联合信任电子证据固化保全系统
自动网页取证服务对外接口
说明文档

Version 1.0

www.TSA.cn

保密事宜： 本文档包含北京联合信任技术服务有限公司的专有商业信息和保密信息。接受方同意维护本文档所提供信息的保密性，承诺不对其进行复制，或向评估小组以外、非直接相关的人员公开此信息。对于以下三种信息，接受方不向北京联合信任技术服务有限公司承担保密责任：

1 ） 接受方在接收该文档前，已经掌握的信息。

2 ） 可以通过与接受方无关的其它渠道公开获得的信息。

3 ） 可以从第三方，以无附加保密要求方式获得的信息。

目录

1. 前言

《联合信任电子证据固化保全系统》由北京联合信任技术服务有限公司独立部署并运营，用于对电子数据进行取证、固化和保全，证明所获取电子证据的客观存在性和内容完整性，以解决当事人取证困难、维权成本高、司法认定复杂的问题。通过该系统可以客观真实的还原事实真相，及时固定证据、防止证据灭失，证据勘验简单快捷。《联合信任电子证据固化保全系统》提供的自动网页取证接口服务，用于申请者通过接口方式实现对网页证据进行取证、固化和保全。本文档为自动网页取证服务的对外接口说明文档。

2. 接入说明

2.1 接入前的准备

接入者需要先与时间戳服务中心签订合作协议，通过注册成为《联合信任电子证据固化保全系统》（http://evidence.tsa.cn）的用户，自动网页取证接口服务由接入者自行在《联合信任电子证据固化保全系统》上开通。

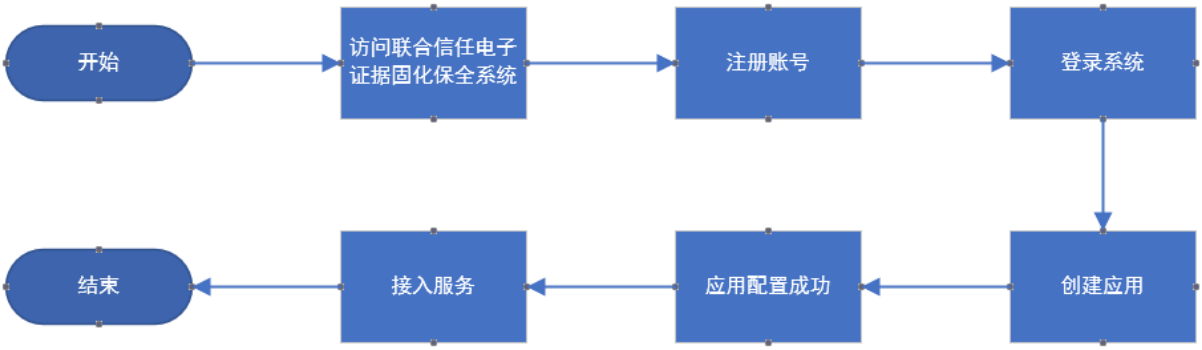
2.2 接入方式

HTTP 方式

2.3 接入 URL

服务器地址：

2.4 接入流程



3. 自动网页取证服务

3.1 申请取证接口（grab）

请求 URL：http://ip:port/gateway
请求方法：POST
请求头：Content-Type: application/json;charset=utf-8
请求参数：appId, method, applicant, url, remark, notifyUrl

参数	说明	类型	长度	是否必填
appId	应用 ID，创建应用时提供	String	32	是
method	方法	String	32	是
sign	参数签名,签名算法	String	255	是
applicant	申请人	String	255	是
url	取证网址	String	无限制	是
remark	证据备注说明	String	50	否
notifyUrl	回调地址	String	255	是

返回参数：

参数	说明	类型
code	返回码	String
seriaNo	证据唯一编号	String

message	返回信息	String
sign	返回参数签名	String

3.2 取消证据接口(cancel)

请求 URL：http://ip:port/gateway
请求方法：POST
请求头：Content-Type: application/json;charset=utf-8
请求参数：appId、method、serialNo

