24.图解支付文件网关：文件交互的底座_V20240217

在上一篇《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》，我们深入讨论了报文网关的设计与实现，今天和大家聊聊文件网关的设计与实现。

在这篇文章中，你可以了解到文件网关的作用，什么情况下文件网关需要独立部署，文件网关的几种常见形态，如何抽象通用的模块进行复用。

1. 前言

在支付系统中，和外部银行通道一般有两种信息交互模式：

1. **实时报文交互：**组装指定的格式（JSON，KV等）报文实时发给外部银行通道，银行一般是实时返回结果，或异步通知返回结果。现在一般是通过https进行传输，以前还有直接通过socket传输，现在很少见到。

   实时报文交互一般用于支付通道的支付、退款、签约、解约等。

2. **文件交互**：组装文件，发给外部银行通道，或者外部银行通道生成文件，供我方下载处理。一般是通过SFTP进行传输，有些特殊的是通过邮件或https进行传输。

文件交互一般用于支付通道的清算文件，流出转账渠道的流出指令等。

2. 术语

1. 文件交互

文件交互是指支付平台与银行或其他金融机构之间通过文件形式进行数据交换的过程。与实时接口（如API调用）相比，文件交互通常用于处理批量指令，实时性要求不高。常见的场景包括银行提供的清算文件处理、流出转账的批量指令等。

2. 文件接口

文件接口是指支持文件格式数据交换的通信界面。在支付系统中，文件接口允许系统批量导入或导出数据，如转账指令、清算数据等，通常以CSV、XML或其他特定格式的文件进行数据传输。

3. 批处理

批处理是指将一组数据或指令作为一个批次进行处理的过程。在支付系统中，批处理常用于处理大量类似的交易或数据更新任务，如夜间批量处理银行清算文件或执行批量转账指令。

4. 批转单

批转单是将一批数据或指令分拆成单条进行处理的过程。比如外部银行的一个清算文件通常包含了几百万条数据，把这些文件的内容解析后，一条条发给内部系统进行处理，就是批转单的应用。

5. 单转批

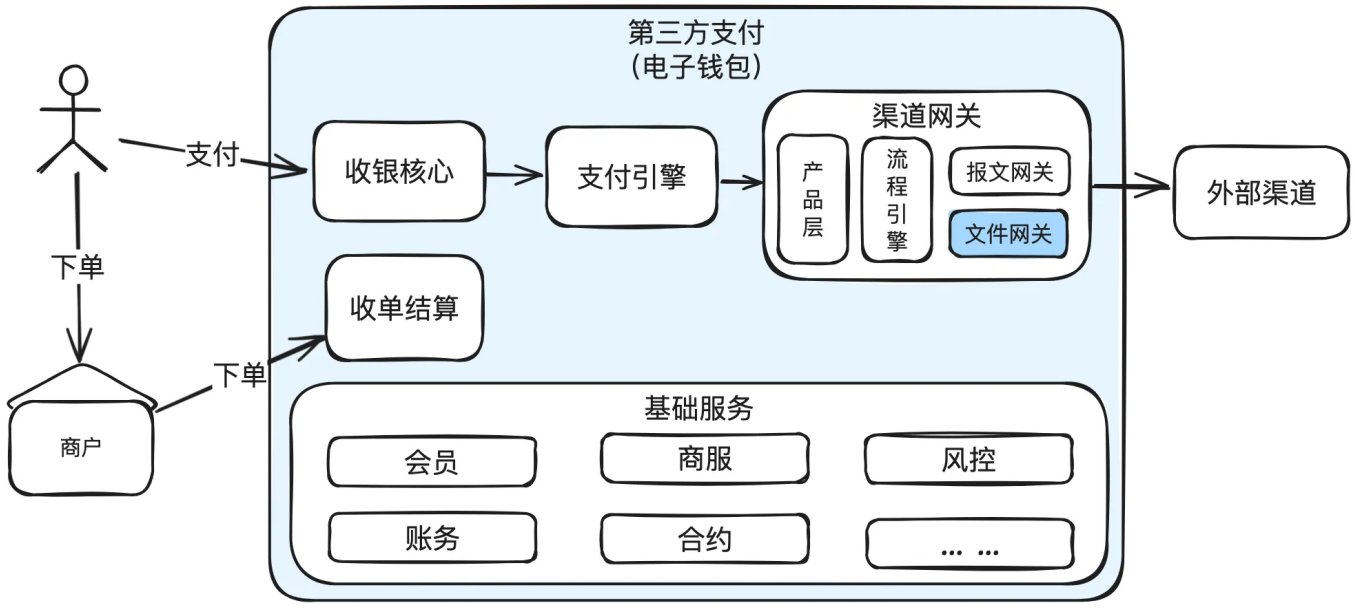
单转批是指将多个单独的指令聚合成一个批次进行统一处理的过程。比如流出转账指令，上游发下来的是一条条单个转账指令，把这多个指令汇总成一个文件发给外部银行，就是单转批的应用。

