微信支付

1.介绍Native

`Native` 支付是指商户系统按微信支付协议生成支付二维码，用户再用微信“扫一扫”完成支付的模式。

2.添加依赖，配置参数，读取配置

```
<dependency>
    <groupId>com.github.wechatpay-apiv3</groupId>
    <artifactId>wechatpay-apache-httpclient</artifactId>
    <version>0.4.5</version>
</dependency>
```

```
#微信配置
wx:
  mch-id: 请填写您的mchId
  wx-pay-appid: 请填写您的appId
  mch-serial-no: 商户证书序列号,需要和证书对应
  api-v3-key: 请填写您的api密钥
  private-key-path: 请填写您的商户私钥（apiclient_key.pem）所在文件（绝对路径或相对路径）
  success-return-url: 支付成功页面跳转
  callback-url: 支付成功，回调通知
```

```
@Data
@Configuration
@ConfigurationProperties(prefix = "wx")
public class WechatPropertiesV3 {
    /**
     * 商户号
     */
    private String mchId;

    /**
     * 公众号id 需要和商户号绑定
     */
    private String wxPayAppid;

    /**
     * 商户证书序列号,需要和证书对应
     */
    private String mchSerialNo;

    /**
     * API V3密钥
     */
    private String apiV3Key;

    /**
     * 商户私钥路径（微信服务端会根据证书序列号，找到证书获取公钥进行解密数据）
     */
    private String privateKeyPath;

    /**
     * 支付成功页面跳转
```

```
     */
    private String successReturnUrl;

    /**
     * 支付成功，回调通知
     */
    private String callbackUrl;
}
```

3.商户私钥证书代码读取

`java` 加载商户证书私钥 以下部分内容来自微信支付文档样例
https://github.com/wechatpay-apiv3/wechatpay-apache-httpclient
商户申请商户 `API` 证书时，会生成商户私钥，并保存在本地证书文件夹的文件 `apiclient_key.pem` 中,商户开发者可以使用方法 `PemUtil.loadPrivateKey()` 加载证书

> 样例：
>
> ```
> # 示例：私钥存储在文件
> PrivateKey merchantPrivateKey = PemUtil.loadPrivateKey(
>  new FileInputStream("/path/to/apiclient_key.pem"));
> # 示例：私钥为String字符串
> PrivateKey merchantPrivateKey = PemUtil.loadPrivateKey(
>  new ByteArrayInputStream(privateKey.getBytes("utf-8")));
> ```

代码加载读取秘钥/定时获取微信签名验证器/获取 `http` 请求对象，会自动的处理签名和验签：

```
/**
 * 代码加载读取秘钥/定时获取微信签名验证器/获取http请求对象，会自动的处理签名和验签
 *
 * @author user
 */
@Configuration
@Slf4j
public class PayBeanConfig {

    private static final int TWO_ZERO_ZERO = 200;

    @Resource
    private WechatPropertiesV3 payConfig;

    /**
     * 加载秘钥
     *
     * @return PrivateKey
     * @throws IOException IOException
     */
    public PrivateKey getPrivateKey() throws IOException {
        InputStream inputStream = new ClassPathResource(payConfig.getPrivateKeyPath().replace("classpath:",
"")).getInputStream();
        String content = new BufferedReader(new
InputStreamReader(inputStream)).lines().collect(Collectors.joining(System.lineSeparator()));
        try {
            String privateKey = content.replace("-----BEGIN PRIVATE KEY-----", "").replace("-----END PRIVATE KEY---
--", "").replaceAll("\\s+", "");
            KeyFactory kf = KeyFactory.getInstance("RSA");
            return kf.generatePrivate(new PKCS8EncodedKeySpec(Base64.getDecoder().decode(privateKey)));
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException("当前Java环境不支持RSA", e);
```

```
        } catch (InvalidKeySpecException e) {
            throw new RuntimeException("无效的密钥格式");
        }
    }

    /**
     * 定时获取微信签名验证器，自动获取微信平台证书（证书里面包括微信平台公钥）
     * 使用定时更新的签名验证器，不需要传入证书
     * AutoUpdateCertificatesVerifier已被弃用，0.3.0ScheduledUpdateCertificatesVerifier,0.4.0移除
     *
     * @return CertificatesVerifier
     */
    @Bean
    public CertificatesVerifier getCertificatesVerifier() throws IOException {
        PrivateKeySigner signer = new PrivateKeySigner(payConfig.getMchSerialNo(), getPrivateKey());
        Credentials credentials = new WechatPay2Credentials(payConfig.getMchId(), signer);
        byte[] apiV3Key = payConfig.getApiV3Key().getBytes(StandardCharsets.UTF_8);
        CertificatesVerifier verifier;
        try {
            verifier = this.autoUpdateCert(credentials, apiV3Key);
        } catch (GeneralSecurityException | IOException var6) {
            throw new RuntimeException(var6);
        }
        return verifier;
    }

    private CertificatesVerifier autoUpdateCert(Credentials credentials, byte[] apiV3Key) throws IOException,
GeneralSecurityException {
        CloseableHttpClient httpClient =
WechatPayHttpClientBuilder.create().withCredentials(credentials).withValidator((validator) -> true).build();

        label80:
        {
            try {
                label81:
                {
                    HttpGet httpGet = new HttpGet("https://api.mch.weixin.qq.com/v3/certificates");
                    httpGet.addHeader("Accept", ContentType.APPLICATION_JSON.toString());
                    CloseableHttpResponse response = httpClient.execute(httpGet);

                    label82:
                    {
                        try {
                            int statusCode = response.getStatusLine().getStatusCode();
                            String body = EntityUtils.toString(response.getEntity());
                            if (statusCode == TWO_ZERO_ZERO) {
                                List<X509Certificate> newCertList = this.deserializeToCerts(apiV3Key, body);
                                if (newCertList.isEmpty()) {
                                    log.warn("Cert list is empty");
                                    break label82;
                                }

                                return new CertificatesVerifier(newCertList);
                            } else {
                                log.warn("Auto update cert failed, statusCode = {}, body = {}", statusCode, body);
                            }
                        } catch (Throwable var9) {
                            if (response != null) {
                                try {
```

```
                            response.close();
                        } catch (Throwable var8) {
                            var9.addSuppressed(var8);
                        }
                    }

                    throw var9;
                }

                response.close();
                break label81;
            }

            response.close();
            break label80;
        }
    } catch (Throwable var10) {
        if (httpClient != null) {
            try {
                httpClient.close();
            } catch (Throwable var7) {
                var10.addSuppressed(var7);
            }
        }

        throw var10;
    }

    httpClient.close();

    return null;
}

    httpClient.close();

    return null;
}

    private List<X509Certificate> deserializeToCerts(byte[] apiV3Key, String body) throws GeneralSecurityException,
IOException {
        AesUtil aesUtil = new AesUtil(apiV3Key);
        ObjectMapper mapper = new ObjectMapper();
        JsonNode dataNode = mapper.readTree(body).get("data");
        List<X509Certificate> newCertList = new ArrayList<>();
        if (dataNode != null) {
            int i = 0;

            for (int count = dataNode.size(); i < count; ++i) {
                JsonNode node = dataNode.get(i).get("encrypt_certificate");
                String cert = aesUtil.decryptToString(node.get("associated_data").toString().replace("\"",
"").getBytes(StandardCharsets.UTF_8), node.get("nonce").toString().replace("\"",
"").getBytes(StandardCharsets.UTF_8), node.get("ciphertext").toString().replace("\"", ""));
                CertificateFactory cf = CertificateFactory.getInstance("X509");
                X509Certificate x509Cert = (X509Certificate) cf.generateCertificate(new
ByteArrayInputStream(cert.getBytes(StandardCharsets.UTF_8)));

                try {
                    x509Cert.checkValidity();
                } catch (CertificateNotYetValidException | CertificateExpiredException var14) {
```

```
                continue;
            }

            newCertList.add(x509Cert);
        }
    }

    return newCertList;
}

/**
 * 获取http请求对象，会自动的处理签名和验签，
 * 并进行证书自动更新
 *
 * @return CloseableHttpClient
 */
@Bean("wechatPayClient")
public CloseableHttpClient getWechatPayClient(Verifier verifier) throws IOException {
    WechatPayHttpClientBuilder builder = WechatPayHttpClientBuilder.create().withMerchant(payConfig.getMchId(),
payConfig.getMchSerialNo(), getPrivateKey()).withValidator(new WechatPay2Validator(verifier));
        // 通过WechatPayHttpClientBuilder构造的HttpClient，会自动的处理签名和验签，并进行证书自动更新
        return builder.build();
    }
}
```

