

OpenAPI 接口对接文档

0. 名词解释

超级管理员 (以下称为超管)	一家企业下仅允许存在一个超管并且仅限企业法定代表人担任，拥有企业账号的最高权限，包括但不限于管理企业账号下的所有数据（印章，合同等）
管理员	一家企业下仅允许存在一个管理员，拥有仅次于超管的读写权限，包括但不限于管理企业账号下的所有数据（印章，合同等）
对公打款	即向对公账户打款
企业开户行代码	即开户行联行号，是一个地区银行的唯一识别标志。

1. 接入指南

1.1 接入流程

电子牵 API 基于 HTTPS 协议，您可自行封装请求进行调用，以下对自行封装请求进行说明。

1.1.1 调用流程

生成签名 -> 封装请求 -> 发起请求 -> 得到响应

1.1.2 签名方式

a. 参数排序

对所有 API 请求参数（包括公共参数和业务参数，但除去 token 参数和 byte[]类型的参数），根据参数名称的 ASCII 码表的顺序排序。

如果参数的值为 Null OR None OR nil 或者为空字符串不参与签名；

如

foo=1, bar=2, foo_bar=3, foobar=

排序后的顺序是

bar=2, foo=1, foo_bar=3

注意

- 参数名区分大小写；
- 验证签名时，传送的 **token** 参数不参与签名，服务方会将生成的签名与该 **token** 值作校验。

b. 组装拼接

将排序好的参数名和参数值拼装在一起，根据上面的示例得到的结果为：bar=2foo=1foo_bar=3

在参数后拼接 PATH，如 path=/api/get/得到，对于/open-api/{account}/的 path，需要替换{account}后，再进行拼接：

bar=2foo=1foo_bar=3/api/get/

将上述字符串用 UTF-8 编码后，进行 MD5，得到十六进制小写字符串

30a18746b778761ccd061c34c3c57744

将 MD5 得到的字符串, 拼接 appSecret (由电子牵分配) 得到 MD5secret 字符串, 在这里以 appSecret=e123 为例说明:

拼接结果: 30a18746b778761ccd061c34c3c57744e123

对 MD5secret 进行 sha1, 获取最后 token

5fe5dfe214a6f7a4e2accccdcd84346e49492449

1.1.3 发起请求

- **POST** 方式提交的请求, 使用 **body** 传输参数, 内容使用 **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。
- 如果出现 **POST** 请求中需要上传文件, 需要用 **multipart/form-data** 的形式传输

1.1.4 基础参数

- 所有的接口请求中, 必须包含以下基础参数。后续参数列表中不再包含如下参数:

参数名称	参数说明	是否必须	数据类型	schema
timestamp	调用时间戳, 精确到毫秒	TRUE	long	

token	签名, 签名方式见主页	TRUE	String	
appCode	appCode, 由服务提供方分配	TRUE	String	
version	版本, 目前可选值: v1	TRUE	String	

1.2 请求示例

1.2.1 python 示例

```
# python demo

# -*- coding: utf-8 -*-

import hashlib

import time

import requests

def get_token(param_dict, secret, path):

    params_str = ""

    keys = param_dict.keys()

    keys.sort()

    for k in keys:

        # 空串 None 不参与计算

        if param_dict.get(k) == 0 or (param_dict.get(k) is not None and

param_dict.get(k) != ""):

            if isinstance(param_dict.get(k), baseString):

                params_str += (k + "=" + param_dict.get(k))

            else:
```

```
        params_str += (k + "=" + str(param_dict.get(k)))

params_str += path

md5 = hashlib.md5(params_str).hexdigest()

sign_key = md5 + secret

sign = hashlib.sha1(sign_key).hexdigest()

return sign

def get_url():

    path = "/saas-api/signature/image"

    url = "https://domain" + PATH

    param_dict = {

        # 业务 AppCode

        "appCode": "appCode",

        "timestamp": int(time.time() * 1000),

        "signatureCode": 1,

        "operatorUserCode": 1,

        "accountCode": 1,

        "transactionCode": 1,

        "signTypeLimits": "1",

        "contractCode": "1",

        "entityName": "1",

        "userCode": "1",
```

```
    "entityType": 1,
    "signValidMethod": 2,
    "version": "v1"
}

token = get_token(param_dict, "secret", path)

param_dict["token"] = token

s = ""

for k, v in param_dict.items():
    s = s + k + "=" + str(v) + "&"

u = s[:-1]

url = url + "?" + u

print url

resp = requests.get(url)

print resp.json()
```

```
def post_file():

    path = "/open-api/contract/opt/uploadTemplate"

    url = "https://domain" + path

    param_dict = {

        # 业务 AppCode

        "appCode": "appCode",

        "timestamp": int(time.time() * 1000),
```

```

        "companyOpenCode": 1,

        "personOpenCode": 2,

        "version": "v1"
    }

    with open("测试.pdf", "r") as contractFile:
        - files = {'templateFile': (test.pdf, contractFile,
'application/octet-stream', {'Expires': '0'})}

        token = get_token(param_dict, "secret", path)

        param_dict["token"] = token

        resp = requests.post(url, params=param_dict, files=files)

        print resp.json()

def post_url():

    path = "/open-api/contract/sign/applyForSign"

    url = "https://domain" + path

    param_dict = {"companyOpenCode": "1",

                  "contractCode": "1",

                  "personOpenCode": "1",

                  "signTypeLimits": "",

                  "signValidMethod": "",

                  # 业务 AppCode

                  "appCode": "appCode",

