OMS 基础收单系统环境搭建说明

一.系统构成

1.收单系统构成

搭建收单系统需使用的收单系统程序如下：

omscli 工具程序:https://gitlab.100bm.cn/micro-plat/oms/omscli

apiserver oms接口服务程序:https://gitlab.100bm.cn/micro-plat/oms/apiserver

flowserver oms流程服务程序:https://gitlab.100bm.cn/micro-plat/oms/flowserver

oms sdk服务程序:https://gitlab.100bm.cn/micro-plat/oms/oms

获取上述程序存放在本地:$GOPATH/gitlab.100bm.cn/micro-plat/oms/

2.集成使用的基础系统

收单系统的程序中集成了许多外部基础系统，需要将这些基础系统按各自操作文档获取代码到本地，并按文档将对应数据库表生成到收单系统的数据库中。这些基础系统包括：

qtask 任务处理系统:https://github.com/micro-plat/qtask

命令 git clone -b dev https://github.com/micro-plat/qtask.git

beanpay 支付系统:https://github.com/micro-plat/beanpay

命令 git clone -b dev https://github.com/micro-plat/beanpay.git

vds 发货系统:https://gitlab.100bm.cn/micro-plat/vds

命令 git clone https://gitlab.100bm.cn/micro-plat/vds/apiserver.git git clone https://gitlab.100bm.cn/micro-plat/vds/flowserver.git git clone https://gitlab.100bm.cn/micro-plat/vds/vds.git git clone https://gitlab.100bm.cn/micro-plat/vds/vdscli.git

lcs 生命周期系统:https://gitlab.100bm.cn/micro-plat/lcs

命令 git clone https://gitlab.100bm.cn/micro-plat/lcs/lcs.git

dds 数据字典系统:https://gitlab.100bm.cn/micro-plat/dds

命令 git clone https://gitlab.100bm.cn/micro-plat/dds/ddscli.git

获取vds,lcs,dds程序到本地:$GOPATH/gitlab.100bm.cn/micro-plat/

二.收单系统数据表创建

1. 数据库表创建

根据数据库类型进行创建,生成表之前必须先建好数据库. 进入创建程序目录:$GOPATH/gitlab.100bm.cn/micro-plat/oms/omscli

运行以下命令,生成程序:

```
    //mysql数据表生成命令
    go build -o ./out/createdb/bin/createdbserver

    //oracle数据表生成命令
    go build -tags oracle -o ./out/createdb/bin/createdbserver
```

1.1 创建mysql数据表

进入程序目录:./out/createdb/bin

运行以下命令进行数据表创建:

```
    ./createdbserver mysql#[maxOpen]#[maxIdle]#[数据库帐号]:[数据库密码]@tcp\
([数据库地址]\)/[数据库名]?charset=utf8
    实例:
    ./createdbserver mysql#20#10#oms_t:123456@tcp\(192.168.0.36\)/oms_t?
charset=utf8
```

1.2 创建oracle数据表

进入程序目录:$GOPATH/gitlab.100bm.cn/micro-plat/oms/omscli/out/createdb/bin

运行以下命令进行数据表创建:

```
    ./createdbserver oracle#[maxOpen]#[maxIdle]#[数据库名]/123456@[服务器链接
名]
    实例:
    ./createdbserver oracle#200#100#oms_t/123456@orcl136
```

2. 初始化数据添加

2.1 初始化oms数据

（1）产品线[oms_product_line]

字段名	值	说明
line_name	xxx产品	产品线名称
can_package_delivery	1	支持打包发货(0.支持，1.不支持)

