## 应用环境

工作流版本： bps 6.2

数据库版本：Oracle 10.2.0.5

Bps 运行在weblogic 10.3上

业务应用与bps应用服务器为分离部署，通过http远程调用，业务应用也运行在weblogic10.3中。

## 问题现象描述

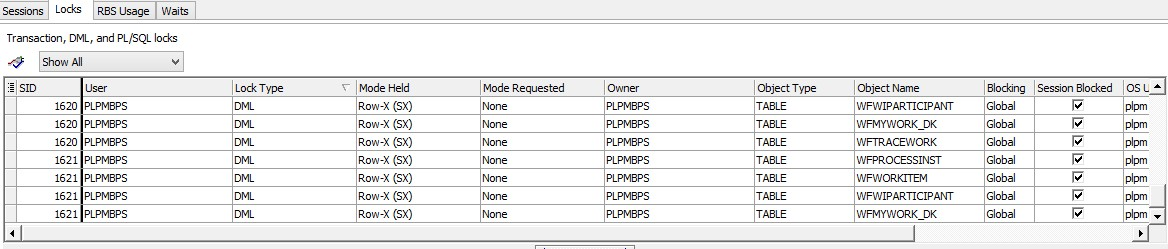
客户在进行压力测试时，由于测试人员在清除旧历史数据时，未将流程中的wfwiparcitipant（参与人）表进行清除。造成bps在启动流程时，报出大量的主键重复异常，同时bps 数据库出现大量的行锁定，导致在压力测试一段时间后，bps服务器不能响应新的请求，只能通过重启bps 应用服务器来释放数据库的连接和锁。

在压力测试过程中，如果bps数据库中的数据均已清理，则bps能够正常运行，并且不会出现数据库行锁的现象。我们通过在不同的测试场景下进行测试，发现问题重现时是发生在bps应用服务器进行数据更新、新增、操作时，如果发生较多的异常的情况下（例如创建流程时让参与人表中有重复主键的数据，从而导致抛出主键重复异常），就一定会产生行锁的情况。

通过打印的日志，将回滚的流程实例ID取出后，在数据库进行查询，发现失败并抛出异常的流程实例信息、工作项信息都没有被回滚，而是已经保存在数据库中。

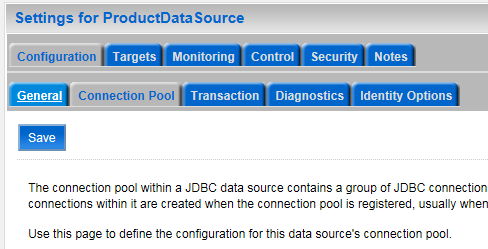
## 问题排查过程

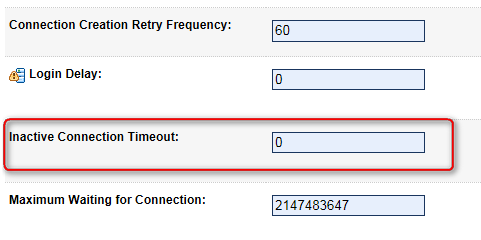
开始未怀疑是由于事务管理器产生的问题，因为通过编程式的事务控制，我们已经确定事务都能够得到提交或回滚。但是通过监控数据库的会话，发现锁定的表和锁定的类型均为事务锁（如下图），因此怀疑是由于事务没有提交导致的。

**图表、监控数据库Locks发现大量Session Blocked**

由于代码中已经对事务进行了显式地提交和回滚，并且代码里面没有获取Connection等进行数据插入、更新的操作，因此我们怀疑是WebLogic中连接不能正常回收导致的连接泄漏。通过在网上查找大量的资料，并不断进行尝试，发现尝试修改WebLogic连接池的Inactive Connection Timeout配置（默认为0，即不启用非活动连接超时回收，如下图表）为50-60（单位为秒）后，连接会被自动释放掉，并且锁也会被释放，但是此时调用bps服务的任何接口，都无法调用成功，并且会提示以下几种异常信息：

