统一支付平台接入手册
供对接使用

杭州恒生芸泰网络科技有限公司

研发部

说　明

文档修改记录

版本	修订人	修订说明	批准人	发布日期
1.0	金仁	初始版本	邱峰	**2019-08-05**
1.1	金仁	补充参数说明	邱峰	**2019-08-07**
1.2	邱峰	调整格式，修正部分描述		**2019-08-07**
1.3	金仁	修改参数说明	邱峰	**2019-08-08**
1.4	邱峰	修改格式及部分描述		**2019-08-09**
1.5	金仁	修改配置部分描述	邱峰	**2019-08-09**
1.6	邱峰	格式调整、配置及服务实现部分修改		**2019-08-11**
1.7	金仁	增加枚举及部分说明	邱峰	**2019-08-13**
1.8	邱峰	修改服务实现部分参数及说明		**2019-08-13**
1.9	邱峰	修改 **git** 下载部分描述		**2019-08-18**
2.0	邱峰	实时订单接口修改参数		**2019-08-19**
2.1	金仁	增加网络测试接口描述	邱峰	**2019-08-22**
2.2	邱峰	删除 **HDP** 配置中启用禁用描述		**2019-08-23**
2.3	金仁	增加 **Q&A** 模块	邱峰	**2019-08-26**
2.4	邱峰	增加支付及退款结果通知		**2019-09-06**
2.5	邱峰	增加对接工程命名规范		**2019-09-20**
2.6	金仁	调整部分接口字段备注		**2019-10-18**
2.7	金仁	修改部分接口截图及描述		**2019-11-01**

目录

1 概述

1.1 文档说明

该文档由统一支付平台小组成员编写，为对接人员接入支付功能提供全面而详细的说明。文档中包含接入的具体步骤、接口的详细说明、枚举项和响应说明。具体内容参见具体章节。

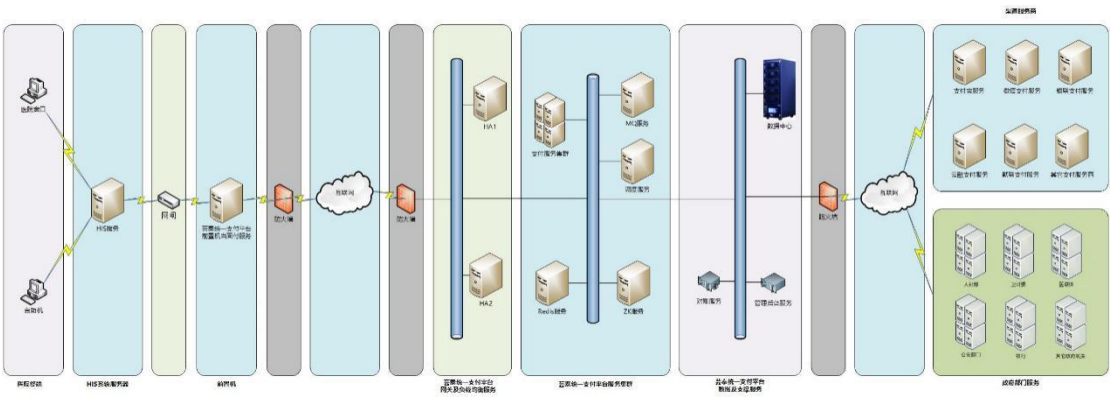
此份文档为芸泰网络公司内部文档，任何成员不可外泄。

1.2 名词解释

名词	解释
UPP	统一支付平台简称。以下简称平台
商户	统一支付平台的最终服务对象，支付宝、微信等支付账户的所有者。在公司内部通常一个商户代表一家医院。
ISV	统一支付平台的接入公司，业务系统服务提供方。在公司内部，北京和杭州等因服务部署区分而分配不同 ISV 信息。
商户客户端	商户的客户端产品，如：商户 APP、支付宝服务窗、微信公众号等。在公司内部即为单院 APP、云医院 APP、支付宝服务窗、微信公众号等产品。以下简称客户端。
商户服务端	商户的服务提供者，也是支付服务调用者。在公司内部即为挂号服务、缴费服务、充值服务等。以下简称服务端。

