23.图解支付报文网关：一种低代码报文网关的设计思路与核心代码实现_V20240201

所有的支付系统都对接了很多的外部支付、流出、外汇等各种类型的通道，这些通道的接口和报文格式各异。今天和大家一起聊聊如何实现一种简洁高效的低代码报文网关设计，主要包括：报文网关的定位，三种形态，低代码报文网关的设计思路，系统架构，核心代码实现。

如果你做过支付系统并写过脚本或代码对接过通道（渠道），或者你好奇如何通过低代码来对接外部千奇百怪的渠道，欢迎一起探索。

1. 前言

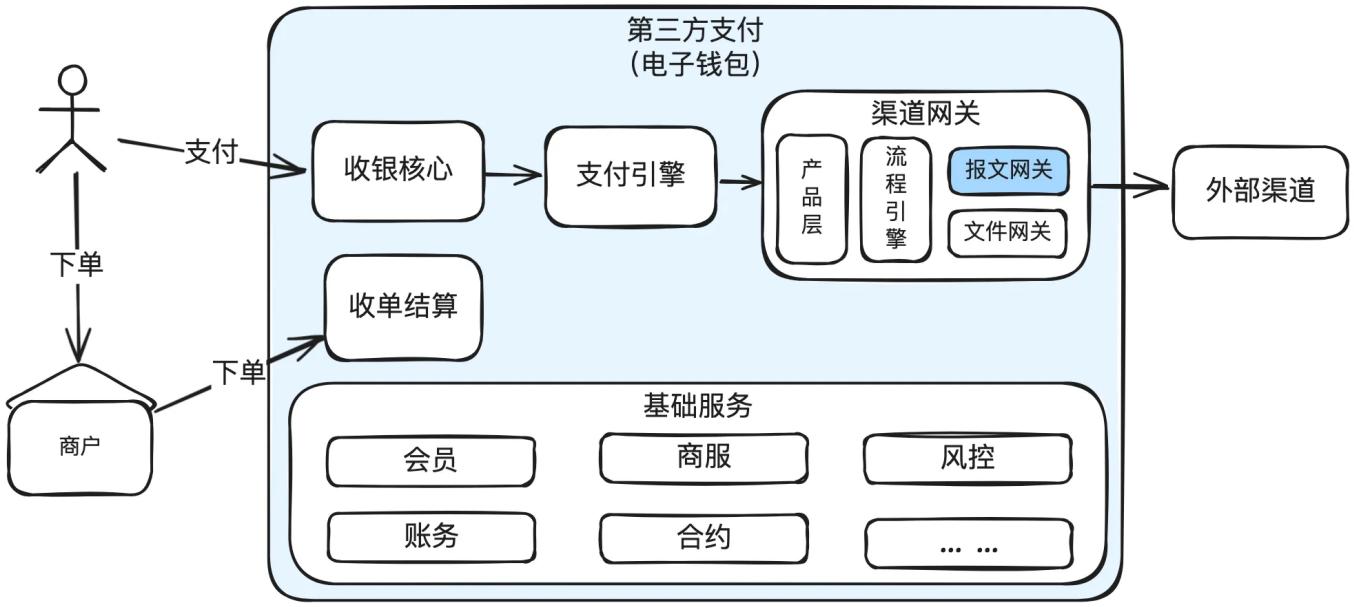
在数字支付领域的深处，存在着一个关键的、却往往被忽视的英雄——报文网关。作为支付系统与外部世界沟通的桥梁，报文网关承担着**参数转换、报文组装与解析、安全加密、签名验签**等多重重要任务。

一般来说，小型公司可能根本就没有报文网关这一说法，直接引入HttpClient包，手撸几个类，就把一个渠道对接搞定。稍大的公司，可能做一些模板方法的抽象，或者一些组件的抽象，也能实现一定的高效接入及复用。但对于更大型跨国公司，如果接入的渠道有几百条，这样手写接入渠道，往往伴随着代码高复杂性和高维护成本。因应这一挑战，"低代码报文网关"的概念应运而生。

在本文中，我们将一起探索这种低代码报文网关的创新设计。我们会从报文网关在支付系统中的角色和重要性入手，然后深入探讨低代码报文网关的工作原理、产品架构、系统架构以及核心代码实现。我们的目标不仅是理解其技术细节，更是领悟其背后的设计哲学——如何在保证系统强大功能的同时，实现更高的接入效率和可维护性。

