be-openapi-javasdk接入文档

# 概述

本文档主要针对java sdk接入的说明。

阅读对象：商户网站的java开发人员。

开发进度：目前网关及sdk高速开发中，文档将频繁变更，请使用方及时更新

# 前置条件

### Jdk

Jdk 8 以上

在命令行中执行校验命令java –version，返回以下类似

java version "1.8.0\_191"

Java(TM) SE Runtime Environment (build 1.8.0\_191-b12)

Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)

如果返回不正确，参考java官方安装文档

### Maven

Maven 3.0以上，在命令行中执行校验命令mvn –version，返回以下类似

Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5ffb20918da4f719f3; 2018-10-25T02:41:47+08:00)

Maven home: D:\apache-maven-3.6.0\bin\..

Java version: 1.8.0\_191, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk1.8.0\_191\jre

Default locale: zh\_CN, platform encoding: GBK

OS name: "windows 7", version: "6.1", arch: "amd64", family: "windows"

如果返回不正确，参考maven官方安装文档

### 导入maven库

导入本地库和远程库只要执行一个即可，根据使用方的研发环境决定。

#### 导入本地库

移动到‘执行库’子目录

mvn install:install-file -Dfile=sign-tool-0.2-SNAPSHOT.jar -DpomFile=sign-tool-0.2-SNAPSHOT.pom

mvn install:install-file -Dfile=be-openapi-0.3-SNAPSHOT.jar -DpomFile=be-openapi-0.3-SNAPSHOT.pom

mvn install:install-file -Dfile=be-openapi-sdk-0.9-SNAPSHOT.jar -DpomFile=be-openapi-sdk-0.9-SNAPSHOT.pom

mvn install:install-file -Dfile=cap-openapi-0.3-SNAPSHOT.jar -DpomFile=cap-openapi-0.3-SNAPSHOT.pom

#### 导入远程库()

移动到‘执行库’子目录

mvn deploy:deploy-file -Durl=公司maven库url路径 -DrepositoryId=库名 -Dfile= sign-tool-0.2-SNAPSHOT.jar -DpomFile= sign-tool-0.2-SNAPSHOT.pom

mvn deploy:deploy-file -Durl=公司maven库url路径 -DrepositoryId=库名 -Dfile= be-openapi-0.3-SNAPSHOT.jar -DpomFile= be-openapi-0.3-SNAPSHOT.pom

mvn deploy:deploy-file -Durl=公司maven库url路径 -DrepositoryId=库名 -Dfile= be-openapi-sdk-0.9-SNAPSHOT.jar -DpomFile=be-openapi-sdk-0.9-SNAPSHOT.pom

mvn deploy:deploy-file -Durl=公司maven库url路径 -DrepositoryId=库名 -Dfile= cap-openapi-0.3-SNAPSHOT.jar -DpomFile= cap-openapi-0.3-SNAPSHOT.pom

### 使用方项目导入maven依赖声明

<dependency>

<groupId>io.bitexpress.openapi</groupId>

<artifactId>be-openapi-sdk</artifactId>

<version>0.9-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>io.bitexpress.openapi</groupId>