```

```

        "version": "v1",
        "timestamp": int(time.time() * 1000)}

header = {
    "Content-Type": "application/json"
}

token = get_token(param_dict, "secret", path)

param_dict["token"] = token

resp = requests.post(url, json=param_dict, headers=header)

print resp.json()

if __name__ == "__main__":
    post_file()

```

1.2.2 Java 示例

```

import com.google.common.collect.Maps;

import org.apache.commons.codec.digest.DigestUtils;

import org.apache.commons.lang.StringUtils;

import java.util.*;

public class WebApiTokenUtil {

    public static String signature(Map<String, String> params, String secret,
String path) {

        List<String> list = new ArrayList<String>();

```



```
Iterator<Map.Entry<String, String>> iter =
params.entrySet().iterator();

while (iter.hasNext()) {

    Map.Entry<String, String> entry = iter.next();

    String key = entry.getKey();

    String val = entry.getValue();

    if (StringUtils.isBlank(val)) {

        continue;

    }

    //(1)connecting all parameters

    list.add(key + "=" + val);

}

//(2)sort all Strings

Collections.sort(list);

StringBuilder buf = new StringBuilder();

for (String s : list) {

    buf.append(s);

}

buf.append(path);

//(3) md5

String md5 = DigestUtils.md5Hex(buf.toString());

String waitSign = md5 + secret;
```

```

        //(4) sha1

        String sign = DigestUtils.sha1Hex(waitSign);

        return sign;
    }

    public static void main(String[] args) {

        Map<String, String> map = Maps.newHashMap();

        map.put("name", "张三");

        map.put("age", "18");

        String secret = "secret";

        String uri = "/open-api/{account}";

        String path = uri.replace("{account}", "123");

        System.out.println("path:" + path);

        System.out.println("sign:" + signature(map, secret, path));

    }
}

```

1.2.3 Golang 示例

```

import (
    "bytes"
    "crypto/md5"
    "crypto/sha1"
    "encoding/hex"
    "fmt"

```

```
"io"

"log"

"mime/multipart"

"net/http"

"os"

"path/filepath"

"sort"

"time"
)

// 计算签名

func signature(params map[String]String, secret String, path String) String {

    var keys []String

    for k := range params {

        keys = append(keys, k)

    }

    sort.Strings(keys)

    var paramsStr String

    for _, k := range keys {

        paramsStr += k + "=" + params[k]

    }

    paramsStr += path

    fmt.Println("paramStr:", paramsStr)
```

```

md5Client := md5.New()

md5Client.Write([]byte(paramsStr))

md5Str := hex.EncodeToString(md5Client.Sum(nil))

fmt.Println("md5:", md5Str)

waitSign := md5Str + secret

fmt.Println("waitSign:", waitSign)

sha1Client := sha1.New()

sha1Client.Write([]byte(waitSign))

sign := hex.EncodeToString(sha1Client.Sum(nil))

return sign
}

// 请求示例

func do() {

    // 构建参数

    params := map[String]string{

        "appCode": "appCode",

        // 时间戳, ms

        "timestamp":      fmt.Sprintf("%d", time.Now().UnixNano()/1e6),

        "companyOpenCode": "1",

        "personOpenCode": "2",

        // 版本, 目前只有 v1

        "version": "v1",

