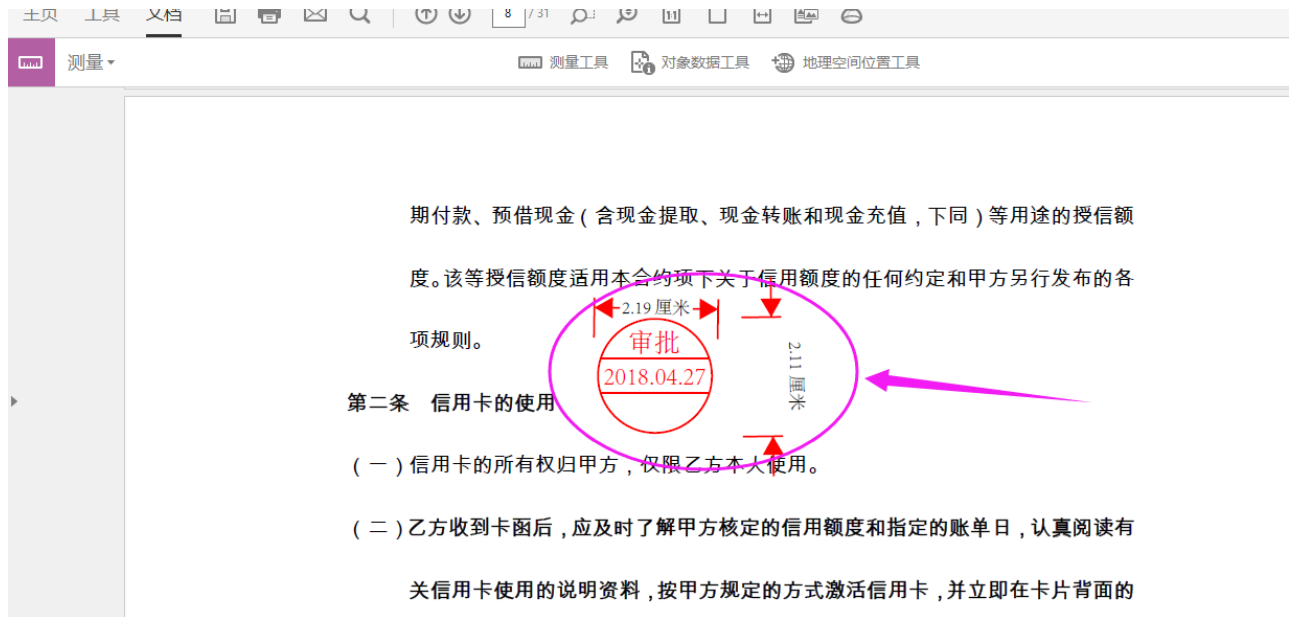
4.2 厘米≈119 磅

5 厘米≈142 磅

5.8 厘米≈164 磅

A4 纸宽 20.9 厘米≈593 磅，A4 纸高 29.7 厘米≈842 磅

## 4.1、图片尺寸的计算方法



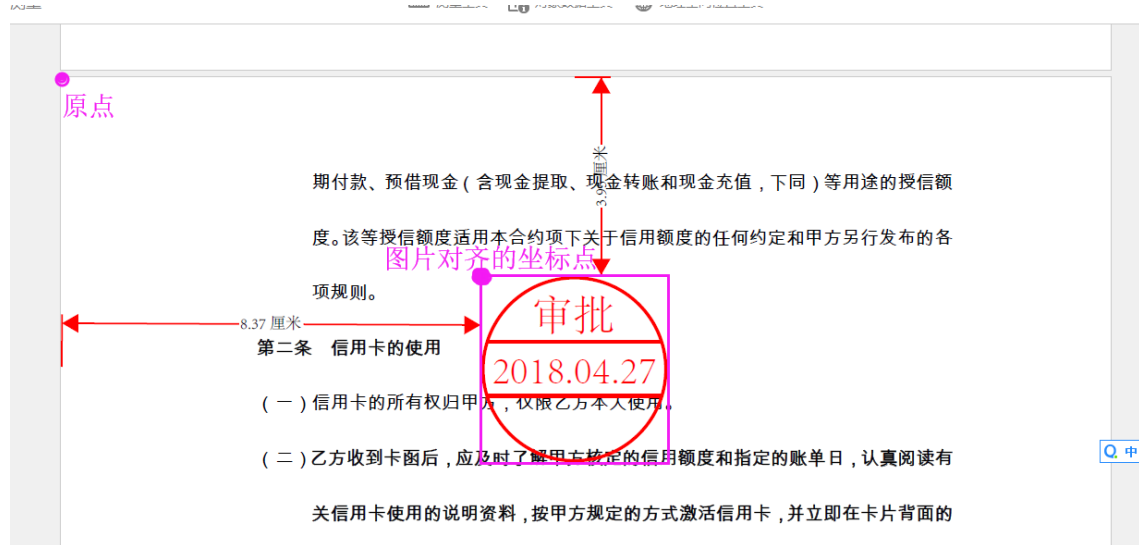
如上图所示，一个图片在一份 PDF 想要显示的大小是有一个具体尺寸的，比如常见的圆形公章有 3.5 厘米×3.5 厘米、4.0 厘米×4.0 厘米等规格的尺寸。而计算机在处理图片时，需要用到的图片尺寸单位为磅值，换算比例（来源百度）为 1 厘米≈28.35 磅。

如果想要一个 3.5 厘米×3.5 厘米公章图片在 PDF 上体现为真实大小，那么在接口参数中就应该这样传（取四舍五入后的整数）：3.5 厘米×28.35 磅/厘米=99.225 磅≈99 磅：

```
"signatureImageWidth": "99",  
"signatureImageHeight": "99"
```

## 4.2、位置坐标的计算方法

在上上签的坐标体系中，以页面的左上角为坐标原点、以签名/印章图片的左上角来计算图片所在的位置坐标，位置坐标为一个带小数点的比例数值，因此按像素单位计算、按磅值单位计算、或者按厘米单位计算都可以，以下用厘米单位举例：



如上图, 印章图片左上角距离左侧页边距为 8.37 厘米、距离上侧页边距 3.95 厘米, 页面为 A4 纸大小, 宽 20.9 厘米, 高 29.7 厘米, 那么这个图片在页面上坐标的计算方法为:

$$X = 8.37 \text{ 厘米} \div 20.9 \text{ 厘米} = 0.400478$$

$$Y = 3.95 \text{ 厘米} \div 29.7 \text{ 厘米} = 0.132997$$

在参数中应该这样传坐标:

```
"x": "0.400478",  
"y": "0.132997",
```

## 四、标配 API 功能列表

### 1、用户服务

#### 1.1、注册个人用户并申请证书

说明: 创建一个上上签个人用户, 并提交用户的证件信息, 为该用户申请数字证书。后续有关该用户的所有操作都使用该用户的 account。

由于证书是异步申请, 不会立即返回证书申请结果, 所以在签署之前, 需要确认一次该用户的证书是否申请成功, 调用接口“1.11、异步申请状态查询”确认。

Request Path: [sdk-host]/user/reg/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数（REQUEST BODY）：

| 参数名        | 说明       | 是否必填 | 最大长度 |                          |        |      |      |  |
|------------|----------|------|------|--------------------------|--------|------|------|--|
| account    | 用户帐号     | Y    |      | 用户的唯一标识，可以是邮箱、手机号、证件号等不限 |        |      |      |  |
| name       | 用户名称     | Y    | 100  | 必须和证件上登记的姓名一致            |        |      |      |  |
| userType   | 用户类型     | Y    | 1    | 1 表示个人                   |        |      |      |  |
| mail       | 用户邮箱     | N    | 50   |                          |        |      |      |  |
| mobile     | 用户手机号    | N    | 20   |                          |        |      |      |  |
| credential | 用户证件信息对象 | Y    |      | 用户证件信息对象，json 格式么具体参数如下： |        |      |      |  |
|            |          |      |      | 参数                       | 说明     | 是否必填 | 最大长度 |  |
|            |          |      |      | identity                 | 用户证件号  | Y    | 30   | 必须和证件上登记的号码一致  |
|            |          |      |      | identityType             | 用户证件类型 | N    | 2    | 默认为“0”，<br>0-居民身份证<br>1-护照<br>B-港澳居民往来内地通行证<br>C-台湾居民来往大陆通行证<br>E-户口簿<br>F-临时居民身份证 |
|            |          |      |      | contactMobile            | 联系手机   | N    | 20   |  |
|            |          |      |      | contactMail              | 联系邮箱   | N    | 50   |  |
|            |          |      |      | province                 | 省份     | N    | 10   |  |
|            |          |      |      | city                     | 城市     | N    | 50   |  |
|            |          |      |      |                          |        |      |      |  |



|           |        |   |   |         |    |   |     |  |
|-----------|--------|---|---|---------|----|---|-----|--|
|           |        |   |   | address | 地址 | N | 100 |  |
| applyCert | 是否申请证书 | Y | 1 | 申请填写为 1 |    |   |     |  |

示例:

```
{
  "account": "test@bestsign.cn",
  "mail": "test@bestsign.cn",
  "mobile": "13800001234",
  "name": "张三",
  "userType": "2",
  "credential": {
    "identity": "100123199001010011",
    "identityType": "0",
    "contactMail": "test@bestsign.cn",
    "contactMobile": "13800001234",
    "province": "浙江省",
    "city": "杭州市",
    "address": "xx 路 xx 号"
  },
  "applyCert": "1"
}
```

响应 (RESPONSE) :

| 参数名    | 说明  |
|--------|---|
| taskId | 异步申请证书队列中的任务编号, 在 24 小时内可用于查询异步申请状态, taskId 过 24 小时后就失效 |

正常示例:

```
{
  "errno": 0,
  "cost": 533,
  "data": {
    "taskId": "170217170217942"
  },
  "errmsg": ""
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

}

## 1.2、注册企业用户并申请证书

说明：创建一个上上签企业用户，并提交用户的证件信息，为该用户申请数字证书。后续有关该用户的所有操作都使用该用户的 **account**。

由于证书是异步申请，不会立即返回证书申请结果，所以在签署之前，需要确认一次该用户的证书是否申请成功，调用接口“1.11、异步申请状态查询”确认。

Request Path: **[sdk-host]**/user/reg/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数 (REQUEST BODY)：

| 参数名      | 说明   | 是否必填 | 最大长度 |   |
|----------|------|------|------|---|
| account  | 用户帐号 | Y    |      | 用户的唯一标识，可以是邮箱、手机号、证件号等不限                        |
| name     | 企业名称 | Y    | 100  | 必须和企业证件上登记的名称一致<br>如个体工商户在营业执照上无企业名称的，请填“经营者”名字 |
| userType | 用户类型 | Y    | 1    | 2 表示企业  |
| mail     | 用户邮箱 | N    | 50   |   |
| mobile   | 用户手机 | N    | 20   |   |

|            |                      |   |  |  |   |
|------------|----------------------|---|--|--|---|
|            | 号                    |   |  |  |   |
| credential | 企业<br>证件<br>信息<br>对象 | Y |  | 企业证件信息对象，json，<br>三证合一的统一社会信用代码这样传：<br>regcode = "统一社会信用代码"<br>orgcode = "统一社会信用代码"<br>taxcode = "统一社会信用代码"<br>老三证这样传：<br>regcode = "工商注册号"<br>orgcode = "组织机构代码证"<br>taxcode = "税务登记证"<br>个体户这样传：<br>regcode = "工商注册号"<br>orgcode = ""<br>taxcode = ""<br>具体参数如下： |   |
|            |                      |   |  | 参数   | 说明<br>是否必填<br>最大长度  |
|            |                      |   |  | regCode  | 工商<br>注册<br>号<br>Y<br>30<br>企业用户设置。如果是三证合一，regCode/taxCode/orgCode 三个填一样的值<br>regCode 和 orgCode 必须输入一个，否则异常返回：240006: regCode/orgCode must be have one              |
|            |                      |   |  | orgCode  | 组<br>织<br>机<br>构<br>代<br>码<br>Y<br>30<br>企业用户设置。如果是三证合一，regCode/taxCode/orgCode 三个填一样的值<br>regCode 和 orgCode 必须输入一个，否则异常返回：240006: regCode/orgCode must be have one |
|            |                      |   |  | taxCode  | 税<br>务<br>登<br>记<br>证<br>Y<br>30<br>企业用户设置。如果是三证合一，regCode/taxCode/orgCode 三个填一样的值  |

|  |  |  |                                |                           |   |    |   |
|--|--|--|--------------------------------|---------------------------|---|----|---|
|  |  |  |                                | 号                         |   |    |   |
|  |  |  | <b>legalPerson</b>             | 法定<br>代表<br>人姓<br>名       | Y | 50 | 法定代表人姓名或经办人姓名   |
|  |  |  | <b>legalPersonIdentity</b>     | 法定<br>代表<br>人证<br>件号      | Y | 30 | 法定代表人证件号或经办人证<br>件号   |
|  |  |  | <b>legalPersonIdentityType</b> | 法定<br>代表<br>人证<br>件类<br>型 | Y | 2  | 法定代表人证件类型或经办人<br>证件类型，与<br>“ <b>legalPersonIdentity</b> ”要匹<br>配，默认为“0”。<br>0-居民身份证<br>1-护照<br>B-港澳居民往来内地通行证<br>C-台湾居民来往大陆通行证<br>E-户口簿<br>F-临时居民身份证 |
|  |  |  | <b>legalPersonMobile</b>       | 法定<br>代表<br>人手<br>机号      | N | 20 | 法定代表人手机号或经办人手<br>机号   |
|  |  |  | <b>contactMobile</b>           | 联<br>系<br>手<br>机          | Y | 20 | 联系手机必填，为 CA 年检抽查<br>时联系使用   |

|           |        |   |   |             |      |   |     |                 |
|-----------|--------|---|---|-------------|------|---|-----|-----------------|
|           |        |   |   | contactMail | 联系邮箱 | N | 50  | credential 对象属性 |
|           |        |   |   | province    | 省份   | N | 10  | credential 对象属性 |
|           |        |   |   | city        | 城市   | N | 50  | credential 对象属性 |
|           |        |   |   | address     | 地址   | N | 100 | credential 对象属性 |
| applyCert | 是否申请证书 | Y | 1 | 申请填写为 1     |      |   |     |                 |

示例:

