\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

中国移动统一支付平台

# 二期实时交易标准化培训文档

版本号：V 1.2.0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 引 言

本文档主要讲述的是统一支付二期实时交易的业务以及业务的相关流程，主要目的是帮助我们新同事可以快速的对项目代码和业务进行了解学习。该文档涉及的业务有：天猫充值、天猫查询、异步入库、签到签退、前线库导后线库、公参更新等。

## 目录

[二期实时交易标准化培训文档 - 1 -](#_Toc464583989)

[引 言 - 2 -](#_Toc464583990)

[目录 - 3 -](#_Toc464583991)

[1 概述 - 4 -](#_Toc464583992)

[2 JDK安装 - 4 -](#_Toc464583993)

[3 ACTIVEMQ下载和安装 - 8 -](#_Toc464583994)

[4 ACTIVEMQ介绍 - 11 -](#_Toc464583996)

[5 TOMCAT下载和安装 - 14 -](#_Toc464583997)

[6 本地项目环境搭建 - 16 -](#_Toc464583998)

[7 项目的部署 - 18 -](#_Toc464583999)

[8 仿真项目的部署 - 19 -](#_Toc464584000)

[9 充值流程 - 21 -](#_Toc464584001)

[10 查询流程 - 23 -](#_Toc464584002)

[11 异步入库的流程 - 25 -](#_Toc464584003)

[12 签到签退的流程 - 26 -](#_Toc464584004)

[13 前线库导后线库流程 - 27 -](#_Toc464584006)

[14 公参更新流程 - 30 -](#_Toc464584009)

### 概述

整个环境搭建工作需要在服务器上完成以下程序：

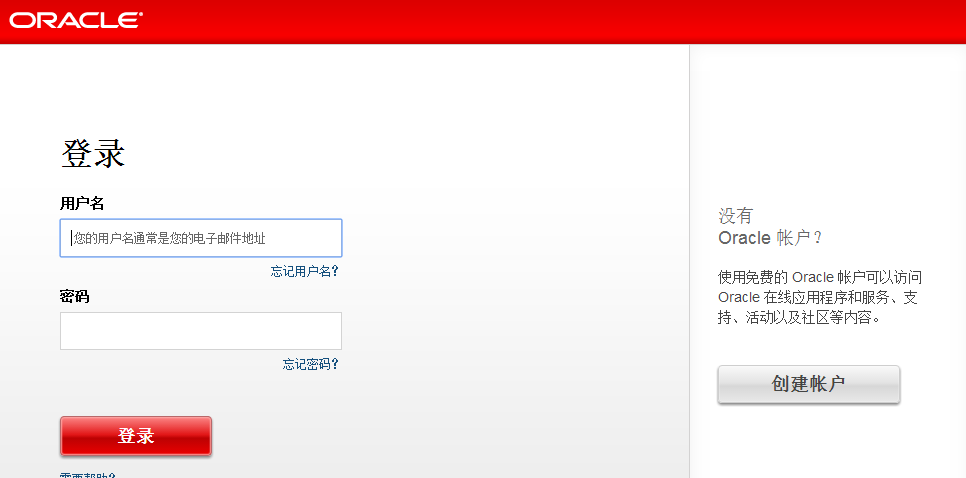
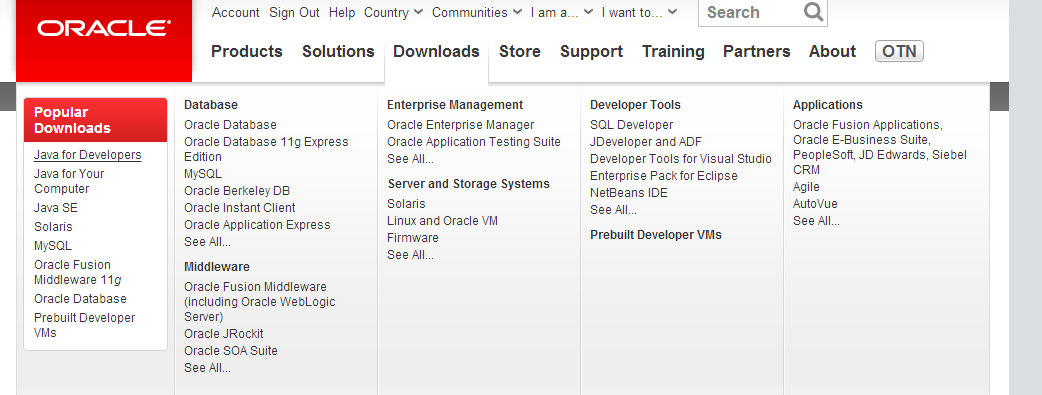
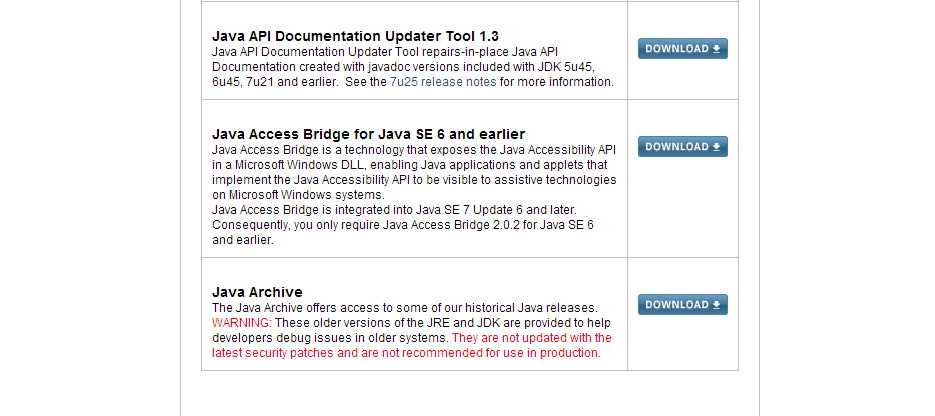
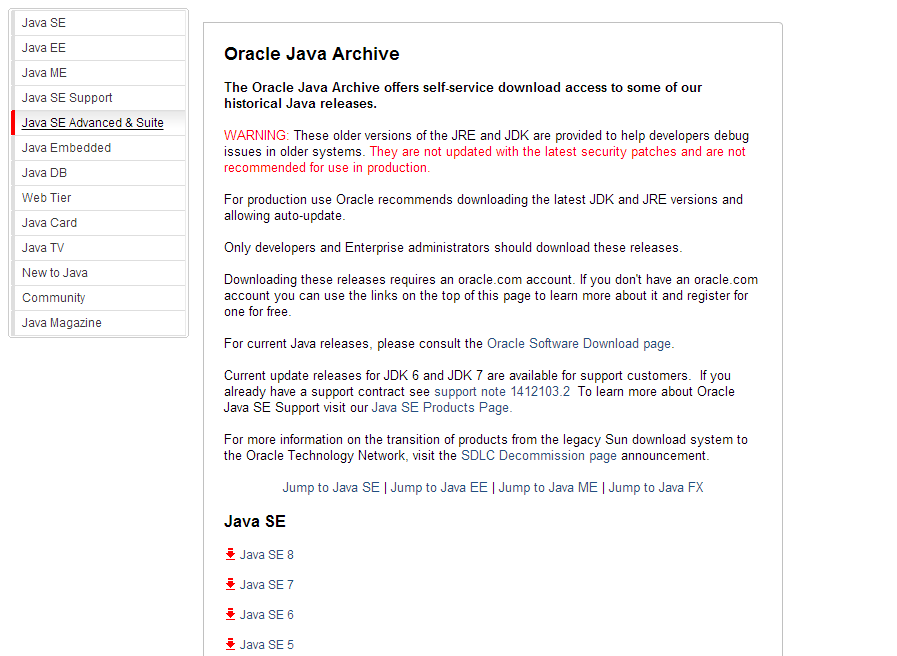
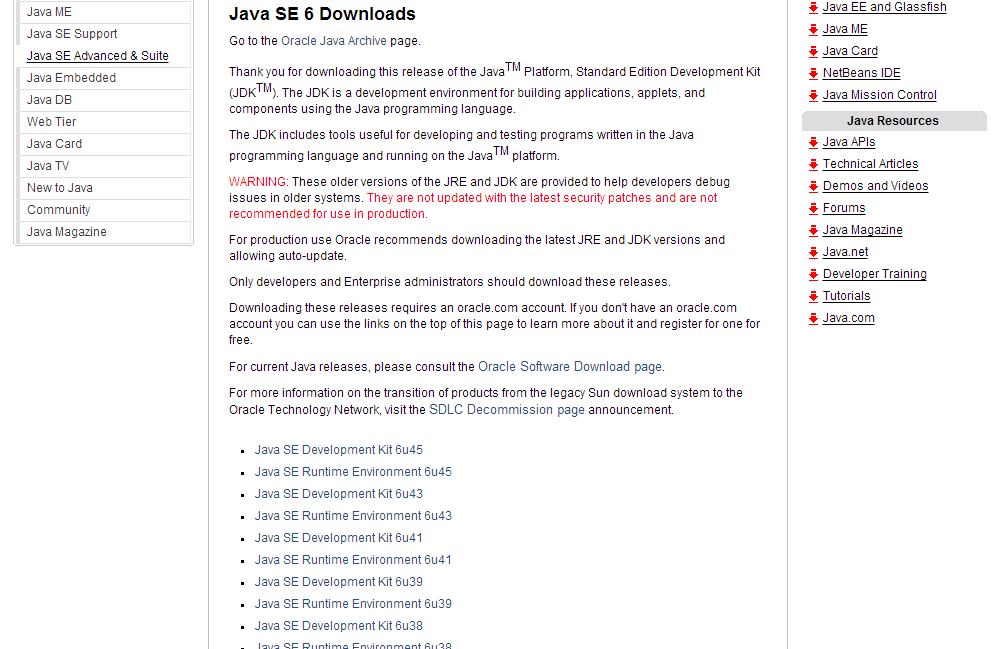
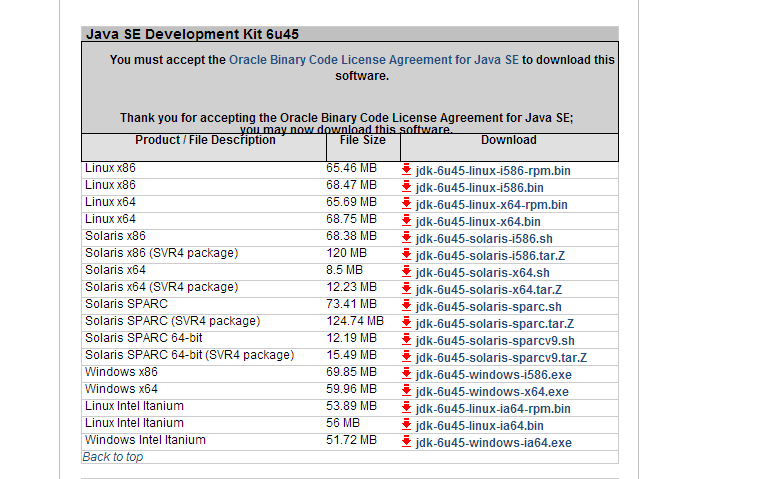
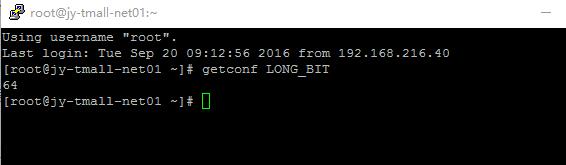
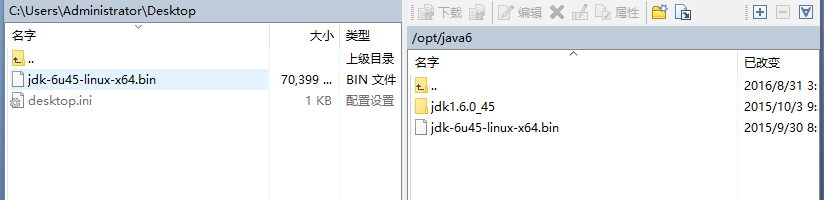
1、项目运行需要的工具：JDK、ACTIVEMQ、TOMCAT