这篇文章旨在为广大支付技术从业者、软件开发者以及对支付系统感兴趣的读者们，提供一个全新视角来理解和应用低代码报文网关。

2. 报文网关在支付系统中的定位



报文网关最核心的职责就是**对接外部渠道的API接口，**把内部的请求发出去，把渠道返回的数据转成内部的参数。

这里面还涉及到很多技术细节，比如参数转换、签名验签、加密解密、报文组装解析，发送接收等。

在前面的两篇文章中，我们介绍了渠道网关，两者的区别在于：

渠道网关：是一个更大范围的网关，还包括渠道路由、渠道开关、渠道咨询等能力。

报文网关：是渠道网关的一部分，**只负责对接渠道的接口，**小团队可能只是一个小模块，大团队可能会独立出应用。

3. 报文网关的几种形态

一般来说，从简单到复杂、从固定到灵活，报文网关会存在四种形态：

1. **纯手撸代码**：

- 在这种最初级的形态中，每个外部渠道都需要单独的代码实现。这意味着为每个新接入的银行或支付服务，开发团队需要编写一套新的接口逻辑。
- **优点**：针对性强，可以精确控制每个渠道的交互细节。
- **缺点**：随着接入渠道的增多，代码变得越来越复杂，维护和扩展的成本急剧上升。

2. **模板方法报文网关**：

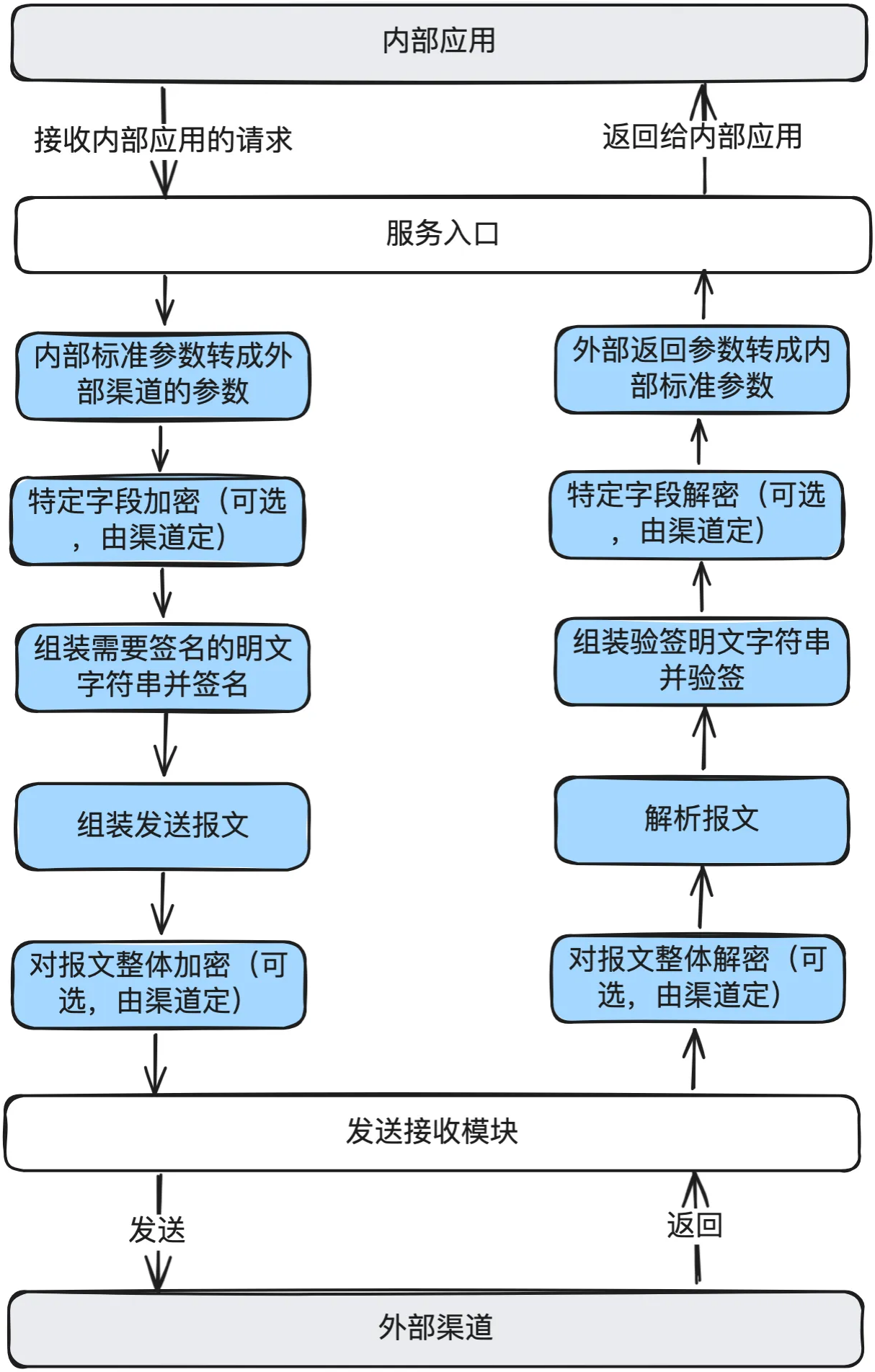
- 这种形态通过引入模板方法模式，将报文处理流程的共通部分抽象出来，为各个渠道提供统一的处理框架，同时留有接口供具体渠道实现其特定逻辑。
- **优点**：提高了代码的重用性，降低了维护成本。
- **缺点**：对于一些特殊需求，模板方法可能仍然不够灵活，需要额外的定制。