```
{
  "account": "test@bestsign.cn",
  "mail": "test@bestsign.cn",
  "mobile": "13800001234",
  "name": "浙江 xx 公司",
  "userType": "2",
  "credential": {
    "regCode": "5566789",
    "taxCode": "5566789",
    "orgCode": "5566789",
    "contactMail": "test@bestsign.cn",
    "contactMobile": "13800001234",
    "legalPerson": "张三",
    "legalPersonIdentity": "100123199001010011",
    "legalPersonIdentityType": "0",
    "legalPersonMobile": "17712341234",
    "province": "浙江省",
    "city": "杭州市",
    "address": "xx 路 xx 号"
  },
  "applyCert": "1"
}
```

响应 (RESPONSE) :

| 参数名    | 说明  |
|--------|---|
| taskId | 异步申请证书队列中的任务编号, 在 24 小时内可用于查询异步申请状态, taskId 过 24 小时后就失效 |

正常示例:

```
{
  "errno": 0,
  "cost": 533,
  "data": {
    "taskId": "170217170217942"
  },
  "errmsg": ""
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

### 1.3、查询证书编号

说明: 查询某一用户的数字证书编号。

Request Path: [sdk-host]/user/getCert/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY) :

| 参数名     | 说明   | 是否必填 | 最大长度 |                   |
|---------|------|------|------|-------------------|
| account | 用户帐号 | Y    |      | 查询哪个用户的证书就填该用户的帐号 |

示例:

```
{
  "account": "1674224491900698632"
}
```

响应 (RESPONSE) :

RESPONSE DATA 参数如下:

| 参数名     | 说明                         |
|---------|----------------------------|
| account | 用户帐号                       |
| cert    | 证书编号，为兼容以前的版本所留的参数，可以不用此参数 |
| certId  | 证书编号，与 cert 相同             |

正常示例：

```
{
  "errno": 0,
  "cost": 533,
  "errmsg": "",
  "data": {
    "account": "1674224491900698632",
    "cert": "CFCA-33-20171010142326279-12345",
    "certId": "CFCA-33-20171010142326279-12345"
  }
}
```

异常示例：

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 1.4、查询个人用户证件信息

说明：获取个人用户的证件信息，用来确认当初设置的证件信息是否正确。

Request Path: [sdk-host]/user/getPersonalCredential/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名     | 说明   | 是否必填 | 最大长度 |                     |
|---------|------|------|------|---------------------|
| account | 用户帐号 | Y    |      | 查询哪个用户的证件信息就填该用户的帐号 |

示例：

```
{
  "account": "1674224491900698632"
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

| 参数名           | 说明                   |
|---------------|----------------------|
| account       | 用户帐号                 |
| identity      | 证件号                  |
| identityType  | 证件类型。默认为“0”，“0”表示身份证 |
| name          | 姓名                   |
| contactMail   | 联系邮箱                 |
| contactMobile | 联系手机                 |
| province      | 省份                   |
| city          | 城市                   |
| address       | 地址                   |

正常示例：

```
{
  "errno": 0,
  "cost": 533,
  "errmsg": "",
  "data": {
    "account": "1238912387313",
    "identity": "100123199001010011",
    "identityType": "0",
    "contactMail": "test@bestsign.cn",
    "contactMobile": "13800001234",
    "name": "张三",
    "province": "浙江省",
    "city": "杭州市",
    "address": "xx 路 xx 号"
  }
}
```

异常示例：

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```



## 1.5、查询企业用户证件信息

说明：获取企业用户的证件信息，用来确认当初设置的证件信息是否正确。

Request Path: [sdk-host]/user/getEnterpriseCredential/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名     | 说明   | 是否必填 | 最大长度 |                     |
|---------|------|------|------|---------------------|
| account | 用户帐号 | Y    |      | 查询哪个用户的证件信息就填该用户的帐号 |

示例：

```
{
  "account": "1674224491900698632"
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

| 参数名                     | 说明                        |
|-------------------------|---------------------------|
| account                 | 用户帐号                      |
| regCode                 | 工商注册号                     |
| taxCode                 | 税务登记号                     |
| orgCode                 | 组织机构代码                    |
| name                    | 姓名/名称                     |
| legalPerson             | 法定代表人                     |
| legalPersonIdentity     | 法定代表人证件号                  |
| legalPersonIdentityType | 法定代表人证件类型。默认为“0”，“0”表示身份证 |
| legalPersonMobile       | 法定代表人手机号                  |
| contactMail             | 联系邮箱                      |
| contactMobile           | 联系手机                      |
| province                | 省份                        |
| city                    | 城市                        |
| address                 | 地址                        |

正常示例：

```
{
  "errno": 0,
  "cost": 533,
}
```

```
"errmsg": "",
"data": {
  "regCode": "556789",
  "taxCode": "556789",
  "orgCode": "556789",
  "legalPerson": "张三",
  "legalPersonIdentity": "100123199001010011",
  "legalPersonIdentityType": "0",
  "legalPersonMobile": "17712341234",
  "contactMail": "test@bestsign.cn",
  "contactMobile": "13800001234",
  "name": "浙江 xx 公司",
  "province": "浙江省",
  "city": "杭州市",
  "address": "xx 路 xx 号"
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 1.11、异步申请状态查询

说明:

异步申请证书请求发送成功后, 根据返回的 `taskId` 调用本接口查询申请结果。

Request Path: `[sdk-host]/user/async/applyCert/status/`

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY):

| 参数名     | 说明   | 是否必填 | 最大长度 |  |
|---------|------|------|------|--|
| account | 用户帐号 | Y    |      | 需要查询证书申请状态的用户账号                                    |
| taskId  | 任务编号 | Y    |      | 由异步申请数字证书接口返回得到, <code>taskId</code> 在 24 小时内可用于查询 |

示例:

```
{
  "account": "mlimd@163.com",
  "taskId": "170217170217942"
}
```

响应 (RESPONSE):

RESPONSE DATA 参数如下:

| 参数名     | 说明   |
|---------|--|
| status  | 证书申请状态<br>1: 新申请<br>2: 申请中<br>3: 超时<br>4: 申请失败<br>5: 成功<br>-1: 无效的申請 (数据库无此值)<br>0: taskId 不存在或已过期 |
| message | 状态描述   |

正常示例:

```
{
  "errno": 0,
  "cost": 533,
  "data": {
    "message": "",
    "status": "5"
  },
  "errmsg": ""
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 1.12、获取证书详细信息

说明: 根据某一用户的证书编号, 获取证书的详细信息, 此详细信息与查看 pdf 签名证书详情时看到的详细信息一致。

Request Path: [sdk-host]/user/cert/info/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY):

| 参数名     | 说明   | 是否必填 | 最大长度 |  |
|---------|------|------|------|--|
| account | 用户帐号 | Y    |      | 查询哪个用户的证书就填该用户的帐号                                      |
| certId  | 证书编号 | Y    | 10   | 通过“1.3、查询证书编号”接口查询到的 certId, 如 CFCA-100123199001010011 |

示例:

```
{
  "account": "1674224491900698632",
  "certId": "CFCA-100123199001010011"
}
```

响应 (RESPONSE):

RESPONSE DATA 参数如下:

| 参数名           | 说明                                |
|---------------|-----------------------------------|
| certId        | 证书编号, 在上上签系统使用                    |
| serialNumber  | CA 证书序列号, 与查看 pdf 签名证书详情时看到的序列号一致 |
| subjectDN     | 证书主题, 即该证书代表的个人或企业的名称等信息          |
| issuerDN      | 颁发机构                              |
| startTime     | 有效期开始时间                           |
| stopTime      | 有效期截止时间                           |
| revokedTime   | 吊销日期, 正常使用中的证书无此项内容               |
| revokedReason | 吊销原因, 正常使用中的证书无此项内容               |
| status        | 状态码 1, -2                         |
| statusMsg     | 状态描述, 1: 激活, -2: 吊销               |

正常示例:

```
{
  "errno": 0,
  "cost": 88,
  "data": {
    "serialNumber": "68811245717",
  }
}
```

```
"statusMsg": "激活",
"issuerDN": "",
"revokedTime": null,
"startTime": "2017-08-09 14:27:51",
"stopTime": "2019-08-09 14:27:51",
"certId": "CFCA-33-20170809142751732-81453",
"revokedReason": null,
"subjectDN": "",
"status": 1
},
"errmsg": ""
}
```

异常示例:

```
{
  "errno": 241310,
  "cost": 88,
  "data": null,
  "errmsg": "using cert is not same with query cert"
}
```

## 2、签名/印章图片服务

### 2.1、生成用户签名/印章图片

说明：调用此接口可根据用户注册的名称、也可以根据指定的文本，系统生成用户签名图片或者印章图片（个人用户生成签名图片，企业用户生成印章图片），并且设置为用户默认签名。

Request Path: [sdk-host]/signatureImage/user/create/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY) :

| 参数名     | 说明            | 是否必填 |   |
|---------|---------------|------|---|
| account | 用户帐号          | Y    | 创建哪个用户的签名/印章图片就填该用户的帐号。企业印章横向文默认为“电子签约专用章”。 |
| text    | 签名/印章图片上生成的文本 | N    | 可不填，系统默认使用证件信息上的名称                          |

|           |                   |   |   |
|-----------|-------------------|---|---|
| fontName  | 字体名称（仅针对个人类型账号有效） | N | 目前枚举值如下：<br>SimHei 黑体<br>SimSun 宋体<br>SimKai 楷体   |
| fontSize  | 字号（仅针对个人类型账号有效）   | N | 12~120，默认 30，此参数影响签名字体的清晰度和签名图片大小，字号越高，字体显示越大，清晰度越高。<br>注：过小的字号在手动签的预览页面上显示会与实际大小有差别，但签署之后的 PDF 上的大小正常。 |
| fontColor | 字体颜色（仅针对个人类型账号有效） | N | 指定字体的颜色，支持：<br>red（红），black（黑），blue（蓝），purple（紫），grey（灰），brown（棕），tan(褐色), cyan(青色)                     |

示例：

```
{  
  "account": "13456833929",  
  "fontName": "SimKai",  
  "fontSize": "36",  
  "fontColor": "red"  
}
```

响应（RESPONSE）：

正常示例：

```
{  
  "errno": 0,  
  "cost": 533,  
  "data": { },  
  "errmsg": ""  
}
```

异常示例：

```
{  
  "errno": 241207,  
  "cost": 533,  
  "data": null,  
  "errmsg": "user not exist"  
}
```

## 2.2、上传用户签名/印章图片

说明：

- a) 为某一用户上传用户签名图片、或印章图片。
- b) 印章图片的尺寸请按照实际尺寸（PS 编辑图片时以厘米为单位的尺寸）设置，比如印章是“5 厘米×3.5 厘米”大小的，印章图片就设置为“5 厘米×3.5 厘米”，印章图片的 dpi 建议使用 96 或 120，dpi 太低签完后的清晰度不够，dpi 太高则图片体积太大。
- c) 上上签不对上传的图片做图像处理（如背景透明化等），请在上传前将印章图片处理好。
- d) 使用自动签的用户可以略过此接口，但使用手动签的企业用户必须先创建或者上传签名图片（个人用户在手动签时还可以选择手绘签名）。

Request Path: [sdk-host]/signatureImage/user/upload/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY):

| 参数名       | 说明        | 是否必填 | 最大长度 |   |
|-----------|-----------|------|------|---|
| account   | 用户帐号      | Y    |      | 上传哪个用户的签名/印章图片就填该用户的帐号  |
| imageData | 图片文件内容    | Y    |      | 图片经 Base64 编码后的字符串  |
| imageName | 签名/印章图片名称 | N    | 50   | 传空或 default 表示更新默认的签名/印章图片。<br><br>企业用户如果有多个印章，可以指定印章名称，签署时用指定的印章名称 |

示例:

```
{
  "account": "abc@163.com",
  "imageData": "....."
}
```

响应 (RESPONSE):

正常示例:

```
{
  "errno": 0,
  "cost": 533,
}
```

```
"data": { },  
"errmsg": ""  
}
```

异常示例:

```
{  
  "errno": 241207,  
  "cost": 533,  
  "data": null,  
  "errmsg": "user not exist"  
}
```