参数	说明	类型	长度	是否必填
appId	应用 ID，创建应用时提供	String	32	是
method	方法	String	32	是
serialNo	序列号	String	32	是
sign	参数签名,签名算法	String	255	是

返回参数：

参数	说明	类型
code	代码	String
message	返回信息	String
sign	返回参数签名	String

3.3 获取证据预览地址接口(preview)

请求 URL：http://ip:port/gateway
请求方法：POST
请求头：Content-Type: application/json;charset=utf-8
请求参数：appId、method、serialNo

参数	说明	类型	长度	是否必填
appId	应用 ID，创建应用时提供	String	32	是
method	方法	String	32	是
serialNo	序列号	String	32	是
sign	参数签名,签名算法	String	255	是

返回参数：

参数	说明	类型
code	代码	String

message	返回信息	String
sign	返回参数签名	String
preview	预览地址	String

3.4 证据确认接口(confirm)

请求 URL：http://ip:port/gateway
请求方法：POST
请求头：Content-Type: application/json;charset=utf-8
请求参数：appId、method、serialNo

参数	说明	类型	长度	是否必填
appId	应用 ID，创建应用时提供	String	32	是
method	方法	String	32	是
sign	参数签名,签名算法	String	255	是
serialNo	序列号	String	32	是

返回参数：

参数	说明	类型
code	代码	String
message	返回信息	String
sign	返回参数签名	String

3.5 放弃证据接口(giveUp)

请求 URL：http://ip:port/gateway
请求方法：POST
请求头：Content-Type: application/json;charset=utf-8
请求参数：appId、method、serialNo

参数	说明	类型	长度	是否必填
appId	应用 ID，创建应用时提供	String	32	是
method	方法	String	32	是
sign	参数签名,签名算法	String	255	是
serialNo	序列号	String	32	是

返回参数：

参数	说明	类型

code	代码	String
message	返回信息	String
sign	返回参数签名	String

3.6 下载证据接口(download)

请求 URL：http://ip:port/gateway
请求方法：POST
请求头：Content-Type: application/json;charset=utf-8
请求参数：appId、method、serialNo

参数	说明	类型	长度	是否必填
appId	应用 ID，创建应用时提供	String	32	是
method	方法	String	32	是
sign	参数签名,签名算法	String	255	是
serialNo	序列号	String	32	是

返回参数：

参数	说明	类型
code	代码	String
message	返回信息	String
sign	返回参数签名	String
download	下载地址	String

3.7 删除证据接口(delete)

请求 URL：http://ip:port/gateway
请求方法：POST
请求头：Content-Type: application/json;charset=utf-8
请求参数：appId、method、serialNo

参数	说明	类型	长度	是否必填
appId	应用 ID，创建应用时提供	String	32	是
method	方法	String	32	是
sign	参数签名,签名算法	String	255	是
serialNo	序列号	String	32	是

返回参数：

参数	说明	类型
code	代码	String
message	返回信息	String
sign	返回参数签名	String

3.8 查看证据状态接口(status)

请求 URL：http://ip:port/gateway
请求方法：POST
请求头：Content-Type: application/json;charset=utf-8
请求参数：appId、method、serialNo

参数	说明	类型	长度	是否必填
appId	应用 ID	String	32	是
method	方法	String	32	是
sign	参数签名,签名算法	String	255	是
serialNo	序列号	String	32	是

返回参数：

参数	说明	类型
code	代码	String
message	返回信息	String
sign	返回参数签名	String
status	任务状态	String

任务状态：

参数	说明
0	任务初始化
1	任务抓取中
2	抓取成功
3	抓取失败
4	证据待确认
5	证据已确认
6	证据已放弃
7	证据已删除

4. 附录

4.1 返回码

代码	说明
N0000	成功
N0001	参数不合法
N0002	方法未授权
N0003	应用不存在/应用被停用
N0004	错误
N0005	任务取证进行中
N0006	任务未确认，请先确认
N0007	证书文件不存在
N0008	预览文件不存在
N0009	任务不存在
N0010	取消、放弃、删除失败
N0011	次数归还失败