```

```
}  
  
domain := "https://domain"  
  
path := "/open-api/contract/opt/uploadTemplate"  
  
// 密钥  
  
secret := "secret"  
  
// 计算签名  
  
sign := signature(params, secret, path)  
  
params["token"] = sign  
  
fmt.Println(params)  
  
req, _ := NewFileUploadRequest(domain+path, "filepath", params)  
  
// 请求  
  
client := &http.Client{}  
  
resp, err := client.Do(req)  
  
if err != nil {  
    fmt.Printf("error to request to the server:%s\n", err.Error())  
  
    return  
}  
  
body := &bytes.Buffer{}  
  
_, err = body.ReadFrom(resp.Body)  
  
if err != nil {  
    log.Fatal(err)  
}  
}
```

```
defer resp.Body.Close()

fmt.Println(body)
}

// 构建文件上传请求

func NewFileUploadRequest(url, filePath string, params map[string]string)
(*http.Request, error) {

    // 校验文件是否存在

    file, err := os.Open(filePath)

    if err != nil {

        return nil, err

    }

    defer file.Close()

    body := &bytes.Buffer{}

    // 文件写入 body

    writer := multipart.NewWriter(body)

    part, err := writer.CreateFormFile("templateFile", filepath.Base(filePath))

    if err != nil {

        return nil, err

    }

    _, err = io.Copy(part, file)

    // 其他参数列表写入 body

    for k, v := range params {
```

```

        if err := writer.WriteField(k, v); err != nil {
            return nil, err
        }
    }

    if err := writer.Close(); err != nil {
        return nil, err
    }

    req, err := http.NewRequest(http.MethodPost, url, body)

    if err != nil {
        return nil, err
    }

    req.Header.Add("Content-Type", writer.FormDataContentType())

    return req, err
}

```

1.2.4 php 示例

```
// PHP 上传文件 demo
```

```
<?php
```

```
$PATH = "/open-api/contract/opt/uploadTemplate";
```

```
$URL = "https://domain" . $PATH;
```

```
$HEADER = [
```

```
    "Content-Type:multipart/form-data",
```

```
];
```

```
/**
 * 计算签名
 * @param $data
 * @param $path
 * @return String
 */
function makeSign($data, $path)
{
    // 业务方密钥
    $appSecret = 'secret';

    //签名步骤一：按字典序排序参数
    ksort($data);

    $String = toUrlParams($data, $path);

    echo "md5:" . $String;

    echo "\n";

    //签名步骤二：拼接 secret
    $String = $String . $appSecret;

    //签名步骤三：sha1 加密
    $result = sha1($String);

    echo "sha1:" . $result;

    echo "\n";

    return $result;
}
```



```
}
```

```
/**
```

```
* @param $data
```

```
* @param $path
```

```
* @return String
```

```
*/
```

```
function toUrlParams($data, $path)
```

```
{
```

```
    $buff = "";
```

```
    foreach ($data as $k => $v) {
```

```
        if ($k != "sign" && $v != "" && !is_array($v)) {
```

```
            $buff .= $k . "=" . $v;
```

```
        }
```

```
    }
```

```
    $buff .= $path;
```

```
    echo "buf" . $buff;
```

```
    echo "\n";
```

```
    return md5($buff);
```

```
}
```

```
/**
```

```
* @param $path
```

```
* @return array
*/

function buildParam($path)
{
    $param = array(
        "companyOpenCode" => "1",
        "contractCode" => "1",
        "personOpenCode" => "1",
        "signTypeLimits" => "",
        "signValidMethod" => "",
        // 业务方 AppCode
        "appCode" => "appCode",
        "version" => "v1",
        "timestamp" => time());

    $token = makeSign($param, $path);

    $f = curl_file_create("filepath");

    $param["templateFile"] = $f;

    $param["token"] = $token;

    print_r($param);

    return $param;
}

/**
```

```

* 发送请求

* @param $param

* @param $header

* @param $url

*/

function doCurl($param, $header, $url)
{
    $ch = curl_init();

    curl_setopt($ch, CURLOPT_URL, $url);

    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

    curl_setopt($ch, CURLOPT_POST, 1);

    curl_setopt($ch, CURLOPT_POSTFIELDS, $param);

    curl_setopt($ch, CURLOPT_HTTPHEADER, $header);

    $output = curl_exec($ch);

    curl_close($ch);

    echo $output;
}

$param = buildParam($PATH);

doCurl($param, $HEADER, $URL);