4.微信点歌单支付接口 `url` 常量配置

```
public class WechatPayConstant {

    /**
     * 微信支付主机地址
     */
    public static final String HOST = "https://api.mch.weixin.qq.com";

    /**
     * Native下单
     */
    public static final String NATIVE_ORDER = HOST + "/v3/pay/transactions/native";

    /**
     * Native订单状态查询，根据商户订单号查询
     */
public static final String NATIVE_QUERY = HOST + "/v3/pay/transactions/out-trade-no/%s?mchid=%s";

    /**
     * 关闭订单接口
     */
public static final String NATIVE_CLOSE_ORDER = HOST + "/v3/pay/transactions/out-trade-no/%s/close";

    /**
     * 申请退款接口
     */
public static final String NATIVE_REFUND_ORDER = HOST + "/v3/refund/domestic/refunds";

    /**
     * 退款状态查询接口
     */
public static final String NATIVE_REFUND_QUERY = HOST + "/v3/refund/domestic/refunds/%s";
```

```
}
```

5.微信支付

```
@Slf4j
@Service
public class WeChatServiceImpl implements WeChatService {
    @Resource
    private WechatPropertiesV3 properties;
}
```

```
/**
 * http公用方法
 *
 * @author user
 */
@Slf4j
public class HttpUtils {
@Resource
private static CloseableHttpClient wechatPayClient;

public static String doPost(String , String url) {
    log.info("请求参数:{}", );
    String responseStr = "";
    StringEntity entity = new StringEntity(, "utf-8");
    entity.setContentType("application/json");
    HttpPost httpPost = new HttpPost(url);
    httpPost.setHeader("Accept", "application/json");
    httpPost.setEntity(entity);
    try (CloseableHttpResponse response = wechatPayClient.execute(httpPost)) {
        //响应码
        int statusCode = response.getStatusLine().getStatusCode();
        //响应体
        responseStr = EntityUtils.toString(response.getEntity());
        if (statusCode == 200) {
            //处理成功
            log.info("成功啦,响应码:{},返回结果:{}", statusCode, responseStr);
        } else if (statusCode == 204) {
            //处理成功，无返回
            log.info("成功啦,响应码:{},无返回结果", statusCode);
        } else {
            log.info("失败,响应码:{},返回结果:{}", statusCode, responseStr);
            log.info("微信支付错误信息" + responseStr);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return responseStr;
}

public static String doGet(String url) {
    String responseStr = "";
    HttpGet httpGet = new HttpGet(url);
    httpGet.setHeader("Accept", "application/json");
    try (CloseableHttpResponse response = wechatPayClient.execute(httpGet)) {
        //响应码
        int statusCode = response.getStatusLine().getStatusCode();
        //响应体
```

```
            responseStr = EntityUtils.toString(response.getEntity());
            log.info("查询响应码:{},返回结果:{}", statusCode, responseStr);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return responseStr;
    }
}
```

```
/**
 * 另一个工具类
 * 微信订单工具类
 *
 * @author user
 */
@Slf4j
public class WechatOrderUtils {
    public static final String ORDER_PAY_PREFIX = "ray_pay_";
    public static final String ORDER_REFUND_PREFIX = "ray_refund_";
    public static final String FORMAT = "yyyyMMddHHmmssSSS";

    /**
     * 生成商户订单号
     *
     * @return
     */
    public static String genOutTradeNo() {
        String order = ORDER_PAY_PREFIX + new SimpleDateFormat(FORMAT).format(new Date());
        log.debug("生成商户订单号: [{}]", order);
        return order;
    }

    /**
     * 生成退款订单号
     *
     * @return
     */
    public static String genRefundOrder() {
        String order = ORDER_REFUND_PREFIX + new SimpleDateFormat(FORMAT).format(new Date());
        log.debug("生成退款订单号: [{}]", order);
        return order;
    }
}
```

```
/**
 * 微信支付的一些信息,演示用
 *
 * @author user
 */
@Data
public class PayInfoVO {
    private String outTradeNo;
    private String description;
    private Integer total;
    private String outRefundNo;
    private String reason;
    private Integer refund;
}
```

