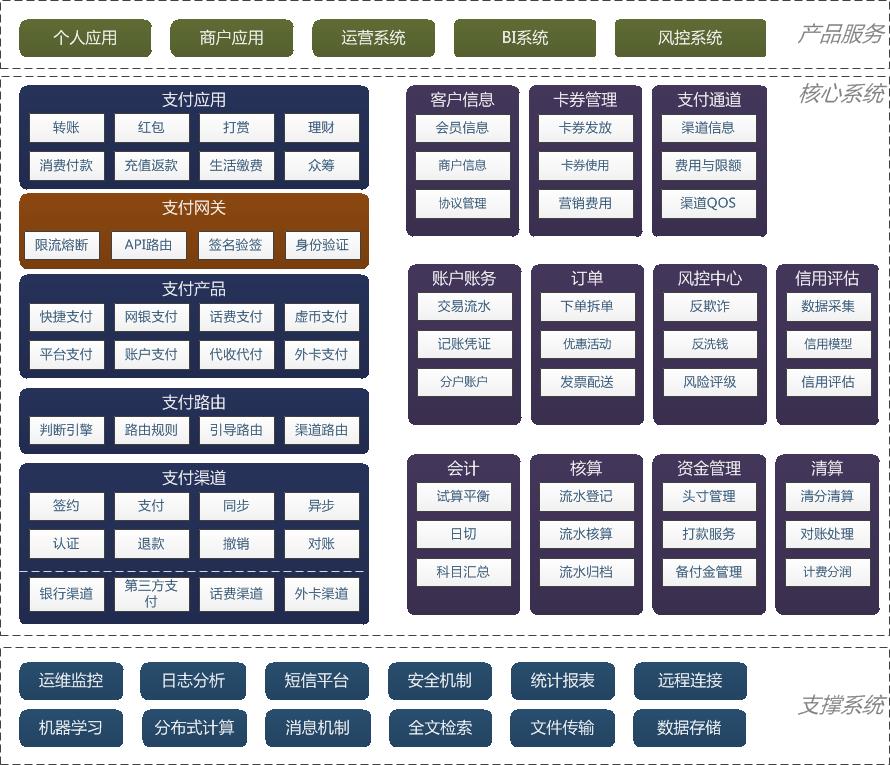
# 支付网关

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。

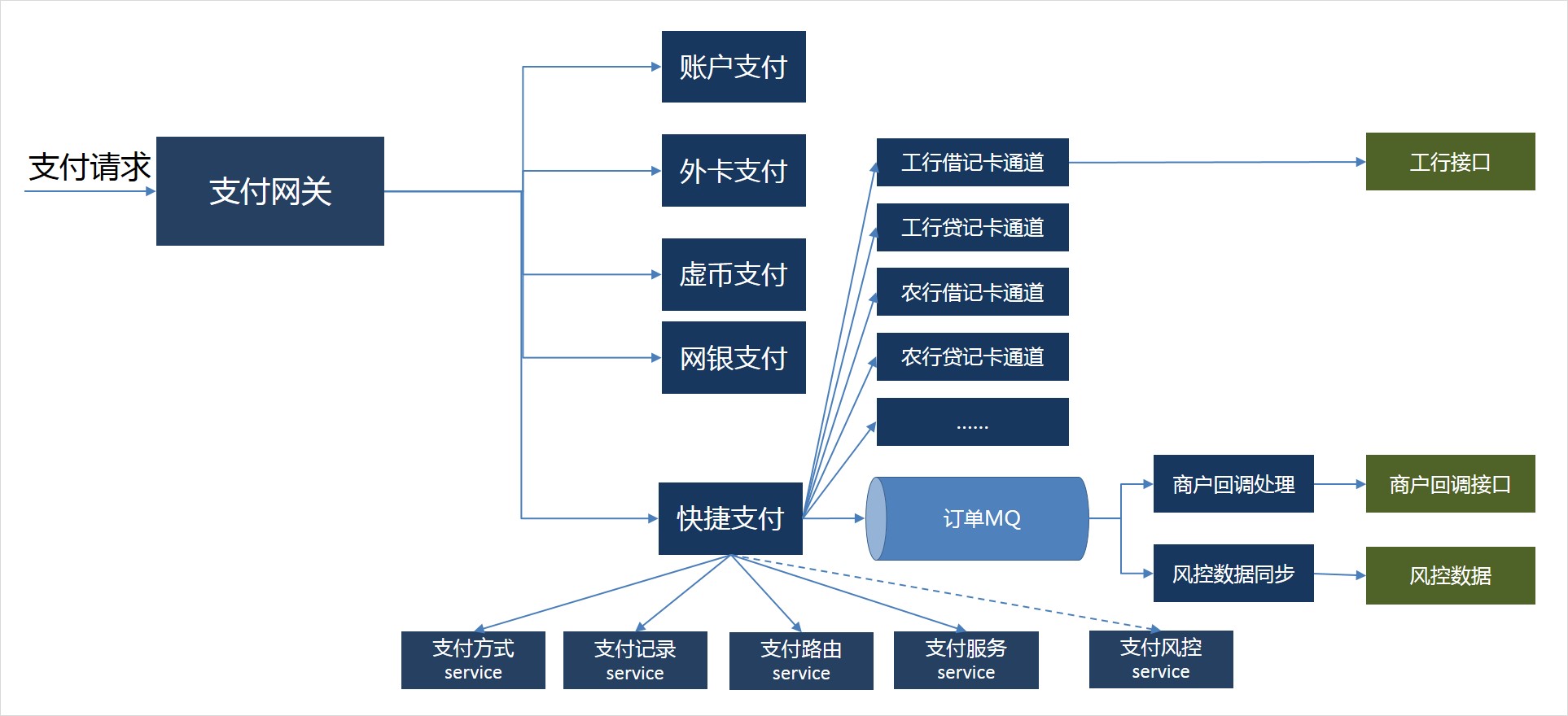
支付网关在支付系统参考架构图中的位置如下图所示



## 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。

一般来说，支付主流程会涉及到如下模块：



1. 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。
2. 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。