3. **低代码报文网关**：
    - 低代码报文网关把所有核心的代码逻辑实现后，只需要写几个配置文件，就可以完成渠道的接入。
    - **优点**：极大地提高了灵活性和易用性，加快了新渠道的接入速度，核心代码由有经验的资深员工编写，减少出错可能性。
    - **缺点**：复杂场景下可能需要写一些内联函数，造成了一定复杂度。

4. **产品化配置报文网关**：
    - 在低代码报文网关基础上，提供图形化配置界面，进一步降低使用难度。
    - **优点**：极大地提高了易用性，加快了新渠道的接入速度，以前写代码可能需要4、5天才接一个接口，变成可能0.5天就能接一个接口。
    - **缺点**：平台的初始研发成本很高，如果总体接入的渠道不多，ROI可能不高。

每种形态都反映了各公司在特定时期的技术水平和方案造型，但总体来说，对于中大型公司来说，低代码报文网关和产品化配置报文网关是一个比较不错的选择。一方面可以提高效率，另一方面也有足够的研发资源来建平台。

4. 一种低代码报文网关设计思路


内部应用
接收内部应用的请求
返回给内部应用
服务入口
内部标准参数转成外部渠道的参数
特定字段加密（可选，由渠道定）
组装需要签名的明文字符串并签名
组装发送报文
对报文整体加密（可选，由渠道定）
外部返回参数转成内部标准参数
特定字段解密（可选，由渠道定）
组装验签明文字符串并验签
解析报文
对报文整体解密（可选，由渠道定）
发送接收模块
发送
返回
外部渠道