字段名	值	说明
payment_queue	'oms:order:pay'	支付队列
bind_queue	'oms:order:bind'	绑定队列
delivery_queue	'oms:order:delivery'	发货队列
up_payment_queue	'oms:order:up_pay'	上游支付队列
notify_queue	'oms:order:notify'	通知队列
return_queue	'oms:refund:return'	退货队列
refund_queue	'oms:refund:pay'	退款队列
up_refund_queue	'oms:refund:up_pay'	上游退款队列
refund_notify_queue	'oms:refund:notify'	退款通知队列
order_refund_queue	'oms:overtime:refund'	订单失败退款队列
order_overtime_queue	'oms:overtime:order_deal'	订单超时处理队列
delivery_unknown_queue	'oms:overtime:delivery_unknown'	发货未知处理队列
refund_overtime_queue	'oms:overtime:refund_deal'	退款超时处理队列
return_unknown_queue	'oms:overtime:return_unknown'	退货未知处理队列
return_finish_queue	'oms:refund:return_complete'	退货结束队列

（2）下游渠道[oms_down_channel]

字段名	值	说明
channel_no	ycdown	渠道编号
channel_name	xxx渠道	渠道名称
status	0	状态

（3）下游货架[oms_down_shelf]

字段名	值	说明
shelf_name	xxx下游货架	货架编号
channel_no	'需要对应的渠道编号,ycdown'	渠道编号
order_overtime	根据需求设置，例如：600	订单超时时长
refund_overtime	根据需求设置，例如：600	退款超时时长
status	0	状态

（4）下游商品[oms_down_product]

字段名	值	说明
shelf_id	'需要对应的货架编号'	货架编号
line_id	'需要对应的产品线编号'	产品线编号
carrier_no	'根据需求设置，例如：ZSH'	运营商
province_no	'根据需求设置，例如：CQ'	省份
city_no	'根据需求设置，例如：*'	城市
invoice_type	'根据需求设置'	开票方式（1.不开发票，0.不限制，2.需要发票）
can_refund	'需要对应的货架编号'	支持退款(0.支持，1.不支持)
face	'根据需求设置，例如:200'	面值
sell_discount	'根据需求设置，例如:0.9'	销售折扣（以面值算）
commission_discount	'根据需求设置，例如:0.2'	佣金折扣（以面值算）
service_discount	'根据需求设置，例如:0.1'	服务费折扣(公司收取，以面值算)
payment_fee_discount	'根据需求设置，例如:0.1'	手续费折扣（以销售金额算）
can_split_order	'根据需求设置，例如:0'	是否允许拆单（0.允许，1不允许）
split_order_face	'根据需求设置，例如:100'	拆单面值（默认全部是产品面值，电子券不允许修改）
limit_count	'根据需求设置，例如:10'	单次最大购买数量
status	0	状态

（5）上游渠道[oms_up_channel]

字段名	值	说明
channel_no	'根据需求设置，例如:ycup'	上游渠道编号
channel_name	'xxx上游渠道'	名称
status	0	状态

（6）上游货架[oms_up_shelf]

字段名	值	说明
shelf_name	'xxx上游货架'	货架名称
channel_no	'需要对应的渠道编号,ycup'	渠道编号
delivery_overtime	'根据需求设置，例如:300'	发货超时时间
return_overtime	'根据需求设置，例如:300'	退货超时时间

字段名	值	说明
status	'0'	货架状态

（7）上游商品[oms_up_product]

字段名	值	说明
shelf_id	'需要对应的上游货架编号'	货架编号
line_id	'产品线编号需要与下游产品线相同'	产品线
carrier_no	'根据需求设置，例如:ZSH'	运营商
province_no	'根据需求设置，例如:Q'	省份
city_no	'根据需求设置，例如:×'	城市
invoice_type	'根据需求设置，例如:1'	开票方式（1.不开发票，2.上游开发票）
can_refund	'根据需求设置，例如:1'	支持退货（0.支持，1不支持）
face	'根据需求设置，例如:100'	面值
cost_discount	'根据需求设置，例如:0.1'	成本折扣（以面值算）
commission_discount	'根据需求设置，例如:0.1'	佣金折扣（以面值算）
service_discount	'根据需求设置，例如:0.1'	服务费折扣(公司收取，以面值算)
limit_count	'根据需求设置，例如:1'	单次最大发货数量
status	'0'	状态

