# 支付系统

[支付系统 1](#_Toc17206935)

[支付系统 2](#_Toc17206936)

[什么是支付系统 2](#_Toc17206937)

[支付系统模型 2](#_Toc17206938)

[平台 3](#_Toc17206939)

[业务系统 3](#_Toc17206940)

[支付系统 3](#_Toc17206941)

[支付系统架构 4](#_Toc17206942)

[支撑层 4](#_Toc17206943)

[核心层 6](#_Toc17206944)

[产品层 7](#_Toc17206945)

[支付系统设计与实现 8](#_Toc17206946)

[三户模型 8](#_Toc17206947)

[客户管理系统 9](#_Toc17206948)

[用户管理系统 11](#_Toc17206949)

[账户管理系统 13](#_Toc17206950)

[账务体系 19](#_Toc17206951)

[支付网关系统 24](#_Toc17206952)

[支付产品系统 26](#_Toc17206953)

[支付产品详情介绍 33](#_Toc17206954)

[银行卡支付 33](#_Toc17206955)

[快捷支付产品 38](#_Toc17206956)

[绑卡签约 45](#_Toc17206957)

[支付系统之应用内支付 47](#_Toc17206958)

[支付路由系统 47](#_Toc17206959)

[支付渠道系统 51](#_Toc17206960)

[支付清结算系统 54](#_Toc17206961)

[支付流程系统 59](#_Toc17206962)

[对账系统设计和实现 65](#_Toc17206963)

[渠道侧处理 70](#_Toc17206964)

[电商侧处理 76](#_Toc17206965)

[收单记账 80](#_Toc17206966)

[充值记账 89](#_Toc17206967)

[风控系统 89](#_Toc17206968)

[支付风控数据仓库建设 95](#_Toc17206969)

[支付风控模型和流程分析 102](#_Toc17206970)

[支付系统的基础设施建设 108](#_Toc17206971)

[支付基础设施 112](#_Toc17206972)

[支付系统的监控与报警 112](#_Toc17206973)

[系统重构与进阶 117](#_Toc17206974)

[三户模型互联网应用 117](#_Toc17206975)

[三类账户 118](#_Toc17206976)

[支付系统整体架构实例 118](#_Toc17206977)

[支付网关实例 118](#_Toc17206978)

[退款处理 122](#_Toc17206979)

[一、退款逻辑 122](#_Toc17206980)

[二、渠道侧处理 122](#_Toc17206981)

[三、关于手续费 126](#_Toc17206982)

[系统的问题与反思 126](#_Toc17206983)

## 支付系统

### 什么是支付系统

**支付总是伴随着交易。**

几百万年前，地球上诞生了人类。从原始部落到现代社会，无论人类文明如何进化，交易和货币始终伴随着人类社会的发展。

**以物易物**

假设我有很多苹果，但是我想吃橘子。刚好你有很多橘子，想换些苹果。那么我们可以做简单的交换，各取所需。

**商品货币**

市场使得产品具有了交易价值，是推动产品向商品转换的场所。在市场上， 产品转换为商品。

**信用货币**

货币出现后，改变了交易的流程。 从以贝壳等为代表的实物货币，发展到以金子为代表的商品货币，直到现在各国自己发行法定货币的信用货币阶段。

**银行**

银行的出现也是必然的，它是合理利用社会资源、归集利用闲散资金的中介服务机构。

因为银行安保齐全，我们把黄金存在银行。此时，银行会返给我们一个收据（纸币）来证明我们存入的黄金。交易收据（纸币）意味着交易黄金本身。

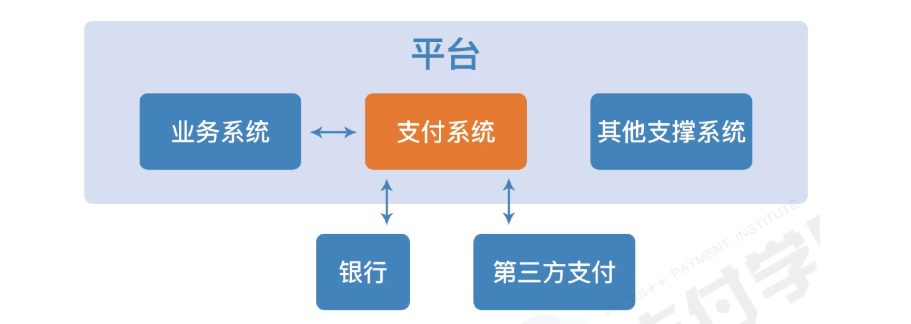
**记账**

自古以来，所有的商业活动都会产生货币的收款与付款行为。在人类漫长的历史长河中，记录收付款行为的方式不断迭代：古代的账房先生通过手工记账，工业社会通过收银机机械记账……

**支付系统**

支付系统伴随着电子商务的出现而出现，为各类电子商务经营活动实现在线收付款交易以及管理交易资金等功能，是具有一定独立性的内部系统模块。

## 支付系统模型



### 平台

开展电子商务经济活动的主体。

### 业务系统

实现平台用户注册、商品定价、营销活动等相关功能。

业务系统将用户购买行为通过各种交易订单的形式进行记录，并交付支付系统进行处理，最终由支付系统完成收款与付款。

### 支付系统

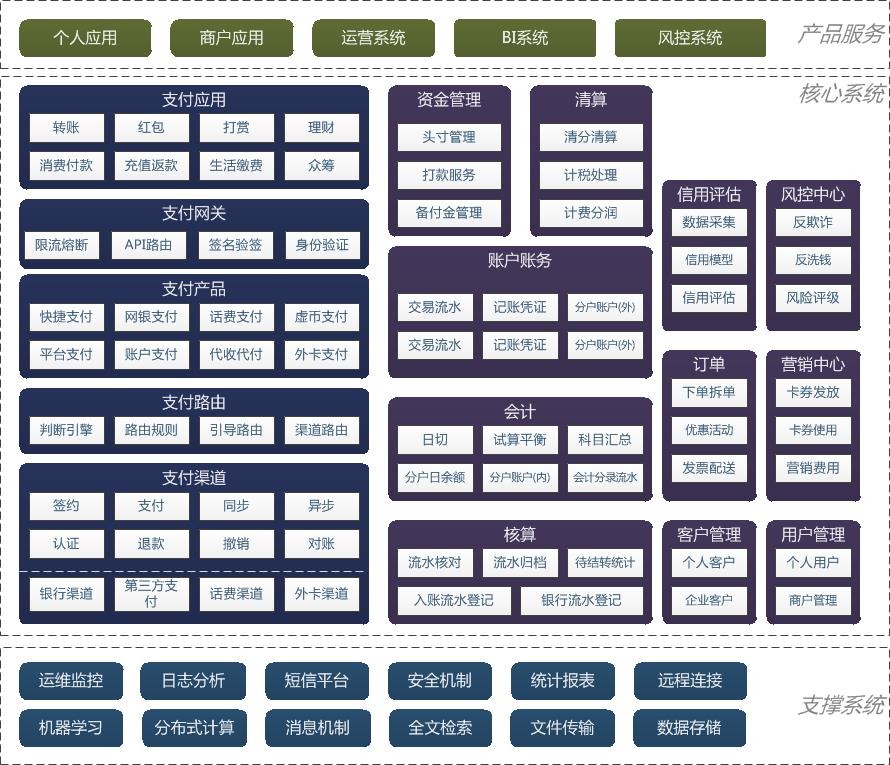
根据央行的现行规定，人民币交易处理仅限于银行及第三方持牌支付机构，因此支付系统在实现上述功能时，需要通过外部银行、第三方持牌支付机构完成交易资金处理。因此，支付系统需要具备：

统一封装处理的交易接口，以对接外部交易渠道，为业务系统实现交易订单处理的功能。

根据业务系统设置的资金分配规则，在一笔交易有多个收款方参与的情况下根据资金分配规则完成交易资金的自动化清分与结算，而后通过已对接的外部交易渠道完成划付。

账务数据记录功能，上述的交易、清分、结算形成的资金变动信息，需要支付系统通过账务数据记录功能加以记录，对交易资金进行统计并完成交易资金核对等财会工作。

## 支付系统架构



支付系统从架构上来说，分为三层;

### **支撑层**

用来支持核心系统的基础软件包和基础设施， 包括运维监控系统、日志分析系统等。

支撑系统是一个公司提供给支付系统运行的基础设施。 主要包括如下子系统：

#### 运维监控

支付系统在下运行过程中不可避免的会受到各种内部和外部的干扰，光纤被挖断、黑客攻击、数据库被误删、上线系统中有bug等等，运维人员必须在第一时间内对这些意外事件作出响应，又不能够一天24小时盯着。这就需要一个运维监控系统来协助完成。

#### 日志分析

日志是支付系统统计分析、运维监控的重要依据。公司需要提供基础设施来支持日志统一收集和分析。

#### 短信平台

短信在支付系统中有重要作用： 身份验证、安全登录、找回密码、以及报警监控，都需要短信的支持。

#### 安全机制

安全是支付的生命线。 SSL、证书系统、防刷接口等，都是支付的必要设施。

#### 统计报表

支付数据的可视化展示，是公司进行决策的基础。

#### 远程连接管理

#### 分布式计算

#### 消息机制

#### 全文检索

#### 文件传输

#### 数据存储

#### 机器学习

### 核心层

支付系统的核心模块，内部又分为两个部分： 支付核心模块以及支付服务模块。

#### 支付核心模块

支付核心系统指用户执行支付的核心流程，包括：

#### 支付服务模块

支持支付核心系统所提供的功能。服务系统又分为基础服务系统、资金系统、风控和信用系统。

**基础服务系统**提供支撑线上支付系统运行的基础业务功能：

##### 用户管理

包括对用户的注册登录等管理

##### 客户管理

包括对用户、商户的实名身份、基本信息、协议的管理；

##### 账户管理

管理账户信息、银行卡等信息

##### 卡券管理

 对优惠券、代金券、折扣券的制作、发放、使用流程的管理；

##### 支付通道管理

通道接口、配置参数、费用、限额以及QOS的管理；

##### 账务系统

管理账户信息以及交易流水、记账凭证等。这里的账务一般指对接线上系统的账务，采用单边账的记账方式。

内部账记录在会计核算系统中。

##### 订单系统

 一般订单系统可以独立于业务系统来实现的。这里的订单，主要指支付订单。

**资金系统**指围绕财务会计而产生的后台资金核实、调度和管理的系统，包括：

##### 会计核算

 提供会计科目、内部账务、试算平衡、日切、流水登记、核算和归档的功能。

##### 资金管理

管理公司在各个支付渠道的头寸，在余额不足时进行打款。 对第三方支付公司，还需要对备付金进行管理。

##### 清算分润

对于有分润需求的业务，还需要提供清分清算、对账处理和计费分润功能。

风控系统是支付系统必备的基础功能，所有的支付行为必须做风险评估并采取对应的措施；

##### 风控系统

信用系统是在风控基础上发展的高级功能，京东的白条，蚂蚁花呗等，都是成功的案例。

##### 信用系统

### 产品层

通过核心层提供的服务组合起来，对最终用户、商户、运营管理人员提供的系统。

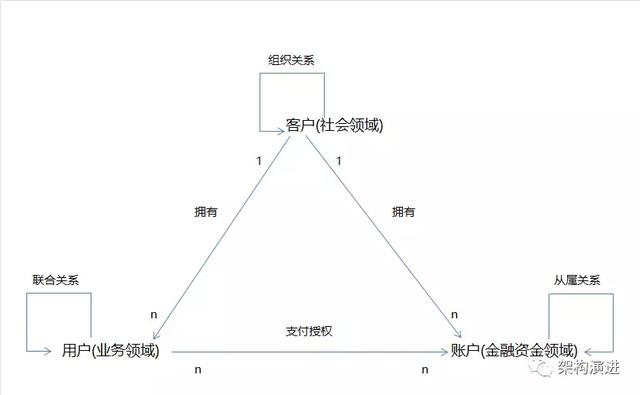
支撑系统、核心系统和服务系统，在每个公司的架构上应该是大同小异的，都是必不可少的模块。而支付应用是每个公司根据自己的业务来构建的，各不相同。 总的来说，可以按照使用对象分为针对最终用户的应用、针对商户的应用、针对运营人员的运营管理、BI和风控后台。

## 支付系统设计与实现

### 三户模型

#### 什么是三户模型

三户指**客户（Customer）**、**用户（User）**和**账户(Account)**。三户模型（客户，用户，账户）是一个成熟的经典用户模型，是根据”以客户为中心“理念产生的，客户需求成为支撑系统信息模型迭代升级的核心动力。



图示可见，三者之间的关联是一个归属和映射的关系，三个实体本身是相互独立的，分别体现完全不同的领域信息，客户体现社会领域的信息，用户体现业务领域的信息，账户体现金融资金领域的信息。

##### 客户

是指一个自然人或法人机构，团体，组织，公司企业等具有社会化属性的实体，客户的基本信息，如自然人的姓名，手机号，身份证，邮箱地址等，公司包括相关执照，经营行业，法人，公司地址等自然存在的属性信息。

##### 用户

是指客户签署了服务协议，订购开通系统业务产品服务时，系统为客户提供的一个身份实体，是一种逻辑上的概念。客户使用系统时对应的实体，信息包括系统中的登陆账号信息：用户名，密码，角色权限等。法人客户注册开通服务也可称为“商户”。

##### 账户

指客户用于存放资产，资金的实体，相对于银行里的实际账户，在系统当中指的是虚拟交易账户，用于存放虚拟货币，积分，甚至实际货币。主要涉及交易，记账，充值，提现等操作，判断实体是否是账户的一个标准是该实体是否具有独立会计核算的特性。

##### 关系

客户开通了一个业务系统服务，就产生一个用户，用户作为业务服务的承载主体。

一个客户可以有多个用户，既开通几个业务系统服务就有几个用户；同时一个用户只属于一个客户。

一个客户可以有多个账户，一个账户只属于一个客户。

一个账户可以为多个用户付费，同时一个用户也可以有多个账户为其付费。

如果一个客户为公司，那么它的用户就是公司用户，账户就是公司账户。

客户与客户之间存在群组关系，如公司和职员之间的所属关系；用户之间存在群组关系，如在业务系统当中的用户群组（如：普通用户群，VIP用户群，聊天群等）；账户之间的群组关系，如公司账户和职员个人账号（个人账号汇总到公司账号等）。

### 客户管理系统

指自然人或者法人。法人一般被称之为企业客户。如无特指，一般客户指个人客户。

#### 个人客户

在一个信息平台系统中，通过系统给客户分配的UUID来唯一标识客户信息，既UUID，一一对应自然人的身份证号，相同身份证号会识别为同一个客户。

#### 企业客户

企业客户本身是公司组织，在系统平台上有用户和账户归属于企业客户，但用户和账户都必须经由企业内部人员操作，此操作人本身为个人客户，其代表企业客户使用系统，也是系统个人用户；

企业客户可以授权给个人用户代表企业用户来操作企业账户，但实际上系统通常不做这样的授权处理，而是通过角色权限机制来实现。

同个人客户一样，企业客户在银行或支付平台开设资金账户，资金账户归属于此客户。企业客户是一个组织，其账户必然是组织授权内部人员去操作。但是这个操作人，同个人客户一样，只是系统的使用者，即用户。企业的资金比较大，并且有严格的业务流程，所以在系统使用上，一般是多个用户操作一个或多个资金账户。这种关系本身来说，也是一种授权关系，企业授权相应的用户来操作特定的资金账户，只不过为了管理方便，可以引入角色管理机制来实现。对于支付公司来说，企业客户通常都是发展商户过程中产生的。企业客户的识别同个人客户识别也是一样的，通过企业证件来统一识别。

建立企业客户的好处在于

有些企业本身只开通了企业服务业务，而不开通商户服务；

一个企业可以开通多个商户，企业客户是这些多个商户的统计口径

#### 客户模型



要从客户系统对提供的业务主题来分。比如：对于个人客户，客户识别就是一个服务主题，对于一个客户有多种多样的识别方式，除了证件外，还包括生物识别，比如画像、指纹等。

基本信息也是一个服务主题：包括客户的最基本的信息，姓名、年令、职业等;

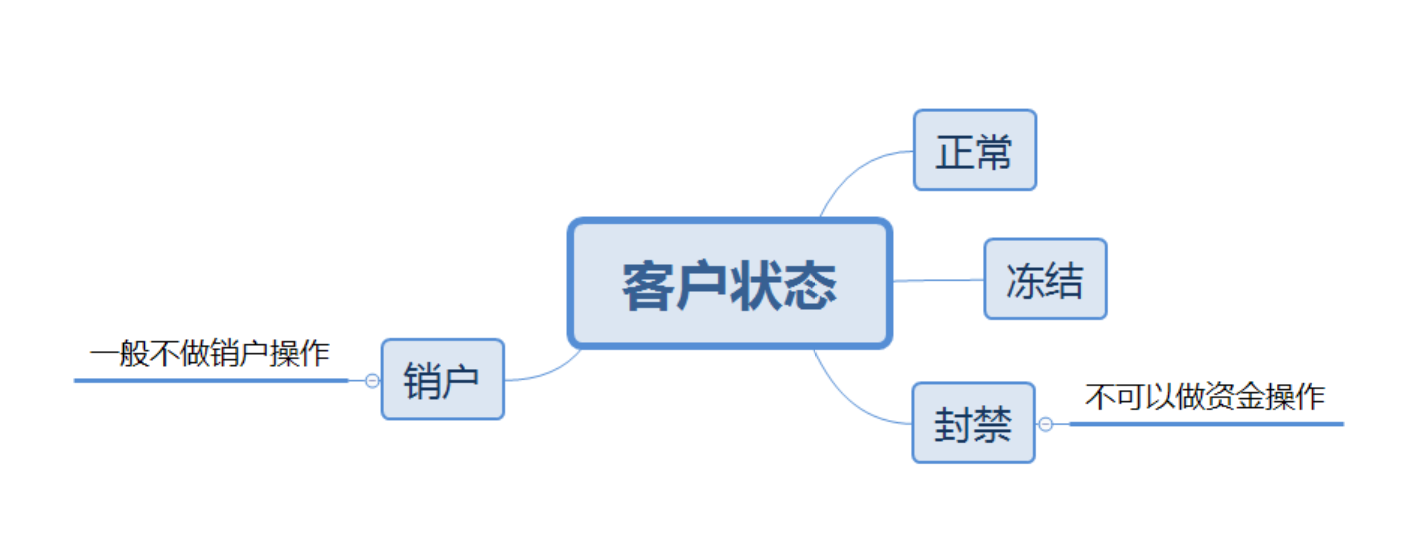
管理信息也是一个服务主题：比如这个客户的评级、是否集团员工等；

客户标签也是一个服务主题：所谓标签，就是从不同的维度来给客户做个标记，一个客户有多个标签，当然，前提是要对标签做好规划；

协议管理也是客户的一个服务主题：这里面有授权协议、委托协议等。

当然，客户还有其它的一些主题，这个要看公司的业务了。

#### 客户状态



客户信息自然存在，所以无需做“销户”处理，异常或恶意客户做“封禁”处理即可，同时为了在客户层面做到账号统一控制，增加“冻结”状态，“冻结”客户的账户时，账户无法操作账户资产。以司法协查为例，某个客户被协查以致账户冻结，需要冻结该客户下所有资金账户，这些账户都被止付。

#### 客户数据库设计

客户信息表

|  |  |
| --- | --- |
| 客户表(Customer) | |
| ID | 主键ID |
| name | 客户姓名 |
| type | 客户类别  1企业 2个人 |
| id\_type | 证件类别 1个人身份证 2 企业执照 |
| id\_number | 证件号码 |
| is\_real | 是否验证 |

个人客户表

|  |  |
| --- | --- |
| 个人客户表(Customer\_Person) | |
| ID | 主键ID |
| CID | 客户ID |
| name | 姓名 |
| phone | 手机号 |

企业客户表

|  |  |
| --- | --- |
| 企业客户表(Customer\_Corp) | |
| ID | 主键ID |
| CID | 企业客户ID |
| name | 企业名称 |

#### 客户接口设计

##### 客户验证

银行卡二要素:身份证号码+个人名称

银行卡三要素:身份证号码+个人名称+银行卡号码

银行卡四要素: 身份证号码+个人名称+银行卡号码+手机号

##### 客户归并

如果客户信息为同一个客户，则把客户信息合并

### 用户管理系统

#### 用户

通常互联网产品中，用户是通过简易注册方式创建的，这种情况下生成的用户是预开设状态的用户，需要进一步完善信息，才能成为正常用户；如果用户采用标准注册方式填写信息注册成为用户，则可以直接成为正常用户，可以使用业务服务。

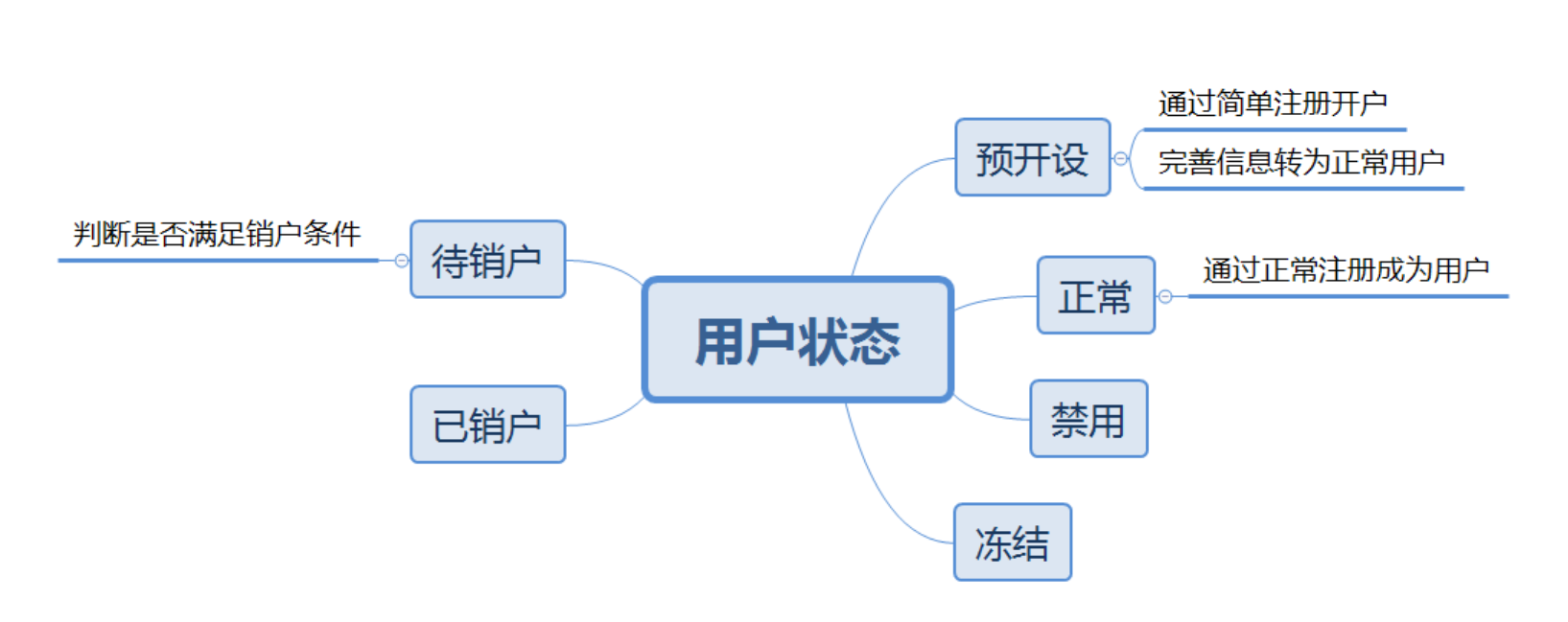
#### 商户

商户既指企业客户在系统中订购业务服务时，业务服务的逻辑对象主体，业务实际操作，是由商户指定授权的个人用户完成。被授权操作的资产账户实际归属于企业客户。

商户是企业客户的一个业务影子，或是看成资金账户分组的一个手段。商户是客户一个外围业务，如果把它看成用户平级层面也是可以的，即：此商户所有业务产生的资金进入到一个分类资金账户里。不论怎么说，一个企业不论开多少个商户，每个商户又开通多少个资金账户，都改变不了资金账户的归属关系，它是现实客户这个实体的。

#### 用户模型

#### 用户状态



通常互联网产品中，用户是通过简易注册方式创建的，这种情况下生成的用户是预开设状态的用户，需要进一步完善信息，才能成为正常用户；比如有些平台，在用户第一次登录时要求设置密保问题及答案。

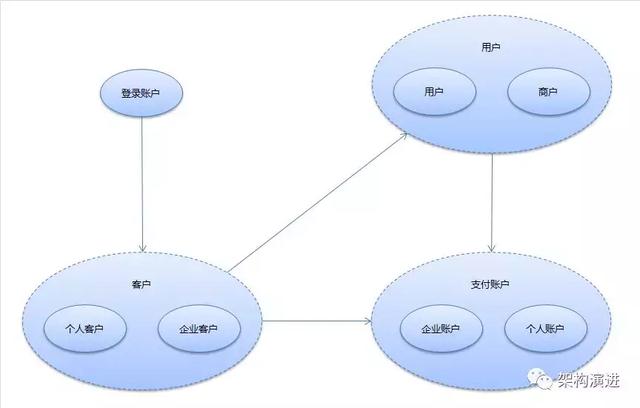
如果用户采用标准注册方式填写信息注册成为用户，则可以直接成为正常用户，可以使用业务服务。

当系统收到用户销户申请后，各个业务系统需要确认是否尚有未完成业务，以及该用户操作的各资金账户是否有有未支付，债务等，如果没有满足销户条件，则先注销资金账户，然后注销用户。

如果用户想要销户，收到销户申请后，不能直接销户，前面我们说过了，客户是通过用户来进行资金账户的管理与操作的，没有用户，资金账户就有可能没法玩了（假设客户只授权自己的一个而且只有一个用户来操作资金账户），特别是一些账户还存在债务。所以，此时有个确认过程，要求各业务系统确认此用户下的所有账户是否可以销户，如果没有问题，先销资金账户，当用户下的所有资金账户都销户完毕，再销用户，用户销户完成后，会释放出此用户占用的资源，如注册手机号。

　　用户除了生命周期状态外，还有一个管理状态，比如冻结，从现实模型中来说，这个是不应该放在用户层面的而是放在资金账户层面上的，但互联网模式下，一个用户有多个资金账户，为了用户体验，把这些放在了用户层面上了，就如同支付密码放在用户层面上一样。

### 账户管理系统



#### 账户

提及账户往往涉及到交易，账户是支撑交易的基本条件，交易过程的实现必须有账户的支持，通常的交易过程，是交易主体(发起交易的一方)的资金流出到交易对手(接收交易的一方)既完成一次交易，但互联网交易系统，因为没有内部清算的资质，所以资金的交易必须通过调用支付渠道的接口来完成，那么除了交易主体，交易对手两个账户之外，就还涉及到支付渠道账户。

由于电商系统本身并无清结算的资质，所有资金从交易主体到交易对手的账户的流动，在大部分情况下，并没有经过电商系统，而是由电商系统调用支付渠道提供的接口，由它来完成真正的支付过程。当然，渠道也不是活雷锋，在这个过程中，渠道要收取费用。所以，在电商系统中，一次交易会涉及到三个账户： 交易主体账户、交易对手账户以及支付渠道账户。

##### 登录账户

登录账户是用户在系统中的一个登录凭证和个人信息，跟支付账户是完全不同的两个概念

##### 支付账户

支付账户是指用户在支付系统当中用于交易的资金所有者权益凭证。

##### 关系

一个用户可以有多个登录账户，一个登录账户可以有多个支付账户，比如零钱账户，储值卡账户等。 一般来说，支付账户不会在多个登录账户之间共用。如果没有特殊说明，下文中的账户，都默认指支付账户。

##### 会计科目与账户

公司的会计需要对每一笔交易都要做详细的记录，即记账。公司每天都产生大量的交易行为，为了便于管理和统计，一个简单的方法是对交易进行分类，比如食品、带宽、办公用品等等。这个分类，按照公司的规模和业务复杂度，可以有一级、二级、三级或者更多级的结构，这被称之为**会计科目**。记账时，除了交易明细，还需要在每个级别上对交易额进行汇总。 一般来说，一级科目上汇总称为**总帐科目**，而详细记录称为**明细科目**。在电商系统中，由于涉及到的参与方较多，记账也相对复杂，但基本方法也是类似的。电商的参与者可以分为商户、买家和渠道，对这三类参与者，都需要分别建立总帐账户和明细账户。

##### 内部账户和外部账户

当用户使用银行卡来支付时，电商支付系统需要和银行对接，从用户银行卡所代表的账户上扣除资金。对接了银行，第三方支付等机构的电商支付系统，它需要连接到用户在这些机构的账户来执行扣款或者充值操作，这些账户或称为外部账户。对外部账户，支付系统只能记录账户在本系统的明细以及累计消费额，无法得知账户真正余额。 不少电商在玩零钱的概念，也就是让用户充值到零钱，使用的时候就直接从零钱中扣除。这就需要零钱账号。这是电商系统中自己设立的账号，所以也叫内部账号，可以知道账号的全部消费明细和余额。 当然，除了零钱账号，也可以有储值卡账号，信用账号等。那问题来了，什么时候需要建立账户，比如优惠券，需要账户吗？ 一次消费的储值卡和可以充值的储值卡，需要建立账户吗？这里先埋个雷，后续介绍支付和记账时，给出答案。

##### 收款账户和收单账户

当电商要对接银行时，往往都会被要求开设一个收款账户。用户通过这个银行来支付时，钱就被转到这个账户上。对第三方支付也是一样。收款账户是开设在银行或者第三方支付这边的，即渠道侧。一般来说，渠道每天都可以提供这个账户的交易流水供电商对账用。这样在电商这边，渠道就成为一个收单机构。所以在电商这边，建立这个收款账户对应的对账用的收单账号，用来记录通过这个渠道进行的各项交易流水。

#### 账户设计需求



在支付系统中，账户的设置，主要是从如下几个方面来考虑：

交易的需求，比如检查账户是否被锁定、余额是否足够、是否有效等。

记账的需求，按照公司会计需求记录账户上的所有行为，包括支出、充值、转账等。

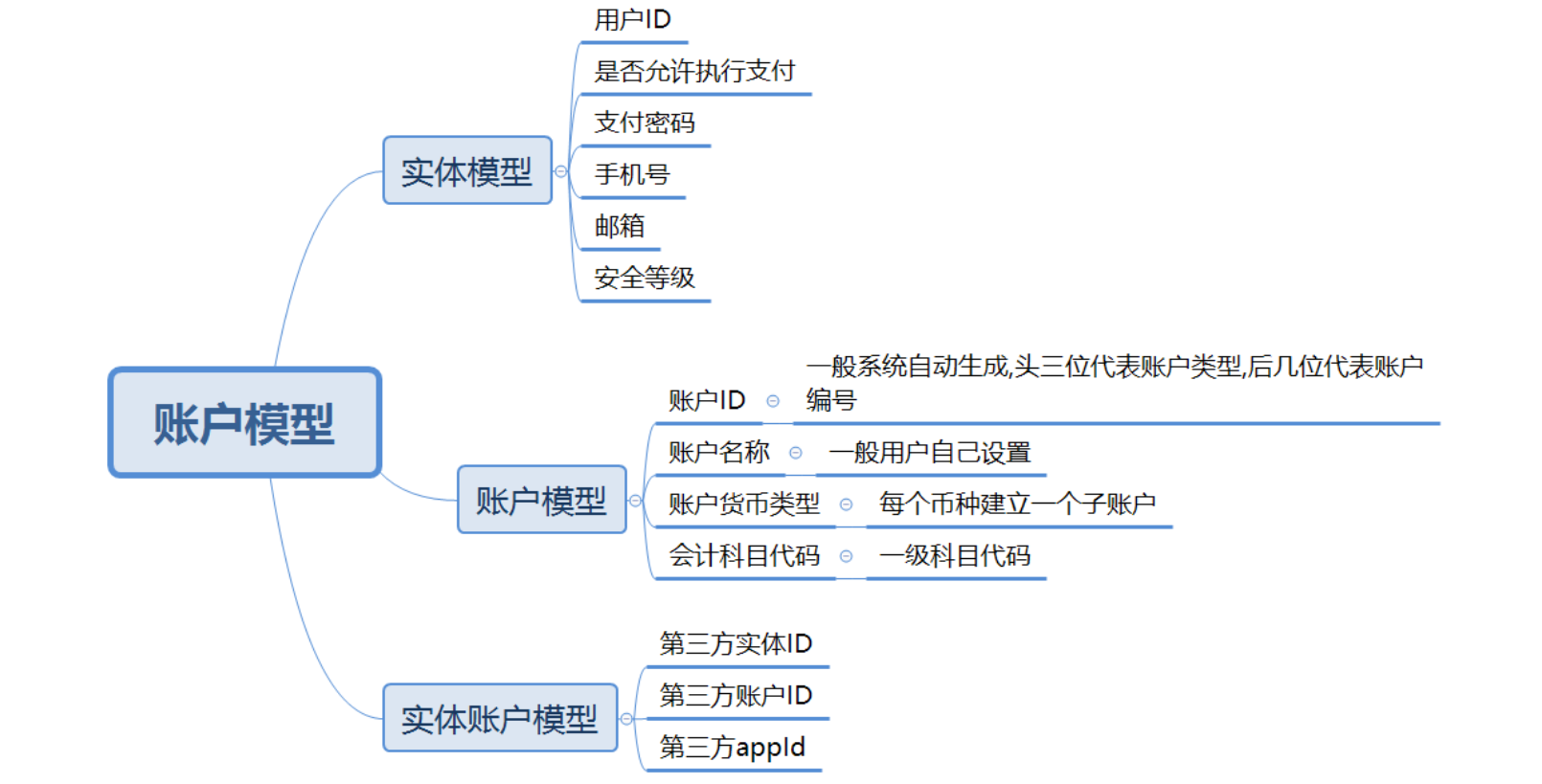
对账的需求，包括和支付渠道、商户、个人的对账需求，核对交易和账户余额是否正确。

风控的需求，如反洗钱、反欺诈等，都需要依赖于账户体系来提供核心数据。本文暂不分析这个内容，将在《支付风控》、《支付反洗钱》这两篇文章中详细分析

信用的需求，对用户、资产、商户等主体进行信用评估时，也需要依赖账户体系来提供的核心数据。本文也暂不分析这内容，将在《信用与支付》一文中分析。

这五个需求，按照其设计的优先级，也是从支付、记账、对账、风控来进行。 支付系统根据其发展所处的阶段，逐步将新增需求纳入设计中。

#### 账户建模



说了这么多，目的是为了对账户建模。 账户模型是和公司业务密切相关的，公司不同规模，发展的不同阶段需要不同的模型。 账户建模本身包括三大核心模型：实体模型、账户模型和交易模型。 从交易模型中可以衍生出针对各个角色的账户流水，即明细模型，用于支持对账。

##### 实体模型

实体模型和用户、商户模型有重叠的地方，这里专门针对支付而设置的各个实体属性。 一般来说，支付相关的实体模型需要包括如下的属性：

用户ID，一般直接映射到登录账户的ID；。

是否允许执行支付；

支付密码；

用于设置或者重置支付密码的手机号；

用户设置或者重置支付密码的邮箱；

用户的安全等级，根据业务需要来设置。

##### 账户模型

根据业务需要，可以设置多种账户，如支付账户、预付卡账户、代扣账户、零钱账户、结算账户等。 从类别上来说，这里的账户，一般指总账账户。一般来说电商系统中涉及的账户类型有：

虚拟币账号：用户和使用虚拟币的商户都需要建立虚拟币账户。

代扣账号： 用来支持订阅类型的定期代扣；

零钱账号：即电商的内部账号，用户、商户、清算单位需要建立零钱账户

第三方支付账号：用户在第三方支付机构建立的账户。

银行卡账号：用户的银行卡信息，每个卡对应一个账户。

结算账号：用来支持和第三方支付公司、银行进行结算用。 第三方支付需要为每个商户号建立结算账号；银行需要为借记卡、贷记卡分别建立结算账号（有必要吗？银行卡直连时使用）。

代扣代缴账户：用来支持代扣税款业务。

对这些账户，需要设置如下属性： 基本属性，包括：

账户号，或称为账户ID，一般是系统自动生成。特别注意的是，要事先约定好账户ID的规则。比如头三位用来表示账户类型，后几位用来表示账户编号等。务必保证根据账号号能够快速确定账户类型，并且保证账户号是不重复的。

账户名称，一般是由用户自己设置的，显示用。

账户使用的货币类型，注意虽然一张银行卡可以支持多个币种，实际在内部，还是针对每个币种建立独立的子账户。 涉及到多币种的账户，也可以采用类似的建模方案。

会计科目代码，一般是一级会计科目的代码。

账户控制相关：

是否允许充值；

是否允许提现；

是否允许透支；

是否允许支付；

是否允许转账进入；

是否允许转账转出；

是否有安全保障；

是否激活；

是否冻结；

资金相关：

当前账户余额：等于可用余额+冻结余额；

当前账户可用余额；

当前账户冻结的余额。冻结余额指在账户上暂不能使用的额度。在支付的时候，往往是先冻结，商品出库后， 再实际执行扣款。

银行卡、第三方支付信息：

第三方实体的ID；

第三方账号，如银行卡号或者在第三方支付的open\_id等；

第三方的app\_id；

账号的失效日期，该账号什么时候失效。

注意，有些第三方信息是不能保存的，如用户的账号密码、信用卡的CV号等。 为了避免账户信息被爬库或者数据库信息意外泄露，一般还需要对敏感字段，如密码等，进行加密保存，甚至保存到另外的表中。 更进一步，为了避免账户信息被意外修改，还可以增加一个校验字段，在写入数据时设置该字段，在读取数据时做校验，一旦发现数据有问题，则关闭该账号。

##### 交易模型

交易记录，交易流水，账户流水，交易台账，这三个容易混淆的概念，从数据上来说，却并不复杂，它们的核心是交易流水，账户流水是从账户视角的交易流水。那对一笔交易，涉及到的方方面面内容很多，有哪些需要记录的呢？考虑到交易记录将被用于风控和信用分析，能收集到的信息是越全面越好。

流水号：每一笔交易的流水号都不一样。需要根据业务情况详细设计流水号。这个号往往也是对交易表做分表分库的依据。

交易记录创建时间；

交易记录最后修改时间；

会计科目代码

关联的订单号，由商户提供；

订单名称、描述、关联的地址等信息；

费用信息，包括： 结算货币类型、原始费用、实际费用等；

交易主体信息，记录主体ID、类型、名字、账号、账号类型、使用的IP地址、手机号、平台、通知邮箱、当前位置等。 这些信息虽然可以从主体表中获取，但考虑主体表信息随时会被修改，所以这里需要记录详细的各原始信息。

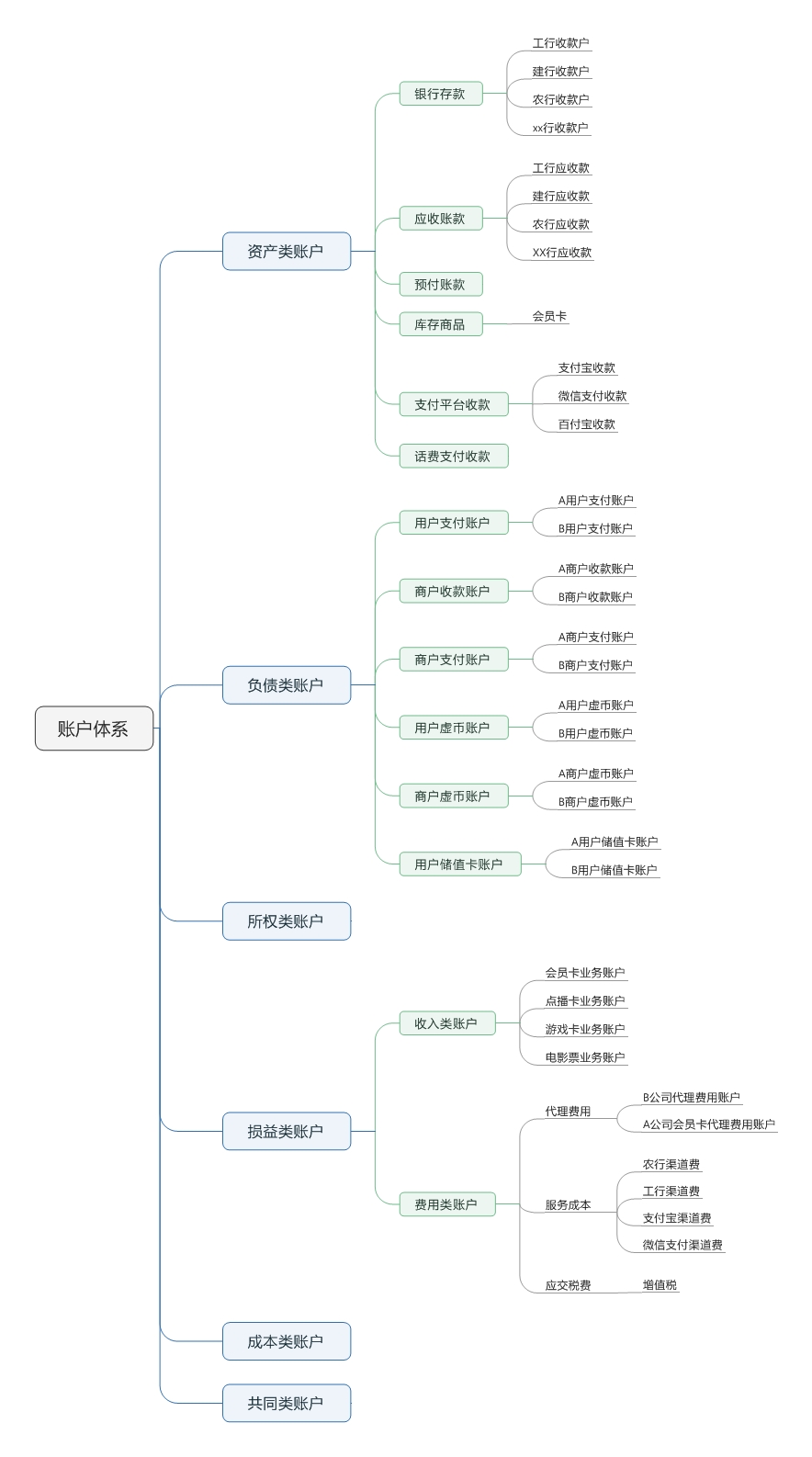
交易对手信息，记录对手主体的ID，类型，名字，账号，账号类型，手机号，平台，通知邮箱等。