5.1 下单

```
@Override
public String nativeOrder(PayInfoVO vo) {
    JSONObject payObj = new JSONObject();
    payObj.put("mchid", properties.getMchId());
    payObj.put("out_trade_no", vo.getOutTradeNo());
    payObj.put("appid", properties.getAppId());
    payObj.put("description", vo.getDescription());
    payObj.put("notify_url", properties.getCallbackUrl());
    JSONObject amountObj = new JSONObject();
    amountObj.put("total", vo.getTotal());
    amountObj.put("currency", "CNY");
    payObj.put("amount", amountObj);
    String body = payObj.toJSONString();
    return HttpUtils.doPost(body, WechatPayConstant.NATIVE_ORDER);
}
```

请求参数

参数名	变量	类型[长度限制]	必填	描述
应用ID	appid	string[1,32]	是	由微信生成的应用ID，全局唯一。 请求基础下单接口时请注意APPID的应用属性， 例如公众号场景下，需使用应用属性为公众号的APPID 示例值：wxd678efh567hg6787
直连商户号	mchid	string[1,32]	是	直连商户的商户号，由微信支付生成并下发。 示例值：1230000109
商品描述	description	string[1,127]	是	商品描述 示例值：Image形象店-深圳腾大-QQ公仔
商户订单号	out_trade_no	string[6,32]	是	商户系统内部订单号，只能是数字、大小写字母_-*且在同一个商户号下唯一 示例值：1217752501201407033233368018
交易结束时间	time_expire	string[1,64]	否	订单失效时间，遵循rfc3339标准格式，格式为yyyy-MM-DDTHH:mm:ss+TIMEZONE， yyyy-MM-DD表示年月日，T出现在字符串中，表示time元素的开头， HH:mm:ss表示时分秒，TIMEZONE表示时区 （+08:00表示东八区时间，领先UTC8小时，即北京时间）。 例如：2015-05-20T13:29:35+08:00表示， 北京时间2015年5月20日 13点29分35秒。 订单失效时间是针对订单号而言的， 由于在请求支付的时候有一个必传参数prepay_id只有两小时的有效期， 所以在重入时间超过2小时的时候需要重新请求下单接口获取新的prepay_id。 建议：最短失效时间间隔大于1分钟 示例值：2018-06-08T10:34:56+08:00
附加数据	attach	string[1,128]	否	附加数据，在查询API和支付通知中原样返回，可作为自定义参数使用， 实际情况下只有支付完成状态才会返回该字段。 示例值：自定义数据
通知地址	notify_url	string[1,256]	是	通知URL必须为直接可访问的URL，不允许携带查询串，要求必须为https地址。 格式：URL 示例值：https://www.weixin.qq.com/wxpay/pay.php
订单优惠标记	goods_tag	string[1,32]	否	订单优惠标记 示例值：WXG
- 订单金额	amount	object	是	订单金额信息
总金额	total	int	是	订单总金额，单位为分。 示例值：100
货币类型	currency	string[1,16]	否	CNY：人民币，境内商户号仅支持人民币。 示例值：CNY
- 优惠功能	detail	object	否	优惠功能
订单原价	cost_price	int	否	1、商户侧一张小票订单可能被分多次支付，订单原价用于记录整张小票的交易金额。 2、当订单原价与支付金额不相等，则不享受优惠。 3、该字段主要用于防止同一张小票分多次支付，以享受多次优惠的情况，正常支付订单不必上传此参数。 示例值：608800
商品小票ID	invoice_id	string[1,32]	否	商家小票ID 示例值：微信123
-- 单品列表	goods_detail	array	否	单品列表信息 条目个数限制：【1，6000】
-- 商户侧商品编码	merchant_goods_id	string[1,32]	是	由半角的大小写字母、数字、中划线、下划线中的一种或几种组成。 示例值：1246464644
-- 微信支付商品编码	wechatpay_goods_id	string[1,32]	否	微信支付定义的统一商品编号（没有可不传） 示例值：1001
-- 商品名称	goods_name	string[1,256]	否	商品的实际名称 示例值：iPhoneX 256G
-- 商品数量	quantity	int	是	用户购买的数量 示例值：1
-- 商品单价	unit_price	int	是	商品单价，单位为分 示例值：828800
- 场景信息	scene_info	object	否	支付场景描述
用户终端IP	payer_client_ip	string[1,45]	是	用户的客户端IP，支持IPv4和IPv6两种格式的IP地址。 示例值：14.23.150.211
商户端设备号	device_id	string[1,32]	否	商户端设备号（门店号或收银设备ID）。 示例值：013467007045764
-- 商户门店信息	store_info	object	否	商户门店信息
-- 门店编号	id	string[1,32]	是	商户侧门店编号 示例值：0001
-- 门店名称	name	string[1,256]	否	商户侧门店名称 示例值：腾讯大厦分店
-- 地区编码	area_code	string[1,32]	否	地区编码，详细请见省市区编号对照表。 示例值：440305
-- 详细地址	address	string[1,512]	否	详细的商户门店地址 示例值：广东省深圳市南山区科技中一道10000号