2、初始化vds数据

（1）渠道基本信息[vds_channel_info]

字段名	值	说明
channel_no	'渠道编号与oms上游渠道编号相同'	渠道编号
service_class	'根据需求设置，例如:10'	服务类型 10:加油卡充值 20:电子券发货 21:电子券退货 30:话费充值
request_url	'www.api.recharge.com'	上游请求地址(发货,充值等)
notify_url	'www.api.notify.com'	通知回调地址
request_replenish_time	'根据需求设置，例如:100'	发货后补间隔时间(秒)
status	'0'	状态(0:启用 1:禁用)
can_query	'根据需求设置，例如:0'	是否支持查询(0:是 1:否)
query_url	'www.api.query.com'	查询地址

字段名	值	说明
first_query_time	'根据需求设置，例如:600'	首次查询时间(秒)
query_replenish_time	'根据需求设置，例如:180'	查询后补间隔时间(秒)

（2）渠道错误码[vds_channel_error_code]

字段名	值	说明
channel_no	'渠道编号与oms上游渠道编号相同'	渠道编号
service_class	'根据需求设置，例如:10'	服务类型 10:加油卡充值 20:电子券发货 21:电子券退货 30:话费充值
deal_code	'根据需求设置，例如:0'	处理码(0:成功 10:普通失败 90:未知)
error_code	'根据需求设置，例如:910'	错误码(R_0000:成功,R_5124:普通失败,R_9999:未知)
error_code_desc	'根据需求设置，例如:xxx错误'	错误码描述

渠道错误码需要配置成功,失败,未知三种错误码

3、初始化beanpay数据

字段名	值	说明
account_name	"XXX账户"	帐户名称
ident	oms	系统标识符
groups	down_channel	外部账户类型(down_channel:下游渠道账户,down_commission:下游佣金账户,down_service下游服务费账户,up_channel:上游渠道账户,up_commission上游佣金账户,up_service:上游服务费账户)
eid	whj_down	外部用户账户编号,对应上游和下游渠道编号
balance	9999999	帐户余额，单位：分
credit	0	信用余额，单位：分
status	'0'	账户状态 0：正常 1:锁定

二.独立部署

独立部署和集成部署根据需要任选其一

1. oms程序打包

根据部署需要,生成程序安装包! 进入对应的程序目录:$GOPATH/gitlab.100bm.cn/micro-plat/oms/omscli

1.1 测试环境程序打包

运行如下命令,进行测试安装包生成:

```
./build.sh //支持mysql数据库
或
./build.sh oracle //支持oracle数据库
```

生成程序存放目录: $GOPATH/gitlab.100bm.cn/micro-plat/oms/omscli/out/apiserver/bin
$GOPATH/gitlab.100bm.cn/micro-plat/oms/omscli/out/flowserver/bin

1.2 生产环境程序打包

运行如下命令,进行生产环境安装包生成:

```
./build.sh prod//支持mysql数据库
或
./build.sh prod oracle //支持oracle数据库
```

生成程序存放目录: $GOPATH/gitlab.100bm.cn/micro-plat/oms/omscli/out/apiserver/bin
$GOPATH/gitlab.100bm.cn/micro-plat/oms/omscli/out/flowserver/bin

2. vds jcmockserver是上游发货打庄程序 打包

vds是通过集成的方式集成到oms中的，所以需要对vds进行集成打包 进入vds目录:$GOPATH/gitlab.100bm.cn/micro-plat/vds/vdscli

2.1 测试环境程序打包

运行如下命令,进行测试安装包生成:

```
./jcbuild.sh //支持mysql数据库
或
./jcbuild.sh oracle //支持oracle数据库
```

生成程序存放目录: $GOPATH/gitlab.100bm.cn/micro-plat/vds/vdscli/out/jcmockserver/bin

2.2 生产环境程序打包

运行如下命令,进行生产环境安装包生成:

```
./jcbuild.sh prod//支持mysql数据库
或
./jcbuild.sh prod oracle //支持oracle数据库
```

生成程序存放目录: $GOPATH/gitlab.100bm.cn/micro-plat/vds/vdscli/out/jcmockserver/bin

3. 运行程序

运行程序的[注册中心地址]和[启动节点名称]必须和安装时的保持一致.