交易渠道信息，记录所使用的交易渠道的实体id，渠道账户，渠道执行支付的时间、渠道侧返回的订单号等。如果有错误发生，还需要记录从渠道接收到的错误信息和错误码。

### 账务体系

#### 账户体系

在设计清结算系统前，首先需要完成账户体系的梳理。 账户是用来记录会计科目所反映的业务内容的工具，它根据会计科目来开设的。 账户有多种维度的分类。 按照经济内容来说，账户分为资产类账户、负债类账户、所有者权益类账户、损益类账户、成本类账户和共同类账户。 按照会计周期内期末是否有余额，也分为实账户和虚账户。



##### **资产类账户**

用来反映资产增加、减少以及增减变动结果的账户。和支付系统相关的主要资产类账户有： 银行存款、应收账款、预付账款、库存商品、发出商品等。 资产增加登记在借方，减少登记在贷方，期末有余额的话，一般出现在借方。 在一个会计期间，所有借方金额的累加为“借方本期发生额”，所有贷方金额的累加为“贷方本期发生额”。

资产账户的余额=借方期初余额+借方本期发生额-贷方本期发生额。

##### **负债类账户**

负债类账户也是实账户，记账规则跟资产类相反，负债增加记为贷，负债减少记为借，期末如有余额，一般在贷方，表明期末有债务实有额，负债类账户的余额计算：

贷方期末余额=贷方期初余额+贷方本期发生额-借方本期发生额。

##### **所有者权益类账户**

所有者权益类账户用来反映所有者权益增加、减少和变动结果的账户， 记账规则跟负债类账户一致：所有者权益增加记为贷，减少记为借。和支付系统有关的所权账户包括 本年利润、利润分配等账户。 企业取得的收入最终会使得所有者权益增加，因此收入类账户的记账方法跟所有者权益一致：增加记为贷，减少或者转销记为借，通常该账户期末无余额（因为期末收入都会转为所有者权益，如未分配利润等），属于虚账户。

##### **损益类账户**

损益类账户分为收入类和费用类账户。

收入类账户指各种收入、补贴、投资收益，如主营业务收入、其他业务收入和营业外收入等，增加记为贷，减少记为借。

企业在日常经营活动中会发生各种各样的耗费，这些耗费在会计学上称为成本费用，它们是收入的抵减项目，在抵销收入之前，可以视为一种资产，因此成本费用类账户的记账规则跟资产类一样：增加记为借，减少或者转销记为贷。费用类账户包括：主营业务成本、其他业务成本、营销费用等。

按照企业会计制度的规定，损益类账户的科目余额，应该结转入利润分配科目，期末余额为零，为虚账户。

##### **成本类账户**

有成本核算的企业需要设立的账户，包括生产成本、劳务成本等。

##### **共同类账户**

对于有支付牌照、可以执行清结算的支付平台上，共同类账户用于管理待清算的充值款项。

#### 账户结构

我们采用复式借贷记账法。对于分户账，或者说明细账，如下示例：

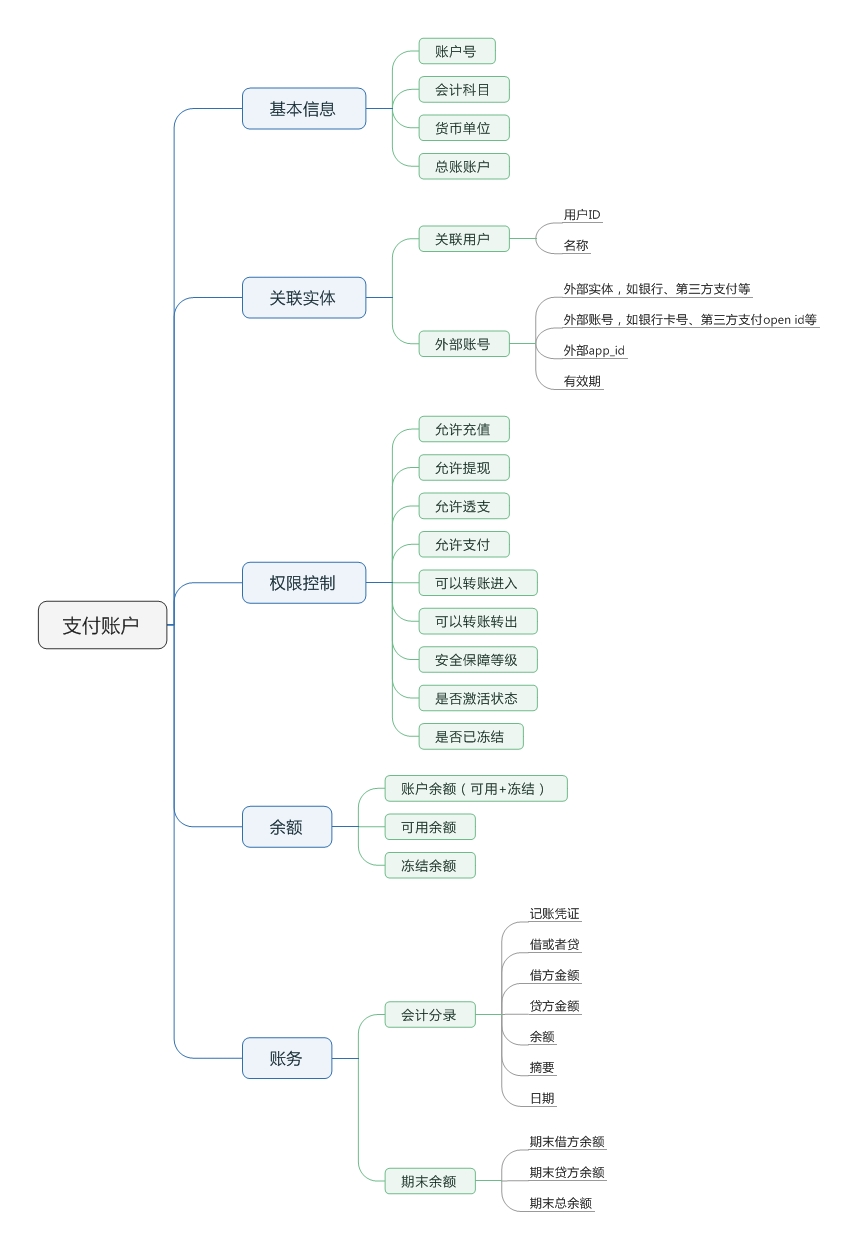


在这个实例中， 账户中账务相关的结构包括：

* 账户名称：如上述的“应收账款-工行收款”
* 会计分录： 除了登记借方金额、贷方金额，还需更新账户余额
* 期末借方余额、期末贷方余额、期末余额：按期定时计算。在日切时，计算日发生额和余额。在按月、季度和年作为会计周期时也采用类似的方法处理。 除了日切是必须的，其它时间段的处理是根据财务需要来实现。

在实现上，账户的各个属性更新时间并不一致，所以在设计账户表的时候，可以按照更新时机来划分表。

总的来说，账户的结构如下图所示，包括基本信息、关联实体、权限控制、余额和账务相关信息。



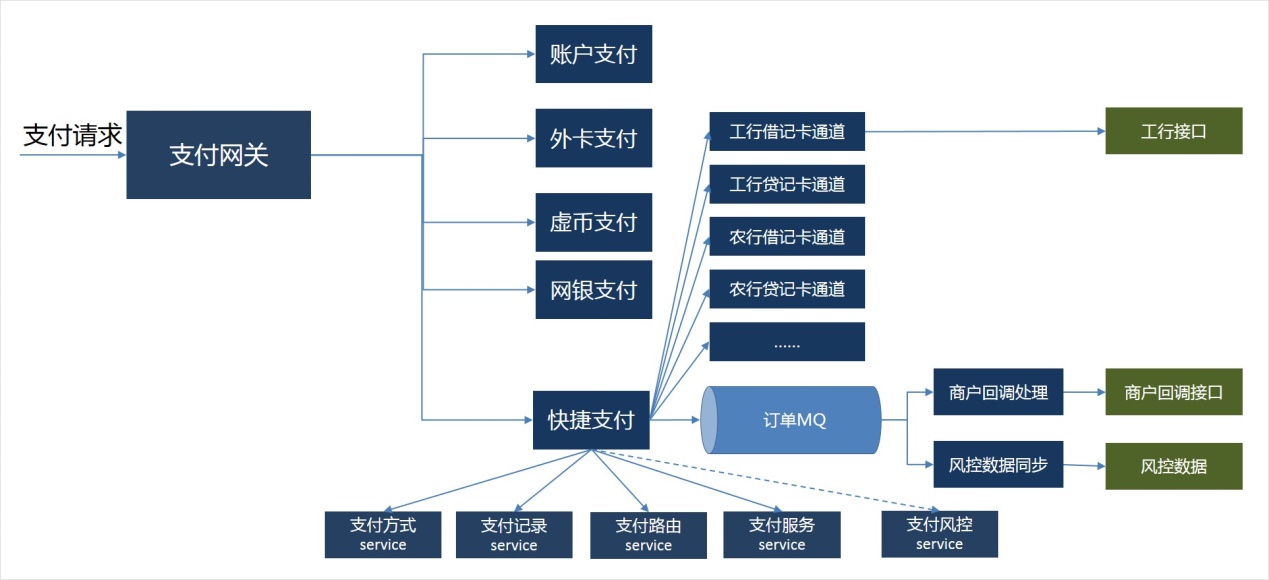
在存储上，账务相关信息一般是和账户其他信息相互独立处理，处理账务相关信息的子系统被称为账务子系统或者记账子系统。

### 支付网关系统

在支付系统中，支付网关和支付渠道的对接是最核心的功能。其中支付网关是对外提供服务的接口，所有需要渠道支持的资金操作都需要通过网关分发到对应的渠道模块上。一旦定型，后续就很少，也很难调整。而支付渠道模块是接收网关的请求，调用渠道接口执行真正的资金操作。每个渠道的接口，传输方式都不尽相同，所以在这里，支付网关相对于支付渠道模块的作用，类似设计模式中的wrapper，封装各个渠道的差异，对网关呈现统一的接口。而网关的功能是为业务提供通用接口，一些和渠道交互的公共操作，也会放置到网关中。

#### 功能概述

支付系统对其他系统，特别是交易系统，提供的支付服务包括签约，支付，退款，充值，转帐，解约等。有些地方还会额外提供签约并支付的接口，用于支持在支付过程中绑卡。 每个服务实现的流程也是基本类似，包括下单，取消订单，退单，查单等操作。每个操作实现，都包括参数校验，支付路由，生成订单，风险评估，调用渠道服务，更新订单和发送消息这7步，对于一些比较复杂的渠道服务，还会涉及到异步同通知处理的步骤。



商户侧应用发起支付请求。注意，这个请求一般是从服务器端发起的。比如用户在手机端提交“立即支付”按钮后，商户的服务器端会先生成订单，然后请求支付网关执行支付。

支付请求被发送到支付（API)网关上。网关对这个请求进行一些通用的处理，比如QPS控制、验签等，然后根据支付请求的场景（网银、快捷、外卡等），调用对应的支付产品。

支付产品对用户请求进行预处理，包括执行参数校验、根据支付路由寻找合适的支付通道、评估交易风险、生成订单、调用通道落地执行支付、响应通道的结果并将交易结果通知到商户侧。

支付产品调用支付通道执行支付。这个请求并不是直接落地到通道上，而是通过支付通道前置来封装，由支付通道前置来完成和通道的交付。 支付产品是按照可以提供的支付服务来设计的。

支付通道前置，（以下在不引起混淆的情况下，都简称支付通道）负责和支付通道之间的通讯，调用支付通道接口完成最终的支付操作。

不同类型的支付产品，其对外提供的接口也会有区别。

#### 业务流程

当接口被调用时， 首先执行参数校验，确认输入的参数的合法性，验证参数签名是否正确。确认过程包括调用账户、用户、支付方式、路由等服务来验证用户ID、账户、支付卡号、支付金额等参数。

根据输入的支付方式，调用支付路由服务，获取对应的支付渠道。

调用风控接口进行验证，如果有交易风险，则阻断本次交易。

生成交易记录；

调用支付渠道提供的服务执行支付。

根据支付结果，更新订单状态；

通知商户订单执行结果。

#### 设计原则

如上所述，支付网关、支付产品和支付渠道的职责分工为：

按照支付能力来划分支付产品。

同一支付能力的公共支付流程，在支付产品中实现。 支付产品提供的是和渠道无关的、和支付能力流程相关的功能。

在各支付产品中，其和支付能力无关的公共功能，在支付网关上实现。

按照这个分工，在支付网关上实现的主要功能：

API路由。在聚合支付场景下，当有多个支付产品可以提供支持时，使用支付网关可以让接入方对接时无需考虑支付产品的部署问题。

接口安全： 熔断、限流与隔离。 这对支付服务来说尤为重要。

如下功能，是在支付产品中提供：

风控拦截： 风控是和支付产品有关，不同产品的风控措施、处理对策也是不同的，所以风控是在产品层实现。

支付路由： 路由也是和产品有关。不同产品路由策略也不同。

参数校验： 这也是和支付产品相关的，不同的产品接口其参数也不同。

支付流程： 生成交易记录、落地渠道执行支付、同步和异步通知等操作。

如下功能，可以在产品层或者网关层实现：

身份验证： 确认付款方、收款方、渠道是否有执行当前操作的权限。 在那一层实现取决于这些信息是否有提炼为公共行为。

验签： 对接口参数进行签名并验证其签名。这是为了避免接口被盗刷和篡改的必要手段。如果对各个接口采用统一的签名规则，则可以在网关层实现。

#### 支付网关功能

##### 签名和验签

对接口进行签名是防止接口被盗刷的重要手段。大部分第三方支付和银行的接口签名规则类似。 query string格式参数可以参考[支付宝的签名过程](https://doc.open.alipay.com/docs/doc.htm?spm=a219a.7629140.0.0.SDgN7a&treeId=291&articleId=105971&docType=1), XML格式的可以参考[微信支付的签名过程](https://pay.weixin.qq.com/wiki/doc/api/jsapi.php?chapter=4_3)。其实两者都是类似的。 他们的签名和验签过程可以为支付系统服务器端和商户侧交互提供参考。

##### 加密算法

主流的加密算法有RSA、MD5和DES。支付宝使用RSA, 微信支付使用MD5。

使用RSA来签名，需要商户侧提供RSA的公钥给支付系统，将私钥自己保存。商户侧使用私钥来加密请求字符串，支付系统使用公钥来解密。

使用MD5来签名，需要商户侧和支付系统都保留MD5的Key。商户侧和支付系统都使用这个Key来加密请求字符串，验证结果是否一致。

##### 加密过程

将各个参数拼接成一个有序的字符串。 参数是key=value的格式， 按照key的字符顺序排序，以&或者其他符号来拼接。

appid**=**wxd930ea5d5a258f4f**&**body**=**test**&**device\_info**=**1000**&**mch\_id**=**10000100**&**nonce\_str**=**ibuaiVcKdpRxkhJA

使用RSA对字符串进行签名，生成签名字符串。

cYmuUnKi5QdBsoZEAbMXVMmRWjsuUj%2By48A2DvWAVVBuYkiBj13CFDHu2vZQvmOfkjE0YqCUQE04kqm9Xg3tIX8tPeIGIFtsIyp%2FM45w1ZsDOiduBbduGtRo1XRsvAyVAv2hCrBLLrDI5Vi7uZZ66Lo5J0PpUUWwyQGt0M4cj8g%3D

将签名字符串拼接到原请求中，生成最终的字符串。

appid=wxd930ea5d5a258f4f&body=test&device\_info=1000&mch\_id=10000100&nonce\_str=ibuaiVcKdpRxkhJA&sign=cYmuUnKi5QdBsoZEAbMXVMmRWjsuUj%2By48A2DvWAVVBuYkiBj13CFDHu2vZQvmOfkjE0YqCUQE04kqm9Xg3tIX8tPeIGIFtsIyp%2FM45w1ZsDOiduBbduGtRo1XRsvAyVAv2hCrBLLrDI5Vi7uZZ66Lo5J0PpUUWwyQGt0M4cj8g%3D

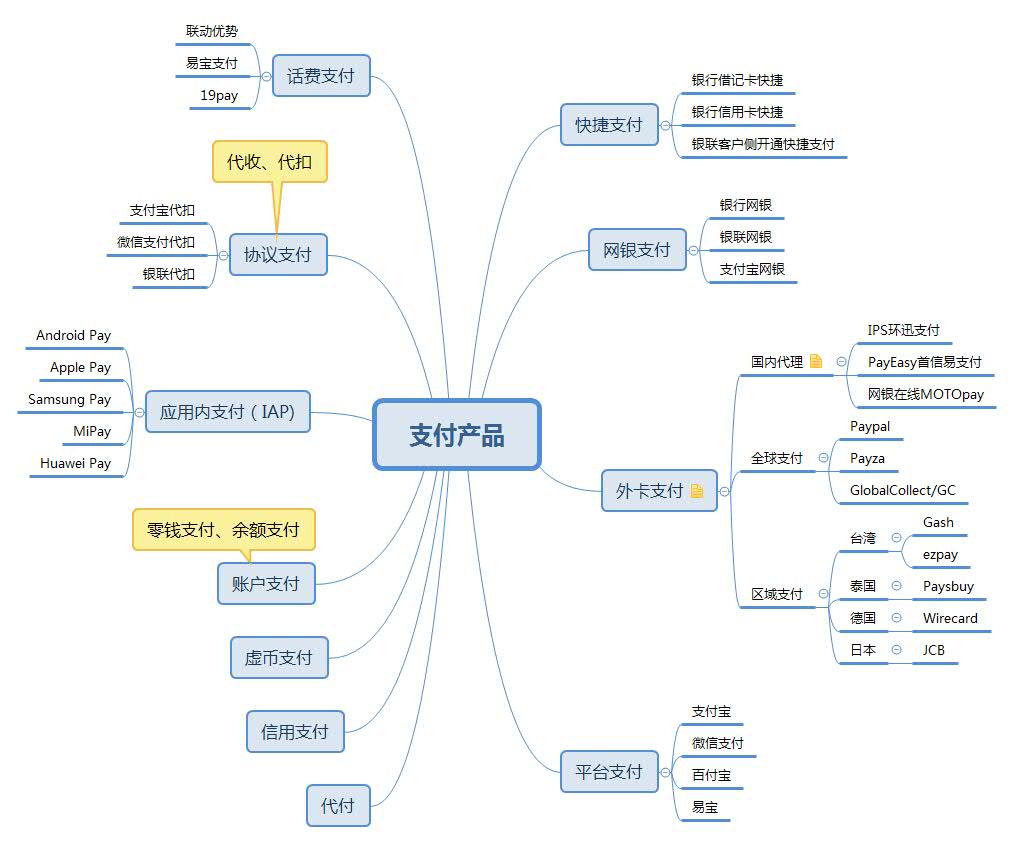
服务器端在接收到这个请求后，使用RSA的公钥来解密sign字段，如果解密成功，则对比解密结果和原始请求是否一致。 如果是使用MD5，则在商户侧和支付系统都使用这个过程来加密，检查最终的结果是否一致。

### 支付产品系统

支付产品模块是按照支付场景来为业务方提供支付服务。这个模块一般位于支付网关之后，支付渠道之前。 它根据支付能力将不同的支付渠道封装成统一的接口，通过支付网关来对外提供服务。所以，从微服务的角度，支付产品本身也是一个代理模式的微服务，它透过支付网关响应业务方请求， 进行一些统一处理后，分发到不同的支付渠道去执行，最后将执行结果做处理后，通过支付网关再回传给业务方。

#### 产品分类

在不同的公司由于接入渠道和应用的差异，对支付产品分类略有不同。综合支付场景和流程，支付产品可以分为如下几类：



支付产品是由支付系统对支付渠道进行封装而对业务方提供的支付能力。整体上来说，可以提供如下支付产品：

##### **快捷支付**

用户在完成绑卡之后，在支付的时候，不需要再输入卡或者身份信息，仅需要输入支付密码就可以完成支付。对于小额度的支付，甚至可以开通小额免密，直接完成支付。 这种支付方式不会打断用户的体验，是目前主要的在线支付方式。一般快捷支付产品是通过封装银行或者第三方支付平台提供的快捷支付接口或者代付接口来实现的。

##### **网银支付**

用户在支付的时候，需要跳转到银行网银页面来完成支付。在网银页面，需要输入用户的卡号和身份信息。这种支付方式会中断用户当前的体验，一般仅用于PC Web上的支付。 网银支付是封装银行提供的网银支付来实现。

##### **协议支付**

协议支付也称代收或者代扣，代收指渠道授权商户可以从用户的银行账户中扣款，一般用于定期扣款，不用于日常消费。比如水电煤气、有线电视费。协议支付是通过封装银行、第三方支付提供的代扣或者快捷接口来实现。

##### **平台支付**

使用微信、支付宝等第三方支付平台来完成支付。使用时，一般需要用户预先安装支付平台系统（手机上），注册并登录到第三方支付平台，并且已经在该平台上完成绑卡等操作。 由于微信、支付宝已经被大量使用，用户也产生对这些平台的信任，平台支付往往是电商公司的主要支付方式。

##### **外卡支付**

对于由海外支付的需求，还需要提供外卡支付支持。 国内不少支付渠道都能支持外卡支付，如支付宝全球购等。直接对接Paypal，也是目前用的最多的外卡支付渠道。 关于外卡支付，以后会有专文介绍。

##### **话费支付**

对于有包月小额类型的支付，手机话费也是一个不错的选择。目前也有一些平台可以支持话费支付，比如虹软、联动优势等。

##### **虚币支付**

不少公司会有自己的虚拟币，比如京豆、Q币等。这些虚币也可以作为一种支付方式。

##### **账户支付**

也成为余额支付、零钱支付等。 指为用户建立本地账户， 支持充值，之后可以使用这个账户来完成支付。

##### **信用支付**

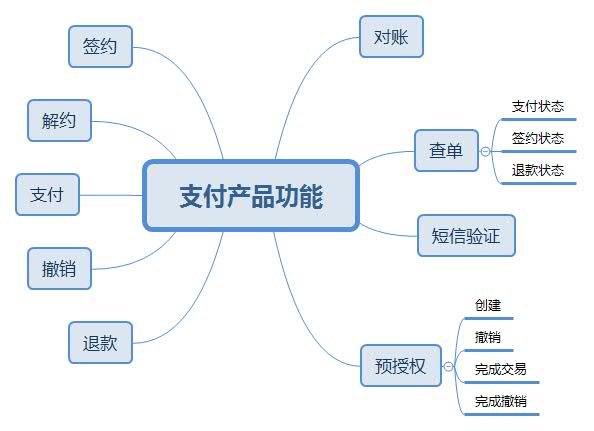
如京东的白条，蚂蚁花呗等，指使用信用账户进行透支，类似信用卡支付。

##### **代付**

和代扣相反，代付是平台将钱打给用户。

#### 模块功能

支出产品根据其支付能力，对外提供不同的功能。整体上来说，一般支付产品需要提供如下接口：



##### 签约和解约

在快捷支付、代扣等产品中，用户在使用前，需要先完成签约。签约可以在渠道侧进行，一般第三方支付采用这种方式，当电商需要接入时，让第三方给授权。 银行和银联的签约一般是在电商侧进行， 电商侧负责收集用户的信息，调用银行和银联的接口进行签约。签约后，后续的支付行为就使用签约号来进行，无需再输入个人信息。 和签约相对应，解约则是取消签约关系。

##### 支付

支付是少不了的操作。 不同产品中支付行为不一样。快捷支付是在电商服务器上发起，请求渠道进行支付；网银支付则是跳转到银行支付网关上进行; 而账户支付、虚币支付，则是在本地进行的。

##### 撤销和退款

有些渠道区分撤销和退款，比如银联、农行等，撤销指取消当天在渠道侧未结算的交易； 而退款仅针对已经结算的交易。有些渠道则不作区分。

##### 查询签约状态

对于需要签约的交易，可以通过这个接口来查询签约状态。

##### 查询订单状态

通过这个接口来查询支付清单状态以及退款的订单状态。

##### 预授权

预授权交易用于受理方向持卡人的发卡方确认交易许可。受理方将预估的消费金额作为预授权金额，发送给持卡人的发卡方。

##### 预授权撤销

对已成功的预授权交易，在结算前使用预授权撤销交易，通知发卡方取消付款承诺。预授权撤销交易必须是对原始预授权交易或追加预授权交易最终承兑金额的全额撤销。

##### 预授权完成交易

对已批准的预授权交易，用预授权完成做支付结算。

##### 预授权完成撤销

预授权完成撤销交易必须是对原始预授权完成交易的全额撤销。预授权完成撤销后的预授权仍然有效。

##### 对账

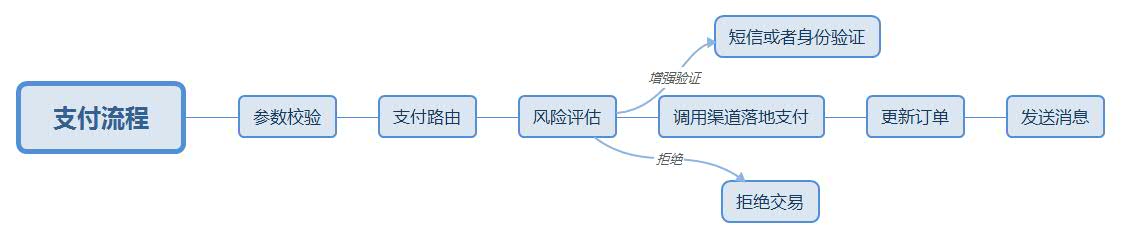
通过FTP或者HTTP方式提供对账文件供商户侧对账。

##### 余额查询

查询商户的交易账户的余额，避免由于余额不足导致交易失败。 注意，不是客户的余额。 当然，不是所有的银行或者第三方支付都提供这个接口。

#### 业务流程

上述操作，除了对账、查单外，每个操作实现的主流程，一般会包括参数校验，支付路由，生成订单，风险评估，调用渠道服务，更新订单和发送消息这7步，对于一些比较复杂的服务，还会涉及到异步同通知处理的步骤。



##### **执行参数校验**

所有的支付操作，都需要对输入执行参数校验，避免接口受到攻击。

验证输入参数中各字段的有效性验证，比如用户ID,商户ID,价格，返回地址等参数。

验证账户状态。交易主体、交易对手等账户的状态是处于可交易的状态。

验证订单：如果涉及到预单，还需要验证订单号的有效性，订单状态是未支付。为了避免用户缓存某个URL地址，还需要校验下单时间和支付时间是否超过预定的间隔。

验证签名。签名也是为了防止支付接口被伪造。 一般签名是使用分发给商户的key来对输入参数拼接成的字符串做MD5 Hash或者RSA加密，然后作为一个参数随其他参数一起提交到服务器端。如[支付网关设计](http://doc.cocolian.cn/essay/2017/03/15/account-10-gateway/)所介绍，签名验证也可以在网关中统一完成。

##### **根据支付路由寻找合适的支付服务**

根据用户选择的支付方式确定用来完成该操作的合适的支付渠道。用户指定的支付方式不一定是最终的执行支付的渠道。比如用户选择通过工行信用卡来执行支付，但是我们没有实现和工行的对接，而是可以通过第三方支付，比如支付宝、微信支付、易宝支付，或者银联来完成。那如何选择合适的支付渠道，就通过支付路由来实现。支付路由会综合考虑收费、渠道的可用性等因素来选择最优方案。

##### **评估交易风险**

检查本次交易是否有风险。风控接口返回三种结果：阻断交易、增强验证和放行交易。

阻断交易，说明该交易是高风险的，需要终止，不执行第5个步骤；

增强验证，说明该交易有一定的风险，需要确认下是不是用户本人在操作。这可以通过发送短信验证码或者其他可以验证用户身份的方式来做校验，验证通过后，可以继续执行该交易。

放行交易，即本次交易是安全的，可以继续往下走。

##### **生成交易订单**

将订单信息持久化到数据库中。当访问压力大的时候，数据库写入会成为一个瓶颈。

##### **调用支付渠道提供的服务**

所有的支付服务都需要第三方通道来完成执行。一般银行渠道的调用比较简单，可以直接返回结果。一些第三方支付，支付宝，微信支付等，会通过异步接口来告知支付结果。

##### **更新订单**

对于同步返回的结果，需要在主线程中更新订单的状态，标记是支付成功还是失败。对于异步返回的渠道，需要在异步程序中处理。

##### **发送消息**

通过消息来通知相关系统关于订单的变更。风控，信用BI等，都需要依赖这数据做准实时计算。

##### **异步通知**

如上述流程，其中涉及到调用远程接口，其延迟不可控。如果调用方一直阻塞等待，很容易超时。引入异步通知机制，可以让调用方在主线程中尽快返回，通过异步线程来得到支付结果。对于通过异步来获取支付结果的渠道接口，也需要对应的在异步通知中将结果返回给调用方。 异步通知需要调用方提供一个回调地址，一般以http或者https的方式。这就有技术风险，如果调用失败，还需要重试。而重试不能过于频繁，需要逐步拉大每一次重试的时间间隔。 在异步处理程序中，订单根据处理结果变更状态后，也要发消息通知相关系统。

### 支付产品详情介绍

### 银行卡支付

#### 银行卡支付

先说大家比较熟悉的银行卡支付，它分为线上支付和线下支付两种形式。线下支付就是通常说的POS收单，这里不介绍这个内容。对线上支付，按照卡的类别，分为贷记卡支付，也叫motopay、ePOS，即信用卡支付；和借记卡支付。按照支付形态，又分为认证支付、网银支付、快捷支付几种形态。银行卡网银支付要求银行卡必须开通在线支付功能，而快捷支付并不需要开通在线支付功能。主要利用支付验证要素（卡号、密码、手机号、CVN2、CVV2等），结合安全认证（例如短信验证码），让持卡人完成互联网支付。

#### 认证支付

指用户在绑卡时，将卡信息提供给电商。这样在支付时，用户无需再输入这些信息，由电商在服务器侧保留用户的账户信息，比如身份证号，卡号，手机号。在用户支付时，无需再输入这些内容，最多就提供个密码或者校验码，就可以完成支付。这基本不会打断用户的使用体验，所以也是电商喜欢的支付方式。但认证支付最让人诟病的就是安全性。一方面需要向电商暴露个人信息，一旦被窃取，资金就容易被盗走。还有在手机上执行支付，一旦手机丢失，窃取者就可以轻而易举的使用或者转移资金。

#### 快捷支付

快捷支付和认证支付类似，不同点在于绑卡之后，有些银行接口会返回token，后续使用token来作为支付凭证，无需提供卡号信息，这样电商也不需要本地保留卡号了。目前主要是银联有提供token接口。

#### 网银支付

相对来说，网银支付要安全很多。网银支付是由银联或者银行提供支付界面，用户必须在页面上输入卡号，密码等验证信息才可以执行支付。大部分银行还要求用户使用U盾或者其它安全硬件。但安全和易用永远是个矛盾。网银使用会打断用户体验，增加用户使用难度。对使用硬件加密的支付，不可能天天带着U盘跑。另外网银主要用在web端，在手机端，嵌入网银页面，还是比较难看的

#### 支付流程

走一个具体的例子看看吧。比如用户在电商系统中买了200块钱的东西，然后通过浦发银行卡做结算，用的是快捷支付。这个过程是：

用户在交易界面上，提交订单到交易系统中； 交易系统确认订单无误后，请求支付系统进行结算。这是在交易系统做的，后面工作就进入支付系统。

用户被引导到收银台页面， 让用户确认交易金额，选择支付方式，调用支付系统接口。

支付系统接收到支付请求，验证请求的各个字段是否有问题，确认无误后，调用支付网关执行支付。

支付网关请求浦发银行的快捷支付接口执行支付。

支付网关接收到支付结果报文后，对结果报文做解析，获取结果，并将结果告知交易系统。这可以通过URL或者RPC调用来实现。

商城系统收到支付结果后，开始执行后续操作。如果是支付成功，则开始准备出库。这一步在交易系统中处理，这里不做介绍。

网银支付，和快捷相比，就在第4步，插入一个步骤，将用户导航到网银页面输入支付信息，后续步骤是一样的。在资金流上也是相同的。 而在第五步获取返回结果上，一般银行就直接同步返回，银联是分为同步和异步返回。同步告知操作成功或者失败，异步告知扣款成功或者失败。同步操作和异步操作都需要调用方提供一个回调的URL地址，银联会将参数附加在这个地址上。通过解析这些参数可以得到执行结果。异步操作一般有2-3秒的延迟，取决于网络，以及该交易处理的复杂度。

#### 资金流

上一节说的是支付的信息流，那资金流应该是怎么走的？ 在第三步，会触发资金流。资金从用户个人账户上转移到电商公司的账户。当然，银行也不是活雷锋，这一笔交易是要收手续费的。资金是实时到账的，手续费一般是按月结算。有按交易笔数计费的，但大部分还是按照交易金额来收费。

同行快捷支付是比较简单的场景，让我们来逐步增加难度。如果支付系统没有对接浦发银行，那对浦发卡，就得走其它支付方式:银联或者第三方支付。

先说银联快捷。银联提供的多种接入方式，常说的快捷支付，在银联文档中叫商户侧开通token接口。通过这个接口，可以实现同行和跨行资金结算。不管收款行是浦发还是其它行，都可以完成结算。对本地和用户来说，体验是一样的。而在银联侧，后台资金流处理却不一样。了解这个资金流，有助于在异常情况下，了解资金到底跑到哪里了。

如果收款行也是浦发银行，银联发报文给浦发，浦发使用内部系统完成两个账户间的转帐，即时完成。

如果收款行是他行，比如工行。银联发指令给浦发和工行，分别完成各自账户上资金余额的增减，对个人和电商来说，这笔资金算是落地了。但实际资金流并不是立即发生。银联会在半夜做清结算后处理这笔资金。这个过程就是金融机构之间的清结算了，一般不需要关注。

如果使用的是第三方支付，对用户来说，处理的流程和银联一样。但资金流会不一样。 第三方支付在浦发和工行一般都会有落地的托管资金。 发生交易后，一般来说不会产生跨行资金流动。用户在浦发行的钱会被结算到第三方支付在浦发行的托管账户，而在工行的钱，会由第三方支付在工行的账户打到客户账户上。 这就降低了跨行资金流动成本。

目前国内主要银行都提供快捷和直联的接口。对电商来说，要对接哪些银行是个需要考虑的问题。

#### 银联Token支付

一般来说，大部分银行都提供直联和网银接口，但不需要直接对接所有银行。银联和第三方支付也提供直联接口，可以直接对接国内主要银行。也不是所有银行都被银联支持，这和银联签约的接口有关，需要在对接时咨询银联。从我们使用情况看， 浦发借记卡、邮储银行卡是不支持的。 另外 交行、平安（含原深发）、上海银行、浦发、北京银行，上述银行卡需通过 [这个地址](https://www.95516.com/portal/open/init.do?entry=open) 开通银联在线支付业务。

#### 对接银行

大部分银行提供的银行卡支付接口，借记卡支付和贷记卡支付是不一样的。但也有几个好心的银行，可以用一套接口同时开通借记卡和贷记卡。点名赞一下这些银行： 宇宙第一大行工商银行和建设银行。其他同学对接中如果也发现借记卡和贷记卡用一个接口的，也请及时告知。 作为国内最保守的软件团队，和银行对接时务必做好足够的准备。在商务谈判完成、拿到银行的接口文档后，需要考虑两个问题：专线问题、加密问题。

#### 卡bin

对接银行有一个逃避不了的问题，就是如何根据卡号判断发卡行？这就需要卡bin。 BIN号即银行标识代码的英文缩写。BIN由6位数字表示，出现在卡号的前6位，由国际标准化组织(ISO)分配给各从事跨行转接交换的银行卡组织。银行卡的卡号是标识发卡机构和持卡人信息的号码，由以下三部分组成：发卡行标识代码(BIN号)、发卡行自定义位、校验码。目前，国内的 银行卡 按照数字打头的不同分别归属于不同的银行卡组织，其中以BIN号“4”字打头的银行卡属于VISA卡组织，以“5”字打头的属于MASTERCARD卡组织，以“9”字和“62”、“60”打头的属于中国银联，而“62”、“60”打头的银联卡是符合国际标准的银联标准卡 ，可以在国外使用，这也是中国银联近几年来主要发行的银行卡片。 大部分银行卡号前6位即可确定发卡行和卡类型，但也有非标卡需要6-10位才可以判断出来。需要维护一个卡bin库。[附件](http://blog.lixf.cn/attach/card-bin.zip)是一个比较完整的卡bin库， csv格式的。

| **发卡机构** | **BIN 范围** | **卡号长度** | **验证方式** |
| --- | --- | --- | --- |
| 美国运通国际股份有限公司 （America Express） | 340000~349999 370000~379999 | 15位 | LUHN |
| 银联 | 620000~629999 | 16-19位 | LUHN |
| 大莱信用卡有限公司 （Diners Club） | 300000-305999 300000-305999 309000-309999 360000-369999 380000-399999 | 14位 | LUHN |
| 大莱信用卡有限公司 （美国加拿大） | 540000-559999 | 16位 | LUHN |
| 日本JCB | 352800-358999 | 16位 | LUHN |
| 万事达卡 （MasterCard） | 510000-559999 222100-272000 | 16位 | LUHN |
| 维萨卡 Visa | 400000 - 499999 | 13,16,19位 | LUHN |

上述LUHN验证方式，是一种验证银行卡号是否合法的算法，其具体步骤为：  
1.从卡号最后一位数字开始，逆向将奇数位(1、3、5…)相加。  
2.从卡号最后一位数字开始，逆向将偶数位数字先乘以2，如果乘积结果为两位数，则将其减去9，再求和。  
3.将奇数位总和加上偶数位总和，结果应该可以被10整除。

在加卡的时候先做LUHN，可以排除一部分输错卡号的情况。但要注意，国内有些银行的卡号不符合上述各种规定。

#### 短信和身份验证

一般绑卡操作第五步需要银行下发短信验证码。 短信验证的接口，不同银行还不一样。有些银行是短信和身份验证一起做了；有些银行是可以配置身份验证是否同时发短信。还有些比较奇葩的机构，比如某联，接口中让你传身份信息，但实际上没传也是可以的，也不验证身份信息到底对不对。这在对接渠道时需要特别注意。

此类接口一般包含如下内容：

版本号：当前接口的版本号；

编码方式： 默认都是UTF-8，指传输的内容的编码方式；

签名和签名方法： 生成报文的签名。 不是所有的字段都需要放到签名中，文档中会说明哪些字段需要签名；

签名算法：生成签名的算法，RSA, RSA128， MD5等。

商户代码：在渠道侧注册的商户号。

商户订单号：即发送给渠道的订单号；

发送时间：该请求送出的时间。

账号和账号类型： 银行卡、存折、IC卡等支持的账号类型以及对应的账号；

卡的加密信息：如信用卡的CVN2，有效期等。

开户行信息：开户行所在地以及名称；大部分是不需要的。

身份证件类型和身份证号： 可以用于实名验证的证件，指 身份证、军官证、护照、回乡证、台胞证、警官证、士兵证等。不同银行可以支持的证件类型不一样，这也不是问题。大部分就是身份证了。