1.3 网络拓扑

前置机与统一支付平台以及各系统间交互的网络拓扑大致如下图所示：



2 步骤

2.1 配置商户（如已配置可忽略）

若商户未配置，请先联系各环境配置管理员进行商户配置，开发环境联系支付组成员，测试、预发环境联系测试负责人，生产环境联系各部门配管人员。

商户配置后得到商户唯一 partnerId，目前与医院 hosId 值相等。

2.2 配置 ISV（如已配置可忽略）

若 ISV 未配置，请先联系各环境配置管理员进行商户配置，开发环境联系支付组成员，

测试、预发环境联系测试负责人，生产环境联系各部门配管人员。

ISV 配置后得到唯一 isvId，若前置机工程仅被 HIS 调用，则配置 isvId 为固定值 20000 即可，当前置机工程服务被多个公司调用，则根据需要配置不同 isvId。

2.3 复制 demo 工程并新建项目

Git 地址：http://10.26.10.76/pay/access/upp-access-demo.git，代码下载分支为 master。

联系支付组负责人或者运维管理人员，在 http://10.26.10.76/pay/access 地址中创建项目，命名规则为：upp-access-${hosId}。

创建完成项目后，下载创建的项目，并新建分支，然后复制 upp-access-demo 工程代码，修改 pom 文件打包名称(请勿修改 pom 依赖)，最后进行开发，并将项目代码提交至此。

如：浙医二院需要上线当面付服务时，创建项目：upp-access-147，下载该项目，并创建分支 1.0.0，然后从 upp-access-demo 工程中复制代码至 upp-access-147，并修改 pom 打包名称，最后进行开发并提交代码。

2.4 修改 upp-client 工程依赖

通过修改 pom 文件中的 upp-client.jar 的版本选择对应版本导入

引入 pom 代码如下：

```
<!-- 【架构体系内的模块引用 start】 -->
<dependency>
    <groupId>com.yuntai.pay</groupId>
    <artifactId>upp-client</artifactId>
    <version>1.0.0-SNAPSHOT</version>
</dependency>
<!-- 【架构体系内的模块引用 end】 -->
```

maven 连接公司私服即可下载该 jar 包

（http://gitlab.yuntaibus.com:8081/repository/maven-public/）

当前版本为：1.0.0-SNAPSHOT，如有升级会通过群组邮件通知。

2.5 修改工程配置

前置机工程在部署是有部分配置需要按照特定要求进行配置，详细内容参见第三章节。

2.6 实现必要服务

前置机工程有部分服务需要对接人员开发实现，详细内容参见第四章节。

3 配置

3.1 配置结构

配置文件目录结构如下图：

```
-| resources                                  资源文件
    -| assembly                               打包配置文件夹
        -| assembly.xml                       maven打包配置文件
    -| deploy                                 工程配置文件夹
        -| dev.properties                     开发环境配置文件
        -| pet.properties                     压测环境配置文件
        -| prd.properties                     生产环境配置文件
        -| test.properties                    测试环境配置文件
        -| uat.properties                     预发布环境配置文件
    -| mapper                                 Mapper文件夹
        -| XXXXMapper.xml                     DAO映射文件
    -| application.yml                        spring-boot主配置文件
    -| cache.yml                              工程常量缓存配置文件
    -| log4j2.yml                             log4j2日志配置文件
    -| run.sh                                 Linux环境下项目启动脚本
```

配置文件目录中，对接开发人员如无特殊要求，则只需关注 deploy 目录下各个环境的配置，以及 cache.yml 中的常量配置，其它配置文件无需关注和修改。

3.2 数据库配置(按实际场景所需配置)

需要数据库配置时（数据源支持 mysql&oracle【具体配置切换方式本文暂不做详细赘述】），按照 upp-client-demo 中 mapper 下的 DemoMapper.xml 为例子编写 DAO 映射。数据源配置位于/resource/deploy 文件夹下的 xxx.properties 配置文件内，具体配置内容如下图：

```
############################
#   settings for database   #
############################
# 前置机数据库是否启用(true: 启用;false: 禁用)
spring.datasource.druid.enabled=true
# 数据库地址
spring.datasource.druid.url=jdbc:mysql://112.124.33.7:3306/upp?zeroDateTimeBehavior=convertToNull&allowMultiQueries=true&characterEncoding=utf8&serverTimezone=Asia/Shanghai
# 数据库用户名
spring.datasource.druid.username=med
# 数据库密码
spring.datasource.druid.password=medicalcore01
```

无需数据库配置时，删除 mapper 文件夹(含)下所有文件，并将 xxx.properties 配置文件内数据源配置成部分如下图所示：

```
############################
#   settings for database   #
############################
# 前置机数据库是否启用(true: 启用;false: 禁用)
spring.datasource.druid.enabled=false
# 数据库地址
spring.datasource.druid.url=
# 数据库用户名
spring.datasource.druid.username=
# 数据库密码
spring.datasource.druid.password=
```

