48. 图解支付系统返回码映射踩过的坑及避坑指南

大家好，我是隐墨星辰，深耕境内/跨境支付架构设计十余年。

今天和大家简单聊下返回码映射存在哪些坑，以及如何避坑。

我在支付行业呆了十来年，和返回码映射导致的线上生产问题交手无数次，有很多是因为影响用户体验，也有一些直接导致了线上资损，所以有必要开一篇小文聊一下。

1. 返回码还是错误码

有些人喜欢用“返回码”，有些人喜欢用“错误码”。两者本质相同，都是用于标识通信或交易的状态。但我更倾向于使用“返回码”，因为它不仅涵盖错误状态，还包括成功状态，而成功并非错误，所以使用“返回码”更为合适。

2. 返回码的本质

我很喜欢探寻事物的本质，那么返回码的本质是什么？我觉得是解决2个问题：

1. **标识单一系统内的业务处理结果：**在一个单一系统内，如何表达业务处理的结果，比如参数不对，余额不足，还是成功等。
2. **完成异构系统或应用之间的处理结果同步：**在多个系统之间如何同步业务处理的结果，比如渠道的结果同步给支付平台的网关系统，网关系统同步给支付引擎等。

3. 返回码最核心的关注点

返回码最核心的关注点也只有2个：

1. **同一系统**内定义是否足够**清晰明确**。减少歧义，减少误解。
2. **异构**系统或应用之间的**映射**是否足够准确。映射不好，轻则影响用户体验，重则有资损。

4. 曾经碰到过的坑

踩过的坑很多，大致可以归为以下几类：

1. **对客映射不准确，导致用户持续重试失败，影响用户体验**。比如“余额不足”或“风控不过”，返回给用户“系统异常，请重试”，有些用户就疯狂地重试。
2. **外部渠道没有明确成功或失败，内部映射成明确成功或失败，造成资损**。比如：
   a. 支付同步请求渠道响应还没有回来，发起了查询，查询返回“订单不存在”，直接推进失败，但最后银行扣款成功。
   b. 退款同步请求渠道响应返回“系统异常”，直接推进到失败，但最后银行退款成功。
3. **外部渠道有双层返回码，没有做完整判断**。比如第1层只表示接口是否成功（通信层面），第2层才是表示业务是否成功，但是只判断了接口层面，就推进了内部订单的业务状态。
4. **返回码制定过于笼统或太细**。

5. 最佳实践

5.1. 基本原则

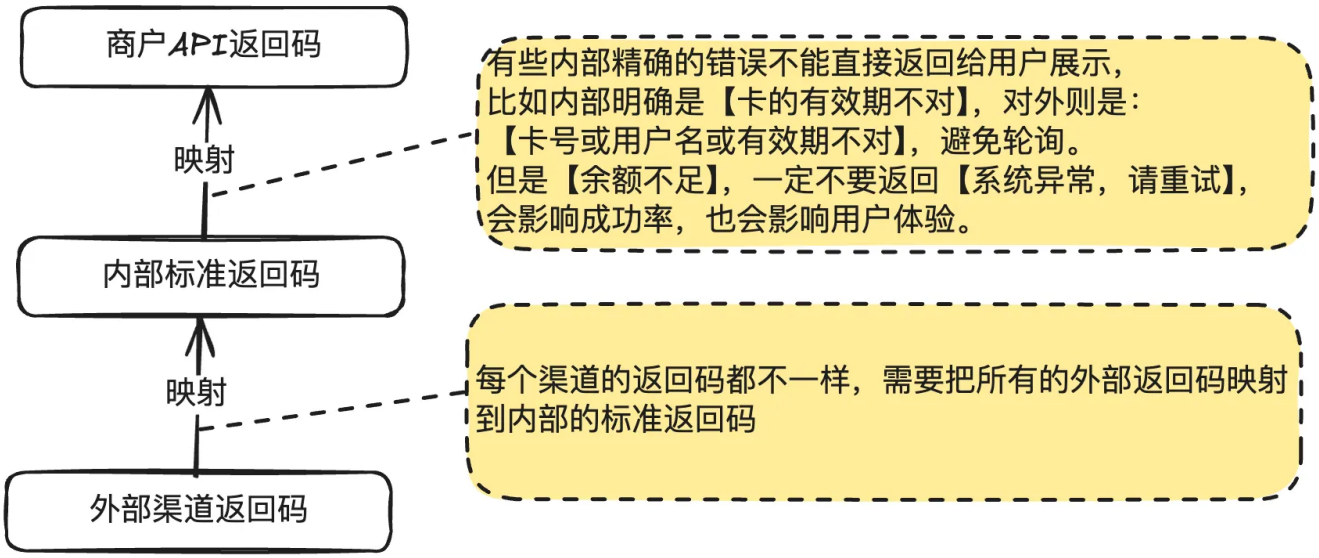
- **制定统一返回码规范**：在团队或公司层面制定统一的返回码规范，明确各个返回码的含义，确保各模块一致性。