## 2.3、下载用户签名/印章图片

说明: 下载用户的默认签名图片、印章图片数据, 便于核对。

Request Path: [sdk-host]/signatureImage/user/download/

Method: GET

其他 url 参数: account, imageName

需要参与签名的其他 URL 参数:

| 参数名       | 说明        | 是否必填 | 最大长度 |   |
|-----------|-----------|------|------|---|
| account   | 用户帐号      | Y    |      | 下载哪个用户的签名/印章图片就填该用户的帐号                                    |
| imageName | 签名/印章图片名称 | Y    | 50   | 空和 default 都表示默认的签名/印章图片。<br>企业用户如需下载自定义印章则此处填写上传时指定的印章名称 |

请求参数 (REQUEST BODY): 无

示例:

GET

[host]/signatureImage/user/download/?account=72dun32&developerId=d445f5432&imageName=&rtick=17364899234&sign=a2334df21323&signType=rsa

响应 (RESPONSE):

正常返回的 response 内容是签名图片的二进制数据 (文件流), 获取后自行存储。

异常示例:

```
{  
  "errno": 240012,  
  "cost": 533,  
  "data": null,  
}
```



```
"errmsg": "check sign wrong "
}
```

### 3、文件存储服务

#### 3.1、上传合同文件

说明：

- a) 此接口是将文件保存到自定义的存储中，目前仅支持上传 PDF 文件。
- b) 上传成功后得到一个文件编号，后续流程可以直接使用此文件编号发起合同。

Request Path: [sdk-host]/storage/upload/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名     | 说明                 | 是否必填 | 最大长度 |   |
|---------|--------------------|------|------|---|
| account | 用户帐号               | Y    |      | 必须要指定一个用户帐号作为操作者  |
| fdata   | 文件数据，<br>base64 编码 | Y    |      | 例如：<br><pre>FileInputStream file = new FileInputStream("d: \\test\\接口系统.pdf"); byte[] bdata = IOUtils.toByteArray(file); String fdata = Base64.encodeBase64String(bdata);</pre> |
| fmd5    | 文件 md5 值           | Y    |      | 例如：<br><pre>FileInputStream file = new FileInputStream("d: \\test\\接口系统.pdf"); byte[] bdata = IOUtils.toByteArray(file); String fmd5 = DigestUtils.md5Hex(bdata);</pre>         |
| ftype   | 文件类型               | Y    | 10   | 如 PDF 等   |
| fname   | 文件名                | Y    | 50   | 文件名必须带上后缀名，例如“XXXX.pdf”、“XXXX.docx”   |
| fpages  | 文件页数               | Y    | 3    |   |

示例：

```
{
  "account": "u-1112343242",
```

```
"fmd5": "3424324112343242",
"ftype": "PDF",
"fname": "contract.PDF",
"fpages": "7",
"fdata": "....."
}
```

响应 (RESPONSE) :

RESPONSE DATA 参数如下:

| 参数名 | 说明   |
|-----|------|
| fid | 文件编号 |

正常示例:

```
{
  "errno": 0,
  "cost": 533,
  "errmsg": "",
  "data": {
    "fid": "f1343423222"
  }
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

### 3.2、PDF文件添加元素

说明: 为 PDF 文件添加元素, 目前支持的元素类型为 text、image。添加元素后会得到一个新的文件编号, 此文件编号可以用于后续流程发起合同。

Request Path: [sdk-host]/storage/addPDFElements/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY) :

| 参数名     | 说明   | 是否必填 | 最大长度 |               |
|---------|------|------|------|---------------|
| account | 用户帐号 | Y    |      | 必须要指定一个用户帐号作为 |

|          |          |   |    |                 |
|----------|----------|---|----|-----------------|
|          |          |   |    | 操作者             |
| fid      | 源文件编号    | Y |    | 源文件必须是 PDF 文件格式 |
| elements | 要添加的元素集合 | Y | 10 | json array 格式   |

示例:

```
{
  "account": "u-1112343242",
  "fid": "f1343423222",
  "elements": [
    {
      "pageNum": "新添加元素所在的页码",
      "x": "新添加元素 x 坐标, 用百分比表示, 取值 0.0~1.0",
      "y": "新添加元素 y 坐标, 用百分比表示, 取值 0.0~1.0",
      "type": "新添加元素的类型. 目前支持: text, image 两种",
      "value": "新添加元素的内容. 如果是 text 类型, 为文本;如果是 image 类型, 为 base64 编码后的图片内容",
      "fontSize": "如果新添加元素是 text 类型, 可以用来指定新添加元素的字体大小. 默认 14"
    }
  ]
}
```

响应 (RESPONSE):

RESPONSE DATA 参数如下:

| 参数名 | 说明       |
|-----|----------|
| fid | 产生新的文件编号 |

正常示例:

```
{
  "errno": 0,
  "cost": 533,
  "errmsg": "",
  "data": {
    "fid": "f1343423222"
  }
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
```

&gt;

### 3.3、下载文件

说明：

根据文件编号（fid）下载指定文件。

Request Path: [sdk-host]/storage/download/

Method: GET

其他 url 参数：无

需要参与签名的其他 URL 参数：

| 参数名 | 说明   | 是否必填 | 最大长度 |               |
|-----|------|------|------|---------------|
| fid | 文件编号 | Y    | 20   | 上传合同文件得到的文件编号 |

请求参数（REQUEST BODY）：无

示例：

```
GET /storage/download/?
developerId=d445f5432&rtick=17364899234&sign=a2334df21323&signType=rsa&fid=150496133501000001
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：文件流（byte[]）

正常示例：

正常则返回文件流，开发者自行另存到指定的路径。

异常示例：

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

### 3.4、PDF文件验签

说明：

上传 PDF 文件验证签名，返回 PDF 文件是否有证书签名、签名/文件在签名后是否有篡改，以及每个证书签名的具体信息。

Request Path: [sdk-host]/pdf/verifySignatures/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名     | 说明     | 是否必填 | 最大长度 |                       |
|---------|--------|------|------|-----------------------|
| pdfData | PDF 文件 | Y    |      | PDF 文件 base64 编码过的字符串 |

示例：

```
{  
  "pdfData": "ZHNmYXNnZHNnZHNhZ3Nn.." }  
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

| 参数名              | 说明                        |                                  |                                    |
|------------------|---------------------------|----------------------------------|------------------------------------|
| verifyResult     | 验证结果                      | 0-文档无证书签名<br>1-文档未被篡改<br>2-文档被篡改 |                                    |
| signatureDetails | 签名明细，<br>json array<br>格式 | 参数                               | 说明                                 |
|                  |                           | signatureNumber                  | 签名序号（ADOBE<br>PDF 阅读器查看的修<br>订版序号） |
|                  |                           | dn                               | 证书 DN 项                            |
|                  |                           | signTime                         | 签署时间                               |
|                  |                           | serialNumber                     | 证书序列号                              |
|                  |                           | singer                           | 签署者名称                              |
|                  |                           | signIsModified                   | 该签名有无被篡改：<br>0-未篡改 1-已篡改           |
|                  |                           | documentIsModified               | 签名后文档有无被篡<br>改：<br>0-未篡改 1-已篡改     |

正常示例：

```
{  
  "errno": 0,  
  "errmsg": "",  
  "cost": 533,  
  "data": {  
    "verifyResult": 0,  
    "signatureDetails": [  
      {  
        "signatureNumber": 1,  
        "dn": "CN=Zhang San, O=BestSign",  
        "signTime": "2023-10-27 10:10:10",  
        "serialNumber": 1,  
        "singer": "Zhang San",  
        "signIsModified": 0,  
        "documentIsModified": 0  
      }  
    ]  
  }  
}
```

```
"verifyResult": "1",
"signatureDetails": [
  {
    "signatureNumber": 1,
    "singer": "测试企业有限公司",
    "serialNumber": "1003623585",
    "dn": "C=CN,O=Zhejiang Digital Certificate Authority,CN=ZJCA OCA2",
    "signTime": "2017-12-22 15:54:26",
    "signIsModified": "0",
    "documentIsModified": "0"
  },
  {
    "signatureNumber": "3",
    "serialNumber": "74c8003d0001d901",
    "singer": "测试_青岛企业",
    "signTime": "2017-12-22 15:56:35",
    "signIsModified": "0",
    "dn": "C=CN,O=Zhejiang Digital Certificate Authority,CN=ZJCA OCA3",
    "documentIsModified": "0"
  }
]
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 4、单文档合同服务

- 关于 toB/toC 合同：合同签署人中除了发送者自己外还有至少一个企业就是 toB 合同，否则就是 toC 合同。

### 4.1、上传并创建合同

说明:

- 此接口是将“3.1、上传合同文件”与“4.11、创建合同”的封装在一起，方便开发者调用。调用此接口后，将文件保存到自定义的存储中，暂时只支持上传 PDF 文件。上

传成功后得到一个合同编号，后续流程可以直接使用此合同编号进行签署。

- 如果开发者需要用到空白合同当模版，添加不同的 PDF 元素时，请分开使用“3.1、上传合同文件”、“3.2、PDF 文件添加元素”与“4.11、创建合同”。

Request Path: **[sdk-host]**/storage/contract/upload/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY):

| 参数名     | 说明             | 是否必填 | 最大长度    |  |
|---------|----------------|------|---------|--|
| account | 用户帐号           | Y    | 64      | 必须要指定一个用户帐号作为操作者   |
| fmd5    | 文件 md5 值       | Y    | 100     | 例如:<br><pre>FileInputStream file = new FileInputStream("d: \\test\\接口系统.pdf");<br/>byte[] bdata = IOUtils.toByteArray(file);<br/>String fmd5 = DigestUtils.md5Hex(bdata);</pre>        |
| ftype   | 文件类型           | Y    | 10      | 仅支持 PDF  |
| fname   | 文件名            | Y    | 50      | 文件名必须带上后缀名，例如“XXXX.pdf”  |
| fpages  | 文件页数           | Y    | 3       |  |
| fdata   | 文件数据，base64 编码 | Y    | 5120000 | 例如:<br><pre>FileInputStream file = new FileInputStream("d: \\test\\接口系统.pdf");<br/>byte[] bdata = IOUtils.toByteArray(file);<br/>String fdata =Base64.encodeBase64String(bdata);</pre> |
| title   | 合同标题           | Y    | 50      | 客户可以将自己的业务合同编号、或合同标题放此处  |

|                              |             |   |     |   |
|------------------------------|-------------|---|-----|---|
| <b>expireTime</b>            | 合同能够签署的到期时间 | Y |     | 合同必须在指定的到期时间之前完成签署，一旦过期则此合同将无法被签署。格式为秒级的 unix 时间戳，如希望“2017/12/30 10:21:52”到期，则设置为“1514600512”；  |
| <b>description</b>           | 合同描述        | N | 200 |   |
| <b>hotStorage<br/>Period</b> | 热存周期        | N |     | 此参数是合同文件在热存储中保留的时间长度，单位为秒。保存在热存储中的合同数据，自合同结束时间算起，超过此参数设定时长的合同文件，会转移到冷存储中。计算示例：如保存 365 天，则值为 $3600 \times 24 \times 365 = 31536000$ 。参数可为空，为空时默认值为 1 年（31536000）。取值范围为 3600（1 小时）~157680000（5 年） |

示例：

```
{
  "account": "mlimd@163.com",
  "fmd5": "文件的 md5 值",
  "ftype": "pdf",
  "fname": "滴滴电子发票.pdf",
  "fpages": "1",
  "fdata": "文件的 base64 字符串",
  "title": "测试合同标题",
  "description": "测试合同描述",
  "expireTime": "1531242131"
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

| 参数名               | 说明                |
|-------------------|-------------------|
| <b>contractId</b> | 创建的合同 ID，在后续接口中使用 |

正常示例：

```
{
  "errno": 0,
  "cost": 533,
  "data": {
    "contractId": "150993706101000005"
  },
  "errmsg": ""
}
```



异常示例：

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 4.2、签署合同（即自动签）

说明：签署合同，可直接用于无感知的快捷签署。此接口要求签署者必须拥有自己的数字证书。可以无需先调用“添加签署者”接口与“设置合同签署参数”接口。

Request Path: [sdk-host]/storage/contract/sign/cert/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名                | 说明                    | 是否必填 | 最大长度 |                |
|--------------------|-----------------------|------|------|----------------|
| contractId         | 合同编号                  | Y    |      |                |
| signer             | 签署者                   | Y    |      | 即签署者的 account  |
| signaturePositions | 指定的签署位置，json array 格式 | Y    |      | 每一项为一个 json 对象 |
|                    |                       |      |      |                |
|                    |                       |      |      | x              |
|                    |                       |      |      | y              |
|                    |                       |      |      | pageNum        |
|                    |                       |      |      | rptPageNums    |

|                             |          |   |  |  |  |
|-----------------------------|----------|---|--|--|--|
|                             |          |   |  | 其他页。参数启用之后，当前页的签名会同时出现在其他页的相同位置。参数值格式如下：多个页码使用英文逗号(,)分隔。参数不存在或者参数值为空表示不复制，参数值为 0 表示复制到所有页，参数值为其他表示复制到指定页。参数值示例："0"<br><br>"1,2,3,4,5,6" "1,3,5,7,9"    |  |
| <b>signatureImageName</b>   | 签名图片名称   | N |  | 企业用户可以指定上传过的印章图片中的某一张作为本次签署的印章图片，取“上传用户签名/印章图片”接口设置的 <b>imageName</b> 的值   |  |
| <b>signatureImageData</b>   | 签名图片     | N |  | 用户指定的签名图片，Base64 字符串。允许为空，为空时使用用户默认的签名图片。<br><br>使用自动签的用户，可以越过“2.1、创建用户签名/印章图片”接口、或“2.2、上传用户签名/印章图片”接口，直接在此提供签名/印章图片即可。                                |  |
| <b>signatureImageWidth</b>  | 签名图片显示宽度 | N |  | 本次签署使用的签名/印章图片在合同 PDF 上显示的宽度，72dpi 下的磅值。如果指定位置上没有设置签名图片的宽度和高度，那么按照如下默认值处理：<br>个人签名：默认高度：48 磅，最大宽度：130 磅，宽度一般小于等于 130 磅。<br>企业印章：默认宽度：130 磅，默认高度：130 磅。 |  |
| <b>signatureImageHeight</b> | 签名图片显示高度 | N |  | 本次签署使用的签名/印章图片在合同 PDF 上显示的宽度，72dpi 下的磅值。如果指定位置上没有设置签名图片的宽度和高度，那么按照如下   |  |

|  |  |  |  |   |
|--|--|--|--|---|
|  |  |  |  | <p>默认值处理：</p> <p>个人签名：默认高度：48 磅，最大宽度：130 磅，宽度一般小于等于 130 磅。</p> <p>企业印章：默认宽度：130 磅，默认高度：130 磅。</p> |
|--|--|--|--|---|

示例：