2、天猫项目实时交易要部署的项目：

仿真、前置、核心、省前置、异步入库、签到签退、前线库导后线库

### JDK安装

要安装JDK，首先获取到安装文件。

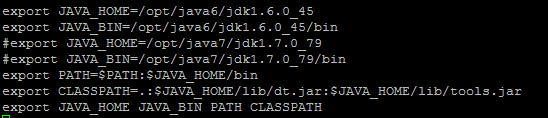
* 1. 下载JDK方式有很多种，推荐到Oracle官网下载，保证安全性和正规性。操作方式如下:
     1. 登陆oracle官网（<http://www.oracle.com>），输入用户名和密码登录，没有用户名的需要先注册一个。如果现在不登录，等到一会儿下载JDK的时候还是会要求你登陆，所以先登录再进行以后操作就省去了麻烦
     2. 登录后回到网站首页，回首页的最直接方式是点击左上角的网站logo。把鼠标悬浮到网站导航栏的Downloads选项，会出现很多的内容（如下图），找到最常下载(Popular Downloads)及下面的Java开发(Java for Developers)。
     3. 点击后在网页中间部分出现下图的JDK下载页面。
     4. 默认会出现最新版JDK下载选项，但是我们在项目中使用的是JDK6.所以在这个页面的最下面找到如下图所示的Java Archive点击后面的下载（DOWNLOAD）按钮，即跳转到新的页面
     5. 新页面包含了所有版本的JDK下载链接
     6. 点击上图的Java SE 6，来到下载页面在此页面可以看到JDK6的不同版本，选择最新更新版本Java SE Development Kit 6u45即可，点击它
     7. 点击后会跳转到页面下部的具体下载页面，如下图这里列出了不同操作系统下的JDK安装文件，我这里选择64位linux版本，下载第四个——jdk-6u45-linux-x64.bin；如果使用的32位的linux系统，需要下载第二个——jdk-6u45-linux-i586.bin。
     8. 查看linux系统版本可使用下图的命令：getconf LONG\_BIT
  2. 安装JDK
     1. 使用winscp把下载好的安装包——jdk-6u45-linux-x64.bin上传到服务器的指定目录中
     2. 使用如下命令解压软件
        1. 获取文件的操作权限：chmod u+x ./jdk-6u45-linux-x64.bin
        2. 解压文件：./jdk-6u45-linux-i586.bin获取jdk安装文件使用权限.png
        3. 上述命令执行后会出现提示，输入yes。然后jdk被解压到自动创建的./jdk1.6.0\_45目录下，使用命令pwd查看当前目录，下面配置环境变量的时候需要用到jdk安装目录.png
        4. 和在windows中一样，为了更方便使用JDK，需要配置环境变量。把上一步获得的JDK安装路径配置到环境变量里。Linux的环境变量在/etc/profile文件里，使用命令进行编辑：vi /etc/profile。打开文件后在末尾添加上如下配置：

export JAVA\_HOME=JDK安装路径

export JAVA\_BIN=JDK安装路径/bin

export PATH=$PATH:$JAVA\_HOME/bin

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

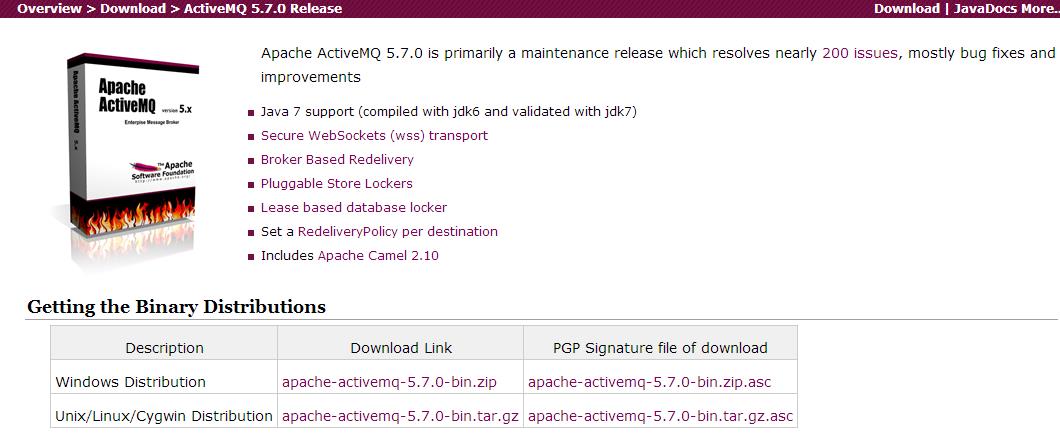
export JAVA\_HOME JAVA\_BIN PATH CLASSPATH例子如下图所示添加完成后，使用命令 :wq保存并退出编辑器。并使用source /etc/profile使环境变量的更改生效。

* + - 1. 测试安装是否成功

在任意目录下使用 java命令，如果正常输出java信息，即安装成功，如果是提示没有找到命令（command not found）等字样，则表示安装有问题。

### ACTIVEMQ下载和安装

* 1. ACTIVEMQ下载
     1. 打开官网<http://activemq.apache.org/>
     2. 选择右上角导航栏的Download，这里会默认推荐最新版本下载，可是最新版本会对JDK版本有要求，所以不能下载最新版本，下载和JDK6匹配的版本，一般是版本5.1-5.7都可以。这里下载5.7
     3. 点击[Download Archives](http://activemq.apache.org/download-archives.html) 字样的超链接，进入到以前版本的下载页面，如下图，找到5.7版本
     4. 点击5.7版本下载链接后就到达下载页面，如下图，选择linux版本——apache-activemq-5.7.0-bin.tar.gz



* 1. ACTIVEMQ安装配置
     1. 把下载到的安装文件使用FTP(winscp)软件上传到服务器指定目录
     2. 运行命令tar -xzvf /server/apache-activemq-5.11.1-bin.tar.gz解压文件
     3. 修改activemq运行文件的权限 cd /server/apache-activemq-5.11.1/bin chmod 755 activemq 运行./activemq
     4. 运行 三种运行方式：

