**软件使用说明文档**

二零二三年九月

目录

[1. 区块链底层平台特性 1](#_Toc146557116)

[1.1. 架构 1](#_Toc146557117)

[1.1.1. 应用链 1](#_Toc146557118)

[1.1.2. 核心链 2](#_Toc146557119)

[1.2. 使用概念 2](#_Toc146557120)

[1.2.1. 节点 2](#_Toc146557121)

[1.2.2. 联盟链组织结构与用户 3](#_Toc146557122)

[1.2.3. 联盟链权限角色 5](#_Toc146557123)

[1.3. 权限 6](#_Toc146557124)

[1.4. 共识 7](#_Toc146557125)

[1.5. 智能合约 8](#_Toc146557126)

[1.6. 账本 9](#_Toc146557127)

[2. 运行环境 10](#_Toc146557128)

[2.1. 硬件要求 10](#_Toc146557129)

[2.2. 软件要求 10](#_Toc146557130)

[3. 操作手册 11](#_Toc146557131)

[3.1. 节点部署与启动 11](#_Toc146557132)

[3.1.1. 容器化部署方式 11](#_Toc146557133)

[3.1.2. 独立启动模式 13](#_Toc146557134)

[3.2. 节点的配置 23](#_Toc146557135)

[3.2.1. 查看网络状态 23](#_Toc146557136)

[3.2.2. 节点配置命令 24](#_Toc146557137)

[3.3. SDK使用 26](#_Toc146557138)

[3.3.1. 示例项目 26](#_Toc146557139)

[3.3.2. 示例项目构成 26](#_Toc146557140)

[3.4. 智能合约开发 26](#_Toc146557141)

[3.4.1. 安装rust环境 26](#_Toc146557142)

[3.4.2. 安装wasm编译插件 26](#_Toc146557143)

[3.4.3. 将合约编译成wasm文件 27](#_Toc146557144)

[3.4.4. 将wasm文件上传到应用链 27](#_Toc146557145)

[3.4.5. 部署合约 27](#_Toc146557146)

[3.4.6. 调用合约 28](#_Toc146557147)

[3.4.7. 查询合约 28](#_Toc146557148)

[3.4.8. 升级合约 28](#_Toc146557149)

[3.5. BaaS平台 29](#_Toc146557150)

[3.5.1. 登录 29](#_Toc146557151)

[3.5.2. 首页 29](#_Toc146557152)

[3.5.3. 通知中心 30](#_Toc146557153)

[3.5.4. 可视化监控大屏 32](#_Toc146557154)

[3.5.5. 用户管理 33](#_Toc146557155)

[3.5.6. 资源管理 37](#_Toc146557156)

[3.5.7. 应用链管理 39](#_Toc146557157)

[3.5.8. 核心链管理 42](#_Toc146557158)

[3.5.9. 节点管理 45](#_Toc146557159)

[3.5.10. 合约管理 47](#_Toc146557160)

[3.5.11. 跨链管理 50](#_Toc146557161)

[3.5.12. 提案管理 54](#_Toc146557162)

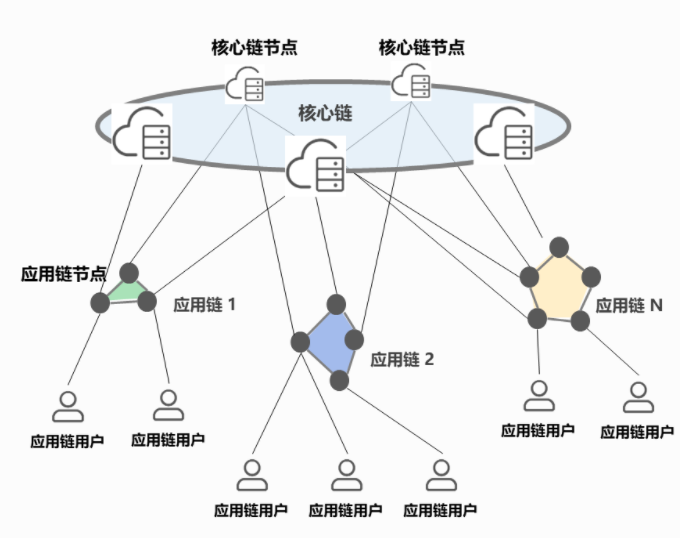
[3.6. Restful接口命令 55](#_Toc146557163)

[3.7. 出错与故障恢复 64](#_Toc146557164)

# 区块链底层平台特性

## 架构

区块链底层基础平台采用“分层多链”总体架构设计，如下图所示。“分层”是指区块链底层基础平台分为核心链和应用链两个层次，“多链”是指区块链底层基础平台能够容纳多条应用链并存，并且支持多条应用链接入核心链，即“1+N”结构。若跟通信网络结构进行类比的话，应用链相当于通信网络中的“接入网”，核心链相当于通信网络中的“核心网”概念。



### 应用链

应用链可以认为是一条具体的联盟链，应用链专门为特定应用场景建立，针对某项业务或者多个机构组建的应用联盟服务。应用链可以按照具体应用需求进行个性化定制（例如共识算法、智能合约语言等）。每条应用链具备独立完整账本（保持数据隔离性），应用链与应用链之间相互隔离通常不发生业务关系（跨链业务除外）。应用链具备独立的运营与治理能力，应用链内部的用户、应用链节点、智能合约部署、应用场景开发等，由应用链运营机构进行管理。

### 核心链

核心链是一条具备特定功能的联盟链，由城市级区块链基础设施运营机构负责建设和运营。核心链连接了多条应用链，并负责完成应用链的管理和注册，负责应用链之上的全局共识，以及应用链之间安全的消息传递，完成应用链与应用链之间的跨链服务功能。核心链不存储应用链内部数据，只存储应用链管理维护、跨链交易等相关的数据。

## 使用概念

### 节点

区块链网络主要由节点组成。节点是网络的基本元素，因为节点存储了账本和智能合约。账本不可篡改地保存着智能合约生成的所有交易。智能合约和账本将网络中共享的流程和信息对应地封装起来。

节点：节点可以被创建、启动、停止、重新配置甚至删除。他们暴露了一系列的 API，这就可以让管理者和应用程序同这些 API 提供的服务互动。

节点分为：

1）验证节点，参与出块、网络共识、交易执行；

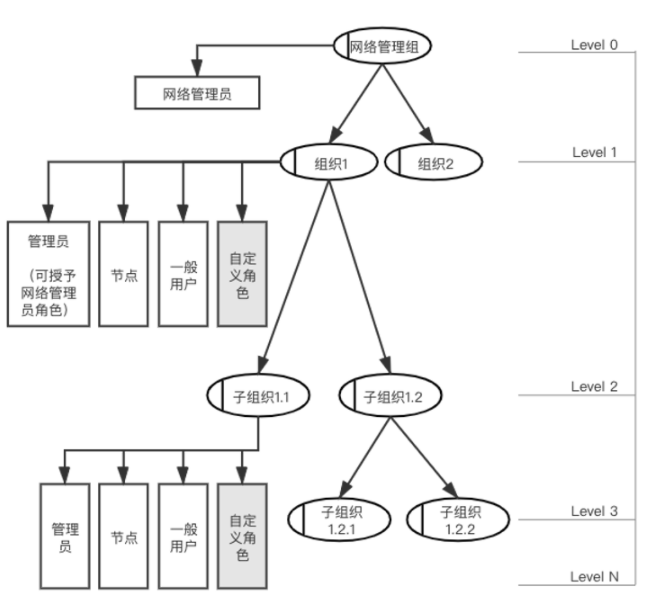
2）记账节点，只参与交易的同步和执行。

### 联盟链组织结构与用户

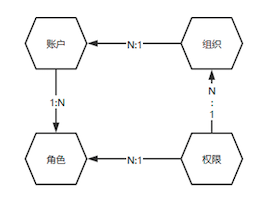
联盟链顶层管理角色（委员会）叫做网络管理组，可设有多名网络管理员，以及一级组织内的管理员共同参与。

以网络管理组为 root，可以向下建立各层级组织。组织内可添加多个账号，账号可赋予多种角色。

组织层级机构如下图所示：



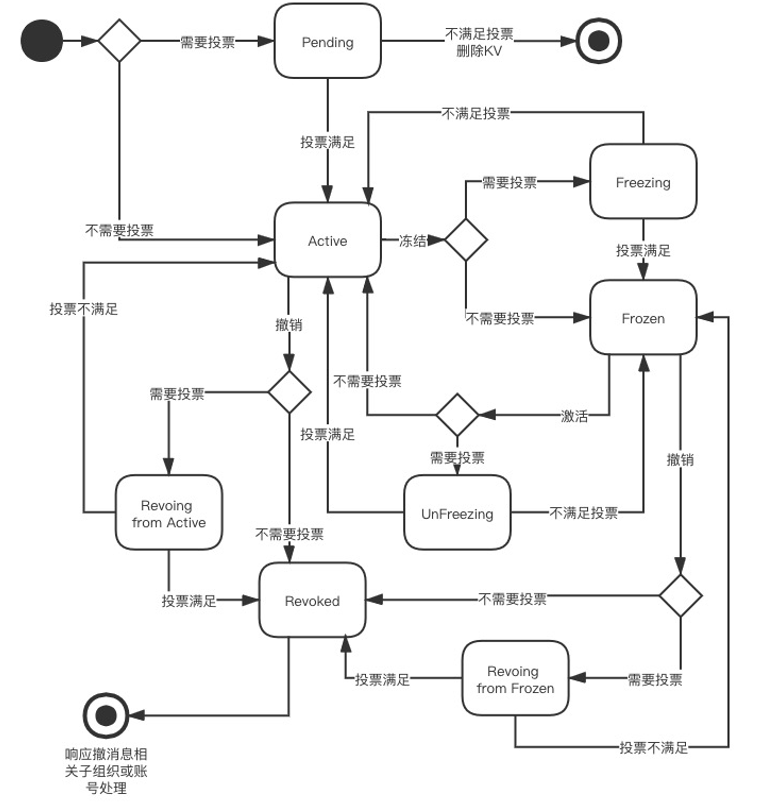
组织、账号、角色对应关系如下图所示：



组织定义：

* 网络管理组（Level 0）：设有网络管理员，链启动后初始化。
* 组织（Level 1）：由网络管理员创建，需要通过网络管理员们投票，半数以上批准同意。
* 子组织（Level 2-N）：由祖先的组织管理员创建，不能由网络管理员创建。不需要投票

组织的增删改查具有状态属性，状态及转移过程如下图所示，其中 Leve1 组织添加、冻结、激活、恢复、撤销等全生命周期的状态转换等需要网络管理员投票。



### 联盟链权限角色

角色总体分为系统预设角色和组织自定义角色。 预设角色包括：

* 网络管理员--委员会成员（nwAdmin）：参与全部需要投票的环节，包括 Level 1 组织，网络管理员、Level 1 组织管理员的变更操作等，在创世块中至少确定一个网络管理员。
* 组织管理员(admin)：本组织和子组织管理及账号管理。
* 节点(peer)：参与共识或记账。
* 一般用户(client)：各类模块和合约调用交易发起。
* 通配用户角色（member），所有角色继承自 member。 组织自定义角色，由各个组织自行创建，可继承自预设角色，也可以继承自已存在的自定义角色，角色的继承关系在权限检查过程中向上兼容。

账户定义：

可向组织添加账号由账号 Address 作为账号的唯一 Key 值，组织与账号一对多关联，账号可关联多个角色。

在网络管理组多个网络管理员账号，网络管理员参与一切需要操作投票的业务流程。Level 1 级别的组织管理可以被授予网络管理员角色，即同时拥有网络管理员和 Level 1 组织管理员角色，能够参与提案和投票。某种情况来讲，也可以删除掉网络管理组的网络管理员账号，而由多个 Level 1 的组织管理员联合担任。