姓名：真实姓名，必须和身份证一致；

手机号：在所在银行注册的手机号。

系统会返回上述数据的验证结果。如果验证通过，则会发短信。但这不是所有的渠道都是这样。哪些字段会参与验证、需不需要发短信，需要注意看接口文档。

#### 绑卡接口

绑卡接口和发短信接口类似，还需要将用户的卡号，身份证等信息传递过去。在绑卡成功后，会返回一个签约号。这个签约号是后续调用支付，解约等接口所必须的。 这里有个问题，已经绑卡的用户，调用绑卡签约接口再绑一次，会出现什么情况？这个和银行实现有关。 大部分银行，如农业、浦发、建行等，对绑卡签约接口调用，会首先验证身份信息，如果验证不通过，则不执行后续操作。验证通过后，再检查这个卡在该商户下是否已经绑过了， 如果没有绑过，则执行绑卡，否则会提示卡已经绑定过了，不能重复签约。 但工行的实现不一样，他是首先验证这个卡是不是已经绑过了，如果已经绑卡，则不继续验证身份信息。 总之，银行都不支持重复绑卡。

#### 银联绑卡

银联直联绑卡和银行绑卡类似，但是得注意验证接口，仅验证卡号和姓名，不验证身份证号和手机号。这导致第5步无法正常进行。银联只有到第六步执行绑卡时才做身份验证。 所以在处理上，还需要做一些调整，来确保和银行的流程的一致。 一种处理方法是，对银联，在第五步就开始调用银联接口执行绑卡操作，但是在本地标记为预绑卡状态；商户侧发送短信验证码，验证通过后，才将状态设置为绑卡成功。

银联网银绑卡处理起来比较麻烦。用户在电商页面上输入卡号，然后被导航到银联页面上去完成绑卡操作，成功后，银联返回一个token作为签约号，用于支持后续操作。这问题就来了，用户可以在银联页面上绑定一个别人的卡，而电商侧是无法知道这个卡的情况的。所以这种方式尽量不要用。

#### 实名认证

绑卡操作有个不错的副产品，就是实名认证。常说的二要素，三要素，四要素认证，可以通过这个操作完成。 二要素指姓名和身份证号，三要素加上银行卡号，四要素则加上手机号。看起来，似乎银行都应该支持四要素验证，但大部分银行接口仅支持三要素，毕竟手机号还是非常容易变。 当然，实名认证，也就是二要素认证，是应用最多的认证了。国内唯一的库是在公安部这，由NCIIC负责对外提供接口。可以提供如下功能：

**简项核查**：返回“一致”“不一致”“库中无此号”

**返照核查**：返回“一致+网纹照片”“不一致”“库中无此号”

**人像核查**：返回“同一人”“不同人”“库中无此号”

官方接口收费是 5元/条。 市面上主要的第三方服务提供商有国政通（简项、返照）、诺证通（简项）、IDface（三接口）等，收费简项核查：0.5~2.0元、返照核查为0.8~2.1元、 人像核查2.0~8.0元不等。一般都和访问量有关，量大从优。 当然，这里也要注意，涉密人员是没法查到相关信息的。 性能上， XX通一般在200ms内即可返回结果，普通商用应该是没问题的。 有些公司还会额外提供四要素接口，以XX通为例，它号称支持大部分银行卡的四要素认证。但是实现上有点儿懵，居然是实时请求银行的接口，这就导致接口延迟非常高，1秒以上的占大部分，甚至10秒以上的都不少见，基本无法商用。这种情况下，还不如直接上银联呢。

需要注意的问题：

验证短信由谁来发送。

对接的是总行还是分行接口；

是否支持重复绑卡，重复绑卡返回的Token是一样的还是不同的？

身份验证是三要素还是四要素验证

Token和账号是否相同。

### 快捷支付产品

快捷支付指用户在电商网站上执行支付时，不需要输入卡信息，仅根据短信或者其他的验证方式确认身份后，就可以执行扣款的支付方式。 这是目前电商网站采用的主要支付方式。 快捷支付分为首次支付和非首次支付，他们的流程是不一样的，区别就在于首次支付的时候，用户需要绑卡。而非首次支付则是直接使用已经绑定的卡来执行扣款，仅需做身份验证就行。注意，这里介绍的是电商网站为业务提供的快捷支付接口，而不是银行或者其他渠道提供给电商的快捷支付接口。

#### 一、支付场景

快捷支付第一步是要求用户做绑卡操作。绑卡是将用户的银行卡信息提供给电商，以后电商就用这个信息去银行完成支付。绑卡实际上是一个授权，让用户允许商家自动从他的账户上扣除资金。所以绑卡也叫签约，用户和银行，商家的三方签订的支付合约。 但我们知道，绑卡对用户和商户来说都存在巨大风险。如果说用户绑卡是图省事，那商户为什么要做这个事？快捷支付在支付场景上的优势在如下几个方面：

**1. 提升支付成功率**  
网银支付需要用户去银行开通网银，而对快捷支付来说，这并不是必须的。 这样使得可以使用快捷支付的用户群体得到扩展。 使用网银支付成功率在20%左右，银联直联成功率一般在50%左右，银行卡直联可以提升到70%左右。这是相当可观的数据。所以，当你看到绑卡送洗衣粉之类做法时，不需要担心商家会不会赔本。

**2. 提升支付安全性**  
网银是需要用户自己来保证支付环境的安全。在支付宝推出快捷支付和卡通支付之前，网银还是主要的支付方式，这导致各种的木马，钓鱼病毒肆虐，目标之一就为窃取用户的银行卡信息。银行发放U盘U盾，各种的证书，都是为了保证网银支付的安全，这也带来了各种支付的不便。 哪天忘了带了，就无法完成支付。而快捷支付是由商家来保证支付信息的安全，可靠性得到保障。当然，至于商家如何保证这些信息安全，还有待商榷。总的来说，这也比个人自己抵御黑客黑产要靠谱多了。

**3. 提升用户体验**  
想想网银支付的流程，用户在跳转到银行页面后，还得插入U盾，各种的核对。产品经理应该知道，每增加一个操作，都会带来一定的掉单率。 而快捷支付仅需在第一步完成绑卡后，以后的扣款，最多一个短信就搞定了，败家更方便。 在快捷支付中，绑卡和支付都不需要和渠道打交道，都是通过服务器来中转。而且在绑卡成功后，操作都在服务器上进行。这使得这种支付方式可以跨平台使用。手机，PC都可以。甚至在其他终端完成绑卡后，可以在电视，嵌入式设备上做支付。

**4. 获取可靠支付数据**  
快捷支付还有一个很好的副作用， 那就是得到真实用户信息，特别是身份证信息、银行卡号、手机号、验证方式、真实姓名等。再结合用户的订单信息，大家就可以知道为什么淘宝、京东等这些电商网站现在的商品推荐会这么精准了。

#### 二、绑卡流程

怎么绑卡？我们知道对接银行有两种途径，直接对接银行接口和通过银联来间接对接。这两种情况下绑卡处理也不同。 直观的，电商网站会在用户后台提供一个绑卡的入口，让用户直接绑卡。以支付宝绑卡流程为例，我们可以体验下:



这里有如下要点：

只能绑自己的卡，这主要从安全角度考虑。作为自己的卡，指银行卡在银行预留的姓名和身份证与网站预留的一致。

需要用户在银行侧预留的手机号进行短信验证。但不是所有银行都需要。这个时候，为了统一处理，可以考虑自己发验证短信。

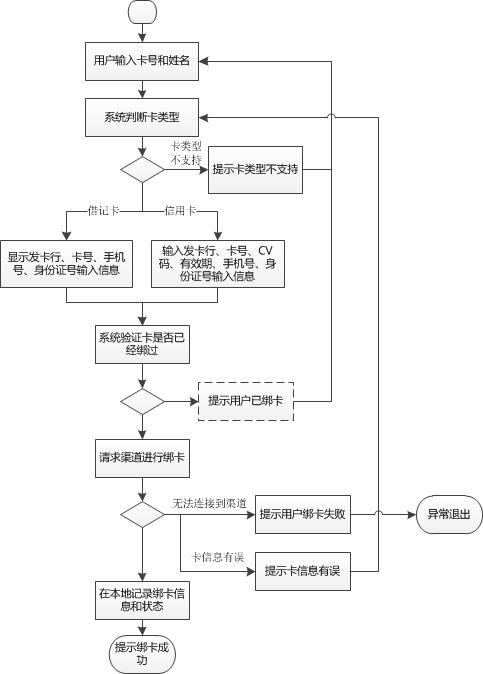
对这个入口不要指望太多，更多的用户是在支付中绑卡。也就是提交订单后，发现没有银行卡了，就开始绑卡。 和纯绑卡流程不同的是，最后一步，绑卡成功后，一般都同时完成支付。有些渠道会提供绑卡并支付的接口，减少交互次数。

为了保证卡的安全，绑卡有这些前置需求:

用户必须已经绑定了手机号。该手机号用于修改支付密码。

用户需设置了支付密码。支付密码不同于登录密码。

针对用户不同状态，绑卡流程上有区别。当然，绑卡是安全操作，要求用户必须登录到系统中。为了避免和服务器端的交互被劫持，所有操作必须在安全协议中进行，即使用https。

[](http://static.cocolian.cn/img/in-post/fast-bind.jpg)

**1. 输入卡号**  
用户输入卡号，系统对卡号执行初步验证。 验证的依据是卡bin和LUHN算法。参见[支付系统之银行卡支付](http://doc.cocolian.cn/essay/2016/10/12/account-3-bank/)。 当然，还有些系统会提供扫卡识码的功能，比如微信支付。 扫码识别的准确率可以达到99%，有些卡的卡号颜色和背景色一致的，就会识别出错。 如果用户没细看，进入下一步，就会报告错误了，这种错误还比较难发现。自动识别卡号，还需要考虑在识别错误时如何圆过去的问题。

**2. 获取卡信息**  
首次绑卡需要提供卡信息。借记卡需要卡号，用户真实姓名和身份证，这个所有银行都一样。(有不一样的，留言告知，谢谢) 信用卡就复杂点。大部分信用卡还需提供CV码和有效期。但是如果和银行关系好，拿到合适的接口，把这两个因素都免了，也是有可能的。

**3. 要素验证**  
首先在服务器端做验证。主要是验证卡是否已经被绑过。 如果一个用户有多个账户，系统还需要考虑是否支持这些账户都绑到一个卡上。 接着调用银行绑卡验证接口进行绑卡。这里有一个四要素验证的概念。由于国内要求实名制，所有银行卡都是实名办理的，所以银行可以验证姓名，身份证号，银行卡号和手机号是不是一致的。如果没问题，则会发短信到手机上。  
这里还有几个注意点:  
1.关于手机号。大家都知道，银行预留的手机号一般都是办卡的时候留的，过了几年，换手机了，很多人就忘了同步到银行。所以很多银行就不验证手机号。 2.关于验证短信，手机号都不是必须的，那短信就可能都不发了。这在流程设计时需要统一处理。银行不发短信就的自己发。  
3.重复绑卡问题。如果系统支持多账户，那不可避免的出现一个人绑卡到多个账号上。渠道侧绑卡，有接口支持重复绑卡，有些是不支持的。所以如果需要重复绑卡，还得在服务器端处理。

**4. 执行绑卡**  
用户输入短信验证码并确认绑卡，服务器端将用户实名信息以及短信验证码组合形成报文，发送给银行，执行签约操作。银行侧签约成功后，返回签约号给商户。 这一个处理逻辑放在支付渠道侧介绍。银行会返回如下结果：

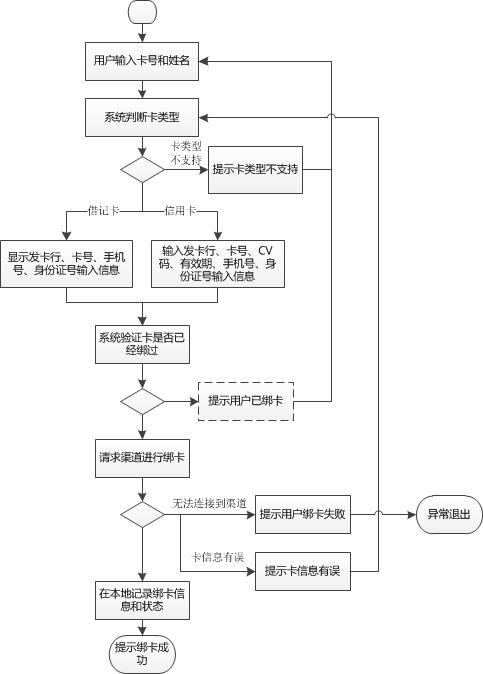
签约成功：这意味着可以建立签约关系。而签约关系在支付系统中则通过虚拟账户来表示。 具体的账户设计参见[账户模型](http://doc.cocolian.cn/essay/2016/10/08/account-1/)。

重复签约： 按照业务考虑是否支持重复签约。 一般针对一个银行卡仅保留一个签约关系，建立一个虚拟账户。

签约失败： 需要提示具体失败原因。

#### 三、扣款流程

在完成签约后，支付处理就相对简单不少。扣款流程如下：



流程要点：

订单系统生成订单后，请求支付系统执行支付；

支付服务器生成支付记录，请求渠道执行支付。如果该渠道需要短信支持，请求渠道发送短信。

服务器端在生成支付记录后，请求渠道执行支付。对于需要短信验证或者其他身份验证方式的支付行为，还需要首先请求渠道发送短信，之后让用户输入短信验证码。之后将订单、短信验证码、支付金额送到渠道侧，执行支付。

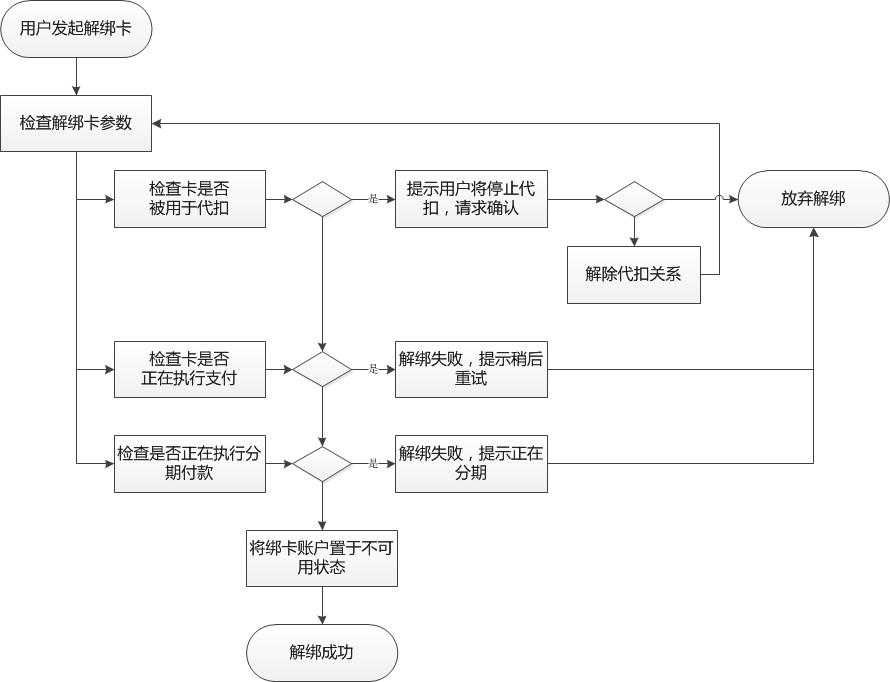
这里需要关注如下问题：

**1. 订单有效期**  
用户必须在订单有效期内完成支付。支付时，必须为每个订单设置有效期。这个有效期不能太长，一般不会超过一天。有效期可以从下单开始算，避免使用相对日期。 在执行支付时，需要验证下订单是否还在有效期内。

**2. 同步和异步处理**  
银联提供的商户侧开通快捷支付接口，不同步返回最终扣款结果，而是通过异步通知的方式来返回。 异步通知会多次调用回调接口， 直到调用成功。 所以，一个订单支付成功的通知，有可能会收到多次回调。

#### 四、解约流程

解约流程一般是由用户自己发起。当然，存储在本地的签约信息只是被设置为无效，而不是实际删除。 解约时，还需要注意相关的订单是否都已经完成。



#### 五、退款流程

没有平白无故的退款。支付都是由交易调起的，那退款就都对应的由退货发起的。 针对已经扣款成功的交易执行退款。如果交易还在进行中，比如渠道侧还没有返回结果，则执行退款会导致状态不可预知。

注意，退款的钱并不一定立即到账。不同银行、不同渠道到账时间不一样。

##### 六、接口概述

一般来说，快捷支付需要提供如下接口：

**签约**， 也叫“绑卡签约”、“开通交易”等，指用户在商户网站上开通快捷支付的功能，他需要将银行卡相关信息提供给电商。

**解约**， 也叫“解绑卡”， 指用户取消在该网站上的快捷支付功能。一般也会删除该用户在该网站上的相关的银行卡信息。

**扣款**， 也叫“支付”， 指用户使用签约的卡来执行一笔扣款。

**退款**， 针对已经扣款成功的交易执行退款操作，一般同时也会把用户权益或者对应的订单撤销。并不是所有订单都可以执行退款。

**查单**, 查询某次交易的处理状态。

**签约查询**， 即检查某个用户是否已经开通了签约功能。

##### 七、备份通道

由于快捷支付是常用的一个通道，其带来的问题是，如果这个通道出问题了，应该怎么办？回顾之前的文章，我们提到过，银行卡快捷支付不仅仅只有发卡行的通道，银联、第三方支付也都可以提供银行卡快捷支付接口。就发卡行来说，也不仅仅总行通道，很多发卡行的各省分行，由于历史因素，手头也有通道资源。 这也为我们实现通道切换提供了可能。 切换支付通道最大的问题是如何解决签约。 比如用户原来使用工行快捷支付接口来完成工行借记卡支付。某一天，工行通知接口要维护，不可用了，这就需要提前把工行接口切到其他备用通道，比如银联接口。 当然，直接切换是不行的，和银联还没签约呢。 那如何处理备用通道？ 关于备用通道的签约时间。可以在用户签约到主通道时，同时签约到备用通道，这种情况要求备用通道必须是无短信验证的，否则一次签约发两条短信，用户必须会怀疑的。还可以在第一次支付的时候签约，这时候将签约支付短信一并发出，避免影响用户体验。 关于通道选择，参考[支付路由设计](http://doc.cocolian.cn/essay/2017/02/06/account-9-services/)一文。

#### 八、接口实现

一般来说，快捷支付接口可以通过封装如下通道接口来实现：

当然，首先是银行自己提供的快捷支付接口；

银行的代扣接口；

银联的客户侧开通Token接口。

第三方支付的代扣接口。

#### 九、安全与合规

和银行卡对接，特别是快捷支付接口，有三个重要的合规文档，必须通读。

《非银行支付机构网络支付业务管理办法》(中国人民银行公告〔2015〕第43号公布）

中国人民银行关于进一步加强银行卡风险管理的通知, 银发〔2016〕170号

*严禁留存非本机构的支付敏感信息（包括银行卡磁道或芯片信息、卡片验证码、卡片有效期、银行卡密码、网络支付交易密码等），确有必要留存的应取得客户本人及账户管理机构的授权。*

*各商业银行、支付机构应在客户端软件与服务器、服务器与服务器之间进行通道加密和双向认证，对重要信息关键字段进行散列或加密存储，保障信息传输、存储、使用安全。*

*自2016年12月1日起，各商业银行、支付机构应使用支付标记化技术( Tokenization)，对银行卡卡号、卡片验证码、支付机构支付账户等信息进行脱敏处理，并通过设置支付标记的交易次数、交易金额、有效期、支付渠道等域控属性，从源头控制信息泄露和欺诈交易风险。*

*严格限制使用初始交易密码并提示客户及时修改，建立交易密码复杂度系统校验机制，避免交易密码过于简单（如“111111”、“123456”等）或与客户个人信息（如出生日期、证件号码、手机号码等）相似度过高。*

网络支付报文结构及要素技术规范(V1.0版), 银办发[2016]222号

### 绑卡签约

快捷支付指用户在电商网站上执行支付时，不需要输入卡信息，仅根据短信或者其他的验证方式确认身份后，就可以执行扣款的支付方式。 这是目前电商网站采用的主要支付方式。 在快捷支付中，用户首次支付，需要提供卡的信息，之后就可以凭短信验证码，甚至不需要密码，也可以执行扣款。

#### 接口概述

一般来说，快捷支付需要提供如下接口：

1. **签约**， 也叫“绑卡签约”、“开通交易”等，指用户在商户网站上开通快捷支付的功能，他需要将银行卡相关信息提供给电商。
2. **解约**， 也叫“解绑卡”， 指用户取消在该网站上的快捷支付功能。一般也会删除该用户在该网站上的相关的银行卡信息。
3. **扣款**， 也叫“支付”， 指用户使用签约的卡来执行一笔扣款。
4. **退款**， 针对已经扣款成功的交易执行退款操作，一般同时也会把用户权益或者对应的订单撤销。并不是所有订单都可以执行退款。
5. **查单**, 查询某次交易的处理状态。
6. **签约查询**， 即检查某个用户是否已经开通了签约功能。

#### 绑卡签约

为什么要求用户绑卡？这和快捷支付有关。参见上一篇文章的分析，绑卡是将用户卡信息提供给电商，以后电商就用这个信息去银行完成支付。绑卡实际上是一个授权，让用户允许商家自动从他的账户上扣除资金。所以绑卡也叫签约，用户和银行，商家的三方签订的支付合约。

但我们知道，绑卡对用户和商户来说都存在巨大风险。如果说用户绑卡是图省事，那商户为什么要做这个事？首先当然是提升用户体验了，让用户花钱更容易。其次，提升支付成功率。使用网银支付成功率在20%左右，银联直联成功率一般在50%左右，银行卡直联可以提升到70%左右。这是相当可观的数据。所以，当你看到绑卡送洗衣粉之类做法时，不需要担心商家会不会赔本。

怎么绑卡？我们知道对接银行有两种途径，直接对接银行接口和通过银联来间接对接。这两种情况下绑卡处理也不同。 直观的，电商网站会在用户后台提供一个绑卡的入口，让用户直接绑卡。以支付宝绑卡流程为例，我们可以体验下:

这里有如下要点：

* 只能绑自己的卡，这主要从安全角度考虑。
* 需要用户在银行侧预留的手机号进行短信验证。但不是所有银行都需要。这个时候，为了统一处理，可以考虑自己发验证短信。

对这个入口不要指望太多，更多的用户是在支付中绑卡。也就是提交订单后，发现没有银行卡了，就开始绑卡。 和纯绑卡流程不同的是，最后一步，绑卡成功后，一般都同时完成支付。有些渠道会提供绑卡并支付的接口，减少交互次数。

#### 绑卡流程

先介绍比较简单的银联直联绑卡。为了保证卡的安全，绑卡有这些前置需求:

1. 用户必须已经绑定了手机号。该手机号用于修改支付密码。
2. 用户需设置了支付密码。支付密码不同于登录密码。

针对用户不同状态，绑卡流程上有区别。当然，绑卡是安全操作，要求用户必须登录到系统中。为了避免和服务器端的交互被劫持，所有操作必须在安全链接中进行，即使用https。当用户开始绑卡时，执行如下流程：

1. 检查用户是否有手机号。没有则进入设置手机号流程。  
   2.检查用户是否设置支付密码。如果已经设置，则需要用户输入密码。确认后开始绑卡。否则，也是先进去绑卡后设置密码。  
   3.用户输入卡号，系统根据卡号判断卡的发卡行，并显示给用户。有些实现，如微信支付，会提供扫卡识码功能。  
   4.用户输入银行预留手机。对于没有绑过卡的用户，需要用户提供真实姓名和身份证号。对于信用卡，还需要输入cv码和有效期。这一步，卡的信息都收集全了。  
   5.调用银行绑卡验证接口进行绑卡。这里有一个四要素验证的概念。由于国内要求实名制，所有银行卡都是实名办理的，所以银行可以验证姓名，身份证号，银行卡号和手机号是不是一致的，如果没问题，则会发短信到手机上。  
   6.用户输入短信验证码并确认绑卡，服务器端将用户实名信息以及短信验证码组合形成报文，发送给银行，执行签约操作。银行侧签约成功后，返回签约号给商户。

### 支付系统之应用内支付

应用内支付指使用手机操作系统自带的支付功能来支持支付。目前国内主要的应用内支付有 Google Pay， Apple Pay， 小米支付、华为支付等。 其中Apple Pay是典型的一个应用内支付，Android平台的各种支付也一般是沿用Apple Pay的设计。 相对来说，应用内支付的用户体验，和微信支付、支付宝相比，还是有一定差距的，但是为什么要开发应用内支付呢？ 这个和苹果的AppStore的审核政策有关。

### 支付路由系统

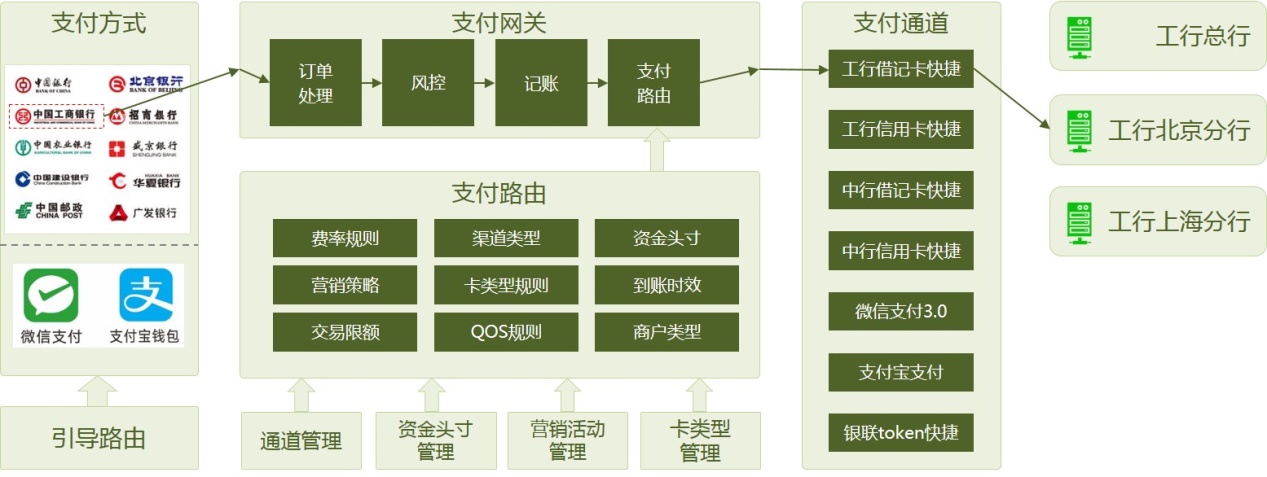
用户在前端选择一种支付方式，比如使用招行借记卡来支付后，系统不一定就是调用招行的接口来执行支付。支付宝、百付宝等第三方支付平台以及银联等，都支持招行借记卡支付。 这种将支付方式落地到具体的支付接口的模块，就是支付路由。

#### 设计目标

支付路由在支付系统中的核心作用，除了本职工作路由外，还承担如下职责：

1. 省钱，省钱，省钱，这是支付路由选择支付通道的最主要的规则。 哪个通道省钱，基本会优先考虑这个通道。
2. 提升支付产品的QOS。这体现在系统的可靠性、稳定性、性能和可用性上。通过屏蔽掉无法连接、不稳定、性能低的通道来提升这些指标。
3. 支持营销。通过优先选择有优惠活动的通道，可以帮助业务提升付费客户量。
4. 降低运营成本。一个设计良好的支付路由，可以大大降低运营投入。

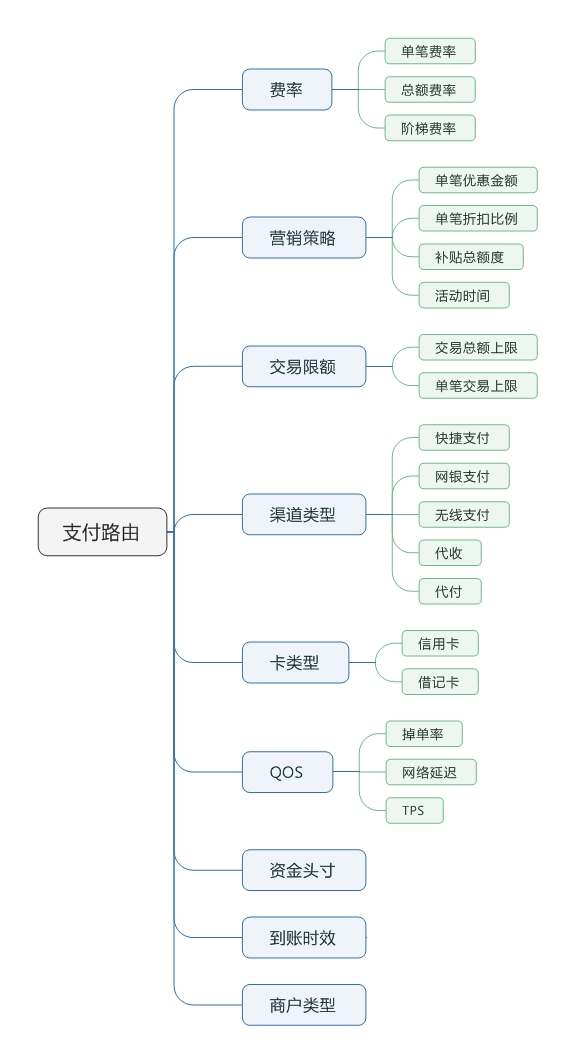
#### 软件架构



支付路由并不会直接对接前端的支付产品或者后端的支付渠道，它是[支付网关](http://doc.cocolian.cn/essay/2016/11/02/account-7-gateway/)的一部分。如果是基于微服务的架构，支付路由作为一个独立的服务，被支付网关所调用。

#### 计算因子

路由规则是支付路由的核心。在规则设置上，需要和公司的业务、支付服务的scope来综合考虑。 这里讲述的是通用的规则设计，供具体实现时参考。



**产品类型** 当然，路由时首选需要考虑渠道可以支持的支付产品。

**费率**  
费率是路由最重要的一个指标。一般银行是按照额度来收费，部分是按照交易笔数来收费，复杂点的是阶梯收费，比如10万一个费率档次，100万一个费率档次。

**优惠活动**  
银行、第三方支付为了延揽客户，经常也会提供一些补贴给对接的商户，对于使用该渠道的交易进行补贴。而优惠的策略也是多种多样。

**交易限额** 不同通道会限制每次交易的金额上限，以及针对每个账户每天的额度限制。超过这个额度，需要变换通道。

**卡类型** 通道支持的信用卡或者借记卡。

**通道的QOS** 掉单率、网络延迟以及接口能支持的TPS，是路由的一个重要衡量因素。

**资金头寸** 电商公司在银行或者第三方平台的资金头寸。如果资金头寸不足，则不能使用这个通道来执行。

**到账时效** 对于转账，资金什么时候到目标账户上，也是影响路由选择的一个因素。

**商户类型** 不同类型的商户可以选择不同的通道。 比如高性能、费率高的通道，预留给高级商户。

#### 模块设计

支付路由从架构上来说，一般是作为支付网关的一部分。采用微服务架构时，路由模块作为一个独立的服务来部署，为支付网关所调用。所涉及的模块如上所示:

**支付通道管理**

提供通道支持的产品类型、费率等信息。

**支付通道QOS监控**

收集通道使用过程中的错误信息，接口延迟，超时情况等信息，用于统计QOS。

**资金头寸管理**

用于监控公司在各个支付通道上的头寸，并提供头寸的信息。

**优惠活动**

银行、第三方支付为了延揽客户，经常也会提供一些补贴给对接的商户，对于使用该渠道的交易进行补贴。而优惠的策略也是多种多样：

1. 支付策略；针对使用该通道的所有支付进行补贴；仅针对首次使用该通道的用户进行补贴；仅针对绑卡的用户进行补贴。
2. 补贴时，按照支付金额来设置优惠额度，或者按比例打折。
3. 一般活动都会设置补贴总额度。该额度用完了就停止补贴。   
   当然，活动也都会设置开始和截止时间。

#### 路由计算

**人工路由**

大部分支付系统在接入渠道不多时，人工路由也是一个不错的选择。运营人员指定支付渠道和产品之间的映射关系。出问题时人工切换即可。这种路由的优势是性能高，路由结果可控，出问题时易于排查问题。当接入通道数量增加，营销活动频繁时，人工路由会是一个巨大的投入。

**基于规则的路由**

这是相对比较简单的自动路由设计。 按照业务要求设置各种路由规则，比如：

if(支付方式==招商借记卡 && 额度<100) then 目标通道==银联token支付

if(支付方式==招商借记卡 && 额度>100) then 目标通道==招商快捷支付

技术上，规则可以使用drools来描述。

**基于权重的路由**

规则路由的难点各种规则的制定。在路由因子增多的情况下，规则的维护会是一个噩梦。基于权重的路由则可以缓解这个问题。这种计算方式，简单说，就是对各个通道打分，分数最高的就命中。难点在于制定打分的标准以及计算公式。 比如可以从费率、优惠额度、QOS和使用率角度来评分，给优惠额度高一点的比重，这会导致高优惠额度的通道被优先命中。注意每个维度上的计算公式也不是一成不变的，比如使用率和QOS都是动态打分计算。权重的调整是一个挑战，需要在运行过程中不断的调试。必要时，可以使用旁路测试来比较两种算法的优劣。

路由是支付的核心模块，稳定性是第一要素，其次是性能，最后才是怎么省钱。路由系统的设计，需要和公司业务发展保持一致，并适度超前。简单的if-else实现可以满足大多数场景下的需求。避免在系统建设初期引入过于复杂的路由。

### 支付渠道系统

#### 什么是支付渠道

支付渠道，顾名思义就是平台上支持用户支付的通道，这些支付渠道帮助平台用户完成交易金额的支付，并且支持平台与银行之间进行资金流转、对账和清分，比如微信、支付宝、通联、易宝等。一般交易平台都会对接多家支付渠道公司。

#### ****主流的支付渠道有哪些****



#### 为什么对接支付渠道

##### ****提供扣款的途径****

交易平台对接支付渠道可以为平台用户支付提供一个途径。用户可以利用该支付通道在平台上选定商品或者服务后发起支付，通过支付渠道完成其个人银行卡账户余额的扣款，并完成整个交易流程。

##### ****支付账户的设立****

平台支付系统的账户模型往往分为支付账户和登录账号，支付账户指的是用户在支付系统中用于交易扣款的资金所有者权益的凭证，而登录账号只是用户在系统中用作登录的信息凭证。

一个用户可以有多个登录账户，一个登录账户下也可以有多个支付账户。

支付账户的作用包括支付交易、记账、对账、风控和信用需求。

平台上交易支付行为的发生都会产生交易金额在不同的账户之间进行流转：

当用户在某互金平台上完成完成绑定银行卡的行为时，那么将会通过支付渠道在该平台对应的托管银行里建立一个支付账户。

而用户在平台上进行付款时需要验证其支付凭证才能从其绑定的银行卡中完成扣款，在这个过程中，支付账户就是用户在支付系统中交易资金的所有者权益的凭证。

而交易完成后，资金本质上是从个人的支付账户流动到平台的支付账户中。

因此，支付渠道对于用户和平台的支付账户的建立至关重要。

PS: 平台如果扣了用户的钱，资金本质上是从个人的支付账户流动到平台的支付账户中。需要把这部分资金放到支付公司在银行的托管账户下面，然后，等到t+1结算的时候，托管账户的钱会结算到平台的自有账户中；

##### ****资金的清分****

用户订单支付完成后，钱款流入支付公司在银行托管的账户中，而在固定的时间点对账无误后，支付公司会按照既定的规则将各个平台商家的所得货款分账到各个商家的银行账户中，这就是支付公司到商户端（平台商家）的资金的清分。

PS:但是在互联网金融公司中，是支付公司到商户端（互金公司等平台）的资金的清分；

#### ****通常需要至少对接三家支付渠道****

##### ****1.稳定性****

支付渠道首先需要保持足够的稳定性，不稳定的支付渠道可能会导致支付流程崩溃、掉单等情况的发生。

##### ****2.成功率****

支付渠道的成功率也是非常重要的，支付渠道的成功率较低的话会很容易导致大量的掉单的情况，用户的支付体验较差。

##### ****3.手续费****

支付渠道的使用并非免费的，通过支付渠道的每一笔交易都会被支付渠道公司收取一定百分比的手续费，平台存在大量交易的情况下，选择手续费高的支付渠道会导致平台支付渠道的成本变高。

因此，对比多家支付渠道的情况下，选择手续费较低且稳定性和成功率有保障的公司是最佳的。

一般大流量的平台往往可以拿到较低的手续费率，比如支付宝和微信等第三方支付渠道给大型交易平台的支付手续费一般会在0.3%以下，甚至更低；而个人商户或者小平台的费率比较高，可能达到0.6%左右。

##### ****4.支付限额****

出于资金安全和风控的角度考虑，很多支付渠道都会定义其对应银行支付的支付限额，比如使用某支付渠道单日支付金额限制不超过5W。平台在选择支付渠道时，支付限额较高的渠道相对来讲具有更大的支付便捷性，在用户支付大额的订单金额时，不会很容易被限制而无法完成单笔支付。

##### ****5.其他因素（支付流程)****

支付流程主要是关于支付渠道的的产品细节沟通，比如该支付渠道公司的支付走的是认证支付还是快捷支付，还是两者都有？是通过API接口形式还是SDK嵌入的形式？

SDK嵌入形式会导致底层数据平台端无法获取，平台可以获得到的就是一个支付结果，但是API的对接形式平台自己可以监控整个的支付流程，包含支付中发生的异常情况监测，比如响应超时的情况等；还有，需要确认字段信息，支付四要素（姓名、身份证、银行卡号、预留手机号）等；

#### ****技术对接****

在确认好业务支付流程和具体的产品方案细节后，就将进入技术对接的阶段。这个阶段内双方公司的研发同学会进行技术层面的对接和调试，根据确定的支付流程细节的方案来确定需要开发的内容，并按照支付公司提供的接口文档和流程图等资料来进行支付功能的开发，；比较核心的内容就是”支付“和”对账“：关于支付主要考虑支付在交易流程中如何调用来唤起支付，而对账主要是进行公司内部对账、公司与商家对账、公司与支付渠道对账的数据记录。

PS：一般这种支付信息对账都是T+1进行的。

### 支付清结算系统

#### 支付流程

先说个比较简单的支付场景，用户（姑且称他为小明）用绑定的银行卡（用宇宙第一大行工行为例）来购买某电商公司（老熊公司）的产品。小明需要先在老熊公司网站上完成银行卡绑定的操作。绑卡以后，就可以使用这个卡来购买商品。 首先是挑选商品和下单，其后是执行支付。下单之前的流程不做介绍， 我们从支付开始，来说明支付过程中的清结算问题。  
为了简化，我们先从比较简单的同渠道、公司内购买的场景来开始。商品也先假定为虚拟产品，比如会员卡。 为了实现这个流程，有一些前置的操作需要完成：

老熊公司已经对接了工行的快捷支付接口。通过这个接口，可以实现绑卡（签约）、支付、退款、查单等操作。。

老熊公司已经按照工行的要求，在工行开了备付金账户。老熊公司通过工行接口的所有收款、退款等资金往来，都发生在这个账户上。

小明在老熊公司的应用中绑定了自己的一张卡，为了简化处理，小明绑定的也是工行的卡，先省略掉跨行结算的步骤。用户小明在手机或者PC Web上购买了100元的虚拟产品，比如很多公司会使用的会员卡。这里我们先从虚拟物品入手，因为实体物品情况会复杂一点，供应链和物流也是一个大课题，购买实体物品就需要考虑这个问题，而虚拟产品就可以暂不考虑。然后小明在网站上执行下单、支付操作。

老熊公司的支付系统接收到小明的支付操作请求后，系统首先会校验订单是否有问题，然后调用工行快捷支付接口，从用户的工行卡上扣除100元， 用户的工行卡的扣款是实时进行的，也就是说，这个操作完成后，小明查看他的工行余额和流水，会有一笔100元的交易，并且账户余额也减少了100元。 但是这个钱并不是直接就进入了老熊公司的（结算）账户上的。工行在第二天凌晨会对前天的交易进行清算和结算。在计算收入的同时，也从中扣除掉通道费用，得到最终应该划拨到结算账户上的金额。在这个例子中我们假定手续费按支付金额收费，比例为0.1%。 这里笔交易，支付给工行0.10元，公司收入99.9元。这里有个需要注意的地方。有些银行是在扣除手续费后，将前一天的余额全部划拨到结算账户上；有些银行是先全额划拨所有收入到结算账户上，然后扣除手续费。

#### 交易流水

用户执行支付后，系统首先需要记录交易流水，流水的内容包括：

交易主体：即发起本次交易的出款的用户，一般是记录ID、姓名等信息。

交易账户：即用户购买时使用的出款的账户，这是用户在工商银行的卡，实际账户是建立在工行，但在电商系统中，为了便于结算，为这个账户建立一个代理。这个账户在系统中的ID是10001（数据本身无其他含义）