参数名	变量	类型[长度限制]	必填	描述
- 结算信息	settle_info	object	否	结算信息
是否指定分账	profit_sharing	boolean	否	是否指定分账 示例值：false

返回参数

参数名	变量	类型[长度限制]	必填	描述
二维码链接	code_url	string[1,512]	是	此URL用于生成支付二维码，然后提供给用户扫码支付。 注意：code_url并非固定值，使用时按照URL格式转成二维码即可。 示例值：weixin://wxpay/bizpayurl/up?pr=NwY5Mz9&groupid=00

5.2 查询订单

```
@Override
public void nativeQuery(String outTradeNo) {
    String url = String.format(WechatPayConstant.NATIVE_QUERY, outTradeNo, properties.getMchId());
    HttpUtils.doGet(url);
}
```

请求参数

参数名	变量	类型[长度限制]	必填	描述
直连商户号	mchid	string[1,32]	是	直连商户的商户号，由微信支付生成并下发。 示例值：1230000109
商户订单号	out_trade_no	string[6,32]	是	商户系统内部订单号，只能是数字、大小写字母_-*且在同一个商户号下唯一。 特殊规则：最小字符长度为6 示例值：1217752501201407033233368018

返回参数

参数名	变量	类型[长度限制]	必填	描述
应用ID	appid	string[1,32]	是	直连商户申请的公众号或移动应用appid。 示例值：wxd678efh567hg6787
直连商户号	mchid	string[1,32]	是	直连商户的商户号，由微信支付生成并下发。 示例值：1230000109
商户订单号	out_trade_no	string[6,32]	是	商户系统内部订单号，只能是数字、大小写字母_-*且在同一个商户号下唯一， 详见【商户订单号】。 示例值：1217752501201407033233368018
微信支付订单号	transaction_id	string[1,32]	否	微信支付系统生成的订单号。 示例值：1217752501201407033233368018
交易类型	trade_type	string[1,16]	否	交易类型，枚举值： JSAPI：公众号支付 NATIVE：扫码支付 APP：APP支付 MICROPAY：付款码支付 MWEB：H5支付 FACEPAY：刷脸支付 示例值：MICROPAY
交易状态	trade_state	string[1,32]	是	交易状态，枚举值： SUCCESS：支付成功 REFUND：转入退款 NOTPAY：未支付 CLOSED：已关闭 REVOKED：已撤销（仅付款码支付会返回） USERPAYING：用户支付中（仅付款码支付会返回） PAYERROR：支付失败（仅付款码支付会返回） 示例值：SUCCESS
交易状态描述	trade_state_desc	string[1,256]	是	交易状态描述 示例值：支付成功
付款银行	bank_type	string[1,32]	否	银行类型，采用字符串类型的银行标识。 银行标识请参考《银行类型对照表》 示例值：CMC
附加数据	attach	string[1,128]	否	附加数据，在查询API和支付通知中原样返回， 可作为自定义参数使用，实际情况下只有支付完成状态才会返回该字段。 示例值：自定义数据
支付完成时间	success_time	string[1,64]	否	支付完成时间，遵循rfc3339标准格式，格式为yyyy-MM-DDTHH:mm:ss+TIMEZONE， yyyy-MM-DD表示年月日，T出现在字符串中，表示time元素的开头， HH:mm:ss表示时分秒，TIMEZONE表示时区 （+08:00表示东八区时间，领先UTC 8小时，即北京时间）。 例如：2015-05-20T13:29:35+08:00表示，北京时间2015年5月20日 13点29分35秒。 示例值：2018-06-08T10:34:56+08:00
- 支付者	payer	object	是	支付者信息
用户标识	openid	string[1,128]	是	用户在直连商户appid下的唯一标识。 示例值：oUpF8uMuAJO_M2pxb1Q9zNjWeS6o
- 订单金额	amount	object	否	订单金额信息，当支付成功时返回该字段。
总金额	total	int	否	订单总金额，单位为分。 示例值：100
- 场景信息	scene_info	object	否	支付场景描述
商户端设备号	device_id	string[1,32]	否	商户端设备号（发起扣款请求的商户服务器设备号）。 示例值：013467007045764
- 优惠功能	promotion_detail	array	否	优惠功能，享受优惠时返回该字段。
券ID	coupon_id	string[1,32]	是	券ID 示例值：109519
优惠名称	name	string[1,64]	否	优惠名称 示例值：单品惠-6
优惠范围	scope	string[1,32]	否	GLOBAL：全场代金券 SINGLE：单品优惠 示例值：GLOBAL
优惠类型	type	string[1,32]	否	CASH：充值型代金券 NOCASH：免充值型代金券 示例值：CASH
优惠券面额	amount	int	是	优惠券面额 示例值：100
活动ID	stock_id	string[1,32]	否	活动ID 示例值：931386
微信出资	wechatpay_contribute	int	否	微信出资，单位为分 示例值：0
商户出资	merchant_contribute	int	否	商户出资，单位为分 示例值：0

