# 某站支付改版

目录



某站以二次元社区著称，15 年开启了用户端商业化之路，主营收入来自番剧、直播打赏、游戏联运等，各个业务为了快速迭代，自行定制实现支付系统，随着业务的迭代，原有的支付系统维护成本高到难以扩展。下文称为老支付，改版后的支付称为新支付。

笔者 17 年加入某站，作为技术人员，之前在支付系统的经验可谓寥寥，因此本文分享仅是个人针对改版过程的遇到的一些问题交流，难免有很多思考不完善处，请诸位专家们多多指点。

## 为什么要改版

老支付系统设计所有消费基于虚拟币 X 币系统，用户的一次支付，在内部系统实际经历的是充值并消费过程，且虚拟余额系统、支付网关系统完全耦合在一个服务中，难以拆解，系统经常会出现充值但未消费等情况。总结一下老支付问题：

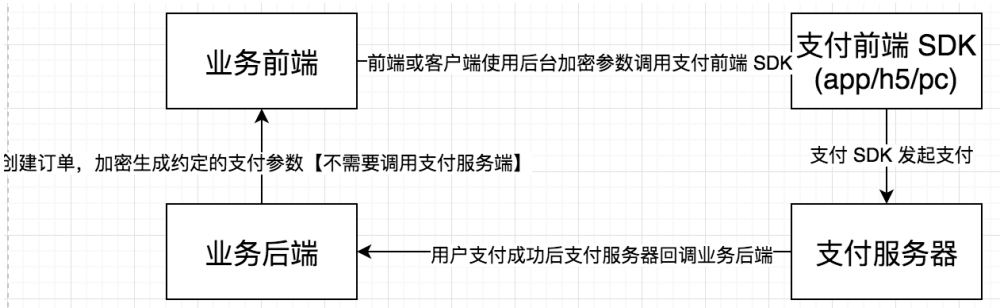
1. 没有网关系统概念，新增支付渠道开发完全独立，成本高。
2. 没有统一收银台概念，各业务自行开发收银台，支付系统迭代配置升级困难。
3. 所有支付、消费通过 X 币结算，不适合电商、周边、游戏等业务。
4. 业务接入需要同时注册业务与商品，类似 IAP 支付商品注册，制约业务新增商品迭代。
5. 系统组件陈旧、迭代升级困难