交易对手：即出卖虚拟产品的业务部门，一般记录部门的ID、名称等信息。

结算收益： 交易对手能够拿到的金额。这里是 支付金额-渠道费用，即99.9元

对手账户：即虚拟产品的收款账户，为了便于结算，公司一般会对每项业务设置独立的结算账户。这个账户在系统中的ID是 20001（数据本身无其他含义）。

交易渠道：即工商银行的快捷支付，还需要记录渠道的ID， 名称等；

渠道结算账号：这也是个代理账号，记录在渠道侧的交易流水。

渠道提交时间：请求渠道执行支付的时间；

渠道支付时间：渠道一般会在返回的报文中说明本次交易的执行时间。 如果没有，则使用渠道的支付接口返回时间。

渠道费率：渠道的手续费，这里假定工行是按支付金额收费，比例为支付金额的0.1%。

渠道费用：这里是支付金额\*手续费率， 即0.1元。

发起交易日期：2016年12月12日 13:00:10，即用户提交订单后，虚拟产品业务调用支付系统接口执行支付的时间。

执行交易日期： 2016年12月12日 13:00:11，即支付系统接口调用时间。

支付截止日期：必须在此日期前完成支付。

订单信息：在本例子中是会员卡，一般需要记录业务方订单ID、名称、内容等信息。

订单金额：提交过来的原始订单的金额 100元；

支付金额： 用户实际支付的金额，由于没有使用优惠券、打折卡等，这里支付金额等于订单金额，都是100元。

没有使用卡券、没有和合作方分成，这两块内容暂不记录。

交易流水是在完成支付时实时生成的。这个流水信息是后续记账的依据，所以务必在流水中真实记录能收集到的所有的现场信息。 这里从：

交易主体，即掏钱的小明

交易对手，即收钱的业务方

交易渠道，即工行快捷

交易商品，即会员卡

角度来多方位全角度的描述这笔交易。大家会注意到这里有不少冗余信息。实际上对交易涉及到所有可能会被修改的信息，比如用户姓名，商品名称，商品价格，都需要在这里留一个快照，以便后续回溯和审核。

#### 会计主体

不用说，这一笔账是老熊公司的账务，不是工行的账务，也不是小明家的账务。虽然这里会有工行和小明的信息，但记账的目的是为了了解和改进老熊公司的经营状况服务的。 老熊公司不是某个大公司的分公司或者子公司，它是一家独立核算的、具有独立的资金和经营业务的单位，从会计学角度来说，他是一个独立的会计实体。

#### 会计要素

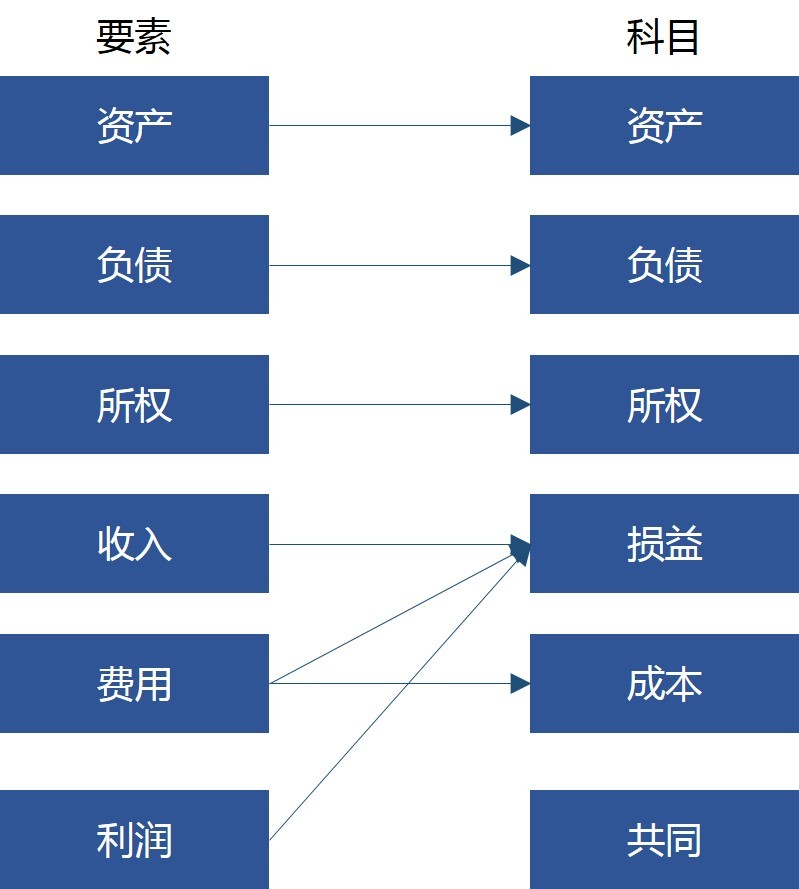
从概念上来说，所有和钱有关的活动，买会员、用户充值、支付手续费等，都需要记账，这些活动，称之为会计对象。 每个公司都有不同的会计对象，有时候同一类活动，叫法还不一样。如果直接用这些活动内容来记账，那就没法比较每个公司的情况。 比如新浪说我家微博广告收入300万，网易说我家卖猪收入了300万，到底谁家更赚钱？需要有一个记账的标准，让大家分门别类的做记录。对会计对象做规范化的管理，这就引入会计要素的概念。

会计要素是对会计对象进行的基本分类，是会计核算对象的具体化。 如果说会计对象是个Object，则会计要素是定义这个Object的Class。 不同的国家对会计要素有不同的规定。 国际会计准则委员会（IASC）在《编制和呈报财务报表的结构》将会计要素其归类为资产、负债、权益、收益和费用五个要素。美国财务会计准则委员会（FASB）在《财务会计概念公告》中将会计要素归类为资产、负债、所有者权益(净资产)、业主投资、派给业主款、综合收益、营业收入、费用、利润、损失十个要素。 我国《会计准则》将会计要素归类为资产、负债、所有者权益、收入、费用和利润六个要素。 其中资产、负债和所有者权益，是反应公司的财务状况的；它满足如下恒等式：



#### 会计科目

六大会计要素指明了需要记账的scope，但毕竟粒度还是太大了。为了更详细地了解公司财务情况，引入会计科目来对会计对象进行第二层次的划分。使用IT的语言来说，会计科目其实就是一个分类体系，用来分门别类地记账。 在实现上，他也是一个编号+名称，IT俗称字典表。 从定义上来说，会计科目是指一个涵义明确、概念清楚、简明扼要、通俗易懂的**标准名称**。 会计科目按照经济内容的性质不同，可以分为资产类科目、负债类科目、所有者权益类科目、损益类科目，成本类科目，有些金融企业还有资产负债共同类科目。在每一类会计科目下，还可以继续细分，详细内容可以参考2016年财政部发布的新会计准则。 会计科目和要素之间的关系：



会计科目还分为总账科目和明细科目。从IT角度，可以认为总账科目是一级分类，而明细科目则是这个一级分类下的二级、三级，甚至更多级别的详细的科目。 记账时，会同时记录到总账、明细科目。 在电商的支付系统中，一般会设置如下科目：



#### 会计账户

账户是指对会计要素的具体内容所作的科学的分类，其包括两方面的内容：账户的名称、账户的用途与结构。会计科目是设置账户的依据，也是账户的名称。 比如对银行存款这个会计科目，也会设置一个对应的银行存款账户，用来跟踪公司在银行存款的变动。 在这个案例中，将设置的账户同会计科目。

#### 记账凭证

想想在以前没有电脑的时候，去买公交卡，公交公司阿姨会认真地记录你买的卡的卡号、买卡人的姓名、卡的面值等信息，运气好的时候还会给个发票。 一般来说，阿姨会将购买记录登记到一个账册上，形成记账凭证，并在这里会登记发票号码。在现在高科技时代，这个凭证还是少不了了。 先说明细账，记录内容如下：



这里详细记录每一条交易信息，当然，通过计算机系统，可以记录更多详情，包括时间、地点等。

#### 会计分录和记账

大家经常看到的记录应该是这样的：



如上， 银行存款、服务成本、主营业务收入，属于总账科目，而工行收款、会员卡、工行手续费，属于明细科目。 这里采用的是复式记账法中借贷记账法。 对应的账户结构如下：



借贷复式记账法的特点是：

采用借、贷作为记账符号，建立在会计恒等式基础上，遵循**有借必有贷，借贷必相等**的原则。

账户基本结构是： 左侧为借，右侧为贷。

一般采用如上图所示的T行账户的形式来描述。

借贷所代表的增加、减少的含义并不固定，和账户的性质有关。



更多问题

作为清结算的入门介绍，这里介绍的是最简单的场景，以此来解释清结算相关的概念，特别是会计学一些从IT角度不容易理解的名词。实际上，这个场景还有很多问题：

严格的说，会员卡的收入，还不能立即作为公司主营业务收入。会员卡是预付款项，用户开始使用会员卡，公司需要为这个使用提供服务；用户结束使用会员卡之后，这一笔开支才算是真正落入公司主营业务收入中。

会员卡在使用期间，公司针对会员业务的各种开销，要分摊到这一段期间的会员上。将开销分摊到每张会员卡上，计算其使用成本，最终才能够计算出收益。

用户会员卡购买的款项是立即反映到备付金账户的，但并不是立即到结算账户的，一般是T+1结算，也就是第二天银行才会将清算好的资金打到公司结算账户上，这种情况应该如何记账？

如果支付过程中使用了代金券和优惠券，那又应该如何考虑？

此外，还有退款、充值等场景的清结算，这些问题都将在本系列的文章中详细介绍。本文仅介绍一些相关的概念， 欢迎大家继续关注后续内容。

#### 清算CODE

|  |  |  |
| --- | --- | --- |
| **账户 DEAL\_PRODUCT\_CODE** | **清算：FUNC\_CODE** | **备注** |
| PROD\_00\_RS\_0001 | 1001 | 预授权 |
| PROD\_00\_RS\_0002 | 1002 | 预授权完成 |
| PROD\_00\_RS\_0003 | 3001 | 红包兑换 |
| PROD\_01\_RS\_0001 | 3001 | 君融贷投资户投资 |
| PROD\_01\_RS\_0002 | 3001 | 君融贷还款 |
| PROD\_01\_RS\_0003 | 3001 | 现金券转让 |
| PROD\_01\_RS\_0004 | 3001 | 债券转让 |
| PROD\_01\_RS\_0005 | 3001 | 君融贷企业投资 |
| PROD\_01\_RS\_0006 | 3001 | 申购活期产品 |
| PROD\_01\_RS\_0007 | 3001 | 赎回活期产品 |
| PROD\_00\_RS\_0007 | 4011 | 储蓄消费 |
| PROD\_00\_RS\_0008 | 4012 | 信用消费 |
| PROD\_00\_RS\_0009 | 4013 | 代收 |
| PROD\_00\_RS\_0010 | 40131 | 实时代收 |
| PROD\_00\_RS\_0011 | 4014 | 代付 |
| PROD\_00\_RS\_0012 | 4015 | 充值 |
| PROD\_00\_RS\_0013 | 4016 | 提现 |
| PROD\_00\_RS\_0014 | 4017 | 冻结 |
| PROD\_00\_RS\_0015 | 40171 | 冻结授权码 |
| PROD\_00\_RS\_0016 | 4018 | 解冻 |
| PROD\_00\_RS\_0017 | 40181 | 账户资金解冻（授权码） |
| PROD\_00\_RS\_0018 | 4019 | 消费前退款(冲正) |
| PROD\_00\_RS\_0019 | 4020 | 消费后退款 |
| PROD\_00\_RS\_0020 | 5024 | 退票 |
| PROD\_00\_RS\_0021 | 10012 | 清结算冲正 |
| PROD\_00\_RS\_0022 | 10012 | 清结算抹账 |
| PROD\_00\_RS\_0023 | 10013 | 手动冲正 |
| PROD\_00\_RS\_0024 | 10014 | 订单退款 |
| PROD\_00\_RS\_0028 | 4014\_1 | 一分钱验证 |

### 支付流程系统

#### 支付流程

接着之前的小明购买会员卡的案例，不考虑优惠券和卡采购的情况，会计分录：

借：

应收账款-工行收款 100-100\*0.1% = 99.9

服务成本-工行手续费 100\*0.1% = 0.1

贷： 主营业务收入-会员卡 100

在线上的实时处理流程如下：

用户购买会员卡，提交订单，会员卡向订单系统请求生成订单，订单系统向支付系统发出支付请求；

支付系统生成支付记录，并向银行发出请求；

银行实时从小明的银行卡账户上扣款100元，通知支付系统小明支付成功；

支付系统账务子系统在自己的账户体系中记录小明的这一笔消费支出，给会员卡业务账户增加对应的资金，通知会员卡系统发送卡给小明。

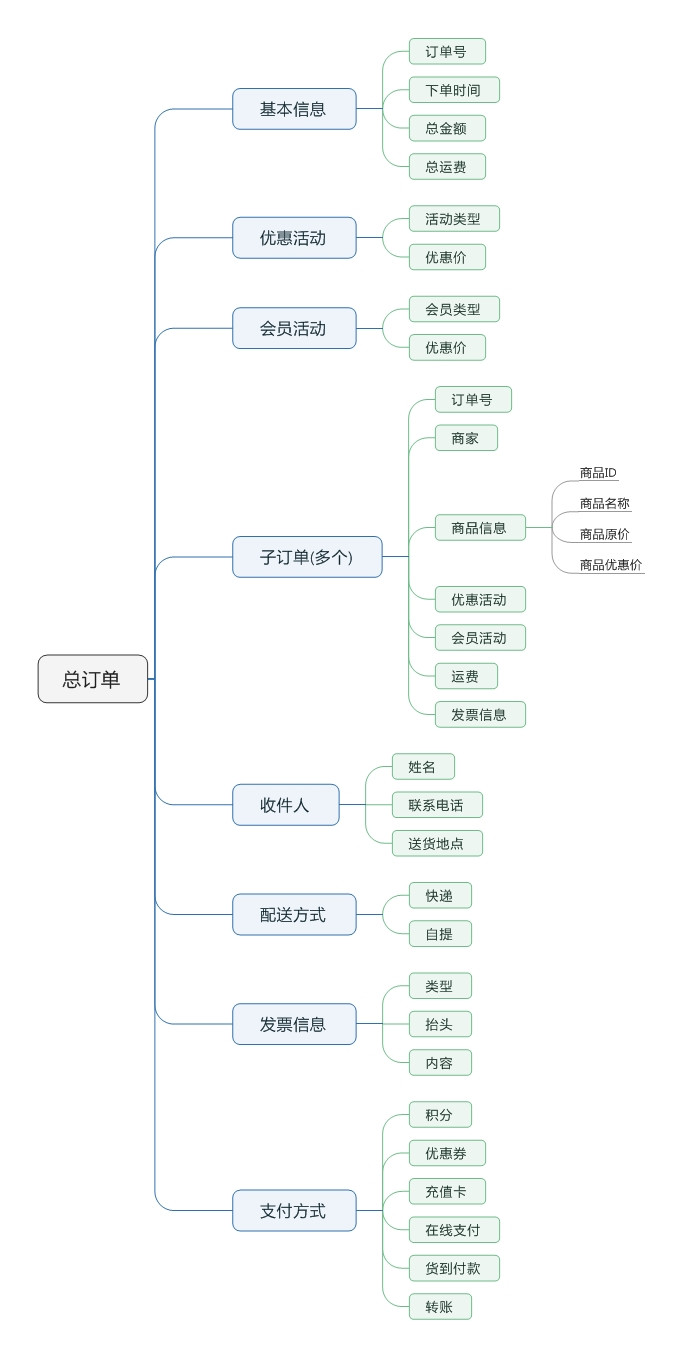
支付系统发送消息异步通知会计系统进行记账。

这5个步骤都是线上的流程，在此过程中，各个子系统之间的交互如下图所示。 我们逐步分析这里涉及到的对象。



#### 支付订单

这个流程中，首先生成的是支付订单。这是一个比较简单的订单，仅涉及到一个商家和一个商品。 在比较复杂的电商场景中，一个订单会涉及到多个商家、多种商品以及对应的优惠活动。也就是，一个总订单会被拆分为多个子订单。这部分内容将在后续的订单系统设计一文中详细介绍。 而订单中和资金相关的内容，都需要在账户体系中建立对应的科目和账户。在请求支付时，只会将总订单提交支付，拆分子订单是在订单系统中完成的。

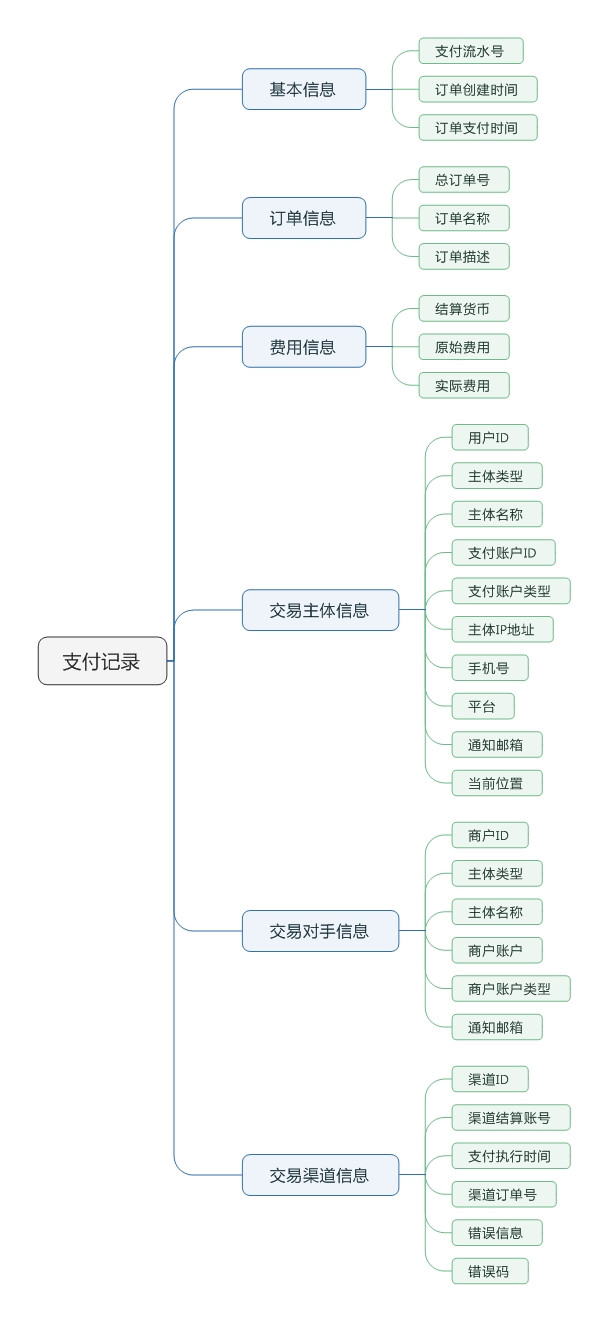


针对上述场景，为了简化处理，假定老熊公司当天总共完成了三笔会员卡交易，支付订单如下：



#### 支付记录

在这个流程中，订单系统向支付系统请求支付时，支付系统将产生支付记录（支付订单）。 支付记录内容比较多，这些数据是后续进行记账的基础。



上述的三个订单，两笔通过工行支付，一笔通过支付宝支付，其产生的支付记录如下，此处省略了其他和记账无关的字段内容：



这里需要注意的几个属性：

**订单号**: 这是总订单号。支付系统不再对订单进行拆分。 订单拆分是订单系统的功能。

**支付流水号**：在支付记录中，针对每个（总）订单号，会有对应的支付流水号。 如果用户使用组合支付，如上述场景，小明使用余额支付了20元，使用银行卡支付了剩余的80元，那这将产生2个支付流水号。为了避免洗钱风险，简化订单处理，包括淘宝在内，现在一般都不再提供组合支付的支持。

**支付方式和支付渠道**： 用户选择的支付方式和实际执行支付的渠道可以是不一样的。比如用户选择了广发信用卡来支付，但实际上电商公司没有直接对接农行，而是通过支付宝来对接， 那对应的支付渠道就是支付宝。

**本条记录在支付成功后，会产生记账凭证和对账凭证。** 这一条记录将产生多条会计分录，记账凭证是关联这些会计分录和支付记录的字段。对账凭证是根据对账周期来分配的。在日切后，对账凭证号相应的也会做更新。

在产生支付记录后，在上述流程的第5步通过消息机制来异步触发记账流程。账务系统接收到记账消息后，开始更新账户信息。记账分为两个阶段：

**支付记账**，针对线上的账户实时更新的需求，需要让用户及时看到账户余额和订单状态，账务信息记录到用户和商户上，采用单边账的形式。

**会计记账**，采用复式记帐法，满足会计记账需求，记录会计分录和余额，为对账和清结算提供支持。

#### 支付记账

支付记账是在支付流程中完成的，目的是让用户完成购买后，能够立即看到支付结果和账户余额。为了提升性能，支付记账一般采用单边账的形式，即将会计分录登记在用户侧或者商户侧。

在上述案例中，第6步处理，在服务器上与银行侧同步登记一笔从小明银行卡的支出，并在会员业务账户上登记一笔收入。 如果使用的是零钱支付，这一个步骤就很重要，从零钱账户上扣除费用计算余额，添加对应的消费记录，是在一个事务中完成。

#### 会计记账

会计记账采用复式记账，不同业务记账方式也不一样。小明买卡的案例中，需要记录的条目有：

在工行收款账户下，登记 99.9的借记 条目

在工行手续费的账户下，登记0.1元的借记条目

在主营业务收入-会员卡的账户下，登记100元的贷记条目。

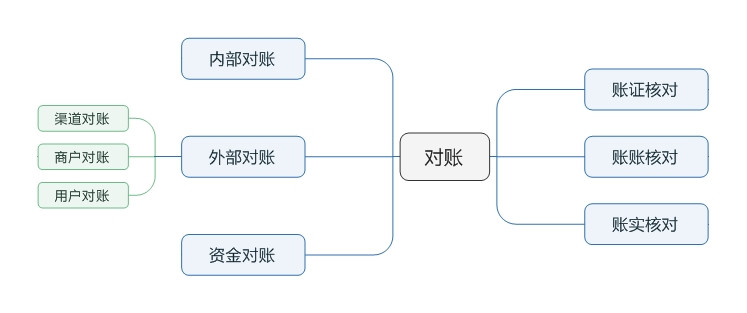
这3条记录是通过事务处理一次生成。当天发生的三笔交易，产生的记账内容如下：



实际实现上，科目一列，使用账号ID来替代。 每个账户的本期发生额，可以在另一个表中单独异步计算。

### 对账系统设计和实现

在会计上，对账分三个部分：账证核对、账账核对和账实核对，做到账证相符、账账相符和账实相符。在电商支付系统中，需要完成的对账工作包括内部对账、外部对账和资金对账。 其中外部对账又分为和支付渠道的对账、和商户的对账以及和用户的对账。 这里简单介绍每个对账流程以及对记账的影响，详细的对账和轧账流程，将在下一篇文章中说明。



#### ****内部对账****

内部对账的内容包括：

核实账户系统中的账务与支付记录的一致性；

核实会计系统中的账务与支付记录的一致性。

这是后续账账核对和账实核对的基础，也是对外对账的基础。 这两个对账任务一般是在后台定时运行（5分钟运行一次），除非系统有bug或者发生故障，内部对账一般不会出现差错。如有差错，也需要人工处理。

#### ****渠道对账****

一般银行、第三方支付提供T+1的对账单，这是执行渠道对账的依据。 在上述案例中，支付系统每天拉取工行前一天的对账单，核对交易流水。与此同时，按照从工行获取的对账单，记录资金归集的账务。

接收到工行对账单后，按照对账单生成凭证：



注意上述的日期、凭证号，以及借贷关系。

#### ****账账核对****

完成渠道对账后，需要进行日结和试算平衡：

总分类账各账户本期借方发生额合计与贷方余额合计是否相等；

总分类账各账户借方余额合计与贷方发生额合计是否相符；

核对各种明细账及现金、银行存款日记账的本期发生额及期末余额同总分类账中有关账户的余额是否相等。

从科目维度，计算：

科目期初余额+科目当日发生额=科目期末余额

下级科目余额总和=上级科目余额（科目总分检查）

#### ****账实核对****

对电商公司来说，最重要的账实核对，是验证银行存款的变化和实际资金流向是一致的。一般是需要登录到银行网银系统中来人工核实。 如果对接的银行多，银行提供网银对账单，也可以自动进行，查询出入款总额。

详细的对账和轧账流程，将在下一篇文章中说明。

#### 对账设计

可以说，对账是支付系统最头疼的事情。每一笔交易，都要做到各参与者的记录能够吻合，没有偏差。对账系统的工作，是发现有差异的记录，即轧帐；然后通过人工或者自动的方式，解决这些差异，即平帐。

对电商系统来说，每一笔交易，在所有相关主体侧都要能对得上：

交易主体，如果发起人是个人，必须能够从个人交易历史记录中找到这笔交易。但大部分人不会保留电子记录，所以一般是提供可以下载的账单或交易记录，让用户自己对去。

交易对手，一般是商户。商户侧对账处理同用户侧，也仅仅提供对账单。

交易渠道侧，这是对账的重点，一是核实交易流水，二是核实交易佣金，毕竟是租用人家通道做结算的。

那有哪些记录需要对账？ 目前主要是两个：一个是交易记录；一个是退款记录。

#### 对账处理流程

一般来说，对账流程涉及到如下步骤： 渠道对账单下载、本地交易记录准备、轧账、平账。

#### 渠道对账单下载

银行，第三方支付，银联等，基本都会提供对账单下载的功能。不过也有少数工作做不到位或者太到位的银行，只提供账单查询后台，不提供对账单下载功能。

对开发人员来说，这里有几个坑：

对账单格式不一。文本，XML，csv的都有。为了后续能够统一处理，在账单下载完成后，需要进行标准化处理。

下载方式不一，HTTP，HTTPS，FTP的，都有。下载程序需要按照渠道的协议来处理。

下载时间不一，一般是凌晨1点后，到中午12才能用的也有。如果在预定的时间取不到数据，需要注意重试读取。

稳定性差。FTP服务器出问题那是常有的事。渠道侧解决方案往往就是重启。所以重试机制是必要的。

看一下第三方支付的对账单情况：



银行直连的对账情况：



技术选型上，HTTP(S)用apache httpclient即可实现链接池和断点续传， FTP也可以使用Apache Commons Net API。 但不管是哪一个，都需要设置重试次数和链接超时间。重试次数和间隔的设置需要小心，重试太频繁，容易把服务器打死.；时间间隔太大，又会阻塞后续处理步骤。5～10分钟是一个合适的重试间隔区间。

链接超时指在服务器出现问题时，连接在指定时间内获取不到数据即自动断开。这个很容易被忽略。我们有一次系统出问题，是渠道侧的FTP假死后重启，导致我们的客户端挂住，一直在等待重新链接。

#### 渠道对账单标准化

找个例子大家看看， 比如微信的对账单，他是csv格式的，包括如下信息：

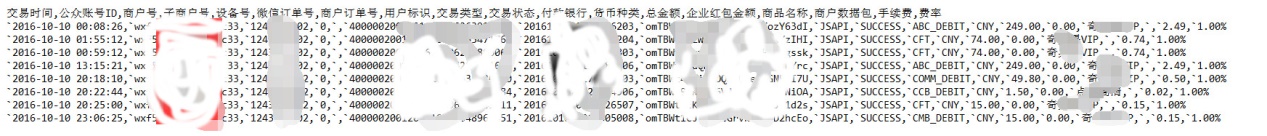
交易时间：这是在微信侧的支付完成的时间。 这个时间会成为一个陷阱。

公众账号ID，商户号,子商户号,设备号： 这些信息需要做验证，确保是自己的单子，不要让微信把老王家的单子也给发过来了；

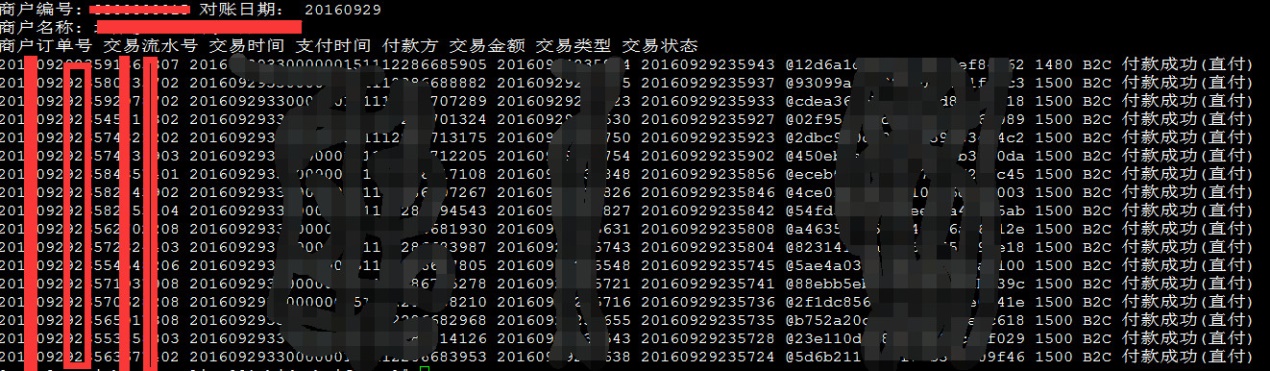
微信订单号,商户订单号： 这两个是对单的核心。前者是微信侧产生的订单号，在微信支付接口返回值中有。但是万一收不到这个返回值，那在本地记录中可能就空了。 后者是我们发送给微信的订单号，一般用这个来做对单依据。两边的数据中都会有这个值。

用户标识,交易类型,交易状态,付款银行,货币种类,总金额,企业红包金额： 这几个就是对单的核心字段，必须确保双方是一致的。

商品名称,商户数据包,手续费,费率：这些是可选验证。



而某宝的对账单，是文本格式的，用空格隔开。他们家的就简单很多，只有商户订单号，交易流水号，交易时间，支付时间，付款方，交易金额，交易类型，交易状态这些字段。



由于每个渠道的账单格式都不尽相同， 在得到账单后，下一步是对账单做标准化处理，这样轧帐以及后续工作就可以统一处理了。 标准化后的账单数据可以放在文件系统或者数据库中。这取决于交易数据量。每天百万以上的量，还是使用文件系统，比较合适。数据库操作相对比较慢，也浪费资源。

基于文件系统的标准化涉及如下内容：

**文件格式标准化：**统一使用csv或者json或者xml格式。如果是使用hadoop或者spark来对账，使用csv是个不错的选择。

**文件存储统一化：**文件目录，文件名都需要遵循统一命名规范。

为了加快处理速度，我们使用hdfs作为文件系统，有利于后续的对账的处理。

#### 本地交易记录准备

本地交易记录的准备，总的来说有如下方法: – 啥都不做，直接用原始数据。鉴于大部分系统使用的是mysql，这也意味着在MySQL上做对账。对账时需要大量的数据查找工作，必然会影响线上业务。在数据规模较大，比如超过100万时，就不太合适了。

当然，还有一个选择是使用备库来执行对账，这样既简单，也不影响线上业务。这是典型的空间换时间的做法。

如果业务大到需要分表分库才能处理，那对账数据准备也不一样。使用分库也不现实，因为分库一般是按照主体id，而不是渠道id，来分库，这样对账就需要在多个库上进行，效率反而降低了。而对分表分库建立从库也非常耗费资源。这种情况下，需要同步一份数据到(hdfs)文件系统中，或者NOSQL数据库上。

由于交易记录是支付系统核心数据，有大量的应用，如信用、风控等，都需要交易记录数据。这些应用对交易记录的需求还不完全一致，为了提升性能， 交易记录会使用异步的方式来将数据投递给使用方。 交易记录在入库时，投递消息到消息系统中。使用方监听这个消息，一旦收到新消息，则从交易记录库中查询数据，获取数据并更新到库中。关于此类数据同步的文章不少，这里就不详细介绍。

#### 轧帐

轧帐是按照客户订单号来比较本地交易记录和渠道交易记录是否一致。从算法角度，是计算两个数组的差异。在单机运行时，可以采用的算法不少，这里不详细介绍。 我们推荐采用mapreduce来轧帐，这有个优势，可以按照订单号将渠道提供的记录和本地记录shuffle到同一个reduce处理上，这样就可以很容易进行数据比对。 轧帐中最大的坑，莫过于切分点的问题。

比如以整0点为切分点，那存在一个问题，本地23:59发起的交易，到了渠道侧，可能会在00:01处理，这一笔交易变成第二天的帐了。实际处理中，一笔交易在渠道侧处理，花上几分钟都有可能。 对于切分点附近无法确认的帐，做一个时间窗，在时间窗内的数据，留待第二天对账时继续处理。

#### 平帐

发现两边不一致的数据，那应该如何处理？数据量不大时，记录起来，人工甄别就行。但如果数据量很大，每天上千条，人工处理就成本太高了。这个没有统一的处理方法，需要根据有问题的数据，做个分析，然后做自动处理。 针对交易记录的对账的处理，主要有如下情况：

本地未支付，支付渠道已支付。这主要是本地未正确接收到渠道下发的异步通知导致。 一般处理是将本地状态修改为已支付，并做响应的后续处理，比如通知业务方等。

本地已支付，支付渠道已支付，但是金额不同，这个需要人工核查。

本地已支付，但是支付渠道中无记录；或者本地无记录，支付渠道有记录。在排除跨日因素外，这种情况非常少见，需要了解具体原因后做处理。

针对退款的对账处理，主要有如下情况：

本地未退款，支付渠道已退款，则以支付渠道为准，修改本地为已退款状态，并触发后续处理。

本地已退款、支付渠道已退款，但是金额不同，需要人工核查；

本地已退款，但是支付渠道无记录；或者支付渠道有记录，但是本地没有。 在排除跨日因素外， 这种情况非常少见，需要了解具体原因后做处理。

#### 账户跟多渠道对账

一.对账逻辑：

1.1从交易表中查询账期是T-1日的多渠道数据(data\_from=3)，且stauts\_id=21 and func\_code in (4015,4017);

1.2从交易表中查询查询账期>=T-3日的账户数据(data\_from=0),且stauts\_id<>21 and func\_code in (4015,4017);

将上诉多渠道和账户的数据进行对账，对账结果存入SETTLE\_BALANCE\_ENTRY表，

RULE\_TYPE=2，SETTLE\_BALANCE\_ENTRY表中其它的字段含义同清算一期。

另外需要将平账、错账、短款标记到交易表中账户的数据上，注意多渠道的数据不做

任何改动。

二.对账key和对账项见数据库表配置 : SELECT \* FROM `SETTLE\_RULE` a where a.RULE\_TYPE ='AM01';

### 渠道侧处理

这里我们一步步详细介绍清结算中在渠道侧如何进行。这里的渠道侧，既包括银行，也包括第三方支付以及银联。

#### 支付

上一篇文章中的小明通过工行在老熊公司买了一张100元的会员卡，工行按照支付金额的0.1%收取手续费， 产生了如下会计分录

借： 银行存款-工行收款 100

贷： 主营业务收入-会员卡 100

借： 服务成本-工行手续费 0.1

贷：银行存款-工行收款 0.1

这里我们详细介绍在支付时的清结算过程，分别从银行直连、第三方支付的角度来说明。

#### 银行直连

用户通过工行支付的款项，在银行侧并不是直接落地到老熊公司的结算账户上，而是先到一个中间账户上。这就需要了解银行侧的账户和资金处理流程。针对第三方支付和电商直连银行， 在银行侧的账户和资金处理流程是不一样的。 电商公司直连银行，不需要复杂的备付金管理。 当电商公司(第三方支付公司类似)和工行签约开通支付接口后，工行会为这个接口开设一个结算账户。通过这个接口流转的资金最终会结算到这个账户上。这也是电商公司可以提现的一个账户。另外，为了方便资金清算，工行还会为电商公司开设一个收付账户，这个账户是直接对接线上资金处理的，这是工行内部账户， 电商公司无法操作这个账户，但是可以看到这个账户的收付情况。

支付时，资金在工行侧的流转流程：

1.用户提交支付请求，支付系统调用工行快捷支付接口执行支付。  
2.工行服务器检查用户账户余额和状态，如果余额不足或者状态异常，则停止支付。继续检查电商公司的收款账户的状态。如果状态正常，开始执行支付。资金从用户账户转到收付账户上。

工行采用T+1的结算，将T日用户支付的所有款项，扣除退款，扣除手续费等费用后，将剩余资金打入电商结算账户上。

而在商户侧，对前期所述的记账流程，我们需要有调整。用户付费不能直接进入公司存款账户，而是记在应收账款下。需等待银行完成清结算后，才能将商家在该银行所有的收款转到存款中。

用户付费时，会计分录：

借： 应收账款-工行收款 99.9

服务成本-工行手续费 0.1

贷： 主营业务收入-会员卡 100

在电商侧执行日切，将当前的应收账款归集到存款上，在接收到工行对账单后，对这个日切结果也要对账。 假定当前收入为200万元，手续费为2千元，对应的会计分录为：

借： 银行存款-工行 1,998,000

贷： 应收账款-工行收款 1,998,000

通过这个会计分录，怎么知道这一笔手续费是谁付的？ 要知道这只是从会计角度做的记录，在具体存储库的设计时，还需要添加更多的信息，包括交易发生的时间、会员卡号等信息。

对工行来说，这些资金都是行内流转，不涉及到跨行转账的问题。 因此在银行内部的处理也比较简单，一般就是一个内部的信息流，分别将商家账户和用户账户的资金做了增减，并无实际的资金流转。

#### 第三方支付平台

对于没有直连的银行，可以通过第三方支付来收款。 第三方支付侧的资金处理，和银行的类似。以支付宝为例，当商家在支付宝开通支付接口后，支付宝为商家建立账户以及商户号。 每个业务对应一个商户号。 用户提交支付后，支付宝完成收款后，直接进行分润，扣除手续费，将剩余资金转移到商户号上。

借： 应收账款-支付宝会员账户收款 99.9

服务成本-支付宝会员手续费 0.1

贷： 主营业务收入-会员卡 100

这里记为应收账款，因为T+1后，支付宝还要将资金按照商户的要求，转移到指定银行账户上。假定这个银行账户为建行的账户，当天的 收入为200万元，手续费为2千元，对应的会计分录为：

借： 银行存款-建行 1,998,000

贷： 应收账款-支付宝会员账户收款 1,998,000

那对于这一笔收款，支付宝是如何处理？这就涉及到备付金的问题了。

#### 备付金

关于备付金，建议大家详细阅读[《支付机构客户备付金存管办法》](http://doc.cocolian.cn/reference/2013/06/07/zfjgbfj/)全文，对备付金的使用和管理有详细的介绍。当用户在电商网站（如淘宝网）购买商品，下单后，资金并不是立即进入商家的账户上，而是暂存在第三方支付公司。当用户确认接收到商品后，资金从第三方支付公司转到商家账户上。这个资金被称为备付金。即 “指支付机构为办理客户委托的支付业务而实际收到的预收待付货币资金”。备付金由银行托管，备付金银行分为存管银行和合作银行。

存管银行只能有一家，负责处理备付金跨行收付，并对支付机构存放在各银行的备付金信息做归集，合并与监督。

合作银行可以有多家，可以办理针对本银行的备付金存取和监管。

和备付金相关的账户有三类：

存管账户： 可以以现金形式接收客户备付金、以银行转账方式办理客户备付金收取和支取业务的专用存款账户。针对备付金存管银行，可以一个省开设一个存管账户。 只有存管账户才能够支持跨行付款。

收付账户： 可以以现金形式或以银行转账方式接收客户备付金、以银行资金内部划转方式办理客户备付金支取业务的专用存款账户。在同一个备付金合作银行只能开设一个收付账户。

汇缴账户： 仅支持同行收款以及退款时原路退回。在备付金存管和合作银行都可以开设。 备付金银行应当于每日营业终了前，将备付金汇缴账户内的资金全额划转至支付机构的备付金存管账户或在同一备付金合作银行开立备付金收付账户。

有了收付账户，还有必要开设汇缴账户吗？ 这要看交易规模了，如果规模大，那是有必要的，因为有些地方银行可以给非常低的手续费用，可以省一大笔钱。这几个账户都是**备付金专用**的**活期存款**账户。汇缴账户到收付账户的划拨，是通过银行自己行内转账进行；收付账户到存管账户，就走人行的大小额、超级网银或者同城系统进行划拨。 当然，这些接口一般第三方支付是不能直接访问的，但可以通过银行提供的接口来进行。

这样在第三方支付公司，用户支付的资金首先进入汇缴账户或者收付账户，日终后，汇缴账户的资金全部汇总到收付账户上。 在上述小明的例子中，如果小明使用的卡是招行的卡， 在有备付金支持的情况下，小明支付的100元是通过招行的内部转账直接进入了支付宝在招行的备付金收付账户或者汇缴账户。这样将跨行转账变成行内资金划拨，可以减少通道费的支出，也使得资金到账更为迅捷。 第三方支付备付金的使用和调拨是个专门的话题。这里不再详细展开。