(1)普通启动 ./activemq start

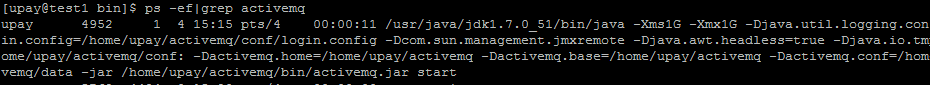
(2)启动并指定日志文件(mslog) ./activemq start >tmp/smlog

(3)后台启动方式nohup ./activemq start >/tmp/smlog

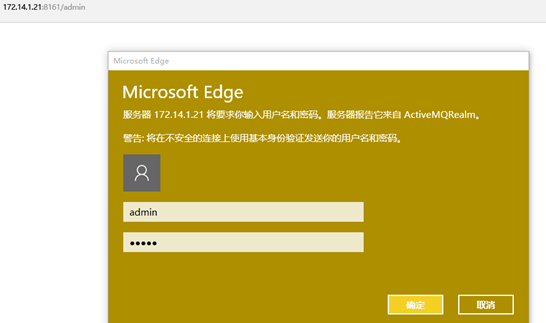
注： 一般第一种情况下即可正常启动。

前两种方式下在命令行窗口关闭时或者ctrl+c时可能导致进程退出，采用第三种方式则可以避免这种情况。

第一种方式后面添加 & 符号，即./activemq start & 也可以后台运行。如果启动失败，具体情况需要具体分析

* + 1. 测试是否启动成功。有两个步骤
       1. activemq运行后没有提示，可以使用命令 ps –ef|grep activemq查看是否有刚启动的相关进程，如下
       2. 能否登陆管理后台。

管理后台默认地址格式: IP地址：port(端口,默认8161）/admin，如192.168.54.239:8161/admin。管理后台默认用户/密码：admin/admin



存在进程且能正常进入管理后台则表示activemq安装并启动成功。



### ACTIVEMQ介绍

#### MQ简介

MQ全称为Message Queue, 消息队列（MQ）是一种应用程序对应用程序的通信方法。应用程序通过写和检索出入列队的针对应用程序的数据（消息）来通信，而无需专用连接来链接它们。消息传递指的是程序之间通过在消息中发送数据进行通信，而不是通过直接调用彼此来通信，直接调用通常是用于诸如远程过程调用的技术。排队指的是应用程序通过队列来通信。队列的使用除去了接收和发送应用程序同时执行的要求。

MQ特点：

MQ的消费-生产者模型的一个典型的代表，一端往消息队列中不断的写入消息，而另一端则可以读取或者订阅队列中的消息。MQ和JMS类似，但不同的是JMS是SUN Java消息中间件服务的一个标准和API定义，而MQ则是遵循了AMQP协议的具体实现和产品。

使用场景：

在项目中，将一些无需即时返回且耗时的操作提取出来，进行了异步处理，而这种异步处理的方式大大的节省了服务器的请求响应时间，从而提高了系统的吞吐量。

#### JMS简介

JMS的全称是Java Message Service，即Java消息服务。它主要用于在生产者和消费者之间进行消息传递，生产者负责产生消息，而消费者负责接收消息。把它应用到实际的业务需求中的话我们可以在特定的时候利用生产者生成一消息，并进行发送，对应的消费者在接收到对应的消息后去完成对应的业务逻辑。对于消息的传递有两种类型，一种是点对点的，即一个生产者和一个消费者一一对应；另一种是发布/订阅模式，即一个生产者产生消息并进行发送后，可以由多个消费者进行接收。

JMS即Java消息服务（Java Message Service）应用程序接口是一个Java平台中关于面向消息中间件（MOM(Message Oriented Middleware)）的API，用于在两个应用程序之间，或分布式系统中发送消息，进行异步通信。Java消息服务是一个与具体平台无关的API，绝大多数MOM提供商都对JMS提供支持。

定义：

JMS是Java平台上有关MOM的技术规范，它便于消息系统中的Java应用程序进行消息交换,并且通过提供标准的产生、发送、接收消息的接口简化企业应用的开发，翻译为Java消息服务。

简介：

JMS是一种与厂商无关的 API，用来访问消息收发系统消息。它类似于JDBC(Java DatabaseConnectivity)：这里，JDBC 是可以用来访问许多不同关系数据库的 API，而 JMS 则提供同样与厂商无关的访问方法，以访问消息收发服务。许多厂商目前都支持JMS，包括 IBM 的 MQSeries、BEA的 Weblogic JMS service和 Progress 的 SonicMQ，这只是几个例子。 JMS 使您能够通过消息收发服务（有时称为消息中介程序或路由器）从一个 JMS 客户机向另一个JMS客户机发送消息。消息是 JMS 中的一种类型对象，由两部分组成：报头和消息主体。报头由路由信息以及有关该消息的元数据组成。消息主体则携带着应用程序的数据或有效负载。根据有效负载的类型来划分，可以将消息分为几种类型，它们分别携带：简单文本(TextMessage)、可序列化的对象 (ObjectMessage)、属性集合 (MapMessage)、字节流 (BytesMessage)、原始值流 (StreamMessage)，还有无有效负载的消息 (Message)。

#### JMS和MQ的关系

JMS是一个用于提供消息服务的技术规范，它制定了在整个消息服务提供过程中的所有数据结构和交互流程。而MQ则是消息队列服务，是MOM的最终实现，是真正的服务提供者；MQ的实现可以基于JMS，也可以基于其他规范或标准。

#### ActiveMQ简介

是Apache出品，最流行的，能力强劲的开源消息总线。ActiveMQ 是一个完全支持JMS1.1和J2EE 1.4规范的 JMS Provider实现,尽管JMS规范出台已经是很久的事情了,但是JMS在当今的J2EE应用中间仍然扮演着特殊的地位。

主要特点：

1. 多种语言和协议编写客户端。语言: Java, C, C++, C#, Ruby, Perl, Python, PHP。应用协议: OpenWire,Stomp REST,WSNotification,XMPP,AMQP

2. 完全支持JMS1.1和J2EE 1.4规范 (持久化,XA消息,事务)

3. 对spring的支持,ActiveMQ可以很容易内嵌到使用Spring的系统里面去,而且也支持Spring2.0的特性

4. 通过了常见J2EE服务器(如 Geronimo,JBoss 4, GlassFish,WebLogic)的测试,其中通过JCA 1.5 resource adaptors的配置,可以让ActiveMQ可以自动的部署到任何兼容J2EE 1.4 商业服务器上

5. 支持多种传送协议:in-VM,TCP,SSL,NIO,UDP,JGroups,JXTA

6. 支持通过JDBC和journal提供高速的消息持久化

7. 从设计上保证了高性能的集群,客户端-服务器,点对点

8. 支持Ajax

9. 支持与Axis的整合

10. 可以很容易得调用内嵌JMS provider,进行测试

11. ActiveMQ速度非常快；一般要比jbossMQ快10倍。

优点：是一个快速的开源消息组件(框架)，支持集群，同等网络，自动检测，TCP，SSL，广播，持久化，XA，和J2EE1.4容器无缝结合，并且支持轻量级容器和大多数跨语言客户端上的Java虚拟机。消息异步接受，减少软件多系统集成的耦合度。消息可靠接收，确保消息在中间件可靠保存，多个消息也可以组成原子事务。

缺点：ActiveMQ默认的配置性能偏低，需要优化配置，但是配置文件复杂，ActiveMQ本身不提供管理工具；示例代码少；主页上的文档看上去比较全面，但是缺乏一种有效的组织方式，文档只有片段，用户很难由浅入深进行了解，二、文档整体的专业性太强。在研究阶段可以通过查maillist、看Javadoc、分析源代码来了解。

