# 1.背景

由于历史原因,旧支付系统和其他业务系统出现严重耦合,特别是跟支付相关的pay\_order表加入了很多业务数据.同时系统代码编写也比较混乱,不利系统扩展与升级.因此,需要对现有支付系统进行重写.重写后,新支付系统将做为独立的应用系统存在,只负责与外部支付系统进行对接,不依赖其他内部其他系统,也不会处理与内部系统业务相关的逻辑.

# 2.系统规划

由于当前有支付相关的需求需要开发,故重写的第一个版本只完成本次需求所需功能,即扫码支付,重点还是在系统可扩展性方面考虑,后面会根据业务需求逐步对接app支付,小程序支付,h5支付等.同时第一个版本与内部系统对接时,暂不支持签名验签功能,后面考虑会引入RSA2做安全方面升级.

# 3.系统对接两种方式

## 3.1.rpc方式调用

### 3.1.1.添加maven依赖

|  |
| --- |
| <dependency>  <groupId>com.sandu</groupId>  <artifactId>sandu-yun-pay-service</artifactId>  <version>1.0.0-SNAPSHOT</version>  </dependency> |

### 3.1.2.spring配置扫码支付bean

|  |
| --- |
| <dubbo:reference interface=*"com.sandu.gateway.pay.forward.service.PayService"*  id=*"aliScanCodePay"* protocol=*"dubbo"* timeout=*"30000"* retries=*"0"* version=*"2.0.0"* group=*"aliScanCodePay"*/>    <dubbo:reference interface=*"com.sandu.gateway.pay.forward.service.PayService"*  id=*"wxScanCodePay"* protocol=*"dubbo"* timeout=*"30000"* retries=*"0"* version=*"2.0.0"* group=*"wxScanCodePay"*/> |

### 3. 1.3代码编写

**1.bean获取**:通过以下两个方式可以获取上方定义的spring bean

|  |
| --- |
| @Resource(name="wxScanCodePay")  **private** PayService payService; |

或者

|  |
| --- |
| PayService payService = SpringContextHolder.*getBean*("wxScanCodePay"); |

**2.接口调用**

|  |
| --- |
| String obj = payService.doPay(payParam); |

### 3.1.4.样例代码

|  |
| --- |
| **public** Object doPayTest() {  **try** {  PayParam payParam = **new** PayParam();  payParam.setIntenalTradeNo("trade1000");  payParam.setTradeDesc("tradeDesc");  payParam.setTotalFee(1L);  payParam.setPayMethod(  PayParam.PAY\_METHOD\_ALI\_SCANCODE\_PAY);  payParam.setIp("127.0.0.1");  payParam.setNotifyUrl("http://127.0.0.1:8089/v1/gateway/pay/callback/test/notify");  payParam.setOperator(1L);  payParam.setPlatformCode("miniProgram");  payParam.setSource(PayParam.SOURCE\_SYSTEM);  PayService payService = SpringContextHolder.*getBean*(payParam.getPayMethod());  String obj = payService.doPay(payParam);  **return** obj;  } **catch** (BizException e) {  } **catch** (Exception e) {  *logger*.error("系统错误:", e);  }  } |

## 3.2.rest方式调用

### 3.2.1 API接口

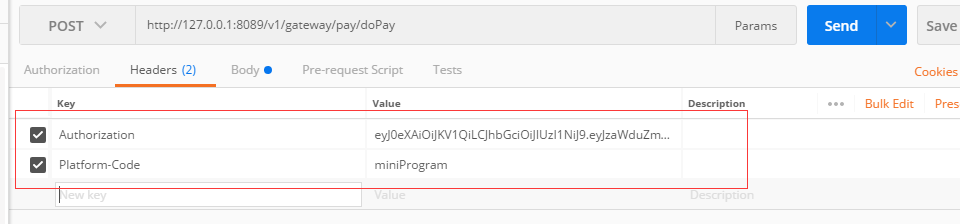
**集成请求地址:** https://zhifu.ci.sanduspace.com/v1/gateway/pay/doPay