- **严格遵守返回码定义**：研发人员在编码时，应严格按照规范返回对应的返回码，确保返回码与实际状态匹配。**明确成功才推进成功，明确失败才推进失败，其它全部按“未知”处理。**
- **区分接口/通信成功与业务成功。**
- **流入到平台的（支付、充值等），谨慎映射到成功。从平台流出的（提现，代发等），谨慎映射到失败。**

5.2. 三级返回码体系

外部商户对接支付平台，支付平台内部有自己的业务处理，同时还对接了外部的很多渠道，所以需要管理**三套返回码**：

- **提供给商户OpenAPI使用的返回码**：这块可以直接参考微信支付、支付宝等机构的门户网站。
- **内部各应用使用的标准返回码**：用于内部业务的处理。
- **渠道返回码**：外部渠道提供的返回码，每个渠道都不一样，需要映射到内部标准返回码。



为什么需要三层？主要有3个原因：

1. 内部应用使用的标准返回码需要精确，便于内部系统运行的监控。
2. 给商户OpenAPI的返回码需要业务语义明确，但不能过于精确。比如内部出现“卡的有效期不正确”，对外则是“卡号或持卡人或有效期不正确”，避免轮询攻击。
3. 外部渠道返回码不能全部一对一映射到内部，因为外部渠道太多，容易膨胀。

5.3. 商户OpenAPI返回码设计

这部分建议直接参考微信支付、支付宝或者ISO20022标准，这几家代表了行业的最高水准。

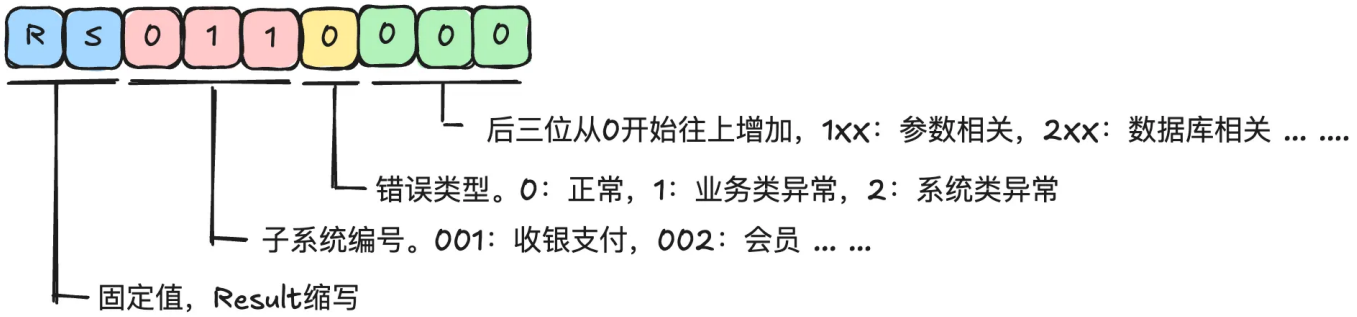
一般来说最少有两个字段：resultCode和message，一个表示码，一个表示码的描述。