在此背景下，我们决定全新搭建一套全新的支付平台。

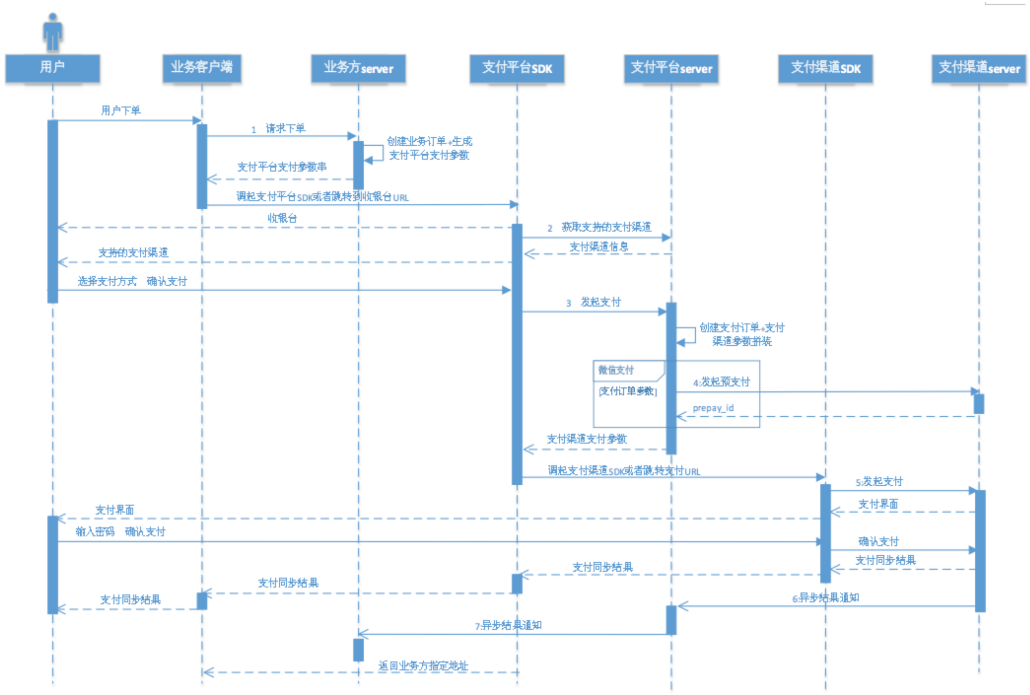
### 新支付聚合收银台设计

1. **SDK 模式**，为了统一支付体验，我们提供了三端收银台，业务直接调用收银台完成支付。采用类似支付宝的 SDK 直接下单功能，业务服务端不直接与支付服务端交互。这与老支付不同，老支付恰巧采用的是预下单模式。

- [业务简易交互图示]



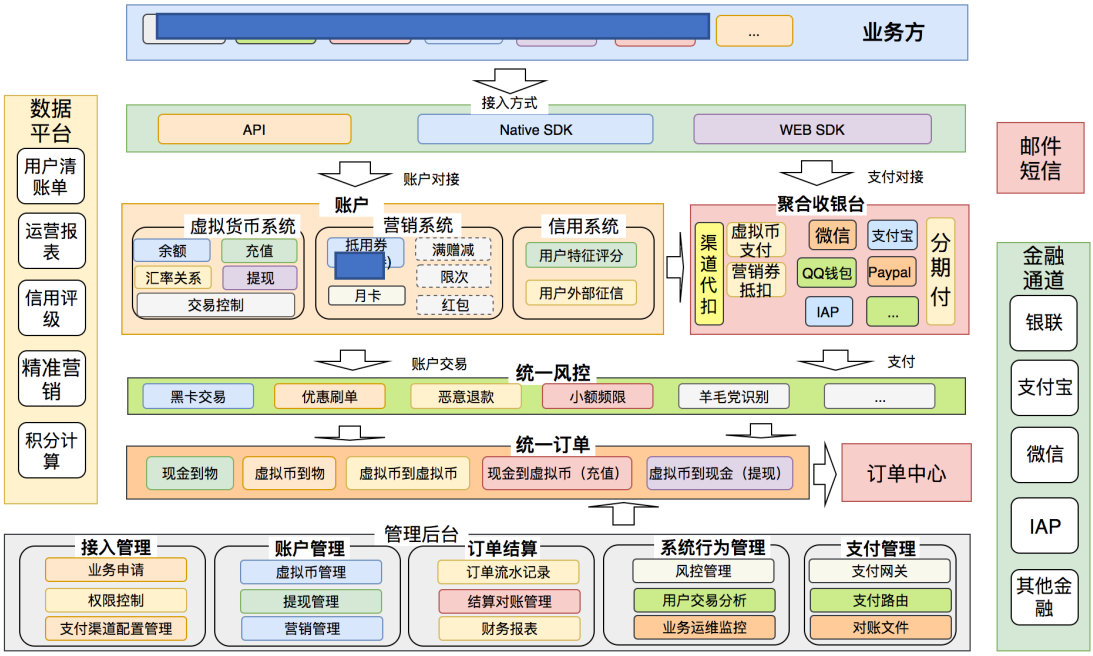

- [业务详细交互用例图]




这么做，我们考虑的好处是，将下单的成本交给 SDK 来进行迭代。未来支付服务访问相关的优化都交给 SDK 来完成。 有个前提是， 一旦业务接入并正常使用支付，未来的支付相关的迭代，让业务来配合一定是一个推动的过程。还不如一开始就完全收拢。