<artifactId>cap-openapi</artifactId>

<version>0.3-SNAPSHOT</version>

</dependency>

<!--http组件默认使用apache httpclient4,也可以切换成okhttp -->

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>4.5.6</version>

</dependency>

以上声明会将级联依赖一并导入

sdk级联依赖具体内容如下

io.bitexpress.openapi:be-openapi-sdk:jar:0.4-SNAPSHOT

+- io.bitexpress.openapi:be-openapi:jar:0.3-SNAPSHOT:compile

| +- javax.validation:validation-api:jar:2.0.1.Final:compile

| \- com.fasterxml.jackson.core:jackson-annotations:jar:2.9.0:compile

+- org.apache.commons:commons-lang3:jar:3.7:compile

+- com.fasterxml.jackson.core:jackson-databind:jar:2.9.8:compile

| \- com.fasterxml.jackson.core:jackson-core:jar:2.9.8:compile

+- org.springframework:spring-web:jar:5.1.6.RELEASE:compile

| +- org.springframework:spring-beans:jar:5.1.6.RELEASE:compile

| \- org.springframework:spring-core:jar:5.1.6.RELEASE:compile

| \- org.springframework:spring-jcl:jar:5.1.6.RELEASE:compile

+- io.bitexpress.utility:sign-tool:jar:0.2-SNAPSHOT:compile

| +- org.bouncycastle:bcprov-jdk15on:jar:1.60:runtime

| +- org.bouncycastle:bcpkix-jdk15on:jar:1.60:runtime

| +- commons-codec:commons-codec:jar:1.11:compile

| +- commons-io:commons-io:jar:2.6:compile

| \- org.slf4j:slf4j-api:jar:1.7.25:compile

+- org.apache.httpcomponents:httpclient:jar:4.5.6:compile

| \- org.apache.httpcomponents:httpcore:jar:4.4.10:compile

+- org.testng:testng:jar:6.14.3:test

| +- com.beust:jcommander:jar:1.72:test

| \- org.apache-extras.beanshell:bsh:jar:2.0b6:test

+- ch.qos.logback:logback-classic:jar:1.2.3:test

+- ch.qos.logback:logback-core:jar:1.2.3:test

\- org.slf4j:jcl-over-slf4j:jar:1.7.25:test

# 公密钥准备



公密钥对格式都是pem格式，使用app生成用户私钥，并使用app将公钥部分上传到开放平台，并使用从商户控台下载开放平台公钥。

# 接口说明

## 概览图



## 功能说明

### GatewayClient及GatewayClientImpl

#### 初始化

GatewayClientBuilder builder = **new** GatewayClientBuilder();

// 设置用户uid

builder.setUid(***123L***);

// 设置服务地址

builder.setUrl("http://localhost:8080/gateway.do");

// 设置客户端私钥

builder.setClientPrivateKeyFilePath("src/test/resources/user\_key\_pair/private\_key.pem");

gatewayClient = builder.build();

### 创建及时到账交易

String outTradeNo = "20190524134059";

CashierAcquireRequest request = new CashierAcquireRequest();

request.setAmount(new BigDecimal("2.0012301"));

request.setName("test2100");

request.setNotifyUrl("http://localhost:8081/createAcquireOrder/callback");

request.setAssetCode("ETH");

request.setOutTradeNo(outTradeNo);

ServiceIndex serviceIndex = new ServiceIndex("createAcquireOrder", GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

CashierAcquireResponse response = gatewayClient.invoke(serviceIndex, request,

CashierAcquireResponse.class);

String url = response.getUrl();

System.out.println(response);

### 创建法币收单交易

String outTradeNo = "20190524164030";

FiatAcquireRequest request = new FiatAcquireRequest();

request.setOutTradeNo(outTradeNo);

request.setCurrency(Currency.CNY);

request.setEmail("abc123@163.com");

request.setLanguage("zh");

request.setName("legalPayment");

request.setNotifyUrl("http://10.65.100.57:8081/createFiatAcquireOrder/callback");

request.setAssetCode("USDT\_ERC20");

request.setFiatAmount(new BigDecimal("21"));

ServiceIndex serviceIndex = new ServiceIndex("createFiatAcquireOrder",

GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

CashierAcquireResponse response = gatewayClient.invoke(serviceIndex, request,

CashierAcquireResponse.class);

String url = response.getUrl();

System.out.println(response);

### 查询及时到账交易

String outTradeNo = "20190524135359";

CashierAcquireQueryRequest queryRequest = new CashierAcquireQueryRequest();

queryRequest.setOutTradeNo(outTradeNo);

ServiceIndex serviceIndex = new ServiceIndex("querySingleOrder", GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

CashierAcquireOrder cashierAcquireOrder = gatewayClient.invoke(serviceIndex, queryRequest,

CashierAcquireOrder.class);

System.out.println(cashierAcquireOrder);

### 付款到地址

String outTradeNo = "20190529171345";

WithdrawRequest withdrawRequest = new WithdrawRequest();

withdrawRequest.setAmount(new BigDecimal("1.111111"));

withdrawRequest.setNotifyUrl("http://10.65.100.57:8081/payToAddress/callback");

withdrawRequest.setReceiveAddress("0xf8d6478ee5bbb23e5084fbec67f6c7ad6001d026");

withdrawRequest.setAssetCode("ETH");

withdrawRequest.setOutTradeNo(outTradeNo);

ServiceIndex serviceIndex = new ServiceIndex("payToAddress", GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

