|  |
| --- |
|  |

**法人用户单点登录组件接口规范**

浙江省数据管理中心目 录

[第一章 概述 2](#_Toc515916836)

[1.1 对接说明 2](#_Toc515916837)

[1.2 用户等级 2](#_Toc515916838)

[1.2.1 非实名用户 3](#_Toc515916839)

[1.2.2 用户名密码认证用户 3](#_Toc515916840)

[1.2.3 数字证书认证和电子营业执照认证用户 3](#_Toc515916841)

[1.3 敏感信息前台展现要求 3](#_Toc515916842)

[第二章 单点登录规范 3](#_Toc515916843)

[2.1 单点登录流程 4](#_Toc515916844)

[2.2 单点登录地址 4](#_Toc515916845)

[2.3 回调地址 5](#_Toc515916846)

[2.3.1 功能说明 5](#_Toc515916847)

[2.3.2 回调地址Url 6](#_Toc515916848)

[2.3.3 回调方式：POST 6](#_Toc515916849)

[2.3.4 回调参数 6](#_Toc515916850)

[2.4 验证令牌并获取用户的登录信息 7](#_Toc515916851)

[2.4.1 功能说明 7](#_Toc515916852)

[2.4.2 接口类型 7](#_Toc515916853)

[2.4.3 请求方式 7](#_Toc515916854)

[2.4.4 接口调用地址 7](#_Toc515916855)

[2.4.5 请求头信息设置 7](#_Toc515916856)

[2.4.6 输入参数 8](#_Toc515916857)

[2.4.7 返回参数 8](#_Toc515916858)

[第三章 对接联系方式 10](#_Toc515916859)

[附表1：企业（法人）字段信息说明 11](#_Toc515916861)

# 概述

本文档介绍了应用和系统接入浙江政务服务网企业（法人）用户认证体系涉及的对接流程、用户授权信息获取以及一些注意事项。

## 对接说明

接入单位在对接时须向浙江政务服务网提出接入申请，接入单位需要去IRS里的组件目录，搜索 政务服务网法人单点登录 点击立即申请进行提交对接申请审批。

接入单位在对接时，需注意用户信息的数据类接口必须由服务端进行调用，用户信息由应用服务端获取后若需在应用前端页面展示必须按照1.2敏感信息前台展现要求 进行数据脱敏处理。

## 用户等级

根据认证程度不同，将浙江政务服务网企业（法人）用户分为三个等级：非实名用户、用户名密码认证用户、数字证书认证用户以及电子营业执照认证用户，接入系统按照各自实际情况决定不同等级的用户的网上办事权限。

### 非实名用户

realLevel返回值为4时表示为非实名企业（法人）用户。

### 用户名密码认证用户

realLevel返回值为2时表示为现场用户名密码认证企业（法人）用户。

realLevel返回值为3时表示为在线用户名密码认证企业（法人）用户。

### 数字证书认证和电子营业执照认证用户

realLevel返回值为1时表示为数字证书认证或者电子营业执照认证企业（法人）用户。

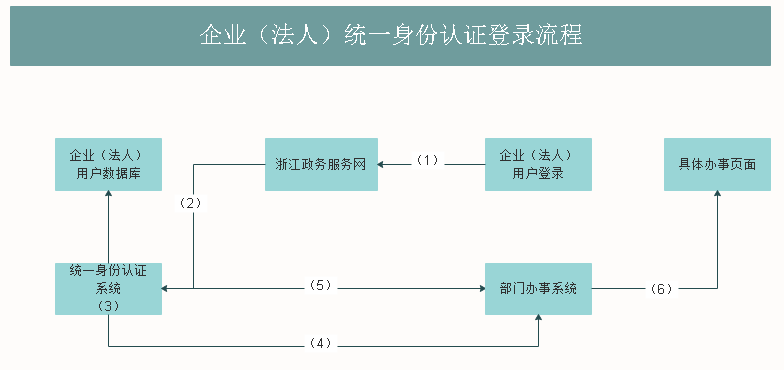
## 敏感信息前台展现要求

详见《附件11、浙江政务服务网用户敏感信息脱敏处理规范》文档。

# 单点登录规范

接入应用服务中需要使用企业（法人）用户实名身份信息进行相关查询和办理业务时，由应用服务在获得用户授权的基础上，通过可信认证平台完成安全互信过程和身份信息获取，最大限度确保业务查询和办理请求是由用户本人发起。