首先我们要知道报文网关核心只做这么几件事：

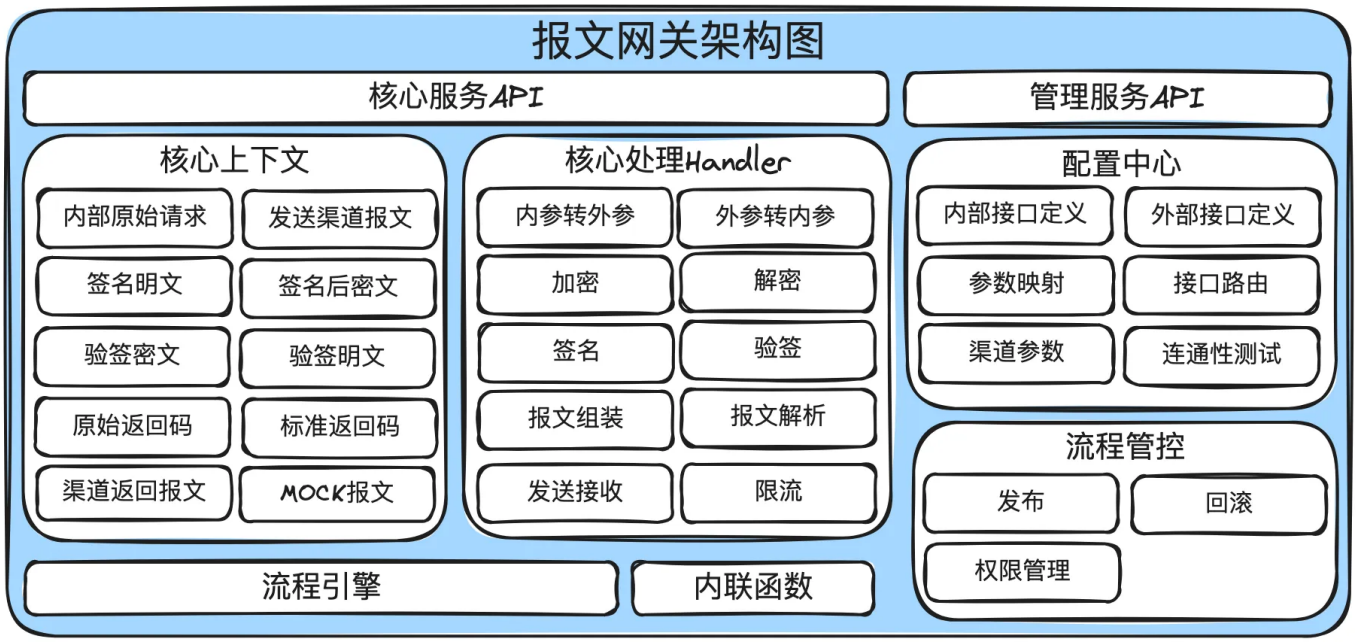
1. 接收内部应用的请求。
2. 内部标准参数转成外部渠道的参数。比如内部叫mobileNo，渠道可能叫：mobile_id。
3. 特定字段加密（可选，由渠道定）。
4. 组装需要签名的明文字符串并签名。
5. 组装发送报文，比如有json，kv，xml等。
6. 对报文整体加密（可选，由渠道定）。
7. 发送请求，并接收请求。
8. 对报文整体解密（可选，由渠道定）。
9. 解析报文。
10. 组装验签明文字符串并验签。
11. 特定字段解密（可选，由渠道定）。
12. 外部返回参数转成内部标准参数。
13. 返回给内部应用。

通过上面的流程分析，我们很容易想到流程引擎或责任链处理，再加一个上下文，就可以实现全部的操作。

具体包括：

1. **可视化配置**：提供直观的用户界面，允许用户通过图形化方式配置报文的处理流程，无需编写复杂的代码。
2. **模块化处理逻辑**：将报文处理的各个步骤模块化，如报文解析、加密/解密、数据映射等，用户可以根据需要组合这些模块来构建完整的处理流程。
3. **灵活的适配能力**：支持灵活地适配不同的支付渠道，包括但不限于银行接口、电子钱包、第三方支付平台等。
4. **动态配置管理**：所有配置信息动态管理，支持实时更新，无需重启系统或重新部署。

5. 低代码报文网关的系统架构



说明：

1. 最上层是API：核心服务和管理服务分离。
2. 核心上下文：主要是用于保存接口配置，原始请求，中间处理结果等。
3. 核心处理Handler：参数转换、加密、解密等全部基础功能全部组件化。
4. 流程引擎：负责串起所有的Handler依次运行。
5. 内联函数：解决特殊场景下的报文转换。
6. 配置中心：配置内部接口，外部接口，参数映射等。
7. 流程管控：发布、回滚、权限管理等能力。

通过这种系统架构，低代码报文网关不仅能够提供强大灵活的配置能力，还能确保处理流程的稳定执行，同时保持高度的可维护性。这样的架构使得报文网关能够适应不断变化的业务需求，同时降低了总体的维护成本和技术复杂性。

6. 低代码报文网关的核心代码实现

代码实现有很多种方式，且报文网关也是一个独立的应用，把所有的代码展示出来也不太现实。这里给出上面系统架构图中的核心代码示例。读者可以扩展实现细节，有兴趣的同学也可以私信我。

整体思路如下：

1. 设计数据库表结构：用于存储内部接口定义，外部接口定义，接口映射关系，外部渠道参数等。这部分在这里略过。
2. 设计运行上下文：GatewayContext（保存处理的上下文信息，包括接口配置、内部请求参数、加解密临时数据、加验签临时数据、发送渠道报文、渠道返回原始报文、解析后报文等）。
3. 设计各种处理Handler：内部参数转外部参数Handler，加密Handler，签名Handler，报文组装Handler，发送Handler，报文解析Handler，验签Handler，解密Handler，外部参数转内部参数Handler等。
4. 设计Handler工厂类：GatewayHandlerFactory（通过getHandlers()获取处理的责任链）。
5. 设计内联函数：用于处理一些复杂操作。比如外部渠道返回的币种和金额两个字段转成内部的一个Money类。这部分在这里略过。