参数名	变量	类型[长度限制]	必填	描述
其他出资	other_contribute	int	否	其他出资，单位为分 示例值：0
优惠币种	currency	string[1,16]	否	CNY：人民币，境内商户号仅支持人民币。 示例值：CNY
-- 单品列表	goods_detail	array	否	单品列表信息
-- 商品编码	goods_id	string[1,32]	是	商品编码 示例值：M1006
-- 商品数量	quantity	int	是	用户购买的数量 示例值：1
-- 商品单价	unit_price	int	是	商品单价，单位为分 示例值：100
-- 商品优惠金额	discount_amount	int	是	商品优惠金额 示例值：0
-- 商品备注	goods_remark	string[1,128]	否	商品备注信息 示例值：商品备注信息

5.3 关闭订单

```
@Override
public void nativeCloseOrder(String outTradeNo) {
    String url = String.format(WechatPayConstant.NATIVE_CLOSE_ORDER, outTradeNo);
    JSONObject payObj = new JSONObject();
    payObj.put("mchid", properties.getMchId());
    String body = payObj.toJSONString();
    HttpUtils.doPost(body, url);
}
```

请求参数

参数名	变量	类型[长度限制]	必填	描述
直连商户号	mchid	string[1,32]	是	直连商户的商户号，由微信支付生成并下发。 示例值：1230000109
商户订单号	out_trade_no	string[6,32]	是	path 商户系统内部订单号， 只能是数字、大小写字母_-*且在同一个商户号下唯一 示例值：1217752501201407033233368018

5.4 退款

```
@Override
public void nativeRefundOrder(PayInfoVO vo) {
    //  商户订单号
    String outTradeNo = WechatOrderUtils.genOutTradeNo();
    //  商户退款单号
    String refundNo = WechatOrderUtils.genRefundOrder();
    // 请求body参数
    JSONObject refundObj = new JSONObject();
    //订单号
    refundObj.put("out_trade_no", vo.getOutTradeNo());
    refundObj.put("out_refund_no", vo.getOutRefundNo());
    refundObj.put("reason", vo.getReason());
    refundObj.put("notify_url", properties.getCallbackUrl());
    JSONObject amountObj = new JSONObject();
    //退款金额
    amountObj.put("refund", vo.getRefund());
    //实际支付的总金额
    amountObj.put("total", vo.getTotal());
```

```
    amountObj.put("currency", "CNY");
    refundObj.put("amount", amountObj);
    String body = refundObj.toJSONString();
    HttpUtils.doPost(body, WechatPayConstant.NATIVE_REFUND_ORDER);
}
```

请求参数

参数名	变量	类型[长度限制]	必填	描述
微信支付订单号	transaction_id	string[1, 32]	二选一	原支付交易对应的微信订单号 示例值：1217752501201407033233368018
商户订单号	out_trade_no	string[6, 32]	二选一	原支付交易对应的商户订单号 示例值：1217752501201407033233368018
商户退款单号	out_refund_no	string[1, 64]	是	商户系统内部的退款单号， 商户系统内部唯一， 只能是数字、大小写字母_-|*@ ， 同一退款单号多次请求只退一笔。 示例值：1217752501201407033233368018
退款原因	reason	string[1, 80]	否	若商户传入， 会在下发给用户的退款消息中体现退款原因 示例值：商品已售完
退款结果回调url	notify_url	string[8, 256]	否	异步接收微信支付退款结果通知的回调地址， 通知url必须为外网可访问的url， 不能携带参数。 如果参数中传了notify_url， 则商户平台上配置的回调地址将不会生效， 优先回调当前传的这个地址。 示例值：https://weixin.qq.com
退款资金来源	funds_account	string[1,32]	否	若传递此参数则使用对应的资金账户退款 否则默认使用未结算资金退款 （仅对老资金流商户适用） 枚举值： AVAILABLE：可用余额账户 示例值：AVAILABLE
- 金额信息	amount	object	是	订单金额信息
退款金额	refund	int	是	退款金额，单位为分， 只能为整数，不能超过原订单支付金额。 示例值：888
-- 退款出资账户及金额	from	array	否	退款需要从指定账户出资时， 传递此参数指定出资金额 （币种的最小单位，只能为整数）。 同时指定多个账户出资退款的使用场景 需要满足以下条件 1、未开通退款支出分离产品功能； 2、订单属于分账订单， 且分账处于待分账或分账中状态。 参数传递需要满足条件： 1、基本账户可用余额出资金额 与基本账户不可用余额出资金额 之和等于退款金额； 2、账户类型不能重复。 上述任一条件不满足将返回错误
--出资账户类型	account	string[1, 32]	是	下面枚举值多选一。 枚举值： AVAILABLE : 可用余额 UNAVAILABLE : 不可用余额 示例值：AVAILABLE
--出资金额	amount	int	是	对应账户出资金额 示例值：444
原订单金额	total	int	是	原支付交易的订单总金额， 单位为分，只能为整数。 示例值：888
退款币种	currency	string[1, 16]	是	符合ISO 4217标准的三位字母代码， 目前只支持人民币：CNY。 示例值：CNY
- 退款商品	goods_detail	array	否	指定商品退款需要传此参数， 其他场景无需传递
商户侧商品编码	merchant_goods_id	string[1, 32]	是	由半角的大小写字母、数字、 中划线、下划线中的一种或几种组成 示例值：1217752501201407033233368018
微信支付商品编码	wechatpay_goods_id	string[1, 32]	否	微信支付定义的统一商品编号 （没有可不传） 示例值：1001
商品名称	goods_name	string[1, 256]	否	商品的实际名称 示例值：iPhone6s 16G

参数名	变量	类型[长度限制]	必填	描述
商品单价	unit_price	int	是	商品单价金额，单位为分 示例值：528800
商品退款金额	refund_amount	int	是	商品退款金额，单位为分 示例值：528800
商品退货数量	refund_quantity	int	是	单品的退款数量 示例值：1