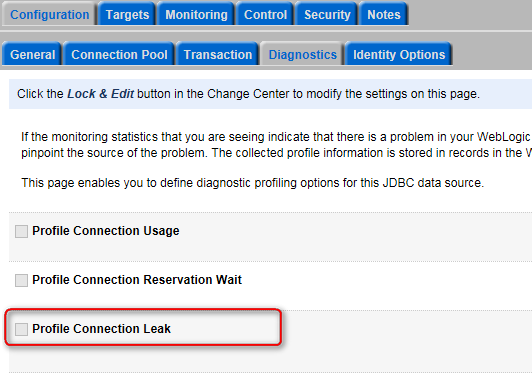
|  |
| --- |
| ***1、java.sql.SQLException: The transaction is no longer active - status: 'Rolling Back. [Reason=weblogic.transaction.internal.TimedOutException: Transaction timed out after 29 seconds***  ***BEA1-1ED161006F7C90328790]'. No further JDBC access is allowed within this transaction.***  ***2、java.sql.SQLRecoverableException: Io exception: Socket closed***  ***3、java.sql.SQLException: Closed Connection***  ***4、java.sql.SQLException: Closed Statement*** |

****

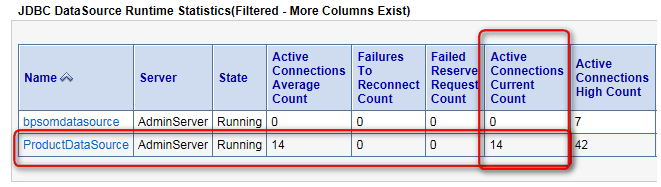
****

**图表、WebLogic Connection pool Inactive Connection Timeout选项**

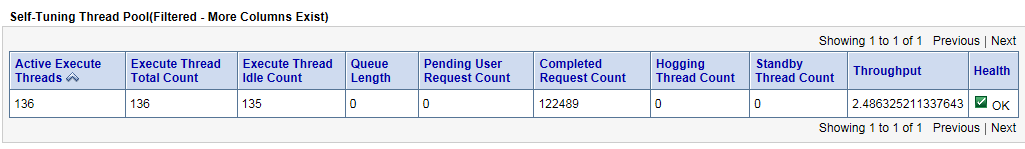
在通过启用weblogic 的 jdbc connection pool 的 Profile Connection Leak 选项（如下图表），重新进行压力测试，重现数据库行锁后；我们停止压力测试并完全退出测试一段时间后，在weblogic 控制台的监控页面观察到活动的jdbc连接仍然有14个，而线程监控中亦无正在执行或挂起的线程，如下图所示：

****

**图表、启用连接泄漏统计**

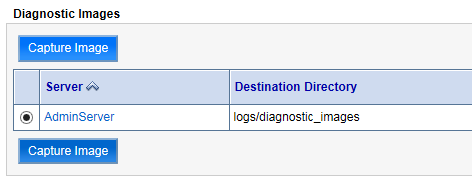
****

**图表、jdbc data source 统计信息**

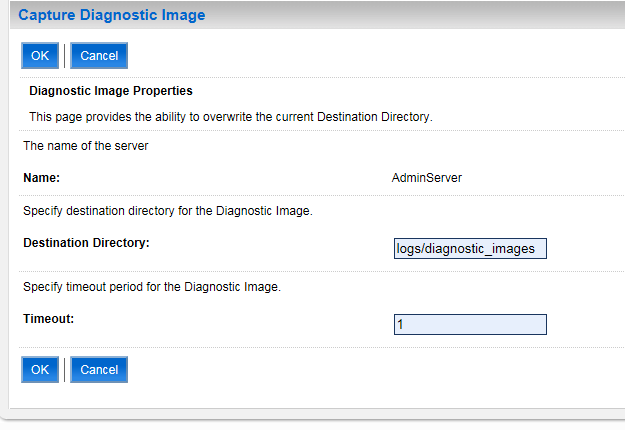
****

**图表、线程统计信息**

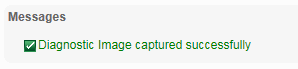
此时在weblogic 的Diagnostic Images选项中获取堆栈信息，如下图所示：

