UCOB

United Corporation On Blockchain

目录

一、简介

区块链上的联合公司。本项目设计了一种运行在区块链上的联合公司，结合区块链特性实现了联合公司治理的若干方法，包括项目管理、资金管理、投票治理等。

本智能合约项目使用 solidity 开发，主要有三大模块：联合治理模块、公司模块、项目模块。

按照模块划分，主要实现功能有，

1、联合治理模块：提案与投票系统，联合公司成员管理系统，联合公司间资金流转系统；

2、公司模块：单个公司管理系统，单个公司内部资金流转系统；

3、项目模块：多个公司的联合项目管理，单个公司的内部项目管理。

本项目合约代码可部署在 EVM 虚拟机上运行，已通过 Remix、FISCO BCOS 测试，本文档主要依据 FISCO BCOS 展开。

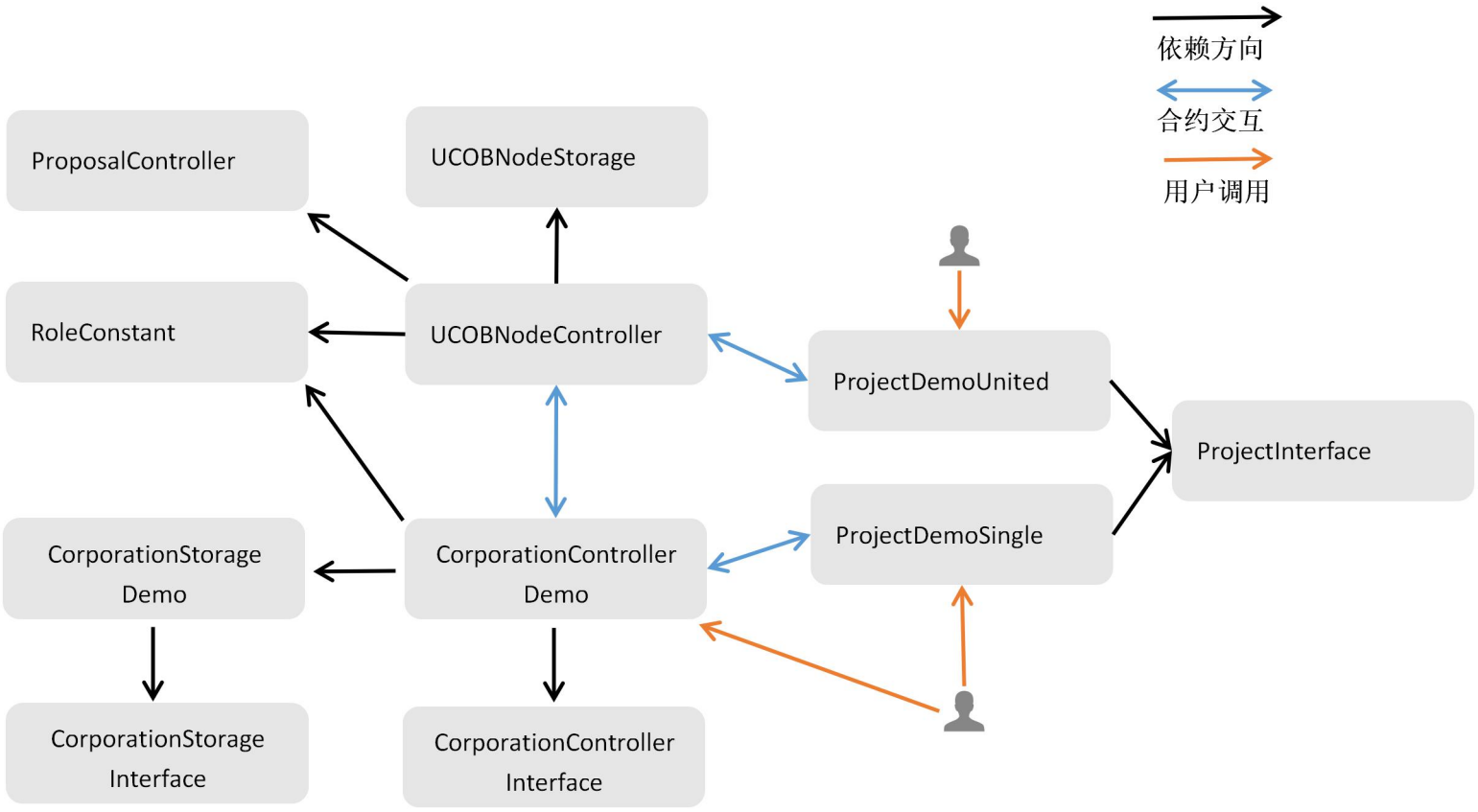
二、合约代码说明

lib……OpenZeppelin 开源代码库
CorporationControllerDemo.sol……单个公司控制器智能合约的 demo
CorporationControllerInterface.sol……公司控制器智能合约实现接口
CorporationStorageDemo.sol……单个公司存储智能合约的 demo
CorporationStorageInterface.sol……公司存储智能合约实现接口
ProjectDemoSingle.sol……单个公司内部项目的 demo
ProjectDemoUnited.sol……多个公司参与的联合项目 demo
ProjectInterface.sol……项目智能合约需实现的接口
ProposalController.sol……提案与投票管理控制器
RoleConstant.sol……人员角色常量
UCOBNodeController.sol……UCOB 联合公司控制器智能合约
UCOBNodeStorage.sol……UCOB 联合公司存储智能合约

事实上，*CorporationControllerDemo.sol* 只是单个公司的一种智能合约实现而已，任一公司在区块链上的智能合约，只要实现了 *CorporationControllerInterface.sol* 接口便可以自定义内部的逻辑，这带来了公司内部管理的隐私性与灵活性。

同样，不管是联合项目智能合约 *ProjectDemoUnited.sol*，还是单个公司内部项目智能合约 *ProjectDemoSingle.sol*，只要实现了 *ProjectInterface.sol* 接口，就可以自定义项目业务逻辑了。请注意，“联合项目智能合约”需要通过 *UCOBNodeController.sol* 的投票与注册。