## 单点登录流程



① 用户登录浙江政务服务网后点击办事事项链接；

② 政务服务网重定向至法人统一认证系统；

③ 统一身份认证系统检查是否存在当前用户的会话，如果存在，获取其ssotoken；若不存在，创建会话并生成ssotoken；

④ 统一身份认证系统重定向至部门办事系统法人信息回调地址并推送ssotoken和ssoticket；

⑤ 部门办事系统调用“浙江省组件超市RESTful接口”接口，采用Https接口与法人统一身份认证系统交互，传入ssotoken和请求头信息获取用户登录信息。法人统一身份认证系统验证ssotoken有效性，验证通过后返回用户登录信息；

⑥ 成功获取法人信息，完成登录，跳转到具体办事页面。

## 单点登录地址

应用提供单位可根据需求在具体业务环节发起单点登录，通过单点登录规则拼接授权页面链接引导用户打开授权。

单点登录地址规则如下：

正式环境：

<https://esso.zjzwfw.gov.cn/opensso/spsaehandler/metaAlias/sp?spappurl>=回调地址

测试环境：  
<http://essotest.zjzwfw.gov.cn/opensso/spsaehandler/metaAlias/sp?spappurl>=回调地址

单点登录地址请求方式：Get[http://www.baidu.com/](http://www.baidu.com)

spappurl参数说明：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **说明** | **是否必填** | **数据类型** | **长度（字节）** |
| spappurl | 回调地址用于接收ssotoken的信息。 | 是 | String | 无限制 |

附注：

用户单点登录后展示具体的办事页面地址，由spappurl后跟着的回调地址获取到法人信息后，后台重定向到业务办理地址（前后端分离也可实现），如果后台不知道跳到哪个办理地址，可以通过goto参数带过去。后台重定向示例：



前端页面跳转地址示例：

浙江政务服务网单点登录参考链接（假设回调地址为http://localhost:8080/ssocheck.jsp，登录后希望跳转到[www.tsign.cn](http://www.tsign.cn)页面）：

正式环境：

<https://esso.zjzwfw.gov.cn/opensso/spsaehandler/metaAlias/sp?spappurl=http://localhost:8080/ssocheck.jsp?goto=https://www.tsign.cn>  
测试环境：

<http://essotest.zjzwfw.gov.cn/opensso/spsaehandler/metaAlias/sp?spappurl=http://localhost:8080/ssocheck.jsp?goto=https://www.tsign.cn>[http://www.baidu.com/](http://www.baidu.com)

## 回调地址

### 功能说明

该地址是实现业务系统开发回调地址接收SSO传递过来的令牌及票据，以便之后实现页面系统登录功能。

### 回调地址Url

业务系统自定义

### 回调方式：POST

### 回调参数

Request参数列表:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **说明** | **是否必填** | **数据类型** | **长度（字节）** |
| ssoticket | 票据，24分钟内使用。过期后失效。 | 是 | String | 无限制 |
| ssotoken | 令牌，25分钟内使用。过期后失效。 | 是 | String | 无限制 |

获取令牌及票据的方法示例：

// 获取ssotoken令牌

request.getParameter("ssotoken");

// 获取ssoticket票据

request.getParameter("ssoticket")

附注：

ssotoken是用于获取登录信息的临时令牌，有效时间为25分钟。

票据是在跨session，跨域的应用系统接入时候用到，如通过

https://esso.zjzwfw.gov.cn/opensso/spsaehandler/metaAlias/sp?spappurl=应用系统法人登录入口&ssoticket=法人登录的票据，则可以实现跨session跨域的单点登录；

## 验证令牌并获取用户的登录信息

### 接口功能说明

用于浙江省组件超市RESTful接口接口请求调用，获取对应数据资源。页面集成类组件调用时，需将请求头参数作为普通请求参数进行调用。

### 接口类型

https

### 请求方式

post

### 接口调用地址

政务外网环境接口地址：

https://bcdsg.zj.gov.cn:8443/restapi/prod/IC33000020220309000001/rest/user/query  
互联网环境接口地址：  
https://ibcdsg.zj.gov.cn:8443/restapi/prod/IC33000020220309000001/rest/user/query

### API鉴权参数说明

常见问题：**获取调用组件的**AccessKey**&**SecretKey**，需要先到组件超市进行组件使用申请，待组件使用申请审核通过后可以通过页面查看**AccessKey**&**SecretKey**。**