3. 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。
4. 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。
5. 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。

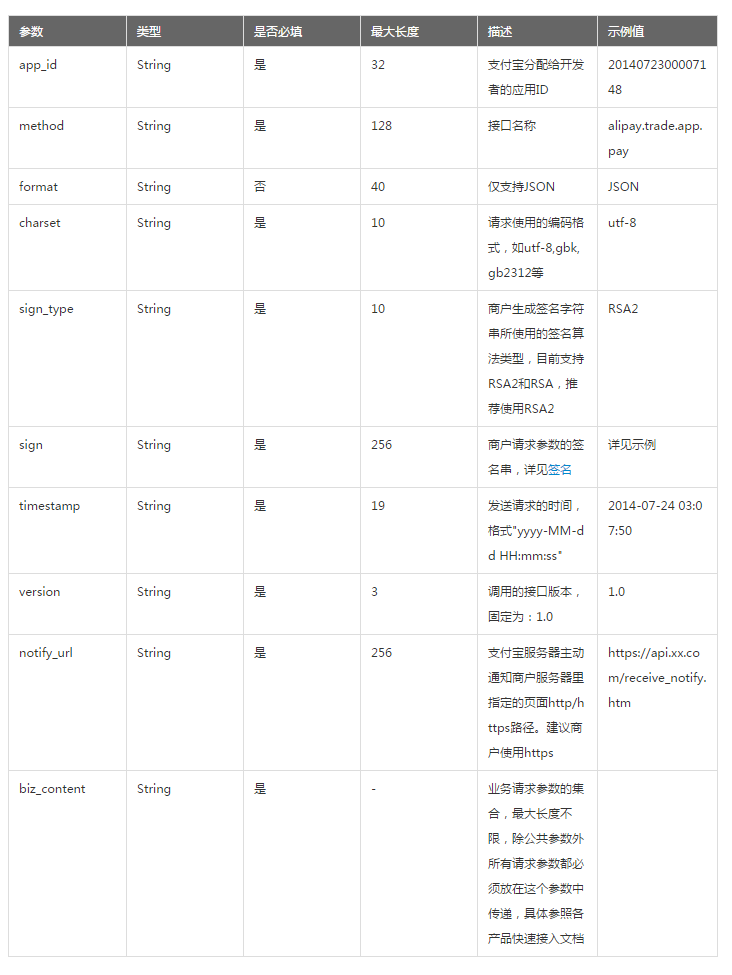
## 支付(API)网关

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。

[](http://static.cocolian.cn/img/in-post/gateway-alipay.png)