对网络管理员和 Level 1 的组织管理员的增删改操作需要经过投票，同时账号的状态转移与组织章节状态转换一致。

组织管理员和子组织管理员每个组织内部有且只能有一个，不能做删除操作只能做替换操作。

## 权限

权限定义：

交易执行权限主要针对模块的交易执行和合约的执行权限控制，权限 struct 主要由以下几个属性组成

* 资源名称（Resource = ModuleName MsgType）：由模块名称和消息类型，针对某个交易消息进行权限检查。例如：member addOrg
* 策略：ACCEPT/DROP，白名单或黑名单模式。
* Status：参考组织中状态转换，模块相关的权限修改需要网络管理员提案与投票，合约权限。
* 合约管理员：可以由合约上传者指定，管理员和合约者可以修改合约执行权限，不需要投票。未来考虑管理组+读写权限等。
* 权限表达式列表：每个权限表达式，由组织表达式列表和角色 ID 列表构成，权限的判断由组织表达式列表和角色 ID 列表的笛卡尔积决定（组织表达式是或的关系，角色 ID 是或的关系），返回值为 bool 类型，true 为通过，false 为拒绝。权限表达式之间是或的关系，增加权限表达式意味着放宽权限的范围。
* 组织表达式支持精确匹配和模糊匹配，精确匹配形如org1.dep1.group1，模糊匹配支持\*、\*\*通配符，\*代表该层级的任意组织名，\*\*匹配任意深度的组织名。\*\*匹配任意Level组织。org1.dep1.group1匹配举例：

org1.dep1.\*

\*.\*.group1

org1.\*\*（适配org1，本身及其所有子组织）

\*\*.group1

* 完整的全新表达式举例：

(org1.\*, org2.\*\*.h) & (peer, client) || org1 & peer || org2 & client || org2&member || \*\*&client

## 共识

城域区块链基础设施链网协同平台为了简化不同共识算法的适配问题，将设计一套可插拔的共识机制，为城域区块链基础设施链网协同平台提供一系列的共识公共组件，支持多种共识机制无缝切换，支持动态的调整网络拓扑，实现节点动态加入及主动退出功能。实现了对于主流共识算法的可插拔化支持，任意一种共识算法都可以通过可插拔的方式选择对应的框架和结构，从而使城域区块链基础设施链网协同平台的实际业务落地中有选择性的使用一种最合适的共识机制。

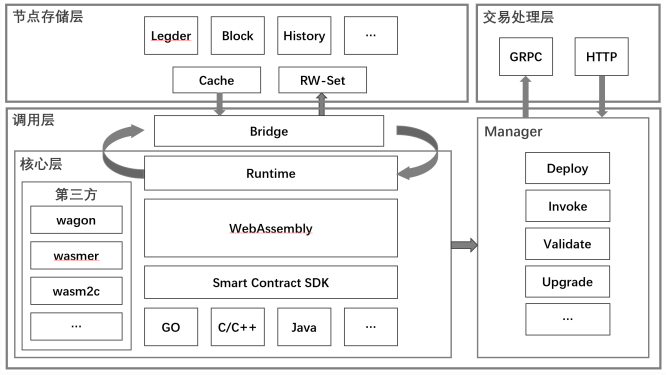
该可插拔共识系统分为两层：

* 底层为实现共识通用功能模块的共识公共组件；
* 上层为实现共识独有功能模块的可插拔交互共识算法。

双层架构将多种共识算法通用和独有的模块功能进行了颗粒化的区分和抽象，从而实现更灵活更有效的调用。不同的共识引擎之间应当能够进行切换，使用者可以根据自身的需要对具体的功能模块进行组合，以达到特定的功能或性能要求。

## 智能合约

城域区块链基础设施链网协同平台智能合约系统采用基于WASM的虚拟机方式实现，具备独立性、图灵完备、多语言支持、灵活扩展等特性，其与区块链的整体架构分层如下图所示：



* 核心层：智能合约核心逻辑实现，提供WASM第三方工具处理、多语言SDK支持、虚拟机运行环境系统方法、合约的编译、实例化、运行等核心生命周期方法等；
* 调用层：作为智能合约与节点通讯的入口，主要分为Manager和Bridge两个部分，Manager为节点提供智能合约全生命周期管理入口；Bridge则作为智能合约与节点通讯的实现，包括wasm虚拟机与节点、wasm虚拟机与账本或存储层等；
* 节点存储层：实现区块链数据存储缓存机制，统一处理智能合约读写集与世界账本和区块、交易数据相关；
* 交易处理层：区块链接入层负责交易转换、签名、验证并通过参数与合约调用层进行通讯实现智能合约的全生命周期控制；

## 账本

根据雄安城市级区块链建设需求，构建分层多链分布式账本；针对不同的业务应用场景，构建数据隔离、链间安全和高可扩展的账本；减少节点数据存储和处理压力，为上层业务需求提供更加快速的数据支撑能力和业务服务能力，满足复杂度越来越高区块链应用场景的需求。

针对账本数据存储需求提供高性能、高扩展的底层存储，支持可插拔的多重底层存储引擎，针对不同的应用场景，构建用户自主可控的区块链异构数据存储，提供在不同业务场景下对海量数据存储和高速检索的支持。

# 运行环境

## 硬件要求

最小硬件要求

|  |  |
| --- | --- |
| CPU | 1核1GHz |
| 内存 | 4G |
| 网络 | 带宽1M |
| 硬盘 | 50G |

## 软件要求

|  |  |
| --- | --- |
| JDK | 1.8 |
| Go | 1.14 |
| 系统环境 | Ubuntu 14及以上、CentOS 7及以上 |

# 操作手册

## 节点部署与启动

### 容器化部署方式

#### 启动节点

* 下载文件xchain-alone.tar。

下载路径<https://gitee.com/xabl/xchain/blob/master/xchain.tar>

* 导入docker镜像docker load < xchain.tar
* 运行节点

mkdir -p ~/node1/.xcd

mkdir -p ~/node1/.xccli

docker run -d --name xchain\_node1 -v ~/node1/.xcd:/root/.xcd -v ~/node1/.xccli:/root/.xccli -p 26656:26656 -p 1317:1317 -p 26657:26657 xchain:1.0.2

其中数据目录可根据需求重新自定义。

进入容器内部，可查看init\_genesis.sh文件。创建了jack为网络管理员，创建了组织org1，管理员名称为org1Admin，通过 docker logs xchain\_node1查看jack和org1Admin的助记词配合SDK使用。

用户账号举例

{

"name": "jack",

"type": "local",

"address": "xchain1qtjfjvpclf5em8yuxvyekhqpyt4qsj0apscj9e",

"pubkey": "xchainpub1ulx45dfpq2t06qcx5q0qj9naet5pq74jzus6jtu4jdgjr83x03x93pr3a7aakvjr8yx",

"mnemonic": "banner round pupil crew direct negative ankle smart entry west balcony large october mom duty absurd hire custom puppy another fitness clap fancy morning"

}

至此一个最简单单节点网络已经可以运行和相关测试。

查看docker下有哪些容器docker ps

进入xchain容器命令行里docker exec -it xchain /bin/bash

#### 扩展网络加入新节点

* 复制创世块配置文件

将拟加入区块链的创世块文件~/node1/.xcd/config/genesis.json复制到该节点的数据目录下~/node2/.xcd/config/genesis.json。

* 启动第2个节点容器

NODE\_ID=xcd tendermint show-node-id

docker run -d --name xchain\_node2 -v ~/node2/.xcd:/root/.xcd -v ~/node2/.xccli:/root/.xccli -p 36656:26656 -p 2317:1317 -p 36657:26657 -e NODEORDER=follow xchain:1.0.2 --p2p.persistent\_peers=$NODE\_ID@xchain\_node1:26656

--p2p.persistent\_peers为已加入区块链节点的节点信息。

nodeId获取命令：xcd tendermint show-node-id

* 获取第二个节点的地址

NODE\_ADDRESS=$(docker exec xchain\_node2 xcd tendermint show-node-address)

* 在节点1中授权加入第2个节点

docker exec xchain\_node1 xccli tx member addAccount $NODE\_ADDRESS org1 peer --from jack -y

#### 容器四节点启动模式

* 启动./xchain\_net.sh up

文件路径：https://gitee.com/xabl/xchain/blob/master/xchain\_net.sh

* 关闭./xchain\_net.sh down

### 独立启动模式

#### 节点程序简介

* xcd：节点服务程序；
* xccli：客户端命令行工具；
* so：xcd和xccli运行依赖库；

以上程序下载路径：https://gitee.com/xabl/xchain

#### 单节点启动

* xcd、xccli拷贝到linux服务器工作目录
* so拷贝到linux服务器/lib目录
* 运行/sbin/ldconfig -v 加载依赖库
* 将xcd、xccli路径加入PATH环境变量
* 开设区块链

1. 运行 init\_genesis.sh脚本。 #注意保存jack与org1Admin的助记词，在sdk中需要引用

2. 执行xcd start启动节点。

3. 执行xccli status查看节点状态。

#### 多节点启动

* 先在一台机器启动一个节点作为第一个节点，跟单节点启动一样。
* 在第二个节点运行 init\_follow.sh脚本。
* 将拟加入区块链的创世块文件genesis.json复制到该节点的数据目录下($HOME/.xcd/config/)。
* 将本节点的节点地址通过已有节点加入到网络。

节点地址获取命令：

xcd tendermint show-node-address

在第一个节点查看有哪些组织：

xccli query member orgs

选择其中一个组织，看orgAdminAddress属性存储的组织管理员地址，例如以下org2组织的管理员地址为xchain1tsat26zcw2gr4c4ldejs527jrzvhhhc253y2s6：

{

"orgId": "org2",

"orgFullId": "org2",

"level": "1",

"orgAdminAddress": "xchain1tsat26zcw2gr4c4ldejs527jrzvhhhc253y2s6",

"subOrgIds": null,

"orgAccounts": [

"xchain1tsat26zcw2gr4c4ldejs527jrzvhhhc253y2s6",

"xchain1u5gpat4l7u36ph9m73g8s3qj8drgu4l5hcadv6"

],

"orgRoleIds": null,

"parentId": "",

"ultParent": "org2",

"status": "1",

"preRevokeStatus": "0"

},

在第一个节点执行xccli keys list，在结果集合中对比刚才查到的组织管理员地址，查出组织管理员的名字。例如xchain1tsat26zcw2gr4c4ldejs527jrzvhhhc253y2s6的名字叫org2Admin。

{

"name": "org2Admin",

"type": "local",

"address": "xchain1tsat26zcw2gr4c4ldejs527jrzvhhhc253y2s6",

"pubkey": "xchainpub1ulx45dfpqvel03xt6y5xc0s5kfszswempn32dz2gr3lllc2mu34xnqrw3vskkywrr56"

},

在第一个节点增加账号：xccli tx member addAccount <第二个节点的地址> <组织名> peer –from <组织管理员>，例如：

xccli tx member addAccount xchain1wqdf5f66tdmj0dh8pyptppaqu2w2569z033vne org2 peer --from org2Admin

执行完后，在第一个节点查询组织，看第二个节点有没有成功加到某个组织里。以下是成功加到某组织的例子：

xccli query member orgs

{

"orgId": "org2",

"orgFullId": "org2",

"level": "1",

"orgAdminAddress": "xchain1tsat26zcw2gr4c4ldejs527jrzvhhhc253y2s6",

"subOrgIds": null,

"orgAccounts": [

"xchain1tsat26zcw2gr4c4ldejs527jrzvhhhc253y2s6",

"xchain1u5gpat4l7u36ph9m73g8s3qj8drgu4l5hcadv6",

"xchain1wqdf5f66tdmj0dh8pyptppaqu2w2569z033vne"

],

"orgRoleIds": null,

"parentId": "",

"ultParent": "org2",

"status": "1",

"preRevokeStatus": "0"

},

5.在第二个节点执行命令启动节点

xcd start --p2p.persistent\_peers=nodeId@host:ip

--p2p.persistent\_peers为已加入区块链节点的节点信息。 nodeId获取命令：xcd tendermint show-node-id

第二个节点成功启动并且连上第一个节点的输出例子：

I[2021-01-19|15:38:17.841] starting ABCI with Tendermint module=main

E[2021-01-19|15:38:17.898] Can't add peer's address to addrbook module=p2p err="Cannot add non-routable address 1563d78a654fbcd238f5bd6acf3700f066544561@192.168.0.25:26656"

Query path: /p2p/filter/addr/192.168.0.25:26656

Query path: /p2p/filter/id/1563d78a654fbcd238f5bd6acf3700f066544561

I[2021-01-19|15:38:21.153] Executed block module=state height=1 validTxs=0 invalidTxs=0

I[2021-01-19|15:38:21.156] Committed state module=state height=1 txs=0 appHash=32298BE2D8A95C0F0692C03FD641ADE32153BB5D8E109A3D5C8CFA4DEF5D4961

