**浙江省税务局城乡两费管理系统交互方案**

**【税局<——>银行】**

# 拓扑结构



拓扑结构说明：

1. 各银行和浙江税务之间采用专线连接；
2. 各银行和浙江税务之间采用前置机、并部署接口进行数据交互；
3. **本文资源要求针对图中红色部分的服务器及线路；**

# 交互概述

## 与银行数据交互整体结构



与银行数据交互整体结构图

## 主要流程

### 批扣局方与银行的交互流程：



1.局方生产批扣清册，分包发送给对应银行大约每包5000条左右的记录；

1. 银行收到报文后将接收结果返回给局方。
2. 银行对报文进行扣款，扣款结果按每包返回给局方。
3. 局方24小时内收到银行返回结果进行处理。
4. 超过24小时局方仍然没有收到返回结果将主动向银行发起扣款结果查询。
5. 如果扣款未处理完成，将通知主管税务机关管理员和银行虚拟户管理员进行处理。
6. 如果扣款完成，银行应将整个批扣清册的包返回给局方。

说明：批扣清册里yzpzxh一样的应一起进行扣款。

### 对账局方与银行的交互流程：



#### 批扣对账

1. 局方按批次发起对账请求。
2. 银行接收到请求返回相应的总金额总条数。
3. 局方的总金额和总条数如果与银行返回的一致，对账成功，局端进行后续业务。
4. 如果不一致，局端发起明细对账请求。
5. 银行返回对应的明细，局方以银行返回的扣款明细结果为准进行后续业务处理。

#### 日中对账：

1. 银行按日切日期（或一天内的一段时间）向局方发起对账请求。
2. 局方接收到请求返回相应的总金额总条数以及扣款明细。
3. 银行根据局方返回结果进行对账。
4. 银行对账结果返回给局方。
5. 银行根据对账结果进行后续业务。

# 部署说明

## 资源要求

|  |  |  |  |
| --- | --- | --- | --- |
| 设备类型 | 建议配置 | 部署要求 | 数量 |
| 专线 | 业务规模150万户以上：100M带宽；  业务规模150万户以下：50M带宽； | 2根互备或热备；建议热备；  建议不同运营商 | 2 |
| PC服务器 | 业务规模150万户以上：32C，64G ram，800G HDD；产品结构1U；  业务规模150万户以下：16C，32G ram，400G HDD；产品结构1U； | 双机互备或热备；建议热备； | 2 |
| 其他设备 | 网络设备及安全设备 | 不做要求，可利旧； | / |

参考型号：

1）业务规模150万户以上：戴尔PowerEdge R640 ，产品结构1U，cpu 2颗，内存64G，磁盘800G可用，raid 1/5/10 ；

2）业务规模150万户以下：戴尔PowerEdge R630 ，产品结构1U，cpu 2颗，内存32G，磁盘400G可用，raid 1/5/10 。

## 部署要求

1）操作系统要求：CENTOS 7.6.1810 for x86\_64 ，语言环境为：en\_US.utf-8；

2）网卡：建议双网卡绑定；

3）用户要求：uid=zjsw\_cxlf gid=zjsw\_cxlf ,修改ulimit值为unlimited或较大值；举例如下：

data seg size (kbytes, -d) unlimited

file size (blocks, -f) unlimited

max memory size (kbytes, -m) unlimited

open files (-n) 65536

cpu time (seconds, -t) unlimited

max user processes (-u) unlimited

virtual memory (kbytes, -v) unlimited

file locks (-x) unlimited

4）分区要求：划分单独分区，容量占全部磁盘的65%以上，挂载/data目录，用于存放所有业务数据及接口程序;