6. PGP加解密

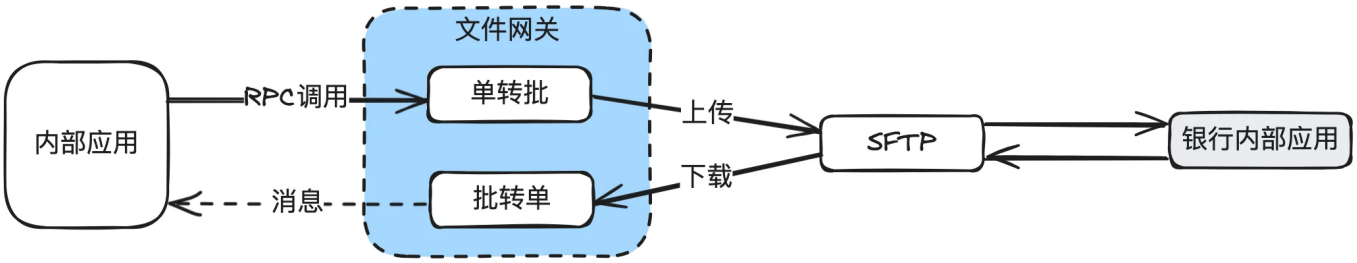
Pretty Good Privacy，是一种加密程序，用于信息传输的加密和解密，保证电子数据的安全性。它通过使用公钥加密和私钥解密的方法，实现了信息的机密性和完整性，同时也支持数字签名以确保信息来源的真实性。

有个形象的说法，PGP就像一个信封一样，不但加密保护里面的信息，还通过签名确定真实的来源。

3. 文件网关在支付系统中的地位



文件网关最核心的作用就是解决文件交互的问题。



再细化分为两种：

1. 单转批：把内部指令，打批成一个文件，然后上传到外部银行的SFTP上。
2. 批转单：把外部银行生成的文件，下载后并解析一条条内部需要的数据，然后发给内部应用处理。

4. 文件网关在什么情况下需要独立应用

在一些小型支付公司，一般没有独立的文件网关，可能只是一个小模块，正常也是能运行的。

但是对于大型支付公司来说，和外部银行通道一天的交易可能有几千万笔，一个清算文件有上百M，如果和实时交易系统共用应用，有可能会影响实时交易（比如支付业务），在这种情况，就需要把文件网关独立出去。也就是实时交易和非实时交易拆分开，互相不影响。

5. 文件网关的几种形态

一般来说，从简单到复杂、从固定到灵活，文件网关和实时报文网关会一样，也存在四种形态：

1. **纯手撸代码**：
- 在这种最初级的形态中，每个外部文件都需要单独的代码实现。
- **优点**：针对性强，可以精确控制每个文件的交互细节。
- **缺点**：随着接入通道的增多，代码变得越来越复杂，维护和扩展的成本急剧上升。

2. **模板方法文件网关**：
- 这种形态通过引入模板方法模式，将文件处理流程的共通部分抽象出来，提供统一的处理框架，同时留有接口供具体文件实现其特定逻辑。
- **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. **低代码文件网关**：
    - 低代码文件网关把所有核心的代码逻辑抽象出来，每个文件只需要写一个配置文件，就可以完成文件的对接。
    - **优点**：极大地提高了灵活性和易用性，加快了新文件的接入速度，核心代码由有经验的资深工程师编写，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。