WithdrawOrder withdrawOrder = gatewayClient.invoke(serviceIndex, withdrawRequest, WithdrawOrder.class);

System.out.println(withdrawOrder);

### 查询付款到地址

String outTradeNo = "20190529171345";

WithdrawQueryRequest withdrawQueryRequest = new WithdrawQueryRequest();

withdrawQueryRequest.setOutTradeNo(outTradeNo);

ServiceIndex serviceIndex = new ServiceIndex("queryPayToAddress", GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

WithdrawOrder withdrawOrder = gatewayClient.invoke(serviceIndex, withdrawQueryRequest, WithdrawOrder.class);

System.out.println(withdrawOrder);

### 付款到账户

String outTradeNo = "20190527131246";

TransferRequest transferRequest = new TransferRequest();

transferRequest.setAmount(new BigDecimal("22.112"));

transferRequest.setNotifyUrl("http://localhost:8081/payToAccount/callback");

transferRequest.setReceiverMobile("+8613922988196");

transferRequest.setAssetCode("ETH");

transferRequest.setOutTradeNo(outTradeNo);

ServiceIndex serviceIndex = new ServiceIndex("payToAccount", GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

TransferOrder transferOrder = gatewayClient.invoke(serviceIndex, transferRequest, TransferOrder.class);

System.out.println(transferOrder);

### 查询付款到账户

String outTradeNo = "20190527132021";

TransferQueryRequest transferQueryRequest = new TransferQueryRequest();

transferQueryRequest.setOutTradeNo(outTradeNo);

ServiceIndex serviceIndex = new ServiceIndex("queryPayToAccount", GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

TransferOrder transferOrder = gatewayClient.invoke(serviceIndex, transferQueryRequest, TransferOrder.class);

System.out.println(transferOrder);

### 查询单一币种账号余额

QueryBalanceRequest queryRequest = new QueryBalanceRequest();

queryRequest.setAssetCode("ETH");

ServiceIndex serviceIndex = new ServiceIndex("queryBalance", GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

Balance balance = gatewayClient.invoke(serviceIndex, queryRequest, Balance.class);

System.out.println(balance);

### 创建为订单付款交易

String outTradeNo = formatter.format(LocalDateTime.now());

CreatePaymentForOrderRequest request = new CreatePaymentForOrderRequest();

request.setAmount(new BigDecimal("2.982954"));

request.setNotifyUrl("http://localhost:8081/createPaymentForOrder/callback");

request.setPayeeUid("459");

request.setTargetOrderId("20190529000045026");

request.setAssetCode("USDT\_ERC20");

request.setOutTradeNo(outTradeNo);

ServiceIndex serviceIndex = new ServiceIndex("createPaymentForOrder",

GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

PaymentOrder paymentOrder = gatewayClient.invoke(serviceIndex, request,

PaymentOrder.class);

System.out.println(paymentOrder);

### 为订单付款的交易查询

String outTradeNo = "20190524135515";

QueryPaymentForOrderRequest queryRequest = new QueryPaymentForOrderRequest();

queryRequest.setOutTradeNo(outTradeNo);

ServiceIndex serviceIndex = new ServiceIndex("queryPaymentForOrder",

GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

PaymentOrder paymentOrder = gatewayClient.invoke(serviceIndex, queryRequest, PaymentOrder.class);

System.out.println(paymentOrder);

### 为订单付款的交易分页查询(暂不能使用)

QueryPaymentForOrderPageRequest queryPageRequest = new QueryPaymentForOrderPageRequest();

Date now = new Date();

Date startTime = DateUtils.addMinutes(now, -10);

queryPageRequest.setStartTime(startTime);

queryPageRequest.setEndTime(now);

SortPageParam sortPageParam = new SortPageParam();

sortPageParam.setSize(5);

sortPageParam.setNumber(0);

queryPageRequest.setSortPageParam(sortPageParam);

ServiceIndex serviceIndex = new ServiceIndex("queryPaymentForOrderPage",

GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

JavaType constructParametricType = gatewayClient.getObjectMapper().getTypeFactory()

.constructParametricType(Pagination.class, PaymentOrder.class);

Pagination<PaymentOrder> paymentOrderPagination = gatewayClient.invoke(serviceIndex,

queryPageRequest, constructParametricType);

System.out.println(paymentOrderPagination);