#### 查看ActiveMQ队列消费情况

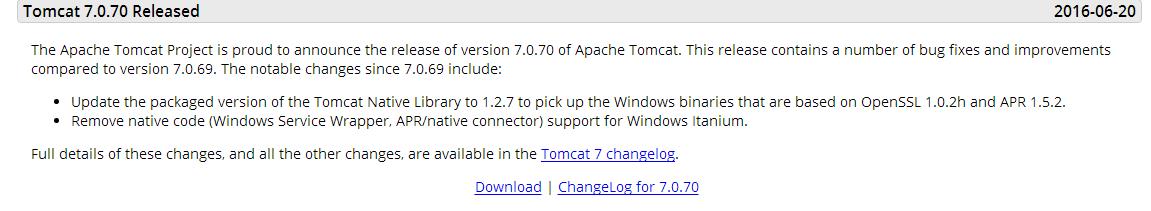
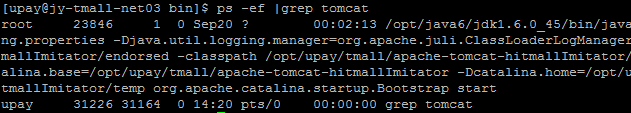
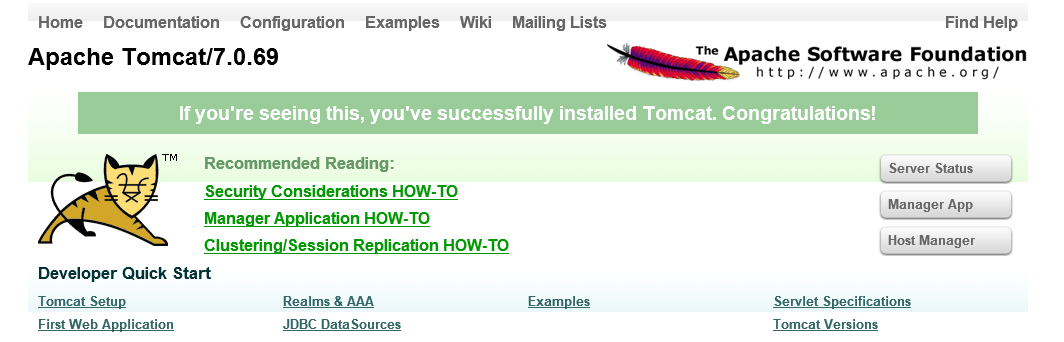
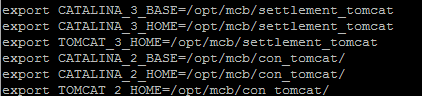
<http://localhost:8161/admin/>访问本地ActiveMQ（下面简称MQ）的web console，默认用户名密码是admin/admin。登录后在菜单栏找到Queues选项，点击进入队列消息页面，在它的右边还有一个Topics选项，这里面是主题消息页面。

不同版本的MQ可能有不同，在5.7版本的队列消息页面会看到队列消息列表，其中有程序自带的死信队列：ActiveMQ.DLQ，这里放的是超时没有被消费的消息。这些消息是可以重新分配给其他队列的。

每个消息都有7列信息，第一列是队列名（Name），第二列是队列中的消息（Number Of Pending Messages ），第三列是队列消费者数目（Number Of Consumers ），第四列是接收到的队列消息数目（Messages Enqueued ），第五列是发送出去的消息数目（Messages Dequeued），第六列是查看队列详细信息（Views），第七列是可以对消息进行的操作（Operations）。

通过查看MQ队列的消费情况，可以了解程序是否正常运行。

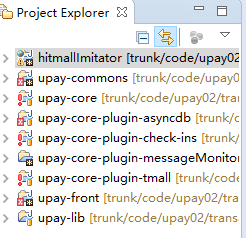
### TOMCAT下载和安装

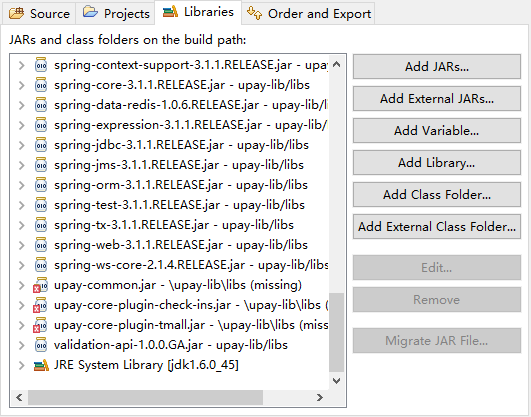
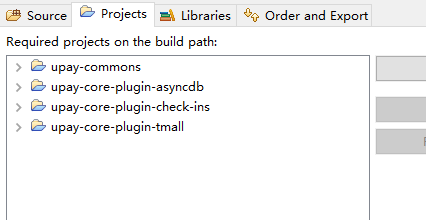
* 1. 下载 tomcat
     1. 打开官网<http://tomcat.apache.org/>
     2. 找到如下图的tomcat7
     3. 点击Download 超链接文字，到达如下下载页，选择Full documentation下的tar.gz文件下载，下载完成后得到文件apache-tomcat-7.0.72 .tar.gz
  2. 安装部署
     1. 把安装包上传到指定的服务器目录中，并使用解压命令：tar -xvzf apache-tomcat-7.0.72 .tar.gz
     2. 解压后得到文件夹 apache-tomcat-7.0.72，命令：cd apache-tomcat-7.0.69/bin 进入存放启动tomcat的start.sh的文件夹
     3. 使用命令：sh start.sh启动tomcat
     4. 测试是否启动成功
        1. 启动后，使用命令：ps –ef|grep tomcat 查看是否有相关进程如图中有两个进程，第一个是tomcat的进程，第二个是查询tomcat进程是否存在的进程，所以如果只有第二个是不行的，必须有第一个
        2. 在本地浏览器访问启动tomcat的服务器的8080端口，查看是否有tomcat启动。本地是http://localhost:8080，出现tomcat主页即为安装部署成功
        3. 如果启动出错，可以到tomcat安装路径/logs 目录下，找到catalina.当前日期.log 查看启动出错原因
     5. 增加系统变量： 变量名：TOMCAT\_HOME；变量值：D:\DevTools\apache-tomcat-8.0.3
     6. 修改系统变量classpath，添加%TOMCAT\_HOME%\lib\servlet-api.jar同时修改系统变量path，添加%TOMCAT\_ HOME%\bin
  3. 异常处理
     1. 修改Tomcat默认端口
        1. 在一个服务器上部署多个tomcat，或者服务器8080端口被其他应用程序占用的情况下，需要修改tomat默认的启动端口
        2. 找到 tomcat安装路径/conf/server.xml ，文件里面的<Connector port="8080" protocol=”HTTP/1.1”....../>，把port的值改为其他的未被使用的端口号就可以了
        3. 查看端口是否被占用：netstat -apn|grep 端口号。如果有进程则被占用，否则没有被占用
     2. 有时候tomcat启动没有报错，没有发现对应的线程，却有其他tomcat的线程。这种情况可能是服务器上部署有多个tomcat，并且那个tomcat被添加到了环境变量中。这个时候我们启动自己的tomcat，系统会默认启动在环境变量中设置的tomcat，所以我们需要修改环境变量和tomcat启动文件startup.sh。
        1. 修改环境变量：vi /etc/profile，在里面添加自己的tomcat变量:如下图 每个tomcat都需要设置自己的tomcat，分别设置CATALINA\_BASE、CATALINA\_HOME、TOMCAT\_HOME
        2. 在tomcat的startup.sh和shutdown.sh里配置export CATALINA\_BASE=$CATALINA\_3\_BASE export CATALINA\_HOME=$CATALINA\_3\_HOME

### 本地项目环境搭建

* 1. 下载项目，svn账号密码找相关人员咨询，路径：https://192.168.59.18/svn/UPAY2\_SVN/trunk/code/upay02/transaction/release

项目从svn下载下来之后会有一些错误，需要手动进行调整。

刚下载下来的项目如右图所示，有一些错误。

* 1. 先解决upay-commons项目的出错，这个项目里的测试类有错误，直接注释掉或者删除出错的测试文件就好啦
  2. Upay-front项目也是测试类出错，直接删除或注释掉就可以了
  3. 剩下的都是核心相关的文件出错了。打开upay-core项目如图所示有这样的错误，把出错的jar包移除。并且左边的Projects选项卡里设置项目依赖，upay-core项目要依赖upay-common和其他的核心插件项目upay-core-plugin-tmall、upay-core-plugin-check-ins、upay-core-plugin-asyncdb，如下图所示。
  4. 红色感叹号代表是jar引用错误，所以把生下来出错的项目的出错jar包移除。核心插件项目默认引用upay-common.jar和upay-core.jar，改为依赖upay-common项目，upay-core.jar重新引用，该jar包在upay-core编译后的项目下
  5. 在处理完项目的jar包问题引起的红色感叹号错误之后，会发现upay-core-plugin-tmall和upay-core-plugin-asyncdb还剩下红叉错误，打开项目后发现，后者是因为没有引用bcp的jar包，但是项目中又需要使用这个包里的文件而出错，所以从upay-lib项目下引入upaybcp.jar就可以了。前者的错误是因为核心jar包有问题，重新编译一下核心jar包就可以，运行upay-core项目下的upayCoreBuild.xml文件就可以了，如果出错，重新编译一下upay-common项目
  6. 做完以上操作如果还有异常，寻求其他人员帮助

### 项目的部署

项目部署主要分5步（以upay-front项目为例）：

* 1. 打包

在项目里找到upayFrontbuild.xml，选中文件右键选择Run As->Ant Build

当控制台打印出BUILD SUCCESS字样则表示打包成功，否则失败。

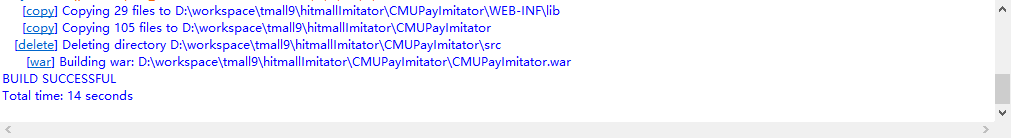
* + 1. 失败后的解决方案：

如果出现”找不到符号“等字样的错误时，重新编译upay-commons项目，编译upay-commons项目的方法是找到upay-commons项目里的upayCommonbuild.xml进行编译