**请求业务协同和数据共享网关的每个HTTP请求都必须在请求头（HTTP Header）中设置如下4个参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **说明** | **类型** | **是否必填** |
| X-BG-HMAC-SIGNATURE | API输入参数签名结果 | String | 是 |
| X-BG-HMAC-ALGORITHM | 签名的摘要算法，当前仅支持hmac-sha256。 | String | 是 |
| X-BG-HMAC-ACCESS-KEY | 分配给应用的accessKey，例如：12345678。 | String | 是 |
| X-BG-DATE-TIME | 时间戳，时区为GMT+8，格式为：Tue, 09 Nov 2021 08:49:20 GMT。API服务端允许客户端请求最大时间误差为100秒。 | String | 是 |

其中X-BG-HMAC-SIGNATURE的计算公式为：

|  |
| --- |
| signature = HMAC-SHA256-HEX(secret\_key,signing\_string) |

各字段解释如下：

1. secret\_key为接口申请完成后获取到的secret\_key
2. signing\_string由请求方法、URI、请求参数等拼接获得，具体如下：

|  |
| --- |
| HTTP\_METHOD+\n+HTTP\_URI+\n+QUERY\_STREAM+\n+X-BG-HMAC-ACCESS-KEY+\n+X-BG-DATE |

参数解释如下图，详细代码可参见附录2签名计算代码

|  |  |
| --- | --- |
| **参数名** | **说明** |
| HTTP Method | 指 HTTP 协议中定义的 GET、PUT、POST 等请求方法，必须使用全大写的形式。 |
| HTTP URI | 请求路径，要求必须以“/”开头，不以“/”开头的需要补充上，空路径为“/” |
| X-BG-DATE | 请求头中的 Date （ GMT 格式 ）格式为：“Tue, 09 Nov 2021 08:49:20 GMT” |
| QUERY\_STREAM | 是对于 URL 中的 query（ query 即 URL 中?后面的 key1=valve1&key2=valve2 字符串）进行编码后的结果。以 key 按照字典顺序（ ASCII 码由小到大）排序，并使用 & 符号连接起来，生成相应的query\_string。 |

### 2.4.6输入参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 说明 | 是否必填 | 数据类型 | 长度（字节） |
| token | sso令牌 | 是 | String | 无限制 |

附注：POST请求参数类型为application/json类型

请求Body示例：

{"token":"12f5e1c6-883b-40fd-a55f-42c22424eb3f"}

### 返回参数

| **参数名** | **是否必填** | **类型** | **说明** |
| --- | --- | --- | --- |
| message | 是 | string | 错误信息 |
| data | 是 | object | 业务数据 |

data返回参数说明

其中业务数据详细返回参数详见源接口，页面集成类接口返回形式包括图片或HTML代码等，可以利用浏览器对HTML进行渲染，或联系组件开发商进行了解使用。

成功返回示例

{

      "message": "请求成功!",

    "data": null,

   }

失败返回示例

{

      "message": "请求成功!",

    "data": null,

   }  
data返回参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 说明 | 数据类型 | 长度（字节） |
| errCode | 结果：  0：成功  1410020:接口权限校验不通过  1410001:参数无效  1410006:参数为空  1410020:接口权限校验不通过  1410025:Token不存在  1410036:TOKEN无效  1410026:Token已过期 |  |  |
| msg | 错误说明 | String |  |
| info | 企业（法人）信息（详见附表2） | Map |  |

Info企业（法人）信息示例：（附件中有参数说明）



## 法人登出地址

登出地址：

http://esso.zjzwfw.gov.cn/opensso/UI/Logout?goto=https://oauth.zjzwfw.gov.cn/oauth/logout.do?redirect=登录地址