3.3 HDP 配置

配置位于/resource/deploy 文件夹下的 xxx.properties 配置文件内，配置示意如下图：

```
############################
#    settings for hdp      #
############################
# hdp ip
hdp.ip=120.55.66.16
# hdp port
hdp.port=8088
# hdp token
hdp.token=c87956357c1505443ec093f2bd5580e6
# hdp resource-id
hdp.resouce-id=999
```

3.4 日志配置

配置位于/resource/deploy 文件夹下的 xxx.properties 配置文件内，配置示意如下图：

```
############################
#    settings for log4j2   #
############################
# 日志路径
log.path=/data/ftp/log/upp
# 日志输入等级(文件)
log.level.root=debug
# 日志输入等级(控制台)
log.level.console=debug
```

3.5 商户常量配置

配置位于/resource 文件夹下的 cache.yml 配置文件内，统一支付平台的前置机服务支持多家医院、多个 ISV 共同调用的场景。如有需要，则配置多个 ISV ID 及 partner ID 即可。

partnerId：即为商户 ID，在公司内部通常即为医院的 hosId；

isvId：即服务商 ID，即前置机服务的调用公司标识，20000 表示 HIS，如果前置机需要被 HIS 访问，则必须配置 20000。

配置示意如下图：

```
cache:
  partner:
    - partner-id: 999          # 商户ID，在公司内部通常即为医院的hosId
      isv:
        - isv-id: 20000        # 服务商ID，即前置机服务的调用公司标识，20000 表示 HIS，如果前置机需要被 HIS 访问，则必须配置 20000
#         - isv-id: 20001      #（若一家商户被多个ISV使用，则在此追加ISV ID配置，如上示例中 999 需要被 20000 和 20001 共同使用）
#     - partner-id: 147        #（若前置机服务需要面向多家商户，则在此追加 partnerId 配置，如此配置表示服务需要面向 999 和 147 共同使用）
#       isv:
#         - isv-id: 20000
```

4 服务实现

4.1 代码结构

前置机当面付服务的代码结构如下图所示：

```
-| com.yuntai.upp.access                          工程基础包目录
    -| demo
        -| BillDataMock.java                      对账账单下载模拟HIs返回数据
        -| PresentDataMock.java                   实时订单查询模拟HIs返回数据
    -| helper
        -| BillHelper.java                        实现具体对账账单下载功能，为上层调用方提供返回数据
        -| PaymentCallbackHelper.java             实现支付结果通知功能，当面付支付通知给HIS支付结果
        -| PresentHelper.java                     实现具体实时订单查询功能，为上层调用方提供返回数据
        -| RefundCallbackHelper.java              实现退款结果通知功能，当面付退款通知给HIS退款结果
    -| mapper
        -| DemoMapper.java                        DAO映射
    -| service
        -| DemoService.java                       功能操作接口
        -| impl
            -| DemoServiceImpl.java 功能操作接口实现
    -| ProviderBoot.java                          spring boot 启动入口
```

在对接接入医院时，有几个服务接口必须要求实现：

1、 对账账单下载

2、 实时订单查询

3、 支付结果通知

4、 退款结果通知

具体实现方式以及则根据各医院场景独立实现即可。接口的介绍以及实现中一些特殊要求参见下文具体描述。

需要额外注意：前置机在实现服务接口时和上层调用者以及下游的 HIS 数据提供者交互时，需要保留完整的日志，明确打印上层出参入参，明确打印下游 HIS 提供的数据值。

4.2 接口-对账账单下载

必须重写 BillsHelper 类(禁止修改 bean 名、类名)，仅允许重写 data 方法。

该接口用于对账单下载，进行系统三方对账或实时对账。

注意：

1、 在用该接口获取账单数据时，任何一条数据在获取失败时，则整个账单不可用，返回上层数据响应属性的 result 需置为 false，且 data 置为空，否则会造成对账产生不正常的结果，进而导致多退款等问题；