```

1.3 返回值结构

```

{
    "code": 0,                # 返回 Code

```

```
"description": "success", # 返回结果描述
"data": ……., # 返回数据域, 可以是多种数据形式
"logId": "XXXXXX" # 标识请求唯一 ID, 方便问题排查
}
```

1.4 返回 Code 说明

1.5 接入时序图

签署方

第一步：创建个人/企业账号（以个人为例） 1、通过页面链接创建 2、通过接口创建

1、通过页面链接创建

创建个人账户

返回认证链接



2. 接口清单

2.1 个人注册认证

2.1.1 获取个人认证链接 (H5)

接口地址 `/open-api/certification/getCertUrl`

请求方式 `POST`

接口请求 `Content-Type` `["application/json"]`

接口描述

- 对接方调用本接口，用于获取个人认证页面链接，提供给期望被认证的用户。用户在电子牵个人认证页面上完成认证流程，生成电子牵个人账号。
- 用户通过页面认证通过后，电子牵系统会替被认证的用户生成一个默认手写签章，签章内容为被认证人的姓名。
- 对接方可以在调用本接口时，传入 **refRedirectUrl**、**refAsyncNotifyUrl**，用于通过回调形式接收认证完成回调信息，以及控制用户认证成功后页面跳转地址

请求参数

请求参数样例

```
{
```

```
"timestamp": 1598449320956,
"token": "d9deaa1ef14a4941ad095948d8547bd6",
"appCode": "E784329069023",
"version": "v1",

"personOpenCode": "PTX3655339",
"refCode": "3f88c0e6456e4773a070d0428a711e49",
"assignCertFlows": "1,3,4",
"personName": "张三",
"personIdCard": "450127198901012271",
"personMobile": "15720170000",
"refRedirectUrl": "https://www.github.com",
"refAsyncNotifyUrl": "https://www.github.com/callback/certification"
}
```

返回结果 data 参数

返回结果示例

```
{
  "code": 0,
  "description": "success",
  "data": {
```



```
    "url":  
    "https://domain.com/certificate?urlCode=7897080231ukjhekwqejlkkj",  
    "urlCode": "7897080231ukjhekwqejlkkj"  
    "refCode": "7897080231ukjhekwqejlkkj"  
    "certStatus": 2  
  },  
  "logId": "202006191051250100110690763828C44"  
}
```

认证通过后回调内容:

请求方式 POST

接口请求 Content-Type ["application/json"]

请求体:

```
{  
  "refCode": "7897080231ukjhekwqejlkkj",  
  "certStatus": 1  
}
```

2.1.2 个人手机号注册实名

接口地址 /open-api/certification/noUser/person3/

请求方式 POST

接口请求 Content-Type ["multipart/form-data"]

接口描述

- 对接方调用本接口，通过提供姓名、身份证号、手机号进行信息校验，校验通过后生成电子牵个人账号。
- 认证通过后，电子牵系统会替被认证的用户生成一个默认手写签章，签章内容为被认证人的姓名。

请求参数

请求参数样例

```
{  
  "timestamp": 1598449320956,  
  "token": "d9deaa1ef14a4941ad095948d8547bd6",  
  "appCode": "E784329069023",  
  "version": "v1",  
  
  "personOpenCode": "PTX3655339",  
  "refCode": "3f88c0e6456e4773a070d0428a711e49",  
  "assignCertFlows": "1,3,4",  
  "personName": "张三",  
  "personIdCard": "450127198901012271",  
  "personMobile": "15720170000",  
  "personIdCardFrontFile": FileObject,
```

```
"personIdCardBackFile": FileObject
}
```

返回结果 data 参数

返回结果示例

```
{
  "code": 0,
  "description": "success",
  "data": false,
  "logId": "202006191051250100110690763828C44"
}
```

2.2 企业注册认证

2.2.1 企业法人注册

接口地址 /open-api/certification/generateCorporate

请求方式 POST

接口请求 Content-Type ["multipart/form-data"]

接口描述

- 对接方调用本接口，通过企业法人认证的方式创建企业账号，此接口只是提交信息初审，初审通过会进行对公打款，

对接方还需要调用**回填校验金额接口 (2.2.5)** 校验对公打款金额方可完成企业法人认证

- 创建企业账号前需要先经 **2.1** 个人注册认证接口创建一个个人账号并通过本接口传入，企业认证完成后该个人自动为企业的超级管理员
- 完成认证创建账户的同时会生成一个默认的企业印章
- 注册认证完成后，会根据 **certCallbackUrl** 回调业务方，通知认证成功