### 按资产代码查询OTC汇率

QueryOtcQuoteByAssetCodeRequest queryRequest = new QueryOtcQuoteByAssetCodeRequest();

queryRequest.setAssetCode("USDT\_ERC20");

ServiceIndex serviceIndex = new ServiceIndex("queryQuoteByAssetCode",

GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

JavaType constructParametricType = gatewayClient.getObjectMapper().getTypeFactory()

.constructCollectionType(List.class, OTCQuote.class);

List<OTCQuote> list = gatewayClient.invoke(serviceIndex, queryRequest, constructParametricType);

for (OTCQuote otcQuote : list) {

System.out.println(otcQuote);

}

### 按法币币种查询OTC汇率

QueryOtcQuoteByCurrencyCodeRequest queryRequest = new QueryOtcQuoteByCurrencyCodeRequest();

queryRequest.setCurrencyCode("CNY");

ServiceIndex serviceIndex = new ServiceIndex("queryQuoteByCurrencyCode",

GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

JavaType constructParametricType = gatewayClient.getObjectMapper().getTypeFactory()

.constructCollectionType(List.class, OTCQuote.class);

List<OTCQuote> list = gatewayClient.invoke(serviceIndex, queryRequest, constructParametricType);

for (OTCQuote otcQuote : list) {

System.out.println(otcQuote);

}

### 创建资产处置

CreateFiatFundoutAssetDispositionRequest request = new CreateFiatFundoutAssetDispositionRequest();

request.setTargetOrderId(targetOrderId);

request.setAssetDisposition("RECEIVE");

request.setAssetCode("USDT\_ERC20");

request.setAssetAmount(new BigDecimal("10"));

request.setDeliveryMemo("test asset disposition");

ServiceIndex serviceIndex = new ServiceIndex("createFiatFundoutAssetDisposition", GatewayClientImpl.DEFAULT\_SIGNATURE\_VERSION);

AssetDispositionData dispositionData = gatewayClient.invoke(serviceIndex, request, AssetDispositionData.class);

## 异步通知消息解析

异步通知消息解析仅处理数据解析及鉴权操作，消息报文本身需要研发自己获取

### 初始化

MapGatewayNotificationParserBuilder builder = new MapGatewayNotificationParserBuilder();

builder.setUid(UID);

builder.setServerSidePublicKeyFilePath("src/test/resources/user\_key\_pair/public\_key.pem");

Map<ServiceIndex, Class> notificationClassMap = new HashMap<>();

notificationClassMap.put(new ServiceIndex("payToAccount", new BigDecimal("1.0")), TransferOrder.class);

notificationClassMap.put(new ServiceIndex("payToAddress", new BigDecimal("1.0")), WithdrawOrder.class);

notificationClassMap.put(new ServiceIndex("createAcquireOrder", new BigDecimal("1.0")),

CashierAcquireOrder.class);

notificationClassMap.put(new ServiceIndex("createFiatAcquireOrder", new BigDecimal("1.0")),

CashierAcquireOrder.class);

notificationClassMap.put(new ServiceIndex("createPaymentForOrder", new BigDecimal("1.0")), PaymentOrder.class);

builder.setNotificationClassMap(notificationClassMap);

GatewayNotificationParser parser = builder.build();

反序列化的映射关系是要自己维护，样例中的映射关系是目前用到，后续会随着功能增加同步更新配置代码

### Servlet接收消息（springmvc）

接收通知的方式有很多种，这里只是举例，研发应该根据自己的情况灵活变更代码。

@PostMapping(value = "/notification.do", consumes = MediaType.APPLICATION\_JSON\_VALUE, produces = MediaType.APPLICATION\_JSON\_UTF8\_VALUE)

@ResponseBody

public String handleNotification(ServletRequest request) throws IOException {

try (ServletInputStream inputStream = request.getInputStream()) {

String string = IOUtils.toString(inputStream, StandardCharsets.UTF\_8);

OpenApiRequestContent<PaymentOrder> parseNotification = parser.parseNotification(string);//自动下塑造型

// 处理业务

boolean success = true;// 处理业务结果

return parser.getAckString(success);

}

}

### 接收为订单付款消息

String requestEnvelopeStr = FileUtils.readFileToString(new File("src/test/resources/notification.json"),

StandardCharsets.UTF\_8);//数据来源根据实际情况获取

OpenApiRequestContent<PaymentOrder> parseNotification = parser.parseNotification(requestEnvelopeStr);

