**BPS分布式回退设计**

# 业务场景

在很多时候BPS是做为分布式的结构来部署,在银联机构入网项目组中,BPS是一个单独部署的web应用,业务系统通过webservice接口驱动BPS流程流转.在此种部署结构下如果业务系统出现异常事务正常回滚,但是BPS却不能相应的回滚从而造成不一致问题.

# 设计思路

BPS提供了基于两个节点之间的回退，在业务系统调用结束工作项或者结束人工活动出现异常时业务系统无法知道应该从哪个节点退回到哪个节点。在调用BPS 结束工作项或者结束人工活动webservice接口时会返回被结束的人工活动实例ID，再调用BPS回退接口并传入被结束的人工活动实例ID，在回退接口中根据活动实例ID得到此活动实例的所有后继活动实例，并将所有的后继活动退回到原来的状态。有了这三个值就可以调用BPS的回调接口实现BPS回滚。如果后继人工活动有多个则调用“自定义增强的回退接口”将所有的后继活动全部退回到原来的状态。此处需要使用到的BPS接口如下：

1：[WFActivityInstManager](http://127.0.0.1:49394/help/topic/com.primeton.studio.workflow.library/javadoc/client/bps-api/com/eos/workflow/api/impl/WFActivityInstManager.html" \t "classFrame" \o "com.eos.workflow.api.impl �е���)类中

public java.util.List<[WFActivityInst](http://127.0.0.1:49394/help/topic/com.primeton.studio.workflow.library/javadoc/client/bps-api/com/eos/workflow/data/WFActivityInst.html" \o "com.eos.workflow.data 中的类)> queryNextActInsts(long actInstID)

根据活动实例ID，查询后继活动实例.

2：WFBackActivityManager类中

void backActivity(long currentActInstID,

long destActInstID,

java.lang.String rollBackStrategy)

根据某策略回退到某活动

# 实现细节

## BPS端补偿接口实现

现在项目中需要补偿的BPS接口大致可以分为以下两类：

1. 启动流程
2. 人工活动流转包括结束工作项、结束人工活动和完成自动活动等

针对这两种接口依次实现回退的webservice接口

1. 启动流程补偿接口：根据启动后的流程实例ID终止流程
2. 人工活动流转补偿接口：根据当前活动实例ID和原来的活动实例ID基于path退回活动，特别注意：如果是自动活动需要设置自动活动的结束方式为手动结束，否则退回到自动活动时自动活动会自动结束。

## 业务系统的回退实现

### 面临的问题

1. 同一事务内BPS接口全部回滚：在一个事务中可能调用多次业务接口和多个BPS we接口，业务系统可以基于事务机制和事务传播机制一次性“全部”回退，BPS回退也需要一次性将同一个事务中调用的BPS接口全部回退。
2. BPS接口回滚时机：需要在调用BPS接口失败或者业务系统抛出异常时统一回滚BPS接口，否则回滚可能导致回滚的代码导出分布不方便维护。

### 使用ThreadLocal解决全部回滚问题

使用ThreadLocal保存当前线程中BPS请求队列对应的回退队列。如果某个接口需要补偿接口则在调用此接口之后将此接口对应的回退接口以及回退接口所需参数保存到ThreadLocal的回退堆栈(可以保证后进先出)中。

### 使用异常拦截器异常时触发回退调用

定义异常拦截器在业务系统请求线程出现异常捕捉异常，并根据异常的类型判断是否调用BPS回退接口。如果需要回退则从ThreadLocal中得到回退堆栈内容，并将回退堆栈中的所有接口全部回退

# 图解实例说明

下图为用户某个请求的图解说明。

图1 用户请求回退说明