****

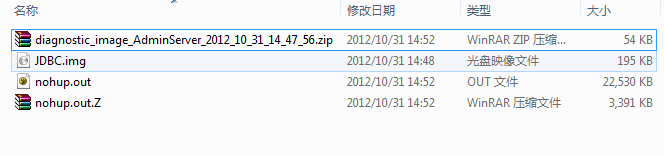
**图表、获取当前应用服务器的快照信息**

****

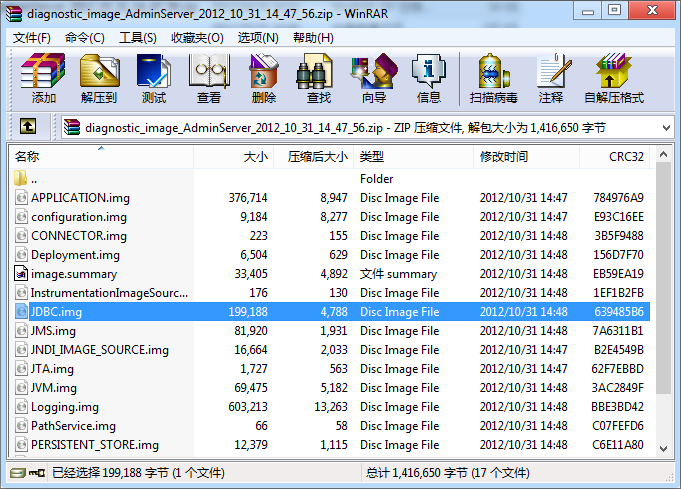
**图表、默认统计信息保存的路径**

****

**图表、统计成功的信息**

****

**图表、diagnostic\_img\_AdminServer\*.zip文件**

****

**图表、JDBC.img文件**

通过检查生成的diagnostic\_image下的JDBC.img（完整请信息请参见随附件发送的diagnostic\_image\_AdminServer\_2012\_10\_31\_14\_47\_56.zip中的JDBC.img），发现有jdbc 泄漏的现象（两段红色加粗代码）：