I[2021-01-19|15:38:21.161] Executed block module=state height=2 validTxs=0 invalidTxs=0

I[2021-01-19|15:38:21.163] Committed state module=state height=2 txs=0 appHash=32298BE2D8A95C0F0692C03FD641ADE32153BB5D8E109A3D5C8CFA4DEF5D4961

I[2021-01-19|15:38:21.169] Executed block module=state height=3 validTxs=1 invalidTxs=0

I[2021-01-19|15:38:21.170] Committed state module=state height=3 txs=1 appHash=1A01D03323A2802FE63B3EDDBEC07C00A63E38D9006AA61B1B23F24BEA5F2BDE

I[2021-01-19|15:38:21.177] Executed block module=state height=4 validTxs=0 invalidTxs=0

I[2021-01-19|15:38:21.179] Committed state module=state height=4 txs=0 appHash=1A01D03323A2802FE63B3EDDBEC07C00A63E38D9006AA61B1B23F24BEA5F2BDE

I[2021-01-19|15:38:21.185] Executed block module=state height=5 validTxs=1 invalidTxs=0

I[2021-01-19|15:38:21.187] Committed state module=state height=5 txs=1 appHash=D0ADCCA8656C9C27F49FD9E60A732CC4D502F26D1DE1CB914DD09C5B1F47B1D4

I[2021-01-19|15:38:21.194] Executed block module=state height=6 validTxs=0 invalidTxs=0

I[2021-01-19|15:38:21.196] Committed state module=state height=6 txs=0 appHash=D0ADCCA8656C9C27F49FD9E60A732CC4D502F26D1DE1CB914DD09C5B1F47B1D4

I[2021-01-19|15:38:21.202] Executed block module=state height=7 validTxs=1 invalidTxs=0

I[2021-01-19|15:38:21.204] Committed state module=state height=7 txs=1 appHash=A7CC3F313EB3586F75800B1507C7BCF80659AE9D5C71C74C5751B1A47719066E

I[2021-01-19|15:38:22.669] Executed block module=state height=8 validTxs=0 invalidTxs=0

I[2021-01-19|15:38:22.671] Committed state module=state height=8 txs=0 appHash=A7CC3F313EB3586F75800B1507C7BCF80659AE9D5C71C74C5751B1A47719066E

第二个节点成功连上第一个节点，第一个节点之前xcd start启动的界面会继续输出连上的信息，得到第二个节点的id，701a9a275a5b7727b6e70902b087a0e29caa68a2，例如（最后三行）：

I[2021-01-20|01:38:27.706] starting ABCI with Tendermint module=main

Query path: custom/acc/account

Query path: custom/acc/account

I[2021-01-20|01:41:08.170] Executed block module=state height=5 validTxs=1 invalidTxs=0

I[2021-01-20|01:41:08.177] Committed state module=state height=5 txs=1 appHash=D0ADCCA8656C9C27F49FD9E60A732CC4D502F26D1DE1CB914DD09C5B1F47B1D4

I[2021-01-20|01:41:13.210] Executed block module=state height=6 validTxs=0 invalidTxs=0

I[2021-01-20|01:41:13.215] Committed state module=state height=6 txs=0 appHash=D0ADCCA8656C9C27F49FD9E60A732CC4D502F26D1DE1CB914DD09C5B1F47B1D4

Query path: custom/member/orgs

Query path: custom/acc/account

Query path: custom/acc/account

I[2021-01-20|01:50:06.992] Executed block module=state height=7 validTxs=1 invalidTxs=0

I[2021-01-20|01:50:06.995] Committed state module=state height=7 txs=1 appHash=A7CC3F313EB3586F75800B1507C7BCF80659AE9D5C71C74C5751B1A47719066E

I[2021-01-20|01:50:12.012] Executed block module=state height=8 validTxs=0 invalidTxs=0

I[2021-01-20|01:50:12.017] Committed state module=state height=8 txs=0 appHash=A7CC3F313EB3586F75800B1507C7BCF80659AE9D5C71C74C5751B1A47719066E

Query path: custom/member/orgs

Query path: /p2p/filter/addr/172.17.0.1:59384

Query path: /p2p/filter/addr/172.17.0.1:59388

Query path: /p2p/filter/id/701a9a275a5b7727b6e70902b087a0e29caa68a2

6.在第一个节点做交易，jack转账给bob，xccli tx send < jack address > < bob address > 55stake –from jack。

jack和bob的地址可以用以下代码查询：

xccli keys list

查询结果例子：

{

"name": "bob",

"type": "local",

"address": "xchain1k8dng03r58gqqa4qafuvtngsg2htx2ha5qp720",

"pubkey": "xchainpub1ulx45dfpqw7920nduaulukgy8levxd4mwq5vyhc5p0ykgnvwlzdwsx83xw5773ycuqq"

},

...

{

"name": "jack",

"type": "local",

"address": "xchain1gptswzmh68q5l8jskvvtm8s664v83u3nt7vds9",

"pubkey": "xchainpub1ulx45dfpqw7yqwvejdysxarsvr98kqng4wxavqun0mkdhafhm8klxs05ure06tseqxt"

},

执行转账例子：

xccli tx send xchain1gptswzmh68q5l8jskvvtm8s664v83u3nt7vds9 xchain1k8dng03r58gqqa4qafuvtngsg2htx2ha5qp720 55stake --from jack

查看第一、第二个节点有没有自动出块。

第一个节点自动出块输出例子：

Query path: custom/acc/account

Query path: custom/acc/account

I[2021-01-20|02:31:37.039] Executed block module=state height=9 validTxs=0 invalidTxs=1

I[2021-01-20|02:31:37.042] Committed state module=state height=9 txs=1 appHash=ABDA51616E6EF12138EEB713A7F0AA5778BC6CF6E608B07A95E49F8688B92DEA

I[2021-01-20|02:31:42.092] Executed block module=state height=10 validTxs=0 invalidTxs=0

I[2021-01-20|02:31:42.106] Committed state module=state height=10 txs=0 appHash=ABDA51616E6EF12138EEB713A7F0AA5778BC6CF6E608B07A95E49F8688B92DEA

第二个节点自动出块输出例子：

I[2021-01-19|16:05:12.765] Executed block module=state height=9 validTxs=0 invalidTxs=1

I[2021-01-19|16:05:12.768] Committed state module=state height=9 txs=1 appHash=ABDA51616E6EF12138EEB713A7F0AA5778BC6CF6E608B07A95E49F8688B92DEA

I[2021-01-19|16:05:17.811] Executed block module=state height=10 validTxs=0 invalidTxs=0

I[2021-01-19|16:05:17.813] Committed state module=state height=10 txs=0 appHash=ABDA51616E6EF12138EEB713A7F0AA5778BC6CF6E608B07A95E49F8688B92DEA

7.执行以下代码获取所有validator

xccli query poa validators

当时只有node01有validator：

[

{

"operator\_address": "xchain1gptswzmh68q5l8jskvvtm8s664v83u3nt7vds9",

"validator\_address": "xchainvaloper1gwjucjm8ccnmk9kngpa4ypyakuf9nkx9r4ne5u",

"consensus\_pubkey": "xchainvalconspub1ulx45dfpqv8kqgdj8t0w4mwup3fl5dvcjw8g599zkgnlnrehhztf0c0wsudav40f38c",

"jailed": false,

"status": 2,

"weight": "100",

"pending\_weight": "0",

"description": {

"moniker": "node01",

"identity": "",

"website": "",

"security\_contact": "",

"details": ""

}

}

]

* 在第二个节点执行以下代码，取得第二个节点的validator key

xcd tendermint show-validator

* 在第一或者第二个节点执行xccli tx poa createValidator < validator key > power(0~100)，创建第二个节点的validator，例如：

xccli tx poa createValidator xchainvalconspub1ulx45dfpqdt550rmnakdu7pfal700pucs3rrsxyp04u9cq6ln9g0uhfhz9etuc5l0rs 10 --moniker node02 --from jack -y

执行完会返回txhash等信息，例如：

{

"height": "0",

"txhash": "BF04B75B5BAC0D7873594A8D3FC263D4F1E01E287182AE895649D7474EF5A1AF",

"raw\_log": "[]"

}

* 可以用xccli query tx < txhash >查询刚才交易信息，例如：

xccli query tx BF04B75B5BAC0D7873594A8D3FC263D4F1E01E287182AE895649D7474EF5A1AF

* 执行以下代码获取所有validator

xccli query poa validators

返回node01和node02的validator信息：

[

{

"operator\_address": "xchain1gptswzmh68q5l8jskvvtm8s664v83u3nt7vds9",

"validator\_address": "xchainvaloper1gwjucjm8ccnmk9kngpa4ypyakuf9nkx9r4ne5u",

"consensus\_pubkey": "xchainvalconspub1ulx45dfpqv8kqgdj8t0w4mwup3fl5dvcjw8g599zkgnlnrehhztf0c0wsudav40f38c",

"jailed": false,

"status": 2,

"weight": "100",

"pending\_weight": "0",

"description": {

"moniker": "node01",

"identity": "",

"website": "",

"security\_contact": "",

"details": ""

}

},

{

"operator\_address": "xchain1gptswzmh68q5l8jskvvtm8s664v83u3nt7vds9",

"validator\_address": "xchainvaloper1tjgtsuep5uxssv74drcj5ctaztc56xc63s3224",

"consensus\_pubkey": "xchainvalconspub1ulx45dfpqdt550rmnakdu7pfal700pucs3rrsxyp04u9cq6ln9g0uhfhz9etuc5l0rs",

"jailed": false,

"status": 2,

"weight": "10",

"pending\_weight": "0",

"description": {

"moniker": "node02",

"identity": "",

"website": "",

"security\_contact": "",

"details": ""

}

}

]

* 如果需要启动第三个节点并加入第一、第二个节点搭建起来的网络，则第三个节点启动步骤和第二个节点类似，第三个节点连第一个节点或者第二个节点均可。

## 节点的配置

### 查看网络状态

查看xchain网络状态，前提是一个简单的单点部署xchain网络场景已启动。配置文件和启动文件已预定义。

在命令行中输入xccli status，获取 网络名称、channel、rpc\_address等信息。

xccli status

root@f54752ddbd9b:/opt# xccli status

{

"node\_info": {

"protocol\_version": {

"p2p": "7",

"block": "10",

"app": "0"

},

"id": "1563d78a654fbcd238f5bd6acf3700f066544561",

"listen\_addr": "tcp://0.0.0.0:26656",

"network": "xchain",

"version": "0.33.8",

"channels": "4020212223303800",

"moniker": "node01",

"other": {

"tx\_index": "on",

"rpc\_address": "tcp://0.0.0.0:26657"

}

},

"sync\_info": {

"latest\_block\_hash": "DD863E05EB70224AF9E0160B9E57100A424D41FE95D5DF17A0BF555754788354",

"latest\_app\_hash": "32298BE2D8A95C0F0692C03FD641ADE32153BB5D8E109A3D5C8CFA4DEF5D4961",

"latest\_block\_height": "2",

"latest\_block\_time": "2021-01-18T03:01:47.3004822Z",

"earliest\_block\_hash": "B0D8C9E74F82488F4FCE1F6F278AA3F3D121DD7388E0BD11490FAA0AF378368C",

"earliest\_app\_hash": "",

"earliest\_block\_height": "1",

"earliest\_block\_time": "2021-01-06T02:39:54.4126563Z",

"catching\_up": false

},

"validator\_info": {

"address": "43A5CC4B67C627BB16D3407B52049DB71259D8C5",

"pub\_key": {

"type": "tendermint/PubKeySm2",

"value": "Aw9gIbI63urt3AxT+jWYk46KFKKyJ/mPN7iWl+HuhxvW"

},

"voting\_power": "100"

}

}

### 节点配置命令

节点向其他模块提供节点启动，配置、管理及对链上的查询和交易操作的接口。

使用xcd命令和xccli命令可对xchain进行操作，其中xcd可对服务器节点进行启动和配置等操作，xccli提供客户端命令以进行链上查询和交易等操作。

root@b8fee4e6a926:/opt# xcd

xiongan App Daemon (server)

Usage:

xcd [command]

Available Commands:

init Initialize private validator, p2p, genesis, and application configuration files

collect-gentxs Collect genesis txs and output a genesis.json file

gentx Generate a genesis tx carrying a self delegation

validate-genesis validates the genesis file at the default location or at the location passed as an arg

add-genesis-account Add a genesis account to genesis.json

add-genesis-admin Add a genesis admin to genesis.json

add-genesis-org Add a genesis org to genesis.json

debug Tool for helping with debugging your application

start Run the full node

unsafe-reset-all Resets the blockchain database, removes address book files, and resets priv\_validator.json to the genesis state

tendermint Tendermint subcommands

export Export state to JSON

version Print the app version

help Help about any command

Flags:

-h, --help help for xcd

--home string directory for config and data (default "/root/.xcd")

--log\_level string Log level (default "main:info,state:info,\*:error")

--trace print out full stack trace on errors

Use "xcd [command] --help" for more information about a command.

root@b8fee4e6a926:/opt# xccli

xiongan chain client

Usage:

xccli [command]

Available Commands:

status Query remote node for status

config Create or query an application CLI configuration file

query Querying subcommands

tx Transactions subcommands

rest-server Start LCD (light-client daemon), a local REST server

keys Add or view local private keys

version Print the app version

help Help about any command

Flags:

--chain-id string Chain ID of tendermint node

-e, --encoding string Binary encoding (hex|b64|btc) (default "hex")

-h, --help help for xccli

--home string directory for config and data (default "/root/.xccli")

-o, --output string Output format (text|json) (default "text")

--trace print out full stack trace on errors

Use "xccli [command] --help" for more information about a command.