## 第三方业务系统重定向到业务办理地址规范要求

第三方业务系统获取到企业信息后，重定向到业务办理地址时，重定向的过程需要遵守以下几点规范要求：

1允许重定向到办理地址过程中有http和https的302跳转过程，也可以不跳转。

2不允许办理地址跳转过程中出现表单页跳转到办理页面的情况。

# 对接联系方式

对接联系人：杭州天谷 杨文超：+86-18238996477

# 附表1：企业（法人）字段信息说明

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名称** | **字段名** | **数据类型及格式** | **说明** |
| 工商注册号 | CompanyRegNumber | VARCHAR(25) | 非工商注册法人为空字符串 |
| 机构代码号 | OrganizationNumber | VARCHAR(15) | 可能为空 |
| 法人名称（企业或机构名称） | CompanyName | VARCHAR(100) | 必有项 |
| 实体类型 | OrgType | VARCHAR(2) | 取值说明如下：  1:企业、农专社、个体工商户；  2:其他非法人单位；  4：事业单位；  5：行政单位；  6：社会团体；  7：民办非企业法人；  8：基金会；  9：养老机构；  11：律师事务所 |
| 法人类型 | CompanyType | VARCHAR(15) | 工商注册企业为企业类型，如：有限责任公司；  非工商注册企业为法人类型，如：机关法人。 |
| 注册地址 | CompanyAddress | VARCHAR(100) | 非工商企业为组织机构代码证上的地址。 |
| 法人电话号码 | telephoneNumber | VARCHAR(15) | 保留，为空字符串 |
| 经营范围 | CompanyScope | VARCHAR(100) | 非工商注册法人为空字符串 |
| 登记机构（中文） | CompanyRegOrg | VARCHAR(30) | 非工商注册法人为空字符串 |
| 工商内部序号 | CompanySerialNumber | VARCHAR(15) | 非工商注册法人为空字符串 |
| 用户ID，唯一标识 | userId | VARCHAR(30) | 政务服务网账户唯一标识 |
| 用户名 | username | VARCHAR(30) | 政务服务网用户名 |
| 证书名称 | certName | VARCHAR(200) | 非证书登录时为空字符串 |
| 证书内部序号 | certNbxh | VARCHAR(30) | 非证书登录时为空字符串 |
| 证书注册号 | certRegNo | VARCHAR(30) | 非证书登录时为空字符串 |
| 证书编号 | certHxbh | VARCHAR(30) | 非证书登录时为空字符串 |
| 十六进制数字证书序列号 | certSn | VARCHAR(30) | 非证书登录时为空字符串 |
| 十进制数字证书序列号 | certDecSn | VARCHAR(30) | 非证书登录时为空字符串 |
| 证书内容BASE64 | certBase64 | VARCHAR(4048) | 非证书登录时为空字符串 |
| 登录类型 | loginType | VARCHAR（30） | Cert：数字证书方式登录，  Password：用户名密码方式登录  Elicense：电子营业执照方式登录 |
| 实名认证等级 | realLevel | VARCHAR（2） | 取值为数字，说明如下：  1：第一级数字证书和电子营业执照  2：第二级现场用户名密码实名认证  3：第三级在线用户名密码实名认证  4：第四级用户名密码非实名认证 |
| 行政区划编码 | xzqh | VARCHAR(50) | 可能为空 |
| 统一社会信用代码 | uniscid | VARCHAR(50) | 可能为空 |
| 法定代表人/负责人 | CompanyLegRep | VARCHAR(50) | 可能为空 |
| 法人状态 | status | VARCHAR(10) | 企业法人有效，正常为空 |
| 经办人手机号 | attnPhone | VARCHAR(50) | 经办人手机号  （数字证书用户登录时为空） |
| 经办人姓名 | attnName | VARCHAR(100) | 联系人姓名  （数字证书用户登录时为空） |
| 经办人证件类型 | attnIDType | VARCHAR(10) | 证件类型，51：身份证（老身份证类型）（数字证书用户登录时可能为空）  111：身份证（新身份证类型）；  414：普通护照；  516：港澳居民来往内地通行证；  511：台湾居民来往内地通行证；  553：外国人永久居留身份证 |
| 经办人身份证件号 | attnIDNo | VARCHAR（50） | 联系人身份证件号（数字证书用户登录时可能为空） |
| 经办人固定电话 | attnLandLinePhone | VARCHAR（50） | 联系人固定电话（可能为空） |