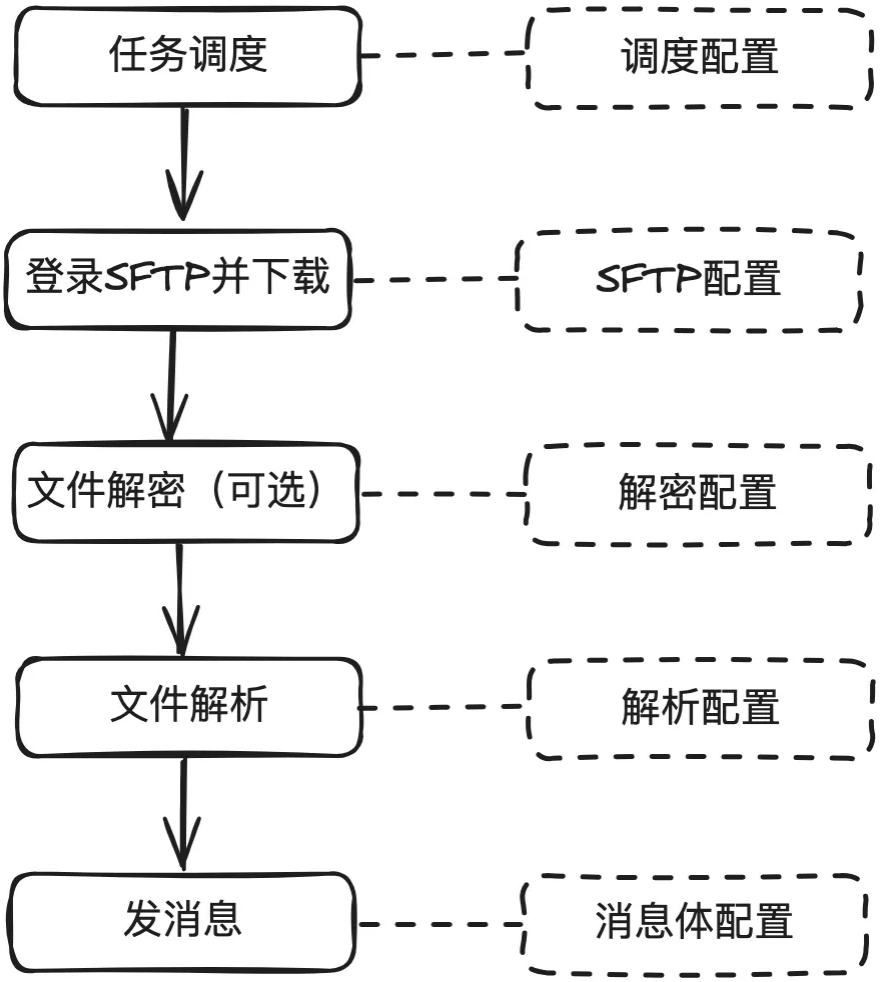
4. **产品化配置文件网关**：
    - 在低代码报文网关基础上，提供图形化配置界面，进一步降低使用难度。
    - **优点**：极大地提高了易用性，加快了新文件的接入速度，以前写代码可能需要4、5天才接一个文件，变成可能0.5天就能接入一个文件。
    - **缺点**：平台的初始研发成本很高，如果接入的文件不多，ROI就不够高。

每种形态都反映了各公司在特定时期的技术水平和方案选型，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，减少出错的

可能，另一方面也有足够的研发资源来建平台。

6. 低代码文件网关的设计思路

看过《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》的同学可能知道，实时报文的核心处理流程是固定的，其实文件网关的处理流程也是固定的。下面以文件下载为例说明设计思路，文件上传逻辑也是差不多的。