5）/data权限： chown -R zjsw\_cxlf:zjsw\_cxlf /data

# 接口说明

## 接口概述

业务数据传输采用HTTP 1.0协议，报文需符合JSON格式规范。其中报文业务内容（包含请求和返回）采用非对称双向加密方式。

银行向局端发起请求：银行端使用局端公钥加密发送报文，局端使用银行公钥加密返回报文，银行端使用银行私钥对局端返回报文进行解密。

局端银行发起请求：局端使用银行端公钥加密发送报文，银行端使用局端公钥加密返回报文，局端使用局端私钥对银行返回报文进行解密。

详细加解密方法见附录。

其中报文业务内容（包含请求和返回），请求应含有token,如：

xxx.do?token=xxxx

报文格式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **项目名称** | **含义** | **格式** | **说明** |
| businessType | 业务类型 | String | 每个交易类型有唯一标识 |
| jylsh | 交易流水号 | String | 32位，用于标识每一笔交易 |
| content | 报文业务内容 | String | 报文业务内容，输出项无此内容 |
| resultCode | 返回代码 | String | 输入报文中无此项 |
| resultMsg | 返回消息 | String | 输入报文中无此项 |
| resultObj | 返回报文业务内容 | Object | 输入报文中无此项 |
| 请求报文示例 | {  “businessType”:”业务类型”,  “jylsh”:”交易流水号”,  “content”:”{报文业务内容}”  } | | |
| 返回报文示例 | {  “resultCode”:”返回代码”,  “resultMsg”:”返回消息”,  “resultObj”:{ 报文业务内容 }  } | | |

说明：请求参数中的业务报文节点必须有，值可以根据说明，以上不包含身份验证接口。

业务类型清册如下表：

|  |  |  |
| --- | --- | --- |
| **银行发起批扣接口：** | | |
|  | 银行返回批扣结果 | DZSWJ.GRSBF.PK.PKJGFH |
|  | 局方发送批扣清册 | DZSWJ.GRSBF.PK.PKQCFS |
|  | 局方查询批次状态 | DZSWJ.GRSBF.PK.PKZTCX |
|  | 局方查询批扣总金额和数量对账 | DZSWJ.GRSBF.PK.PKDZ |
|  | 局方查询明细对账 | DZSWJ.GRSBF.PK.PKDZMX |
| **银行发起单户接口：** | | |
|  | 前置查询（单户） | DZSWJ.GRSBF.QZ.DHQZCX |
|  | 前置查询（批量） | DZSWJ.GRSBF.QZ.PLQZCX |
|  | 参保基本信息查询 | DZSWJ.GRSBF.DJ.JBXXCX |
|  | 单户应征查询 | DZSWJ.GRSBF.JF.DHYZCX |
|  | 应征查询（批量） | DZSWJ.GRSBF.JF.PLYZCX |
|  | 电子缴款凭证打印（pdf） | DZSWJ.GRSBF.PZ.DZJKPZDY |
|  | 缴费记录查询 | DZSWJ.GRSBF.JF.JFJLCX |
|  | 单户缴费保存 | DZSWJ.GRSBF.JF.DHJFBC |
|  | 缴费保存（批量） | DZSWJ.GRSBF.JF.PLJFBC |
|  | 单户缴费撤销 | DZSWJ.GRSBF.JF.DHJFZF |
|  | 查询档次以及对应金额 | DZSWJ.GRSBF.CS.DCJECX |
|  | 银行缴费协议签订 | DZSWJ.GRSBF.DJ.YHJFXYQD |
|  | 银行缴费协议终止 | DZSWJ.GRSBF.DJ.YHJFXYZZ |
|  | 查询税务机关可签约银行 | DZSWJ.GRSBF.CS.KQYYHCX |
|  | 查询银行缴费协议数据 | DZSWJ.GRSBF.DJ.YHJFXYCX |
|  | 退费申请 | DZSWJ.GRSBF.PZ.TFSQ |
|  | 参保预登记 | DZSWJ.GRSBF.DJ.CBYDJ |
|  | 个人缴费证明二维码查验 | DZSWJ.GRSBF.PZ.GRJFZMCY |
|  | 电子缴费凭证二维码查验 | DZSWJ.GRSBF.PZ.DZJFPZCY |
|  | 个人缴费证明查询 | DZSWJ.GRSBF.PZ.GRJFZMCX |
|  | 个人缴费证明PDF | DZSWJ.GRSBF.PZ.GRJFZMDY |
|  | 退费记录查询 | DZSWJ.GRSBF.PZ.TFJLCX |
|  | 实时缴费状态查询 | DZSWJ.GRSBF.JF.SSJFZTCX |
|  | 日中对账批量处理扣款状态 | DZSWJ.GRSBF.JF.RZDZ |
|  | 日中对账批量处理结果返回 | DZSWJ.GRSBF.JF.RZDZJGFH |
| **税局发起：** | | |
|  | 税局发起银行缴费协议签订 | DZSWJ.GRSBF.DJ.SJFQYHJFXYQD |
|  | 税局发起缴费协议终止 | DZSWJ.GRSBF.DJ.SJFQYHJFXYZZ |
|  | 税局发起日中对账批量处理扣款状态 | DZSWJ.GRSBF.JF.SWFQRZDZ |
|  | 局方发送单笔协议实时扣款 | DZSWJ.GRSBF.JF.SWFQDBXYSSKK |