请求参数

请求参数样例

```
{  
  "timestamp": 1598449320956,  
  "token": "d9deaa1ef14a4941ad095948d8547bd6",  
  "appCode": "E784329069023",  
  "version": "v1",  
  
  "personOpenCode": "PTX3655339",  
  "companyOpenCode": "110111000011",  
  "entName": "xxx 科技有限公司",
```

```
"entQualificationType": 1,
"entQualificationNum": "11010781001111",
"corporateName": "李四",
"corporateIdCard": "450127198901012271",
"entLicenseFile": FileObject,
"certCallBackUrl": "https://www.github.com/callback",
"entAccountName": "李四",
"entBankCard": "621226600000000000",
"entBankName": "中国工商银行北京市花园路支行",
"entBankCode": "102400000000"
}
```

返回结果 data 参数

返回结果示例

```
{
  "code": 0,
  "description": "success",
  "data": "06f140f76cfd4a2f9f0fa12a1355d523",
  "logId": "7807e0d3a9de4b47b3bb3e7e9a43efff"
}
```

企业法人整体认证流程通过后回调内容:

请求方式 POST

接口请求 Content-Type ["application/json"]

请求体:

```
{  
    "appCode":"appcode1",  
    "companyOpenCode":"companyOpenCode1"  
    "personOpenCode":"personOpenCode1",  
    "result":true,  
    "verifyId":"6857773053535847176"  
}
```

2.2.2 企业代理人注册

接口地址 /open-api/certification/generateAgent

请求方式 POST

接口请求 Content-Type ["multipart/form-data"]

接口描述

- 对接方调用本接口, 通过企业代理人认证的方式创建企业账号, 此接口只是提交信息初审, 对接方还需要调用**提交授权书接口 (2.2.4)** 提交线下盖章后的授权书和**回填校验金额接口 (2.2.5)** 校验对公打款金额方可完成企业代理人认证

- 创建企业账号前需要先创建一个个人账号并通过本接口传入，企业认证完成后该个人会成为企业的管理员
- 完成认证创建账户的同时会生成一个默认的企业印章
- 注册认证完成后，会根据 **certCallbackUrl** 回调业务方，通知认证成功

请求参数

请求参数样例

```
{  
  "timestamp": 1598449320956,  
  "token": "d9deaa1ef14a4941ad095948d8547bd6",  
  "appCode": "E784329069023",  
  "version": "v1",  
  
  "personOpenCode": "PTX3655339",  
  "companyOpenCode": "110111000011",  
  "entName": "xxx 科技有限公司",  
  "entQualificationType": 1,  
  "entQualificationNum": "11010781001111",  
  "corporateName": "李四",  
  "corporateIdCard": "450127198901012271",  
  "entLicenseFile": FileObject,
```

```
"entAccountName": "李四",
"entBankCard": "6212266000000000000",
"entBankName": "中国工商银行北京市花园路支行",
"entBankCode": "102400000000",
"roleType": 2,
"certCallBackUrl": "https://www.github.com/callback"
}
```

返回结果 data 参数

返回结果示例

```
{
  "code": 0,
  "description": "success",
  "data": {
    "verifyId": "06f140f76cfd4a2f9f0fa12a1355d523",
    "authLicence": "aaa"
  }
  "logId": "7807e0d3a9de4b47b3bb3e7e9a43efff"
}
```

企业代理人整体认证流程通过后回调内容:

请求方式 POST

接口请求 Content-Type ["application/json"]

请求体:

```
{  
  "appCode": "appcode1",  
  "companyOpenCode": "companyOpenCode1",  
  "personOpenCode": "personOpenCode1",  
  "result": true,  
  "verifyId": "6857773053535847176"  
}
```

2.2.3 个体工商户注册

接口地址 /open-api/certification/generateIndividualBiz

请求方式 POST

接口请求 Content-Type ["multipart/form-data"]

接口描述

- 对接方调用本接口，通过个体工商户认证的方式创建企业账号
- 创建企业账号前需要先创建一个个人账号并通过本接口传入，企业认证完成后该个人自动为企业的超级管理员

- 完成认证创建账户的同时会生成一个默认的企业印章
- 注册认证完成后，会根据 **certCallbackUrl** 回调业务方，通知认证成功

请求参数

请求参数样例

```
{  
    "timestamp": 1598449320956,  
    "token": "d9deaa1ef14a4941ad095948d8547bd6",  
    "appCode": "E784329069023",  
    "version": "v1",  
  
    "personOpenCode": "PTX3655339",  
    "companyOpenCode": "110111000011",  
    "entName": "xxx 科技有限公司",  
    "entQualificationType": 1,  
    "entQualificationNum": "11010781001111",  
    "corporateName": "李四",  
    "corporateIdCard": "450127198901012271",  
    "entLicenseFile": FileObject,  
    "certCallbackUrl": "https://www.github.com/callback",  
}
```