2、 接口正常响应数据，必须全部为 HIS 支付成功或者退款成功数据。

```java
/** @description 账单下载-真实实现(禁止修改 bean 名) ...*/
@Component("billsService")
public class BillsHelper extends AbstractBills {

    /** @description 账单数据转换 ...*/
    @Override
    public Outcome<List<BillsVo>> data(@NonNull BillsDto dto) {
        /*...*/
        List<BillsVo> list = new ArrayList<>( initialCapacity: 2048);
        try {
            list.addAll(BillsMock.mock());
        } catch (Exception exception) {
            return Outcome.fail(exception.getMessage());
        }
        return Outcome.success(list);
    }
}
```

以下是接口入参及出参说明：

com.yuntai.upp.client.fresh.model.dto.bill.BillDto				
入参	必填	数据类型	参数含义	附加备注
partnerId	是	Long	商户标识	
startTime	是	LocalTimeDat	导出账单数据的起始时间	格式为yyyy-MM-dd HH:mm:ss
endTime	是	LocalTimeDat	导出账单数据的终止时间	格式为yyyy-MM-dd HH:mm:ss
公共出参				
result	是	Boolean	响应结果	true:响应成功 false:响应失败
msg	否	String	错误提示	
data	否	List	响应数据	List<BillVo>
com.yuntai.upp.client.fresh.model.vo.bill.BillVo				
出参	必填	数据类型	参数含义	附加备注
partnerId	是	Long	商户标识	
billsDate	是	LocalDate	对账日期	yyyy-MM-dd
tradeType	是	String	交易类型	枚举类：com.yuntai.upp.sdk.enums.TradeType
tradeFee	是	BigDecimal(6	交易金额	
tradeTime	是	LocalDateTim	交易时间	yyyy-MM-dd HH:mm:ss
outTradeNo	是	String	芸泰流水	
inTradeNo	是	String	渠道流水	
tradeNo	是	String	HIS 流水	
bizType	是	String	业务类型	枚举类：com.yuntai.upp.sdk.enums.BizType
districtId	否	String	院区标识	
sourceType	否	String	来源	枚举类：com.yuntai.upp.sdk.enums.SourceType
payType	否	String	类型	枚举类：com.yuntai.upp.sdk.enums.CheckBillsPayType
statType	否	String	对账类型	枚举类：com.yuntai.upp.sdk.enums.ChannelType
channelType	否	String	支付渠道	枚举类：com.yuntai.upp.sdk.enums.CheckBillsStatType
empno	否	String	收款员工号	
deviceNo	否	String	收款设备编号	