3.1 运行oms系统apiserver

进入安装包存放目录:$GOPATH/gitlab.100bm.cn/micro-plat/oms/omscli/out/apiserver/bin 运行一下命令运行程序:

```
    ./apiserver-oms run -r [注册中心地址] -c [启动节点名称]

    //测试环境运行实例 如下
    ./apiserver-oms run -r zk://192.168.0.101 -c oms

    //生产环境运行实例 把注册中心地址改为生产环境集群  如下:
    ./apiserver-oms run -r zk://192.168.0.102,192.168.0.102,192.168.0.102 -
c oms
```

3.2 运行oms系统flowserver

进入安装包存放目录:$GOPATH/gitlab.100bm.cn/micro-plat/oms/omscli/out/flowserver/bin 运行一下命令运行程序:

```
    ./flowserver-oms run -r [注册中心地址] -c [启动节点名称]

    //测试环境运行实例 如下
    ./flowserver-oms run -r zk://192.168.0.101 -c oms

    //生产环境运行实例 把注册中心地址改为生产环境集群  如下:
    ./flowserver-oms run -r zk://192.168.0.102,192.168.0.102,192.168.0.102
-c oms
```

3.3 运行jcmockserver,jcmockserver是上游发货打庄程序

进入安装包存放目录:$GOPATH/gitlab.100bm.cn/micro-plat/vds/vdscli/out/jcmockserver/bin 运行一下命令运行程序:

```
    ./jcmockserver run -r [注册中心地址] -c [启动节点名称]

    //测试环境运行实例 如下
    ./jcmockserver run -r zk://192.168.0.101 -c oms

    //生产环境运行实例 把注册中心地址改为生产环境集群  如下:
    ./jcmockserver run -r zk://192.168.0.102,192.168.0.102,192.168.0.102 -c
oms
```

三.集成部署

独立部署和集成部署根据需要任选其一. 集成部署只提供测试环境部署文档.生产环境的部署根据业务方需要,参考测试部署进行操作.

1. 集成vds程序打包

根据部署需要,生成程序安装包! 进入打包工具程序目录:$GOPATH/gitlab.100bm.cn/micro-plat/vds/vdscli

1.1 测试环境程序打包

运行如下命令,进行测试安装包生成:

```
./jcbuild.sh //支持mysql数据库
或
./jcbuild.sh oracle //支持oracle数据库
```

生成程序存放目录: $GOPATH/gitlab.100bm.cn/micro-plat/vds/vdscli/out/jcmockserver/bin jcmockserver是上游发货打庄程序

2.1 安装集成部署程序jcmockserver

进入安装包存放目录:$GOPATH/gitlab.100bm.cn/micro-plat/vds/vdscli/out/jcmockserver/bin 运行一下命令进行安装:

```
./jcmockserver install -r [注册中心地址] -c [启动节点名称]

//测试环境安装实例 如下
./jcmockserver install -r zk://192.168.0.101 -c 5.107
```

3. 运行程序

3.1 运行集成部署程序jcmockserver

进入安装包存放目录:$GOPATH/gitlab.100bm.cn/micro-plat/vds/vdscli/out/jcmockserver/bin 运行一下命令进行安装:

```
./jcmockserver run -r [注册中心地址] -c [启动节点名称]

//测试环境安装实例 如下
./jcmockserver run -r zk://192.168.0.101 -c oms
```

四. 接口和参数说明

1.下单接口

1.1 接口地址

[运行机器ip]:8600/order/request 例:http://192.168.5.72:8600/order/request

1.2 请求参数

参数名	数据类型	可空	说明
channel_no	string	否	下游渠道编号
request_no	string	否	下游渠道订单编号
line_id	string	否	业务线编号
carrier_no	int	否	运营商
province_no	string	是	省份
city_no	int	是	城市
invoice_type	int	否	开票类型 1.不开发票，0.不限制，2.需要发票
num	string	否	下单数量
face	string	否	单个产品面值
amount	string	否	支付金额
notify_url	string	是	下单成功通知地址
rechage_account	string	是	充值账户编号