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。

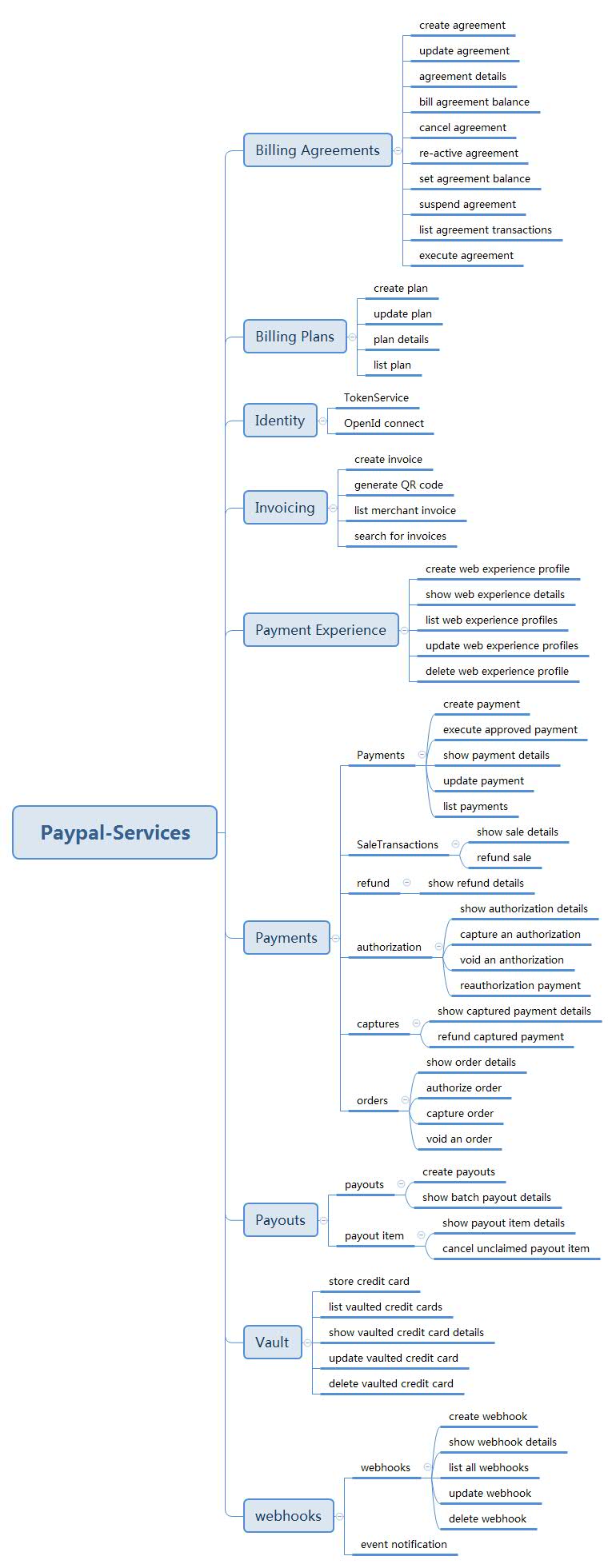
*支付宝的接口混合json格式和query string格式，在参数命名上，既有下划线方式的，也有驼峰的。英文单词的使用也不太规范。期待后续版本能做的更好。*

### **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。[](http://static.cocolian.cn/img/in-post/gateway-paypal.jpg)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。

## 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：

1. 按照支付能力来划分支付产品。
2. 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。
3. 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：

1. 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。
2. 接口安全： 熔断、限流与隔离。 这对支付服务来说尤为重要。 这是微服务架构的基本功能，本文不做描述。

如下功能，是在支付产品中提供：

1. 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。
2. 支付路由： 路由也是和产品有关。不同产品路由策略也不同。
3. 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。
4. 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：

1. 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。
2. 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。

## 签名和验签

API路由、接口安全这两块内容是微服务的基本模式，本文不再介绍。有兴趣同学可以参考相关资料。这里重点说下支付所必须的签名和验签。   
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。

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。

1. 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。
2. 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。

加密的一个通用过程是：

1. 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

mch\_id： 10000100

device\_info： 1000

body： test

nonce\_str： ibuaiVcKdpRxkhJA

这种请求，将被拼接为：

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

1. 使用RSA对字符串进行签名，生成签名字符串。

cYmuUnKi5QdBsoZEAbMXVMmRWjsuUj**%**2By48A2DvWAVVBuYkiBj13CFDHu2vZQvmOfkjE0YqCUQE04kqm9Xg3tIX8tPeIGIFtsIyp**%**2FM45w1ZsDOiduBbduGtRo1XRsvAyVAv2hCrBLLrDI5Vi7uZZ66Lo5J0PpUUWwyQGt0M4cj8g**%**3D

3.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。