|  |
| --- |
| **Resource Pool:ProductDataSource:dumpPool reserved[0] = autoCommit=false,enabled=true,isXA=false,isJTS=false,vendorID=100,connUsed=true,doInit=false,'null',destroyed=false,poolname=ProductDataSource,appname=null,moduleName=null,connectTime=59,dirtyIsolationLevel=false,initialIsolationLevel=2,infected=false,lastSuccessfulConnectionUse=0,secondsToTrustAnIdlePoolConnection=10,currentUser=java.lang.Exception**     at weblogic.jdbc.common.internal.ConnectionEnv.setup(ConnectionEnv.java:308)     at weblogic.common.resourcepool.ResourcePoolImpl.reserveResource(ResourcePoolImpl.java:314)     at weblogic.common.resourcepool.ResourcePoolImpl.reserveResource(ResourcePoolImpl.java:292)     at weblogic.jdbc.common.internal.ConnectionPool.reserve(ConnectionPool.java:425)     at weblogic.jdbc.common.internal.ConnectionPool.reserve(ConnectionPool.java:316)     at weblogic.jdbc.common.internal.ConnectionPoolManager.reserve(ConnectionPoolManager.java:93)     at weblogic.jdbc.common.internal.ConnectionPoolManager.reserve(ConnectionPoolManager.java:106)     at weblogic.jdbc.pool.Driver.connect(Driver.java:149)     at weblogic.jdbc.jts.Driver.getNonTxConnection(Driver.java:642)     at weblogic.jdbc.jts.Driver.connect(Driver.java:124)     at weblogic.jdbc.common.internal.RmiDataSource.getConnection(RmiDataSource.java:338)     at com.primeton.ext.infra.connection.DataSourceWrapper.getAvailableConnectionIfNecessary(DataSourceWrapper.java:169)     at com.primeton.ext.infra.connection.DataSourceWrapper.getConnection(DataSourceWrapper.java:91)     at org.springframework.jdbc.datasource.DataSourceUtils.doGetConnection(DataSourceUtils.java:112)     at org.springframework.jdbc.datasource.DataSourceUtils.getConnection(DataSourceUtils.java:77)     at com.eos.infra.connection.ConnectionFactory.createConnection(ConnectionFactory.java:69)     at com.eos.common.connection.ConnectionHelper.doInGetConnection(ConnectionHelper.java:188)     at com.eos.common.connection.ConnectionHelper.getConnection(ConnectionHelper.java:236)     at com.primeton.workflow.persistence.connection.EOSConnectionProvider.getConnectionInner(EOSConnectionProvider.java:151)     at com.primeton.workflow.persistence.connection.EOSConnectionProvider.getConnection(EOSConnectionProvider.java:139)     at sun.reflect.GeneratedMethodAccessor106.invoke(Unknown Source)     at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)     at java.lang.reflect.Method.invoke(Method.java:597)     at com.primeton.workflow.service.persistence.connection.ConnProviderProxy$DataSourceHandler.invoke(ConnProviderProxy.java:111)     at $Proxy62.getConnection(Unknown Source)     at com.primeton.workflow.service.persistence.sql.BatcherImpl.openConnection(BatcherImpl.java:210)     at com.primeton.workflow.service.persistence.sql.JDBCContext.connect(JDBCContext.java:120)     at com.primeton.workflow.service.persistence.sql.JDBCContext.connection(JDBCContext.java:108)     at com.primeton.workflow.service.persistence.sql.BatcherImpl.prepareQueryStatement(BatcherImpl.java:89)     at com.primeton.workflow.service.persistence.sql.QueryStatementExecutor.prepareQueryStatement(QueryStatementExecutor.java:152)     at com.primeton.workflow.service.persistence.sql.QueryStatementExecutor.executeQuery(QueryStatementExecutor.java:115)     at com.primeton.workflow.service.persistence.PersistenceManager.executeQuery(PersistenceManager.java:968)     at com.primeton.workflow.service.persistence.PersistenceManager.retrieveObject(PersistenceManager.java:201)     at com.primeton.workflow.service.persistence.PersistenceManager.retrieveObject(PersistenceManager.java:125)     at com.primeton.workflow.service.persistence.DatabaseServiceImpl.findPersistantObject(DatabaseServiceImpl.java:310)     at com.primeton.workflow.service.das.database.impl.DataAccessServiceImpl.commonFindPersistentObject(DataAccessServiceImpl.java:81)     at com.primeton.workflow.service.das.database.impl.DataAccessServiceImpl.findProcessInstanceByID(DataAccessServiceImpl.java:222)     at com.primeton.workflow.instpool.smp.DASUtil.findProcessInstanceByID(DASUtil.java:219)     at com.primeton.workflow.instpool.smp.WFInstancePoolImplBasic.findProcessInstanceByID(WFInstancePoolImplBasic.java:321)     at com.primeton.workflow.instpool.smp.WFInstancePoolImplBasic.findWorkItemInstanceByID(WFInstancePoolImplBasic.java:358)     at com.primeton.workflow.instpool.smp.InstPoolSynchronization.getProcInstID(InstPoolSynchronization.java:285)     at com.primeton.workflow.instpool.smp.InstPoolSynchronization.getAllProcessInstIDFromLog(InstPoolSynchronization.java:261)     at com.primeton.workflow.instpool.smp.InstPoolSynchronization.executeRollBack(InstPoolSynchronization.java:246)     at com.primeton.workflow.instpool.smp.InstPoolSynchronization.afterCompletion(InstPoolSynchronization.java:236)     at com.primeton.workflow.instpool.tx.TransactionManagerAdaptor$MySynchronization.afterCompletion(TransactionManagerAdaptor.java:153)     at com.primeton.workflow.commons.txeos.TransactionServiceImpl4EOS$Sync.afterCompletion(TransactionServiceImpl4EOS.java:294)     at org.springframework.transaction.support.AbstractPlatformTransactionManager.invokeAfterCompletion(AbstractPlatformTransactionManager.java:769)     at org.springframework.transaction.support.AbstractPlatformTransactionManager.triggerAfterCompletion(AbstractPlatformTransactionManager.java:741)     