## SDK使用

### 示例项目

将项目目录下的sdkdemo.zip解压，得到SDK的示例项目，项目代码中包含了完整的SDK调用过程。

项目地址https://gitee.com/xabl/xchain/blob/master/sdkdemo.zip

### 示例项目构成

SDK的jar包在src\main\resources\lib\目录下。

测试代码在src\test\java\com\huoshu\sdkdemo\目录下。

测试代码包括了调用合约和查询合约的测试方法，url需要将ip替换为所部署区块链的地址，mnemonic是账户的助记词，可在创建账户时获取。

## 智能合约开发

### 安装rust环境

https://www.rust-lang.org/zh-CN/tools/install

通过以下命令测试环境安装结果:

rustc --version

rustup --version

### 安装wasm编译插件

rustup target add wasm32-unknown-unknown

### 将合约编译成wasm文件

cargo build --release --target wasm32-unknown-unknown

编译完的wasm文件在target/wasm32-unknown-unknown/release/crud.wasm路径下。

已编译好的示例合约下载

### 将wasm文件上传到应用链

xccli tx wasm store crud.wasm rust [policy] --label [text] --execute-perm [contract execute permission] --from jack --gas="50000000" -y

policy:

1/ACCEPT 2/DROP，合约调用权限的白名单或黑名单模式。

组织表达式支持精确匹配和模糊匹配，精确匹配形如org1.dep1.group1，模糊匹配支持\*、\*\*通配符，\*代表该层级的任意组织名，\*\*匹配任意深度的组织名。\*\*匹配任意Level组织。

示例:

xccli tx wasm store crud.wasm rust 1 --execute-perm "org1&member" --from jack --gas="50000000" -y

组织org1下的所有成员均可调用。

xccli tx wasm store crud.wasm rust 2 --from jack --gas="50000000" -y

所有用户均可调用。

通过以下命令查询到上传合约的code\_id

xccli query tx wasm list-code

### 部署合约

xccli tx wasm instantiate [code\_id] "{}" [合约名称] --from jack --gas="80000000" -y

示例:

xccli tx wasm instantiate 1 "{}" test\_contract --from jack --gas="80000000" -y

### 调用合约

PUT='{"create":{"key":"key1","value":"value1"}}'

xccli tx wasm execute [合约名称] "$PUT" --from jack --gas="800000" -y

示例:

xccli tx wasm execute test\_contract "$PUT" --from jack --gas="800000" -y

### 查询合约

QUERY='{"find":{"key":"key1"}}'

xccli query wasm contract-state smart [合约名称] "$QUERY"

示例:

xccli query wasm contract-state smart test\_contract "$QUERY"

### 升级合约

升级合约需将新编译的wasm文件执行第四步，上传到雄安链，然后将新的code\_id作为参数。

xccli tx wasm migrate [合约名称] [new\_code\_id] "{}" --from jack --gas="800000" -y

示例:

xccli tx wasm migrate test\_contract 2 "{}" --from jack --gas="800000" -y

## BaaS平台

### 登录

#### 功能简介

输入分配的账号密码，登录系统

#### 前置条件

平台管理员已经在系统中创建了组织信息和组织管理员的账号信息

#### 操作说明

输入账号、密码、随机验证码，点击登录

### 首页

#### 功能简介

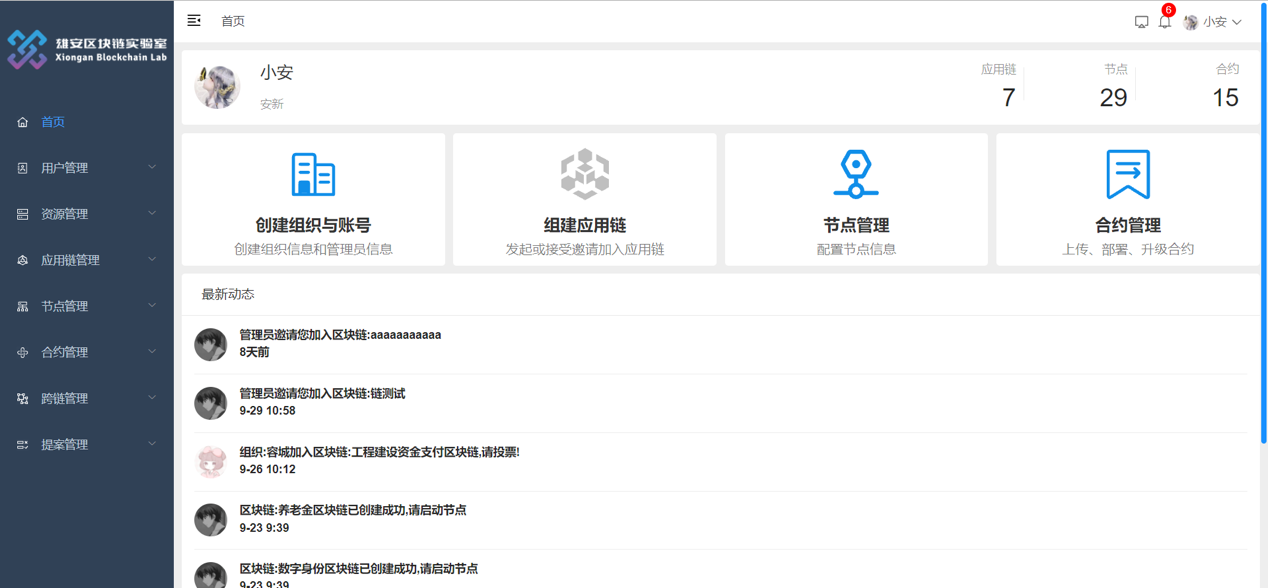
登录系统后，跳转至系统首页。首页展示了组织管理员的姓名、头像和组织名称，应用链相关的一些统计信息，如已经加入应用链的个数，节点和合约的个数。页面中部展示常用功能卡片，点击卡片跳转至对应功能页面。下方展示了与登录账号相关的近期动态。

#### 前置条件

组织管理员登入系统。

#### 操作说明

1. 点击页面中部的功能卡片，跳转至对应的功能模块。



### 通知中心

#### 功能简介

1. 该功能集中显示系统与登录账号相关的所有消息通知，包含待阅的通知类消息和待办的任务类消息。可以查看消息详情，也可以直接对任务类消息进行投票等操作。

#### 前置条件

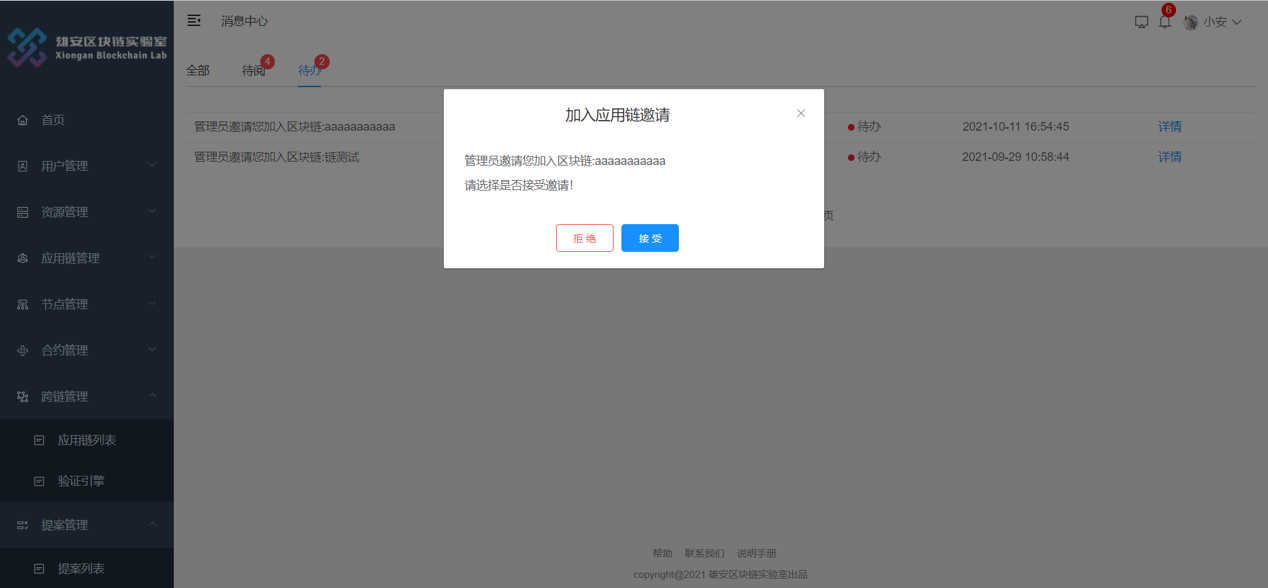
组织管理员登入系统，收到站内消息。

#### 操作说明

1. 点击右上角头像前的小铃铛图标，进入消息中心的页面。默认展示全部的消息，待阅和待办列表右上角红色数字表示对应类别未处理的条目。



1. 消息详情。通知类的消息详情，点击详情按钮，则该消息由未读更新为已读状态；任务类消息详情，点击详情按钮，打开弹窗，根据具体任务详情展示对应的操作，操作完成后，更新消息状态。