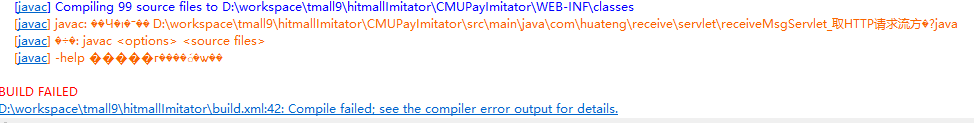
* 1. 上传到服务器，把打包后的zip包上传到服务器，服务器必须先安装和配置好JDK
  2. 解压项目，使用命令：unzip upay-front.zip命令解压文件，解压后的目录名为upay-front
  3. 修改项目配置文件，具体参考 
  4. 启动程序调试，启动程序后，查看启动目录下的nohup.out文件，有Start upay server ok, now ready service!即为启动成功

### 仿真项目的部署

* 仿真项目在生产环境中一般不会部署，只会部署在测试环境中
  1. 打包，仿真项目的打包和其他项目一样使用build方式，找到项目里的build.xml文件，和上一模块同样的操作进行打包，出现如下结果即为编译成功



* + 1. 编译出错

过程中出现这个错误是因为项目中有中文命名的文件，根据指示的路径找到文件并修改文件名为英文就可以了，再次编译后还会报出同样的错误，同样的方式修改后再编译。

* 1. 上传到服务器，打包后得到一个CMUPayImitator文件夹，在文件夹下面有一个CMUPayImitator.war的文件。把这个文件上传到的服务器上，服务器必须先安装和配置好JDK和tomcat
  2. 解压项目，因为仿真项目是一个web项目，所以和其他项目不同。它的解压是直接把war包放到tomcat的webapps目录下，在启动tomcat后就行自动被tomcat服务器解压并部署
  3. 修改项目配置文件CMUPayImitator/WEB-INF/classes/ server.properties，修改如下配置项为实际使用的对应参数。TmallFrontURL=http://192.168.54.239:8088/cmupay/tmall.cgi

CRMAddressURL=http://192.168.54.241:8088//cmupay/crm.cgi这两个配置项一个是天猫充值查询前置，一个省地址。修改IP和端口就可以了

upay1\_imitator=false和log\_imitator=false修改为false

其他需要注意的配置如下：

#签名标识（open为开）\\

signFlag=close

#验签标识（open为开）

checkSignFlag=close

#加密标识（open为开）

encodeFlag=close

#解密标识（open为开）

decodeFlag=close

#http请求超时时间，默认60秒

tmall\_request\_http\_time\_out=60

#省侧延时

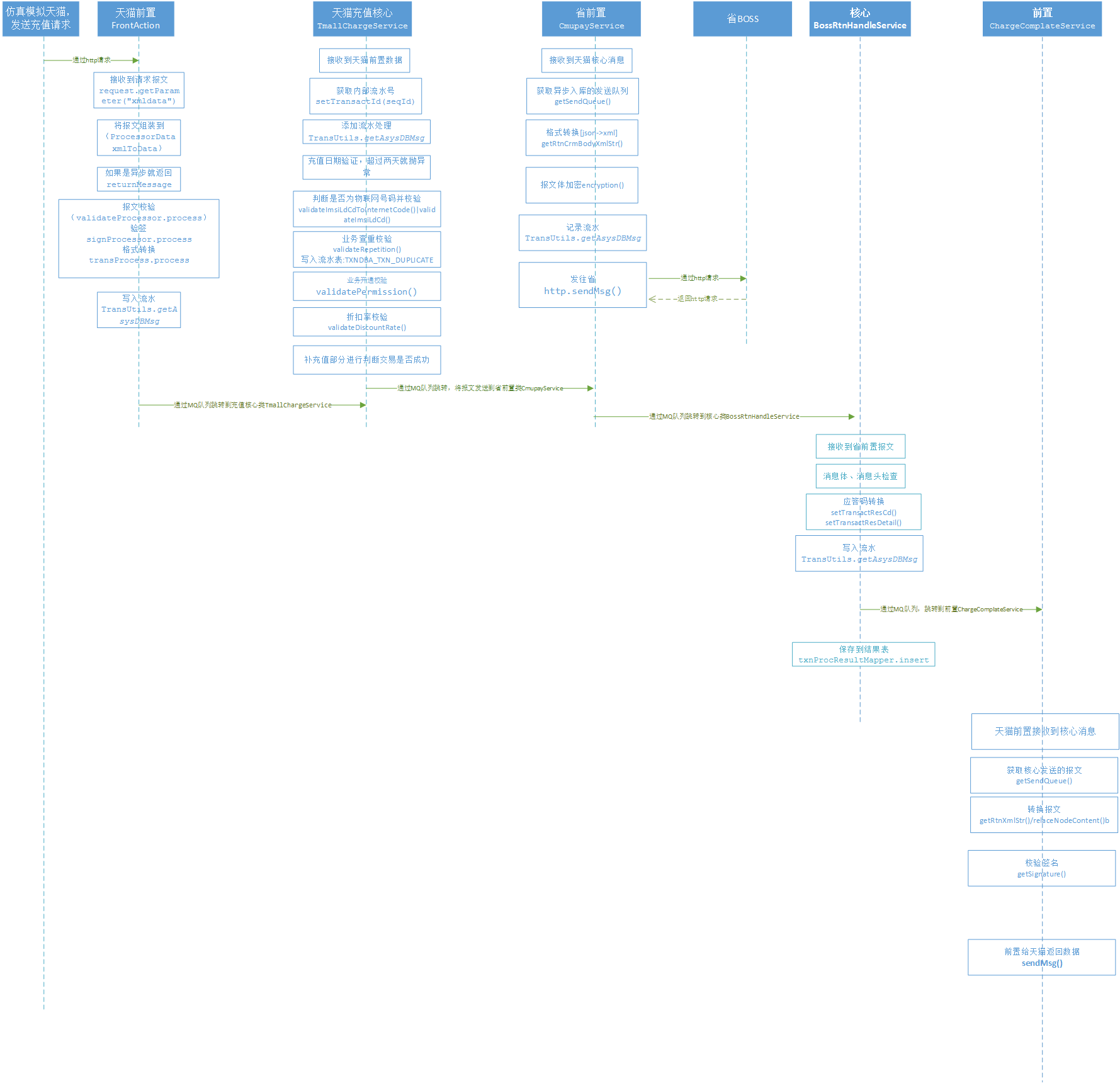
crmDelay=10

#天猫延时

tmallDelay = 10

* 1. 启动程序调试，启动tomcat即启动，tomcat的安装启动参考上面。在浏览器输入地址，出现上图即为启动成功，输入用户名/密码admin/abc-1234即可登陆。用户名和密码可以在项目的server.properties配置文件里配置，不过一般不需要进行修改