返回参数

参数名	变量	类型[长度限制]	必填	描述
微信支付退款单号	refund_id	string[1, 32]	是	微信支付退款单号 示例值：50000000382019052709732678859
商户退款单号	out_refund_no	string[1, 64]	是	商户系统内部的退款单号，商户系统内部唯一， 只能是数字、大小写字母_-|*@ ， 同一退款单号多次请求只退一笔。 示例值：1217752501201407033233368018
微信支付订单号	transaction_id	string[1, 32]	是	微信支付交易订单号 示例值：1217752501201407033233368018
商户订单号	out_trade_no	string[1, 32]	是	原支付交易对应的商户订单号 示例值：1217752501201407033233368018
退款渠道	channel	string[1, 16]	是	枚举值： ORIGINAL：原路退款 BALANCE：退回到余额 OTHER_BALANCE：原账户异常退到其他余额账户 OTHER_BANKCARD：原银行卡异常退到其他银行卡 示例值：ORIGINAL
退款入账账户	user_received_account	string[1, 64]	是	取当前退款单的退款入账方， 有以下几种情况： 1）退回银行卡：{银行名称}{卡类型}{卡尾号} 2）退回支付用户零钱:支付用户零钱 3）退还商户:商户基本账户商户结算银行账户 4）退回支付用户零钱通:支付用户零钱通 示例值：招商银行信用卡0403
退款成功时间	success_time	string[1, 64]	否	退款成功时间，当退款状态为退款成功时有返回。 示例值：2020-12-01T16:18:12+08:00
退款创建时间	create_time	string[1, 64]	是	退款受理时间 示例值：2020-12-01T16:18:12+08:00
退款状态	status	string[1, 32]	是	退款到银行发现用户的卡作废或者冻结了， 导致原路退款银行卡失败， 可前往商户平台-交易中心， 手动处理此笔退款。 枚举值： SUCCESS：退款成功 CLOSED：退款关闭 PROCESSING：退款处理中 ABNORMAL：退款异常 示例值：SUCCESS
资金账户	funds_account	string[1, 32]	否	退款所使用资金对应的资金账户类型 枚举值： UNSETTLED : 未结算资金 AVAILABLE : 可用余额 UNAVAILABLE : 不可用余额 OPERATION : 运营户 BASIC : 基本账户（含可用余额和不可用余额） 示例值：UNSETTLED
- 金额信息	amount	object	是	金额详细信息
订单金额	total	int	是	订单总金额，单位为分 示例值：100
退款金额	refund	int	是	退款标价金额，单位为分，可以做部分退款 示例值：100
-- 退款出资账户及金额	from	array	否	退款出资的账户类型及金额信息
-- 出资账户类型	account	string[1, 32]	是	下面枚举值多选一。 枚举值： AVAILABLE : 可用余额 UNAVAILABLE : 不可用余额 示例值：AVAILABLE
-- 出资金额	amount	int	是	对应账户出资金额 示例值：444
用户支付金额	payer_total	int	是	现金支付金额，单位为分，只能为整数 示例值：90
用户退款金额	payer_refund	int	是	退款给用户的金额，不包含所有优惠券金额 示例值：90

参数名	变量	类型[长度限制]	必填	描述
应结退款金额	settlement_refund	int	是	去掉非充值代金券退款金额后的退款金额， 单位为分， 退款金额=申请退款金额-非充值代金券退款金额， 退款金额<=申请退款金额 示例值：100
应结订单金额	settlement_total	int	是	应结订单金额=订单金额-免充值代金券金额， 应结订单金额<=订单金额， 单位为分 示例值：100
优惠退款金额	discount_refund	int	是	优惠退款金额<=退款金额， 退款金额-代金券或立减优惠退款金额为现金， 说明详见代金券或立减优惠，单位为分 示例值：10
退款币种	currency	string[1, 16]	是	符合ISO 4217标准的三位字母代码， 目前只支持人民币：CNY。 示例值：CNY
- 优惠退款信息	promotion_detail	array	否	优惠退款信息
券ID	promotion_id	string[1, 32]	是	券或者立减优惠id 示例值：109519
优惠范围	scope	string[1, 32]	是	枚举值： GLOBAL：全场代金券 SINGLE：单品优惠 示例值：SINGLE
优惠类型	type	string[1, 32]	是	枚举值： COUPON：代金券，需要走结算资金的充值型代金券 DISCOUNT：优惠券，不走结算资金的免充值型优惠券 示例值：DISCOUNT
优惠券面额	amount	int	是	用户享受优惠的金额 （优惠券面额=微信出资金额+商家出资金额+其他出资方金额）， 单位为分 示例值：5
优惠退款金额	refund_amount	int	是	优惠退款金额<=退款金额， 退款金额-代金券或立减优惠退款金额为用户支付的现金， 说明详见代金券或立减优惠， 单位为分 示例值：100
-- 商品列表	goods_detail	array	否	优惠商品发生退款时返回商品信息
-- 商户侧商品编码	merchant_goods_id	string[1, 32]	是	由半角的大小写字母、数字、中划线、下划线中的一种或几种组成 示例值：1217752501201407033233368018
-- 微信支付商品编码	wechatpay_goods_id	string[1, 32]	否	微信支付定义的统一商品编号（没有可不传） 示例值：1001
-- 商品名称	goods_name	string[1, 256]	否	商品的实际名称 示例值：iPhone6s 16G
-- 商品单价	unit_price	int	是	商品单价金额，单位为分 示例值：528800
-- 商品退款金额	refund_amount	int	是	商品退款金额，单位为分 示例值：528800
-- 商品退货数量	refund_quantity	int	是	单品的退款数量 示例值：1

5.5 退款查询

```
@Override
public void nativeRefundOrderQuery(String refundNo) {
    String url = String.format(WechatPayConstant.NATIVE_REFUND_QUERY, refundNo);
    HttpUtils.doGet(url);
}
```