### 可视化监控大屏

#### 功能简介

该功能将平台运行状态和系统产生的数据进行了集中的可视化展示。

#### 前置条件

管理员登入系统。

#### 操作说明

1. 点击左上角小铃铛前边的图标，进入可视化大屏页面。按下浏览器的全屏快捷键，可以使展示效果更加友好。



### 用户管理

#### 组织列表

##### 功能简介

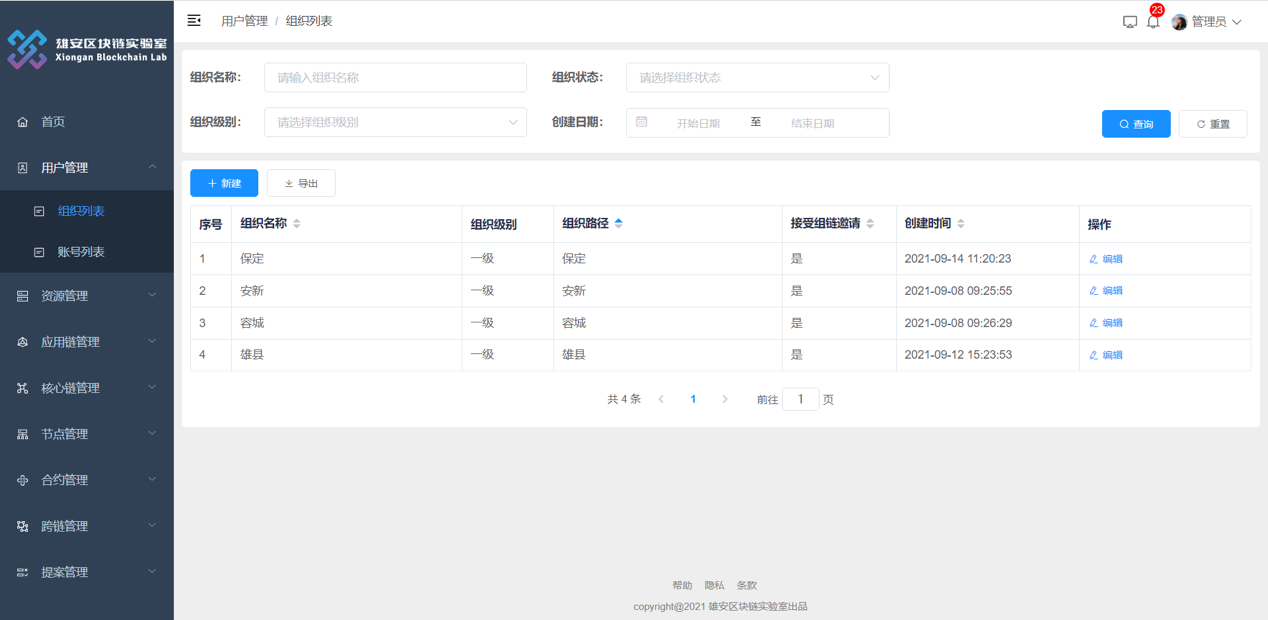
该功能展示了账号所属组织和所属组织下级组织的信息

##### 前置条件

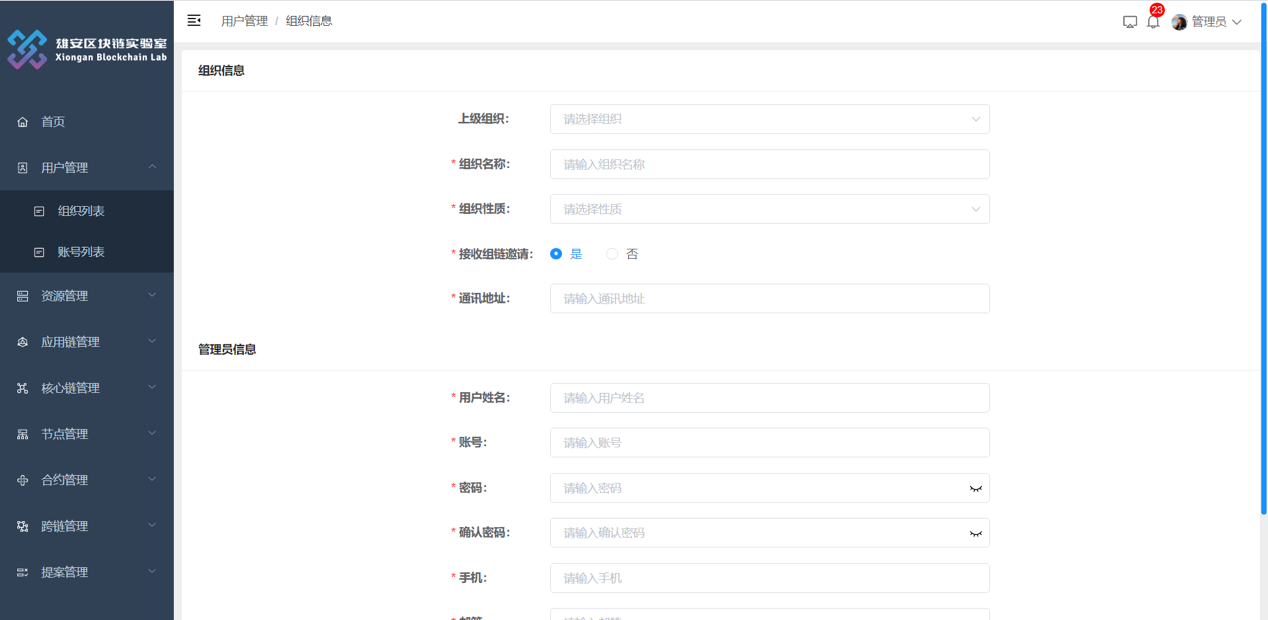
管理员已登入系统。

##### 操作说明

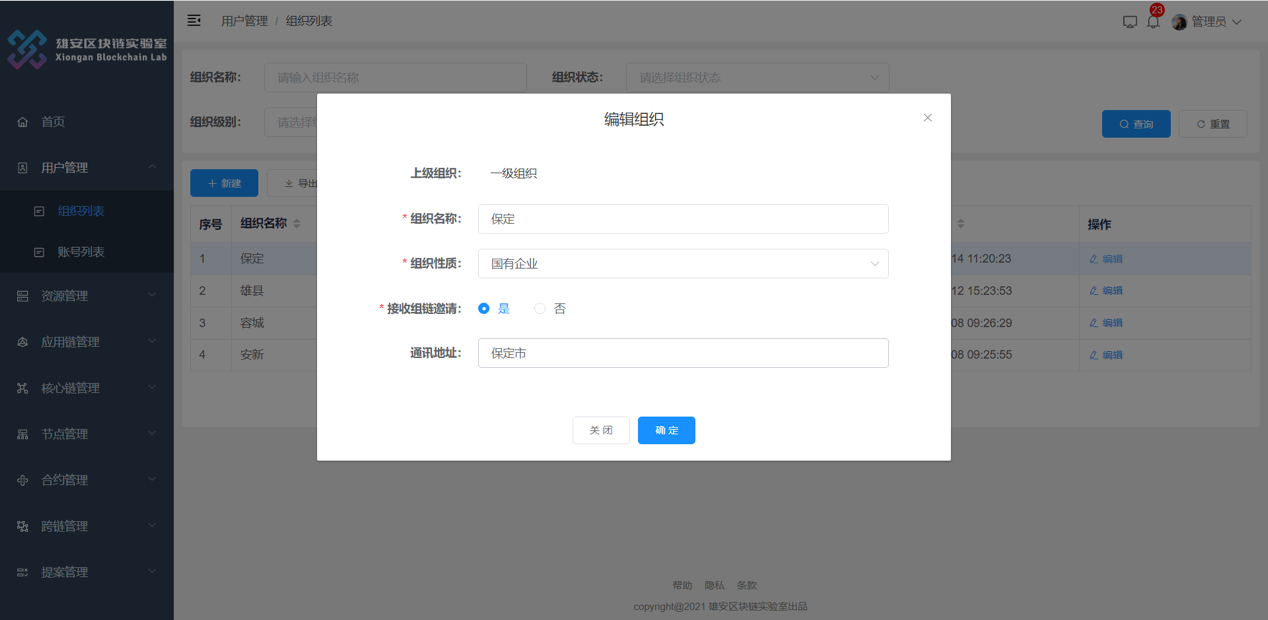
1. 输入查询条件，点击查询，列表展示相应数据。



1. 点击新建，打开新页面，输入对应信息，点击“提交”按钮，同事创建组织和组织的管理员账号信息。



1. 点击列表中操作列编辑，可以更改组织基本信息，修改是否接收组链邀请，选择“否”则不接受任何组织的邀请。



1. 点击列表上方的“导出”按钮，可以将表格中的数据导出为Excel文件。

#### 账号列表

##### 功能简介

该功能展示了组织下所有的用户账号信息，同事可以编辑账号的用户姓名、手机、邮箱信息，操作列日志展示账号启停历史信息，包括启停时间、原因、操作人、附件。

##### 前置条件

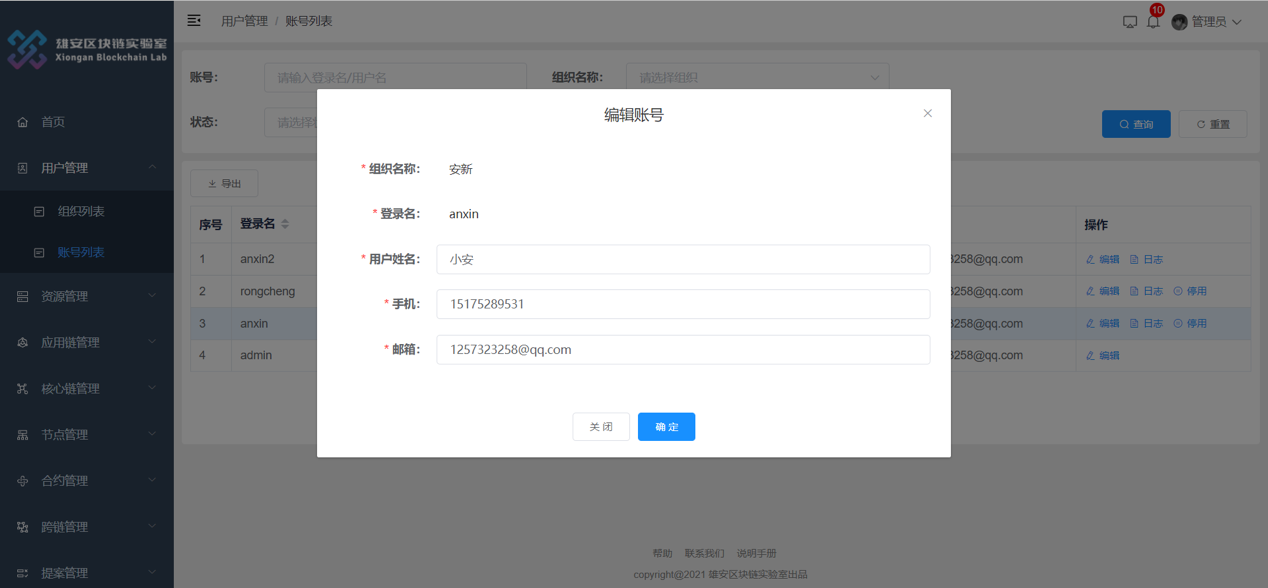
组织管理员已登入系统。

##### 操作说明

1. 输入查询条件，点击查询，列表展示对应信息。



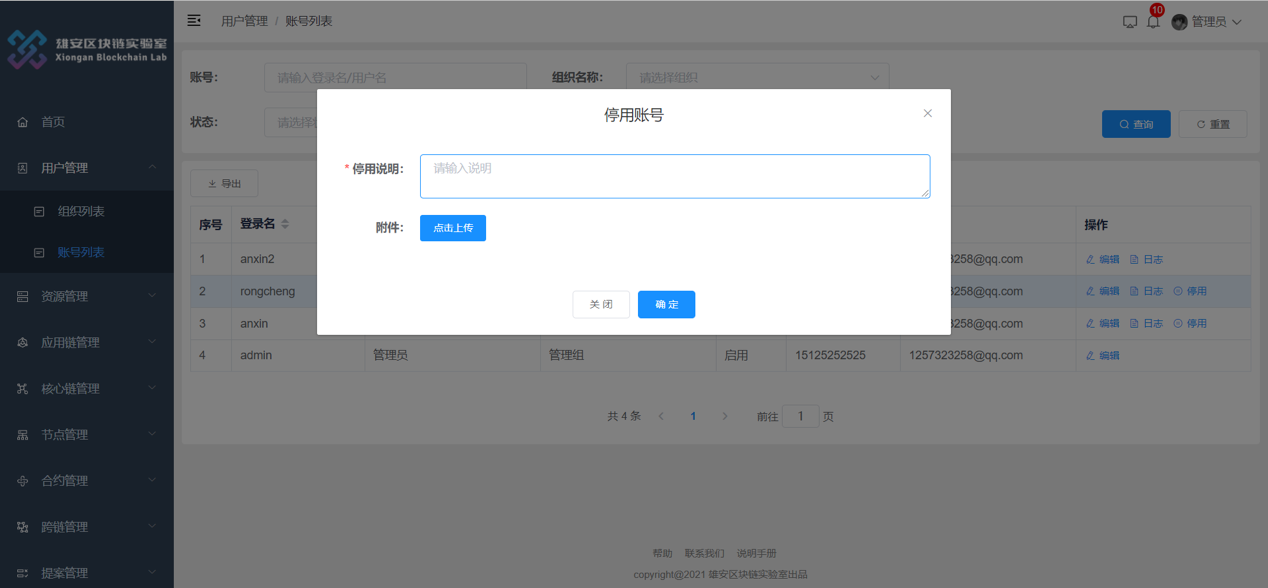
1. 点击操作列编辑，弹出账号基本信息窗口，编辑后点击保存，更新账号基本信息。



1. 点击操作列日志，弹出账号操作日志窗口，展示账号启停原因、操作人、操作时间、启停附件。其中有附件的，显示为蓝色超链接样式，点击可以下载附件；没有附件则显示为灰色无附件，不可以点击。