### 充值流程

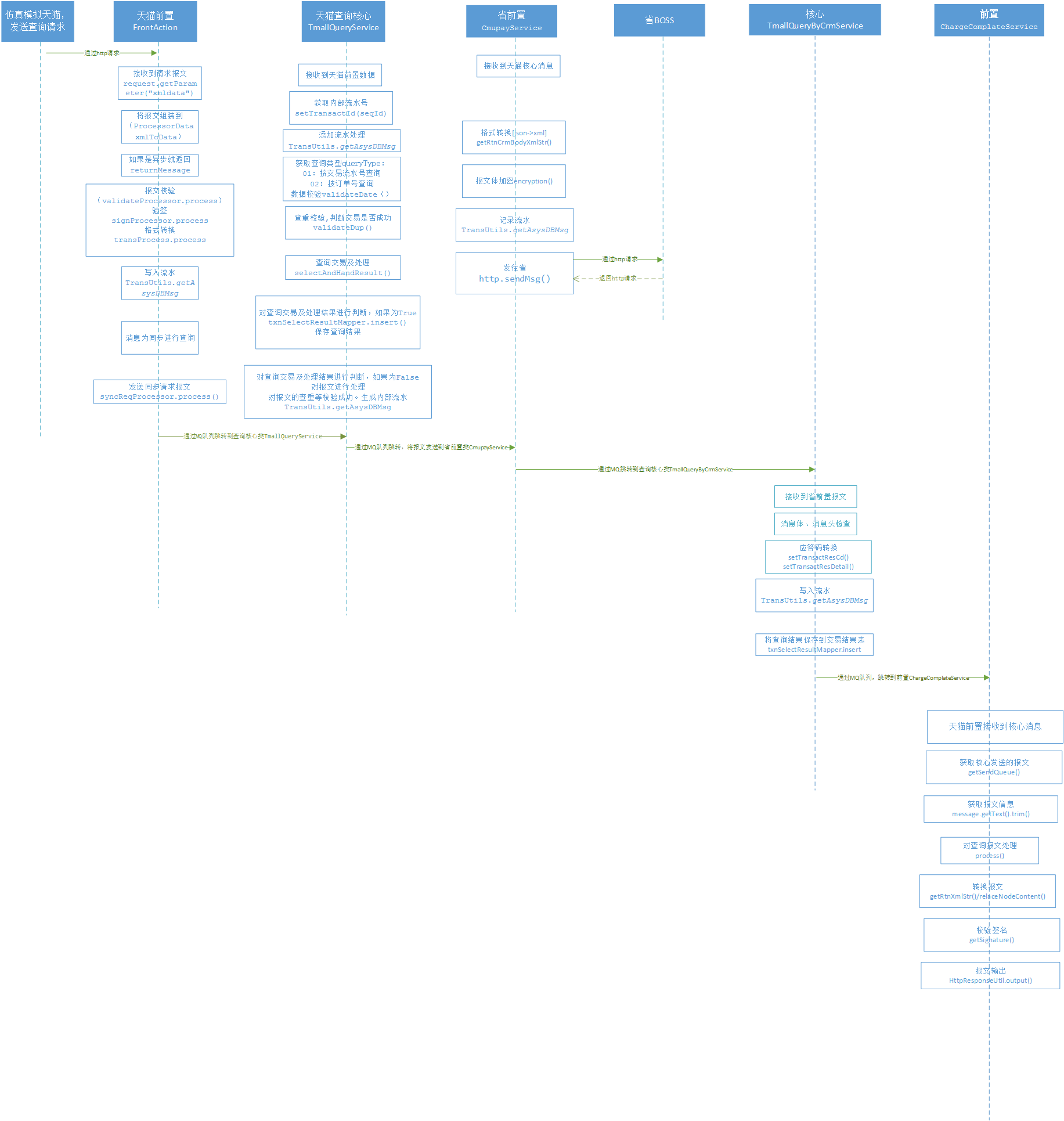


* 1. 仿真模拟天猫发送http post请求 ，将xml报文发送到前置FrontAction 。FrontAction 获取 xmldata（仿真发过来的，通过request.getParameter("xmldata")方法，如果是异步消息直接返回应答消息给天猫，不一定充值成功，只是给一个响应，说明已经成功接收。

校验报文，校验属性的值是否为空，值是否符合规范，通过模版校验，转换报文将xmldata转换成ProcessorData对象，异步入库第1步，processAsyncRequest()通过upay-tmall-core-req MQ队列跳转到核心的TmallChargeService.doService(UnionMsgBean unionMsgBean, Map<String, String> msgStrProperty)方法。在此之前异步入库第2步。

* 1. TmallChargeService 获得内部的交易流水号，添加流水处理
  2. 异步入库第3步 充值日期验证，订单总金额支付总金额验证 物联网号段校验：分为两种1.手机号 2.物联网号，先根据mobile7在缓存中获取号码段信息表，在根据号段信息获取国内长途区号表，业务查重校验入查重表 ，插入有异常认为重复充值，获取外部机构代码来业务开通权限效验，折扣率校验 ，补充值 ：如果getTransactResCd为010A00才需要，记录流水
  3. 异步入库第4步，根据orgID在缓存中获得省份Boss路由信息.并且设置到unionMsgBean中，通过队列upay\_core\_boss\_req ，跳到CmupayService.handle(Message message)方法处理
  4. CmupayService 省前置收到交易核心请求报文，记录流水
  5. 异步入库第5步，根据模版获取bossHeader.xml,bossBody.xml，报文体加密，记录流水
  6. 异步入库第6步，根据路由发送http请求调用省Boss系统，省Boos系统校验并充值话费，并返回结果报文，如果返回为空，httpSend==null，则错误码：UPAY\_T\_015A07超时未收到响应，CmupayService 解密，并解析结果报文 ，获得返回结果。记录流水
  7. 异步入库第7步 ，报文格式转换开始-xml to json 获取CDATA内容，报文解密，获取报文内容，将获取到的报文内容放入unionMsgBean中，将解密后的报文重新组装，记录流水第八步，upay-core-boss-rtn队列到BossRtnHandleService.doService()方法处理
  8. BossRtnHandleService接收到省前置消息， 开始处理，记录流水，异步入库第9步，获得交易返回码，操作返回码，校验返回码，记录流水
  9. 异步入库第10步，保存充值结果表，通过upay-tmall-core-rtn，jms.send()方法跳转到ChargeComplateService.handle(Message msg)方法
  10. ChargeCompleteService 将req\_message的json数据转换成unionMsgBean对象，写入流水
  11. 异步入库第11步 ，根据模板把消息转换为xml格式的字符串，根据配置文件的开关决定是否进行签名处理。异步入库第12步，通过hessian调用远程接口， 然后调用请求方的接口：HttpTransport http = new HttpTransport() ，获取响应浙商回调地址，发送http post请求：http.sendMsg(rspUrl, tmallXml)将将充值结果响应给浙商，收到的响应内容为空则抛出015A07：超时未收到响应，如果响应的xml不为空，获取rsqCode,rspDese,记录全网监控日志。

### 查询流程



* 1. 仿真模拟天猫发送http post请求 ，带xml报文，到前置FrontAction 获取 xmldata（仿真发过来的，通过request.getParameter("xmldata")方法），通过012004.xsd文件模版效验，验签1已签名，0未签名，转换报文将xmldata转换成ProcessorData对象，通过upay-tmall-core-query-req MQ队列跳转到核心的TmallQueryService.doService()处理
  2. TmallQueryService 获得内部流水号，获取查询类型queryType："01", "按交易流水号查询"，"02", "按订单号查询"然后进行数据校验，校验oriReqTransID，OriOrderID是否为空。查询原交易及处理：1.查询查重表数据，将reqsys,reqTransID,settleDate设置到map中当作查询条件，查到说明存在，否则抛出BusinessException异常该笔交易不存在014A05。2.根据条件查询交易查询结果表，如果结果为空则抛出BusinessException异常，错误码014A08交易处理中，没有则进行相应的处理，如果交易结果表存在记录,判断交易状态是否成功，如果成功直接响应消息，returne true,发送revieveQueue.getValue()队列，查询结果表入库。否则：3.业务开通校验，外部机构交易代码映射表和机构交易权限表处理，根据省份找到省Boss路由信息.，routeUrl设置到unionMsgBean对象中，发送upay-core-boss-query-req队列，跳转到CmupayService省前置处理。
  3. CmupayService 省前置收到交易核心传的json数据，转换成unionMsgBean对象，记录流水 第5步，获取bossHeader报文，bossBody报文，报文体加密，记录流水第6步，根据路由发送http请求调用省Boss系统，省Boos系统校验并返回充值结果报文，如果返回报文为空则抛出NetworkExceptionCmupayService异常，错误码：015A07 超时未收到响应。记录流水第7步 ，解密 并解析结果报文 ，获取CDATA内容，报文解密，获取报文内容，将获取到的报文内容放入unionMsgBean中，将解密后的报文重新组装，记录流水第8步，upay-core-boss-query-rtn队列到TmallQueryByCrmService.doService()方法处理。
  4. TmallQueryByCrmService 获得操作返回码，查询结果返回码。效验返回码，如果返回码和现有的返回都对应不上，则抛出ArgumentException异常，错误码：015A06,未知错误，交易查询结果入库，upay-tmall-core-query-rtn对列，跳转到ChargeComplateService.handle()处理。
  5. ChargeComplateService前置，将req\_message json格式的数据转换成unionMsgBean对象中，获取thread\_key, 在中process()处理，天猫前置向浙商:HttpResponseUtil.output(response, Status.OK, xmlmsg);返回给仿真查询结果报文，HttpResponseManger.removeResponseInfo(thread\_key)
  6. 流程图