请求参数

参数名	变量	类型[长度限制]	必填	描述
商户退款单号	out_refund_no	string[1, 64]	是	商户系统内部的退款单号，商户系统内部唯一， 只能是数字、大小写字母_-|*@ ， 同一退款单号多次请求只退一笔。 示例值：1217752501201407033233368018

返回参数

参数名	变量	类型[长度限制]	必填	描述
微信支付退款单号	refund_id	string[1, 32]	是	微信支付退款单号 示例值：50000000382019052709732678859
商户退款单号	out_refund_no	string[1, 64]	是	商户系统内部的退款单号，商户系统内部唯一， 只能是数字、大小写字母_-|*@ ， 同一退款单号多次请求只退一笔。 示例值：1217752501201407033233368018
微信支付订单号	transaction_id	string[1, 32]	是	微信支付交易订单号 示例值：1217752501201407033233368018
商户订单号	out_trade_no	string[1, 32]	是	原支付交易对应的商户订单号 示例值：1217752501201407033233368018
退款渠道	channel	string[1, 16]	是	枚举值： ORIGINAL：原路退款 BALANCE：退回到余额 OTHER_BALANCE：原账户异常退到其他余额账户 OTHER_BANKCARD：原银行卡异常退到其他银行卡 示例值：ORIGINAL
退款入账账户	user_received_account	string[1, 64]	是	取当前退款单的退款入账方，有以下几种情况： 1）退回银行卡：{银行名称}{卡类型}{卡尾号} 2）退回支付用户零钱:支付用户零钱 3）退还商户:商户基本账户商户结算银行账户 4）退回支付用户零钱通:支付用户零钱通 示例值：招商银行信用卡0403
退款成功时间	success_time	string[1, 64]	否	退款成功时间，当退款状态为退款成功时有返回。 示例值：2020-12-01T16:18:12+08:00
退款创建时间	create_time	string[1, 64]	是	退款受理时间 示例值：2020-12-01T16:18:12+08:00
退款状态	status	string[1, 32]	是	款到银行发现用户的卡作废或者冻结了， 导致原路退款银行卡失败， 可前往商户平台-交易中心， 手动处理此笔退款。 枚举值： SUCCESS：退款成功 CLOSED：退款关闭 PROCESSING：退款处理中 ABNORMAL：退款异常 示例值：SUCCESS
资金账户	funds_account	string[1, 32]	否	退款所使用资金对应的资金账户类型 枚举值： UNSETTLED : 未结算资金 AVAILABLE : 可用余额 UNAVAILABLE : 不可用余额 OPERATION : 运营户 BASIC : 基本账户（含可用余额和不可用余额） 示例值：UNSETTLED
- 金额信息	amount	object	是	金额详细信息
订单金额	total	int	是	订单总金额，单位为分 示例值：100
退款金额	refund	int	是	退款标价金额，单位为分，可以做部分退款 示例值：100
-- 退款出资账户及金额	from	array	否	退款出资的账户类型及金额信息
-- 出资账户类型	account	string[1, 32]	是	下面枚举值多选一。 枚举值： AVAILABLE : 可用余额 UNAVAILABLE : 不可用余额 示例值：AVAILABLE
-- 出资金额	amount	int	是	对应账户出资金额 示例值：444
用户支付金额	payer_total	int	是	现金支付金额，单位为分，只能为整数 示例值：90
用户退款金额	payer_refund	int	是	退款给用户的金额，不包含所有优惠券金额 示例值：90

参数名	变量	类型[长度限制]	必填	描述
应结退款金额	settlement_refund	int	是	去掉非充值代金券退款金额后的退款金额， 单位为分， 退款金额=申请退款金额-非充值代金券退款金额， 退款金额<=申请退款金额 示例值：100
应结订单金额	settlement_total	int	是	应结订单金额=订单金额-免充值代金券金额， 应结订单金额<=订单金额，单位为分 示例值：100
优惠退款金额	discount_refund	int	是	优惠退款金额<=退款金额， 退款金额-代金券或立减优惠退款金额为现金， 说明详见代金券或立减优惠， 单位为分 示例值：10
退款币种	currency	string[1, 16]	是	符合ISO 4217标准的三位字母代码， 目前只支持人民币：CNY。 示例值：CNY
- 优惠退款信息	promotion_detail	array	否	优惠退款信息
券ID	promotion_id	string[1, 32]	是	券或者立减优惠id 示例值：109519
优惠范围	scope	string[1, 32]	是	枚举值： GLOBAL：全场代金券 SINGLE：单品优惠 示例值：SINGLE
优惠类型	type	string[1, 32]	是	枚举值： COUPON：代金券，需要走结算资金的充值型代金券 DISCOUNT：优惠券，不走结算资金的免充值型优惠券 示例值：DISCOUNT
优惠券面额	amount	int	是	用户享受优惠的金额 （优惠券面额=微信出资金额+商家出资金额+其他出资方金额）， 单位为分 示例值：5
优惠退款金额	refund_amount	int	是	优惠退款金额<=退款金额， 退款金额-代金券或立减优惠退款金额为用户支付的现金， 说明详见代金券或立减优惠， 单位为分 示例值：100
-- 商品列表	goods_detail	array	否	优惠商品发生退款时返回商品信息
-- 商户侧商品编码	merchant_goods_id	string[1, 32]	是	由半角的大小写字母、数字、中划线、下划线中的一种或几种组成 示例值：1217752501201407033233368018
-- 微信支付商品编码	wechatpay_goods_id	string[1, 32]	否	微信支付定义的统一商品编号（没有可不传） 示例值：1001
-- 商品名称	goods_name	string[1, 256]	否	商品的实际名称 示例值：iPhone6s 16G
-- 商品单价	unit_price	int	是	商品单价金额，单位为分 示例值：528800
-- 商品退款金额	refund_amount	int	是	商品退款金额，单位为分 示例值：528800
-- 商品退货数量	refund_quantity	int	是	单品的退款数量 示例值：1