1. 点击停用，弹框录入停用说明，可以根据需求上传附件。



1. 已停用的账号，操作列显示启用按钮。点击启用，弹框录入启用说明，可以根据需求上传附件。



1. 点击列表上方的“导出”按钮，可以将表格中的数据导出为Excel文件。

### 资源管理

#### 资源列表

##### 功能简介

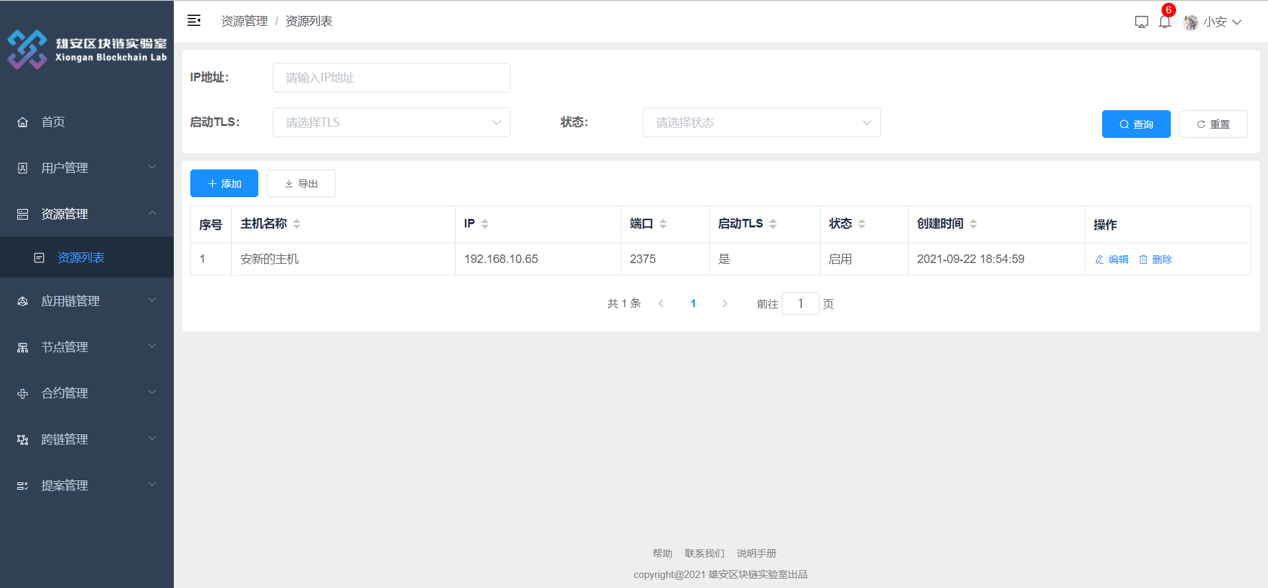
该功能展示了本组织下的主机资源信息，可以添加、编辑、删除主机信息。

##### 前置条件

组织管理员或核心链管理员已登入系统。

##### 操作说明

1. 输入查询条件，点击查询按钮，列表展示对应信息。点击重置按钮，清空查询条件，列表展示全部数据。



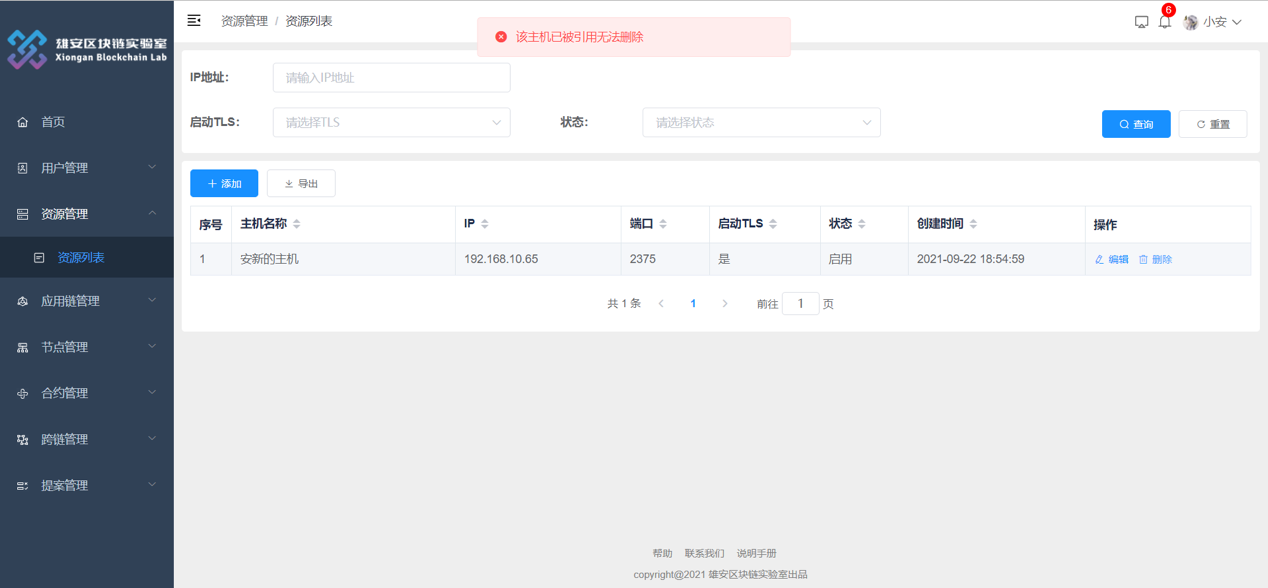
1. 添加主机，点击列表上方的添加按钮，弹出窗口，录入主机信息，点击确定添加主机资源。



1. 编辑主机信息，点击列表操作列编辑按钮，展示主机当前信息，输入要编辑的项目，点击确定，完成编辑操作。



1. 删除主机，点击列表操作列删除按钮，弹出确认框，是否确认删除对应主机信息。如果该主机正在使用中，则无法删除。



### 应用链管理

#### 应用链列表

##### 功能简介

该功能展示了当前组织所有已经加入的应用链的简略信息，点击列表操作列的浏览按钮，打开区块链浏览器的页面，展示当前链更加详细的信息，例如块高、出快速度、交易量、区块详细信息和交易详细信息等。

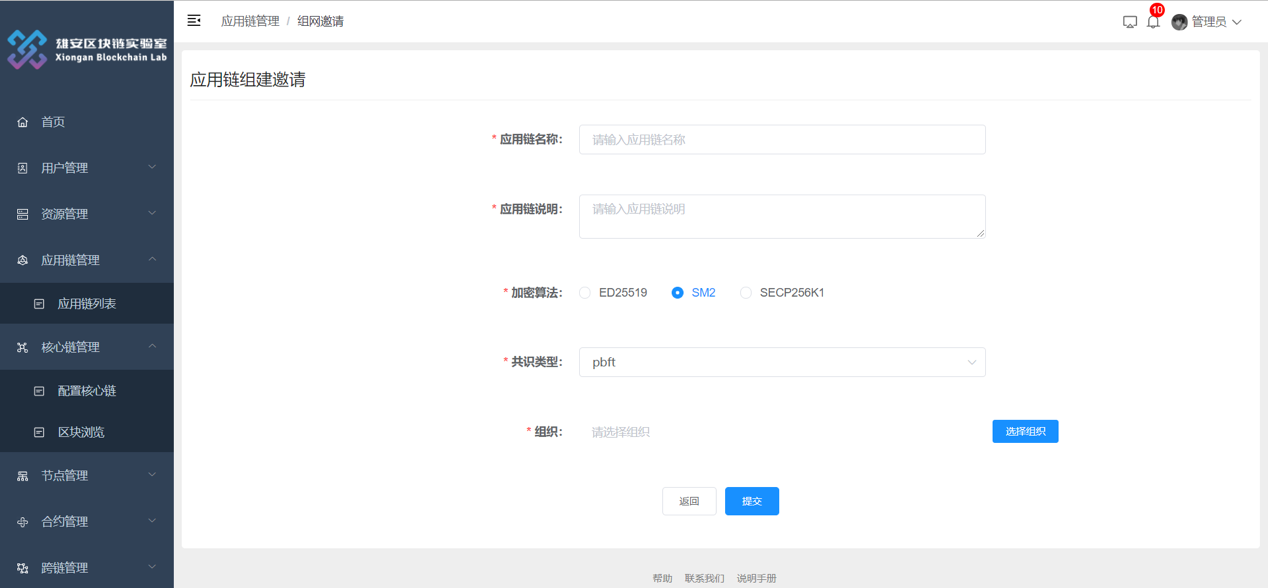
超级管理员登录系统时，列表上方展示组链邀请按钮，可以邀请组织组建应用链。

##### 前置条件

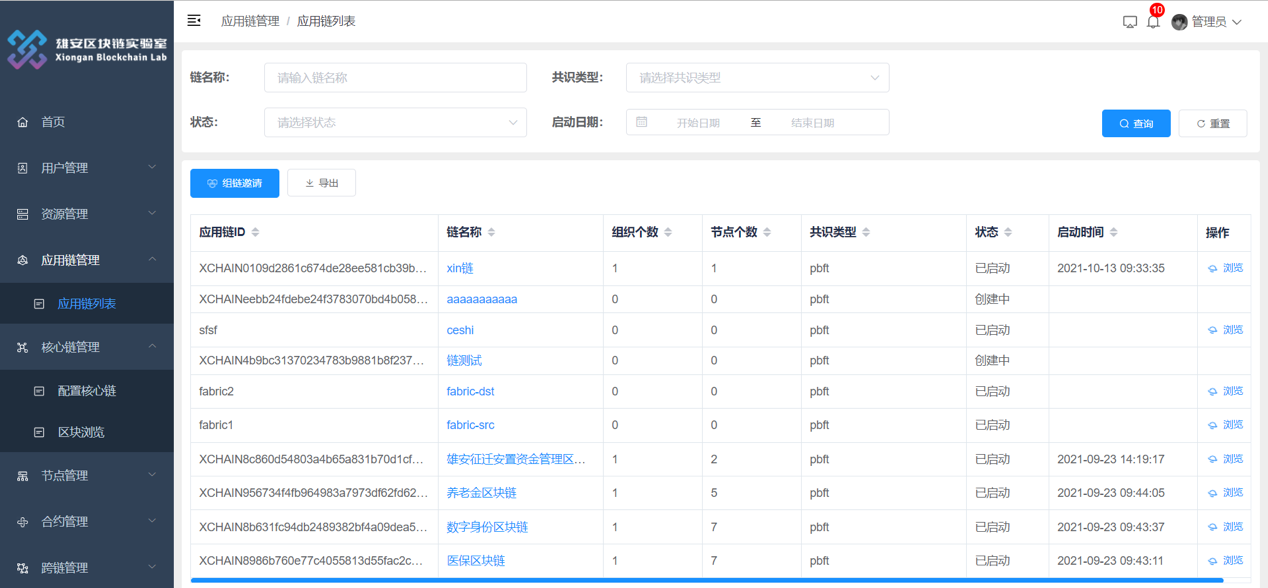
使用超级管理员登录系统或者组织管理员登录系统并且已经加入至少一个应用链。

##### 操作说明

1. 超级管理员登录系统，打开应用链列表页面，点击组链邀请，进入应用链组建邀请页面，录入应用链信息，选择初始组织，点击提交，则向已选择的组织发出组建邀请。



1. 组织管理员登录系统，打开应用链列表页面。默认展示当前加入的所有应用链。输入查询条件，点击查询，列表展示对应条件下的应用链信息。点击重置按钮，清空查询条件，刷新列表信息为初始状态。



1. 点击已经加入的应用链名称，展示该链的详细信息，包括链基本信息、链上组织、节点、合约和账号信息。



1. 点击操作列浏览按钮，进入区块链浏览器页面。其中左上角的节点图，当节点展示为蓝色时，表示节点健康运行，展示为红色时，表示该节点健康度不佳。在区块信息模块，点击区块哈希，展示该区块的详细信息。交易信息模块，点击交易哈希，展示该交易内的详细信息。