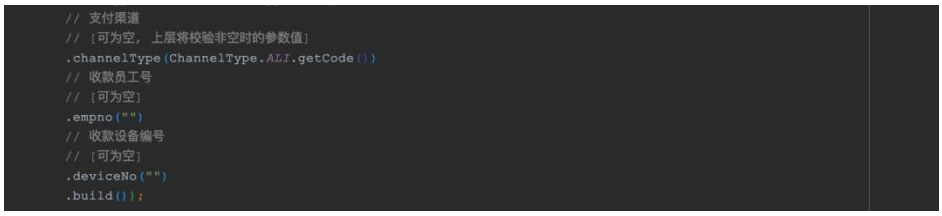
上层将会对整个响应进行第一层解析，判断 result 是否为 true。

若 result 为 false，则直接提示异常信息 msg，否则进行解析 data 中包包含对数据

data 数据将进行基本空校验（上述图中必填项）及枚举有效性校验，若出现任一一项无法通过断言校验，则提示错误信息。

模拟代码示例：

```
/*
 * 以下各值请务必按规则|枚举进行赋值(上层有规则校验限制)
 */
vos.add(BillVo.builder()
        // 商户标识
        // 该值可直接通过 BillHelper 中的 data 方法中的形参 dto 中直接获取
        // partnerId = dto.getPartnerId()
        // [不可为空, 参数返回后必校验]
        .partnerId(999L)
        // 账单日期
        // 真实场景可通过起始时间进行转换
        // end_time - start_time <= 24H && start_time >= 00:00
        // [不可为空, 参数返回后必校验]
        .billsDate(LocalDate.now())
        // pay:支付 refundFee:退款
        // [不可为空, 参数返回后必校验]
        .tradeType(Math.random() > 0.5D ? TradeType.PAY.getCode() : TradeType.REFUND.getCode())
        // 注意!!! 金额数字没有负数,取绝对值
        // [不可为空, 参数返回后必校验]
        .tradeFee(new BigDecimal( val: Math.random() * 100).abs().setScale( newScale: 2, BigDecimal.ROUND_HALF_UP))
        // 交易时间
        // [不可为空, 参数返回后必校验]
        .tradeTime(LocalDateTime.now())
        // 芸泰流水
        // [不可为空, 参数返回后必校验]
        .outTradeNo(UUIDUtil.create())
        // 第三方流水
        // [不可为空, 参数返回后必校验]
        .inTradeNo(UUIDUtil.create())
        // HIS 流水
        // [不可为空, 参数返回后必校验]
        .tradeNo(UUIDUtil.create())
        // 业务类型 请参照 BizType 枚举类
        // [不可为空, 参数返回后必校验]
        .bizType(BizType.REGISTER.getCode())
        // 院区标识(存在这赋值,部分商户需要)
        // [可为空, 上层将校验非空时的参数值]
        .districtId(UUIDUtil.create())
        // 来源 outpatient:门诊 inpatient:住院
        // [可为空, 上层将校验非空时的参数值]
        .sourceType(SourceType.INPATIENT.getCode())
        // 类型 self:自费 medical:医保
        // [可为空, 上层将校验非空时的参数值]
        .payType(CheckBillsPayType.SELF.getCode())
        // 对账类型 改值 @服务端对账开发人员,按照服务端配置赋值
        // [可为空, 上层将校验非空时的参数值]
        .statType(CheckBillsStatType.ONE.getCode())
        // 支付渠道
        // [可为空, 上层将校验非空时的参数值]
        .channelType(ChannelType.ALI.getCode())
```



```
        // 支付渠道
        // [可为空, 上层将校验非空时的参数值]
        .channelType(ChannelType.ALI.getCode())
        // 收款员工号
        // [可为空]
        .empno("")
        // 收款设备编号
        // [可为空]
        .deviceNo("")
        .build());
```

4.3 接口-实时订单查询

必须重写 PresentHelper 类(禁止修改 bean 名、类名)，仅允许重写 data 方法。

该接口用于查询 HIS 订单，进行当日订单的系统自动退款和单边账查询。

注意：

1、 用该接口在获取数据发生任何异常时，则获取的数据结果不可用，返回上层数据响应属性的 result 需置为 false，否则会造成多退款问题， 即**仅在与 HIS 接口发生交互失败的场景下才能返回 Outcome.fail(exception.getMessage()),其他场景均需返回 Outcome.success(list)【若 HIS 不存在该笔订单，则 list 为[]，禁止返回 null】**；

2、 接口正常响应数据，必须全部为 HIS 支付成功或者退款成功数据；

3、 在部分医院如果无法提供根据流水查询订单的功能，则可以根据入参订单创建时间查询前后三天的账单数据，然后再根据流水获取订单信息。

```java
/** @description 实时交易-真实实现 ...*/
@Component("presentService")
public class PresentHelper extends AbstractPresent {

    /** @description 交易数据转换 ...*/
    @Override
    public Outcome<List<PresentVo>> data(@NonNull PresentDto dto) {
        /*...*/
        List<PresentVo> list = new ArrayList<>( initialCapacity: 8);
        try {
            list.addAll(PresentMock.mock());
        } catch (Exception exception) {
            return Outcome.fail(exception.getMessage());
        }
        return Outcome.success(list);
    }
}
```

以下是接口入参及出参说明：

com.yuntai.upp.client.fresh.model.dto.present.PresentDto				
入参	必填	数据类型	参数含义	附加备注
partnerId	是	Long	商户标识	
outPaymentNo	是	String	芸泰支付流水	芸泰云端服务针对某笔订单的唯一支付流水标识
inPaymentNo	否	String	渠道支付流水	支付宝、微信等对某笔订单的唯一支付流水标识
payemntNo	是	String	商户（HIS）支付流水	HIS 某笔订单的唯一支付流水标识
gmtCreate	是	String	订单创建时间	统一支付平台订单创建时间
公共出参				
result	是	Boolean	响应结果	true:响应成功 false:响应失败
msg	否	String	错误提示	
data	否	List	响应数据	List<PresentVo>
com.yuntai.upp.client.fresh.model.vo.present.PresentVo				
出参	必填	数据类型	参数含义	附加备注
partnerId	是	Long	商户标识	
outTradeNo	是	String	芸泰流水	正交易则返回芸泰正交易流水，负交易则返回芸泰负交易流水
inTradeNo	是	String	渠道流水	正交易则返回渠道正交易流水，负交易则返回渠道负交易流水
tradeNo	是	String	商户（HIS）流水	正交易则返回HIS正交易流水，负交易则返回HIS负交易流水
tradeFee	是	BigDecimal(6,2	交易金额	保留小数点后2位，单位元（RMB）
tradeTime	是	LocalDateTime	交易时间	格式：yyyy-MM-dd HH:mm:ss
tradeType	是	String	交易类型	枚举类：com.yuntai.upp.sdk.enums.TradeType