独立部署请求参数实例: http://192.168.5.72:8600/order/request?channel_no=test&request_no=test1200001&line_id=1&carrier_no=ZSH&invoice_type=1&num=1&face=200&amount=198&notify_url=http://xxx.com/recv&rechage_account=100011000000000

1.3 返回参数

参数名	数据类型	可空	说明
channel_no	string	否	渠道编号
request_no	string	否	渠道订单号
order_id	string	否	收单系统订单号
status	int	否	订单状态 0.发货成功，30.正在发货，90.发货失败，91.发货部分成功

1.4 调用结果说明

参数名	说明
channel_no	核对当前渠道编号是否与返回参数的渠道编号相同

参数名	说明
request_no	核对当前渠道订单号是否与返回参数的渠道订单号相同
order_id	核对当前订单号是否与返回参数的订单号相同
status	30表示正在进行，可能处于（发货、绑定、支付等状态）其他状态如status所描述相同，核对状态是否返回正确

1.5 错误类型

错误名	原因
下游渠道(xxx)不存在面值(xxx)的商品	下单的面值错误

2.订单查询接口

2.1 接口地址

[运行机器ip]:8600/order/query 例:http://192.168.5.72:8600/order/query

2.2 请求参数

参数名	数据类型	可空	说明
channel_no	string	否	渠道编号
request_no	string	否	渠道订单号

请求参数实例: http://192.168.5.72:8600/order/query?channel_no=test&request_no=test1200001

2.3 返回参数

参数名	数据类型	可空	说明
channel_no	string	否	渠道编号
request_no	string	否	渠道订单号
order_id	string	否	收单系统订单号
status	int	否	订单状态 0.发货成功，30.正在发货，90.发货失败，91.发货部分成功

2.4 调用结果说明

参数名	说明
channel_no	核对当前渠道编号是否与返回参数的渠道编号相同
request_no	核对当前渠道订单号是否与返回参数的渠道订单号相同
order_id	核对订单号是否与请求参数中的渠道编号和渠道订单号对应的收单系统订单号匹配

参数名	说明
status	30表示正在进行，可能处于（发货、绑定、支付等状态）其他状态如status所描述相同，核对状态是否返回正确

2.5 错误类型

错误	原因
订单不存在,channel_no:xxx,request_no:xxx	channel_no和request_no输入错误

3.普通退款接口

3.1 接口地址

[运行机器ip]:8600/refund/general 例:http://192.168.5.72:8600/refund/general

3.2 请求参数

参数名	数据类型	可空	说明
channel_no	string	否	渠道编号
request_no	string	否	渠道订单号
refund_no	string	否	退款编号
refund_num	string	否	退款数量
notify_url	string	否	退款通知地址

请求参数实例: http://192.168.5.72:8600/refund/general?channel_no=test&request_no=test1200001&refund_no=r1300000001&refund_num=1&notify_url=http://yyyy.com/refund

3.3 返回参数

参数名	数据类型	可空	说明
channel_no	string	否	渠道编号
request_no	string	否	渠道订单号
refund_no	string	否	渠道退款编号
refund_id	string	否	收单系统退款编号
refund_num	int	否	退款数量
status	int	否	退款状态 0.退款成功，30.正在退款，90.退款失败，91.退款部分成功

3.4 调用结果说明

参数名	说明
channel_no	核对当前渠道编号是否与返回参数的渠道编号相同
request_no	核对当前渠道订单号是否与返回参数的渠道订单号相同
refund_no	核对当前渠道退款编号是否与返回参数的渠道退款编号相同
refund_num	核对当前退款数量是否与返回参数的退款数量相同
status	30表示正在进行，可能处于（等待退货或退货中）其他状态如status所描述相同，核对状态是否返回正确