### 核心链管理

#### 配置核心链

##### 功能简介

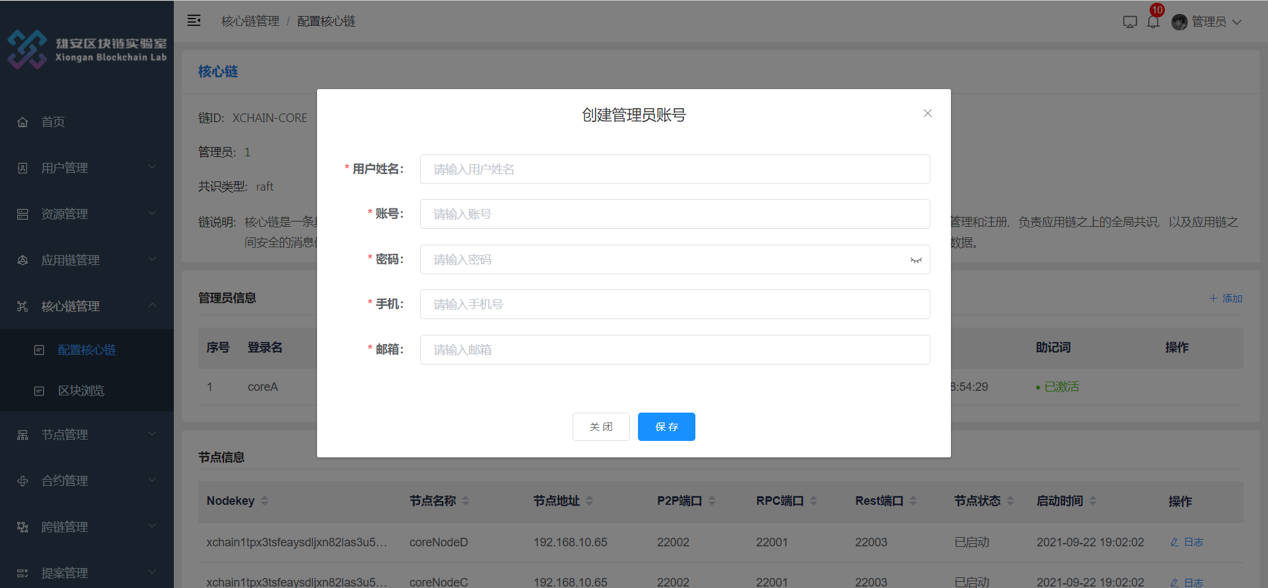
该功能展示了核心链的基本信息，也可以配置核心链的管理员信息、节点信息。

##### 前置条件

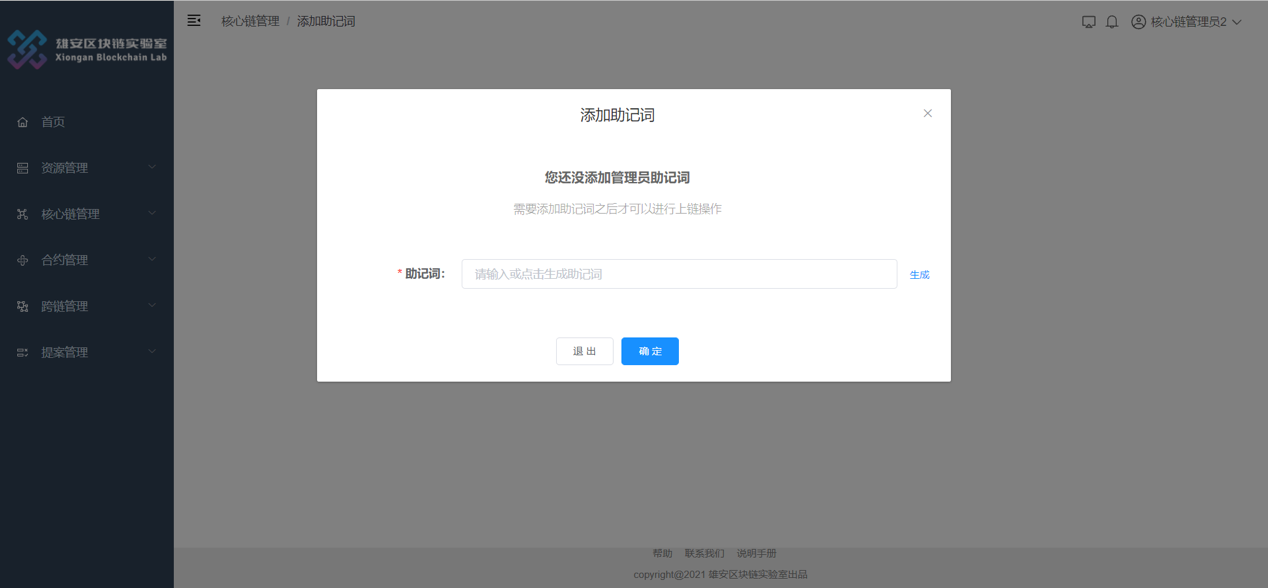
超级管理员登入系统。

##### 操作说明

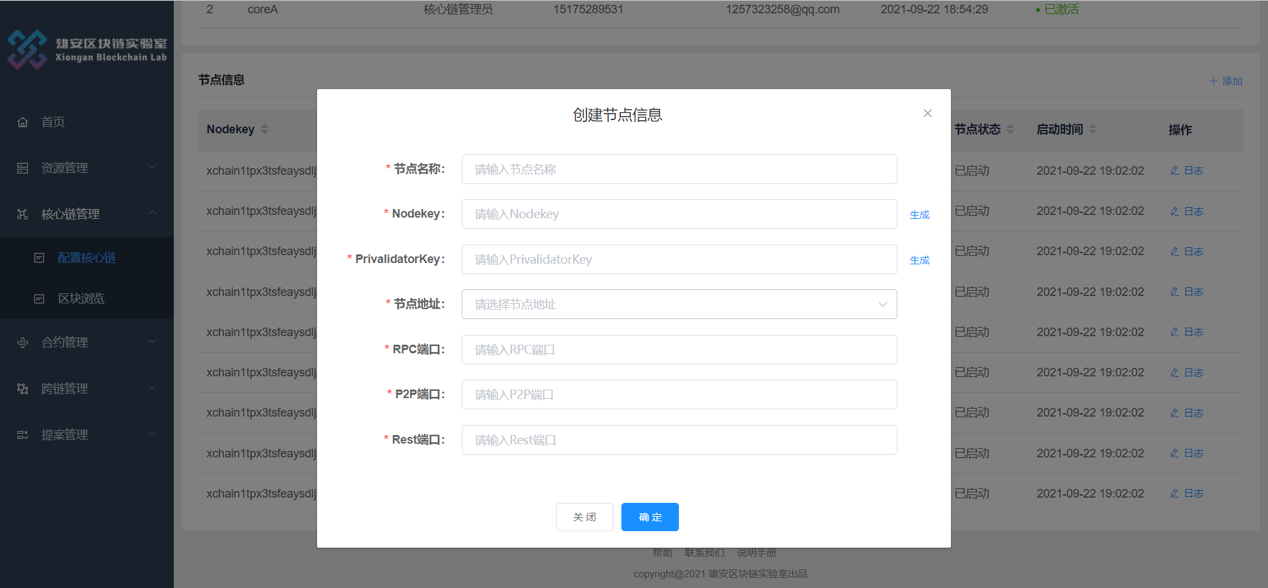
1. 打开配置核心链页面，核心链的基本信息展示初始化信息。管理员列表、节点信息列表默认为无数据。点击管理员列表右上角的添加按钮，添加管理员信息，录入信息保存后，管理员助记词默认为未激活状态。此时，核心链管理员登入系统后，首先要生成助记词，否则无法进行任何操作。







1. 添加核心链节点。使用核心链管理员登入系统，点击节点信息列表右上角的添加按钮，录入节点信息，添加核心链节点操作完成。



1. 点击节点操作列表中的日志按钮，展示该节点上运行日志。

#### 区块浏览

##### 功能简介

该功能是核心链的区块链浏览器，展示核心链的各项运行信息、交易趋势图、区块信息和交易信息。

##### 前置条件

核心链管理员登入系统。

##### 操作说明

1. 点击区块浏览菜单，进入区块链浏览器页面。其中左上角的节点图，当节点展示为蓝色时，表示节点健康运行，展示为红色时，表示该节点健康度不佳。在区块信息模块，点击区块哈希，展示该区块的详细信息。交易信息模块，点击交易哈希，展示该交易内的详细信息。



### 节点管理

#### 节点列表

##### 功能简介

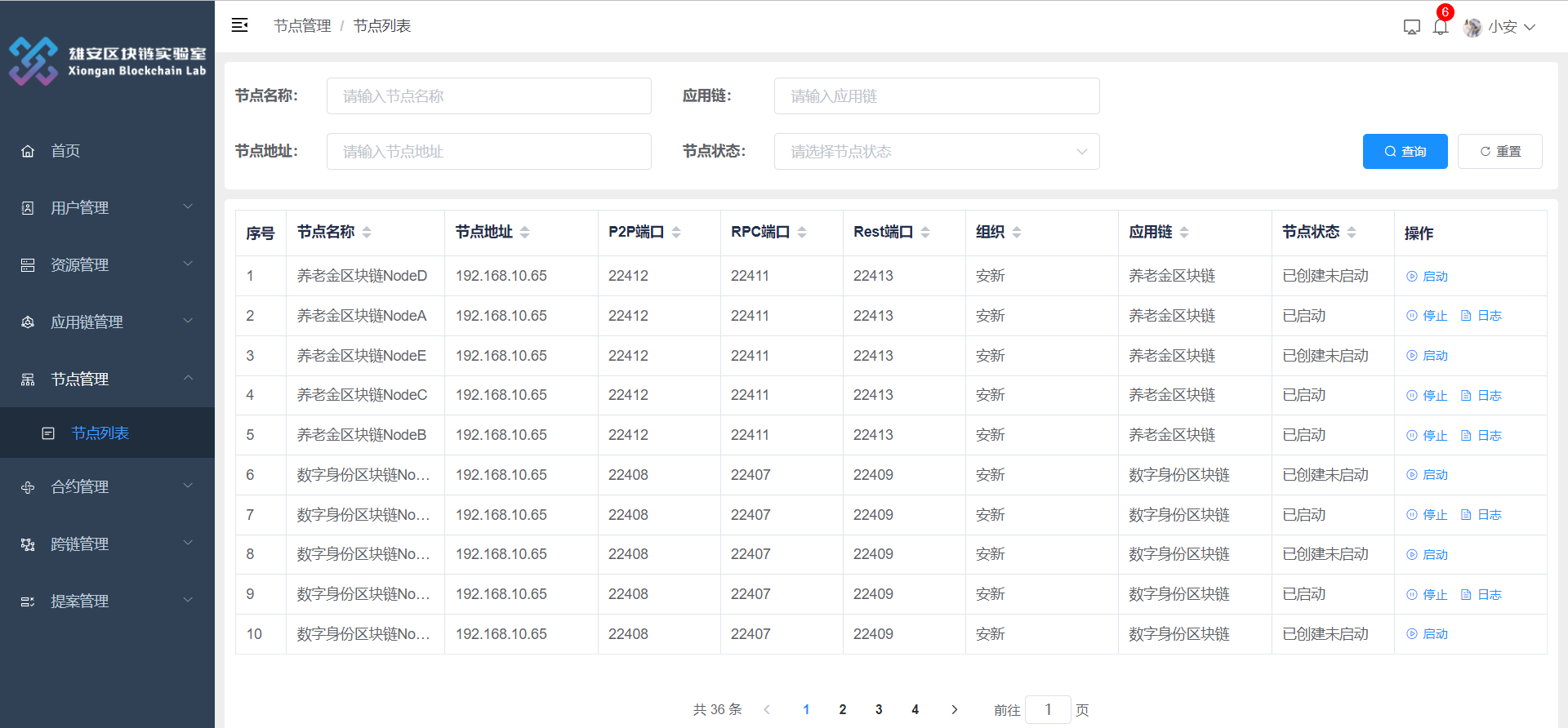
该功能展示了当前组织下所有的节点信息，可以通过操作列的启/停按钮操作节点的运行状态，点击日志按钮可以查看当前节点的运行日志信息。