# 附表2：网关API接口签名计算代码

|  |
| --- |
| import javafx.util.Pair; import lombok.extern.slf4j.Slf4j;  import javax.crypto.Mac; import javax.crypto.spec.SecretKeySpec; import javax.xml.bind.DatatypeConverter; import java.io.UnsupportedEncodingException; import java.net.URI; import java.net.URL; import java.net.URLEncoder; import java.text.DateFormat; import java.text.SimpleDateFormat; import java.util.\*; import java.util.stream.Collectors;  @Slf4j public class HmacAuthUtil {   /\*\*  \* 构造请求 header  \* @param urlStr 请求url，全路径格式，比如：https://bcdsg.zj.gov.cn/api/p/v1/user.get  \* @param requestMethod 请求方法,大写格式，如：GET, POST  \* @param accessKey 应用的 AK  \* @param secretKey 应用的 SK  \* @return  \*/  public static Map<String, String> generateHeader(String urlStr,String requestMethod, String accessKey, String secretKey) {  log.info("params,urlStr={},requestMethod={},accessKey={},secretKey={}",urlStr,requestMethod,accessKey,secretKey);  Map<String, String> header = new HashMap<>();  try {  DateFormat dateFormat = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z", Locale.US);  dateFormat.setTimeZone(TimeZone.getTimeZone("GMT"));  String date = dateFormat.format(new Date());  URL url = new URL(urlStr);  URI uri = new URI(url.getProtocol(), url.getHost(), url.getPath(), url.getQuery(), null);   String canonicalQueryString = getCanonicalQueryString(uri.getQuery());   String message = requestMethod.toUpperCase() + "\n" + uri.getPath() + "\n" + canonicalQueryString + "\n" + accessKey + "\n" + date + "\n";   Mac hasher = Mac.getInstance("HmacSHA256");  hasher.init(new SecretKeySpec(secretKey.getBytes(), "HmacSHA256"));   byte[] hash = hasher.doFinal(message.getBytes());   // to lowercase hexits  DatatypeConverter.printHexBinary(hash);   // to base64  String sign = DatatypeConverter.printBase64Binary(hash);  header.put("X-BG-HMAC-SIGNATURE", sign);  header.put("X-BG-HMAC-ALGORITHM", "hmac-sha256");  header.put("X-BG-HMAC-ACCESS-KEY", accessKey);  header.put("X-BG-DATE-TIME", date);  } catch (Exception e) {  log.error("generate error",e);  throw new RuntimeException("generate header error");  }  log.info("header info,{}",header);  return header;  }   private static String getCanonicalQueryString(String query) {  if (query == null || query.trim().length() == 0) {  return "";  }  List<Pair<String, String>> queryParamList = new ArrayList<>();  String[] params = query.split("&");  for (String param : params) {  int eqIndex = param.indexOf("=");  String key = param.substring(0, eqIndex);  String value = param.substring(eqIndex+1);  Pair<String, String> pair = new Pair<String, String>(key,value);  queryParamList.add(pair);  }   List<Pair<String, String>> sortedParamList = queryParamList.stream().sorted(Comparator.comparing(param -> param.getKey() + "=" + Optional.ofNullable(param.getValue()).orElse(""))).collect(Collectors.toList());  List<Pair<String, String>> encodeParamList = new ArrayList<>();  sortedParamList.stream().forEach(param -> {  try {  String key = URLEncoder.encode(param.getKey(), "utf-8");  String value = URLEncoder.encode(Optional.ofNullable(param.getValue()).orElse(""), "utf-8")  .replaceAll("\\%2B","%20")  .replaceAll("\\+","%20")  .replaceAll("\\%21","!")  .replaceAll("\\%27","'")  .replaceAll("\\%28","(")  .replaceAll("\\%29",")")  .replaceAll("\\%7E","~")  .replaceAll("\\%25","%")  ;  encodeParamList.add(new Pair<>(key, value));  } catch (UnsupportedEncodingException e) {  throw new RuntimeException("encoding error");  }  });  StringBuilder queryParamString = new StringBuilder(64);  for (Pair<String, String> encodeParam : encodeParamList) {  queryParamString.append(encodeParam.getKey()).append("=").append(Optional.ofNullable(encodeParam.getValue()).orElse(""));  queryParamString.append("&");  }   return queryParamString.substring(0, queryParamString.length() - 1);  } } |



测试样例：

|  |
| --- |
| @Test public void test1(){  String url="http://59.202.53.51:9080/restapi/prod/IC33000020220124000009/ip";  String ak = "test11";  String sk = "test333";  String method="POST";  Map<String, String> header = HmacAuthUtil.generateHeader(url, method, ak, sk);   Iterator<String> its = header.keySet().iterator();  while(its.hasNext()){  String next = its.next();  System.out.println(next+":"+header.get(next));  } } |