3.5 错误类型

错误	原因
订单不存在或配置错误	channel_no和request_no输入错误
订单未成功，暂时无法退款	该订单已经失败
当前订单不支持部分退款	该订单属于拆单订单，退款数量补等于了下单数量
不支持退款	该下游商品配置不支持退款

4.强制退款接口

4.1 接口地址

[运行机器ip]:8600/refund/mandatory 例:http://192.168.5.72:8600/refund/mandatory

4.2 请求参数

参数名	数据类型	可空	说明
channel_no	string	否	渠道编号
request_no	string	否	渠道订单号
refund_no	string	否	退款编号
refund_num	string	否	退款数量
notify_url	string	否	退款通知地址

请求参数实例: http://192.168.5.72:8600/refund/mandatory?channel_no=test&request_no=test1200001&refund_no=r1300000001&refund_num=1&notify_url=http://yyyy.com/refund

4.3 返回参数

参数名	数据类型	可空	说明
channel_no	string	否	渠道编号

参数名	数据类型	可空	说明
request_no	string	否	渠道订单号
refund_no	string	否	渠道退款编号
refund_id	string	否	收单系统退款编号
refund_num	int	否	退款数量
status	int	否	退款状态 0.退款成功，30.正在退款，90.退款失败，91.退款部分成功

4.4 调用结果说明

参数名	说明
channel_no	核对当前渠道编号是否与返回参数的渠道编号相同
request_no	核对当前渠道订单号是否与返回参数的渠道订单号相同
refund_no	核对当前渠道退款编号是否与返回参数的渠道退款编号相同
refund_num	核对当前退款数量是否与返回参数的退款数量相同
status	30表示正在进行，可能处于（等待退货或退货中）其他状态如status所描述相同，核对状态是否返回正确

4.5 错误类型

错误	原因
订单不存在	channel_no和request_no输入错误
订单未成功，暂时无法退款	该订单已经失败
当前订单不支持部分退款	该订单属于拆单订单，退款数量补等于了下单数量

5.退款查询接口

5.1 接口地址

[运行机器ip]:8600/refund/query 例:http://192.168.5.72:8600/refund/query

5.2 请求参数

参数名	数据类型	可空	说明
channel_no	string	否	渠道编号
request_no	string	否	渠道订单号
refund_no	string	否	退款编号

请求参数实例: http://192.168.5.72:8600/refund/query?channel_no=test&request_no=test1200001&refund_no=r1300000001

5.3 返回参数

参数名	数据类型	可空	说明
channel_no	string	否	渠道编号
request_no	string	否	渠道订单号
refund_no	string	否	渠道退款编号
refund_id	string	否	收单系统退款编号
refund_num	int	否	退款数量
status	int	否	退款状态 0.退款成功，30.正在退款，90.退款失败，91.退款部分成功

5.4 调用结果说明

参数名	说明
channel_no	核对当前渠道编号是否与返回参数的渠道编号相同
request_no	核对当前渠道订单号是否与返回参数的渠道订单号相同
refund_no	核对当前渠道退款编号是否与返回参数的渠道退款编号相同
refund_num	核对当前退款数量是否与返回参数的退款数量相同
status	30表示正在进行，可能处于（等待退货或退货中）其他状态如status所描述相同，核对状态是否返回正确

5.5 错误类型

错误	原因
查询的数据不存在	channel_no和request_no输入错误

6.退款通知

请求地址：根据实际业务确定，例如：http://192.168.106.174:8600/order/response

6.1 请求参数

参数名	数据类型	可空	说明
channel_no	string	否	渠道编号
request_no	string	否	渠道订单号
refund_no	string	否	退款编号

参数名	数据类型	可空	说明
refund_num	int	否	退款数量
status	int	否	退款状态值(0：成功，90：失败，30：进行中)

6.2 返回参数

参数名	数据类型	说明
notify_msg	string	下游返回的通知结果消息
status	int	下游返回的通知结果状态码