```
{
  "contractId": "cU34344234",
  "signer": "u-1112343242",
  "signatureImageData": "cU34344234",
  "signaturePositions": [
    {
      "pageNum": "1",
      "x": "0.05",
      "y": "0.71",
      "rptPageNums": "2,3,4,5,6,7",
      "signatureImageData": "图片 base64",
      "signatureImageWidth": "100",
      "signatureImageHeight": "70"
    },
    {
      "pageNum": "2",
      "x": "0.05",
      "y": "0.81",
      "signatureImageData": "图片 base64",
      "signatureImageWidth": "100",
      "signatureImageHeight": "70"
    }
  ]
}
```

响应（RESPONSE）：

正常示例：

```
{
  "errno": 0,
  "cost": 533,
  "data": { },
  "errmsg": ""
}
```

异常示例：

```
{
  "errno": 240012,
```

```
"cost": 533,
"data": null,
"errmsg": "check sign wrong "
}
```

#### 4.3、发送合同（即手动签，指定图片大小）

说明：获取合同手动签署的页面 URL，将此 URL 发送给指定用户，即可在此页面进行手动签操作完成签署。

Request Path: [sdk-host]/contract/send/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名                | 说明  |                                      | 是否必填 | 最大长度 | 默认值 |
|--------------------|---|--------------------------------------|------|------|-----|
| contractId         | 合同编号  |                                      | Y    |      |     |
| signer             | 指定给哪个用户看  |                                      | Y    |      |     |
| dpi                | 预览图片清晰度，枚举值：96-低清（默认），120-普清，160-高清，240-超清  |                                      | N    | 3    | 96  |
| expireTime         | 签署链接的过期时间，unix 时间戳。超过此时间则无法打开签署链接，需要获取新的签署链接。格式为秒级的 unix 时间戳，如希望“2017/12/30 10:21:52”到期，则设置为“1514600512”。<br><br>注：并不是合同的可签署到期时间，只是此签署链接的有效期。<br><br>由于返回的是短链接，如果没有设置则默认是 7 天，如果设置了以设置的为准，但有效期最大不能超过 7 天。 |                                      | N    | 20   | 0   |
| signaturePositions | 参数名   | 说明                                   | 最大长度 | Y    |     |
|                    | y   | 纵坐标，同上                               | 10   |      |     |
|                    | x   | 横坐标，按页面尺寸的百分比计算，取值 0.0 - 1.0。以左上角为原点 | 10   |      |     |
|                    | pageNum   | 页码，从 1 开始，不能超过合同最大页数                 | 3    |      |     |

|                                |   |  |   |         |    |
|--------------------------------|---|--|---|---------|----|
|                                | rptPageNums   | 当前位置的签名需要复制到的目标页码列表。该参数用于控制是否将当前位置的签名复制到其他页。参数启用之后，当前页的签名会同时出现在其他页的相同位置。参数值格式如下：<br>多个页码使用英文逗号(,)分隔。 参数不存在或者参数值为空表示不复制，参数值为 0 表示复制到所有页， 参数值为其他表示复制到指定页。 参数值示例: "0" "1,2,3,4,5,6" "1,3,5,7,9" |   |         |    |
|                                | type  | 日期类型专用，当签署位置是“日期”类型的签名时，本参数需要填写“date”  |   |         |    |
|                                | dateTimeFormat  | 日期类型专用，当签署位置是“日期”类型的签名时，本参数需要填写日期的格式，格式支持以下类型：<br>MM-dd-yyyy<br>yyyy-MM-dd<br>MM.dd.yyyy<br>Yyyy.MM.dd<br>MM/dd/yyyy<br>Yyyy/MM/dd   |   |         |    |
|                                | fontSize  | 日期类型专用，当签署位置是“日期”类型的签名时，本参数需要填写字号大小  |   |         |    |
| isAllowChangeSignaturePosition | 在有指定 signaturePOSTion 参数的情况下，是否允许拖动签名位置。取值 1/0。（0：不允许，1：允许）   |  | N | 1       | 0  |
| returnUrl                      | 手动签署时，当用户签署完成后，签署页面指定回跳的页面地址，如果没填此参数则签署完成后跳转到合同预览页面。<br><b>如果 returnUrl 带有“http”或“https”开头则判定 h5 跳转，如果 returnUrl 不带“http”或“https”开头则判定为微信小程序跳转。</b> |  | N |         |    |
| vcodeMobile                    | 手动签署时指定接收手机验证码的手机号，如果需要采用手动签署页面校验短信验证码，则此项必填。<br>如果手动签不需要短信校验，则此项可不填。<br>自动签可不填此项。  |  | N |         | 20 |
| isDrawSignatureImage           | 手动签署时是否手绘签名。取值 0/1/2。<br>0 点击签名图片不会触发手写面板（禁止手写）。<br>1 点击签名图片能触发手写面板（既可手写也可使用默认签名）。<br>2 强制必须手绘签名（只能手写不允许使用默认签名）。                                    |  | N | 1       | 1  |
| signatureImageName             | 签名/印章图片。取值 default 或者指定的印章名称。（ <b>手动签只有参数 version=3 时才支持企业多印章，填写自定义</b>   |  | N | default | 50 |

|                      |  |   |   |   |
|----------------------|--|---|---|---|
|                      | 义的印章名称)  |   |   |   |
| pushUrl              | 此处有配置则签署推送消息优先使用此配置，如果此处没有配置，则取开发者统一配置的异步推送地址  | N |   |   |
| readAll              | 枚举值 0/1，默认为 0<br>0-无需拖动到页面底部即可确认签署<br>1-必须拖动到页面底部，表示阅读完毕才能确认签署   | N |   |   |
| version              | 手动签的版本，默认为 2，枚举值：<br>2-v2 版本，<br>3-v3 版本，UI 优化，移动端手绘面板自动横屏   | N | 1 | 2 |
| signatureImageWidth  | 签名图片显示宽度，72dpi 下的磅值。本次签署使用的签名/印章图片在合同 PDF 上显示的宽度，高度会根据图片比例自动缩放。<br>目前如果指定位置上没有设置签名图片的宽度和高度，那么按照如下默认值处理：<br>个人签名：默认高度：48 磅，最大宽度：130 磅，宽度一般小于等于 130 磅。<br>企业印章：默认宽度：130 磅，默认高度：130 磅。  | N |   |   |
| app                  | app 人脸比对签署（即 app 内刷脸签署）用到的 app 参数，参数内容为 JSBridge，即手动签页面将此参数传递给客户 app 以便客户 app 调用对应的 SDK。<br>一般情况，如果不需要用户跳出手动签页面执行其他 app 操作的话，则 REQUEST BODY 不需要带此参数。<br>新版手动签在返回签署人配置的时候，会将此参数返回给前端，前端需要将此参数传递给 app。   | N |   |   |
| signatureImageWidth  | 本次签署使用的签名/印章图片在合同 PDF 上显示的宽度，72dpi 下的磅值。如果指定位置上没有设置签名图片的宽度和高度，那么按照如下默认值处理：<br>个人签名：默认高度：48 磅，最大宽度：130 磅，宽度一般小于等于 130 磅。<br>企业印章：默认宽度：130 磅，默认高度：130 磅。<br><b>注意：signatureImageWidth 和 signatureImageHeight 必须同时为空，或者同时存在。</b><br><b>同时为空则使用系统默认的高度和宽度（宽度和高度最大 130 磅，小于 130 磅的使用原始尺寸，大于 130 磅的按比例缩放）。</b><br><b>如果同时有值，则图片按照两个宽高值拉伸/压缩填充，效果可能会产生变形。</b> | N |   |   |
| signatureImageHeight | 本次签署使用的签名/印章图片在合同 PDF 上显示的高度，72dpi 下的磅值。如果指定位置上没有设置签名图片的宽度和高度，那么按照如下默认值处理：<br>个人签名：默认高度：48 磅，最大宽度：130 磅，宽度一般小于等于 130 磅。<br>企业印章：默认宽度：130 磅，默认高度：130 磅。   | N |   |   |

|                       |   |   |   |   |
|-----------------------|---|---|---|---|
|                       | 注意：signatureImageWidth 和 signatureImageHeight 必须同时为空，或者同时存在。<br>同时为空则使用系统默认的高度和宽度（宽度和高度最大 130 磅，小于 130 磅的使用原始尺寸，大于 130 磅的按比例缩放）。<br>如果同时有值，则图片按照两个宽高值拉伸/压缩填充，效果可能会产生变形。 |   |   |   |
| isShowHandwrittenTime | 枚举值 0/1，默认为 0<br>1-手写面板及签名图片上显示手写签名操作的时间<br>0-不显示该时间  | N | 1 | 0 |
| isFaceAuth            | 是否使用刷脸签。<br>1-刷脸签，使用刷脸校验；<br>0 或其他-非刷脸签，使用短信校验或无校验；   | N |   |   |

示例：

```
{
  "contractId": "cU34344234",
  "expireTime": "0",
  "dpi": "120",
  "signer": "",
  "signaturePositions": [
    {
      "pageNum": "1",
      "x": "0.4622",
      "y": "0.5852999973297119",
      "rptPageNums": "",
      "type": "date",
      "dateTimeFormat": "MM-dd-yyyy",
      "fontSize": "18"
    },
    {
      "pageNum": "6",
      "x": "0.4622",
      "y": "0.7852999973297119"
    }
  ],
  "isAllowChangeSignaturePosition": "0",
  "returnUrl": "",
  "vcodeMobile": "",
  "isDrawSignatureImage": "1",
}
```

```
"signatureImageName": "default",  
"pushUrl": "https://xxxxxxx"  
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

手动签署页面的 url，签署者在此页面上进行签署操作。

正常示例：

```
{  
  "errno": 0,  
  "cost": 533,  
  "errmsg": "",  
  "data": {  
    "url": "https://....."  
  }  
}
```

异常示例：

```
{  
  "errno": 240012,  
  "cost": 533,  
  "data": null,  
  "errmsg": "check sign wrong "  
}
```

#### 4.4、锁定并结束合同

说明：当所有签署者都签署完毕后，使用此接口添加上上签系统的锁定签名，并且结束整个合同签署过程，将合同状态更新为“已完成”，不允许任何人再进行签署。只有结束的合同才能提供公证服务。

Request Path: [sdk-host]/storage/contract/lock/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名        | 说明   | 是否必填 |  |
|------------|------|------|--|
| contractId | 合同编号 | Y    |  |

示例：

```
{
```



```
"contractId": "1703465339807805447"
}
```

#### 响应 (RESPONSE) :

正常示例:

```
{
  "errno": 0,
  "cost": 533,
  "data": { },
  "errmsg": ""
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 4.5、撤销合同

说明: 当发现合同内容出错等情况时, 可用来直接取消未结束的合同 (由开发者发起的取消, 不需要确认签署者), 避免造成错误的合同完成了签署。

Request Path: [sdk-host]/contract/cancel/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

#### 请求参数 (REQUEST BODY) :

| 参数名        | 说明   | 是否必填 | 最大长度 |
|------------|------|------|------|
| contractId | 合同编号 | Y    |      |

示例:

```
{
  "contractId": "1703466073594003458"
}
```

#### 响应 (RESPONSE) :

正常示例:

```
{
  "errno": 0,
  "cost": 533,
```

```
"data": { },  
"errmsg": ""  
}
```

异常示例:

```
{  
  "errno": 240012,  
  "cost": 533,  
  "data": null,  
  "errmsg": "check sign wrong "  
}
```

## 4.6、查询合同信息

说明: 获取合同的详细信息, 如签署者信息、合同状态等。

Request Path: [sdk-host]/contract/getInfo/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY):

| 参数名        | 说明   | 是否必填 |  |
|------------|------|------|--|
| contractId | 合同编号 | Y    |  |

示例:

```
{  
  "contractId": "cU34344234"  
}
```

响应 (RESPONSE):

RESPONSE DATA 参数如下:

| 参数名           | 说明                                 |                             |
|---------------|------------------------------------|-----------------------------|
| contractId    | 合同编号                               |                             |
| fid           | 文件编号                               |                             |
| userId        | 用户 id                              | 即创建者的 account 在上上签系统内的用户 id |
| senderAccount | 合同创建者唯一标识 (之前版本中的 senderUserId 作废) | 即创建者的 account               |

|             |                                      |   |
|-------------|--------------------------------------|---|
| description | 合同描述                                 |   |
| expireTime  | 合同签署的到期时间（之前版本中的 <b>expireAt</b> 作废） | Unix 时间戳（秒数），格式：1490976000                |
| title       | 标题                                   |   |
| sendTime    | 合同的发送时间                              | 添加第一个签署者的时间，当第一个签署者添加成功即认为合同已发送           |
| finishTime  | 合同的完成时间                              | 调用锁定合同接口之后结束合同的时间                         |
| signers     | 合同签署者 account 集合.json array 形式。      | "signers": [<br>"mlimd@163.com"<br>]      |
| developerId | 开发者编号                                |   |
| pages       | 合同页数                                 |   |
| status      | 合同状态                                 | 2：已创建；3：已发送，正在签署中；4：拒绝签署,已取消；5：已完成；9：已过期； |

正常示例：

```
{
  "errno": 0,
  "cost": 533,
  "data": {
    "fid": "1702695409383636995",
    "senderAccount": "1364808",
    "userId": "1693391393705164808",
    "description": "postman 测试合同",
    "expireTime": "1490976000",
    "title": "测试",
    "sendTime": "2017-04-10 17:31:46",
    "finishTime": "2017-04-10 17:31:46",
    "signers": [
      "mlimd@163.com"
    ],
    "developerId": "1678748413568483333",
    "pages": "1",
    "contractId": "1702698069495119878",
    "status": "3"
  },
}
```

```
"errmsg": ""  
}
```

异常示例:

```
{  
  "errno": 240012,  
  "cost": 533,  
  "data": null,  
  "errmsg": "check sign wrong "  
}
```

## 4.7、查询合同签署参数

说明: 获取已经设置过的签署者合同签署参数。

Request Path: [sdk-host]/contract/getSignerConfig/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY):

| 参数名        | 说明   | 是否必填 | 最大长度 | 默认值 |
|------------|------|------|------|-----|
| contractId | 合同编号 | Y    |      |     |
| account    | 签署者  | Y    |      |     |

示例:

```
{  
  "contractId": "cU34344234",  
  "account": "2348@qq.com"  
}
```

响应 (RESPONSE):

RESPONSE DATA 参数如下:

| 参数名                            | 说明  |
|--------------------------------|---|
| signaturePositions             | 默认的签署位置   |
| isAllowChangeSignaturePosition | 在有指定 signaturePOSTion 参数的情况下, 是否允许拖动签名位置。取值 1/0。 (0: 不允许, 1: 允许, 下面都是这样的) |
| password                       | 合同密码  |
| isVerifySigner                 | 手动签署时是否验证 account 身份。取值 0/1。如果要验证 account 身份, 则忽略"查看密码"和"验证码"的设置          |
| vcodeMail                      | 手动签署时指定接收邮件验证码的邮箱。  |
| vcodeMobile                    | 手动签署时指定接收手机验证码的手机号。   |
| isDrawSignatureImage           | 手动签署时是否手绘签名。取值 0/1  |
| signatureImageName             | 签名/印章图片。取值 default 或者指定的印章名称。   |

| certType | 证书类型 |
|----------|------|
|----------|------|

正常示例:

```
{
  "errno": 0,
  "cost": 533,
  "errmsg": "",
  "data": {
    "signaturePositions": [
      {
        "pageNum": "1",
        "x": "0.05",
        "y": "0.01"
      },
      {
        "pageNum": "2",
        "x": "0.05",
        "y": "0.01"
      }
    ],
    "isAllowChangeSignaturePosition": "0",
    "pushUrl": "",
    "password": "",
    "isVerifySigner": "0",
    "vcodeMail": "",
    "vcodeMobile": "",
    "isDrawSignatureImage": "1",
    "signatureImageName": "default",
    "certType": ""
  }
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 4.8、查询合同签署者状态

说明: 查询合同签署者的签署状态。

Request Path: [sdk-host]/contract/getSignerStatus/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名        | 说明   | 是否必填 |  |
|------------|------|------|--|
| contractId | 合同编号 | Y    |  |

示例：

```
{
  "contractId": "cU34344234"
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

data 内容为合同签署者状态，为 json 格式，格式为“account - 状态码”成对出现。状态码：

1 — 未签署； 2 - 已签署； 3 – 拒绝签署。

正常示例：

```
{
  "errno": 0,
  "cost": 533,
  "data": {
    "mlimd@163.com": "2",
    "13067825582": "2",
    "13867410069": "2"
  },
  "errmsg": ""
}
```

异常示例：

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 4.9、下载合同文件

说明：通过合同编号下载对应的合同文件。

Request Path: [sdk-host]/storage/contract/download/

Method: GET

其他 url 参数：contractId

需要参与签名的其他 URL 参数：

| 参数名        | 说明   | 是否必填 | 最大长度 |
|------------|------|------|------|
| contractId | 合同编号 | Y    |      |

请求参数（REQUEST BODY）：无

示例：

```
GET [sdk-host]/storage/contract/download/?
developerId=d445f5432&rtick=17364899234&sign=a2334df21323&signType=rs
a&contractId=150496133501000001
```

响应（RESPONSE）：

正常示例：

文件流

异常示例：

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 4.10、获取预览页URL

说明：获取合同预览的页面 URL，在此页面预览合同内容。

Request Path: [sdk-host]/contract/getPreviewURL/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名        | 说明  | 是否<br>必填 | 最大<br>长度 | 默认<br>值 |
|------------|---|----------|----------|---------|
| contractId | 合同编号  | Y        |          |         |
| account    | 指定给哪个用户看  | Y        |          |         |
| dpi        | 预览图片清晰度，枚举值：96-低清（默认），120-普清，160-高清，240-超清        | N        | 3        | 96      |
| expireTime | 预览链接的过期时间，unix 时间戳。超过此时间则无法打开预览链接，需要获取新的预览链接。格式为秒 | N        | 20       | 0       |

|         |  |   |   |   |
|---------|--|---|---|---|
|         | 级的 unix 时间戳，如希望“2017/12/30 10:21:52”到期，则设置为“1514600512”。<br><br>注：并不是合同的可签署到期时间，只是此预览链接的有效期。<br><br>如果没有设置则默认是 7 天，如果设置了以设置的为准，但有效期最大不能超过 7 天。 |   |   |   |
| version | 手动签的版本，默认为 2，枚举值：<br>2-v2 版本，<br>3-v3 版本，UI 优化，移动端手绘面板自动横屏   | N | 1 | 2 |

示例：

```
{  
  "contractId": "cU34344234",  
  "expireTime": "0",  
  "dpi": "96",  
  "account": ""  
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

预览合同页面的 url，签署者在此页面上预览合同，但不可操作签署。

正常示例：

```
{  
  "errno": 0,  
  "cost": 533,  
  "errmsg": "",  
  "data": {  
    "url": "https://....."  
  }  
}
```

异常示例：

```
{  
  "errno": 240012,  
  "cost": 533,  
  "data": null,  
  "errmsg": "check sign wrong "  
}
```

## 4.11、创建合同

说明：使用已得到的文件编号创建合同，得到合同编号，此合同还未添加签署者，属于草



稿性质。每份合同必须要有创建者 account。

Request Path: [sdk-host]/contract/create/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY):

| 参数名              | 说明          | 是否必填 | 最大长度 |   |
|------------------|-------------|------|------|---|
| account          | 用户帐号        | Y    |      | 必须要指定一个用户帐号作为合同的创建者（建议统一为开发者的 account，或者需要进行分类统计的某些 account，以便后期按此 account 进行统计）  |
| fid              | 文件编号        | Y    |      | 上传合同文件得到的文件编号   |
| expireTime       | 合同能够签署的到期时间 | Y    |      | 合同必须在指定的到期时间之前完成签署，一旦过期则此合同将无法被签署。格式为秒级的 unix 时间戳，如希望“2017/12/30 10:21:52”到期，则设置为“1514600512”；  |
| title            | 合同标题        | Y    | 50   | 客户可以将自己的业务合同编号、或合同标题放此处   |
| description      | 合同描述        | N    | 200  |   |
| hotStoragePeriod | 热存周期        | N    |      | 此参数是合同文件在热存储中保留的时间长度，单位为秒。保存在热存储中的合同数据，自合同结束时间算起，超过此参数设定时长的合同文件，会转移到冷存储中。计算示例：如保存 365 天，则值为 $3600 \times 24 \times 365 = 31536000$ 。参数可为空，为空时默认值为 1 年（31536000）。取值范围为 3600（1 小时）~157680000（5 年） |

示例:

```
{
  "account": "mlimd@163.com",
  "fid": "1702695409383636995",
  "expireTime": "1513311132",
  "title": "测试",
  "description": "postman 测试合同"
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

| 参数名        | 说明   |
|------------|------|
| contractId | 合同编号 |

正常示例：

```
{
  "errno": 0,
  "cost": 533,
  "data": {
    "contractId": "150993706101000005"
  },
  "errmsg": ""
}
```

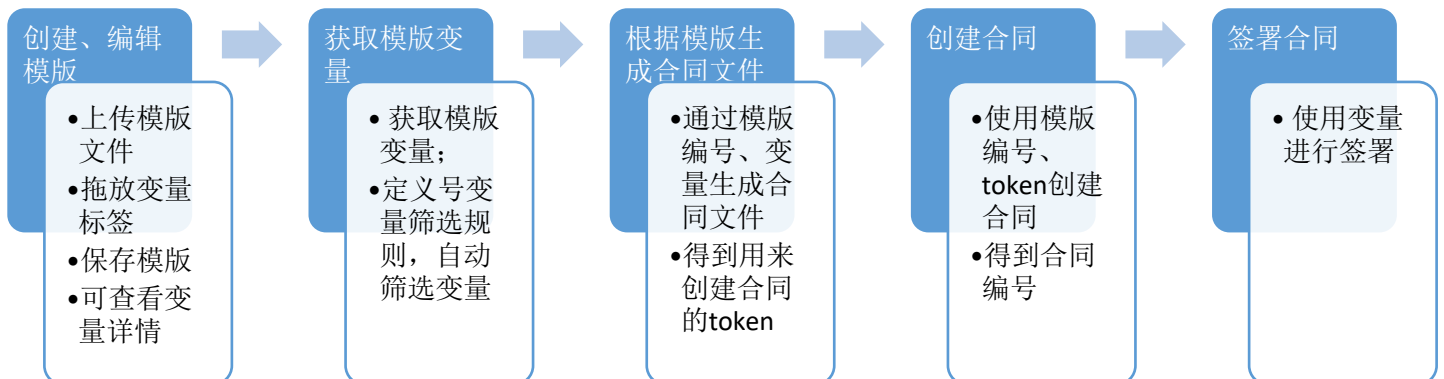
异常示例：

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 4. 20、模版签署功能

模版功能包含了一整套从发到签的接口，使用模版功能，可以实现“编辑一次、重复使用”的目的，并且无需计算签署位置坐标等参数，为开发者对接调试提供极大方便。

此功能与开放平台的页面功能“模版管理”相结合，使用流程为：



在开放平台的“模版管理”页面上可以可视化的创建、编辑模版，保存后，可以通过下列接口进行合同的创建、签署。

在“模版管理”页面上可以查看每一份模版的变量详情，这些变量详情中有一些内容会在“创建合同”、“签署合同”等不同的环节用到。

| 变量类型<br>(type) | 说明       | 用途                               | 何时会<br>用到 |
|----------------|----------|----------------------------------|-----------|
| 50             | 盖章       | 企业用户盖章的位置                        | 签署        |
| 40             | 签名       | 个人用户签名的位置                        | 签署        |
| 20             | 签署<br>日期 | 具体每个用户可能用到的签署日期                  | 签署        |
| 11             | 文本       | 自定义的合同变量，比如在模版上要填充的“地址”、“身份证号”等等 | 创建合<br>同  |

变量详情查看的具体内容如下所示：

```
{
  "fid": "7388583176326841500",
  "templateVars": [
    {
      "isRequired": "1",
      "overflow": "0",
      "type": "50",          //表示“盖章”
      "pageNum": "1",       //签署位置的页码
      "fontFamily": "",
      "dateTimeFormat": "yyyy-MM-dd HH:mm:ss",
      "name": "甲方盖章 1", //调签署接口时用到 name 变量
      "tagType": "0",
      "width": "0.2052896725440806",
      "x": "0.17758186397984888", //签署位置的 x 坐标
      "y": "0.182546749777382",   //签署位置的 y 坐标
      "fontSize": "0",
      "tag": "盖章",
      "text": "",
      "height": "0.12733748886910062"
    },
    {
      "isRequired": "1",
      "overflow": "0",
      "type": "40",          //表示“签名”
      "pageNum": "1",       //签署位置的页码
      "fontFamily": "",
      "dateTimeFormat": "yyyy-MM-dd HH:mm:ss",
      "name": "乙方签名 1", //调签署接口时用到此变量
      "tagType": "0",
      "width": "0.0654911838790932",
      "x": "0.5478589420654912", //签署位置的 x 坐标
    }
  ]
}
```