返回结果 data 参数

返回结果示例

```
{  
  "code": 0,  
  "description": "success",  
  "data": false,  
  "logId": "202006191051250100110690763828C44"  
}
```

个体工商户整体认证流程通过后回调内容:

请求方式 POST

接口请求 Content-Type ["application/json"]

请求体:

```
{  
  "appCode": "appcode1",  
  "companyOpenCode": "companyOpenCode1"  
  "personOpenCode": "personOpenCode1",  
  "result": true,  
  "verifyId": "6857773053535847176"  
}
```

2.2.4 提交授权书

接口地址 /open-api/certification/uploadAudit

请求方式 POST

接口请求 Content-Type ["multipart/form-data"]

接口描述

- 通过企业代理人方式认证需要调用此接口。对接方调用本接口，将**企业代理人注册认证接口 (2.2.2)** 返回授权书的经线下盖章后图片提交至电子牵系统进行审核
- 创建企业账号前需要先创建一个个人账号并通过本接口传入，企业认证完成后该个人自动为企业的超级管理员
- 完成认证创建账户的同时会生成一个默认的企业印章

请求参数

请求参数样例

```
{  
  "timestamp": 1598449320956,  
  "token": "d9deaa1ef14a4941ad095948d8547bd6",  
  "appCode": "E784329069023",  
  "version": "v1",
```

```
"verifyId": "06f140f76cfd4a2f9f0fa12a1355d523",  
"authLicenseFile": FileObject,  
"callBackUrl": "https://www.github.com/callback"  
}
```

返回结果 data 参数

返回结果示例

```
{  
  "code": 0,  
  "description": "success",  
  "data": "06f140f76cfd4a2f9f0fa12a1355d523",  
  "logId": "7807e0d3a9de4b47b3bb3e7e9a43efff"  
}
```

个体工商户整体认证流程通过后回调内容:

请求方式 POST

接口请求 Content-Type ["application/json"]

请求体:

```
{  
  "rejections":  
    {
```

```
"Applicant":["个人三要素身份证号错误"],
"digitalApplication":["企业营业执照识别错误"],
"Subject":["对公打款账号错误"]
},
"result":false,
"verifyId":"6857773053535847176"
}
```

2.2.5 回填验证金额

接口地址 /open-api/certification/payCheckVerify

请求方式 POST

接口请求 Content-Type ["application/json"]

接口描述

- 对接方调用此接口，传入 **2.2.2** 接口所提交对公账号实际收到的打款金额，电子牵系统校验回填金额与打款金额是否一致，两次错误后会判定为认证不通过，需要重新走认证流程
- 同一个 **verifyId** 最多验证 **2** 次金额，若 **2** 次都验证失败，则认证流程整体失败

- 回填金额需要在申请打款且款项实际下发成功后 **3** 天之内执行，否则回填过期，需要重新申请打款

请求参数

请求参数样例

```
{  
  "timestamp": 1598449320956,  
  "token": "d9deaa1ef14a4941ad095948d8547bd6",  
  "appCode": "E784329069023",  
  "version": "v1",  
  
  "verifyId": "06f140f76cfd4a2f9f0fa12a1355d523true",  
  "amount": "0.01"  
}
```

返回结果 data 参数

返回结果示例

```
{  
  "code": 0,  
  "description": "success",  
  "data": false,
```

```
"logId": "202006191051250100110690763828C44"  
}
```

2.3 文件签署相关

2.3.1 文件上传

接口地址 /open-api/contract/opt/upload

请求方式 POST

接口请求 Content-Type ["multipart/form-data"]

接口描述

- 调用接口上传文件，上传成功后方可调用手动签署/自动签署等接口
- 仅支持 **PDF** 格式的文件
- 文件不超过 **20M**

请求参数

请求参数样例

```
{  
  "timestamp": 1598449320956,  
  "token": "d9deaa1ef14a4941ad095948d8547bd6",  
  "appCode": "E784329069023",  
  "version": "v1",
```



```
"companyOpenCode": "110111000011",  
"personOpenCode": "PTX3655339",  
"contractFile": FileObject,  
"signType": 1,  
"canRefuse": 1  
}
```