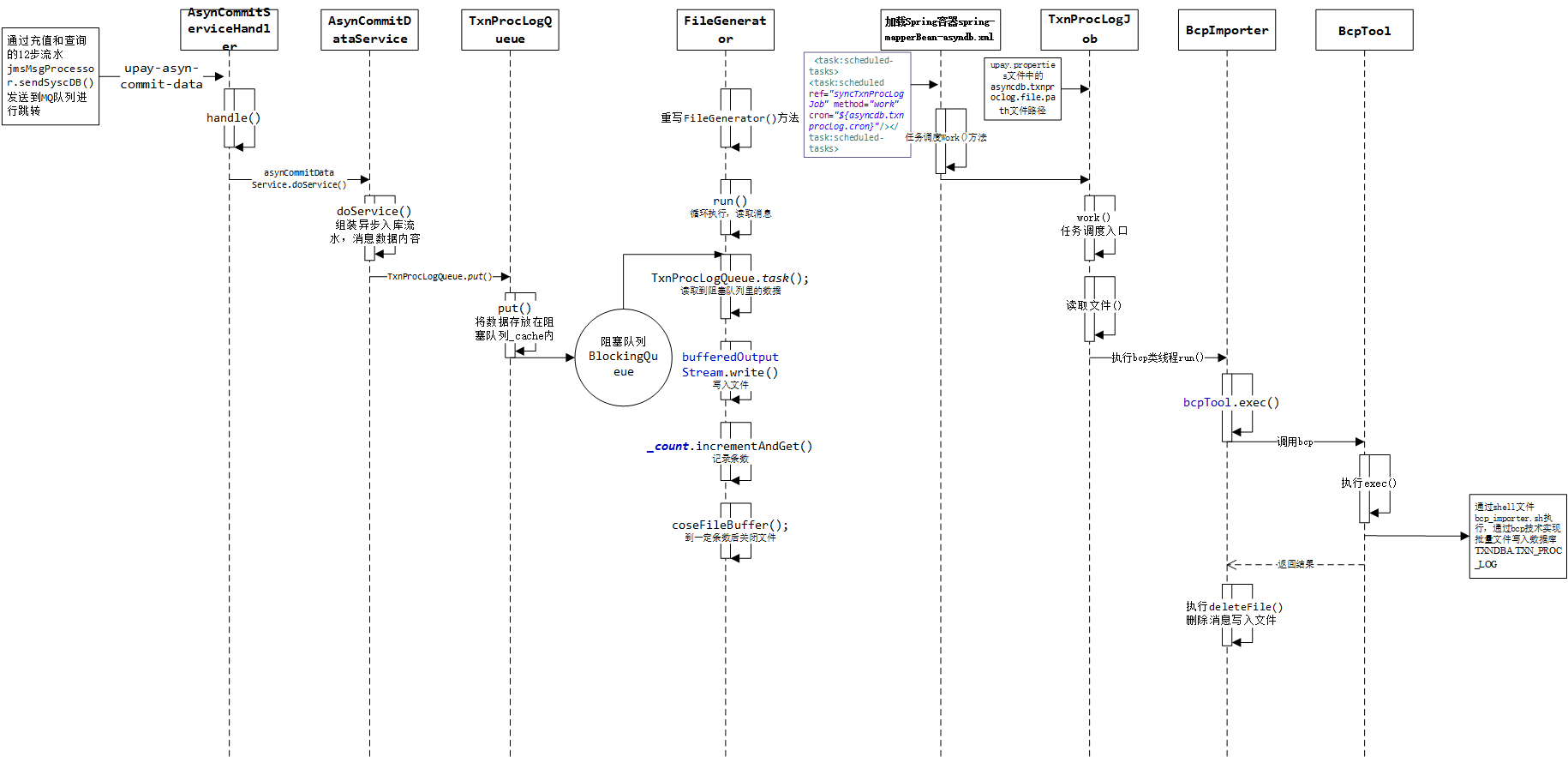
### 异步入库的流程

* 1. 异步入库功能点描述

监听消息服务器的交易流水记录请求队列,当消息到达该队列时接收交易流水记录消息,并将交易流水记录持久化到数据库，再将处理结果发送到交易流水记录响应队列。

插件项目：upay-core-plugin-asyncdb

* 1. 流程图



* 1. 业务流程(发送一条流水为例)

1. 充值业务中的天猫前置FrontAction，执行方法 jmsMsgProcessor.sendSyscDB(asynSenQueueBean.getValue(), json);发送异步入库消息。

2. 接收MQ消息，处理类AsynCommitServiceHandler，执行handle(Message msg)。

3. doService(String jsonBody)组装异步入库消息，并将组装的消息存入阻塞队列TxnProcLogQueue.put(txnProcLogString);

4. 调用FileGenerator类中FileGenerator()方法。

5. 执行updateTmpFileName()方法。获取核心配置文件upay.properties中asyncdb.txnproclog.file.path指定路径下的文件。

6. 调用execute()启动线程。

7. 取出拼装好的流水TxnProcLogQueue.task();

8. 记录取出流水的数目\_count.incrementAndGet()。当取出的数目达到150000（可以在配置文件配置）条，或者文件的大小达到3M时，停止写入。

9. 加载Spring容器spring-mapperBean-asyndb.xml。 通过文件中的

<task:scheduled-tasks>

<task:scheduled ref="syncTxnProcLogJob" method="work" cron="${asyncdb.txnproclog.cron}"/>

</task:scheduled-tasks> 启动调度任务work()。

10. 执行TxnProcLogJob类中work()方法。

11. 获取核心配置文件upay.properties中asyncdb.txnproclog.file.path指定路径下的文件。

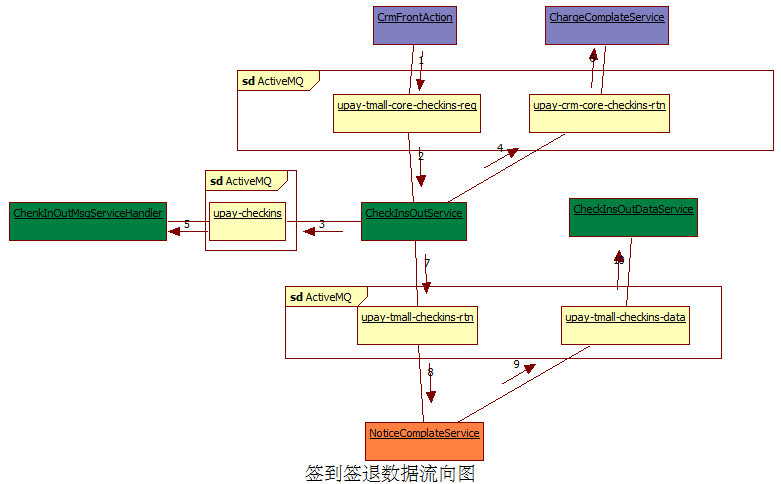
12. 启动线程executor.execute(bcpImporter)，运行run()方法。

13. 在run()方法中，通过调用bcp进行文件导入数据库中 bcpTool.exec(),而其中最终执行bcp导入的方法为：bcpLocalExcute()。文件导入的数据库为 ：TXN\_PROC\_LOG(交易流水表)

14. 文件导入成功，返回true。删除本地文件deleteFile();

### 12 签到签退的流程

概述：监听消息服务器签到签退请求队列,当签到签退请求到达该队列时接收签到签退请求消息,生成内部流水号, 发送签到签退信息流水到异步入库队列,修改签到签退信息到数据库, 发送签到签退信息到TOPIC,发送签到签退流水信息到异步入库队列,记录签到签退结果到结果库,发送签到签退结果消息到前置队列。



插件项目：upay-core-plugin-check-ins

* 1. 省前置(CrmFrontAction)接收到签到签退的消息。
  2. 在入口方法execute（）进行报文数据校验和格式转换。报文里的参数值在使用.xsd校验文件进行校验的时候不符合设定的规则，就会报错MsgCheckException，错误码4A99。Xml格式的报文会使用.json文件被转换成json格式，在这个过程中出现异常，会报出异常MsgTransformException，错误码UPAY\_T\_015A05。
  3. 调用发送方法processSyncRequest() 把请求转发到交易核心处理类CheckInsOutService
  4. CheckInsOutService更新机构权限表、发送topic消息通知其他core进行签到签退、将签到签退信息写入到签到签退结果表，入口方法doService()。
  5. 在更新机构权限表和发送topic消息到其他core的时候发生SQL异常和JMS异常，将直接把签到签退信息写入到签到签退结果表。
  6. ChenkInOutMsgServiceHandler监听upay-checkins队列消息，更新统一支付系统的orgTransCodeDB。
  7. ChargeComplateService接收到upay-crm-core-checkins-rtn队列消息后执行入口方法handle()，把签到签退响应返回给省boss。
  8. NoticeComplateService在监听到upay-tmall-checkins-rtn 队列消息后被调用，入口方法handle()。接收核心的通知报文，把json报文转换成xml格式报文后签名，然后发送到核心。
  9. CheckInsOutDataService的入口方法doService()在监听到upay-tmall-checkins-data消息后被调用。这个类主要就是更新TXN\_CHECKINS\_RESULT或往里面插入数据。



### 前线库导后线库流程

功能点概述：统一支付的数据(oracle)定期导入到清结算数据库中(sysbase)

插件项目：upay-sync-db

* 1. 业务流程
     1. 加载数据库连接配置信息
     2. 加载系统配置信息
     3. 加载spring配置信息，启动一个定时时间计划任务。
     4. 根据加载的系统配置信息中的syncdb.cron配置项每隔配置的时间段执行导出组件
     5. 连接oracle前线库。查询T-1日的交易结果表TXN\_PROC\_RESULT中的交易数据
     6. 将查询出的交易数据按照系统配置信息中的目录，分隔符配置信息导出到文本文件中。
     7. 记录导出时的运行日志
  2. 代码流程

第一步:首先进入RunMain类(程序入口)

第二步:调用 SystemInitor类该类主要对系统进行初始化init()方法进行初始化(

加载数据库连接配置信息

加载系统配置信息

加载spring配置信息，启动一个定时时间计划任务初始化Log,初始化IOC容器,加载xml配置文件到缓存,启动文件监控器, FTP文件下载监控)

第三步：当加载spring配置信息，启动一个定时时间计划任务(spring-syncdb.xml中的任务<task:scheduled-tasks>).里面包含三个任务syncTxnProcResultJob、 syncTxnSelectResultJob、 databaseMaintainJob