```
"y": "0.25645592163846836",    //签署位置的 y 坐标
"fontSize": "0",
"tag": "签名",
"text": "",
"height": "0.030276046304541407"
},
{
  "isRequired": "1",
  "overflow": "0",
  "type": "20",                //表示“签署日期”
  "pageNum": "1",             //签署位置的页码
  "fontFamily": "",
  "dateTimeFormat": "yyyy-MM-dd HH:mm:ss",
  "name": "甲方签署日期 1",   //调签署接口时用到此变量
  "tagType": "0",
  "width": "0.2506297229219144",
  "x": "0.25440806045340053", //签署位置的 x 坐标
  "y": "0.3873552983081033",  //签署位置的 y 坐标
  "fontSize": "16",
  "tag": "签署日期",
  "text": "",
  "height": "0.02315227070347284"
},
{
  "isRequired": "1",
  "overflow": "0",
  "type": "11",                //表示“自定义文本”
  "pageNum": "1",
  "fontFamily": "SimSun",
  "dateTimeFormat": "yyyy-MM-dd HH:mm:ss",
  "name": "甲方地址 1",      //在创建合同时会用到
  "tagType": "0",
  "width": "0.12594458438287154",
  "x": "0.6964735516372796",
  "y": "0.3846838824577026",
  "fontSize": "18.7",
  "tag": "文本",
  "text": "",
  "height": "0.02226179875333927",
  "warp": "true"
},
{
  "isRequired": "1",
  "overflow": "0",
```

```
        "type": "11",
        "pageNum": "1",
        "fontFamily": "SimSun",
        "dateTimeFormat": "yyyy-MM-dd HH:mm:ss",
        "name": "乙方身份证",
        "tagType": "0",
        "width": "0.12594458438287154",
        "x": "0.7670025188916877",
        "y": "0.4790739091718611",
        "fontSize": "18.7",
        "tag": "文本",
        "text": "",
        "height": "0.02226179875333927",
        "warp": "true"
    },
    ],
    "pageCount": 2,
    "fileName": "测试合同.pdf",
    "developerId": "1678748413568483333",
    "description": "未描述",
    "titleHash": "011c945f30ce2cbafc452f39840f025693339c42",
    "title": "1111",
    "isDel": "0",
    "tid": "151393625101000001"    //模版编号
}
```

#### 4.20.0、获取模版变量

说明：使用此接口获取模版的所有变量，用于生成合同文件或签署合同。

Request Path: **[sdk-host]**/template/getTemplateVars/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名               | 说明       | 是否必填 |                       |
|-------------------|----------|------|-----------------------|
| tid               | 模版编号     | Y    |                       |
| isRetrieveAllVars | 是否获取所有变量 | Y    | 0 只返回变量的 type 和 name; |

|  |  |  |              |
|--|--|--|--------------|
|  |  |  | 1 返回变量的所有字段; |
|--|--|--|--------------|

示例:

```
{
  "tid": "111222333",
  "isRetrieveAllVars": "0"
}
```

响应 (RESPONSE) :

RESPONSE DATA 参数如下:

| 参数名          | 说明                 |
|--------------|--------------------|
| templateVars | 模版变量, jsonArray 格式 |

正常示例:

```
{
  "errno": 0,
  "errmsg": "",
  "data": {
    "templateVars": [
      {
        "name": "甲方住址 1",
        "type": "11"
      },
      {
        "name": "签署日期 1",
        "type": "20"
      },
      {
        "name": "甲方签名 1",
        "type": "40"
      },
      {
        "name": "乙方盖章 1",
        "type": "50"
      }
    ]
  }
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
```

}

### 4.20.1、通过模版生成合同文件

说明：

在开放平台的“开发者模版”页面编辑保存好模版之后，就可以通过接口使用该模版。先调本接口通过模版功能创建合同文件。


Request Path: **[sdk-host]**/template/createContractPdf/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名            | 说明                      | 是否必填 | 最大长度 |  |
|----------------|-------------------------|------|------|--|
| account        | 合同创建者账号                 | Y    |      |  |
| tid            | 模版编号                    | Y    |      | “开发者模版”页面模版保存后产生模版编号，可从页面上复制   |
| templateValues | 模版变量，<br>jsonObject 格式  | Y    |      | <p>将模版变量中需要填充的内容（即变量值）在此提交。</p> <p>由于在编辑模版时已经拖放了位置、并且已经保存在上上签系统中，因此这里无需再提交位置参数，只需要传入变量名称与对应的值即可。</p> <p>模版变量可以在开发者模版管理页面获取。</p>  |
| groupValues    | 模版变量组，<br>jsonObject 格式 | N    |      |  <p>模版变量组名，每个变量可以设置归什么组，然后用这个组名来传递变量。</p> <p><b>templateValues 和 groupValues 不能全部为空。</b></p> |

示例:

```
{
  "account": "111222333@qq.com",
  "tid": "1",
  "templateValues": {
    "甲方姓名": "张三",
    "甲方身份证号": "330101198111110311",
    "甲方地址": "浙江省杭州市西湖区灵隐路 1 号",
    "乙方企业名称": "北京朝阳群众信息技术有限公司",
    "乙方证件号码": "330101012345678",
    "乙方地址": "北京市朝阳区朝阳大厦 1024 室"
  },
  "groupValues": {
    "group1": "aaa",
    "group2": "bbb"
  }
}
```

响应 (RESPONSE):

RESPONSE DATA 参数如下:

| 参数名           | 说明   |
|---------------|--|
| templateToken | 通过模版创建合同需要这个 token，格式是一个 Base64 编码字符串 + "." + 一个自签名串 |

正常示例:

```
{
  "errno": 0,
  "errmsg": "",
  "cost": 533,
  "data": {
    "templateToken": "一串字符串"
  }
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```



## 4.20.2、通过模版创建合同

说明：

在“通过模版生成合同文件”得到 **templateToken** 之后，再调用本接口创建合同，得到合同编号。后续用模版变量签署必须使用本接口返回的合同编号。

Request Path: **[sdk-host]**/contract/createByTemplate/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数 (REQUEST BODY)：

| 参数名                     | 说明            | 是否必填 | 最大长度 |   |
|-------------------------|---------------|------|------|---|
| <b>account</b>          | 用户帐号          | Y    |      | 必须要指定一个用户帐号作为合同的创建者（建议统一为开发者的 <b>account</b> ，或者需要进行分类统计的某些 <b>account</b> ，以便后期按此 <b>account</b> 进行统计）                               |
| <b>tid</b>              | 模版编号          | Y    |      | 要用来创建合同的模版编号  |
| <b>templateToken</b>    | templateToken | Y    |      | 上一个接口 <b>/template/createContractPdf/</b> 返回的 <b>templateToken</b>  |
| <b>title</b>            | 合同标题          | Y    | 50   | 客户可以将自己的业务合同编号、或合同标题放此处   |
| <b>description</b>      | 合同描述          | N    | 200  |   |
| <b>expireTime</b>       | 合同能够签署的到期时间   | N    |      | 合同必须在指定的到期时间之前完成签署，一旦过期则此合同将无法被签署。格式为秒级的 <b>unix</b> 时间戳，如希望“2017/12/30 10:21:52”到期，则设置为“1514600512”；如果不填则默认为创建后加 7 天有效期。             |
| <b>hotStoragePeriod</b> | 热存周期          | N    |      | 此参数是合同文件在热存储中保留的时间长度，单位为秒。保存在热存储中的合同数据，自合同结束时间算起，超过此参数设定时长的合同文件，会转移到冷存储中。计算示例：如保存 365 天，则值为 $3600*24*365=31536000$ 。参数可为空，为空时默认值为 1 年 |

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | (31536000)。取值范围为 3600（1 小时）~157680000（5 年） |
|--|--|--|--|--|

示例：

```
{
  "account": "mlimd@163.com",
  "tid": "1702695409383636995",
  "templateToken": "上一个接口返回的一串字符串",
  "expireTime": "1513311132",
  "title": "测试",
  "description": "postman 测试合同"
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

| 参数名        | 说明   |
|------------|------|
| contractId | 合同编号 |

正常示例：

```
{
  "errno": 0,
  "cost": 533,
  "data": {
    "contractId": "150993706101000005"
  },
  "errmsg": ""
}
```

异常示例：

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

#### 4.20.3、用模版变量签署合同

说明：

使用模版功能创建合同后，使用本接口完成签署。

Request Path: **[sdk-host]**/contract/sign/template/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名                | 说明   | 是否必填 | 最大长度           |   |    |    |         |  |                    |  |
|--------------------|--|------|----------------|---|----|----|---------|--|--------------------|--|
| contractId         | 合同编号   | Y    |                |   |    |    |         |  |                    |  |
| tid                | 模版编号   | Y    |                |   |    |    |         |  |                    |  |
| vars               | 模版变量值  | Y    | JSON Object 格式 | <div>每一项为一个 jsonObject，格式类似如下：</div> <div>varName——模版变量名，比如“甲方盖章”、“甲方签署日期”等</div> <table><tr><th>参数</th><th>说明</th></tr><tr><td>account</td><td>签署者账号，同一份合同上的同一个签署者不可分多次签署。不同的签署者分次签署。</td></tr><tr><td>signatureImageData</td><td>签名图片，base64 编码字符串。<div>➤ 如果事先调用上传签名图片接口为签署者上传过签名图片，则此处可以不传，如果传了则优先使用此处上传的签名图片。</div><div>➤ 如果变量是签署日期类型则此项不传，传了会报错。</div></td></tr></table> | 参数 | 说明 | account | 签署者账号，同一份合同上的同一个签署者不可分多次签署。不同的签署者分次签署。 | signatureImageData | 签名图片，base64 编码字符串。 <div>➤ 如果事先调用上传签名图片接口为签署者上传过签名图片，则此处可以不传，如果传了则优先使用此处上传的签名图片。</div> <div>➤ 如果变量是签署日期类型则此项不传，传了会报错。</div> |
| 参数                 | 说明   |      |                |   |    |    |         |  |                    |  |
| account            | 签署者账号，同一份合同上的同一个签署者不可分多次签署。不同的签署者分次签署。   |      |                |   |    |    |         |  |                    |  |
| signatureImageData | 签名图片，base64 编码字符串。 <div>➤ 如果事先调用上传签名图片接口为签署者上传过签名图片，则此处可以不传，如果传了则优先使用此处上传的签名图片。</div> <div>➤ 如果变量是签署日期类型则此项不传，传了会报错。</div> |      |                |   |    |    |         |  |                    |  |
| loginIp            | 登录 IP  | N    |                | 签署者登录开发者平台的 IP 地址   |    |    |         |  |                    |  |
| signIp             | 签署 IP  | N    |                | 签署者操作通过开发者平台操作签署时的 IP   |    |    |         |  |                    |  |

示例：

```
{
  "contractId": "",
  "vars": {
    "甲方盖章": {
      "account": "",
      "signatureImageData": "base64 编码"
    },
    "甲方签署日期": {
      "account": ""
    }
  }
},
```

```
"tid": "",
"signIp": "",
"loginIp": "",
"loginTime": ""
}
```

响应 (RESPONSE) :

RESPONSE DATA 参数如下: 无

正常示例:

```
{
  "errno": 0,
  "cost": 533,
  "data": { },
  "errmsg": ""
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

#### 4.20.4、获取模版信息

说明: 使用此接口获取模版的所有信息, 包括: 文件编号、模版标题、模版变量等。

Request Path: **[sdk-host]**/template/getTemplate/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY) :

| 参数名 | 说明   | 是否必填 |  |
|-----|------|------|--|
| tid | 模版编号 | Y    |  |

示例:

```
{
  "tid": "111222333"
}
```

```
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

| 参数名          | 说明                 |
|--------------|--------------------|
| fid          | 模版文件编号             |
| title        | 模版标题               |
| description  | 模版描述               |
| templateVars | 模版变量, jsonArray 格式 |

正常示例：

```
{
  "errno": 0,
  "cost": 43,
  "data": {
    "template": {
      "fid": "876328732648726872",
      "templateVars": [
        {
          "isRequired": "1",
          "overflow": "0",
          "type": "50",
          "pageNum": "1",
          "fontFamily": "",
          "dateTimeFormat": "yyyy-MM-dd HH:mm:ss",
          "name": "stamp",
          "tagType": "0",
          "width": "0.2055485498108449",
          "x": "0.1450189155107188",
          "y": "0.18788958147818344",
          "fontSize": "18.7",
          "tag": "盖章",
          "maxLength": "20",
          "height": "0.12733748886910062",
          "warp": "true"
        },
        {
          "isRequired": "1",
          "overflow": "0",
          "type": "40",
          "pageNum": "2",
          "fontFamily": "",

```

```
"dateTimeFormat": "yyyy-MM-dd HH:mm:ss",
"name": "sign",
"tagType": "0",
"width": "0.06557377049180328",
"x": "0.1437578814627995",
"y": "0.03829029385574354",
"fontSize": "18.7",
"tag": "签名",
"maxLength": "20",
"height": "0.030276046304541407",
"warp": "true"
},
{
  "isRequired": "1",
  "overflow": "0",
  "type": "20",
  "pageNum": "2",
  "fontFamily": "",
  "dateTimeFormat": "yyyy-MM-dd HH:mm:ss",
  "name": "date",
  "tagType": "0",
  "width": "0.12610340479192939",
  "x": "0.7288776796973518",
  "y": "0.09884238646482636",
  "fontSize": "18.7",
  "tag": "签署日期",
  "maxLength": "20",
  "height": "0.02226179875333927",
  "warp": "true"
},
{
  "isRequired": "1",
  "overflow": "0",
  "type": "11",
  "pageNum": "3",
  "fontFamily": "SimSun",
  "dateTimeFormat": "yyyy-MM-dd HH:mm:ss",
  "name": "text1",
  "tagType": "0",
  "width": "0.2509457755359395",
  "x": "0.10844892812105927",
  "y": "0.07034728406055209",
  "fontSize": "18.7",
  "tag": "文本",
```