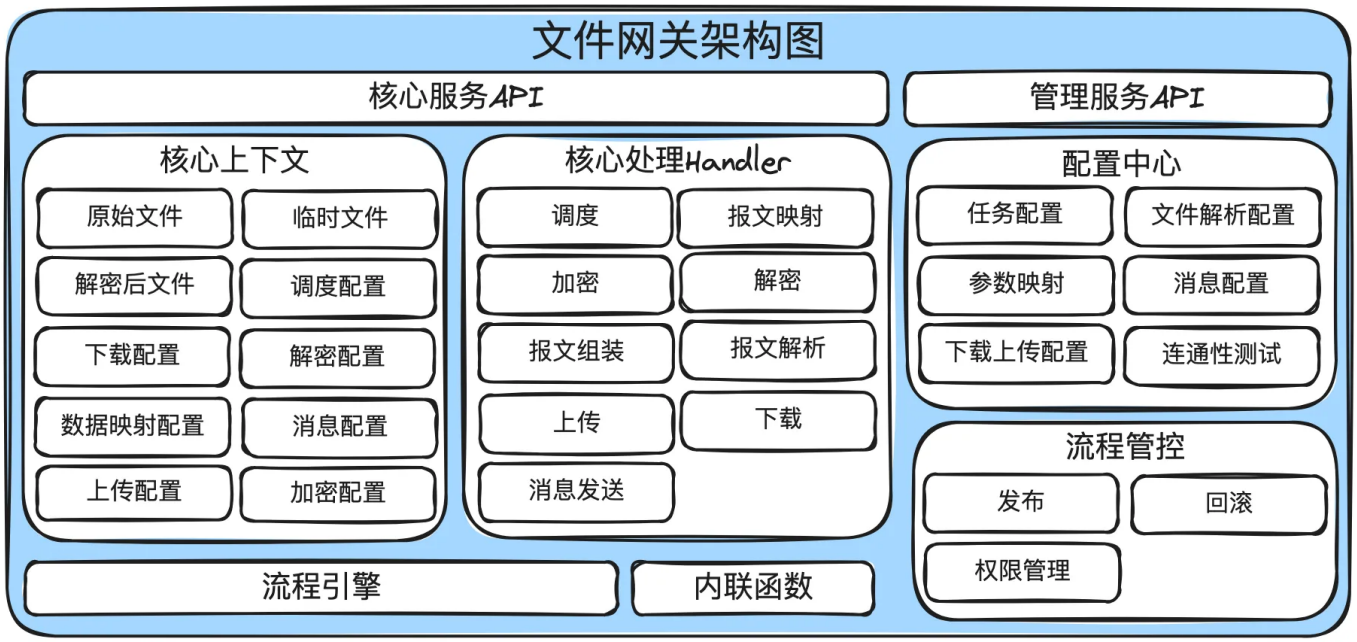
从上图可以看到，标准流程只有5个：

1. **任务调度**：一天的什么时候执行，执行多少次等。一般使用CRON表达式来做。比如银行的清算文件每天凌晨5点生成，那么一般设计任务在5点到7点之间执行，每30分钟尝试一次。
2. **登录SFTP并下载**：一般是SFTP，也有可能是邮件，设计成接口就行。如果是SFTP，一般SFTP的主机名，用户名，密码，目录，文件名规则等。其中文件名一般是每天变化，建议使用正式表达式来做。
3. **文件解密**：这个一般是可选的，有些银行可能会使用PGP加密，这种情况下，就先进行解密，需要用到银行公钥进行验签，我方私钥进行解密。

4. **文件解析**：根据银行接口文档，可能有XML，CSV，EXCEL等多种格式，相关的代码在网上都有。最主要的是解析出我方需要的数据，比如银行可能叫：charge_no，我们内部叫：pay_no，那就把对方的数据解析后并映射到内部的数据。
5. **发消息**：根据消息体配置，把一条数据按照消息体的配置，组装成消息，通过消息中间件发出去。

我们只需要把上面5个步骤所做的操作，全部抽象出单独的代码，然后通过配置文件来驱动即可。

7. 文件网关架构图



几点说明：

1. 核心上下文：把解决配置、临时变量等保存在上下文中，供后面流程引擎驱动文件的解析或打批。
2. 核心处理Handler：把核心处理能力抽象成通用的Handler，比如上传下载，报文解析，报文映射，消息发送等。
3. 流程引擎：把各handler组成一个责任链，依次执行。
4. 配置中心：提供产品化配置的能力。
5. 流程管控：解决发布和回滚问题。

8. 核心代码实现

核心的代码实现思路和报文网关基本是一致的，详见：《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》。

但是具体的代码实现和报文网关是不一样的，比如SFTP下载通常使用jsch来做，PGP加解密也有公开的组件，csv文件使用使用Apache Commons CSV库解析等。