响应状态

返回结果 data 参数

schema 属性说明

上传文件返回参数对象

返回结果示例

```
{  
  "code": 0,  
  "data": {  
    "contractCode": ""  
  },  
  "description": ""  
}
```

```
    "logId": ""  
  }  
}
```

2.3.2 模版上传

接口地址 /open-api/contract/opt/uploadTemplate

请求方式 POST

接口请求 Content-Type ["multipart/form-data"]

接口描述

- 调用接口上传模版文件，上传成功后方可调用“使用模版进行合同合成”接口
- 仅支持 **PDF** 格式的文件
- 文件大小不超过 **20M**

请求参数

请求参数样例

```
{  
  "timestamp": 1598449320956,  
  "token": "d9deaa1ef14a4941ad095948d8547bd6",  
  "appCode": "E784329069023",  
  "version": "v1",  
}
```

```
"companyOpenCode": "110111000011",
"personOpenCode": "PTX3655339",
"templateFile": FileObject
}
```

返回结果 data 参数

schema 属性说明

上传模板返回对象

返回结果示例

```
{
  "code": 0,
  "data": {
    "contractTemplateId": ""
  },
  "description": "",
  "logId": ""
}
```

2.3.3 使用模板组装文件

接口地址 /open-api/contract/opt/generate

请求方式 POST

接口请求 Content-Type ["application/json"]

接口描述

- 调用接口进行文件组装，组装成功后自动进行文件上传，无需另外调用文件上传接口

请求参数

无法复制加载中的内容

请求参数样例

```
{  
  "timestamp": 1598449320956,  
  "token": "d9deaa1ef14a4941ad095948d8547bd6",  
  "appCode": "E784329069023",  
  "version": "v1",  
  
  "companyOpenCode": "110111000011",  
  "personOpenCode": "PTX3655339",  
  "params": {"jiafang": "阿莱克斯"},  
  "fontType": "0",  
  "fontSize": "10.5",
```

```
"signType": 1,  
"canRefuse": 1  
}
```

返回结果 data 参数

schema 属性说明

上传文件返回参数对象

返回结果示例

```
{  
  "code": 0,  
  "data": {  
    "contractCode": ""  
  },  
  "description": "",  
  "logId": ""  
}
```

2.3.4 手动签署

接口地址 /open-api/contract/sign/applyForSign

请求方式 POST

接口请求 Content-Type ["application/json"]

接口描述

- 该接口为页面接口，调用方可以在业务系统嵌入该接口链接，引导客户至电子牵提供的页面进行文件签署。用户进入该签约地址 **URL**，页面会自动加载对应的 **PC** 版本或 **HTML5** 版本。用户可以在电子牵的签署页面进行签署操作。
- 手动签署有两次回调（回调接口提供给对接人配置）：
 1. 签署前回调(可选)：用于签署前回调判断该用户当前是否可签署，当用户在签署页面填写完签署验证码后触发。
 1. 签署成功回调(必填)：用于签署成功后的内容通知，在签署成功后触发。

请求参数

无法复制加载中的内容

请求参数样例

```
{  
    "timestamp": 1598449320956,
```

```
"token": "d9deaa1ef14a4941ad095948d8547bd6",
"appCode": "E784329069023",
"version": "v1",

"companyOpenCode": "110111000011",
"personOpenCode": "PTX3655339",
"contractCode": "38983a254c544481840e905bbb2cfd89",
"keyWord": "盖章处",
"needVerification": "0",
"returnUrl": "https://www.github.com/afterSign",
"scanPage": "1, 2, 5",
"signCoordinate":
{"pdfPage": "1", "vertical": "2168", "horizontal": "1405"} & {"pdfPage": "2", "vertical": "3056", "horizontal": "1192"},
"signTypeLimits": "0",
"signValidMethod": 3,
"transactionCode": "d2268106f45347ca80057946e08fdb6e"
}
```

响应状态

返回结果 data 参数

schema 属性说明

手动签返回参数对象

返回结果示例

```
{  
  "code": 0,  
  "data": {  
    "signUrl": "",  
    "transactionCode": ""  
  },  
  "description": "",  
  "logId": ""  
}
```

2.3.5 自动签署

接口地址 /open-api/contract/sign/autoSign

请求方式 POST

接口请求 Content-Type ["application/json"]