上层将将会对整个响应进行第一层解析，判断 result 是否为 true。

若 result 为 false，则直接提示异常信息 msg，否则进行解析 data 中包包含对数据

data 数据将进行基本空校验（上述图中必填项）及枚举有效性校验，若出现任一一项无法通过断言校验，则提示错误信息。

模拟代码示例：

```java
public static List<PresentVo> mock() {
    List<PresentVo> vos = new ArrayList<>();

    /* 模拟数据,仅供无 HIS 接口时,工程测试使用 - start */
    vos.add(PresentVo.builder()
            // 芸泰流水
            // [不可为空, 参数返回后必校验]
            .outTradeNo(UUIDUtil.create())
            // 第三方流水
            // [不可为空, 参数返回后必校验]
            .inTradeNo(UUIDUtil.create())
            // HIS 流水
            // [不可为空, 参数返回后必校验]
            .tradeNo(UUIDUtil.create())
            // 交易金额(BigDecimal 类型, 保留 2 位小数)
            // [不可为空, 参数返回后必校验]
            .tradeFee((new BigDecimal(Math.random()).setScale( newScale: 2, RoundingMode.HALF_UP)))
            // 交易时间(LocalDateTime 类型)
            // [不可为空, 参数返回后必校验]
            .tradeTime(LocalDateTime.now())
            // 交易类型(TradeType 枚举类内 payment/refund)
            // [不可为空, 参数返回后必校验]
            .tradeType(TradeType.PAY.getCode())
            .build());
    /* 模拟数据,仅供无 HIS 接口时,工程测试使用 - end */
    return vos;
}
```

4.4 接口-支付结果通知

必须重写 PaymentCallbackHelper 类(禁止修改 bean 名、类名),仅允许重写 data 方法。

该接口用于进行当面付订单的支付结果通知，当支付成功时一定会有该通知触发。

注意：

1、 该通知为 HIS 查询支付结果的补偿机制，不建议 HIS 实现接收该通知，建议 HIS 以查询为主；

2、 如 HIS 不实现该功能时，对接实现直接返回 true 即可。

```java
/** @description 交易通知-真实实现(禁止修改 bean 名) ...*/
@Component("paymentCallbackService")
public class PaymentCallbackHelper extends AbstractPaymentCallback {

    /** @description 交易数据转换 ...*/
    @Override
    protected Outcome<PaymentCallbackVo> data(@NonNull PaymentCallbackDto dto) {
        try {...} catch (Exception exception) {
            return Outcome.fail(exception.getMessage());
        }
    }
}
```

以下是接口入参及出参说明：

com.yuntai.upp.client.fresh.model.dto.paymentcallback.PaymentCallbackDto				
入参	必填	数据类型	参数含义	附加备注
partnerId	是	Long	商户标识	
bizType	是	String	业务类型	
bizId	是	String	业务唯一标识	
channelType	是	String	支付渠道	
channelProduct	是	String	渠道产品	
tradeFee	是	BigDecimal	实付订单金额	
tradeStatus	是	String	交易状态	
outPaymentNo	是	String	芸泰支付流水	芸泰云端服务针对某笔订单的唯一支付流水标识
inPaymentNo	是	String	渠道支付流水	支付宝、微信等对某笔订单的唯一支付流水标识
payemntNo	是	String	商户（HIS）支付流水	HIS 某笔订单的唯一支付流水标识
paymentTime	是	LocalDateTime	支付时间	格式为yyyy-MM-dd HH:mm:ss
bizData	否	String	业务数据	
公共出参				
result	是	Boolean	响应结果	true:响应成功 false:响应失败
msg	否	String	错误提示	
data	否	List	响应数据	List<PaymentCallbackVo>
com.yuntai.upp.client.fresh.model.vo.paymentcallback.PaymentCallbackVo				
出参	必填	数据类型	参数含义	附加备注
result	是	boolean	接收通知处理响应标识	true:响应成功 false:响应失败