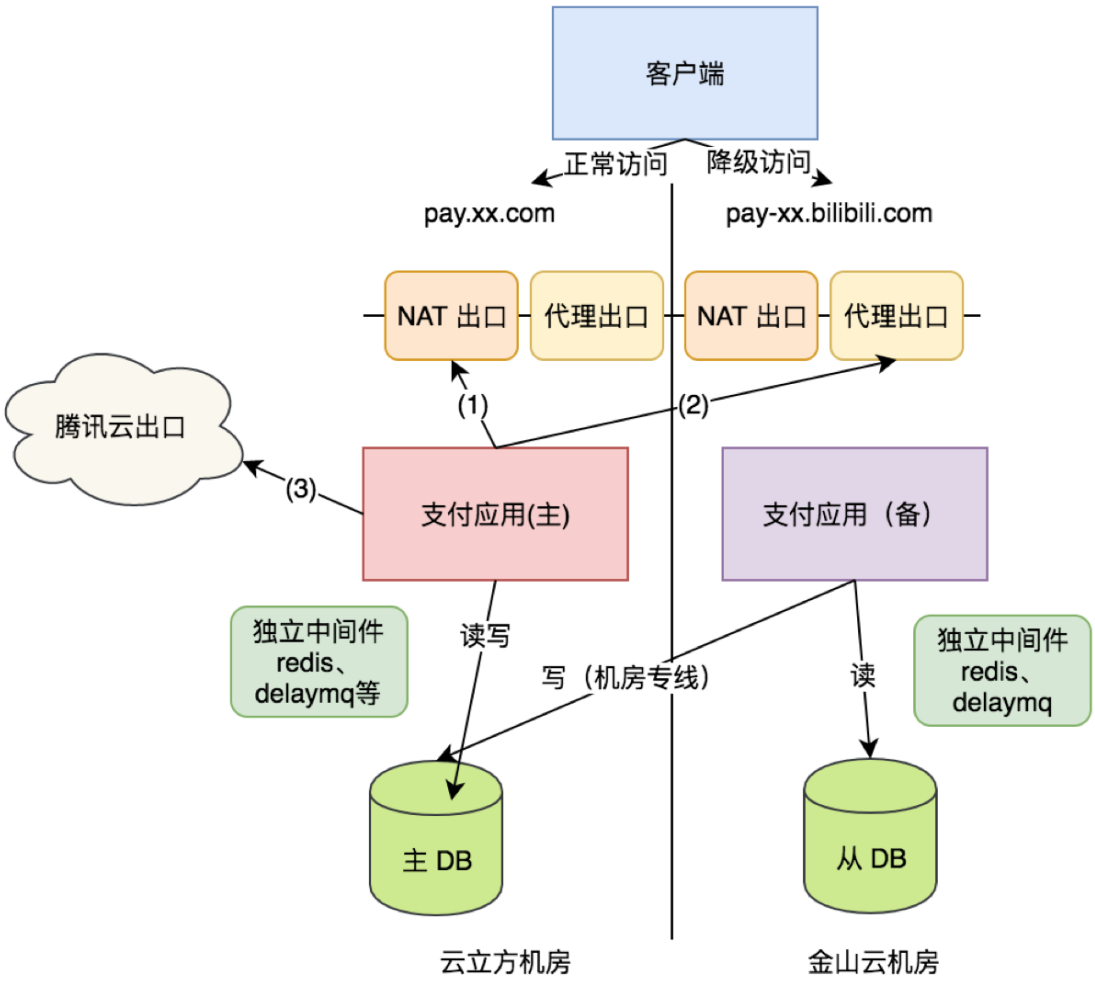
2. **流水单模型** 。一个业务订单对应多笔流水单，每笔流水单对应一笔渠道订单，一个业务订单同时只允许支付成功一次。聚合收银同时对接多个支付通道，意味着用户可能使用多个支付通道支付多次。为了降低业务关注成本，通过多支付退款，已支付成功不允许发起等方式保证一个业务订单只有一笔支付成功订单，业务无需关心通道情况。