at org.springframework.transaction.support.AbstractPlatformTransactionManager.processRollback(AbstractPlatformTransactionManager.java:652)     at org.springframework.transaction.support.AbstractPlatformTransactionManager.rollback(AbstractPlatformTransactionManager.java:600)     at com.primeton.common.transaction.impl.DefaultTransactionManager.rollback(DefaultTransactionManager.java:171)     **at com.eos.workflow.impl.ext.BPSServiceManagerExt.createAndStartProcessInstAndFinishFirstWorkItemWithWfMywork(BPSServiceManagerExt.java:881)**     at sun.reflect.GeneratedMethodAccessor303.invoke(Unknown Source)     at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)     at java.lang.reflect.Method.invoke(Method.java:597)     at org.apache.tuscany.sca.implementation.java.invocation.JavaImplementationInvoker.invoke(JavaImplementationInvoker.java:132)     at org.apache.tuscany.sca.binding.sca.impl.SCABindingInvoker.invoke(SCABindingInvoker.java:61)     at com.primeton.sca.host.impl.AbstractInterceptor.invoke(AbstractInterceptor.java:90)     at com.primeton.sca.host.impl.AbstractInterceptor.invoke(AbstractInterceptor.java:90)     at org.apache.tuscany.sca.core.invocation.JDKInvocationHandler.invoke(JDKInvocationHandler.java:381)     at org.apache.tuscany.sca.core.invocation.JDKInvocationHandler.invoke(JDKInvocationHandler.java:225)     at $Proxy75.createAndStartProcessInstAndFinishFirstWorkItemWithWfMywork(Unknown Source)     at sun.reflect.GeneratedMethodAccessor302.invoke(Unknown Source)     at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)     at java.lang.reflect.Method.invoke(Method.java:597)     at com.primeton.sca.host.impl.ServiceInvokerLoggerJdkInvocationHandler$1.invokeService(ServiceInvokerLoggerJdkInvocationHandler.java:40)     at com.primeton.sca.host.impl.ServiceInvokerLoggerJdkInvocationHandler.invokeService(ServiceInvokerLoggerJdkInvocationHandler.java:84)     at com.primeton.sca.host.impl.ServiceInvokerLoggerJdkInvocationHandler.invoke(ServiceInvokerLoggerJdkInvocationHandler.java:37)     at $Proxy75.createAndStartProcessInstAndFinishFirstWorkItemWithWfMywork(Unknown Source)     at sun.reflect.GeneratedMethodAccessor301.invoke(Unknown Source)     at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)     at java.lang.reflect.Method.invoke(Method.java:597)     at com.primeton.access.client.impl.skeleton.ScaServiceReceiveTarget.invoke(ScaServiceReceiveTarget.java:79)     at com.primeton.access.client.impl.skeleton.ServiceEngine.receive(ServiceEngine.java:36)     at com.primeton.access.client.impl.skeleton.DefaultServiceSkeleton.invokeService(DefaultServiceSkeleton.java:130)     at com.primeton.access.client.impl.skeleton.DefaultServiceSkeleton.invoke(DefaultServiceSkeleton.java:32)     at sun.reflect.GeneratedMethodAccessor114.invoke(Unknown Source)     at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)     at java.lang.reflect.Method.invoke(Method.java:597)     at com.primeton.ext.system.method.LocalMethodCaller.call(LocalMethodCaller.java:242)     at com.primeton.access.client.impl.processor.CommonServiceProcessor.process(CommonServiceProcessor.java:114)     at com.primeton.access.http.impl.ProcessorWebInterceptor.doIntercept(ProcessorWebInterceptor.java:139)     at com.primeton.access.http.impl.WebInterceptorChainImpl.doIntercept(WebInterceptorChainImpl.java:78)     at com.primeton.access.authorization.impl.AccessedHttpResourceFilter.doIntercept(AccessedHttpResourceFilter.java:170)     at com.primeton.access.http.impl.WebInterceptorChainImpl.doIntercept(WebInterceptorChainImpl.java:78)     at com.eos.access.http.UserLoginCheckedFilter.doIntercept(UserLoginCheckedFilter.java:189)     at com.primeton.access.http.impl.WebInterceptorChainImpl.doIntercept(WebInterceptorChainImpl.java:78)     at com.primeton.access.http.impl.WebI18NInterceptor.doIntercept(WebI18NInterceptor.java:91)     at com.primeton.access.http.impl.WebInterceptorChainImpl.doIntercept(WebInterceptorChainImpl.java:78)     at com.primeton.sca.host.webapp.SCAWebServiceServletFilter.doIntercept(SCAWebServiceServletFilter.java:118)     at com.primeton.access.http.impl.WebInterceptorChainImpl.doIntercept(WebInterceptorChainImpl.java:78)     at com.eos.access.http.InterceptorFilter.doFilter(InterceptorFilter.java:176)     at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:42)     at weblogic.servlet.internal.WebAppServletContext$ServletInvocationAction.run(WebAppServletContext.java:3496)     at weblogic.security.acl.internal.AuthenticatedSubject.doAs(AuthenticatedSubject.java:321)     at weblogic.security.service.SecurityManager.runAs(Unknown Source)     at weblogic.servlet.internal.WebAppServletContext.securedExecute(WebAppServletContext.java:2180)     at weblogic.servlet.internal.WebAppServletContext.execute(WebAppServletContext.java:2086)     at weblogic.servlet.internal.ServletRequestImpl.run(ServletRequestImpl.java:1406)     at weblogic.work.ExecuteThread.execute(ExecuteThread.java:201)     at weblogic.work.ExecuteThread.run(ExecuteThread.java:173) ,currentThread=Thread[[ACTIVE] ExecuteThread: '39' for queue: 'weblogic.kernel.Default (self-tuning)',5,Pooled Threads],lastUser=null,currentError=null,currentErrorTimestamp=null,JDBC4Runtime=true,supportStatementPoolable=true,needRestoreClientInfo=false,defaultClientInfo={} |