* 1. syncTxnProcResultJob执行流程
     1. 对应SyncDBJob类，执行work方法：
        1. 删除sybase中临时表的数据
        2. 防止重复导入数据，删除sybase表分区
        3. jdbcTemplate.query查询前线库数据并把参数传给MyRowMapperResultSetExtractor
     2. MyRowMapperResultSetExtractor继承ResultSetExtractor
        1. 执行extractData(ResultSet)方法，调用BcpImportRowHandler类
     3. 依次执行BcpImportRowHandler里的init()、 handler(Map<String, Object>)、 finish()方法
        1. init()方法备份文件
        2. handler(Map<String, Object>)方法把查询到的数据，调用BcpImporter类的writeLine()写入到文本文件里
        3. finish()调用BcpImporter类run方法把数据写入到sybase数据库业务表、临时表、备份到文件目录
     4. upayFileloadStatusStrategy记录清结算状态
  2. databaseMaintainJob执行流程
     1. 对应DatabaseMaintainJob类，执行work()方法
     2. 循环调用ExecShellCmd类的callShell()方法，执行以下脚本

# 数据库维护脚本路径，添加oracle分区，将(tnsname)替换为本地oracle客户端配置的tnsname，替换后不留括号

database.maintain.addparts\_oracle=/upay/home/oracle/scripts/addparts\_oracle.sh -u ${datasource.user2} -p ${datasource.pass2} -t (tnsname)

#　数据库维护脚本路径，删除oracle分区，将(tnsname)替换为本地oracle客户端配置的tnsname，替换后不留括号

database.maintain.dropparts\_oracle=/upay/home/oracle/scripts/dropparts\_oracle.sh -u ${datasource.user2} -p ${datasource.pass2} -t (tnsname)

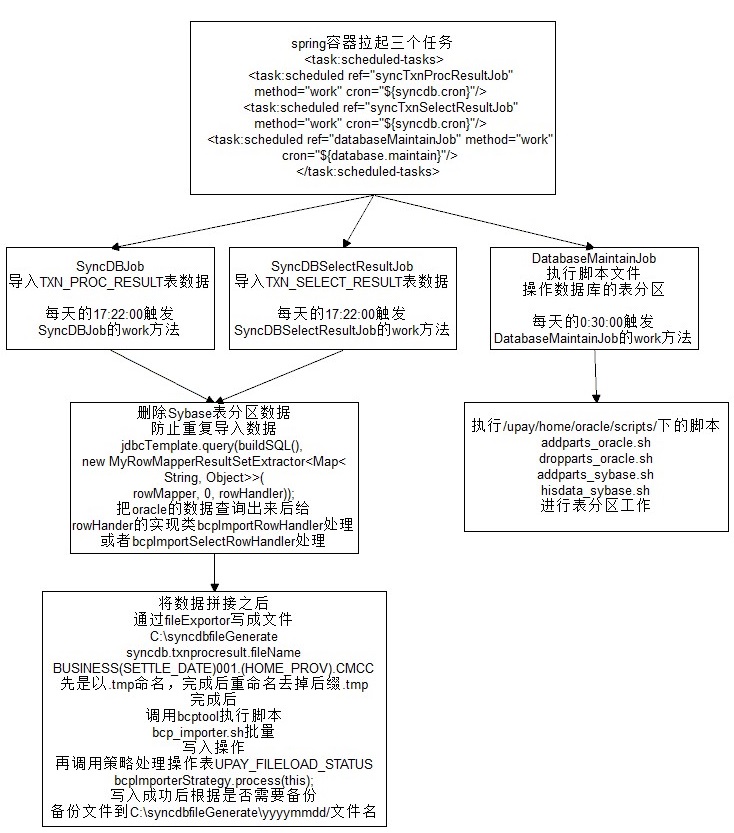
#　数据库维护脚本路径，添加sybase分区，将(serverName)替换为本地sybase客户端配置的serverName，替换后不留括号

database.maintain.addparts\_sybase=/upay/home/oracle/scripts/addparts\_sybase.sh -U ${datasource.user.sybase} -P ${datasource.pass.sybase} -S (serverName)

#　数据库维护脚本路径，数据从业务表迁移至历史表、删除sybase分区，将(serverName)替换为本地sybase客户端配置的serverName，替换后不留括号

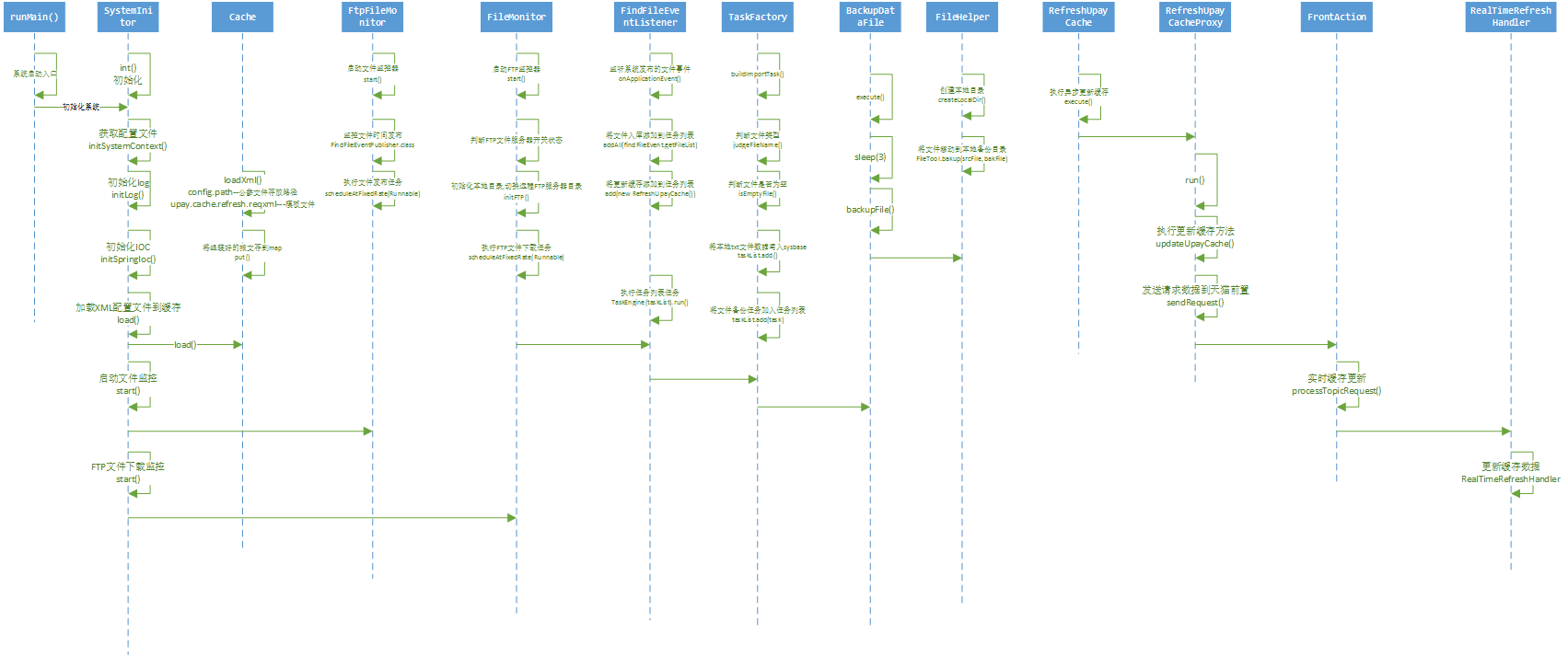
database.maintain.dropparts\_sybase=/upay/home/oracle/scripts/hisdata\_sybase.sh -U ${datasource.user.sybase} -P ${datasource.pass.sybase} -S (serverName)

* 1. 流程图





### 公参更新流程



概述：外部系统定时发布最新公参数据到指定服务器，发布文本为约定txt文件格式，公参同步功能从服务器通过ftp下载文件到本地后，读取文件将数据导入到实时交易数据库(oracle)，并通知实时交易系统更新缓存。



* 1. 代码流程
     1. 系统启动，运行upay-sync-db项目里的RunMain类
     2. 加载配置文件、初始化日志系统、加载缓存、启动ftp监控器FtpFileMonitor，本地文件监控器FileMonitor。
     3. 定时检测ftp服务器是否有发布的公参数数据，发现有则下载到本地路径。可在配置文件里配置pubdba.ftp.get=on/off开启/关闭监控。
     4. 程序将本地目录中经过FileHelper过滤器的正则表达式判断过的，约定的txt数据文件读取为对象，便于操作
     5. 用spring的Event事件处理FindFileEvent事件消息，FindFileEventPublisher发布事件，FindFileEventListener监听事件。FindFileEventListener监听到事件后，对各个文件拉起任务，创建可执行的任务队列taskList，放到任务引擎TaskEngine(taskList)
     6. 执行任务。每一个文件拉起一个插入数据库和备份文件的任务，在任务异常时，恢复任务前的状态或者按设置的重试次数重新执行任务
     7. 更新缓存。RefreshUpayCacheProxy的run()调用updateUpayCache()，该方法获得Cache缓存的报文模板组装报文，发送更新后的缓存数据到在配置文件里配置的tmall路径，如upay.cache.refresh.url=http://127.0.0.1:8088/cmupay/tmall.cgi



* 1. 公参流程图