#### 充值

在爸爸公司指引下，资金沉淀成为电商公司孜孜以求的目标。用户充值既有利于提升支付体验，特别是对高频小额支付而言；由此也可以形成可观的资金沉淀，这对电商公司来说是个一举多得的好事。但此举是迈向违规的危险第一步。 接之前的场景，老熊公司在工行开设了一个对公账户，用于收款，这个账户可以用结算账户。先考虑用户小明使用工行卡充值的场景。 公司需要为每个用户开设一个账户，用来跟踪账户余额。这个账户有两种处理方式，所谓的虚户和实户。

**虚户** 指客户和商户的资金都存放在电商公司开设的对公账户上，老熊公司内部为每个实体维护一个虚拟账户。当客户给商家支付时，资金直接在虚拟账户上划拨，没有实际的资金流转，只有信息流。由于虚户是开设在电商平台上，所以也叫支付账户。大部分电商公司用的都是这种模式。

**实户**指在银行侧为每个客户和商户在对公账户下开设二级账户。每个账户都是相互独立的。对于少数服务大客户的电商公司来说，使用这种模式也比较方便。相对支付账户，这被称为银行账户。

这里介绍虚户的情况。 使用虚户时， 电商需要为客户和商户建立内部账户。 这个内部账户，对电商公司来说，需要记录在负债上。

回到充值的案例： 用户使用工行的卡，给自己的账户充值100块钱，不考虑手续费【手续费处理方法支付】，会计分录：

借： 应收账款-工行收款 100

贷： 客户A账户 100

资金先进入应收账款，因为充值和支付一样，资金也是在清结算完成后才到账的。 用户在其他行充值情况应该如何处理？ 如果电商公司和这个银行有开通支付通道，那就会有对应的对公结算账户，充值款项就会在结算完成后，进入这个账户。如果电商和这个银行没有对接，需要通过其他第三方渠道，如支付宝、微信支付等来完成充值，对应的，资金也会进入第三方渠道。这和支付情况也是类似的。 不管用户用哪个卡充值，最终的钱都记录在客户A的虚拟账户上。

第三方支付在处理充值时流程和上述是类似的。但是当电商调用第三方支付接口来完成充值时，和银行不一样地方在于，这个充值是实时入账的。虽然银行并不是实时把充值的金额结算到第三方支付的银行账户上的，而是到T+1完成结算后才会到账。但对第三方支付而言，在银行接口确认成功后，就可以假定T+1银行肯定可以把钱结算给自己。因此可以实时把电商在第三方支付的账户的金额增加，也即实时到账。

#### 提现

有充值，一般都需要提现。 可提现是一个危险的业务，提供资金逃离电商的途径。对充值和支付来说，就算出问题了，钱还在。而提现就麻烦了，一旦出问题，钱都没了。 对提现的账户和账务处理尤其谨慎。 电商提现的主要途径是银企直联、企业网银和第三方支付代付。

#### 银企直连

对于商户和客户规模不大的公司，通过将公司内部的ERP、财务系统等业务系统和银行接口对接，实现银企直连，可以让财务人员直接通过内部业务系统来打款。

#### 企业网银

一般银行都提供的企业网银接口，支持同行批量打款（代付）功能。银行将企业对公账户扣款，打到对应的个人账户上。对于个人跨行取现，本质上来说，是将对公账户的资金转账到对私账户上，目前银行侧转账需要通过人行的大小额系统和超级网银来进行。人行系统并不直接对企业开放，银行通过企业网银来封装这些接口，批发或者零售给企业使用。

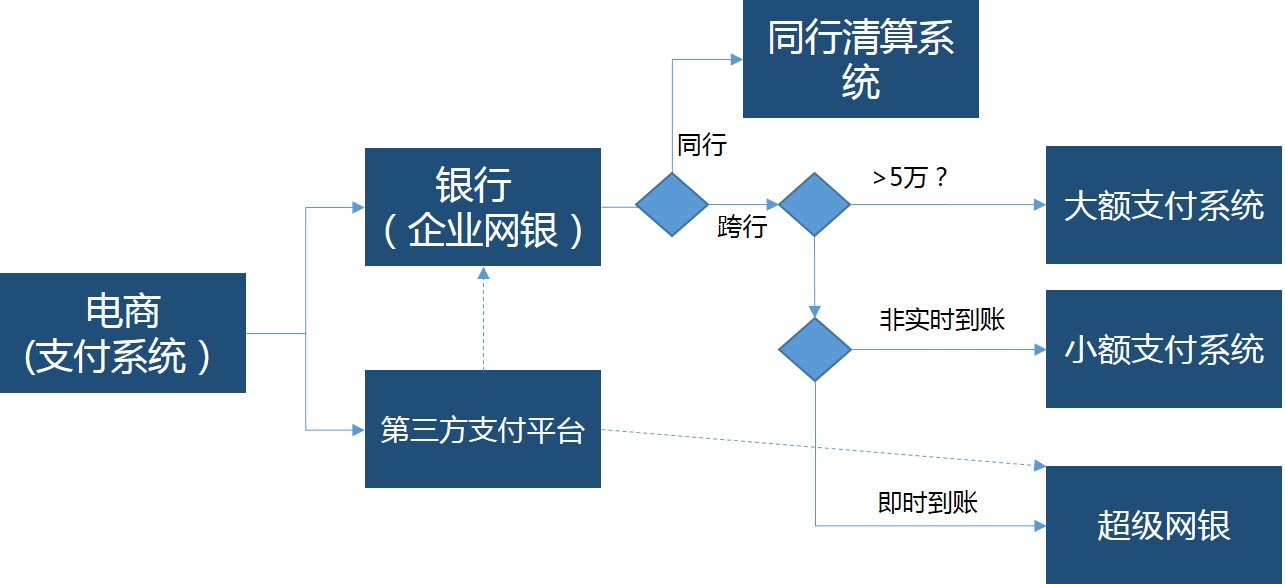
**行内清算系统** 银行内部转账一般通过这系统来进行，不限金额，实时到账，手续费低。 对于同行提现，如上述案例，小明需要将零钱提现到工行账 户上，老熊公司在工行也开设了用于收付的对公账户，则可以通过工行的行内清算系统，直接100元支付给小明。

**小额支付系统** 大小额支付系统同属于央行的中国现代化支付系统（CNAPS）。5万以内的普通贷记业务可以通过央行的小额批量支付系统（BEPS）来进行，这是7\*24小时运行，批量运行，收集到若干交易后统一打包处理，所以是非实时到账的，费用相对大额来说也比较低。此外，小额支付需要提供联行号，即支行信息，一般绑卡流程不会要求用户提供此类信息。

**大额支付系统** 大额实时支付系统（HVPS）每笔交易都实时发送，实时清算的，所以基本上能实时到账，跨行资金零在途。但大额系统运行的时间，仅限于工作日的 8:30 ~ 17:00运行，假节日也不运行。目前对使用大额支付系统进行转账并没有设置最低资金量的要求，也就是一块钱也会给你转过去。 但是费用较高，另外大额支付系统也需要支行信息（联行号）。

**超级网银** 全称是 网上支付跨行清算系统 ，2013年10月份正式投产运行。超级网银是对大小额支付系统的一个补充，接入机构不再限于银行，第三方支付也可以接入，所以有的第三方支付给商户提供的提现代发功能就是基于超级网银做的。7\*24小时实时到账，单笔上限5万元。 超级网银并不是所有的银行都支持，目前至少支持172家主要银行。此外，超级网银交易可以不需要联行号。

对电商来说，由于不能直接访问央行的这些接口，对接银行，通过银行来访问央行的接口，是主要的提现方式。 但不同的银行，针对不同的提现场景，收费不一样。所以在具体实现提现的时候，还需要有提现路由（打款路由），从费率、支持的额度、到账时间等多个维度来评估和选择提现渠道。



#### 第三方支付渠道

支付宝、微信等第三方支付公司一般也都提供代付服务，可以是通过文件来实现批量代付，也可以通过接口来实现提现。

第三方支付与用户提现的银行有合作关系且银行提供实时到账接口，此类银行支持实时到账的提现操作； 如果只提供准实时到账接口（例如2小时到账），则此类合作银行只支持准实时到账提现操作（支付宝为2小时到账）。

第三方支付与用户提现的银行无合作关系，只能通过银行小额支付系统定时跑批，此类只支持2天内到账。

不管通过哪个途径来实现，提现的会计分录和充值相反，借记用户的虚拟账户，贷记对公账户存款。

借： 客户A账户 100

贷： 银行存款-工行 100

#### 合规

本部分内容仅代表个人观点。每个公司的业务情况都不一样，一个业务是否合规，需要咨询法务人员。 由于央行控制支付牌照的发放，现在牌照价格也是水涨船高，每一次交易都刷新了历史记录。作为支付从业人员，在这个伟大的时代，能够参与并旁观这些历史事件，也是非常难得的。对一般的互联网公司来说，现阶段

这篇文章介绍渠道侧的清结算处理，下一篇文章将详细介绍在电商侧的相关处理，包括分润、优惠券等的处理。

### 电商侧处理

这篇文章的内容比较枯燥，就是一堆的业务对应的会计分录。 下一篇文章将基于这个会计分录来设计账户结构。 后续和银行、渠道的对账，也需要基于这些会计分录来进行。

#### 支付

继续小明的案例，小明通过工行在老熊公司买了一张100元的会员卡，工行按照支付金额的0.1%收取手续费， 产生了如下会计分录:

借： 应收账款-工行收款 100-100\*0.1% = 99.9

服务成本-工行手续费 100\*0.1% = 0.1

贷： 主营业务收入-会员卡 100

在电商侧执行日切，将当前的应收账款归集到存款上，在接收到工行对账单后，对这个日切结果也要对账。假定当前收入为200万元，手续费为2千元，对应的会计分录为：

借： 银行存款-工行 2,000,000-2,000= 1,998,000

贷： 应收账款-工行收款 1,998,000

#### 外部结算

如果小明买的是其他公司（凤凰公司）的会员卡， 该卡由老熊公司代理销售，代理价格是90元。这个流程会更复杂。买断式的代理，和仅仅支付手续费的代理在记账上略有区别。 这里仅提供供参考的版本，具体记账的会计科目需要按照公司的情况来具体处理。

老熊公司接收到凤凰公司发送过来的100张会员卡，每张卡售价是100元。 这些卡作为预售的代销商品入库，记录到代销商品中。其会计分录如下：

借：代销商品-凤凰会员卡　90\*100 = 9,000

贷：代销商品款-凤凰会员卡　90\*100 = 9,000

对应的会员卡信息，每张卡的卡号、价格等信息，可以记录到明细表中。

老熊公司销售出一张会员卡给小明，产生的会计分录如下：

借： 应收账款-工行收款 99.9

服务成本-工行手续费 0.1

贷： 主营业务收入-会员卡 100

借： 主营业务成本-会员卡 90

贷： 代销商品-凤凰会员卡 90

借： 代销商品款-凤凰会员卡　90

贷： 应付账款-凤凰会员卡 90

这里是将收入记录到主营业务收入的科目下， 实际情况应该按照公司的会计要求来进行。

老熊公司当天销售了10张卡，银行将当天的收入结算给老熊公司。这个可以单独分录，也可以和银行的其他结算一起分录：

借： 银行存款-工行 9,990

贷： 应收账款-工行收款 9,990

老熊公司当天将卡收入结算给凤凰公司，未考虑税费的问题，会计分录如下：

借：应付账款-凤凰会员卡　9,000

贷：银行存款　9,000

#### 促销

常用促销方式包括代金券、折扣券、买赠、满减等，以下简单介绍各个方式的记账。

#### 代金券

对代金券的财务处理目前没有统一的标准，会计处理方法因公司核算方式不同会有多种，有营销费用，销售折扣，主营业务成本等方式。在这个案例中，我们假定老熊公司发放了100张优惠券，每张价值10元。会计分录:

借: 营销成本-通用优惠券 1000

贷: 预计负债-通用优惠券 1000

当用户使用优惠券购买商品时，10元的优惠券，用户实际付款是90元，手续费按照90元\*0.1%来计算。会计分录：

借： 应收账款-工行收款 89.91

服务成本-工行手续费 0.09

预计负债-通用优惠券 10

贷： 主营业务收入-会员卡 100

这批优惠券到期后，实际发放了80张，还有20张需要回收， 每张价值还是10元， 回收时会计分录：

借: 预计负债-通用优惠券 200

贷:营销成本.通用优惠券 200

这种方式处理优惠券，是按照使用优惠券之前的价格，也就是100元来计算增值税，公司所得税会相应扣减，但处理起来也比较麻烦。还有一种简单的方式是直接在用户消费的时候记账：

借： 应收账款-工行收款 89.91

服务成本-工行手续费 0.09

贷： 主营业务收入-会员卡 90

可以在明细账中记录用户使用的优惠券。

#### 打折销售

折扣销售时，和普通销售一样处理，金额按照打折后的来计算。 老熊公司促销，8折优惠，小明用80块钱买了张会员卡，会计分录如下：

借： 应收账款-工行收款 80-0.08 = 79.92

服务成本-工行手续费 80\*0.1% = 0.08

贷： 主营业务收入-会员卡 80

#### 买赠

买赠和打折销售类似，需要按照比例来分摊。 老熊公司推出买一增一活动，买一张100元的会员卡，赠送一张价值50元的点播卡。会计分录如下：

借： 应收账款-工行收款 100-0.1 = 99.9

服务成本-工行手续费 100\*0.1% = 0.1

贷： 主营业务收入-会员卡 100\*100/(100+50) = 66.67

主营业务收入-点播卡 50\*100/(100+50) = 33.33

#### 满减

满减活动的处理和买赠类似，需要按比例来分摊。老熊公司推出满200减50的活动，小明买了一张100元会员卡，1张50元点播卡，1张50元游戏卡。也就是150元买了3张卡，每张卡的收入按比例来分摊，会计分录如下：

借： 应收账款-工行收款 200-0.2 = 199.8

服务成本-工行手续费 200\*0.1% = 0.2

贷： 主营业务收入-会员卡 100\*(200-50)/200 = 75

主营业务收入-点播卡 50\*(200-50)/200 = 37.5

主营业务收入-游戏卡 50\*(200-50)/200 = 37.5

#### 实物购买以及税务问题

实物购买涉及到送货费用、税费等。送货费用每个公司处理起来差异较大，这里不介绍。 在税费处理上，一般是在计算业务收入时扣除。 实物按照17%的增值税来扣减。如小明在老熊公司买了100块钱的卡通熊玩偶，记账如下：

借： 应收账款-工行收款 99.9

服务成本-工行手续费 0.1

贷： 主营业务收入-卡通熊 100/1.17 = 85.47元

应交税费--应交增值税 100/1.17\*17%=14.53元

#### 退换货

退货根据销售的不同阶段，采取相应的冲销。在货款到对公账户前退货，则对冲应收账款；如果是货款已经到公司银行账户了，则对冲银行存款。 换货可以按先退货再购买的方式来记账。

#### 充值与提现

参考[支付清结算之渠道侧处理](http://doc.cocolian.cn/essay/2017/01/15/clearing-channel/)一文的充值部分。 和支付类似，用户充值也分为两个步骤，第一步是执行充值操作，充值到公司对公账户上，但是资金并不是立即到账的，这个过程的会计分录如下；在第二天资金到账后，应收账款变为公司的存款，这个过程同支付。

借： 应收账款-工行收款 99.9

服务成本-工行手续费 0.1

贷： 客户账户-小明 100

提现流程和充值相反，会计分录如下：

借： 客户账户-小明 100

服务成本-工行手续费 0.1

贷： 应付账款-工行 100.1

### 收单记账

本文描述在电商支付系统的支付记账和会计处理流程。注意，这个流程和第三方支付平台以及银行等渠道的处理是不一样的。主要表现在：

电商支付系统的支付记账处理，不需要考虑备付金的情况。 用户的扣款是通过银行直接入账的， 不考虑担保交易的场景。

第三方平台支付系统的记账处理，支持担保交易。 用户交易资金是通过备付金来进行周转。  
这两个流程区分的关键在于是否有备付金。 本文考虑第一种情况，即无备付金，不需要支付牌照时的支付收单账户和账务的处理。

#### 一、支付流程

接着之前的小明购买会员卡的案例，不考虑优惠券和卡采购的情况。 为了完成支付，支付系统需要预先在支付渠道开通收单账户，用来接收业务方的的收单请求。 注意，有些第三方平台，如微信支付，对不同业务类型的收单费率是不一样的，会要求设置不同的商户号里完成。

在线上的实时处理流程如下：

用户购买会员卡，提交订单，会员卡向订单系统请求生成订单，订单系统向支付系统发出支付请求；

支付系统生成支付记录，并向银行发出扣款请求。

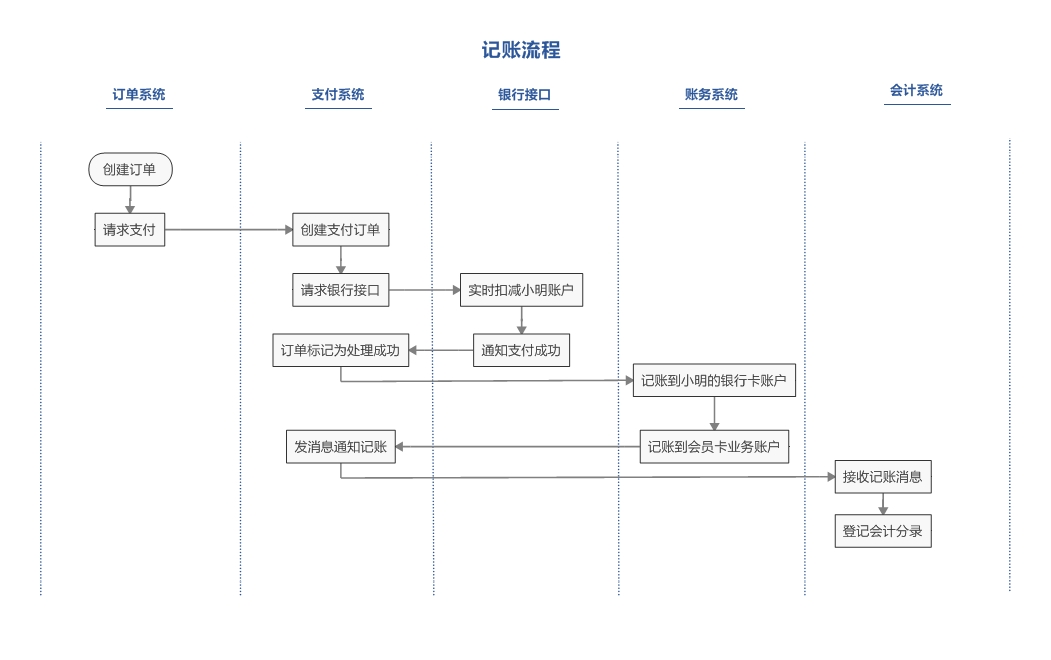
银行实时从小明的银行卡账户上扣款100元，通知支付系统小明支付成功；

支付系统账务子系统在自己的账户体系中记录小明的这一笔消费支出，给会员卡业务账户增加对应的资金，通知会员卡系统发送卡给小明。

支付系统发送消息异步通知会计系统进行记账。

注意， 在这里，渠道（银行）从小明账户上的扣款并不是直接到渠道的收单账户上。

这5个步骤都是线上的流程，在此过程中，各个子系统之间的交互如下图所示。 我们逐步分析这里涉及到的对象。



#### 二、账户设置

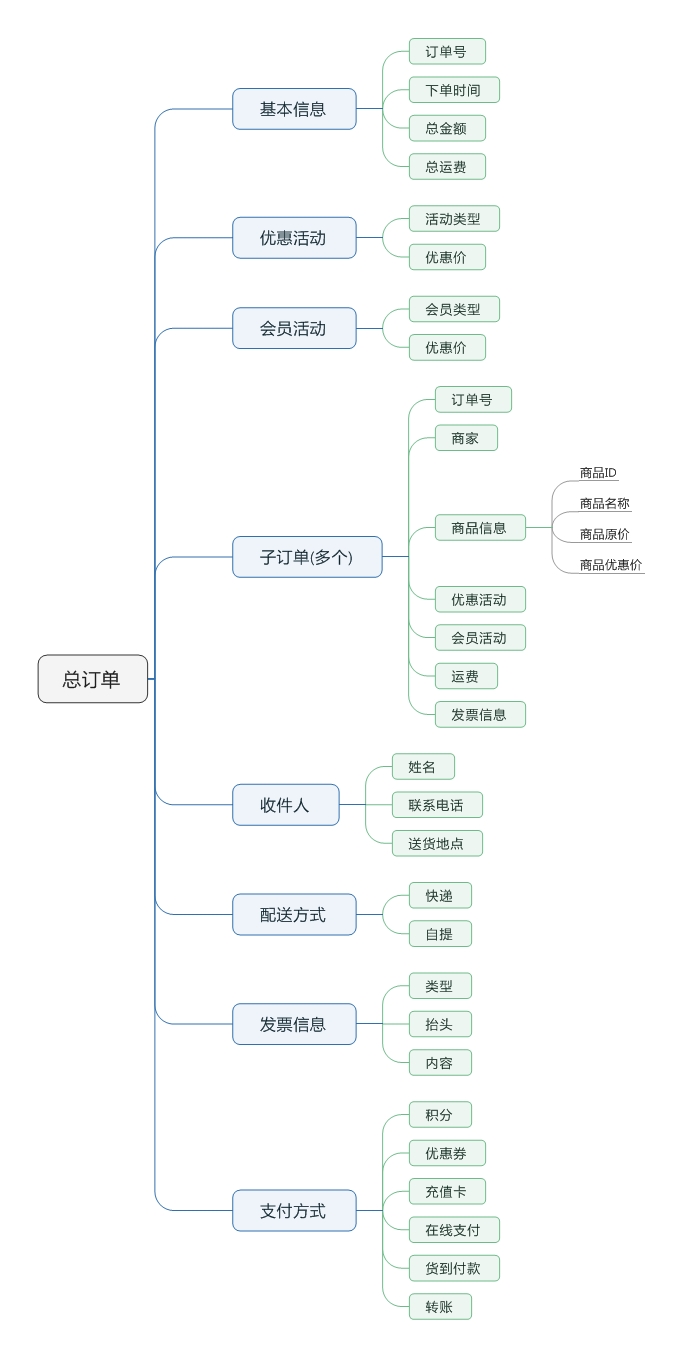
借： 应收账款-工行收款 100-100\*0.1% = 99.9

服务成本-工行手续费 100\*0.1% = 0.1

贷： 主营业务收入-会员卡 100

#### 二、支付订单

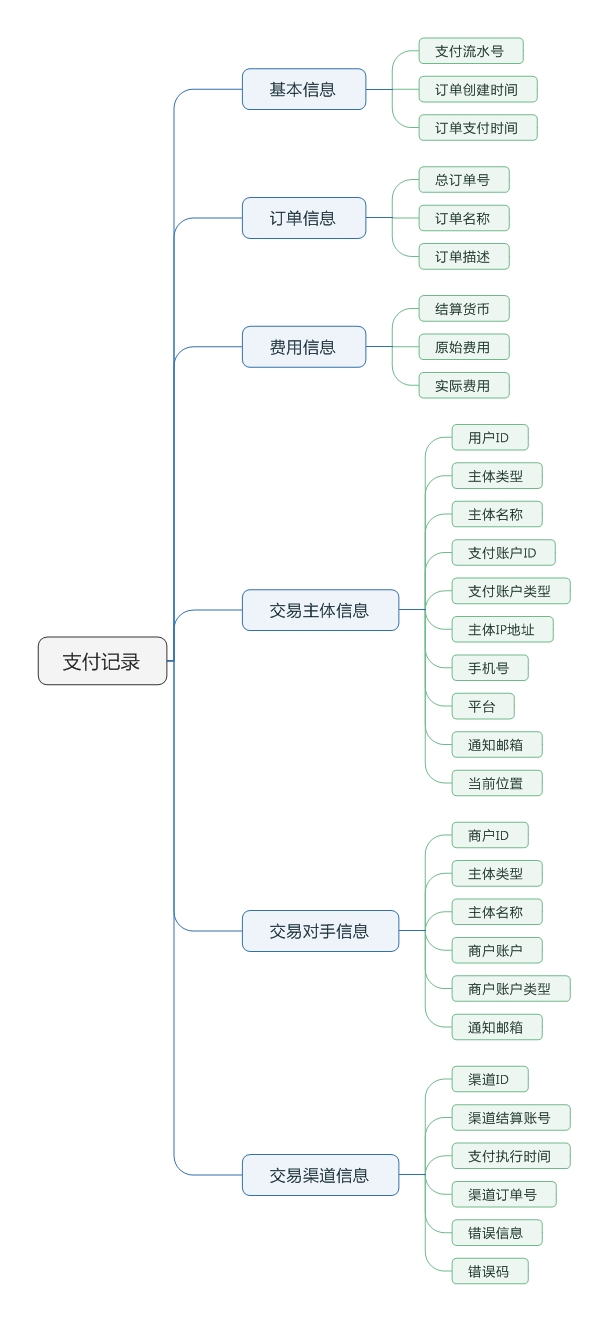
这个流程中，首先生成的是支付订单。这是一个比较简单的订单，仅涉及到一个商家和一个商品。 在比较复杂的电商场景中，一个订单会涉及到多个商家、多种商品以及对应的优惠活动。也就是，一个总订单会被拆分为多个子订单。这部分内容将在后续的订单系统设计一文中详细介绍。 而订单中和资金相关的内容，都需要在账户体系中建立对应的科目和账户。在请求支付时，只会将总订单提交支付，拆分子订单是在订单系统中完成的。



针对上述场景，为了简化处理，假定老熊公司当天总共完成了三笔会员卡交易，支付订单如下：

[](http://static.cocolian.cn/img/in-post/clearing-tb-orders.jpg)

#### 三、支付记录

在这个流程中，订单系统向支付系统请求支付时，支付系统将产生支付记录（支付订单）。 支付记录内容比较多，这些数据是后续进行记账的基础。[](http://static.cocolian.cn/img/in-post/clearing-record.jpeg)

上述的三个订单，两笔通过工行支付，一笔通过支付宝支付，其产生的支付记录如下，此处省略了其他和记账无关的字段内容：

[](http://static.cocolian.cn/img/in-post/clearing-tb-record.jpg)

这里需要注意的几个属性：

订单号: 这是总订单号。支付系统不再对订单进行拆分。 订单拆分是订单系统的功能。

支付流水号：在支付记录中，针对每个（总）订单号，会有对应的支付流水号。 如果用户使用组合支付，如上述场景，小明使用余额支付了20元，使用银行卡支付了剩余的80元，那这将产生2个支付流水号。为了避免洗钱风险，简化订单处理，包括淘宝在内，现在一般都不再提供组合支付的支持。

支付方式和支付渠道： 用户选择的支付方式和实际执行支付的渠道可以是不一样的。比如用户选择了广发信用卡来支付，但实际上电商公司没有直接对接农行，而是通过支付宝来对接， 那对应的支付渠道就是支付宝。

本条记录在支付成功后，会产生记账凭证和对账凭证。 这一条记录将产生多条会计分录，记账凭证是关联这些会计分录和支付记录的字段。对账凭证是根据对账周期来分配的。在日切后，对账凭证号相应的也会做更新。

在产生支付记录后，在上述流程的第5步通过消息机制来异步触发记账流程。账务系统接收到记账消息后，开始更新账户信息。记账分为两个阶段：

**支付记账**，针对线上的账户实时更新的需求，需要让用户及时看到账户余额和订单状态，账务信息记录到用户和商户上，采用单边账的形式。

**会计记账**，采用复式记帐法，满足会计记账需求，记录会计分录和余额，为对账和清结算提供支持。

#### 四、支付记账

支付记账是在支付流程中完成的，目的是让用户完成购买后，能够立即看到支付结果和账户余额。为了提升性能，支付记账一般采用单边账的形式，即将会计分录登记在用户侧或者商户侧。

在上述案例中，第6步处理，在服务器上与银行侧同步登记一笔从小明银行卡的支出，并在会员业务账户上登记一笔收入。 如果使用的是零钱支付，这一个步骤就很重要，从零钱账户上扣除费用计算余额，添加对应的消费记录，是在一个事务中完成。

#### 五、会计记账

会计记账采用复式记账，不同业务记账方式也不一样。小明买卡的案例中，需要记录的条目有：

在工行收款账户下，登记 99.9的借记 条目

在工行手续费的账户下，登记0.1元的借记条目

在主营业务收入-会员卡的账户下，登记100元的贷记条目。

这3条记录是通过事务处理一次生成。当天发生的三笔交易，产生的记账内容如下：

[](http://static.cocolian.cn/img/in-post/clearing-tb-accounting-2.jpg)

实际实现上，科目一列，使用账号ID来替代。 每个账户的本期发生额，可以在另一个表中单独异步计算。

### 充值记账

### 计息

### 风控系统

#### 支付风控场景分析

风控是一个让人爱恨交加的话题。 对支付来说风控是必不可少的功能。只要老板不想把底裤都赔掉，那就必须上风控。可对互联网公司来说，风控是一个谜一般的话题，无论是对风控专家还是IT工程师而言。随着互联网和大数据技术的引入，风控变成了一个跨学科的领域，可这无疑是互联网公司里面最同床异梦的跨学科。机器学习，深度学习，规则推理，随机森林….光这些名词就足以让人风控专家望而怯步；而风险事件、尽职调查、巴塞尔协议.. 这些名词，一提起来IT人员就头大。这个系列的文章将试图从这两个领域简单梳理下支付风控面临的问题，以及如何从技术角度来解决这些问题。

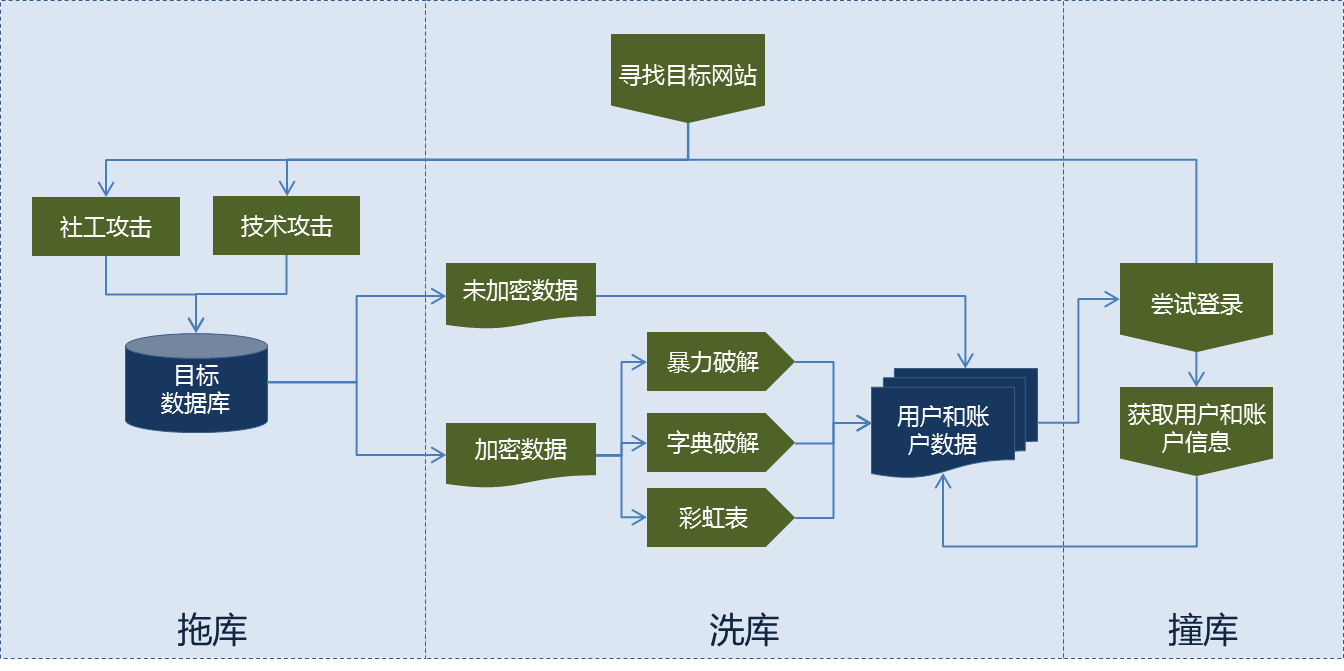
#### 概念定义

按照教科书的说法， 风险是指在特定场景下，特定时间内某个损失发生的可能性，或者说是在某一个特定时间段里，人们所期望达到的目标与实际出现的结果之间的差距。金融领域自从诞生以来，就一直伴随着风险。风险控制是指风险管理者采取各种措施和方法，消灭或减少风险事件发生的各种可能性，或风险控制者减少风险事件发生时造成的损失。这里又引入了一个词，风险事件， 它和风险因素经常容易混淆。 风险事件指造成风险的直接原因，风险因素则是间接原因。 如下雨天路滑导致发生车祸造成人员伤亡。 则车祸是人员伤亡的直接原因，是风险事件。而下雨天是间接原因，属于风险因素。

风控做不好，一个晚上2个亿就出去了。饶是该公司财大气粗，也扛不住几次折腾。一个漏洞搞垮一个小公司也是常有的事。对支付系统来说，安全是第一考虑的问题，特别是资金安全，这需要风控系统来保驾护航。那一般来说，支付系统会面临哪些风险？ 不同文献有不同的风险分类，本文试图从账户、资金、交易、操作、信用风险角度来详细描述。

#### 账户风险

支付系统最常见的，也是在黑产圈中最为成熟的，那就得算账户的风险，即俗话说的“盗号”。近几年来，各大型互联网网站的账户泄露事故层出不穷，携程，京东，CSDN等都中过招，每一次都能引起轩然大波。而在黑产圈，账号窃取都形成了一套完整成熟的产业链。



这是目前在黑产圈中账户攻击的主要流程，以下分析在这个过程中每个阶段的具体操作，为风险系统设计提供依据。

##### 拖库

拖库是实施账户攻击的第一步。考虑到大型网站一般防守比较严密，黑客一般选择从小型网站入手，入侵到一些防守薄弱或有漏洞的网站，将注册用户的资料窃取出来。常见手法包括：

**1. 利用操作系统和系统组件漏洞**

比如近年来杀伤力最大的漏洞之一[Heartlbleed](http://heartbleed.com/) 漏洞。这个漏洞，在2012年OpenSSL软件发布时带有这个bug，而正式公开时间是2014年。Heartbleed漏洞使得黑客有可能通过memory dump的手段来获取到服务器上接受的用户请求、密码、甚至是服务器的私钥。只要持续不断的攻击，任何被加载到内存中且不幸被加载到和OpenSSL在同一个区块内存中的数据，都会被黑客所获取。 这两年期间，有多少黑客使用了这个漏洞来窃取网站信息，就不得而知了。 由于此类漏洞的发现和修复往往有一定的时间差，这也给黑客利用漏洞窃取信息带来了便利。

**2.利用网站所使用的第三方组件漏洞**

如臭名昭著的Apache Struts系列漏洞。从2010年开始， 不断地有漏洞暴露出来， 这些漏洞直奔struts所使用的OGNL表达式，通过构建各种匪夷所思的表达式， 可以远程执行任意命令，包括访问根目录。由于SSH（Springframework + Apache Struts + Hibernate）架构入门简单、上手容易，再加上各种IT培训机构不遗余力的推广，在国内电商、银行、运营商网站上被大量使用。每次Apache Struts 漏洞的发布，都能够掀起一番血雨腥风。 而Apache组织对这些漏洞响应不及时，修复慢，更让这些机构雪上加霜。远离Apache Struts更是支付系统的基本要求。

**3. SQL注入攻击**

基本上所有网站都会用到数据库。而一些新手在写代码的时候，对用户输入数据不做验证或者验证不到位，就把这些数据直接通过拼接SQL语句写入到数据库中，这就很容易导致SQL注入攻击。 比如系统在判断用户名和密码是否正确时，会使用这个SQL语句来查询数据库：

SELECT 1 FROM users WHERE username = 'admin' AND password = 'guest'

攻击者可以尝试修改密码为 ‘ OR ‘a’ = ‘a ， 拼接成SQL语句：

SELECT 1 FROM users WHERE username = 'admin' AND password = 'guest' OR 'a'='a'

由此执行成功，获取管理后台的权限。

这三个是常见的攻击方式。 当然还有其他的方式，如木马，钓鱼网站等等，不再详细描述。

##### 洗库

在攻入服务器，获取到资料，特别是数据库的信息后， 需要对信息进行分析。 不是所有的信息都可以直接使用，部分信息，如密码，身份证等，一般都会加密存储。 通过暴力、字典或者彩虹表的方式来破解，获取到破解后的信息，就拿到用户名，密码等资料。

-**暴力破解**：如果知道用户名或者密码的范围，可以通过枚举的方法逐个尝试。 对密码来说，会受限于密码的长度，如果长度在8位以上，那可枚举项就太多了，需要几天甚至几年的计算才能破解。

-**字典表**：其实也是暴力破解的一种，区别是可以预先计算出来一些常见的组合，比如生日之类的，然后使用这些组合来进行破解。

-**彩虹表**：这是一种破解哈希算法的技术，是黑客必备的跨平台密码破解方法，可以破解MD5进行哈希处理的密码。它的性能优异，在一台普通PC上辅以NVidia CUDA技术，对于Microsoft Windows操作系统使用的NTLM密码加密算法，可以达到最高超过1千亿次每秒的明文尝试。对于广泛使用的MD5也接近一千亿次。

##### 撞库

第三步是撞库，就开始进攻真正的目标网站了。 把拿到的账户信息去尝试登陆大型网站。因为大部分用户，习惯于在多个网站使用同一套账户和密码。如果登录成功，则可以进一步窃取更多的用户信息，比如信用卡信息等。 由此可见， 撞库攻击本质上是利用用户相同的注册习惯，以大量的用户数据为基础，尝试登陆目标网站，从而窃取更多的用户资料。 这也使得黑客无需进行系统攻击的情况下，即可轻易获取目标用户信息。

更进一步，黑客们会把这些资料整理后，形成社工库。这个库也日益壮大， 目前有千万规模。除了用户名密码，还有大量的个人隐私也被挖掘出来。比如如家2000万数据泄露，其中包含开房信息；QQ群用户信息泄露、京东2015年初用户信息泄露。这都导致大量的个人隐私被窃取甚至出售。

由此可见，账户被窃取，往往是网站防护薄弱和用户安全意识薄弱两种因素导致。

#### 交易风险

支付的交易风险主要是交易过程中的各种恶意行为，而这些行为在电商系统中表现特别突出，包括 自动刷单、人工批量下单以及异常大额订单等场景。在秒杀的时候， 由于其价格有很大的优惠力度， 黄牛会采用机器批量注册账号、机器抢购等方式来争取秒杀商品，普通消费者很难享受到秒杀的实惠，使得秒杀活动效果大打折扣。此外，在商家侧，主要的风险在于刷单。不少商家使用刷单、刷评价的方式来以非正常途径提升销量，积分，信誉等。甚至通过刷单的方式来套取补贴，帮助套现。 从阿里公司发布的《互联网信任环境调查报告》来看，大部分用户在购买的时候，会看中商家的资质和诚信，商品的销量、评论也往往会成为购买的一个参考。 在这种情况下， 刷单就成为一个提升店铺交易量的重要手段。而刷单和反刷的猫鼠游戏，也推高了刷单识别的难度。 以电商为例，一般刷单行为有如下特征：

小号刷单。谁也不会用自己的注册账号来刷单，这样被封的代价就太大了。 小号的来源，可能是商家自己组织注册的，但大部分还是从专业刷单机构手中获取的。

使用虚拟机。大部分网站都会为访问设备植入识别码。通过虚拟机，可以在一个物理机上模拟多台机器访问，随用随建。一般使用VMWare来建立虚拟机。而对手机设备，则会采用手机模拟器。

使用VPN。 这样可以伪装使用全国任何一个地区的IP，甚至可以使用国外的VPN。

使用手机IP：移动和联通的IP出口少，所以大部分手机端的出口IP并不多。 这些IP是电商的白名单，把某个IP封了，那会有大量的手机无法正常访问。 所以刷单人员会选择使用这些IP。

刷虚拟物品：虚拟物品不涉及到物流环节，交易流程简单，很容易就可以把量刷上去。

低价刷单：为了降低成本，往往会将单品价格调低，或者成交金额调低来支持刷单。

交易商品少：刷单时，仅选择少量几个商品进行。

互刷： 一些商家会勾结起来，相互刷单。

这些是从刷单行为的角度来分析的结果。看来简单，可对支付系统来说，如何交易记录中识别出小号、互刷、低价等这些特征，都需要使用大量的数据进行分析才能搞定。

#### 资金风险

2016年11月份的时候， 网上突然出现了大量怀疑支付宝沉淀资金用途的帖子，这些帖子在有意无意地引导一个观点： 支付宝将沉淀资金用于恒生HOMS系统的场外配资，用户将资金投放到余额宝有巨大资金风险。 毫无疑问，从监管的角度来看，这是不可能的事情。 但这谣传也揭示了支付系统的另一个风险：资金风险。 发展沉淀资金成为支付系统，特别是第三方支付系统的一个公开的秘密。 沉淀资金主要有两种形式：