另外的区别在于文件网关需要考虑任务调度，失败重试等，而报文网关是不需要考虑的，失败就由上游发起重试。

实现有困难的同学可以私聊。

9. 常见问题及应对方案

文件网关主要处理批量文件，有几个特殊的地方需要重点考虑，否则在后续的运维过程中可能是一头的包：

1. 任务分配不均

在分布式场景下，有多台服务器并发处理任务，如果调度系统设计不好，有可能部分服务器持续高负载处理，部分服务器持续在闲置，一些任务一直在排队，无法得到及时处理。

一般这种情况下，最简单的做法，就是随机调度一台服务器执行。另一种做法就是轮询法，先通过接口判断服务器当前状态是否空闲，如果正在执行任务，就找下一台，直到找到空闲的服务器。

2. 任务踩踏与并发执行

所谓任务踩踏，就是一个任务执行的时间非常久，在还没有结束时，另外一台服务器（也可能是同一台服务器）也被调度起来执行相同的任务，导致一个任务被两个服务器并行处理。

常用的解决方案，就是给任务加锁，但是加锁时需要设计**超时释放**。因为有可能任务跑到一半服务器重启，如果锁不释放，这个任务就无法重新被捞取执行。

3. 瞬时超大并发流量

文件网关解析文件是很快的，有可能几分钟解析出几百万条数据，如果不做任何保护措施，有可能瞬间对内部系统造成上万TPS的高并发压力，进而有可能影响在线实时业务的处理。

一般来说，文件网关不要通过RPC直接调用内部应用，而是通过消息的形式通知内部应用。主要考虑：文件网关短时间内解析出大批量的数据，如果直接调用内部应用，内部应用如果不限流，有可能把内部应用打到宕机，如果做了限流，文件网关还需要做调用失败重试。

如果通过消息中间件的形式通知内部应用，一方面可以由内部应用根据自己的能力来控制数据消费的速度，另一方面可以由消息中间件来缓存数据，无论是文件网关还是内部应用的实现都可以做得比较简单。

4. 通过RPC还是消息交互

如果是内部应用调用文件网关，建议使用RPC调用，这样比较简单。比如流出打款的指令。

如果是文件网关调用内部应用，建议使用消息中间件。解决文件网关瞬时超大并发流量对内部应用的冲击问题。

5. 任务恢复与重试

对于大文件的执行，有可能执行一半服务器就被重启，这种情况下就需要做任务恢复，对应的，上下游都需要做好幂等设计，因为部分数据可能会被重复解析并发送。

6. 异常单据处理

外部银行通道生成的文件，有可能因为系统BUG导致生成的部分数据的格式或内容有问题，无法正常解析。

常用的解决方案一般也是两种：1）记录有问题的数据，继续往下执行。2）任务终止，等外部银行修复文件后，重新执行。

一般是建议记录后，继续执行。主要是考虑外部银行修复会比较慢，有问题的数据放在差异里面去处理更好。

另外，对于部分数据缺失的场景，我们还可以主动补齐数据。比如正常情况下，外部银行通道需要给我方单号，银行单号，金额，币种等数据，假如一个包含几百万条数据的文件中有5条数据没有给我方单号，但是银行单号、金额、币种都能对上，那么我们就自行补全我方单号就行。没必要非得从几百万条数据中取出5条让银行补上。

6. 顺序执行与并发执行

对于银行的清算文件来说，系统正常是按顺序执行的，比如先解析20240101的数据，再解析20240102的数据，如果在解析20240101数据出现问题，通常也有两种方式：一、卡住，等20240101处理完成，才处理20240102。二、不管20240101失败，继续处理20240102的数据。

这里体现的是两种思维方式：顺序执行与并发执行。一般来说，推荐使用并发执行，一个任务失败，不要影响另一个任务。

10. 结束语

在本文中，我们探讨了文件网关在支付系统中的定位、不同形态的发展，到一种低代码设计思路，以及对应的系统架构。一个完整的文件网关代码量比较多，这里没有贴，后面在一些技术专题的拆解中，再贴上一些代码供参考，比如讲工厂模式，讲责任链模式。因为整体代码没有什么特别难的地方，无外就是把原理想清楚，加上一些常见的设计模式就能实现。

这是《百图解码支付系统设计与实现》专栏系列文章中的第（24）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。


隐墨星辰
扫一扫上面的二维码图案，加我为朋友。