上层将将会对整个响应进行第一层解析，判断 result 是否为 true。

若 result 为 false，则直接提示异常信息 msg，否则进行解析 data 中包包含对数据

data 数据将进行基本空校验（上述图中必填项）及枚举有效性校验，若出现任一一项无法通过断言校验，则提示错误信息。

模拟代码示例：

```
/**
 * @description: 交易通知数据模拟
 * @className: CallbackMock
 * @package: com.yuntai.upp.access.demo
 * @author: jinren@hsyuntai.com
 * @date: 2019-09-06 10:17
 * @copyright: 版权归 HSYUNTAI 所有
 */
public class PaymentCallbackMock {

    /**
     * @description 数据模拟(真实场景禁止使用,仅供开发使用)
     * @return boolean
     * @author jinren@hsyuntai.com
     * @date 2019-09-06 10:19
     */
    public static boolean mock() {
        return Math.random() > 0.0D;
    }
}
```

4.5 接口-退款结果通知

必须重写 RefundCallbackHelper 类(禁止修改 bean 名、类名)，仅允许重写 data 方法。

该接口用于查询 HIS 订单，进行当日订单的系统自动退款和单边账查询。

注意：

1、 该通知为 HIS 查询支付结果的补偿机制，不建议 HIS 实现接收该通知，建议 HIS 以查询为主；

2、 如 HIS 不实现该功能时，对接实现直接返回 true 即可。

```java
/** @description 交易通知-真实实现(禁止修改 bean 名) ...*/
@Component("refundCallbackService")
public class RefundCallbackHelper extends AbstractRefundCallback {

    /** @description 交易数据转换 ...*/
    @Override
    protected Outcome<RefundCallbackVo> data(@NonNull RefundCallbackDto dto) {
        try {...} catch (Exception exception) {
            return Outcome.fail(exception.getMessage());
        }
    }
}
```

以下是接口入参及出参说明：

com.yuntai.upp.client.fresh.model.dto.paymentcallback.PaymentCallbackDto				
入参	必填	数据类型	参数含义	附加备注
partnerId	是	Long	商户标识	
bizType	是	String	业务类型	
bizId	是	String	业务唯一标识	
payemntNo	是	String	商户（HIS）支付流水	HIS 某笔订单的唯一支付流水标识
channelType	是	String	支付渠道	
channelProduct	是	String	渠道产品	
requestNo	是	String	退款请求号	
refundFee	是	BigDecimal	退款金额	
tradeStatus	是	String	交易状态	
outRefundNo	是	String	芸泰支付流水	芸泰云端服务针对某笔订单的唯一支付流水标识
refundNo	是	String	退款对账流水标识	
refundTime	是	LocalDateTime	退款时间	格式为yyyy-MM-dd HH:mm:ss
bizData	否	String	业务数据	
公共出参				
result	是	Boolean	响应结果	true:响应成功 false:响应失败
msg	否	String	错误提示	
data	否	List	响应数据	List<RefundCallbackVo>
com.yuntai.upp.client.fresh.model.vo.refundcallback.RefundCallbackVo				
出参	必填	数据类型	参数含义	附加备注
result	是	boolean	接收通知处理响应标识	true:响应成功 false:响应失败

若 result 为 false，则直接提示异常信息 msg，否则进行解析 data 中包包含对数据

data 数据将进行基本空校验（上述图中必填项）及枚举有效性校验，若出现任一一项无法通过断言校验，则提示错误信息。

模拟代码示例：

```java
/**
 * @description: 交易通知数据模拟
 * @className: CallbackMock
 * @package: com.yuntai.upp.access.demo
 * @author: jinren@hsyuntai.com
 * @date: 2019-09-06 10:17
 * @copyright: 版权归 HSYUNTAI 所有
 */
public class RefundCallbackMock {

    /**
     * @description 数据模拟(真实场景禁止使用,仅供开发使用)
     * @return boolean
     * @author jinren@hsyuntai.com
     * @date 2019-09-06 10:19
     */
    public static boolean mock() {
        return Math.random() > 0.0D;
    }
}
```

5 枚举

工程中所有枚举类禁止在工程类重复定义(upp-client 已提供枚举类供开发人员使用)