6.微信支付V3版本回调+验签流程

6.1 回调验签流程介绍

官方文档

https://pay.weixin.qq.com/wiki/doc/apiv3/apis/chapter3_4_5.shtml

https://pay.weixin.qq.com/wiki/doc/apiv3/wechatpay/wechatpay4_1.shtml

> 注意:
> 同样的通知可能会多次发送给商户系统，商户系统必须能够正确处理重复的通知
> 确保回调URL是外部可正常访问的，且不能携带后缀参数

> 微信回调通知重复问题（不一定准确按照时间间隔推送，需要保证幂等性处理）
> 重复通知的时候，微信的请求id是一样的，用这个做请求幂等性处理
> 响应给微信的内容不规范 或者 超过5秒没响应
> 测试的问题：如果有多个未响应的，则测试的请求 `id` ，可能有之前的请求继续回调过来

6.2 核心流程操作

1. 获取报文
2. 验证签名（确保是微信传输过来的）
3. 解密 `AES` 对称解密出原始数据）
4. 处理业务逻辑
5. 响应请求

```
# 支付成功，回调通知
wx:
  callback-url: http://我们的域名或者ip/api/callback/order/v3/wechat
```

```
@RestController
@RequestMapping("/api/callback/order/v3/")
@Slf4j
public class PayCallbackController {
    @Autowired
    private WechatPropertiesV3 wechatPayConfig;
    @Autowired
    private CertificatesVerifier verifier;

    /**
     * * 获取报文
     * <p>
     * * 验证签名（确保是微信传输过来的）
     * <p>
     * * 解密（AES对称解密出原始数据）
     * <p>
     * * 处理业务逻辑
     * <p>
     * * 响应请求
     *
     * @param request  request
     * @param response response
     * @return Map
     */
    @PostMapping("wechat")
    public Map<String, String> wehcatPayCallback(HttpServletRequest request, HttpServletResponse response) {
        //获取报文
        String body = getRequestBody(request);
        //随机串
        String nonceStr = request.getHeader("Wechatpay-Nonce");
        //微信传递过来的签名
        String signature = request.getHeader("Wechatpay-Signature");
        //证书序列号（微信平台）
        String serialNo = request.getHeader("Wechatpay-Serial");
        //时间戳
        String timestamp = request.getHeader("Wechatpay-Timestamp");
        //构造签名串
        //应答时间戳\n
```

```
        //应答随机串\n
        //应答报文主体\n
        String signStr = Stream.of(timestamp, nonceStr, body).collect(Collectors.joining("\n", "", "\n"));
        Map<String, String> map = new HashMap<>(2);
        try {
            //验证签名是否通过
            boolean result = verifiedSign(serialNo, signStr, signature);
            if (result) {
                //解密数据
                String plainBody = decryptBody(body);
                log.info("解密后的明文:{}", plainBody);
                Map<String, String> paramsMap = convertWechatPayMsgToMap(plainBody);
                //处理业务逻辑

                //响应微信
                map.put("code", "SUCCESS");
                map.put("message", "成功");
            }
        } catch (Exception e) {
            log.error("微信支付回调异常:{}", e);
        }
        return map;
    }

    /**
     * 转换body为map
     *
     * @param plainBody plainBody
     * @return Map
     */
    private Map<String, String> convertWechatPayMsgToMap(String plainBody) {
        Map<String, String> paramsMap = new HashMap<>(2);
        JSONObject jsonObject = JSONObject.parseObject(plainBody);
        //商户订单号
        paramsMap.put("out_trade_no", jsonObject.getString("out_trade_no"));
        //交易状态
        paramsMap.put("trade_state", jsonObject.getString("trade_state"));
        //附加数据
        paramsMap.put("account_no", jsonObject.getJSONObject("attach").getString("accountNo"));
        return paramsMap;
    }

    /**
     * 解密body的密文
     *
     * @param body body
     * @return String
     */
    private String decryptBody(String body) throws GeneralSecurityException {
        AesUtil aesUtil = new AesUtil(wechatPayConfig.getApiV3Key().getBytes(StandardCharsets.UTF_8));
        JSONObject object = JSONObject.parseObject(body);
        JSONObject resource = object.getJSONObject("resource");
        String ciphertext = resource.getString("ciphertext");
        String associatedData = resource.getString("associated_data");
        String nonce = resource.getString("nonce");
        return aesUtil.decryptToString(associatedData.getBytes(StandardCharsets.UTF_8),
nonce.getBytes(StandardCharsets.UTF_8), ciphertext);
    }
```

```
    /**
     * 验证签名
     *
     * @param serialNo  微信平台-证书序列号
     * @param signStr   自己组装的签名串
     * @param signature 微信返回的签名
     * @return boolean
     */
    private boolean verifiedSign(String serialNo, String signStr, String signature) {
        return verifier.verify(serialNo, signStr.getBytes(StandardCharsets.UTF_8), signature);
    }

    /**
     * 读取请求数据流
     *
     * @param request request
     * @return String
     */
    private String getRequestBody(HttpServletRequest request) {
        StringBuilder sb = new StringBuilder();
        try (ServletInputStream inputStream = request.getInputStream(); BufferedReader reader = new
BufferedReader(new InputStreamReader(inputStream))) {
            String line;
            while ((line = reader.readLine()) != null) {
                sb.append(line);
            }
        } catch (IOException e) {
            log.error("读取数据流异常:{}", e);
        }
        return sb.toString();
    }
}
```