```
        "text": "",
        "maxLength": "20",
        "height": "0.03561887800534283",
        "warp": "true"
    },
    {
        "isRequired": "1",
        "overflow": "0",
        "type": "11",
        "pageNum": "3",
        "fontFamily": "SimSun",
        "dateTimeFormat": "yyyy-MM-dd HH:mm:ss",
        "name": "text2",
        "tagType": "0",
        "width": "0.2509457755359395",
        "x": "0.6620428751576293",
        "y": "0.21015138023152272",
        "fontSize": "18.7",
        "tag": "文本",
        "text": "",
        "maxLength": "20",
        "height": "0.05966162065894924",
        "warp": "true"
    }
],
"pageCount": 8,
"fileName": "xxxxxxxxx1201.pdf",
"developerId": "1852688952381669926",
"description": "未描述",
"titleHash": "fb1ae6b457f910a813fff6d0d1a6431a5b02ef17",
"title": "混合云-dzx-test",
"isDel": "0",
"tid": "198472374283748"
}
},
"errmsg": ""
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 53,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 4.20.5、用模版变量的手动签

说明：获取合同手动签署的页面 URL，将此 URL 发送给指定用户，即可在此页面进行手动签署操作完成签署。本接口根据模版变量定位签署位置。


Request Path: [sdk-host]/contract/sendByTemplate/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名        | 说明   | 是否必填 | 最大长度 | 默认值 |
|------------|--|------|------|-----|
| contractId | 合同编号   | Y    |      |     |
| signer     | 指定给哪个用户看   | Y    |      |     |
| dpi        | 预览图片清晰度，枚举值：96-低清（默认），120-普清，160-高清，240-超清   | N    | 3    | 96  |
| expireTime | <p>签署链接的过期时间，unix 时间戳。超过此时间则无法打开签署链接，需要获取新的签署链接。格式为秒级的 unix 时间戳，如希望“2017/12/30 10:21:52”到期，则设置为“1514600512”。</p> <p>注：并不是合同的可签署到期时间，只是此签署链接的有效期。</p> <p>由于返回的是短链接，如果没有设置则默认是 30 分钟，如果设置了以设置的为准，但有效期最大不能超过 7 天。</p> | N    | 20   | 0   |
| tid        | 模版 ID  | Y    |      |     |
| varNames   | <p>模板的变量名称，针对“模版”中的签署字段的变量名称，可以填多个。</p>  <p>变量名称获取方法有多种：</p> <p>①在开放平台编辑完模版后，将设置的变量名称记录下</p>                                   | Y    |      |     |



|                                |   |   |         |    |
|--------------------------------|---|---|---------|----|
|                                | <p>来；</p> <p>②用“4.20.0、获取模版变量”接口来获取，挑选出 response 中本次手动签需要用到的“type”为“50、40”的“name”字段值即需要使用的变量名称，拼接成字符串格式，不同变量名称之间用英文逗号间隔。日期类型的变量为“date”、“date1”、“date2”等 date 带流水号的格式。</p> <p>使用示例如下：</p> <p><b>"varNames": "甲方盖章 1,甲方盖章 2,date,date1"</b></p> |   |         |    |
| isAllowChangeSignaturePosition | <p>在有指定 signaturePOSTion 参数的情况下，是否允许拖动签名位置。取值 1/0。（0：不允许，1：允许）</p>  | N | 1       | 1  |
| returnUrl                      | <p>手动签署时，当用户签署完成后，签署页面指定回跳的页面地址，如果没填此参数则签署完成后跳转到合同预览页面。</p> <p>如果 returnUrl 带有“http”或“https”开头则判定 h5 跳转，如果 returnUrl 不带“http”或“https”开头则判定为微信小程序跳转。</p>   | N |         |    |
| vcodeMobile                    | <p>手动签署时指定接收手机验证码的手机号，如果需要采用手动签署页面校验短信验证码，则此项必填。</p> <p>如果手动签不需要短信校验，则此项可不填。</p> <p>自动签可不填此项。</p>   | N |         | 20 |
| isDrawSignatureImage           | <p>手动签署时是否手绘签名。取值 0/1/2。</p> <p>0 点击签名图片不会触发手写面板（禁止手写）。</p> <p>1 点击签名图片能触发手写面板（既可手写也可使用默认签名）。</p> <p>2 强制必须手绘签名（只能手写不允许使用默认签名）。</p>   | N | 1       | 1  |
| signatureImageName             | <p>签名/印章图片。取值 default 或者指定的印章名称。（<b>手动签都用默认的</b>）</p>   | N | default | 50 |
| pushUrl                        | <p>此处有配置则签署推送消息优先使用此配置，如果此处没有配置，则取开发者统一配置的异步推送地址</p>  | N |         |    |
| version                        | <p>手动签的版本，默认为 2，枚举值：</p> <p>2-v2 版本，</p> <p>3-v3 版本，UI 优化，移动端手绘面板自动横屏</p>   | N | 1       | 2  |
| isShowHandwrittenTime          | <p>枚举值 0/1，默认为 0</p> <p>1-手写面板及签名图片上显示手写签名操作的时间</p> <p>0-不显示该时间</p>   | N | 1       | 0  |
| isFaceAuth                     | <p>是否使用刷脸签。</p> <p>1-刷脸签，使用刷脸校验；</p> <p>0 或其他-非刷脸签，使用短信校验或无校验；</p>  | N |         |    |

示例:

```
{
  "contractId": "cU34344234",
  "expireTime": "0",
  "dpi": "120",
  "signer": "",
  "tid": "1271623948",
  "varNames": "name,name1,date,date1",
  "isAllowChangeSignaturePosition": "0",
  "returnUrl": "",
  "vcodeMobile": "",
  "isDrawSignatureImage": "1",
  "signatureImageName": "default",
  "pushUrl": "https://xxxxxxx"
}
```

响应 (RESPONSE) :

RESPONSE DATA 参数如下:

手动签署页面的 url, 签署者在此页面上进行签署操作。

正常示例:

```
{
  "errno": 0,
  "cost": 533,
  "errmsg": "",
  "data": {
    "shortUrl": "https://.....",
    "longUrl": "https://....."
  }
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

#### 4.20.6、获取创建模版的地址

说明: 使用此接口获取创建模版的页面地址, 用于创建开发者模版, 开发者可以向自己平台上的普通用户开放此页面。

Request Path: **[sdk-host]**/page/template/create/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY):

| 参数名     | 说明       | 是否必填 |  |
|---------|----------|------|--|
| account | 操作者的用户标识 | Y    | 表示开发者是把这个页面提供给哪个用户操作的, 这个用户标识必须已经调用“注册用户”接口在上上签系统里创建过, 否则这里会报“user not exists” |
| sid     | 流水号      | N    | 开发者自定义, 通常是开发者自己的业务流程中需要据此判断操作行为   |

示例:

```
{
  "account": "111222333",
  "sid": "38423785627382658"
}
```

响应 (RESPONSE):

RESPONSE DATA 参数如下:

| 参数名 | 说明                                |
|-----|-----------------------------------|
| url | 创建模版的页面地址, 开发者可以将此地址在自己平台上放开给用户使用 |

正常示例:

```
{
  "errno": 0,
  "cost": 233,
  "errmsg": "",
  "data": {
    "url": "https://xxxxxx"
  }
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 233,
```

```
"data": null,  
"errmsg": "check sign wrong "  
}
```

#### 4.20.7、获取编辑模版的地址

说明：使用此接口获取编辑模版的页面地址，用于编辑开发者模版，开发者可以向自己平台上的普通用户开放此页面。

Request Path: **[sdk-host]**/page/template/modify/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名     | 说明       | 是否必填 |  |
|---------|----------|------|--|
| tid     | 模版编号     | Y    | 从上上签开放平台的模版管理列表获取，或者通过“4.21.8、获取开发者模版列表”接口获取                                 |
| account | 操作者的用户标识 | Y    | 表示开发者是把这个页面提供给哪个用户操作的，这个用户标识必须已经调用“注册用户”接口在上上签系统里创建过，否则这里会报“user not exists” |
| sid     | 流水号      | N    | 开发者自定义，通常是开发者自己的业务流程中需要据此判断操作行为  |

示例：

```
{  
  "tid": "111222333",  
  "account": "mlimd@163.com",  
  "sid": "38423785627382658"  
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

| 参数名 | 说明                          |
|-----|-----------------------------|
| url | 编辑模版的页面地址，开发者可以将此地址在自己平台上放开 |

给用户使用

正常示例：

```
{
  "errno": 0,
  "errmsg": "",
  "data": {
    "url": "https://xxxxxx"
  }
}
```

异常示例：

```
{
  "errno": 240012,
  "cost": 233,
  "data": null,
  "errmsg": "check sign wrong "
}
```

#### 4.20.8、获取开发者模版列表

说明：使用此接口获取编辑模版的页面地址，用于编辑开发者模版。

Request Path: **[sdk-host]**/template/getTemplates/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名          | 说明      | 是否必填 |  |
|--------------|---------|------|--|
| categoryName | 类别名称    | Y    |  |
| pageSize     | 每页显示的条数 | Y    |  |
| pageNum      | 页数      | N    |  |

示例：

```
{
  "categoryName": "劳动合同",
  "pageSize": "20",
  "pageNum": "1"
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

| 参数名       | 说明  |
|-----------|---|
| templates | 模版列表字段， jsonArray 格式，主要包含以下字段：<br>"fid": "文件编号，无需关注"，<br>"developerId": "开发者编号，每个模版都从属于一个开发者，每个开发者仅能查询自己的模版，一般无需关注"，<br>"titleHash": "title 算出来的 hash，无需关注"，<br>"title": "模板名称"，<br>"isDel": "删除标识，0 是正常，已删除的模版会删除记录，无需关注"，<br>"tid": "模版编号"，<br>"categoryName": "模板分类" |

正常示例：

```
{
  "errno": 0,
  "cost": 233,
  "data": {
    "templates": [
      {
        "fid": "7066638189671621264",
        "developerId": "112",
        "titleHash": "07f485445a900b2cb13edd961fa283e69867eb",
        "title": "demo-title",
        "isDel": "0",
        "tid": "151264724401000001",
        "categoryName": ""
      }
    ]
  },
  "errmsg": ""
}
```

异常示例：

```
{
  "errno": 240012,
  "cost": 233,
  "data": null,
  "errmsg": "check sign wrong "
}
```

#### 4.20.9、预览模版

说明：使用此接口获取预览模版的页面地址，用于预览开发者模版的效果，开发者可以向自己平台上的普通用户开放此页面。

Request Path: **[sdk-host]**/page/template/preview/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数 (REQUEST BODY)：

| 参数名     | 说明       | 是否必填 |  |
|---------|----------|------|--|
| tid     | 模版编号     | Y    | 从上上签开放平台的模版管理列表获取，或者通过“4.21.8、获取开发者模版列表”接口获取                                 |
| account | 操作者的用户标识 | Y    | 表示开发者是把这个页面提供给哪个用户预览的，这个用户标识必须已经调用“注册用户”接口在上上签系统里创建过，否则这里会报“user not exists” |
| sid     | 业务流水号    | N    | 开发者自定义的预览流水号（比如第几次预览之类）  |

示例：

```
{
  "tid": "111222333",
  "account": "mlimd@163.com"
}
```

响应 (RESPONSE)：

RESPONSE DATA 参数如下：

| 参数名 | 说明                               |
|-----|----------------------------------|
| url | 预览模版的页面地址，开发者可以将此地址在自己平台上开放给用户使用 |

正常示例：

```
{
  "errno": 0,
  "errmsg": "",
}
```

```
"data": {  
  "url": "https://xxxxxx"  
}  
}
```

异常示例:

```
{  
  "errno": 240012,  
  "cost": 233,  
  "data": null,  
  "errmsg": "check sign wrong "  
}
```

#### 4.21、获取合同文件列表

说明: 获取在合同签署过程中, 产生的过程文件 ID 列表。

Request Path: [sdk-host]/contract/getFileMetaList/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY):

| 参数名        | 说明   | 是否必填 | 最大长度 |
|------------|------|------|------|
| contractId | 合同编号 | Y    | 20   |

示例:

```
{  
  "contractId": "1703466073594003458"  
}
```

响应 (RESPONSE):

RESPONSE DATA 参数如下:

| 参数名          | 说明                                  |            |
|--------------|-------------------------------------|------------|
| fileMetaList | 合同签署过程中的过程文件 meta 信息列表，JSONArray 格式 |            |
|              | 参数编码                                | 参数描述       |
|              | fhash                               | 文件 hash 值  |
|              | fpages                              | 文件总页数      |
|              | fsize                               | 文件大小，单位：字节 |



|  |              |                        |
|--|--------------|------------------------|
|  | <b>dhash</b> | 文件唯一 ID，保存在存储引擎中的文件 ID |
|  | <b>ftype</b> | 文件类型，一般均为 pdf          |
|  | <b>fname</b> | 文件名称                   |
|  | <b>fid</b>   | 每次签署产生的文件 ID           |

正常示例：