其中，com.eos.workflow.impl.ext.BPSServiceManagerExt.createAndStartProcessInstAndFinishFirstWorkItemWithWfMywork(BPSServiceManagerExt.java:881)该代码是我们通过编号为：BPS\_6.0.0.12\_20091117\_C1的补丁在bps上作的扩展。该方法的完整实现如下：

|  |
| --- |
| public long createAndStartProcessInstAndFinishFirstWorkItemWithWfMywork(Map comprDataMap, Map wfRelaDataMap)     throws Exception   {     log.debug("into createAndStartProcessInstAndFinishFirstWorkItemWithWfMywork:comprDataMap+wfRelaDataMap" +        comprDataMap + ":" + wfRelaDataMap);     ITransactionManager manager = null;     long processInstID = -1L;     try {       long t = System.currentTimeMillis();       manager = TransactionManagerFactory.getTransactionManager();        manager.begin(new int[] { 3 });        processInstID =          getIProcessInstManager().createProcessInstance(         (String)comprDataMap.get("processdefname"),          comprDataMap.get("workstationtype").toString(),          (String)comprDataMap.get("processinstdesc"));        if ((wfRelaDataMap != null) && (!wfRelaDataMap.isEmpty())) {         getIRelativeDataManager().setRelativeDataBatch(processInstID, wfRelaDataMap);       }        getIProcessInstManager().startProcessInstAndFinishFirstWorkItem(         processInstID,          false,          null);        comprDataMap.put("processinstid", Long.valueOf(processInstID));        WorkFlowExtUtil.dataTransManagerForNormalWF(comprDataMap, wfRelaDataMap, 0, null);        BPSServiceManagerUtil.getOMParticipantsFromWF(processInstID, comprDataMap);        DataObject wfmywork = WorkFlowExtUtil.transMap2WFDataObject(comprDataMap, 0);       DataObject wftracework = WorkFlowExtUtil.transMap2WFDataObject(comprDataMap, 1);       if ((wfmywork == null) || (wftracework == null)) {         throw new RollbackableBizException("after data trans:wfmywork or wftracework null:please check!");       }       DatabaseExt.getPrimaryKey(wfmywork);        DatabaseUtil.insertEntity("default", wfmywork);        DatabaseExt.getPrimaryKey(wftracework);        DatabaseUtil.insertEntity("default", wftracework);        log.info("createAndStartProcessInstAndFinishFirstWorkItemWithWfMywork:cost time:" + (System.currentTimeMillis() - t) + "ms");       manager.commit();     } catch (Exception e) {       log.error("into createAndStartProcessInstAndFinishFirstWorkItemWithWfMywork test:" + processInstID);       **manager.rollback();//实际代码的第881行**       EOSExceptionHandle.invoke2InfoExceptionWithParticipantRuleInfo(e,          "createAndStartProcessInstAndFinishFirstWorkItemWithWfMywork error..comprDataMap+wfRelaDataMap." + comprDataMap + ":" + wfRelaDataMap);     }      return processInstID;   } |

通过检查JDBC.img中的信息，怀疑在使用事务管理器ITransactionManager对象进行事务回滚时，事务并不能被回滚，同时事务也没有被提交所致。而如果该操作中的事务没有被回滚，也没有被提交的情况下，数据库连接保持活动状态，从而导致bps的相关表被锁。

## 解决办法

向售后部门索取编号为：BPS\_6.2.0.10\_20101231\_P1的补丁，补丁描述： BPS底层使用EOS的事务管理器和数据源，EOS是使用Spring的实现，而在WebLogic下使用Spring的事务管理器+Spring的事务同步回调，在同步回调中再次获取连接进行数据库查询（这个是模拟工作流引擎中的一个实现场景）,而后这个在同步回调中使用的连接就会被Spring绑定到线程上，以至于下次该线程处理新的请求时，会重用该连接，造成不受外部启动的事务管理器控制。解决方式：每个请求锁定流程实例时，将流程实例ID绑定到线程变量中。事务后提交、回滚缓存均使用线程变量中记录的流程实例ID进行回滚。此方案中引擎实现中避免了取连接，能够解决同步回调中重新取连接的问题。

## 小结

该问题产生，必须要有一定的用户接入测试（可通过压力负载工具来生成），并且必须是工作流端需要产生异常的情况下才会重现；如果BPS运行正常，或只是少量抛出异常，都不容易重现该问题。我们发现之初，是在用户测试环境中发现该问题，并且时而重现时而又运行正常，由于项目的特殊性，用户测试环境中发现问题后，往往没有多少时间给我们进行详细的排查，因此也造成了该问题解决时间较长。