4.2 签名算法

签名生成的步骤如下：

第一步：将所有请求参数按照参数名 ASCII 码从小到大排序（字典序），使用 URL 键值对的格式（即 key1=value1&key2=value2…）拼接成字符串 stringA。

第二步：使用私钥对拼接的字符串 stringA 进行加密。

4.3 示例

封装请求参数到 TreeMap 中利用 TreeMap 有序特性对参数进行排序做签名。（详情见 demo）

签名验签需要公私钥

```
private static String publicKey = "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCh5Nk2GLiyQFMIU+h3OEA4UeFbu3dCH5
private static String privateKey = "MIICWwIBAAKBgQCh5Nk2GLiyQFMIU+h3OEA4UeFbu3dCH5sjd/sLTxxvwjXq7JLq\n" +
        "Jbt2rCIdzpAXOi4jL+FRGQnHaxUlHUBZsojnCcHvhrz2knV6rXNogt0emL7f7ZMR\n" +
        "o8IsQGV8mlKIC9xLnlOQQdRNUssmrROrCG99wpTRRNZjOmLvkcoXdeuaCQIDAQAB\n" +
        "AoGAUTcJ1H6QYTOts9bMHsrERLymzir8R9qtLBzrfp/gRxxpigHGLdph8cWmk8dl\n" +
        "N5HDRXmmkdV6t2S7xdOnzZen31lcWe0bIzg0SrFiUEOtg3Lwxzw2Pz0dKwg4ZUoo\n" +
        "GKpcIU6kEpbC2UkjBV4+2E6P1DXuhdgTyHoUA3ycxOdjCAUCQQCyjTzGPXFoHq5T\n" +
```

申请取证方法：

```java
public static String grab(String grabUrl, String applicant, String notifyUrl) throws IOException {
    Security.addProvider(new BouncyCastleProvider());
    TreeMap<String, String> params = new TreeMap<String, String>();
    params.put("appId", appId);
    params.put("method", "grab");
    params.put("url", grabUrl);
    params.put("applicant", applicant);
    params.put("notifyUrl", notifyUrl);
    String sign = sign(params, privateKey);
    params.put("sign", sign);
    CloseableHttpClient client = HttpClientBuilder.create().build();
    HttpPost post = new HttpPost(url);
    post.setEntity(new StringEntity(JSON.toJSONString(params), "UTF-8"));
    CloseableHttpResponse response = client.execute(post);
    return EntityUtils.toString(response.getEntity(), "UTF-8");
}
```

参数签名方法：

```java
public static String sign(TreeMap<String, String> param, String privateKeyStr) {
    try {
        PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new PKCS8EncodedKeySpec(Base64.decodeBase64(privateKeyStr));
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        PrivateKey privateKey = keyFactory.generatePrivate(pkcs8EncodedKeySpec); // 用key工厂对象生成私钥
        Signature signature = Signature.getInstance("SHA1WithRSA");  //  md5 RSA签名对象
        signature.initSign(privateKey);  //初始化签名
        signature.update(joinParam(param).getBytes());
        byte[] result = signature.sign();  //对消息进行签名
        return new String(Base64.encodeBase64String(result));
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

返回参数验签方法：

```java
public static Boolean verify(TreeMap<String, String> param, String publicKeyStr) {
    try {
        X509EncodedKeySpec x509EncodedKeySpec = new X509EncodedKeySpec(Base64.decodeBase64(publicKeyStr));
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        PublicKey publicKey = keyFactory.generatePublic(x509EncodedKeySpec);
        Signature signature = Signature.getInstance("SHA1WithRSA");  //  md5 RSA签名对象
        signature.initVerify(publicKey);
        String sign = param.get("sign");
        param.remove("sign");
        signature.update(joinParam(param).getBytes());
        return signature.verify(Base64.decodeBase64(sign));
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
```

拼接参数方法：

```
private static String joinParam(TreeMap<String, String> param) {
    StringBuffer paramStr = new StringBuffer();
    for (String key : param.keySet()) {
        paramStr.append(key).append("=").append(param.get(key)).append("&");
    }
    return paramStr.substring(0, paramStr.length() - 1);
}
```