接口描述

- 调用接口进行自动签署 (无需在页面进行验证码等身份校验即可签署)
- 仅限调用方所属的、已认证过的公司主体使用, 其他企业/个人若需使用, 请联系项目经理

请求参数

请求参数样例

```
{  
  "timestamp": 1598449320956,  
  "token": "d9deaalef14a4941ad095948d8547bd6",  
  "appCode": "E784329069023",  
  "version": "v1",  
  
  "companyOpenCode": "110111000011",  
  "personOpenCode": "PTX3655339",  
  "contractCode": "38983a254c544481840e905bbb2cfd89",  
  "keyWord": "盖章处",  
  "scanPage": "1, 2, 5",  
  "signCoordinate":  
  [{"pdfPage": "1", "vertical": "2168", "horizontal": "1405"} & {"pdfPage": "2", "vertical": "3056", "horizontal": "1192"}],
```

```
"signCrossPage": false,  
"signatureCode": 3,  
"transactionCode": "d2268106f45347ca80057946e08fdb6e"  
}
```

响应状态

返回结果 data 参数

schema 属性说明

自动签署返回参数对象

返回结果示例

```
{  
  "code": 0,  
  "data": {  
    "transactionCode": ""  
  },  
  "description": "",  
  "logId": ""  
}
```

2.3.6 文件归档

接口地址 /open-api/contract/opt/archive

请求方式 POST

接口请求 Content-Type ["application/json"]

接口描述

- 文件中所有签署方均签署完成后，需要调用本接口进行文件归档
- 归档后电子牵会把该文件相关的操作记录进行归档存证，且该文件不能被继续操作（手动签署、自动签署、文件撤销等操作均无法进行）

请求参数

请求参数样例

```
{  
  "timestamp": 1598449320956,  
  "token": "d9deaa1ef14a4941ad095948d8547bd6",  
  "appCode": "E784329069023",  
  "version": "v1",  
  
  "contractCode": "38983a254c544481840e905bbb2cfd89",  
  "companyOpenCode": "110111000011",  
  "personOpenCode": "PTX3655339"  
}
```

响应状态

返回结果 data 参数

返回结果示例

```
{  
  "code": 0,  
  "data": {},  
  "description": "",  
  "logId": ""  
}
```

2.3.7 文件撤销

接口地址 /open-api/contract/opt/inactive

请求方式 POST

接口请求 Content-Type ["application/json"]

接口描述

- 文件中所有签署方均未签署的情况下，可以调用本接口进行文件撤销
- 文件被撤销后，手动签署、自动签署、文件归档等操作均无法进行

- 文件被撤销后，打开之前已生成的签署链接，无法查看文件内容，会提示“文件已被撤销，请联系发起方”

请求参数

请求参数样例

```
{  
  "timestamp": 1598449320956,  
  "token": "d9deaa1ef14a4941ad095948d8547bd6",  
  "appCode": "E784329069023",  
  "version": "v1",  
  
  "contractCode": "38983a254c544481840e905bbb2cfd89",  
  "companyOpenCode": "110111000011",  
  "personOpenCode": "PTX3655339"  
}
```

响应状态

返回结果 data 参数

返回结果示例

```
{  
  "code": 0,  
  "data": {},  
}
```

```
"description": "",  
"logId": ""  
}
```

3. 常见 QA 签署相关

- 上传文件的接口怎么总是报系统异常?

原因可能有以下几种:

- 使用的 **Content-Type** 是否正确, 应该使用 **multipart/form-data**
- 文件内容是否在 **form** 中传了过来, 可以查看日志判断
- 参数是否通过 **body** 传过来, 可以通过日志判断
- 正确的传入方式应该是参数在 **form** 中传输, 文件内容在 **body** 中传输。

- 手动签署接口 (**applyForSign**) 中 **transactionCode** 是做什么的, 怎么传?

- 用于标识唯一一次签署，可以做幂等使用。在回调的时候，回调的内容中会有此数据，标识此次回调是哪次签署的结果
- 建议使用 "服务方名称 (英文或者英文简称) +uuid"，长度 **32-128** 位
- **applyForSign** 接口中 **keyWord** 和 **signCoordinate** 是做什么的，怎么传？
 - **keyWord** 和 **signCoordinate** 至少传入一个
 - **keyWord** 是指文档中的关键字，可以通过关键字寻找盖章位置
 - **signCoordinate** 是直接指定签署的坐标
 - 两个同时传入的时候 **keyWord** 起作用