在途资金 ：指买卖双方在确认交易后，完成结算前尚未到达卖方账户的资金。在买方没有最终确认收货之前，资金暂时交由第三方支付进行保管。这样在买卖双 方从开始交易到最终完成货款两清的这段时间差内，这些存在于第三方支付平台内部的资金，被称为在途资金。

留存资金：对采用交易担保型账户的支付机构，客户需要开立虚拟账户来完 成交易。机构也会吸引客户进行充值操作，即留存一些资金用于交易。比如微信支付和支付宝的钱包。 当有交易需求时，可以直接从这里进行扣款。这些留存于虚拟账户中的资金也是沉淀资金的一部分。

沉淀资金对支付来说是必要的， 通过这个资金来帮助买卖双方解决信任的问题，有利于提升用户体验。但这个资金也带来不少风险。2013 年 央行出台了《支付机构客户备付金存管办法》，其中明确要求第三方支付机 构对于客户的备付金要进行严格的区分管理，这一定程度上限制了沉淀资金风险的发生。也就是说，沉淀资金是客户的钱，支付公司不能挪用。支付公司可以获得沉淀资金的利息收益，但是不能够用这个资金来进行投资或者公司内部的消费。对这笔资金进行合理监控避免出现风险，也是支付系统需要考虑的问题。

#### 套现风险

我国法律明确禁止使用信用卡套现，使用信用卡套现是违法的。但是在线支付系统中，使用信用卡进行套现，几乎是不需要成本的。 信用卡套现的手段也很多，一般是通过客户和商家的勾结来完成，比如：

虚假购买，客户通过信用卡购买某商品后，商品并未实际发货，商家将购买的款项打回给客户，完成套现。

退货套现：或者通过信用卡来购买商品，然后退货，将退款返回到借记卡或者其他可提现的渠道，也能完成套现。

自买自卖：商家通过信用卡购买自己的商品，将货款打入到借记卡中，完成套现。

上述的套现手段，很难识别。套现很难完全杜绝，除了要求退款资金必须原路返回外，还可以通过数据分析手段来减少发生的频率。

#### 操作风险

按照巴塞尔委员会《操作风险管理》的定义， 操作风险主要是指那些由于用户支付终端操作失误、工作人员违规操作、内控机制失灵等人员操作上的原因引致损失的风险，或者说是外部风险、员工风险和流程风险。

流程风险指由公司的规章制度管理、业务流程不完善而引发的风险。对一些支付公司而言，作为新兴的经济形式，不像银行那样有一套成熟、规范的流程以及完善的培训机制，这就容易触发流程风险。在以“快”为特征的互联网公司，功能创新非常重要，但往往也容易忽视了风险管理相关配套制度的建设和落实，从而为线上运行的新功能带来隐患。当新的支付方式上线后，配套的清结算、记账、对账等功能，未必能够及时地跟上，更不用说相关的内控制度建设、岗位人员配备的工作。

员工风险指的是支付机构的员工不遵守职业道德，违法违规或违章操作，单独或参与骗取、盗用机构资产和客户资金，工作疏忽等行为导致的损失。在缺乏成熟培训机制的互联网公司中，这类问题往往更加突出。

欺诈行为：员工同外部人员相勾结，通过挪用资金、职务侵占等方式非法占有公司财产或者泄露出卖公司商业秘密的行为。

越权行为：员工未经授权、或超越工作权限导致的损失，比如开发人员私自修改数据库给人送优惠券。

错误操作：员工在具体业务操作过程中的失误造成的错误操作。

#### 合规风险

合规风险指机构因未能遵守相关的法律法规从而导致机构可能受到处罚、声誉受损的风险。从2004年的电子签名法开始，和支付相关的法律法规：

| **发布时间** | **发布机构** | **法律法规** |
| --- | --- | --- |
| 2004.8 | 商务部 | 《中华人民共和国电子签名法》 |
| 2005.4 | 中国电子商务协会 | 《网上交易平台自律规范》 |
| 2005.6 | 中国人民银行 | 《支付清算组织管理办法》 |
| 2005.10 | 中国人民银行 | 《电子支付指引（第一号）》 |
| 2007.3 | 商务部 | 《关于网上交易的指导意见（暂行）》 |
| 2008.4 | 商务部 | 《电子商务模式规范》《网络购物服务规范》 |
| 2009.11 | 商务部 | 《关于加快流通领域电子商务发展的意见》 |
| 2010.9 | 中国人民银行 | 《非金融机构支付服务管理办法实施细则》（征求意见稿） |
| 2011.6 | 中国人民银行 | 《关于规范商业预付卡管理的意见》 |
| 2011.10 | 中国人民银行 | 《支付机构预付卡管理办法》（征求意见稿） |
| 2011.11 | 中国人民银行 | 《支付就够互联网支付业务管理办法》（征求意见稿） |
| 2012.1 | 中国人民银行 | 《支付机构互联网支付业务管理办法》（征求意见稿） |
| 2012.3 | 中国人民银行 | 《支付机构反洗钱和反恐怖融资管理办法》 |
| 2012.6 | 中国人民银行 | 《银行卡收单业务管理办法》（征求意见稿） |
| 2013.6 | 中国人民银行 | 《支付机构客户备付金管存办法》（征求意见稿） |

其中2010年的《非金融机构支付服务管理办法实施细则》是一个标志性的法规，标识国家开始认可第三方支付的地位并开始执行监管。 之后，央行又陆续出台一系列法规来规范支付行业的发展。可以说，支付行业的业务创新，是一个不断地由乱而治的过程。而对支付公司来说，滞后的法规建设，也给业务发展带来了巨大的风险。 2013年支付宝推出互联网理财产品余额宝，在短期内迅速发展成为国内最大的基金。随后多家支付机构也开始开展这个业务，后续央行出台了《支付机构网络支付业务管理办法》，对支付公司的业务范围、资金转移金额进行限制，避免了该业务的过度发展。 2014年央行也相继叫停了虚拟信用卡、二维码支付等业务。合规风险是国内第三方公司一个无法规避的风险，在企业发展过程中，需要密切关注央行的动向，减少合规带来的负面影响。

#### 洗钱风险

第三方支付目前成为洗钱的重灾区。 2016年8月，18家支付机构被公安部列为重点整改对象。 这些支付机构提供的服务，存在未落实实名制、风控措施不严格等问题，被犯罪分子所利用，沦为咋骗和洗钱的工具。主要手段包括：通过一些第三方支付平台发行的商户POS机虚构交易套现；将诈骗得手的资金转移到第三方支付平台账户，在线购买游戏点卡、比特币、手机充值卡等物品，再转卖套现；利用第三方支付平台转账功能，将赃款在银行账户和第三方支付平台之间多次切换，使得公安机关无法及时查询资金流向，逃避打击。2012年央行发布的《支付机构反洗钱和反恐怖融资管理办法》，对支付机构如何防范洗钱风险做了明确的规范和要求，需要支付公司严格遵守。

以上是支付系统可能面临的风险分析。支付风控系统是通过采集交易、渠道、商品、账户、用户等信息，对这些数据进行实时和定时的挖掘分析，识别出各种风险，采取各种措施降低损失。这是支付风控系列的第一篇文章，这个系列将包括如下内容：

支付风控场景分析（本文）；

支付风控数据仓库建设；

支付风控模型和流程分析；

支付风控系统架构

### 支付风控数据仓库建设

支付风控系统在数据存储设计上和其它业务不同的地方在于数据获取与使用的流程。一般业务系统会先确定系统数据需求，再设计如何在业务流程中采集数据，以及数据的格式怎么定义。而支付风控面临的是一个无法预知的场景，需要在实践中根据当前运行情况不断调整。它会先把数据采集过来，之后才能从中发现可能存在的问题，并针对该问题制订风控规则。也就是风控是先采集数据，再使用数据。

风控分析不仅要看交易数据，还得研究所有相关联的数据，这才能全面分析出来风险的根源，推断出需要采取的措施。因而数据采集工作对风控系统建设和演化是非常重要的。本文分析风控所需要的数据，如何采集和存储数据，建立支持风控的数据仓库。

#### 一、数据来源

一笔交易的风险等级的计算需要考虑到多个维度。未成年人购买高档酒、促销期间羊毛客刷单、在洗钱高发地区的商户销售的物品成交价格远超实际价格。这些可疑交易的识别，仅依靠支付系统本身是无法完成的。用户的年龄、商品特点(是否高档酒)、是否促销、羊毛号的识别等，需要从各业务系统，甚至公司外部收集和用户、商品、商家、地区、手机号相关的数据，通过对这些数据进行分析，提取特征，识别潜在的风险。

##### 1.1 内部数据

风控几乎需要收集所有相关系统的数据。

用户系统：采集用户的静态信息，姓名、性别、年龄等。风控系统不仅仅关注这些静态信息，还需要重点关注用户的行为信息，包括注册、密码修改、修改个人信息等操作，需要收集这些操作的时间、地点、设备等信息。 此外，用户之间的关系，也是风控系统需要关注的数据。

商户系统：除了采集机构的基本信息，如成立时间、注册时间、人员规模、营业额、销售额、经营范围、注册地点等， 还需要考虑到该商户关联的用户，包括法人代表、公司组织结构、主要员工信息等。

商品系统：商品的静态信息，包括类型、价格、上架时间、库存等信息； 商品的浏览、放入购物车、购买、评论、退货等用户操作，包括这些操作的时间、地点、设备等信息。

社交数据，包括评论、论坛、留言等。

业务系统，如视频系统中的观影记录、类型偏好、时间、地点、设备等信息。

当然，支付数据是风控最重要基础数据。用户在支付系统中涉及到的数据都需要收集整理来支持风控分析。包括但不限于：账户数据，订单数据，交易数据、优惠券数据、账务流水等。这些数据在支付数据库中也存在，风控所需要的数据和业务数据略有不同，除了业务数据外，风控还关心如下数据：

用户当前上下文环境，包括用户所用设备的类型、操作系统、IP地址、设备ID、所在地等，而这些数据往往并不是业务所关心的。而且记录太多的上下文数据也影响性能。

账户，订单等操作实体的状态。在业务数据库中一般仅保留实体的最终状态，比如账户是否已锁定、订单是否已支付等。 而风控需要关心这些状态变更的时机，以及变更的时间间隔。例如，用户频繁更改交易密码，超正常频率提交订单等，就不是一个正常的状态。

这些数据一般可以从日志中采集。

##### 1.2 外部数据

对于大部分业务单一、用户量不大的公司来说，其数据有限而且单一，需要使用外部数据来辅助完成风控计算。 常用的外部数据包括：

公安部的实名认证数据，包括用户姓名、身份证号信息；

央行发布的各种名单，如洗钱区域，恐怖组织名单等。

央行信用报告，这个查询可是要真金白银的。

微博数据，一个人经常了解如何养卡，套现等内容并不是太好的事情。

工商局提供的公司信息。

招聘网站上的公司招聘信息。公司一直有招聘说明业务还不错。

芝麻信用，这个需要申请。

#### 二、采集方式

一般来说，风控的非实时数据采集，不能直接从线上的数据库中读取，这会把数据库打死。主要的数据采集方式有从库采集，日志采集和pingback三种方式。

##### 2.1 数据库从库

主流数据库，如Hbase，Mysql都提供同步数据进从库的功能，读取从库不会影响主库操作。但如上所述，采用从库有如下问题：

分析所需数据和业务数据不同，还需要从其他途径补充数据。

将风控所需数据和业务数据紧耦合起来了。一旦业务有变更，风控系统也需要调整。

##### 2.2 日志

这是风控数据采集的主要方式。 业务方可以将风控所需要的数据输出到日志中，风控系统对接日志来异步采集数据。这使得数据采集不会影响业务处理主流程。 这种方式风险在于：

需要规范日志的格式，否则每个系统一套日志格式，会导致对接工作量巨大。

保持日志的稳定性。一旦代码被修改，打印日志的代码被删除了，会导致日志数据无法采集的风险。

需要注意日志采集系统的可靠性。目前主流的采集框架都有可能会丢失日志。虽然从我们使用的情况来还未发生这种事情，但不排除这个风险。

从技术上来说，日志采集的框架主要框架有

ELK（Elastic + Logstash + Kibana）， Logstash 驻留在日志输出端采集日志，并发送到Elastic 服务器上。 Kibana则是一个日志分析的工具；

Flume + Kafka + Elastic 。 通过Flume进行采集，输出到Kafka，汇总到Elastic进行存储。日志分析可以在Elastic上离线非实时进行，也可以直接对接Kafka准实时分析，即流处理。 使用Storm 或者Spark都可以。

##### 2.3 pingback

Pingback指在页面上埋入脚本来监测用户的操作，特别是点击操作和键盘操作，将检测到的用户行为异步发送到服务器端。这可以侦测到用户在页面停留时间，鼠标点击的区域等信息，由此可以推断用户偏好，情绪等信息。 pingback的挑战在于如何在服务器端应对流量洪峰。pingback数据一般不直接入库，可以先写入Kafka，风控系统对接Kafka来分析pingback数据。

#### 三、数据特征

用于支持风控计算的最终数据，在静态与动态数据为基础计算出来的带置信度的推算数据为主的离散数据，有点绕口，我们详细分析下这里涉及到的几个概念，来说明最终用来支持风控计算的数据有什么特征。

##### 3.1 静态数据与动态数据

上述采集到的数据，大部分是静态数据。也就是这些数据一旦产生，一般不会被修改。但在分析时，还需要一些易变的动态数据来，比如用户的 年龄，每天的访问量，每天消费金额等。

##### 3.2 原始数据与推算数据

不管静态还是动态数据，他们都是从用户输入或者系统采集的方式产生。但我们知道，互联网的数据可靠性是有问题的。网上千娇百媚的姑娘，在现实中可能是一位抠脚大汉。虽然系统中设计了复杂的表格来收集用户信息，但会提供全部信息的用户还是很少，大家对隐私内容还是捂得很紧。所以，在进行风险计算前，还需要对数据进行验证和补充。这都需要借助其他数据来进行推算，这些数据被称为推算数据。推算数据和原始数据不同之处在于它会有多个可能取值，每个值都带有置信度。完全可信为100%，不可信为0。置信度总和为1。比如正常情况下，用户的性别要么男，要么女。假如有个用户注册时选择性别女，但经常买刮胡刀，衬衣，没有买过女性用品，那实际性别为男的置信度就非常高。

##### 3.3 离散数据与连续数据

这是从属性值的取值范围来评估。比如用户每天的订单额，一般来说是连续分布的。而性别，职业，爱好等，是离散值。一般来说，离散值更容易做分析处理，刻画特征，所以在分析前，需要对连续数值做离散化处理。

#### 四、名单数据

名单数据是支付风控数据仓库中最重要的内容。 风控系统数据仓库建设，也一般都从名单数据开始。 名单加上简单的拦截规则，已经可以解决绝大部分风控的问题。就算在更先进的风控系统中，名单仍然是风控中的基础数据。在评估事件风险时，名单往往是用来执行第一道拦截时所用的数据。比如用户交易时使用的手机是黑名单中的手机，则必须终止本次交易。

##### 4.1 黑白灰名单

大家都熟知黑名单与白名单，一个是必须阻止，一个是必须放行。 除此之外，还有灰名单。灰名单用于对一些高风险的用户进行监控。 这些用户的行为不是直接阻止，而是延迟交易，经人工确认无问题后再放行。

##### 4.2 更新周期

相对其它数据来说，名单数据的更新频率不高，按天、周、月更新都有，很少有需要实时更新的内容。对于手机号，证件号等名单，一般可以采取人工更新的策略。每天评估风控数据，对确认有问题的号码，加入到黑名单中。如果采用的是第三方名单，则需要按照第三方的要求对名单做更新。

##### 4.3 名单列表

一般来说，风控系统需要配置的名单列表有：

**个人名单**，如下名单是必备的(后续会及时更新)，

[央行的反洗钱恐怖分子名单](http://www.pbc.gov.cn/fanxiqianju/135153/135267/index.html)

[公安部的通缉犯名单](http://www.zhuatongji.com/gawtjfmd/)

[全国法院失信被执行人名单信息公布与查询](http://shixin.court.gov.cn/)

**IP名单**，没有权威的IP名单。这需要在运行中积累。建立IP名单需要注意如下事项：公司内部IP，合作伙伴IP可以列入白名单列表；手机运营商的IP也要做到白名单中，封一个IP等于封掉一大批手机号；代理服务器可以列入灰名单；访问量大的IP也可能大公司的外网IP，不能仅依赖访问量来识别黑IP。

**公司名单**，必备名单包括央行反洗钱制裁公司名单和工商局失信企业名单

**手机号名单**，这也没有权威数据，电信运营商也不会提供此类服务。支付宝正在推广这个服务，但还没有公开。黑名单数据需要自主收集。

**地域名单**，央行公布的联合国反洗钱地区名单是必须在风控时考虑的名单，其他地域名单也需要自主收集。

**协查名单**， 公检法协查名单，接收到协查请求后，将人员全部信息拉黑。

##### 4.4 名单数据存储

名单数据在使用上的特点：

使用频率高，实时性要求高。各种名单匹配基本都需要在线上做实时计算。

数据粒度小，总量大小不一，但存储空间需求都不高。大部分名单都是一些号码表，几个G的空间都能存储。

更新频率低。名单数据一般都比较稳定，按天更新

在使用中，名单数据一般直接存储在内存中，或者使用内存数据库（Redis，Couchbase）。关系型数据库可以用来保存名单数据，但不会直接被线上应用所访问，它无法满足高访问量的需求。

#### 五、画像数据

名单数据能够快速发现用户在某个维度上的异常行为。在实际使用中，存在过于简单粗暴，一刀切的问题。比如如果限制单次购买金额为5000元，这个规则被试探出来后，攻击者会选择4999元来规避这个限制。画像技术则是尝试从多个维度来评估当前事件的风险。 比如画像刻画某用户平时主要在北京地区登录，购买习惯在10~300元之间。某一天突然发生一笔在东莞的4999元额度的消费，那这笔交易就非常可疑了。而这种交易通过规则比较难发现出来。 支付风控涉及的画像包括用户、设备、商品、地域、操作行为等。 这里重点介绍用户、设备和商品的画像。

##### 5.1 用户画像（persona）

用户画像是从用户的角度来刻画其背景和行为习惯，为判定某交易的风险等级提供支持。 用户画像的内容包括但不限于：

人口信息：一般就叫基本信息，主要包括：姓名、性别、出生日期、出生地、民族、星座等。

联系方式：家庭地址、工作地址、手机、固定电话、紧急联系人、QQ、微信号等。

资产特征：月工资、年收入、工资外收入、房产、车等

家庭特征：婚姻状况、是否有小孩、小孩关联、家庭成员等

交易偏好：交易频率（总计、年、月、日）、交易金额（总计、年、月、日）、常用账户、交易时间偏好、交易地点偏好、交易所使用设备、交易物品、交易物品所属类别等。

行为特征，这是和业务相关的特征。比如对于电商，关注 用户浏览的物品、浏览的物品类别、购买的物品等。而对于视频网站，则关注用户查看的视频、观影时长、类别偏好、观影地点偏好等信息。

对于已登录用户，可以使用用户ID来识别并做画像，但对未登录用户，系统需要通过设备来识别。

##### 5.2 设备画像

一个用户配备多台智能设备已经是很常见的事情了。手机，PAD，笔记本，台式机，都是常用的设备。用户在不同的设备上的行为往往是不一样的。有人偏好在电脑上寻找要购买的商品，却最终使用手机来下单，因为手机支付更便捷。 对设备进行画像，和用户画像类似，实际上是刻画使用设备的用户的特征。 此外，对于未登录用户，由于无法标识，也只能通过设备来代表这个用户。设备画像关注如下信息：

设备信息，包括设备类型、型号、屏幕大小、内存大小、CPU类型、购买时间、购买时价格、现在价格等。

交易偏好，同用户画像；

行为特征，同用户画像。

对设备画像来说，生成一个能唯一识别该设备的标识，即设备指纹，是数据采集中的一个挑战。设备指纹具有如下特点

唯一性，每台机器的指纹都不同，不能重复。

一致性，机器指纹在一台机器上是唯一的，不同应用，不同登录用户中取到的指纹都是一样的。

稳定性，指纹不会随时间变更，不会由于外围设备变更而变更。重装应用，重装操作系统也应该保持不变。

我们将在专门的主题中介绍如何生成设备指纹。

##### 5.3 商品画像

商品画像是从商品的角度来刻画购买或者拥有该商品的人的特性。

基本特征：名称，价格，类别，是否虚拟资产，上架时间，下架时间等

促销信息：价格，开始时间，截止时间

购买者特征：偏离这个特征越多，风险越大。购买时间分布，地点分布，价格分布，数量分布，年龄分布，性别分布等。

##### 5.4 画像数据存储

画像数据有如下特点：

数据粒度大。一个用户的画像数据，成百上千个维度都正常。

大部分数据都是推算数据，也就是数据格式是带置信度的，比如 {性别： 男，80%；女，20%};

每个维度的数据一般最终都需要离散化，比如年龄，虽然0~150的取值区间还不算稀疏，一般还会将年龄再分段。

数据量大。考虑到匿名用户和设备，上千万规模的注册用户，匿名用户和设备会在数十亿规模的量级。

数据结构不稳定。 根据业务需要会频繁添加新的数据维度，甚至添加新实体进来。

数据更新频繁。采用推算数据，每天不仅仅要计算新增数据，也需要重新计算现有数据的维度权重。

数据访问频率高。交易时计算权重，也需要使用画像数据。

很难有一个数据库能够同时满足上述的需求。画像数据存储需要综合采用多种数据库来满足不同应用上的需求。

数据写入库， 需要支持数据批量、快速地写入，Hbase是个不错的选择。

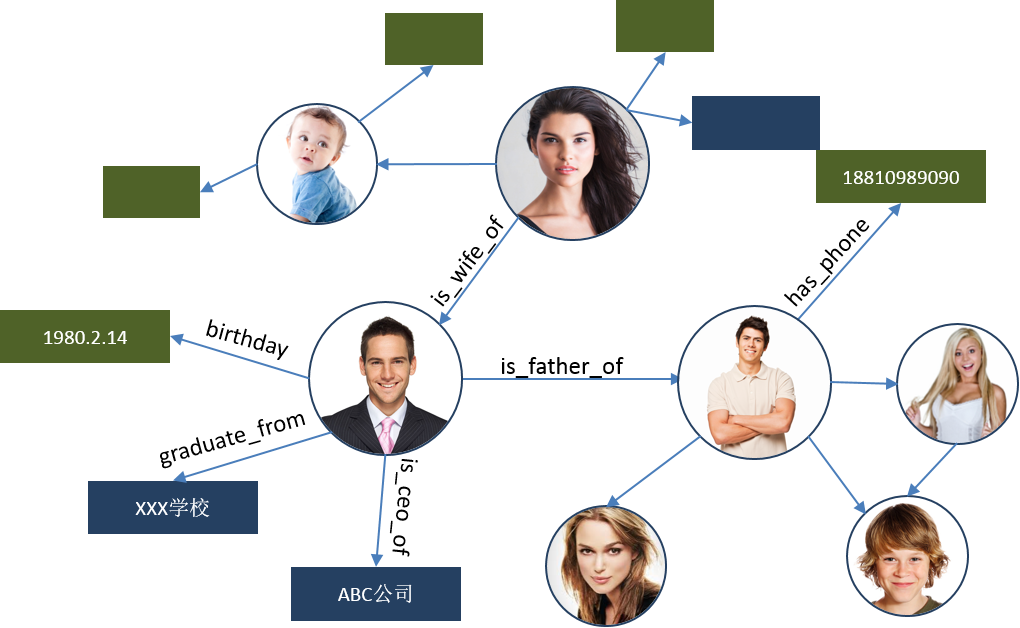
数据读取库，需要支持数据高速读取， couchbase可以满足这个需求。但couchbase不能存储所有数据，这样成本太高。 可以把couchbase作为HBase的缓存来使用。

写库和读库之间的数据同步。可以根据业务量选取合适的消息队列。每天更新的数据规模在百万及其以下，ActiveMQ可以满足需求；而上千万的数据，则需要使用Kafka。

#### 六、知识图谱

画像是从群体和个体的统计角度评估事件的风险，而图谱则更进一步，从关系的角度来评估风险。 知识图谱是由Google提出来并应用到搜索引擎上，其后在多个领域都得到很好的应用。 交易是一种社会行为，所以从关系的角度来评估这个行为，能够更精确的了解行为中存在的风险。一个简单的例子，如果发现A是高风险的用户，而通过社交图谱分析，发现A经常和B有交易关系， 那B的风险等级也相应地会被调高。

图谱在本质上是一个语义网络， 是一种基于图的数据结构， 它由点和边组成的。点代表一个实体，如人、公司、电话、商品、地址等，边代表实体之间的关系。



如上所示， 如果A和B两人之间是夫妻关系， 则在图中， A和B分别被用一个节点来标识， 称为实体，他们的关系是 is\_wife\_of。对电话、出生日期、出生地点、公司等，也可以使用这种方式来表示。 图谱的表达能力，不仅在于描述实体之间的关系，而且通过关系还可以推理出潜在的进一步关系。 比如A是B的母亲， A是C的妻子， 则有很大的概率可以推断出来C是B的父亲。 支付风控需要像建立画像一样建立图谱，需要支持包括人，机构，地区，日期，电话，手机号，设备，商品等实体，以及实体之间的关系。图谱数据源也是和画像一样。此外，还有一些互联网数据也有利于建立图谱 百度百科，有很不错的公司，明星，电影，音乐等信息，一般仅限于国内或者中文版本的资料。由于编审并不严谨，数据质量不高。 wiki，有各种语言的版本，提供各种领域的实体，参与的专业人士多，质量较高。 各专业数据库，

知识图谱是基于图的数据结构，它的存储主要是使用图数据库。关系型数据库和Hbase等nosql数据库在处理图的关系以及关系计算上性能较差，需要专用的图数据库，当前主要的图数据库有neo4j，Titan,Jena等。neo4j是使用最多的图数据库，而且可以和spark graph集成，方便对图谱数据做处理。

### 支付风控模型和流程分析

支付风控涉及到多方面的内容，包括反洗钱、反欺诈、客户风险等级分类管理等。 其中最核心的功能在于对实时交易进行风险评估，或者说是欺诈检测。如果这个交易的风险太高，则会执行拦截。由于反欺诈检测是在交易时实时进行的，在要求不能误拦截的同时，还有用户体验上的要求，即不能占用太多时间，一般要求风控操作必须控制在100ms以内，对于交易量大的业务，10ms甚至更低的性能要求都是必须的。 这就需要对风控模型进行合理的设计。一般来说，要提升风控的拦截效率，就需要考虑更多的维度，但这也会带来计算性能的下降。在效率和性能之间需要进行平衡。

本文重在介绍建立风控模型的方法，每个公司应该根据自己的实际业务情况和开发能力来选择合适的模型。这里列出来的模型仅为了说明问题，提供参考。

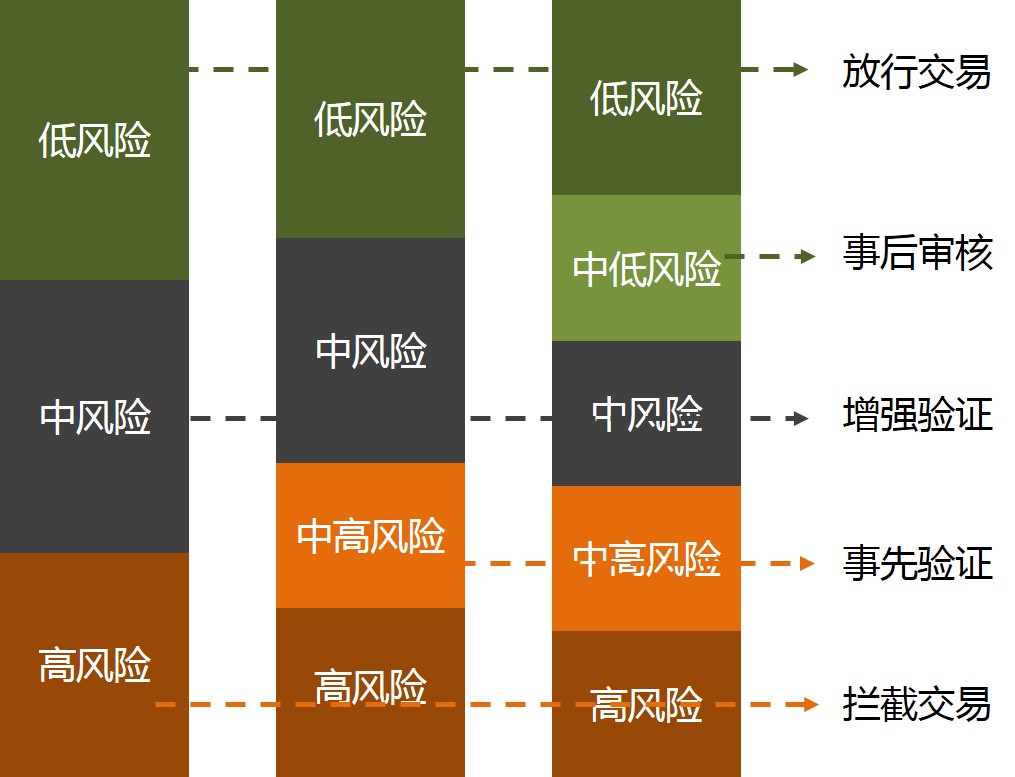
#### 一、风险等级

做风控拦截，首先要回答的问题是风险等级怎么划分？ 目前主流的风险等级划分有三种方式， 三等级、四等级、五等级。

三等级的风险分为 低风险、中风险和高风险。 大部分交易是低风险的，不需要拦截直接放行。 中风险的交易是需要进行增强验证，确认是本人操作后放行。 高风险的交易则直接拦截。

四风险等级，会增加一个中高风险等级。此类交易在用户完成增强验证后，还需要管理人员人工核实，核实没问题后，交易才能放行。

五风险等级，会增加一个中低风险等级。此类交易是先放行，但是管理人员需要进行事后核实。 如果核实有问题，通过人工方式执行退款，或者提升该用户的风险等级。



大部分支付系统是使用三等级的风险。

#### 二、基于规则的风控

规则是最常用的，也是相对来说比较容易上手的风控模型。从现实情况中总结出一些经验，结合名单数据，制定风控规则，简单，有效。 常见的规则有：

1. 名单规则  
使用白名单或者黑名单来设置规则。具体名单如上文所述，包括用户ID、IP地址、设备ID、地区、公检法协查等。 比如：

用户ID是在风控黑名单中。

用户身份证号在反洗钱黑名单中。

用户身份证号在公检法协查名单中。

用户所使用的手机号在羊毛号名单列表中。

转账用户所在地区是联合国反洗钱风险警示地区。

2. 操作规则  
对支付、提现、充值的频率按照用户账号、IP、设备等进行限制，一旦超出阈值，则提升风控等级。

频率需综合考虑（五）分钟、（一）小时、（一）天、（一）周等维度的数据。由于一般计算频率是按照自然时间段来进行的，所以如果用户的操作是跨时间段的，则会出现频率限制失效的情况。 当然，比较复杂的可以用滑窗来做。

对不同的风险等级设置不同的阈值。 比如：

用户提现频次5分钟不能超过2次， 一小时不能超过5次，一天不能超过10次。

用户提现额度一天不能超过1万。

用户支付频次5分钟不能超过2次，一小时不能超过10次，一天不能超过100次。

3. 业务规则   
和特定各业务相关的一些规则，比如：

同一个人绑定银行卡张数超过10张。

同一张银行卡被超过5个人绑定。

同一个手机号被5个人绑定。

一个周内手机号变更超过4次。

同一个对私银行卡接受转账次数一分钟超过5次。

4. 行为异常  
用户行为和以前的表现不一致，比如：

用户支付地点与常用登录地点不一致

用户支付使用个IP与常用IP地址不一致

用户在短时间内，上一次支付的地址和本次支付的地址距离非常远。 比如2分钟前在中国支付的，2分钟后跑到美国去支付了。

5. 风控拦截历史规则  
用户在某个业务上的消费行为被风控网关多次拦截。

规则引擎优点：

性能高： 对订单按照规则进行匹配，输出结果。一般不会涉及到复杂的计算。

易于理解和分析： 交易被拦截到底是触犯了那条规则，很容易输出。

开发相对简单。

规则引擎存在的问题：

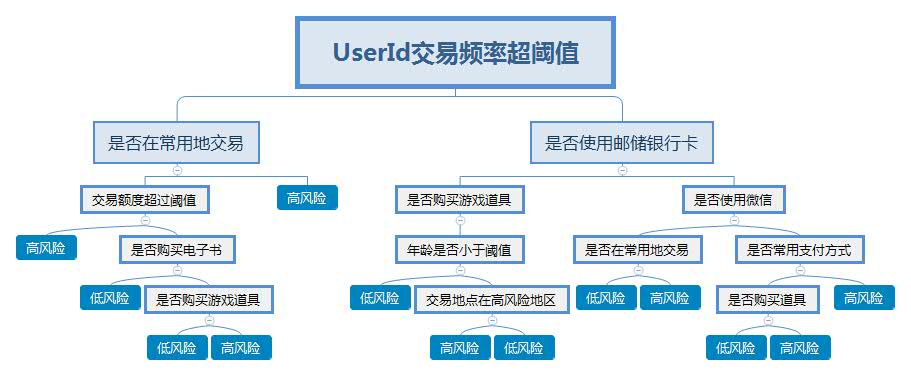
一刀切，容易被薅羊毛的人嗅探到。比如规则规定超过5000元就进行拦截，那羊毛号会把订单拆分成4999元来做。 一天限制10笔，那就薅到9笔就停手了。

规则冲突问题。当一笔交易命中IP白名单和额度黑名单的时候应该如何处理？

规则引擎看起来简单，但也是最实用的一类模型。 它是其它风控模型的基础。实践中，首先使用已知的规则来发现存在问题的交易，人工识别交易的风险等级后，把这些交易作为其它有监督学习的训练数据集。

#### 三、决策树模型

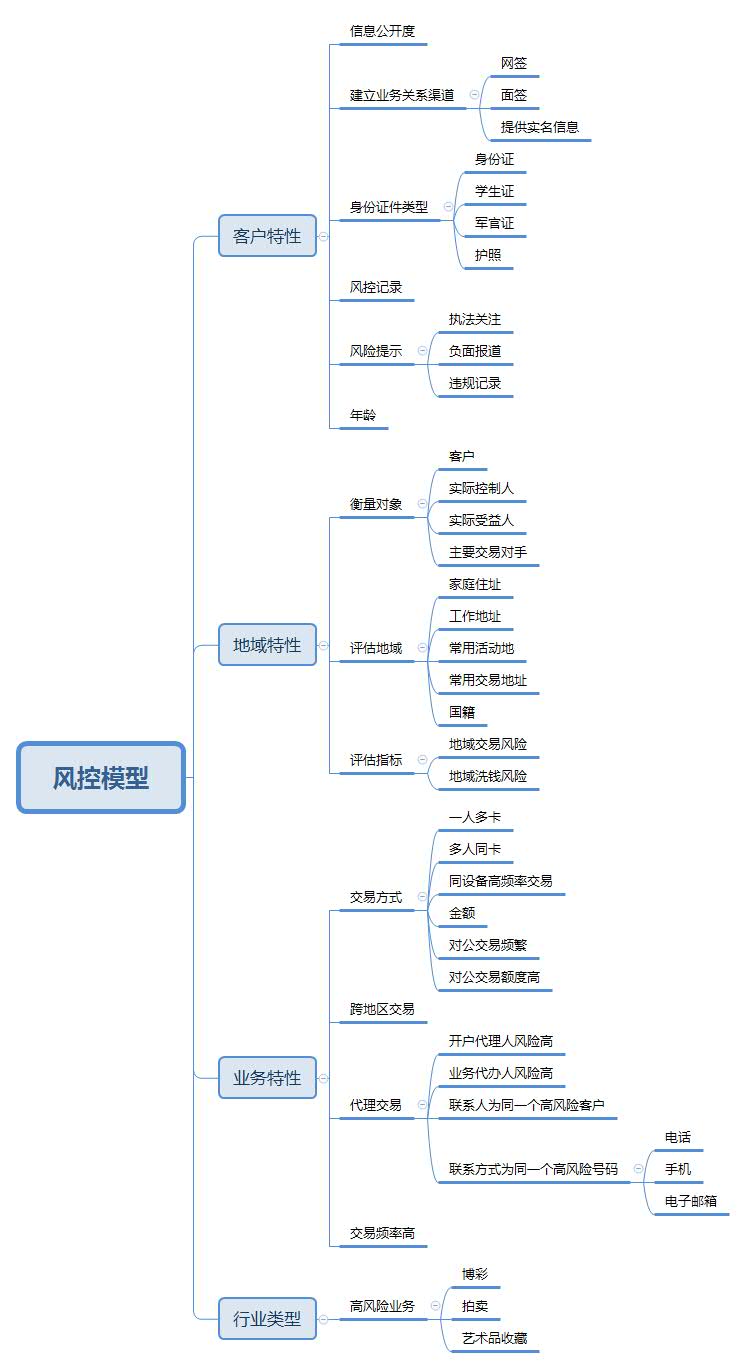
风险评估从本质上来说是一个数据分类问题。 和传统的金融行业风险评估不一样的地方，在于数据规模大、业务变化快、实时要求高。一旦有漏洞被发现，会对公司造成巨大损失。 而机器学习是解决这些问题的利器。 互联网金融风控离不开机器学习，特别是支付风控。 在各种支付风控模型中，决策树模式是相对比较简单易用的模型。 如下的决策树模型，我们根据已有的数据，分析数据特征，构建出一颗决策树。当有一笔交易发生时，我们使用决策树来判断这笔交易是否是高风险交易。

这种模型的优点是非常容易理解，检测速度快。 因而也是现有机构中常用的模型之一。 风控模型存在的主要问题是其产生的结果比较粗略。同样的两个交易被判定为高风险，究竟哪种交易风险更高，决策树模型无法给出答案。

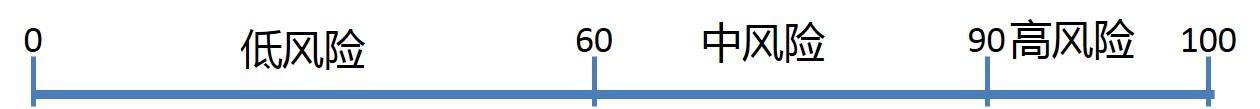
#### 四、评分模型

比决策树模型更进一步，现在也有不少公司在使用评分（卡）模型。 银行在处理信用风险评级、反洗钱风险等级时，往往也是使用这种方法。

每个公司的模型都不一样，一个参考模型如下：



该模型为参考《金融机构洗钱和恐怖融资风险评估及客户分类管理指引》编制，仅具参考意义。虽然银行间的评分模型有很好的参考价值，但互联网公司由于业务和数据的不同，评分模型参考价值不大。