三、合约架构



需要再次强调的是，上图中“*xxxxxxDemo*”合约都是可以灵活实现的，比如上图中智能合约“*CorporationControllerDemo*”依赖智能合约“*RoleConstant*”，但也许在另一家公司的智能合约实现方案中，不必依赖 *RoleConstant.sol*。

四、合约部署说明

可以通过 FISCO 控制台或者网页端 WeBank Web 部署合约，合约部署顺序如下：

<1>部署 UCOBNodeStorage.sol，记录合约地址；

<2>使用<1>中地址作为参数，部署 UCOBNodeController.sol，记录合约地址；

<3>部署 CorporationStorageDemo.sol，记录合约地址；

<4>使用<2>和<3>地址作为参数，部署 CorporationControllerDemo.sol，记录合约地址；

<5>使用<2>地址作为参数，部署 ProjectDemoUnited.sol

<6>使用<4>地址作为参数，部署 ProjectDemoSingle.sol

此章节介绍了合约部署的顺序，部署完成后还需要进行初始化的调用操作才可以转移“storage”类合约的所有权。

本项目中，当业务进行时，“*xxxxxxStorage*”类合约的调用者只能是“*xxxxxxController*”类合约；因此，需要部署者在部署完成后，转移“*Storage*”合约所有权到对应的“*Controller*”合约上。

五、业务流程与合约操作说明

说明：

合约调用操作可参考 UCOBJavaSDK 项目代码。注意，本项目 Java 类中方法命名规则为“合约名_合约方法名”，即“*contractName_methodName*”。

5.1 转移存储合约的所有权，初始化默认管理员角色，通过 JavaSDK 操作，使用合约部署者的账户操作:

5.1.1 修改“UCOBNodeStorage”的 owner 为“UCOBNodeController”

执行方法：
com.ucob.controller.UCOBController.UCOBNodeStorage_transferOwnership()

#检查权限是否成功转移，返回“true”表示成功转移

执行方法：
com.ucob.controller.UCOBController.UCOBNodeController_checkUCOBNodeStorageSafety()

5.1.2 修改“CorporationStorageDemo”的 owner 为“*CorporationControllerDemo*”