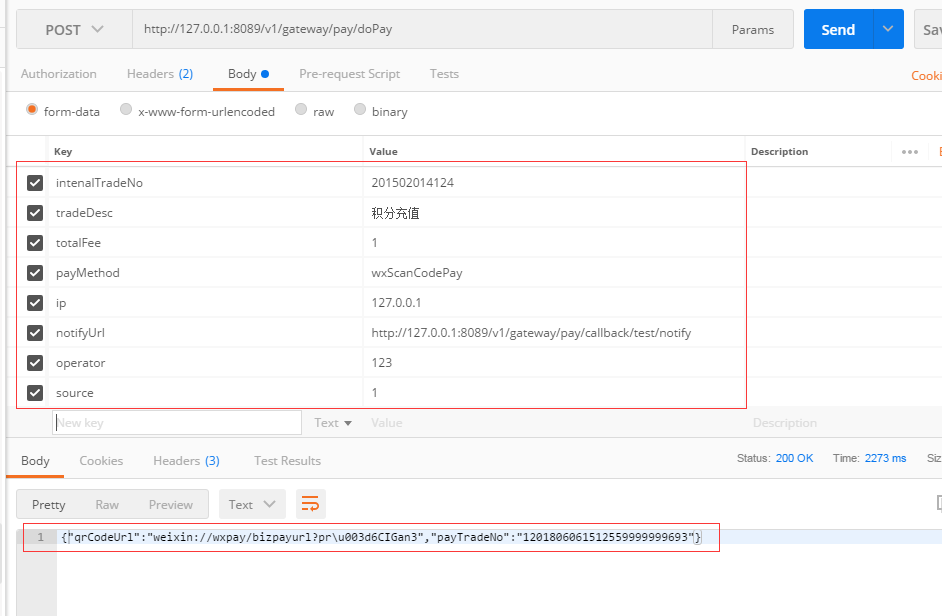
**头部参数:**{ Platform-Code: miniProgram, Authorization:token}

**请求方式:**post

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名** | **变量名** | **必填** | **类型** | **示例值** | **描述** |
| 应用系统交易号 | intenalTradeNo | Y | String | 201502014124 | 通常是订单号 |
| 交易描述 | tradeDesc | Y | String | 积分充值 | 业务场景描述 |
| 交易金额 | totalFee | Y | Long | 1 | 以分为单位 |
| 支付方式 | payMethod | Y | String | aliScanCodePay  wxScanCodePay |  |
| Ip地址 | ip | Y | String | 127.0.0.1 | 调用方ip |
| 回调url | notifyUrl | Y | String | http://order.sd.com/notify | 用户支付成功后,系统会通过此url通知相应系统 |
| 操作人id | operator | Y | Long | 111 | 用户id |
| 交易来源 | source | Y | Long | 1 | 来源哪个应用系统  1.基础服务,  2.订单服务... |
|  |  |  |  |  |  |

### 3.2.2调用示例:

****



### 3.2.3回调

前面调用支付系统时,请求参数中的notifyUrl表示当支付系统收到外部支付通知(支付宝,微信)后,会将相应的支付结果通过此url转发给应用系统.应用系统接收回调参数后,通过resultCode是否等于"SUCCESS"判断支付是否成功

如果支付成功,则调用

response.getWriter().print("SUCCESS");

通知支付系统已经支付成功,否则

response.getWriter().print("FAIL");

通知支付系统支付失败.

#### 3.2.3.1回调参数:

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **变量名** | **示例值** | **描述** |
| 支付结果 | resultCode | SUCCESS/FAIL | SUCCESS:支付成功  FAIL:支付失败 |
| 支付结果描述 | resultMsg | OK/ FAIL |  |
| 应用系统交易号 | intenalTradeNo | 201502014124 | 通常是订单号 |
| 支付系统交易号 | payTradeNo | 112015020141249963 |  |
| 交易描述 | tradeDesc | 积分充值 | 业务场景描述 |
| 交易金额 | totalFee | 1 | 以分为单位 |
| 操作人id | operator | 111 | 用户id |
| 平台编码 | platformCode | pcHouseDraw | 交易来源平台 |
| 交易来源 | source | 1 | 来源哪个应用系统  1.基础服务,  2.订单服务... |

#### 3.2.3.2示例代码:

|  |
| --- |
| @RequestMapping(value = "/test/notify")  **public** **void** test(HttpServletRequest request, HttpServletResponse response) {  **try** {  *logger*.error("通知参数:");  Map<String, String[]> requestParams = request.getParameterMap();  **for** (Iterator<String> iter = requestParams.keySet().iterator(); iter.hasNext();) {  String name = (String) iter.next();  String[] values = (String[]) requestParams.get(name);  String valueStr = "";  **for** (**int** i = 0; i < values.length; i++) {  valueStr = (i == values.length - 1) ? valueStr + values[i] : valueStr + values[i] + ",";  }  // 乱码解决，这段代码在出现乱码时使用。  // valueStr = new String(valueStr.getBytes("ISO-8859-1"), "utf-8");  *logger*.error(name+" : "+valueStr);  }    response.getWriter().print("SUCCESS");  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }    } |

# 4.注意事项

之前的支付场景暂时不进行改造,还是按原来的逻辑走.包括积分充值,平台开通,包年包月,渲染等.后面如果涉及到这部份场景修改再评估是否重构.