每个公司需根据自己的业务情况来制定评分模型，之后为各个指标指定权重比例。 权重评分结果为0~100分的区间，之后按照区间划分，指定风险等级。比如：  


当然，评分区间也需要根据企业的实际情况来制定。 评分模型的优势在于：

性能比较高，针对交易进行指标计算，按照区间来确定风险。

相对于规则，如果指标设置合理，其覆盖度高， 不容易被嗅探到漏洞。

理解和分析也比较容易。 如果交易被拦截了，可以根据其各项打分评估其被拦截的原因。

存在的问题：

模型真的很难建立。指标的选择是一个挑战。

各个参数的调优是一个长期的过程。

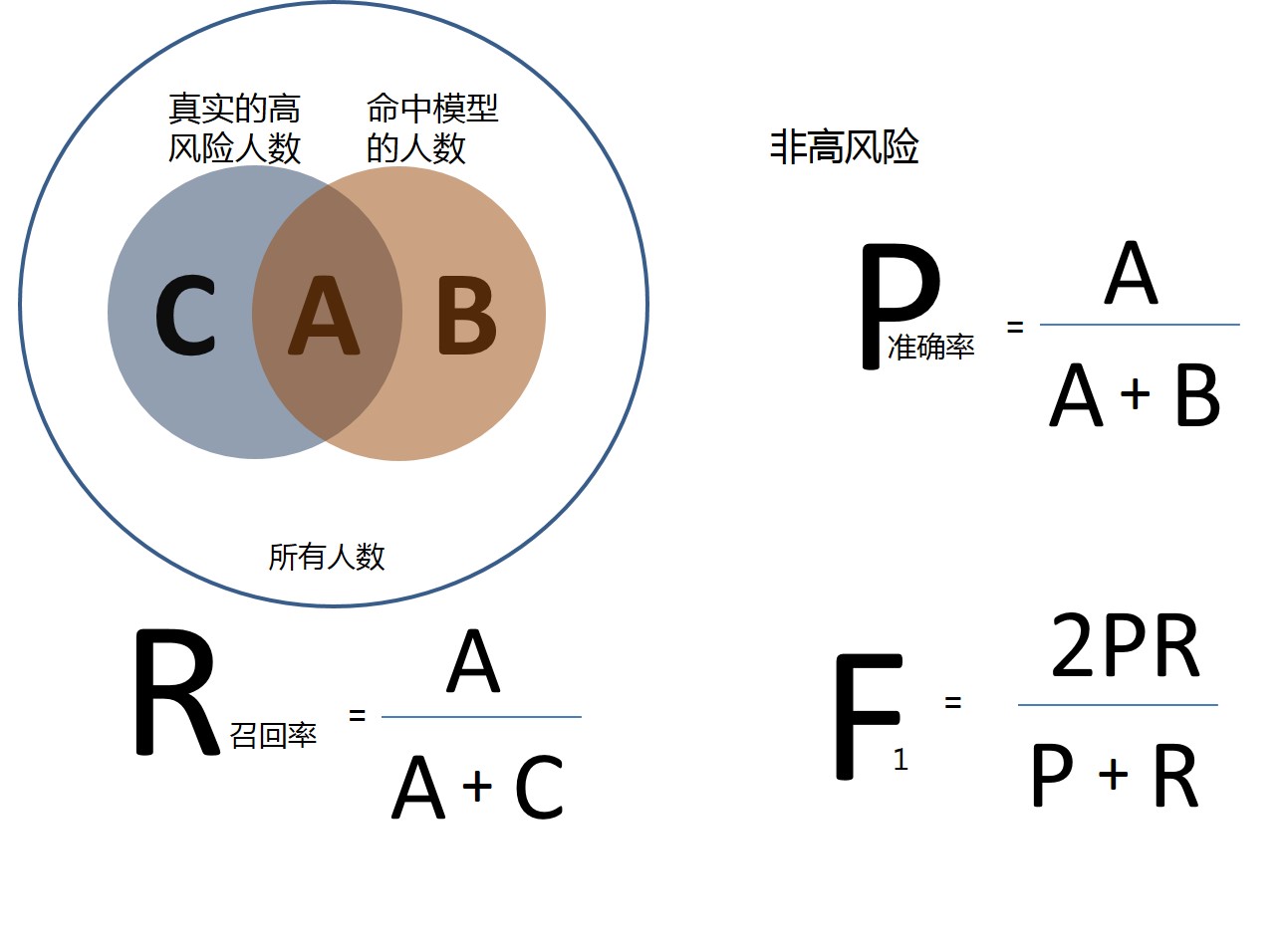
我们知道从一条交易记录中可以挖掘的关联数据有上百个，衍生数据就更多了。比如从支付地址，可以聚类出常用地址，衍生出当前地址和常用地址、上一次支付地址之间的距离，而这些指标在构建模型时都可能使用到。 所以第一个问题是，如何从这些指标中建立一个合适的模型？这就涉及到机器学习的问题了。 模型不能凭空建立，我们可以通过规则来对现有数据进行筛选和标注，确定这些记录集的风险等级。 这些数据作为样本来训练模型。可用的算法包括Apriori、FP-growth等。算法实现请参考相关文档。

在确认相关参数后，模型在使用过程中还需要不断对相关参数进行调整。这是一个拟合或者回归的算法，Logistic算法、CART算法，可以用来对参数做调优。

总之，模型的建立是一个不断学习、优化的过程。 而每一个模型的发布，还需要进行试运行，AB测试和上线。 这个过程，将在下一篇的风控架构中介绍。

#### 五、模型评估

风控本质上是对交易记录的一个分类，所以对风控模型的评估，除了性能外，还需要评估“查全率”和“查准率”。 如下图所示：



以评估高风险人群的效果为例，

Precision, 准确率，也叫查准率，指模型发现的真实的高风险人数占模型发现的所有高风险人数的比例。

Recall，召回率，也叫查全率，指模型发现的真实的高风险人数占全部真实的高风险人数的比例。

理想情况下，我们希望这两个指标都要高。实际上，往往是互斥的，准确率高、召回率就低，召回率低、准确率高。如果两者都低，那就是模型不靠谱了。 对于风控来说，需要在保证准确率的情况下，尽量提高召回率。 那怎么发现实际的高风险人数呢？ 这就需要借助规则模型，先过滤一遍，再从中人工遴选。

从实际应用情况来看，目前国内大部分团队使用Logistic回归+评分模型来做风控，少数人使用决策树。国外的PayPal是支付平台风控的标杆，国内前海征信、蚂蚁金服等会使用到更高级的神经网络和机器学习，但实际效果未见到实证材料。

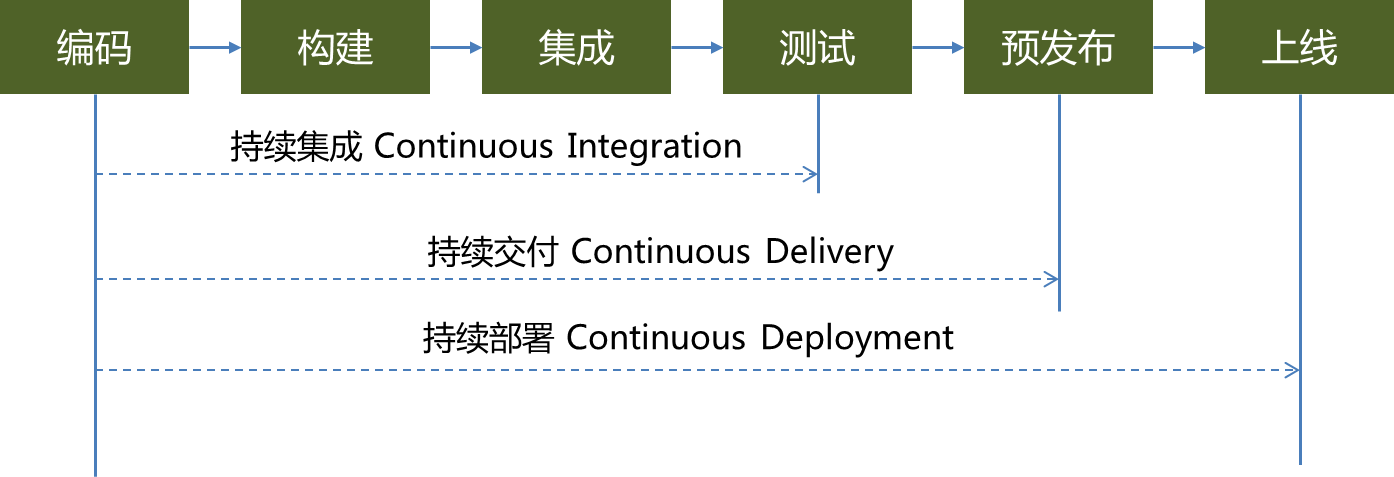
### 支付系统的基础设施建设

快速迭代，持续交付是互联网和微服务开发的核心理念。 这需要大量完善的基础设施的支撑。 对支付系统开发来说，这些基础设施尤为重要。近年来随着互联网公司在金融领域上的发力，竞争越来越激烈，新产品推出速度也在加快，而对安全的要求却是越来越高，不断的有系统或者业务的安全问题爆发。如何在快速完成业务支持任务的同时，保证开发质量，是支付系统开发面临的难题。 微服务架构是解决这个问题的利器，不过也需要强大的基础设施的支持。 “没有金刚钻别揽瓷器活”， 基础设施不完善，会反而影响微服务的开发实施效率。

#### 微服务和自动化

传统的软件过程一个大问题是各个阶段都需要人工干预。 而微服务架构的引入，其思想是通过降低系统的复杂度来使得过程的自动化成为可能。对此，Martin Fowler首先提出了在极限编程（敏捷）中使用持续集成的观点。在微服务架构之后，由提出了持续交付的概念。 微服务架构如何支撑这些过程，可以参考相关资料，本文不再详细介绍。 而从支付系统的角度，它的基础设施建设有什么不同之处？如果从微服务的角度来考察，和其它业务系统开发并无不同之处。无非是要求更严格罢了。构建的原则也是.”人管代码，代码管机器。通过流程自动化，逐步消除软件过程中的人工干预，加快迭代，提升质量。

怎么度量支付系统基础设施建设的成熟度？我们没法从部署了多少个软件，使用了多少台机器这样硬性指标来考察。敏捷开发的一些最新理念可以帮助我们定性地度量这个进度。 软件开发过程包括编码，构建，集成，测试，交付（预发布），部署。我们可以从这流程的自动化程度来度量基础设施建设的阶段。

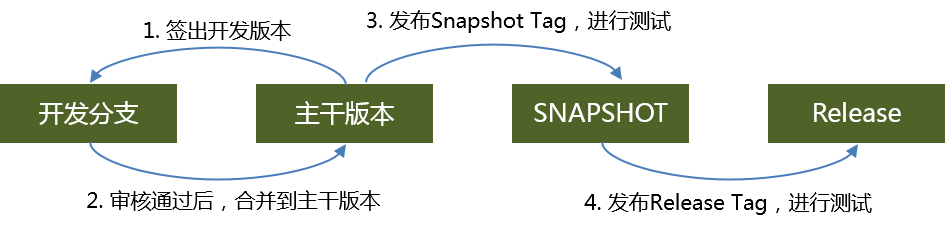


持续集成、持续交付和持续部署，分别对应自动化软件过程的三个阶段。持续集成实现了编译、发布、集成编译的自动化，并最终自动部署到集成测试环境。在测试完成后，需要人工验证和上线。 而持续交付则更进一步，在实现自动测试基础上，能够实现自动部署到预发布环境。在准线上环境确认达到可以上线的要求后，人工部署到线上环境。 持续部署是自动化的最终目标，开发完成后，能够自动地完成验证并实施到线上的系统。

不管是哪个阶段，都需要自动化的基础设施的支持。这里分头介绍其中主要的基础设施软件，以及他们的选型。

#### 版本控制

版本控制所有自动化工作的基础。国内大部分公司已经完成了从subversion到git的改造，git也成为版本控制的标配了。支付系统在版本控制上和其他系统并无太多的差异。这里需要介绍的是针对微服务架构的版本控制。 我们知道对版本控制来说，代码合并是一个很难避免的噩梦。 而微服务化可以很好的解决这个问题，由于服务的粒度小，每次变更一个人就可以搞定。每个服务有都可以独立上线，避免修改冲突。 这样版本控制就相对来说比较简单：



#### 代码审核

支付系统的每一行代码都要执行审核！代码审核对支付来说意义重大，是避免恶意代码不可缺少的一个环节。 一般来说，支付代码要求至少是2人审核通过。代码需要执行日常审核，而不是到快发布时的统一审核。审核的工具一般是需要和版本控制相集成的。 subversion上用reviewboard, git上用gerrit或者gitlab。虽然说gitlab是比较新的系统，不过还是推荐gerrit，可以强制代码审核以及控制代码审核流程，确保两个人都OK后才能入库。

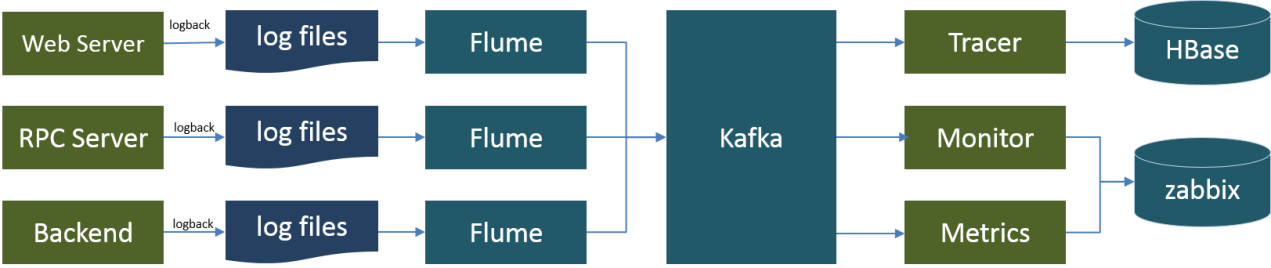
对于使用gitlab的系统而言，它的优势在于可扩展性强。gitlab默认不支持强制代码评审。但如果不强制执行代码评审，那会出现开发人员未经code view就提交自己merge代码。而reviewer 未能够及时review代码，也会影响进度。此外，开发人员没有执行代码审计（sonar）就提交代码，也是常见的事情。 好在gitlab有非常好的可扩展性，通过webhook可以根据需要实现各种额外功能。 webhook是gitlab的一个扩展点，通过用户提供的回调HTTP请求来监听git的push、comments、merge等事件。比如可以要求必须至少两个LGFM（LooksGoodForMe)才可以merge。通过webhook来监听comments，汇总两个LGFM之后，自动将代码merge到trunk上。

#### 代码审计

或者说是静态代码审查。Apache PMD， FindBugs都是常用的工具，推荐用Codehaus Sonar系统，即可以实现和Maven的集成，也可以有Web UI可以查看代码质量。提供的审核规则也比较全面，并且可以根据公司的需求来定制。

#### 日志搜集与分析

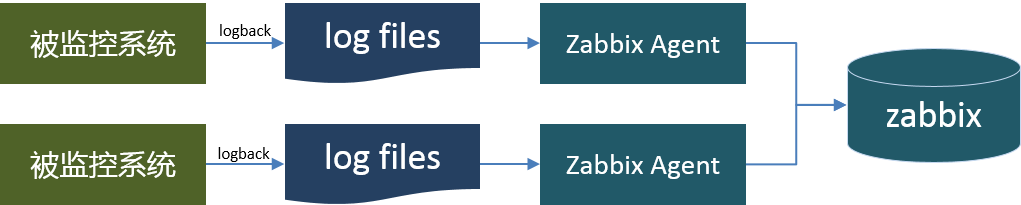
开发同学不碰线上系统，这是支付系统的原则。那线上系统出问题了怎么办？开发人员总是依赖日志来排查问题，一个日志汇总系统是支付平台必备的基础设施。考虑到日志最终都需要归并到一个日志仓库中，这个仓库可以有很多用途，特别是日常维护中的日志查询工作。多数指标可以在日志上完成计算的。 借助这个系统，也可以完成监控：



日志通过Apache Flume来收集，通过Apache Kafka来汇总，一般最后日志都归档到Elastic中。 统计分析工作也可以基于Elastic来做，但这个不推荐。 使用Apache Spark 的 Streaming组件来接入Apache Kafka 完成监控指标的提取和计算，将结果推送到Zabbix服务器上，就可以实现可扩展的监控。 Apache Flume和Logstash都可以用于日志收集，从实际使用来看，两者在性能上并无太大差异。Flume是java系统，Logstash是ruby系统。使用中都会涉及到对系统的扩展，这就看那个语言你能hold住了。 Apache Flume和Logstash都支持日志直接入库，即写入HDFS，Elastic等，有必要中间加一层Kafka吗？太有必要了，日志直接入库，以后分析就限制于这个库里面了。接入Kafka后，对于需要日志数据的应用，可以在Kafka上做准实时数据流分析，并将结果保存到需要的数据库中。

#### 系统监控

现在基本上 Zabbix 成为监控的标配了。 一个常规的 Zabbix 监控实现， 是在被监控的机器上部署Zabbix Agent，从日志中收集所需要的数据，分析出监控指标，发送到zabbix服务器上。



这种方式要求每个机器上部署 Zabbix 客户端，并配置数据收集脚本。Zabbix的部署可以作为必装软件随操作系统一起安装。

#### 持续集成

毫无疑问，Jenkins就是CI的不二之选。 但是这也只是一个工具，怎么用还得结合业务来实现。而上面列出来的这么多工具和使用要求，如何确保开发人员安规范来实现，并且尽可能地自动化，一个解决方案就是用集成工具将这些活动串起来。这里不详细介绍Jenkins的原理或者它和hudson的恩恩怨怨，重点描述如何使用。 Jenkins使用的要点是设置各个Job。而Job的设置，又分为线上和测试的Job。测试环境的Job分为 部署、启动、停止三个群组。线上分为 预部署、部署、启动和停止四个群组。 在每个群组中，每个项目对应一个可执行Job。

在测试环境部署执行如下工作：

获取受控代码。如果有指定版本号，则下载该版本号对应的代码；否则获取最新的Tag分支并下载该代码。集成是必须从代码开始构建，目的是保证线上运行的系统和版本控制服务器上的代码是一致的，而不是从某人机器上修改后的代码直接传上去的。 如果出现问题，只要获取该版本的代码即可定位到出问题的地方；

执行代码审计

执行maven deploy命令，执行编译、单元测试、版本发布等工作。 注意，这个阶段发布的包，都是SNAPSHOT版本的包。

生成javadoc, 接口文档，发布到测试服务器上，测试人员将对这个文档做验证。

生成单元测试覆盖率报告、代码质量报告。

发布系统到测试环境上。

在线上环境部署执行如下工作：

如果有指定版本号，则下载该版本号对应的代码；否则获取最新的Tag分支并下载该代码；

执行mvn setVersion命令， 将所有SNAPSHOT版本依赖修改为正式版本依赖。

执行代码审计

执行maven deploy命令，执行编译、单元测试、版本发布等工作。 注意，这个阶段发布的包，都是RELEASE版本的包。

生成javadoc, 接口文档，这是正式版本的文档。

发布系统到线上环境上。

### 支付基础设施

### 支付系统的监控与报警

关于监控，在各个技术网站，几乎都是一搜一大把。几个大的互联网公司，也都有开发自己的监控系统。 关于这方面也有不少分享。 这里介绍针对支付系统的监控和报警，但大部分内容，应该来说，对其他系统也是通用的 。

现在基本上 Zabbix 成为监控的标配了。 一个常规的 Zabbix 监控实现， 是在被监控的机器上部署Zabbix Agent，从日志中收集所需要的数据，分析出监控指标，发送到zabbix服务器上。 ！[zabbix 监控](http://static.cocolian.cn/img/in-post/monitor-1.png) 这种方式要求每个机器上部署 Zabbix 客户端，并配置数据收集脚本。Zabbix的部署可以作为必装软件随操作系统一起安装。

#### 系统监控

先说相对比较简单的系统监控，一般系统监控关注如下指标：

CPU负载

内存使用率；

磁盘使用率；

网络带宽占用

这些指标在zabbix agent中会提供默认实现，通过简单配置即可激活。装机时可以考虑统一配置这些监控。

#### JVM监控

JMX提供了关于JVM的大部分核心信息，启动时设置参数，支持远程访问JMX，之后即可通过接入JMX来实时读取JVM的CPU,内存等信息。zabbix也支持通过JMX来获取信息。

#### 服务监控

服务监控主要指接口的状态监控。 服务监控关注如下指标：

**QPS，每秒请求数** 对于使用容器的系统，包括Apache Tomcat,Resin,JBoss等，可以从Access Log中采集到每个接口的QPS。没输出Access Log的系统，考虑通过Annotation来规范输出访问计数。当然，这个指标还可以细分为 每秒成功请求数、失败请求数、总请求数等。

**请求响应时间** 在服务器端监控每个接口的响应时间。简单做法是在方法执行前后打时间戳计算，对于HTTP请求，也可以从access log中获取接口执行时间。当然也可以用annotation来实现统一的执行时间监控。

**执行异常数** 指程序运行过程中发生的未捕获处理的异常，一般是对场景考虑不周导致的异常发生，比如空指针、错误参数、数据访问等的异常。 这些异常一旦发现，需要修复代码逻辑。 异常在应用日志中一般都会把错误堆栈打印出来。

| **监控项目** | **监控数据** | **数据源** |
| --- | --- | --- |
| 系统监控 | CPU使用率 | zabbix agent默认采集 |
| 系统监控 | 内存使用率 | zabbix agent默认采集 |
| 系统监控 | 带宽使用率 | zabbix agent默认采集 |
| JVM内存监控 | free, used, heepsize | JMX |
| GC监控 | GC内存 | GC 日志 |
| GC监控 | GC内存回收时间 | GC 日志 |
| Tomcat监控 | 最大会话数、会话数、互动会话数 | JMX |
| Tomcat监控 | 最大线程数、当前线程数、繁忙线程数 | JMX |
| 服务状态监控 | 404/505等status数量 | Tomcat访问日志 |
| 服务状态监控 | 每个接口的访问量 | Tomcat访问日志 |
| 服务状态监控 | 每个接口发送的字节量 | Tomcat访问日志 |
| 服务状态监控 | 每个接口的最大响应时间 | Tomcat访问日志 |
| 服务状态监控 | 每个接口的最平均响应时间 | Tomcat访问日志 |
| 服务监控 | 错误数 | 应用日志 |

#### 数据库监控

数据库是大部分应用的核心和瓶颈，对其监控尤其必要。监控可以 在应用侧执行，也可以在数据库服务器上做。前者通过应用代码中打印日志来实现，或者直接override 链接池中相关方法来统一输出日志。在数据库服务器上执行监控，需要根据数据库的特点分别设计方案。以MySQL为例，可以通过监控其bin log来获取执行的sql语句以及执行时间。使用Alibaba Canal 来对接MySQL 的BinLog， 接收到BinLog消息后，解析消息数据，可以获取请求的SQL、参数、执行时间、错误代码等信息。

数据库监控重点关注如下指标:

每秒请求数

慢查询处理数

SQL语句执行时间

#### 调用链监控

调用链监控指在微服务系统中，跟踪一个请求从发起到返回，在各个相关系统中的调用情况。 调用链监控是跨系统的监控，需要在请求发起时分配一个可以唯一识别本次调用请求（或者成为事务）的ID，这个ID会被分发到每个调用上。之后在调用日志中输出该ID。当所有日志都汇总起来后，可以从日志中分析本次调用的流程。 对于HTTP/HTTPS请求，可以考虑将ID放到Header里面，这样不会影响接口逻辑。

#### 业务监控

业务监控是一个复杂的话题。这里以支付为例，说明业务监控的架构和实现。

##### 支付业务监控

每个支付通道监控包括如下内容：

支付通道接口请求数： 如果一段时间内接口请求环比大幅度下降，可能是该接口出现问题了。

支付通道接口请求失败数，即调用接口失败的数量。

支付通道接口请求延迟。

支付通道支付失败率。每个通道支付有一定的失败率，如果给定时间内突然有超过这个失败率的情况出现，则可能是通道出现问题了。

支付通道同步、异步调用次数。

支付接口，如支付、提现、退款、签约、订阅等，监控如下内容：

总金额，如果总金额有大的波动，则有洗钱的可能。

每笔平均金额。

支付成功率

#### 监控架构

实际上对一个业务来说，大部分系统监控的指标是类似的，而按照这种方式，每个指标在各个被监控系统中还需要单独写脚本实现，工作量大。针对这种情况，可以采用日志集中监控的方式来处理。 考虑到日志最终都需要归并到一个日志仓库中，这个仓库可以有很多用途，特别是日常维护中的日志查询工作。多数指标可以在日志上完成计算的。 借助这个系统，也可以完成监控： ！[zabbix 监控](http://static.cocolian.cn/img/in-post/monitor-2.png)

日志通过Apache Flume来收集，通过Apache Kafka来汇总，一般最后日志都归档到Elastic中。 统计分析工作也可以基于Elastic来做，但这个不推荐。 使用Apache Spark 的 Streaming组件来接入Apache Kafka 完成监控指标的提取和计算，将结果推送到Zabbix服务器上，就可以实现可扩展的监控。 这个架构的优势在于：

监控脚本的跨系统使用。 指定日志规范后， 只要按照这个规范编制的日志，都可以纳入监控，无需额外配置。

服务重新部署时无须考虑监控脚本的部署，所有监控直接生效。

难点在于，提炼一套通用的日志规范，考虑如何通过Spark来分析日志。

#### 日志收集

Flume和logstash都可以用于日志收集，从实际使用来看，两者在性能上并无太大差异。flume是java系统，logstash是ruby系统。使用中都会涉及到对系统的扩展，这就看那个语言你能hold住了。

#### 日志数据流

flume和logstash都支持日志直接入库，即写入hdfs，elastic等，有必要中间加一层kafka吗？太有必要了，日志直接入库，以后分析就限制于这个库里面了。接入kafka后，对于需要日志数据的应用，可以在kafka上做准实时数据流分析，并将结果保存到需要的数据库中。

#### 日志分析

Streaming分析，可以走spark，也可以用storm，甚至直接接入kafka做单机处理。这取决于日志数据规模了。spark streaming是推荐的，社区活跃度高，又集成了多种算法。

#### 日志系统与日志框架

Java主流的日志系统有log4j，JULlogback等，日志框架有apache commons logging，slf等，关于这些系统的历史掌故恩怨情仇八卦趣事，网上有不少资料，这里不详细介绍。

#### 日志系统选型

最好的编程语言是PHP还是Java？ 同样的，也有争论：最好的日志框架是slf还是commons-logging？最好的日志系统是Log4j还是Logback？在使用上，它们的API和使用方式大体类似，slf有模版支持，但这也不是关键需求。而性能方面，从我们测试用例中也没有发现哪个系统或框架有明显优势。对性能有决定性影响的是使用方式。

#### 日志高能预警

根据我们的测试，在高并发系统中，关于日志，有如下结论：

Log4j与logback在高并发下性能上并无太多差异，不用太纠结使用哪个API，.影响性能的是日志内容的写法和数据量

输出类名和行号会严重影响性能，这需要使用到性能不佳的反射机制。执行频率高，性能要求高的代码，禁用反射，禁用new操作。

高峰期系统出错，如果打印错误堆栈，那绝对是雪上加霜，理由同上。

多线程时输出日志，写锁是影响性能的关键因素。缓解写锁的措施，首选加大日志写入缓冲区，其次是异步打印。异步对性能有提升，但不显著。写锁出问题的一个现象是CPU跑满。

日志分级本身无太大意义。

## 系统重构与进阶

### 三户模型互联网应用

#### 传统金融

在传统电信或金融行业当中，通常是现有客户，然后才有的用户和账户。通过一个场景可以清晰理解。如：去银行办卡，填写申请表的时候就已经把自己作为自然人的基本信息（姓名，年龄，性别，身份证号等信息）填写提交给柜员，柜员登记录入系统，此时系统里就有了客户信息，然后再签协议，开通某一种银行卡（包含哪些服务），就有了用户，然后根据开卡需要创建银行账户。

在银行柜台，操作人员通过面签或实名方式在系统中建立客户对象。这个对象的业务主键是证件号（如，身份证，假设目前 18 位证件都不重号），所有相同证件我们都视为同一个客户，不论是不是同一个系统中。同一个证件号，不论在工行还是中行，开出来的客户，其实都对应着现实中的同一个人。

#### 互联网

而互联网产品中，一般是先注册并同意服务协议，创建用户，创建相关账户，然后根据业务需要引导用户进行实名认证，创建其客户信息。

#### 问题

在传统金融行业，会涉及到一个重要的“客户归并”业务，既当相同身份证号进入到系统时，会被合并成一个客户，此过程称为“客户归并”。

在互联网产品中通常是先注册成为用户，然后补充自然人信息认证客户信息，假设在认证客户之前，同一个自然人注册了多个用户，那么全部用户都补全相同的客户信息时，就涉及到“客户归并”问题，但客户归并本身风险非常高，同时涉及到历史业务数据的合并较为复杂，所以互联网产品中基本做法是一个用户提供客户信息认证通过之后，其他用户就不能再用这个客户信息进行认证，只能使用未被认证过的客户信息提交认证。

也就是说一个信息平台下，同一款业务产品内，一个客户只有一个用户，不存在一个客户多个用户的情况；但同一个信息平台下，不同业务产品之间会存在多个用户属于同一个客户情况，因为在不同产品里用户身份并不一致。

#### 总结

传 统 应 用：客户 → 用户 → 账户

互联网服务：用户 → 账户 → 客户

### 三类账户

在设计支付账户系统时，合规是第一要求，和账户相关的法规，最重要的是近期央行发布的三个文件。 从 2013 年 3 月起，国家相继制定一系列“互联网+”计划，推动了移动互联网、大数据等与现代制造业、服务业相结合，引导企业转型，促进经济健康发展。在此背景下，央行相继发布了三个文件：

「**中国人民银行关于改进个人银行账户服务加强账户管理的通知**」，银发〔2015〕392 号，2015 年 12 月 25 日发文，正式启动账户分类管理工作。

「**中国人民银行关于落实个人银行账户分类管理制度的通知**」，银发〔2016〕302 号, 2016 年 11 月 25 日发文，明确了同一个人在同一个银行只能有一个I类账户的要求。

「**中国人民银行关于加强支付结算管理防范电信网络新型违法犯罪有关事项的通知**」，银发〔2016〕261 号， 2016 年 9 月 20 日发文，调整并细化了账户分类及使用要求。

在这些文件中，央行明确对账户实施分类管理的要求。

### 支付系统整体架构实例

### 支付网关实例

支付网关是直接对接业务系统的接口，它本身并不执行任何支付相关的业务逻辑。它将支付产品接口中和业务无关的功能提取出来，在这里统一实现。这样在具体产品接口中，就无需考虑这些和业务无关的逻辑。支付网关设计还和对外的接口参数有关。我们看一下业内几个主流的支付平台的接口设计。

#### **支付宝**

对外接口采用统一参数的方式，参考[APP请求参数说明](https://doc.open.alipay.com/docs/doc.htm?spm=a219a.7629140.0.0.8jvvbW&treeId=193&articleId=105465&docType=1)。 接口参数分为三层： 公共参数、业务参数、还有业务扩展参数。 其中公共参数是各个请求接口中公用的。



业务相关的参数，通过特定的规则拼接再biz\_content上。最后将参数生成签名，放到sign字段中。

支付宝的接口混合json格式和query string格式，在参数命名上，既有下划线方式的，也有驼峰的。英文单词的使用也不太规范。期待后续版本能做的更好。

#### **微信支付**

和支付宝不一样，微信支付是采用XML格式来作为报文传输。在其[接口文档说明](https://pay.weixin.qq.com/wiki/doc/api/app/app.php?chapter=4_1)中, 对XML报文格式有详细的描述。当然，也使用签名字符串来保证接口的安全，签名结果放在sign标签下。

在接口设计上，和支付宝还有一些差距。有些参数命名不一致，比如商户号，有些接口中叫mch\_id，有些接口是partnerid。

#### **PayPal**

PayPal是标准的Restful设计，将支付中涉及到的对象，如Payment， Order， Credit Card等，以资源的形式，支持通过Restful API来操作。

PayPal的定位以及设计目标和国内第三方支付平台不同，它以支持国际营收为主。对国内应用来说，其易用性和支付宝、微信支付相比还稍逊一些，不过Paypal一直是支付API设计的典范。

对电商支付平台来说，其定位更接近于一个聚合支付。聚合多种支付方式，为公司各个业务提供支持。 在这里，支付网关和支付产品的设计尤为关键。合理的接口设计能够大大降低支付渠道对接的开发工作量。一般支付产品不会超过10个，而根据公司的规模，对接的支付渠道超过100个都有可能。



### 退款处理

#### 一、退款逻辑

1、退款请求基本要素判断：订单支付状态、是否支持退款、退款金额、退款帐户、退款描述；

2、通过基本要素判断后的退款请求会透传至渠道侧并生成退款对象；

* 如果渠道侧未明确返回退款处理的结果或状态，则设置为 pending（这里需要注意的是，如果出现请求连接超时或网络异常，将该笔退款状态设置为 pending ，且会进行一次原单重试，若依旧连接超时或网络异常，保持 pending 并通过后续的查询结果更新状态）；
* 如果渠道侧返回了退款处理失败的结果或状态，则设置为 failed；
* 如果渠道侧返回了退款处理成功的结果或状态，则设置为 success；

3、根据渠道的退款成功异步通知或退款查询结果，更新退款的状态；

4、针对所有对接的渠道均设置了退款状态的自动查询逻辑（按照一定的时间间隔），主要因为以下两点原因：

* 部分渠道没有退款成功的异步通知；
* 由于某些魔障的原因导致没能正常接收渠道的异步通知；   
  5、针对部分渠道有会特殊处理逻辑：比如由于部分渠道侧的原因，我们会增加对于退款查询返回的特定错误码标记为处理中（pending），再按照与渠道达成的时间约定再次查询更新，或者进行手动更新，这种情况和概率都比较少，多见于渠道侧的数据不同步、逻辑待优化情况。

#### 二、渠道侧处理

#### 2.1 微信

**1.退款周期**

默认 1 年（客户可自行与渠道 BD 申请延长可退款周期）

**2.退款规则**

##### **1）退款创建**

* 微信支付退款支持单笔交易分多次退款，多次退款需要提交原支付订单的商户订单号和设置不同的退款单号。申请退款总金额不能超过订单金额。一笔退款失败后重新提交，请不要更换退款单号，请使用原商户退款单号；
* 每个支付订单的部分退款次数不能超过 50 次；
* 请求频率限制：150qps，即每秒钟正常的申请退款请求次数不超过150次。错误或无效请求频率限制：6qps，即每秒钟异常或错误的退款申请请求不超过6次。

##### **2）退款查询**

* 提交退款申请后，通过调用该接口查询退款状态。退款有一定延时，用零钱支付的退款20分钟内到账，银行卡支付的退款 3 个工作日后重新查询退款状态；
* 如果单个支付订单部分退款次数超过20次请使用退款单号（refund\_id）查询；
* 微信订单号查询的优先级是： 微信退款单号refund\_id > 商户退款单号out\_refund\_no > 微信订单号transaction\_id > 商户订单号out\_trade\_no；

##### **3）退款通知**

* 开通该功能需要在商户平台-交易中心-退款配置中配置notify\_url ；
* 退款的三种情况会发送异步通知：SUCCESS-退款成功、CHANGE-退款异常、REFUNDCLOSE-退款关闭；

#### 2.2 支付宝

**1.退款周期**

默认 3 个月（客户可自行与渠道 BD 申请延长可退款周期）

**2.退款规则**

##### **1）退款创建（针对 openapi 接口）**

* 支付宝将在收到退款请求并且验证成功之后，按照退款规则将支付款按原路退到买家帐号上。
* 交易超过约定时间（签约时设置的可退款时间）的订单无法进行退款，支付宝退款支持单笔交易分多次退款，多次退款需要提交原支付订单的商户订单号和设置不同的退款单号。
* 一笔退款失败后重新提交，要采用原来的退款单号。总退款金额不能超过用户实际支付金额；
* 返回码 code = 10000 且 fund\_change = Y 时，可以判断退款成功；

##### **2）退款查询**  返回码 code=10000，仅代表本次查询操作成功，不代表退款成功。如果退款查询接口返回了查询数据，则代表退款成功，如果没有查询到则代表未退款成功，可以调用退款接口进行重试。重试时请务必保证退款请求号一致。

##### **3）退款通知**

* mapi 接口（旧）：退款成功时会发送异步通知：REFUND\_SUCCESS-退款成功；
* openapi 接口（新）：无

#### 2.3 QQ 钱包

**1.退款周期**

默认 1 年（客户可自行与渠道 BD 申请延长可退款周期）

**2.退款规则**

##### **1）退款创建**

QQ钱包退款支持单笔交易分多次退款，多次退款需要提交原支付订单的商户订单号和设置不同的退款单号。一笔退款失败后重新提交，要采用原来的退款单号。总退款金额不能超过用户实际支付金额；

##### **2）退款查询**

* 商户退款单号 out\_refund\_no 和 QQ 钱包订单号 transaction\_id 映射关系只保存1个月，超过1个月查 out\_refund\_no 将会查不到订单，提示“REFUND\_NOT\_EXIST”，按照 transaction\_id 来查退款单则不会有问题；
* qq 钱包在落 db 前有很多检查，没落 db 的退款请求在查询时会提示“REFUND\_NOT\_EXIST”，以下退款情况不会落 db ：   
  a.根据交易单号或者钱包单号没有找到支付订单；   
  b.退款的商家帐户余额不足；   
  c.续费查询失败；

##### **3）退款通知**

无

#### 2.4 翼支付

**1.退款周期**

默认 6 个月（客户可自行与渠道 BD 申请延长可退款周期）

**2.退款规则**

##### **1)退款创建**

* 商户向翼支付网关发起退款请求，翼支付网关回传退款请求受理结果（“受理成功”or“受理失败”）；
* 退款请求受理成功后，翼支付网关向银行发起线上退款，由银行进行退款并将退款处理结果回传至翼支付网关（“退款成功”or“退款失败”）；

##### **2)退款查询**

* 线上退款转至线下退款时，因线下退款处理时效较长，商户发起查询退款结果的请求时，翼支付查询订单是根据商户发起请求的时间 ±1 天的时间区间进行查询的，但实际线下退款处理时间可能超过该区间，故可能导致查询不到该笔退款订单；

##### **3)退款通知**

* 线上退款的两种情况会触发异步通知：退款成功、退款失败。线下退款成功不会同步信息至线上且不会发起异步通知。
* 如线上退款失败，商户会收到“退款失败”的异步通知，该笔退款订单第二天由翼支付系统自动向银行再次发起退款，如果退款成功，将异步回调商户“退款成功”，并将翼支付网关系统内该笔订单“退款失败”的状态更新为“退款成功”；如果线上退款失败将转由翼支付线下人工退款，线下退款成功后，翼支付网关系统会将该笔订单“退款失败”的状态更新为“退款成功”，并不再异步回传处理结果。
* 部分银行不支持当日交易退款，在发生当日交易退款时，退款订单将会返回商户“退款失败”，并进入翼支付网关系统待退款表内，系统将于次日自动再次发起退款，并返回处理结果；

#### 三、关于手续费

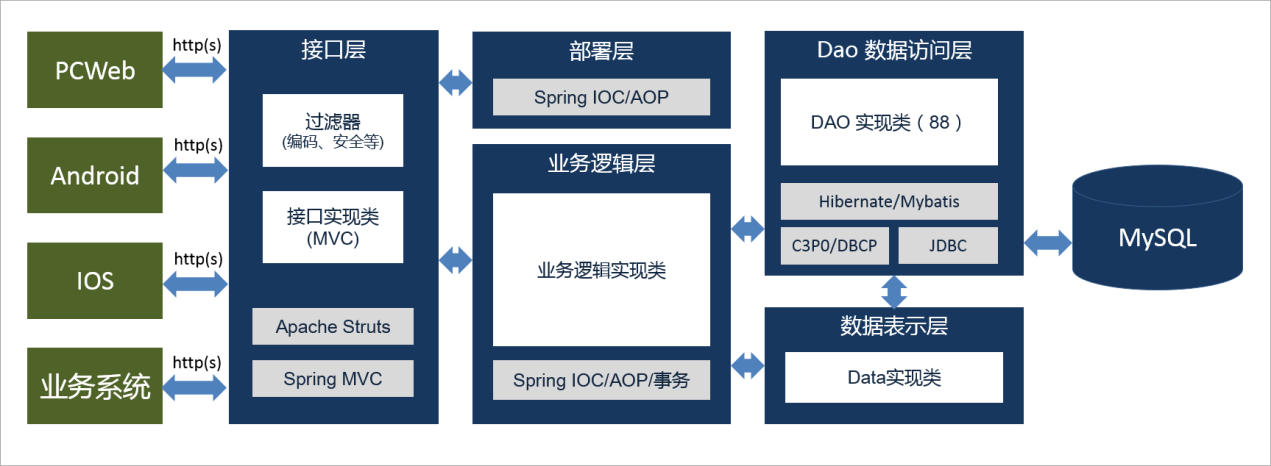
1. 一般银行方不退
2. 支付宝，不退回（默认） 支付宝针对不同的支付产品退款退回手续费策略不一样
   * 当面付产品，退款时候会退回手续费；
   * 非当面付产品主要根据商户与支付宝 BD 签约时候的具体协议，可以申请退手续费；
3. 微信，退回
4. 银联，退回
5. 京东，不退回（默认） 默认退款不退回手续费，商户可以向京东申请退款时候退手续费。
6. QQ 钱包，退回

### 支付网关系统的重构

在支付系统改进中，我们对原有系统做了整体的评估，选择支付网关作为入手点来进行微服务架构的改进。这里详细介绍我们针对该模块的改进过程，供参考。

#### 原有系统情况

早期启动的时候，对接的支付渠道不多，所有支付渠道和支付网关都实现在一个项目中，部署在一起。其中支付网关是整个项目的核心和入手点。它为各个业务方提供支付全流程的调用接口，签约、代扣、支付、验证，都是通过这个接口来实现的。整个系统使用SSH框架，架构如下：



业务流程如下：

当接口被调用时， 首先执行参数校验，确认输入的参数的合法性，验证参数签名是否正确。确认过程包括调用账户、用户、支付方式、路由等服务来验证用户ID、账户、支付卡号、支付金额等参数。