【设计架构图】




## 遇到的一些问题以及处理方式

- 回调延迟问题
  三方渠道回调偶尔会因为线路抖动情况超时，比如支付宝，第一次回调失败，下一次通知在 4 分钟后。对于业务来说，就算偶发的一例回调超时，用户基本都会来进行投诉。
  在这里，我们做了三项工作，
  - 第一，联合支付宝排查了回调失败的原因，发现部署的部分 CDN 节点有一定几率丢失请求。摘掉这些故障节点后，回调超时有明显的改善。
  - 第二，增加主动查询渠道功能。主动查询渠道存在个问题是，支付发起后并不知道用户是否真的在第三方完成了支付，所以需要不停的轮询渠道。这会造成时效性和性能的均衡问题。过于频繁的查询会给系统造成压力；过于稀疏的又无法保证时效性。我们的解决思路是，统计过去不同渠道用户发起支付后到支付完成的时间分布，在分布高的时间阶段密集查询，分布低的时间稀疏查询，进而在牺牲一定时效性的情况下达到一个时效与性能平衡。
  - 第三，一般支付渠道 SDK 会有个同步回调，在同步回调中访问后台直接查询渠道完成支付，这个大大提升了回调的时效，用户体验较好。
- 基础设施稳定性差如何高可用
  某站的基础设施是自建机房，出口和专线可用性只有 99.9%, 这给支付可用性带来了很大的挑战。我们采用了双机房，三出口容灾的方式，将用户实际感知可用性提高到了 99.99% 以上。




- 苹果 IAP 支付
  虚拟类绕不过的坎是 IAP，IAP 因为其特点有很多细坑。
  - 无法知晓购买商品的用户信息，无法验证用户的实际购买 。使用苹果后台验证凭证的方式器可以验证支付凭证的有效性，但不能定位到到用户的某一次购买，因此需要将验证通过后的支付凭证和 transaction_id 存储，保证用户不会再来欺骗购买。
  - 无法退款，用户投诉较多，一般给用户建议向苹果直接申请退款，如果苹果不退，且用户反应强烈，尝试补偿权益或线下退款。退款的用户权限收回也是个难题，目前只能靠用户自觉。
  - 需要分清楚 IAP 的多种类型商品，消耗型，非消耗型，订阅型，自动订阅型，根据产品特点申请相应的产品类型，使用错误，苹果爸爸就可能会拒审。
  - 客户端凭证，掉单问题，IAP 从客户端下发凭证就意味一定会造成掉单。老支付的掉单率奇高，新支付则对掉单做了特别的优化，取得的良好的效果。主要工作有
    - 客户端绑定凭证缓存队列重试
    - 服务端接力客户端验证重试，拉取苹果 VPN 专线来提高验证效率。

对于已经掉单的 CASE，则专门提供客户补单系统，由客服验证用户的购买记录，苹果购买凭证给予补单。也发生过用户 PS 购买凭证后前来申请退款，但发生量较少，用户的补单行为会被作为风控因子计入用户流程。

    - universal app 申请支付或内购项一定要把 APP 加上，否则会被拒审。
    - 全名代充问题，有些非法机构通过黑卡盗刷，票据拦截的方式欺骗苹果和商户来完成 IOS 的充值。我们发现后迅速成立的风控小组，针对此类行为进行分析，并联合苹果开发出了一套风控系统来防御非法黑卡代充的 CASE。具体策略不表。
- paypal
    - paypal 支付采用同步回调方式，部分用户会利用银行漏洞，比如某德国银行 paypal 支付完成后， paypal 会返回验证成功，但子状态为 pending, 代表正在和银行沟通，老支付之前有个漏洞就是直接将 paypal 大状态作为是否支付成功依据，导致用户利用银行与 paypal 结算间隙撤回扣款申请造成的商户损失。
- X 币 余额系统微服务化
    - 新支付与老支付最大的一个区别是，新支付将余额系统独立剥离，作为支付网关的一个渠道，余额系统的两项指标，1. 多用户下的交易 qps, 2. 单用户的交易 qps。 1 很容易做，2 不容易，恰巧我们有单用户的场景，目前单用户最大能做到 50 qps, 更高的提升方案正在尝试。通过 redis 缓存扣减，定时刷新进数据库，这存在 redis 故障的风险。如果 redis 故障，恰好变化的余额还未能同步进数据库。一方面提高 redis 可用性，另外一方面每次记录日志和余额变动消息，一旦 redis 失效，我们将会迅速从日志和余额变动消息中恢复余额。带来的缺陷是 redis 故障时无法支撑高 qps, 但考虑的 redis 本身的可用性，我们觉得这个问题可以接受。

改版后话

由于的支付经验非常有限，很多问题是在边摸索边解决的道路上前行，对于很多问题的考虑和思考都有很多欠缺，感谢老熊的公众号给予的指点。欢迎各位多多指正