### 异步通知消息SpringBoot样例

定义通知实体映射

@Bean("notificationMap")

public Map<ServiceIndex, Class> notificationMap() {

Map<ServiceIndex, Class> notificationClassMap = new HashMap<>();

notificationClassMap.put(new ServiceIndex("payToAccount", new BigDecimal("1.0")), TransferOrder.class);

notificationClassMap.put(new ServiceIndex("payToAddress", new BigDecimal("1.0")), WithdrawOrder.class);

notificationClassMap.put(new ServiceIndex("createAcquireOrder", new BigDecimal("1.0")),

CashierAcquireOrder.class);

notificationClassMap.put(new ServiceIndex("createFiatAcquireOrder", new BigDecimal("1.0")),

CashierAcquireOrder.class);

notificationClassMap.put(new ServiceIndex("createPaymentForOrder", new BigDecimal("1.0")), PaymentOrder.class);

return notificationClassMap;

}

自定义解析器类工厂Bean

@Component("notificationParser")

public class GatewayNotificationParserFactoryBean implements FactoryBean<GatewayNotificationParser> {

@Value("${public.key.path}")

private String publicKeyPath;

@Value("${uid}")

private Long uid;

@Resource(name = "notificationMap")

private Map<ServiceIndex, Class> notificationMap;

@Override

public GatewayNotificationParser getObject() throws Exception {

MapGatewayNotificationParserBuilder builder = new MapGatewayNotificationParserBuilder();

builder.setUid(uid);

builder.setServerSidePublicKeyFilePath(publicKeyPath);

builder.setNotificationClassMap(notificationMap);

return builder.build();

}

@Override

public Class<?> getObjectType() {

return GatewayNotificationParser.class;

}

@Override

public boolean isSingleton() {

return true;

}

}

自定义SpringMVC消息解析器

@Component

public class OpenApiRequestContentConverter<T> implements HttpMessageConverter<OpenApiRequestContent<T>> {

@Resource(name = "notificationParser")

private GatewayNotificationParser parser;

@Override

public boolean canRead(Class<?> clazz, MediaType mediaType) {

return (Objects.equals(mediaType, MediaType.APPLICATION\_JSON) || Objects.equals(mediaType, MediaType.APPLICATION\_JSON\_UTF8)) && clazz == OpenApiRequestContent.class;

}

@Override

public boolean canWrite(Class<?> clazz, MediaType mediaType) {

return false;

}

@Override

public List<MediaType> getSupportedMediaTypes() {

List<MediaType> mediaTypeList = new ArrayList<>();

mediaTypeList.add(MediaType.APPLICATION\_JSON);

mediaTypeList.add(MediaType.APPLICATION\_JSON\_UTF8);

return mediaTypeList;

}

@Override

public OpenApiRequestContent<T> read(Class<? extends OpenApiRequestContent<T>> clazz, HttpInputMessage inputMessage) throws IOException, HttpMessageNotReadableException {

String body = IOUtils.toString(inputMessage.getBody(), StandardCharsets.UTF\_8);

return parser.parseNotification(body);

}

@Override

public void write(OpenApiRequestContent<T> tOpenApiRequestContent, MediaType contentType, HttpOutputMessage outputMessage) throws IOException, HttpMessageNotWritableException {

throw new UnsupportedOperationException("operation forbidden");

}

}

回调服务解析为具体类型，并产生符合条件的响应

@PostMapping(value = {"createAcquireOrder/callback", "createFiatAcquireOrder/callback"}, consumes = MediaType.APPLICATION\_JSON\_VALUE, produces = MediaType.APPLICATION\_JSON\_UTF8\_VALUE)

@ResponseBody

public NotificationAck createAcquireOrder(@RequestBody OpenApiRequestContent<AcquireOrder> content) {

logger.info("create acquire order callback: {}", content);

return parser.getAck(true);

}

@PostMapping(value = "payToAccount/callback", consumes = MediaType.APPLICATION\_JSON\_VALUE, produces = MediaType.APPLICATION\_JSON\_UTF8\_VALUE)

public void payToAccount(HttpServletResponse response, @RequestBody OpenApiRequestContent<TransferOrder> content) throws IOException {

logger.info("pay to account callback: {}", content);

response.getWriter().print(parser.getAckString(true));

}