根据输入的支付方式，调用支付路由服务，获取对应的支付渠道。

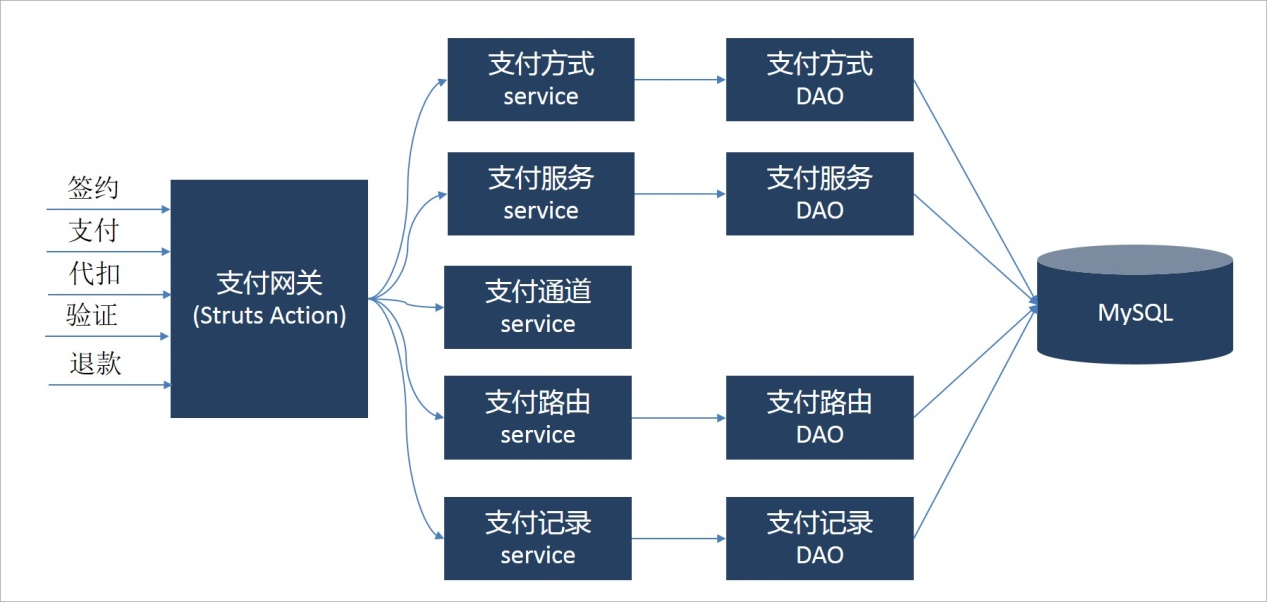
调用风控接口进行验证，如果有交易风险，则阻断本次交易。

生成交易记录；

调用支付渠道提供的服务执行支付。

根据支付结果，更新订单状态；

通知商户订单执行结果。

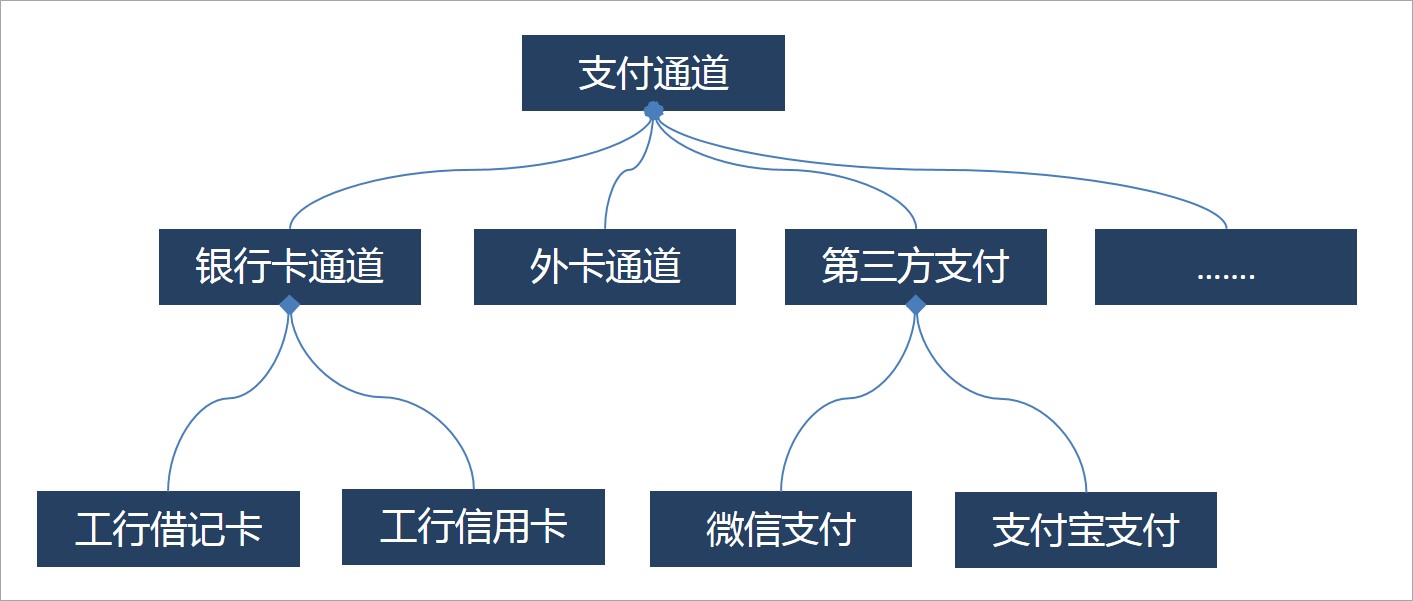
在实现上，原有系统实现的类结构图如下：  


采用SSH架构，支付网关实现为一个大Apache Struts Action类，和支付相关的所有业务逻辑都实现在一个项目中。

支付网关承载大量的功能，实际上，它是将API网关和业务逻辑都混在在一起实现。 签约、支付、代扣、验证，都在这一个类中实现，代码行数超过1000行，逻辑十分复杂。

除了风控是进程外调动，其他的服务都是进程内调用，通过springframework来管理各个service。

最终落地调用的支付渠道，是通过抽象的接口来对网关封装渠道的差异。



最终在这个系统中对接了有30多个渠道，类规模达到2000个。随着业务发展，问题越来越多。高峰期同时有5个渠道在并行开发，还有大量的其他渠道对接问题需要修复。多个人同时修改一个项目代码导致版本控制的工作骤增。上线频发引起服务中断也让业务方很不满。对支付网关的改进是一个循序渐进的过程。这里参考Arun Gupta的[微服务六种设计模式](https://www.javacodegeeks.com/2015/04/microservice-design-patterns.html)，来描述我们所做的改进。

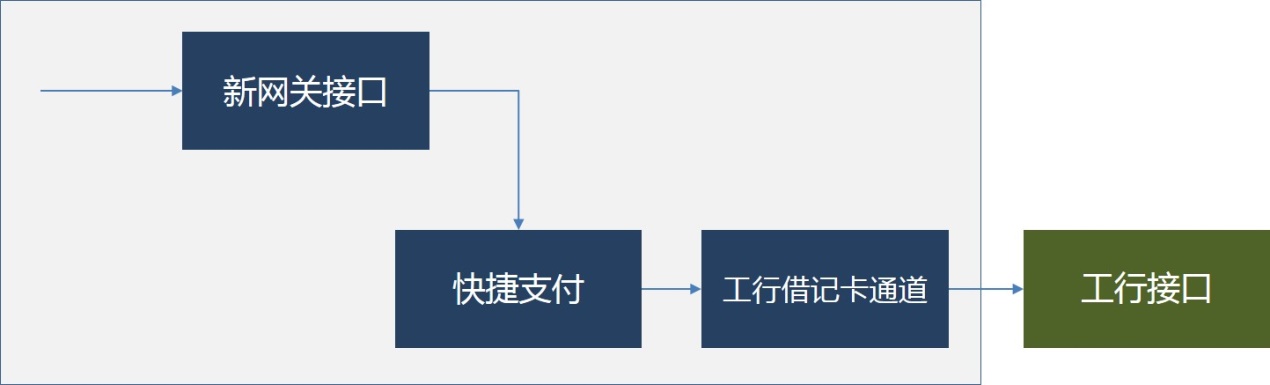
#### 新网关设计 (Chain Pattern)

为了分解旧网关的功能， 我们设计了新的网关。在处理流程上，将其分为三个步骤，采用的是chain模式。



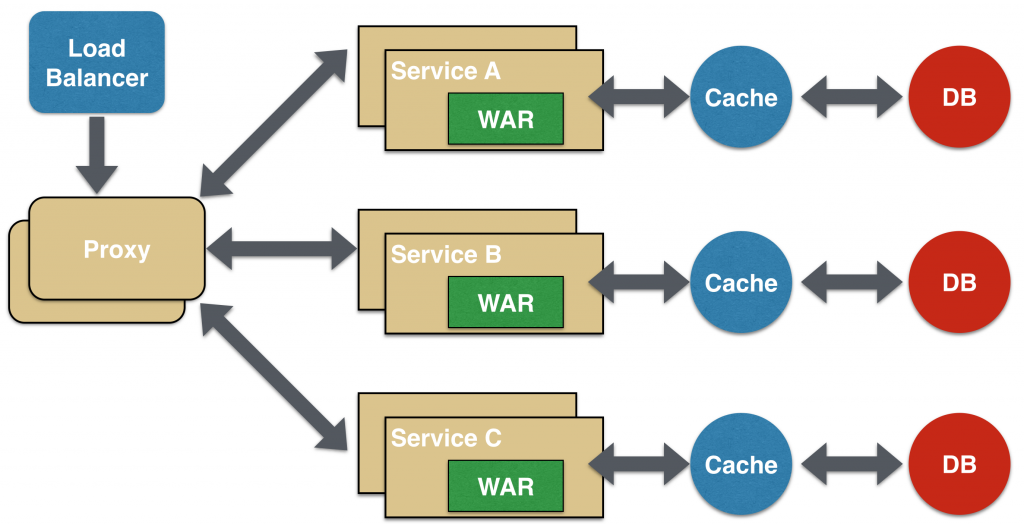
链式模式，如上图所示，它调用服务A来获取结果，而服务A是通过服务B来交互，B则会和C有交互。 整个过程类似同步的HTTP请求、响应处理。 这其中每个阶段的调用，都是阻塞式的同步调用。每一步都会增加一些业务逻辑处理。

原支付网关难以维护的一个重要原因是其所承载的功能过多。我们首先根据用户的使用场景，将支付网关承载的功能，按照支付产品来进行切分。 支付产品包括快捷支付、网银支付、外卡支付等。 不同的产品，其对应的操作所使用的参数和流程也不一样。以快捷产品为例， 新网关接收到请求后，根据用户所选择的支付类型，分发到快捷支付产品接口。快捷支付产品接口调用工行借记卡通道来执行支付，通道最终落地到工行接口的调用来实现支付。 支付操作完成后，工行接口通知到通道，通道通知到产品，最终逆向传递到网关接口，并最终发送给调用方。 在这里面，支付网关负责分发、验签等基本功能，支付产品负责参数校验、路由、生成交易记录等功能。最终的支付操作是落地到支付渠道去执行。



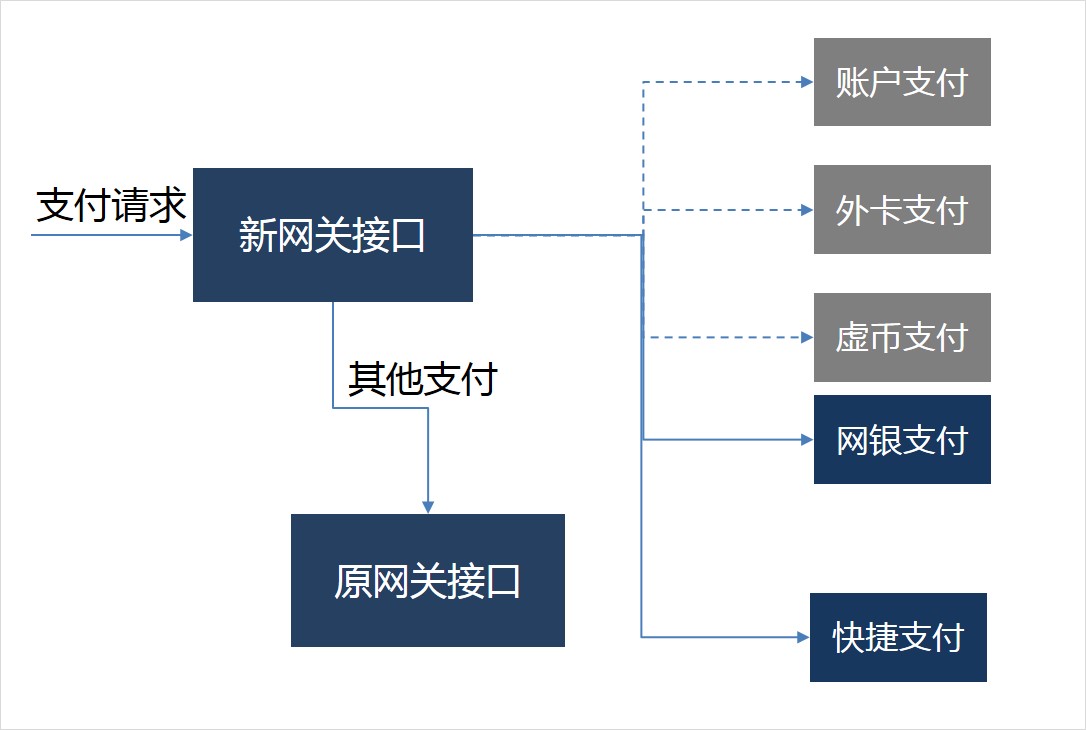
#### 网关拆分(Proxy Pattern)

如上所述，支付网关按照使用场景进行拆分。我们采用完善一个、接入一个的原则，在保留旧网关的功能的同时，开发完善新的网关和支付产品。等所有流量都打到新网关上去之后，旧网关就直接废弃了。为了达到这个目标，我们引入了代理模式：



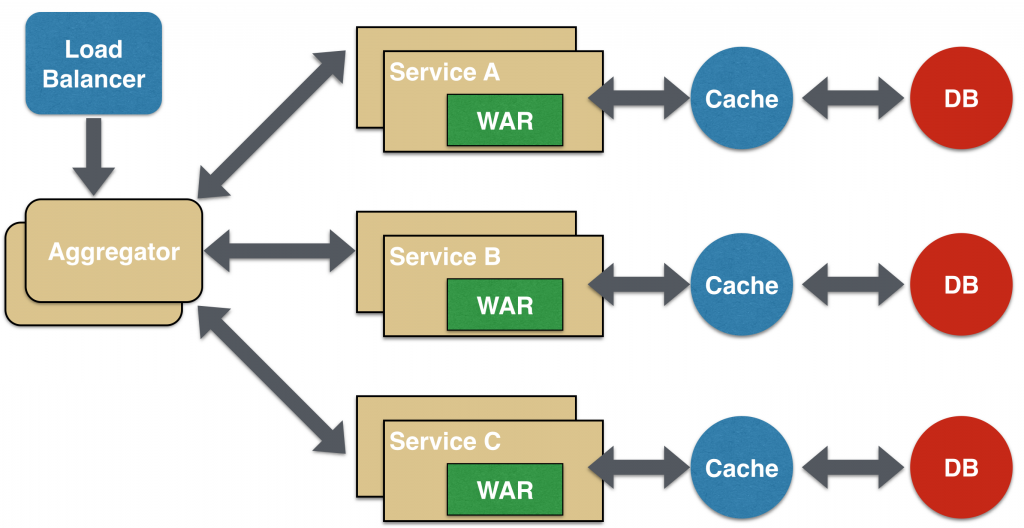
代理模式和聚合模式类似，不同点在于，它会根据业务逻辑需要仅选择一个微服务来调用。微服务中，我们经常会用代理模式来构建API网关。

我们首先按照所支持的支付方式，对支付网关做分解，拆分为为网银、快捷、话费、账户、外卡、虚币等支付产品。新网关接口模块是一个proxy，本身并未实现任何业务逻辑，它的工作是将用户请求发送给合适的支付产品去处理。如果这个产品还没有实现，则将其转发到老网关去执行。这样带来的好处是，我们不需要对老网关做任何改动。而且，如果某个支付产品在重构过程中出现问题，我们可以很快切回到老网关去。



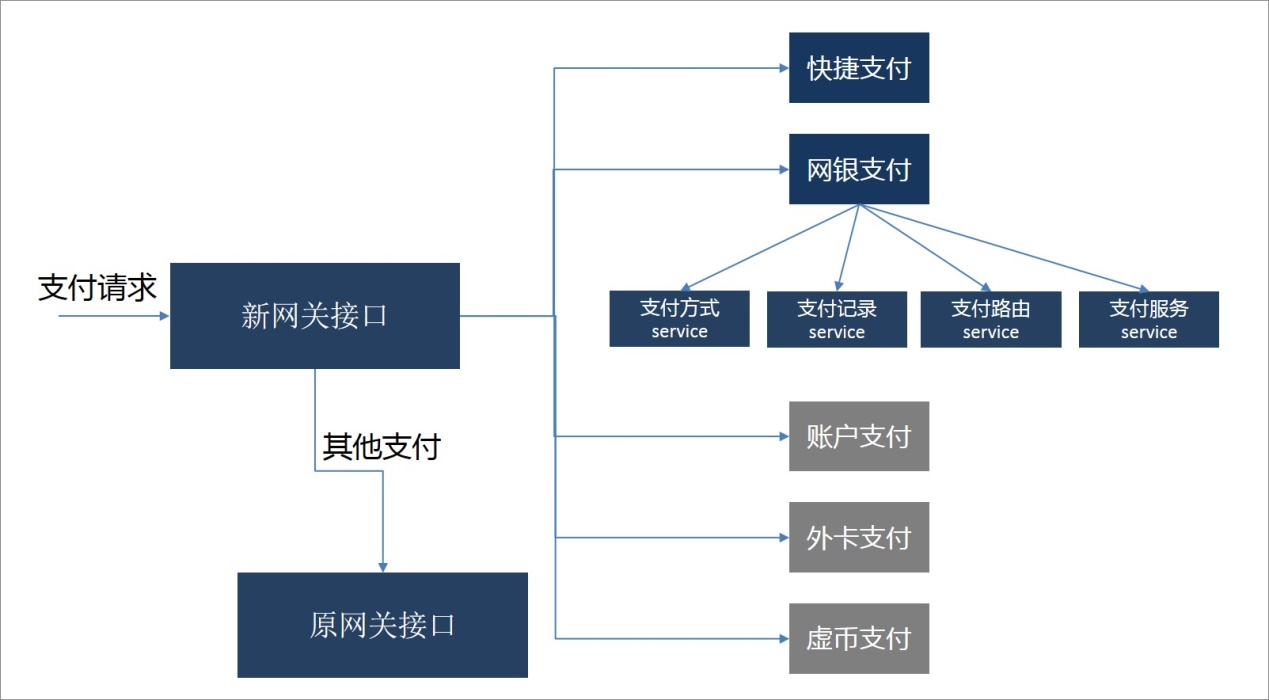
#### 支付产品 (Aggregator Pattern)

支付产品是对原有支付网关的业务流程实现的一个重构，按照各个支付产品所支持的功能以及流程来简化原混合在一起的设计。比如快捷支付需要签约和支付，而网银支付则不需要签约。 在支付产品本身的实现上，我们使用的是聚合模式。



聚合是最常见的微服务设计模式，它是一个高层次的微服务组合，供其他服务调用。 在这种情况下，聚合器会从其他的微服务中收集数据，做业务逻辑处理，然后发布成一个服务终端。其他有需要的服务可以调用它。 聚合器设计的要点是要遵循DRY(Don’t Repeat Yourself)原则。如果有多个服务需要访问A，B，C服务，那建议的处理方式是，针对这些使用，提炼一个处理逻辑出来，将A、B、C封装为一个新的服务，这个服务可以独立的演化。

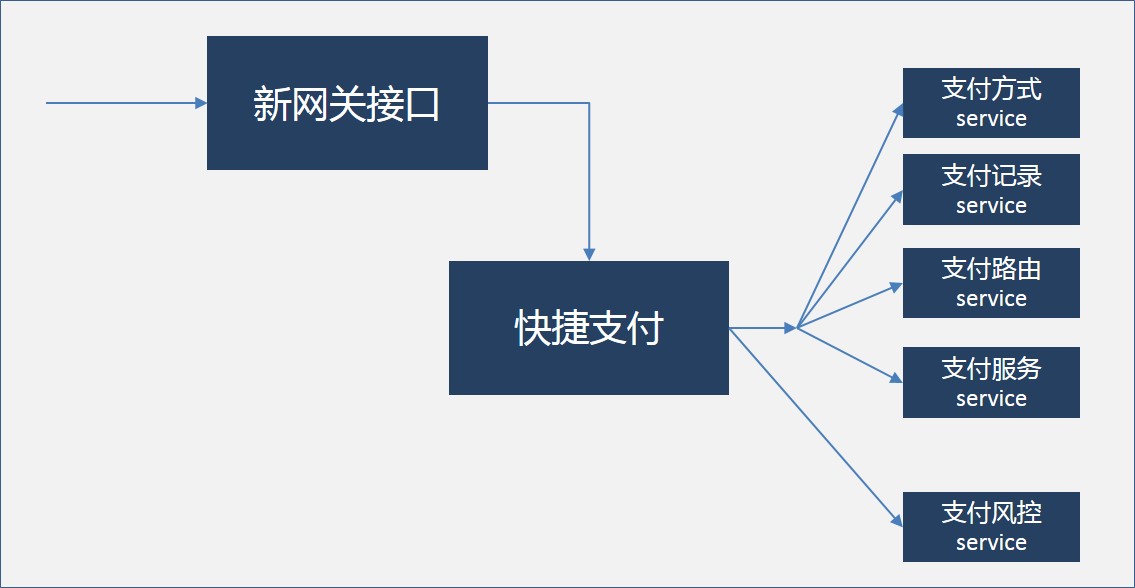
支付产品中调用的各个服务，包括支付方式管理， 支付服务管理，支付路由管理、支付记录管理等，都被重构为微服务，在支付产品的实现中，通过Aggregator 模式进行调用。



在支付产品的实现流程中，首先需要对参数进行校验，校验成功后，调用风控检查该交易是否可以放行。这两个操作，在处理上可以并行，使用的是分支模式。



分支模式是聚合模式的扩展，可以允许同时调用两个或者更多的微服务。



如上，采用分支模式， 使得数据校验和风控可以并发执行。由于风控相对耗时较长，而订单中需要校验的数据较多，这两个操作有必要并发执行。

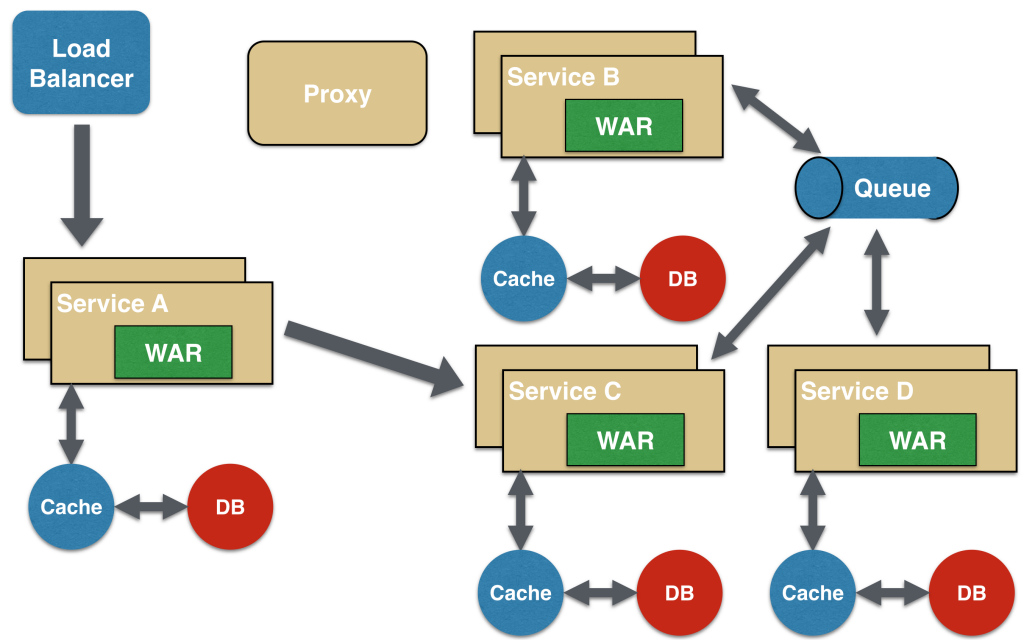
#### 支付通道 (Aggregator Pattern)

支付路由根据用户选择的支付方式对支付通道进行筛选，选取合适的支付通道。支付产品调用该通道的接口来最终落地完成支付服务。 每个支付通道对接也被实现为微服务，在支付产品中调用。



#### 通知商户 (Asynchronous Messaging Pattern)

支付产品执行的最后一个步骤是通知调用方支付的结果。 原系统实现是将这个步骤耦合在原有代码中，容易受到调用方接口的稳定性的影响。 为此，这里采用异步消息的模式来进行重构：



异步消息一般用于对流程中可以异步执行的操作做分解，将它从原流程中分离出来，通过消息机制来异步执行。 支付产品在完成支付服务后，发出消息到订单消息队列中。 商户回调处理程序接收到消息后，调用商户回调接口告知支付结果。 此外，风控、BI系统等，也可以使用这个消息来同步订单数据。



### 支付网关设计

在支付系统中，支付网关和支付渠道的对接是最核心的功能。其中支付网关是对外提供服务的接口，所有需要渠道支持的资金操作都需要通过网关分发到对应的渠道模块上。一旦定型，后续就很少，也很难调整。而支付渠道模块是接收网关的请求，调用渠道接口执行真正的资金操作。每个渠道的接口，传输方式都不尽相同，所以在这里，支付网关相对于支付渠道模块的作用，类似设计模式中的wrapper，封装各个渠道的差异，对网关呈现统一的接口。而网关的功能是为业务提供通用接口，一些和渠道交互的公共操作，也会放置到网关中。

**功能概述**

支付系统对其他系统，特别是交易系统，提供的支付服务包括签约，支付，退款，充值，转帐，解约等。有些地方还会额外提供签约并支付的接口，用于支持在支付过程中绑卡。 每个服务实现的流程也是基本类似，包括下单，取消订单，退单，查单等操作。每个操作实现，都包括参数校验，支付路由，生成订单，风险评估，调用渠道服务，更新订单和发送消息这7步，对于一些比较复杂的渠道服务，还会涉及到异步同通知处理的步骤。这里详细介绍这些步骤的实现要点。

#### 1. 执行参数校验

所有的支付操作，都需要对输入执行参数校验，避免接口受到攻击。

验证输入参数中各字段的有效性验证，比如用户ID,商户ID,价格，返回地址等参数。

验证账户状态。交易主体、交易对手等账户的状态是处于可交易的状态。

验证订单：如果涉及到预单，还需要验证订单号的有效性，订单状态是未支付。为了避免用户缓存某个URL地址，还需要校验下单时间和支付时间是否超过预定的间隔。

验证签名。签名也是为了防止支付接口被伪造。 一般签名是使用分发给商户的key来对输入参数拼接成的字符串做MD5 Hash或者RSA加密，然后作为一个参数随其他参数一起提交到服务器端。

#### 2. 根据支付路由寻找合适的支付服务

根据用户选择的支付方式确定用来完成该操作的合适的支付渠道。用户指定的支付方式不一定是最终的执行支付的渠道。比如用户选择通过工行信用卡来执行支付，但是我们没有实现和工行的对接，而是可以通过第三方支付，比如支付宝、微信支付、易宝支付，或者银联来完成。那如何选择合适的支付渠道，就通过支付路由来实现。支付路由会综合考虑收费、渠道的可用性等因素来选择最优方案。

#### 3. 评估交易风险

检查本次交易是否有风险。风控接口返回三种结果：阻断交易、增强验证和放行交易。

阻断交易，说明该交易是高风险的，需要终止，不执行第5个步骤；

增强验证，说明该交易有一定的风险，需要确认下是不是用户本人在操作。这可以通过发送短信验证码或者其他可以验证用户身份的方式来做校验，验证通过后，可以继续执行该交易。

放行交易，即本次交易是安全的，可以继续往下走。

#### 4.生成交易订单

将订单信息持久化到数据库中。当访问压力大的时候，数据库写入会成为一个瓶颈。

#### 5. 调用支付渠道提供的服务

所有的支付服务都需要第三方通道来完成执行。一般银行渠道的调用比较简单，可以直接返回结果。一些第三方支付，支付宝，微信支付等，会通过异步接口来告知支付结果。

#### 6. 更新订单

对于同步返回的结果，需要在主线程中更新订单的状态，标记是支付成功还是失败。对于异步返回的渠道，需要在异步程序中处理。

#### 7. 发送消息

通过消息来通知相关系统关于订单的变更。风控，信用BI等，都需要依赖这数据做准实时计算。

#### 8. 异步通知

如上述流程，其中涉及到调用远程接口，其延迟不可控。如果调用方一直阻塞等待，很容易超时。引入异步通知机制，可以让调用方在主线程中尽快返回，通过异步线程来得到支付结果。对于通过异步来获取支付结果的渠道接口，也需要对应的在异步通知中将结果返回给调用方。 异步通知需要调用方提供一个回调地址，一般以http或者https的方式。这就有技术风险，如果调用失败，还需要重试。而重试不能过于频繁，需要逐步拉大每一次重试的时间间隔。 在异步处理程序中，订单根据处理结果变更状态后，也要发消息通知相关系统。

#### **整体架构**

整体软件参考架构如下所示：

##### 1. 执行参数校验

所有的支付操作，都需要对输入执行参数校验，避免接口受到攻击。

验证输入参数中各字段的有效性验证，比如用户ID,商户ID,价格，返回地址等参数。

验证账户状态。交易主体、交易对手等账户的状态是处于可交易的状态。

验证订单：如果涉及到预单，还需要验证订单号的有效性，订单状态是未支付。为了避免用户缓存某个URL地址，还需要校验下单时间和支付时间是否超过预定的间隔。

验证签名。签名也是为了防止支付接口被伪造。 一般签名是使用分发给商户的key来对输入参数拼接成的字符串做MD5 Hash或者RSA加密，然后作为一个参数随其他参数一起提交到服务器端。

##### 2. 根据支付路由寻找合适的支付服务

根据用户选择的支付方式确定用来完成该操作的合适的支付渠道。用户指定的支付方式不一定是最终的执行支付的渠道。比如用户选择通过工行信用卡来执行支付，但是我们没有实现和工行的对接，而是可以通过第三方支付，比如支付宝、微信支付、易宝支付，或者银联来完成。那如何选择合适的支付渠道，就通过支付路由来实现。支付路由会综合考虑收费、渠道的可用性等因素来选择最优方案。

##### 3. 评估交易风险

检查本次交易是否有风险。风控接口返回三种结果：阻断交易、增强验证和放行交易。

阻断交易，说明该交易是高风险的，需要终止，不执行第5个步骤；

增强验证，说明该交易有一定的风险，需要确认下是不是用户本人在操作。这可以通过发送短信验证码或者其他可以验证用户身份的方式来做校验，验证通过后，可以继续执行该交易。

放行交易，即本次交易是安全的，可以继续往下走。

##### 4.生成交易订单

将订单信息持久化到数据库中。当访问压力大的时候，数据库写入会成为一个瓶颈。

##### 5. 调用支付渠道提供的服务

所有的支付服务都需要第三方通道来完成执行。一般银行渠道的调用比较简单，可以直接返回结果。一些第三方支付，支付宝，微信支付等，会通过异步接口来告知支付结果。

##### 6. 更新订单

对于同步返回的结果，需要在主线程中更新订单的状态，标记是支付成功还是失败。对于异步返回的渠道，需要在异步程序中处理。

##### 7. 发送消息

通过消息来通知相关系统关于订单的变更。风控，信用BI等，都需要依赖这数据做准实时计算。

##### 8. 异步通知

如上述流程，其中涉及到调用远程接口，其延迟不可控。如果调用方一直阻塞等待，很容易超时。引入异步通知机制，可以让调用方在主线程中尽快返回，通过异步线程来得到支付结果。对于通过异步来获取支付结果的渠道接口，也需要对应的在异步通知中将结果返回给调用方。 异步通知需要调用方提供一个回调地址，一般以http或者https的方式。这就有技术风险，如果调用失败，还需要重试。而重试不能过于频繁，需要逐步拉大每一次重试的时间间隔。 在异步处理程序中，订单根据处理结果变更状态后，也要发消息通知相关系统。

#### 整体架构

整体软件参考架构如下所示：

**支付网关前置**

支付网关前置是对接业务系统，为其提供支付服务的模块。它是所有支付服务接口的集成前置，将不同支付渠道提供的接口通过统一的方式呈现给业务方。这样接入方就只需要对接支付网关，增加和调整支付渠道对业务方是透明的。 支付网关前置的设计对整个支付系统的稳定性、功能、性能以及其他非功能性需求有着直接的影响。

在支付网关中需要完成大量的操作，为了保证性能，这些操作都尽量异步化来处理。

支付网关前置应保持稳定，尽量减少系统重启等操作对业务方的影响。支付网关也避免不了升级和重启。这可通过基于Nginx的LBS(Load Balance System)网关来解决。LBS在这里有两个作用： 一个是实现负载均衡，一个是隔离支付网关重启对调用的影响。 支付网关也采用多台机器分布式部署，重启时，每个服务器逐个启动。某台服务器重启时，首先从LBS系统中取消注册，重启完成后，再重新注册到LBS上。这个过程对调用方是无感知的。

为了避免接口受攻击，在安全上，还得要求业务方通过HTTPS来访问接口，并提供防篡改机制。防篡改则通过接口参数签名来处理。现在主流的签名是对接口参数按照参数名称排序后，做加密和散列，参考微信的签名规范。

**交易流水和记账**

每一笔交易都需要记录流水，并登记到个人和机构的分户账户上，统计和分析也需要根据交易流水来更新相关数据。 而个人和机构账户总额更新、交易流水记录以及库存的处理，更是需要事务处理机制的支持。 从性能角度， 可以弱化了事务处理的要求，采用消息机制来异步化和交易相关的数据处理。

在支付网关前置的主流程中，仅记录交易流水，即将当前的请求保存到数据库中。

完成数据记录后，发送MQ出来，记账、统计、分析，都是接收MQ来完成数据处理。

涉及到本地资金支付，比如钱包支付，会需要分布式事务处理，扣减账号余额，记账，扣减库存等，每个操作失败，都要回滚。阿里有很不错的分享，这里不详细描述。

当交易量上来后，需要考虑交易表的分表分库的事情。分表分库有两个策略，按照流水号或者交易主体id来走。后者可以支持按用户来获取交易记录。我们用的是前者。后者可以走elastic，确保数据库专用。风控，信用和统计所需要的数据，通过MQ同步到Hbase里面。作为支付系统最有价值的数据，在存储上做到专库专用，无可厚非，毕竟存储成本还是廉价的。

**风控模块**

风控对支付的重要性怎么强调都不过分。有些系统在风控出问题时可以旁路风控，但是在支付系统中，风控出问题必须停止交易。 整体上，风控可以分为数据采集，数据分析，实时计算，规则配置，实时拦截等模块。风控本身是个大话题，以后专门讨论。又欠一个债。 但风控和交易的接口比较简单。对每一次交易，风控一般返回三个结果:拦截，增强验证，通过。通过指交易没有问题，可以直接放行。拦截则是阻止本次交易。增强验证则是交易存疑，需要用户进一步核实身份才能继续，比如输入手机号或者身份证号，一般用于身份被盗用的场景。而人工核实则是对交易有疑问，一般用于个人恶意消费满场景。

**支付路由**

支付路由是一个复杂的话题。对支付系统来说，能支持的支付方式越多越好，不能由于支付方式的不支持断了财路。现实中的支付方式多得难以置信。用户随时甩出一张你听都没听说过的卡。如果一个银行卡只有几个用户在用，那针对这个卡开发个对接有点得不尝失。现在第三方支付的爆发，确实给开发支付系统省了不少事。但是公司不可能只对接一个第三方支付，如果这个渠道出问题了，或者闹矛盾了，把链接给掐了，老板还不欲哭无泪。总之，得对接多个渠道。对于交易量大的银行，还得考虑直联。

支付路由的作用是定义对用户选用的银行卡或者其他支付方式，使用什么渠道来完成支付。

一般来说，银行会提供两种支付途径：无跳转的快捷支付接口和带跳转的网银接口。前者在绑卡，支付的时候，不需要跳到银行页面上去处理，后者则需要在银行的网银页面上完成。显然前者对用户来说体验要好多了，用户流程不会被打断。快捷支付要求支付系统在本地保存用户的支付信息，如卡号，登记手机。系统要确保这些信息不被泄漏。风险非常好，所以大部分银行要求接入方必须经过ADSS检验才能够接入快捷支付。

这种固定方式的接入有单点故障的问题，一旦某个渠道出问题，绑定的支付方式就无法使用。改进策略是为每个支付方式定义多个渠道，第一个渠道出问题即选择第二个，以此类推。

当然，更进一步，可以为候选渠道定义权重，按照权重来选择支付方式。当渠道出问题，自动调整权重。

路由实现上还会更复杂，对同一张银行卡，运营上会要求在不同的系统上，比如android，iOS，windows上，或者不同地区，如大陆，台湾，香港，北美等，甚至不同业务上，采用不同渠道来支付。

**支付渠道**

如果采用微服务来实现，整体设计上，可以考虑将支付渠道分离、支付网关前置分离。支付渠道的微服务实现有两种策略，一种是按照服务来拆分，一种是按照渠道来拆分。

按渠道拆分，指每个渠道单独部署在一个容器中，对支付网关提供相同的服务。

按服务拆分，是按接口来拆分，分为支付，对账，退款等子系统，每个服务单独部署，所有容器的服务都实现在一起。

**渠道拆分**

按照服务来拆分的一个典型案例是大众点评网的早期实现。 大众点评支付渠道网关系统的实践之路。 每个支付服务接口实现为一个独立的子系统，独立部署，通过支付网关前置来对外提供服务。 这篇文章里面也提到这种方式存在的问题，

银行的加密客户端会有各种奇葩的需求，有些可以支持linux，有些要windows系统，如何在一个容器中满足所有需求？

这样拆分后，每个渠道接口独立部署。某个渠道出问题也不会影响其他渠道。至于渠道访问量小导致资源浪费问题，可以通过虚机或者docker的资源调度来解决，谁也不会在物理机上玩微服务。

对接渠道难点在于对输入输出做加密和解密，以及组装和解析报文。同一个渠道对不同的服务的加密解密方式是一样的，报文格式也是一样的。按渠道来构建服务可以共用这样方法，减少开发投入。

从安全的角度，按渠道划分也有优势。一般渠道都要求只对接到特定ip的机器，这样每个渠道对接系统所在的机器仅开放对渠道和支付网关前置机的访问白名单即可，尽可能的缩减被暴露的风险。

**接入渠道**

对于支付渠道，首先考虑的是接入哪些渠道。要对接的渠道按优先级有：

第三方支付，对大部分应用来说，支付宝和微信支付都是必须的，一般来说，这两者可以占到90%以上的交易量。用户不需要绑卡，授权后直接支付就行。各种平台都支持，性能和稳定性都不错。对于一些特殊业务，比如游戏，企业支付，可以查看一些专用的第三方支付平台。

银联，这货的存在，极大方便了和银行的对接。和第三方支付主要不同在两个地方一是需要绑卡，也就是用户先把卡号，手机，身份证号提供出来。这一步会折损不少用户。绑卡后，以后的支付操作就简单了，用户只需要输入密码就行。手机客户端不需要像第三方支付那样安装SDK，都在服务器端完成。当然，这是针对快捷支付。网银支付还是挺麻烦的。银联接入也需要ADSS认证。

银行，建议先看这一篇文章，了解下对接银行的难度。那最终需要选择哪些银行？先看个统计数据。 截至 2015 年底，我国银行业金融机构包括 5 家大型商业银行、12 家股份制商业银行、133 家城市商业银行、5 家民营银行、859 家农村商业银行、71 家农村合作银行、1373家农村信用社、1 家邮政储蓄银行、3 家政策性银行、 311 家村镇银行、48 家农村资金互助社。优先选择5家商业银行，他们占40%的交易量。其次是股份制银行和邮储。这就18家银行了。老板要是不满意，城商行和农商行加起来有1000多家呢。一般对接一个银行预计有3周左右的工作量，大部分银行需要专线接入，费用和带宽有关，一年也得几万费用。不同银行对接入环境有不同要求，这也是成本。另外，还有一个重大风险，就是央行在搞得网联系统，毕竟还没有出来，相关资料参考知乎上关于网联的一篇讨论。

手机支付，现在不少厂商都内置了各种支付，比如苹果的In-App支付， 三星支付、华为支付等， 这些支付仅针对特定的手机型号， 支持NFC等，根据业务需要也可以接入。 就是目前用户群不大，收益不明显。

话费支付， 这一块容易被人忽略，但考虑到国内不少职场人士，话费是公司报销的，每个月多的用不完，所以这块支付还是相当有市场的。 问题是，联通和移动两大运营商，不仅接口不能互通，内部各个地域也是各自为政，所以对接起来还是有点麻烦。不过话费支付领域也有类似支付宝微信的第三方支付公司，比如虹软、联动优势等公司。

## 系统的问题与反思