下面是核心代码组件包括`GatewayContext`、`GatewayHandlerFactory`和一系列的`GatewayHandler`基本实现：

`GatewayContext`类用于保存处理过程中的上下文信息，包括临时参数、报文信息等。

```java
public class GatewayContext {
    // 接口配置信息
    private InterfaceInfo interfaceInfo;
    // 原始请求
    private Map<String, Object> originalRequestParam;
    // 转换后参数
    private String requestParam;
    // 签名明文
    private String signPlanContext;

    // 省略部分参数，以及getter和setter方法
    ... ...
}
```

`GatewayHandlerFactory`是一个工厂类，用于构建处理责任链。

```java
public class GatewayHandlerFactory {
    private static final List<GatewayHandler> handlers = new ArrayList<>();

    static {
        handlers.add(new ParameterTransformHandler());
        handlers.add(new EncryptionHandler());
        handlers.add(new SignatureHandler());
        // 添加其他handler
        ... ...
    }

    public List<GatewayHandler> getHandlers() {
        // 添加其他handler
        return handlers;
    }
}
```

`GatewayHandler` 接口定义了处理逻辑的执行方法。

```java
public interface GatewayHandler {
    void execute(GatewayContext context);
}
```

具体的 Handler 实现：

内部参数转外部参数Handler

```java
public class ParameterTransformHandler implements GatewayHandler {
    @Override
    public void execute(GatewayContext context) {
        // 实现内部参数到外部参数的转换逻辑
    }
}
```

签名Handler

```java
public class SignatureHandler implements GatewayHandler {
    @Override
    public void execute(GatewayContext context) {
        // 不需要签名
        if (!context.isNeedSign()) {
            return;
        }

        context.setSignMessage(sign(context.getSignPlainContext(),
                                    context.getInterfaceInfo().getSignConf
ig()));
    }

    // 签名方法略
}
```

其它Handler的实现略。

执行Service

```java
public class GatewayServiceImpl implements GatewayService {
    @Override
    public GatewayResponse execute(GatewayRequest request) {
        // 转成网关的上下文
        GatewayContext context = buildGatewayContext(request);

        // 获取责任链，依次执行
        List<GatewayHandler> handlers = GatewayHandlerFactory.getHandlers();
        for(GatewayHandler hander : handlers) {
            hander.execute(context);
        }

        // 转换返回
        return convertResponse(context.getResponseParam);
    }

    // 其它代码略
    ... ...
}
```

这些组件和处理器共同构成了低代码报文网关的核心功能，允许系统灵活地配置和处理支付系统与外部渠道之间的报文交换。通过这种设计，报文网关可以轻松适应不同支付渠道的接入和业务流程的变更，同时大大减少了传统编码方式所需的开发和维护工作。

7. 结束语

在本文中，我们深入探讨了报文网关在支付系统中的重要性，从其在支付系统中的定位、不同形态的发展，到一种具体的低代码设计思路，以及详细的系统架构。我们还看到了核心代码的实现，展示了如何通过灵活的处理器和上下文管理，实现报文网关的关键功能。

产品化配置的低代码报文网关通过提供直观的配置界面和强大的后端处理能力，使得支付系统更加灵活，能够快速适应新的支付渠道和业务模型。同时，它也降低了技术门槛，使得初始技术人员能够更容易地参与到支付渠道的对接中，而不用担心技术太菜可能导致的各种各样的问题。

这是《百图解码支付系统设计与实现》专栏系列文章中的第（23）篇。和墨哥（隐墨星辰）一起深入解码支付系统的方方面面。

系列文章PDF合集，不定时更新：

Github：https://github.com/yinmo-sc/Decoding-Payment-System-Book

百度网盘：https://pan.baidu.com/s/1l5dlR8SoGH_ly_8Hbv8mXQ?pwd=0000

公众号：隐墨星辰。

有个小群不定时解答一些问题或知识点，有兴趣的同学可先加微信（yinmo_sc）后进入，添加微信请备注：加支付系统设计与实现讨论群。