```
{
  "errno": 0,
  "cost": 233,
  "data": {
    "fileMetaList": [
      {
        "fid": 150500339001000000,
        "fname": "test.pdf",
        "ftype": "pdf",
        "dhash": "4d57c6b7032cfbc67878a9cabf8f397edebc1f38",
        "fsize": 182221,
        "fpages": 2,
        "fhash": "4d57c6b7032cfbc67878a9cabf8f397edebc1f38"
      }
    ]
  },
  "errmsg": ""
}
```

异常示例：

```
{
  "errno": 240006,
  "cost": 233,
  "data": null,
  "errmsg": "contractFileHistory is empty"
}
```

## 4.22、生成合同附页

说明：当合同结束后，可以调用此接口为合同生成一个附页文件，该附页文件是上上签提供的对合同从创建到签署完毕的整个签署过程的证明。

Request Path: [sdk-host]/contract/createAttachment

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名        | 说明   | 是否必填 |  |
|------------|------|------|--|
| contractId | 合同编号 | Y    |  |

示例：

```
{
  "contractId": "1703465339807805447"
}
```

响应（RESPONSE）：

正常示例：

```
{
  "errno": 0,
  "cost": 533,
  "data": { },
  "errmsg": ""
}
```

异常示例：

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 4.23、下载合同附页文件

说明：用此接口可以下载附页文件。

Request Path: [sdk-host]/contract/downloadAttachment

Method: GET

其他 url 参数：contractId

需要参与签名的其他 URL 参数：

| 参数名        | 说明   | 是否必填 |  |
|------------|------|------|--|
| contractId | 合同编号 | Y    |  |

请求参数（REQUEST BODY）：无

示例：

GET

```
/contract/downloadAttachment/?developerId=d445f5432&rtick=17364899234
&sign=a5334df21323&signType=rsa&contractId=d445f5432
```

响应 (RESPONSE) :

正常示例:

返回文件流

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

#### 4.24、关键字定位签署合同

说明：签署合同，可直接用于无感知的快捷签署。此接口要求签署者必须拥有自己的数字证书。可以无需先调用“添加签署者”接口与“设置合同签署参数”接口。

Request Path: [sdk-host]/contract/sign/keywords/

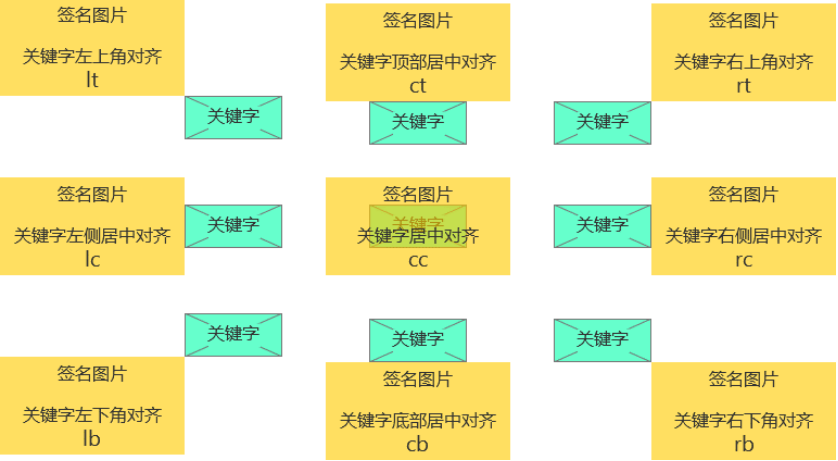
Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY) :

| 参数名           | 说明     | 是否必填 | 最大长度 |  |
|---------------|--------|------|------|--|
| contractId    | 合同编号   | Y    |      |  |
| signerAccount | 签署者    | Y    |      | 即签署者的 account  |
| keywords      | 关键字列表, | Y    |      | 需要添加签名位置的关键字列表, 支持多个关键字。内部以 String 数组进行拼接, 示例如下: ["张三","乙方"] |

|                     |          |   |  |
|---------------------|----------|---|--|
|                     | array 格式 |   |  |
| align               | 对齐方式     | N | <p>签名图片/印章图片与关键字的对齐方式，枚举值如下：</p> <p>lt 左上角对齐（图片位于关键字左上角）</p> <p>lc 左中对齐（图片位于关键字左侧上下居中）</p> <p>lb 左下对齐（图片位于关键字左下角）</p> <p>ct 中上对齐（图片位于关键字上方左右居中）</p> <p>cc 正中对齐（图片与关键字中心对齐）</p> <p>cb 中下对齐（图片位于关键字下方左右居中）</p> <p>rt 右上角对齐（图片位于关键字右上角）</p> <p>rc 右中对齐（图片位于关键字右侧上下居中）</p> <p>rb 右下对齐（图片位于关键字右侧下角）</p>  |
| signatureImageData  | 签名图片     | N | <p>用户指定的签名图片，Base64 字符串。允许为空，为空时使用用户默认的签名图片。</p> <p>使用自动签的用户，可以越过“2.1、创建用户签名/印章图片”接口、或“2.2、上传用户签名/印章图片”接口，直接在此提供签名/印章图片即可。</p>   |
| signatureImageName  | 签名图片名称   |   | <p>签名图片名称，取自【2.2 上传用户签名/印章图片】时设置的 imageName，适用于多签章场景，可在签署时灵活使用不同的签章。</p>   |
| signatureImageWidth | 图片显示宽度   | N | <p>本次签署使用的签名/印章图片在合同 PDF 上显示的宽度，72dpi 下的磅值。如果指定位置上没有设置签名图片的宽度和高度，那么按照如下默认值处理：</p> <p>个人签名：默认高度：48 磅，最大宽度：130 磅，宽度一般小于等于 130 磅。</p> <p>企业印章：默认宽度：130 磅，默认高度：130 磅。</p>  |

|                      |        |   |  |
|----------------------|--------|---|--|
|                      |        |   | <p>注意：signatureImageWidth 和 signatureImageHeight 必须同时为空，或者同时存在。</p> <p>同时为空则使用系统默认的高度和宽度（宽度和高度最大 130 磅，小于 130 磅的使用原始尺寸，大于 130 磅的按比例缩放）。</p> <p>如果同时有值，则图片按照两个宽高值拉伸/压缩填充，效果可能会产生变形。</p>   |
| signatureImageHeight | 图片显示高度 | N | <p>本次签署使用的签名/印章图片在合同 PDF 上显示的高度，72dpi 下的磅值。如果指定位置上没有设置签名图片的宽度和高度，那么按照如下默认值处理：</p> <p>个人签名：默认高度：48 磅，最大宽度：130 磅，宽度一般小于等于 130 磅。</p> <p>企业印章：默认宽度：130 磅，默认高度：130 磅。</p> <p>注意：signatureImageWidth 和 signatureImageHeight 必须同时为空，或者同时存在。</p> <p>同时为空则使用系统默认的高度和宽度（宽度和高度最大 130 磅，小于 130 磅的使用原始尺寸，大于 130 磅的按比例缩放）。</p> <p>如果同时有值，则图片按照两个宽高值拉伸/压缩填充，效果可能会产生变形。</p> |

示例：

```
{
  "contractId": "150347901101000001",
  "signerAccount": "13456833929",
  "keywords": [
    "张三",
    "乙方"
  ],
  "align": "cc",
  "signatureImageData": "",
  "signatureImageName": ""
}
```

响应（RESPONSE）：

正常示例：

```
{
  "errno": 0,
  "cost": 533,
  "data": { },
  "errmsg": ""
}
```

异常示例：

```
{
```

```
"errno": 240012,
"cost": 533,
"data": null,
"errmsg": "check sign wrong "
}
```

#### 4.25、在线验签（通过合同ID和哈希值）

说明：

提供上上签的合同编号、已签署完成的 PDF 文件哈希值，通过校验该哈希值与合同编号的匹配关系验证 PDF 文件是否有被篡改。

Request Path: **[sdk-host]**/contract/verifyContractFileHash

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名        | 说明    | 是否必填 | 最大长度 |                     |
|------------|-------|------|------|---------------------|
| contractId | 合同编号  | Y    |      | 通过上上签创建的合同编号        |
| fhash      | 文件哈希值 | Y    |      | 已签署完成的合同 PDF 文件的哈希值 |

示例：

```
{
  "contractId": "18724324823748723" ,
  "fhash": "ZHNmYXNnZHNnZHNhZ3Nn.."
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

| 参数名    | 说明   |  |
|--------|------|--|
| result | 验证结果 | 1-文档未被篡改<br>0-除未被篡改之外所有的结果都为 0（如一倍篡改、不是通过上上签签署完成的 PDF 文件等） |

正常示例：

```
{
  "errno": 0,
  "errmsg": "",
  "data": {
    "result": "1"
  }
}
```

```
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 4.26、获取存证页URL

说明: 通过此接口可以获取到存证页面的 url, 此页面表示该合同已在 上上签 保存。

- 此页面可联系 上上签 进行配置, 添加跳转链接 (比如仲裁网站), 默认跳转到 上上签 官网。
- 合同尚未结束时 (即还有签署者未签署完成, 没有调用过结束合同接口), 无法获取此页面 url。
- 如果签署者未申请过数字证书, 则页面上关于签署者的证件信息会显示不详。

Request Path: `[sdk-host]/contract/getApplyArbitrationURL/`

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY):

| 参数名        | 说明       | 是否必填 | 最大长度 | 默认值 |
|------------|----------|------|------|-----|
| contractId | 合同编号     | Y    |      |     |
| account    | 指定给哪个用户看 | Y    |      |     |

示例:

```
{
  "contractId": "cU34344234",
  "account": ""
}
```

响应 (RESPONSE):

RESPONSE DATA 参数如下:

合同存证页面的 url, 签署者在此页面上查看合同编号、签署者基本信息等。

正常示例:

```
{
  "errno": 0,
```

```
"cost": 533,
"errmsg": "",
"data": {
  "url": "https://....."
}
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 533,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 12、支付服务

### 12.1、微信支付

说明: 通过此接口可以获取微信支付的二维码, 通过展示二维码进行支付。

Request Path: [sdk-host]/pay/wechatPay/

Method: POST

其他 url 参数: 无

需要参与签名的其他 URL 参数: 无

请求参数 (REQUEST BODY):

| 参数名        | 说明                      | 是否必填 | 最大长度 | 默认值 |
|------------|-------------------------|------|------|-----|
| contractId | 合同编号, 表示这笔支付订单用于支付哪一份合同 | N    |      |     |

示例:

```
{
  "contractId": "cU34344234"
}
```

响应 (RESPONSE):

RESPONSE DATA 参数如下:

| 参数名             | 说明                                |
|-----------------|-----------------------------------|
| qrCodeImageData | 微信支付使用的二维码图片, Base64 编码, 需解码后展示。有 |



|         |                                    |
|---------|------------------------------------|
|         | 30 分钟的有效期，如果超过 30 分钟未支付则需重新获取二维码图片 |
| orderId | 支付订单编号                             |

正常示例：

```
{
  "errno": 0,
  "cost": 233,
  "errmsg": "",
  "data": {
    "qrCodeImageData": "xxxxxx",
    "orderId": "xxxxxx",
  }
}
```

异常示例：

```
{
  "errno": 240012,
  "cost": 233,
  "data": null,
  "errmsg": "check sign wrong "
}
```

## 12.1、查询支付订单

说明：通过支付订单编号查询支付结果。

Request Path: [sdk-host]/pay/getOnlinePayOrder/

Method: POST

其他 url 参数：无

需要参与签名的其他 URL 参数：无

请求参数（REQUEST BODY）：

| 参数名     | 说明     | 是否必填 | 最大长度 | 默认值 |
|---------|--------|------|------|-----|
| orderId | 支付订单编号 | Y    |      |     |

示例：

```
{
  "orderId": "2130366160674232326"
}
```

响应（RESPONSE）：

RESPONSE DATA 参数如下：

| 参数名              | 说明                          |  |
|------------------|-----------------------------|--|
| orderId          | 订单编号                        |  |
| paymentMode      | 该订单所使用的支付方式                 | WECHATPAY——微信支付<br>ALIPAY——支付宝<br>CHINAPAY——银联电子   |
| completeDateTime | 订单完成时间                      | 格式: yyyy-MM-dd HH:mm:ss  |
| orderAmount      | 订单支付的金额 (元)                 |  |
| payStatus        | 支付状态, 只有状态为“SUCCESS”时才是支付成功 | 如果是微信支付, 状态为:<br>SUCCESS<br>FAIL<br>UNPAID<br>PROCESSING<br>UNDO<br>SYSTEM_BUSY<br>UNDEFINED |

正常示例:

```
{
  "errno": 0,
  "errmsg": "",
  "data": {
    "orderId": "2130366160674232326",
    "paymentMode": "WECHATPAY",
    "completeDateTime": "2018-11-21 18:00:53",
    "orderAmount": "0.4",
    "payStatus": "SUCCESS"
  },
  "cost": 140
}
```

异常示例:

```
{
  "errno": 240012,
  "cost": 233,
  "data": null,
  "errmsg": "check sign wrong "
}
```