也可以增加一个参数result，使用S，F，U表示业务状态的成功、失败、未知。

如果是查询类接口，一定要明确说明是接口成功，还是业务成功。

5.4. 内部标准返回码设计

支付平台内部也分了不同域，建议使用一个共同的规范，比如：RS+子系统编号+错误级别+具体返回码。具体如下图所示：



说明：

1-2位：固定值RS，Result缩写。

3-5位：子系统编号。比如001：收银支付，002：会员等。可方便定位哪个系统出的问题。

6位：错误类或等级。比如：0：正常，1：业务级异常，2：系统级异常。

7-9位：各业务线自己定。比如：1xx：参数相关，2xx：数据库相关，3xx：账户状态/Token状态相关等。

这样的好处在于，每个子域或子系统既有全局的规范，又有自己的灵活性，减少沟通成本。

注意：上面只是写了resultCode，还需要有message，用于描述这个码代表什么语义。

核心代码（注：使用chatGPT o1生成，请自行增删）：

Java

```java
public interface IResultCode {
    String getResultCode();
    String getMessage();
}

public enum PaymentResultCode implements IResultCode {
    SUCCESS( "0000", "success"),
    FAIL("2998", "fail"),
    SYSTEM_ERROR("2999",  "system error"),

    // 额度相关 11XX
    INSUFFICIENT_FUND("1101", "insufficient fund"),

    // 风控相关 12XX
    RISK_REJECTED("1201", "risk rejected"),

    // DB相关 21XX

    ;
    private static final String PREFIX = "RS";
    private static final String SYSTEM_CODE = "101";
    private String codeNumber;
    private String message;

    @Override
    public String getResultCode() {
        return PREFIX + SYSTEM_CODE + codeNumber;
    }

    @Override
    public String getMessage() {
        return message;
    }

    PaymentResultCode(String codeNumber, String message) {
        this.codeNumber = codeNumber;
        this.message = message;
    }
}


```

5.5. 渠道返回码映射

每个渠道的返回码都是不一样的，所以需要设计外部渠道返回码映射到内部标准返回码。需要遵守几个原则：

1. **只有明确成功，才能映射到成功**。
2. **只有明确失败，才能映射到失败**。比如渠道返回：订单不存在，或者系统异常，不能直接映射到失败，因为有可能会成功。
3. **涉及个人敏感信息或内部系统敏感信息的，需要转成模糊返回码和描述出去**，不能给最终用户展示精确信息。比如内部一个系统宕机，不能直接把异常抛出去。有效期错误也需要映射成“卡号或姓名或有效期不正确”。
4. **与敏感信息无关的，越准确越好，避免用户无谓的重试**。比如余额不足，就不要映射成“系统异常，请重试”。
5. **查询类接口，务必要区分是接口成功，还是业务成功。**

具体的技术实现，通过使用映射表就足够，加到缓存中，增加运算速度。如果找不到映射关系，就全部转到一个默认的返回码上面，同时对这个默认返回码做监控，定期把这些没有做映射的返回码映射到正确的返回码上面去。避免应该把类似“余额不足”映射成了“系统异常请重试”的场景。

5.6. 返回码监控与告警

大部分团队都会监控成功率，只有少数团队会监控返回码或定期分析返回码。然而当交易量足够大时，成功率的波动可能只有0.5%，很难看出异常，而如果去分析返回码，则可以快速看出并定位问题。

一般来说，有几个建议：

1. **实时监控返回码的突变异常**。比如：最近10分钟，某个返回码突然增加50%，或者比明天突然增加50%等。都需要介入看看。
2. **定期观察返回码曲线表**。如果某个返回码连续多天持续在上升，一般都是有问题的。
3. **建立返回码全链路映射大盘**。比如渠道返回“余额不足”或“风控不通过”，映射到用户展示“系统异常，请重试”，那就有问题。而且类似这种情况还非常常见。
4. **定期分析用户支付行为**。以前在分析用户行为时，发现同一用户重试了20多次，最后排查发

现，就是返回码映射不准确，导致用户无谓的重试。

6. 结束语

卷用户体验和成功率时，往往需要于细微处见真章，而返回码的设计和映射就是如此。做得不好，轻则影响用户体验，重则资损。

希望对大家在设计标准返回码及映射时有所启发，也欢迎点赞转发。

这是《图解支付系统设计与实现》专栏系列文章中的第（48）篇。

深耕境内/跨境支付架构设计十余年，欢迎关注并星标公众号“隐墨星辰”，和我一起深入解码支付系统的方方面面。

专栏系列文章PDF合集不定时更新，欢迎关注我的公众号“隐墨星辰”，留言“PDF”获取。

有个支付系统设计与实现讨论群，添加个人微信（yinmon_sc）备注666进入。

隐墨星辰 个人微信

10年顶尖境内/跨境支付公司架构经验

著有《图解支付系统设计与实现》

备注666进支付讨论群