执行方法：
com.ucob.controller.CorporationController.CorporationStorageDemo_transferOwnership()

#检查权限是否成功转移，返回“true”表示成功转移

执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_checkCorporationStorageSafety()

如果有多家公司，每一家公司都要转移存储合约的所有权到对应公司的控制器合约上。

5.1.3 注册“UCOBNodeController”管理员角色中的第一个成员

本例管理员角色为“CORPORATION_MEMBERS_ROLE”，

#在 UCOBNodeController.sol 合约上，以“第一家公司”的控制器

合约地址作为传参调用“initRoleAdmin()”方法，注册为 UCOB 中的第一个管理员角色

执行方法：
com.ucob.controller.UCOBController.UCOBNodeController_initRoleAdmin()

#检查角色注册是否成功，UCOBNodeController 合约地址和 CorporationController 合约地址都要成功注册

执行方法：
com.ucob.controller.UCOBController.UCOBNodeController_checkRole()

5.1.4 注册 CorporationControllerDemo 中的第一个管理员，默认为“CORPORATION_CEO_ROLE”角色

在 CorporationControllerDemo.sol 合约上，执行“setRoleAdmin()”方法，第一个调用该方法的私钥被认为具备 CEO 角色，

执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_setRoleAdmin()

#检查角色注册是否成功，

执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_checkRole()

5.2 添加多个公司

5.2.1 部署公司存储合约和公司控制器合约，部署者转移所有权，第一个调用“setRoleAdmin()”方法的将设置自己为 CEO 角色，

执行方法：
com.ucob.controller.CorporationController.CorporationStorageDemo_transferOwnership()
执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_setRoleAdmin()

5.2.2 发起提案，对是否允许新公司加入 UCOB 进行投票

由已经在 UCOB 中的公司，使用该公司中具备 CEO 角色的账户，通过代理调用的方式调用 UCOBNodeController.sol 合约发起提案，

执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_startProposal()

#检查提案是否成功提交

执行方法：
com.ucob.controller.UCOBController.UCOBNodeController_getProposal()

5.2.3 发起提案过后，再通过代理方法投票该提案；

执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_voteProposal()

5.2.4 赋予该公司对应的角色，该公司合约成功加入 UCOB

执行方法：
com.ucob.controller.UCOBController.UCOBNodeController_grantRole()

#检查角色注册是否成功

执行方法：
com.ucob.controller.UCOBController.UCOBNodeController_checkRole()

5.2.5 激活公司合约，

执行方法：
com.ucob.controller.UCOBController.UCOBNodeController_activeCorporation()

#查询公司合约账户余额

执行方法：
com.ucob.controller.CorporationController.CorporationStorageDemo_balanceOf()

5.2.6 按照上述步骤，再次注册若干个 UCOB 公司合约

5.3 单个公司内部的资金流出

执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_transfer()

5.4 多个公司之间的资金流转

5.4.1 在公司控制器合约地址 3 中，注册

“CORPORATION_CFO_ROLE”角色，必须以 CEO 角色调用该方法

执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_setRoleAdmin()

#检查角色是否注册成功：

执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_checkRole()

5.4.2 首先由用户调用接口进行跨公司转账申请

执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_transferOut()

5.4.3 SDK 端设置事件监听，由 CFO 核查并执行实际的转出操作

备注：测试阶段可以从网页端（WeBank web 组件）直接获取“key”。

执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_transferOutCallback

5.5 发行资金

发起提案，对提案进行投票，执行发行资金的操作，

前两步操作参考前面，发行操作如下：

执行方法：
com.ucob.controller.UCOBController.CorporationStorageDemo_issue()

5.6 添加多个联合项目

5.6.1 部署联合项目合约，发起投票，初始化联合项目合约

5.6.2 前两步操作参考前面，初始化操作如下：

执行方法：
com.ucob.controller.UCOBController.UCOBNodeController.addUnitedProject()

5.6.3 添加联合项目管理员

只能由每个公司 CEO 角色的账户进行添加操作，

执行方法：
com.ucob.controller.CorporationController.CorporationControllerDemo_addUnitedProjectManager()

5.7 添加单个公司项目

与普通合约部署操作相同，适用于单个公司内部项目需求。