## 接入认证接口

向税局发起接入请求，税局返回access\_token，其他接口在访问时需带上此值。

**请求接口说明**

|  |
| --- |
| http请求方式: get http://ip/token?qdid=zjj&qdmm=fc89e8ed6ffa45e18f4109185d4987cd |

**参数说明**

|  |  |  |
| --- | --- | --- |
| **参数** | **是否必须** | **说明** |
| qdid | 是 | 渠道代码 |
| qdmm | 是 | 渠道密码（md5摘要） |

**返回说明**

|  |
| --- |
| {"access\_token":"access\_token","expires\_in":7200} |

**接口访问时示例**

|  |
| --- |
| http请求方式: get http://ip/grsbfdj/grjbxx/djcbjbxx/jbxx?token=fc89e8ed6ffa45e18f4109185d4987cd |

## 接口详细说明

详见银行交互接口文档。注意随时关注税务局通知，获取新的接口文档。



# 附录

## 请求示例

public static void main(String[] args) throws Exception{

String strurl = "http://xxx.do?token=xxxx";

CloseableHttpClient client = HttpClients.createDefault();

HttpPost httpPost = new HttpPost(strurl);

String data="data";

List<NameValuePair> nvps = new ArrayList<NameValuePair>();

nvps.add(new BasicNameValuePair("data", data));

httpPost.setEntity(new UrlEncodedFormEntity(nvps,HTTP.UTF\_8));

HttpResponse response = client.execute(httpPost);

int code = response.getStatusLine().getStatusCode();

if(code==200){

BufferedReader in = new BufferedReader(new InputStreamReader(response.getEntity().getContent(),"utf-8"));

StringBuffer sb = new StringBuffer("");

String line = "";

String NL = System.getProperty("line.separator");

while ((line = in.readLine()) != null) {

sb.append(line + NL);

}

System.out.println(sb);

String ret=sb.toString();

JSONObject j=JSONObject.parseObject(ret);

}

}

## GZIP压缩方式

发送方：Content内容节点，先进行压缩，再加密

接收方：Content内容节点，先进行解密，再解压缩

// 压缩

**public** **static** String compress(String str) **throws** IOException {

**if** (str == **null** || str.length() == 0) {

**return** str;

}

ByteArrayOutputStream out = **new** ByteArrayOutputStream();

GZIPOutputStream gzip = **new** GZIPOutputStream(out);

gzip.write(str.getBytes());

gzip.close();

**return** out.toString("ISO8859\_1");

}

// 解压缩

**public** **static** String uncompress(String str) **throws** IOException {

**if** (str == **null** || str.length() == 0) {

**return** str;

}

ByteArrayOutputStream out = **new** ByteArrayOutputStream();

ByteArrayInputStream in = **new** ByteArrayInputStream(str

.getBytes("ISO8859\_1"));

GZIPInputStream gunzip = **new** GZIPInputStream(in);

**byte**[] buffer = **new** **byte**[256];

**int** n;

**while** ((n = gunzip.read(buffer)) >= 0) {

out.write(buffer, 0, n);

}

**return** out.toString();

}

## 加解密证书

**1）证书**

公钥：913300000000000000.cer（银行公钥），swj.cer（税务局公钥）

私钥：913300000000000000.pfx（银行私钥）,私钥密码：000000

**2）jar包**

加解密提供bcprov-jdk15on-1.51.jar，zkcryptutil.jar包。

****

**3）加密示例：**

EeCyptKit kit=new EeCyptKit();

kit.setPublicCertFile("h:/913300000000000000.cer");//设置公钥证书路径

String enzipstr=kit.encryptByPublicKey(zipstr);//加密

**4）解密示例：**

kit.setUserPrivateFile("h:/913300000000000000.pfx");//设置私钥证书路径

kit.setUserPrivateFilePassword("000000");//设置私钥证书密码

String dezipstr=kit.decryptByPrivateKey(enzipstr);//解密