5.1 TradeType

枚举值	枚举含义
pay	支付
refund	退款

5.2 BizType

枚举值	枚举含义
register	预约挂号
goods	商品支付
self_pay	自助缴费
recharge	卡充值
ol_diagnosis	在线诊疗
his_drug	医院购药
ol_drug	在线购药
consult	在线咨询
withdraw	提现
third_app	第三方应用
health_reg	体检预约
revisit	复诊
referral	转诊订单
consult_diag	会诊订单
park_pay	停车缴费
health_exams	体检套餐(袋鼠)-开放平台

pharmacist	药剂配送
sport_pres	运动处方-开放平台
charge_course	收费课程
medical_record_copy	病历复印
paid_course	付费课程
paid_qa	付费问答
paid_ask	问诊
agg_scan_pay	聚合支付-业务扫码
card_book_fee	卡本费
biz_scan	业务扫码
trade_transfer	交易转账
scan_pay	扫码支付
barcode_pay	条码支付

5.3 SourceType

枚举值	枚举含义
outpatient	门诊
inpatient	住院

5.4 CheckBillsPayType

枚举值	枚举含义
self	自费
medical	医保

5.5 ChannelType

枚举值	枚举含义
ali	自费
ten	微信
uni	银联
zero	零元
ins	医保
bia	扁鹊

5.6 CheckBillsStatType

枚举值	枚举含义
one	统计分类 1
two	统计分类 2
three	统计分类 3
four	统计分类 4

该枚举具体含义请与@对账开发人员沟通自定义

6 常见问题 Q&A

Q：HIS 接入后，提示签名错误类提示信息？

A：请按照一下相关步骤进行排查操作

1、请判断是前置机提示的签名错误类提示信息还是 HIS 接收参数提示验签错误类提示信息，若为前置机提示的签名错误类提示信息，请按照 2.1 进行操作；若为 HIS 接收参数提示的时提示的验签错误类提示信息，请按 3.1 进行操作

2.1、将日志等级切换至 DEBUG（默认为 INFO），让 HIS 提供源串及盐值

2.2、在前置机日志中搜寻对应验签日志

如下格式：

相同 traceId 表示为同一请求类发生的操作（后续中将有相关描述）

2.3、对比两个源串及盐值即可，若验证相同，这查看 HDP 发送报文及 HDP 接收报文，判断是否为 HDP 响应时提示的验签失败，则联系 jinren@hsyuntai.com，排查解决

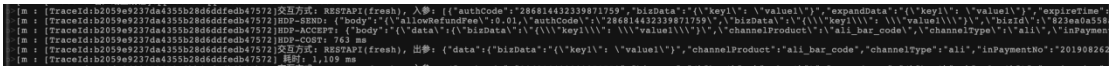
3.1、将日志登记切换至 DEBUG（默认为 INFO），找出接口出参加密基础信息（日志中打印的源串及盐值）

3.2、对比源串及盐值，让 HIS 进行响应的修改（前置机源串及盐值均为最终正确值）

Q：前置机日志打印表述内容？

A：INFO 等级：

如下图：



INFO 级别下打印的为

1、接口入参

2、接口出参

3、HDP 发送参数

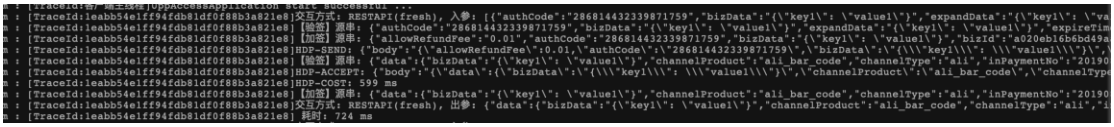
4、HDP 接收参数

5、HDP 耗时

6、接口耗时

DEBUG 等级：

如下图：



DEBUF 等级下打印的为

1、接口入参

2、接口验签基础信息

3、HDP 发送参数加密基础信息

4、HDP 发送参数

5、HDP 接收参数

6、HDP 接口参数验签基础信息

7、HDP 耗时

8、接口出参加密基础信息

9、接口出参

10、接口耗时

Q：下载 upp-access-demo 后无法正常启动？（提示 @log.path@？？？【占位符类错误】）

A：请重新编译工程【出现该问题，请优先排查 maven 依赖的所有 jar 是否正常引入】，问

题原因为工程未正确将对应占位符对应的属性替换成配置文件内配置的属性（或在 pom 文件中加入以下代码，进行 mvn install），若/target/classes/log4j2.yml 中的中的占位符正常被替换，则正常

```
<plugin>
   <artifactId>maven-resources-plugin</artifactId>
   <configuration>
      <encoding>utf-8</encoding>
      <useDefaultDelimiters>true</useDefaultDelimiters>
   </configuration>
</plugin>
```

Q：HIS 接入后，提示[字段 xxx 非法|错误，.....]

A：请按照提示信息检查 HIS 接口入参