##### 前置条件

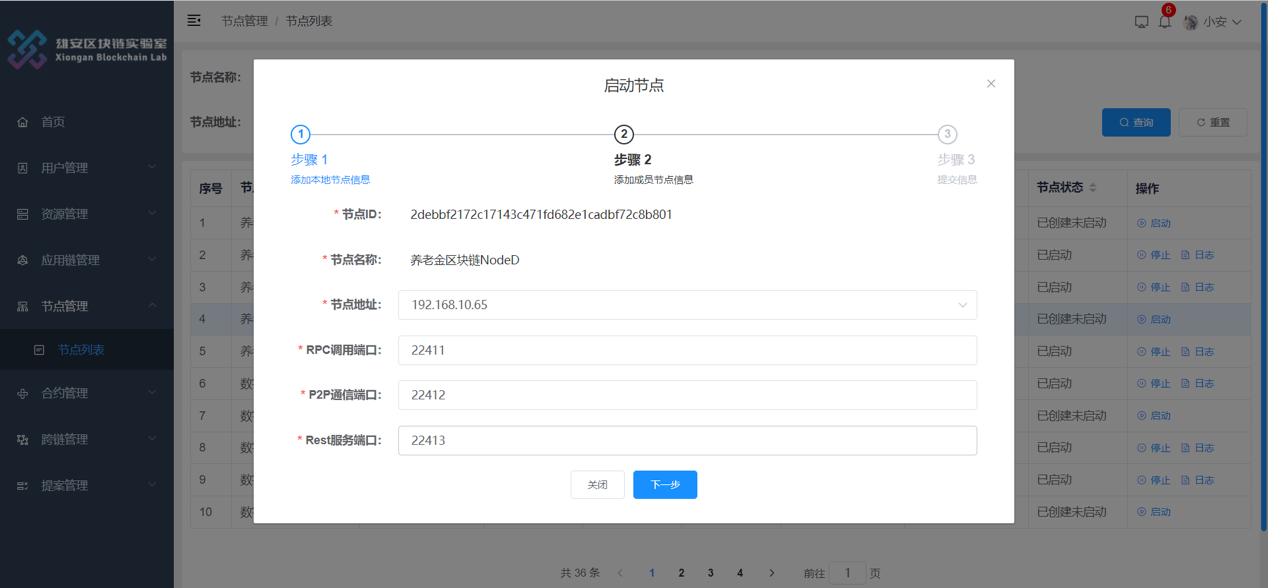
使用组织管理员账号登入系统，且至少拥有一个节点。

##### 操作说明

1. 页面默认以列表的形式展示全部节点信息，输入查询条件，点击查询，列表展示对应条件的节点信息。点击重置按钮，清空查询条件，展示全部节点列表信息。



1. 启动节点。点击启动按钮，分为三步操作，首先录入节点地址和端口信息，点击下一步，添加成员节点信息，点击下一步，确认前两步录入的信息，无误后点击启动。







1. 查看节点日志。点击操作列日志按钮，弹窗展示当前节点的运行日志。

### 合约管理

#### 合约列表

##### 功能简介

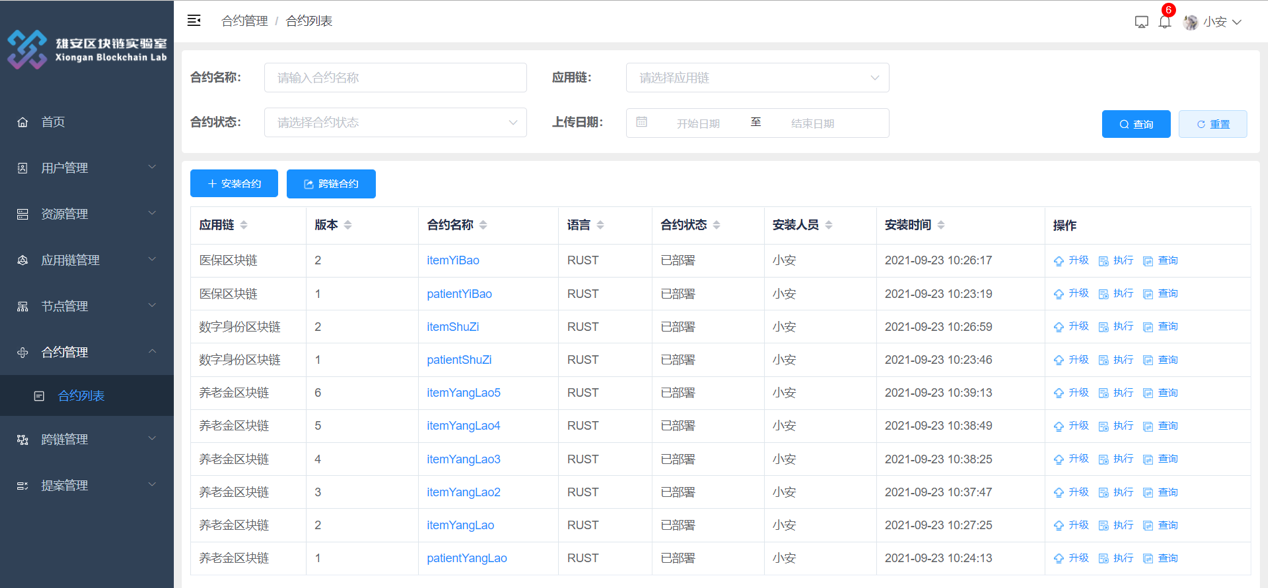
该功能展示了已加入应用链上的合约信息，同时可以查看合约历史版本信息，安装、升级、执行、查询合约。

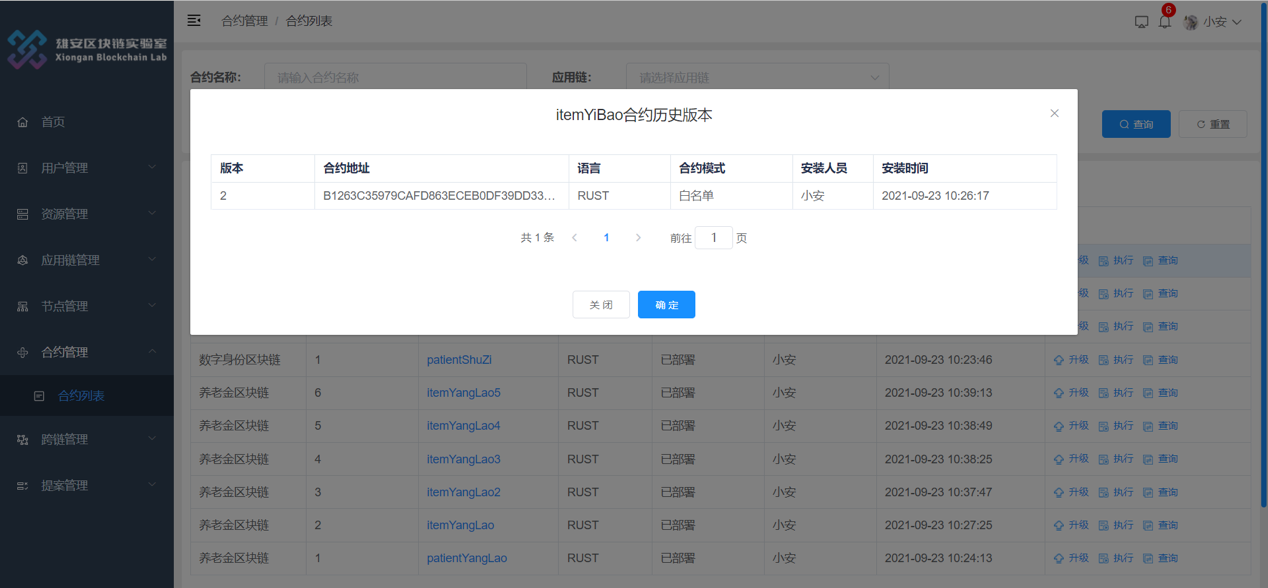
##### 前置条件

组织管理员登入系统，并且已经加入至少一个应用链。

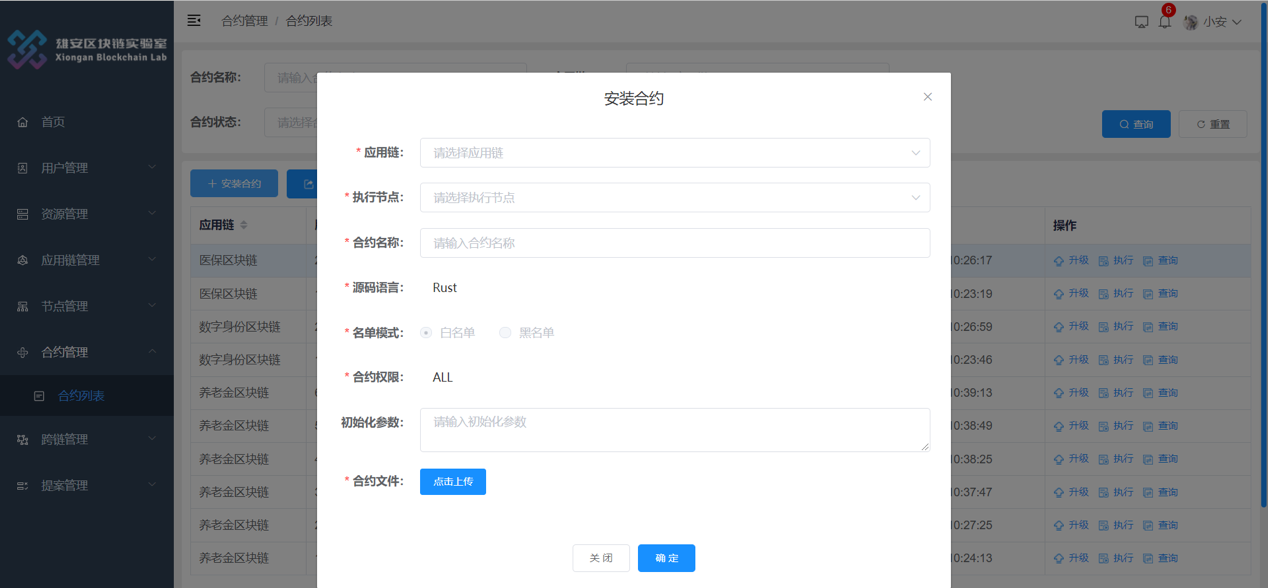
##### 操作说明

1. 合约信息列表。打开合约列表页面，默认展示所有已加入应用链上的合约，输入查询条件，点击查询，列表显示对应的合约信息。点击列表合约名称，可以查看合约历史版本信息。

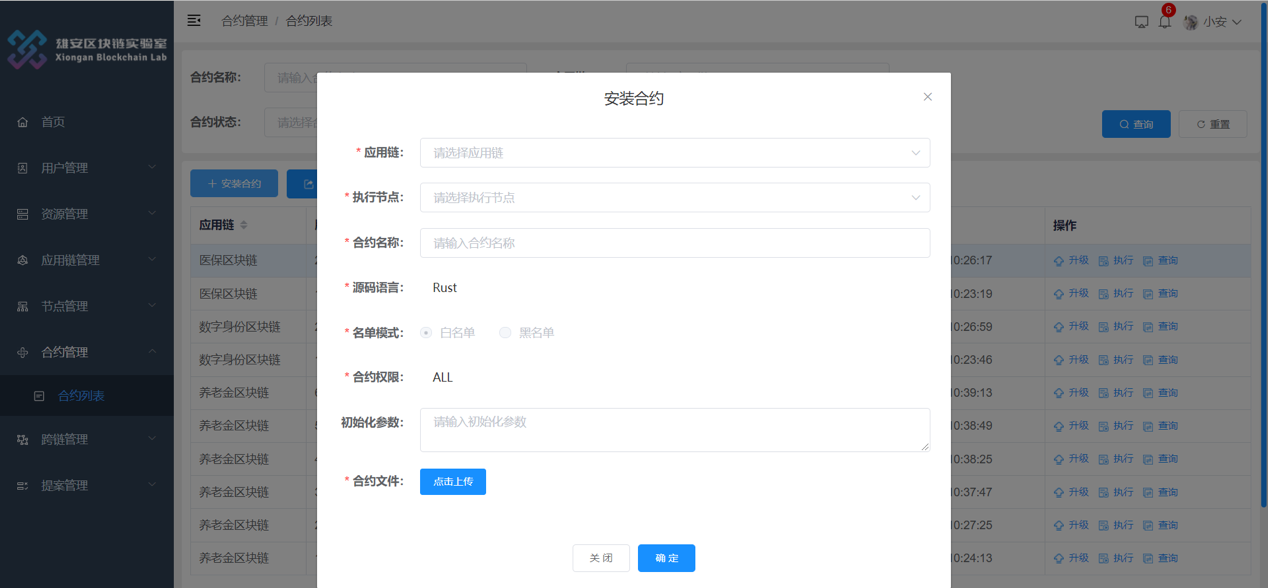




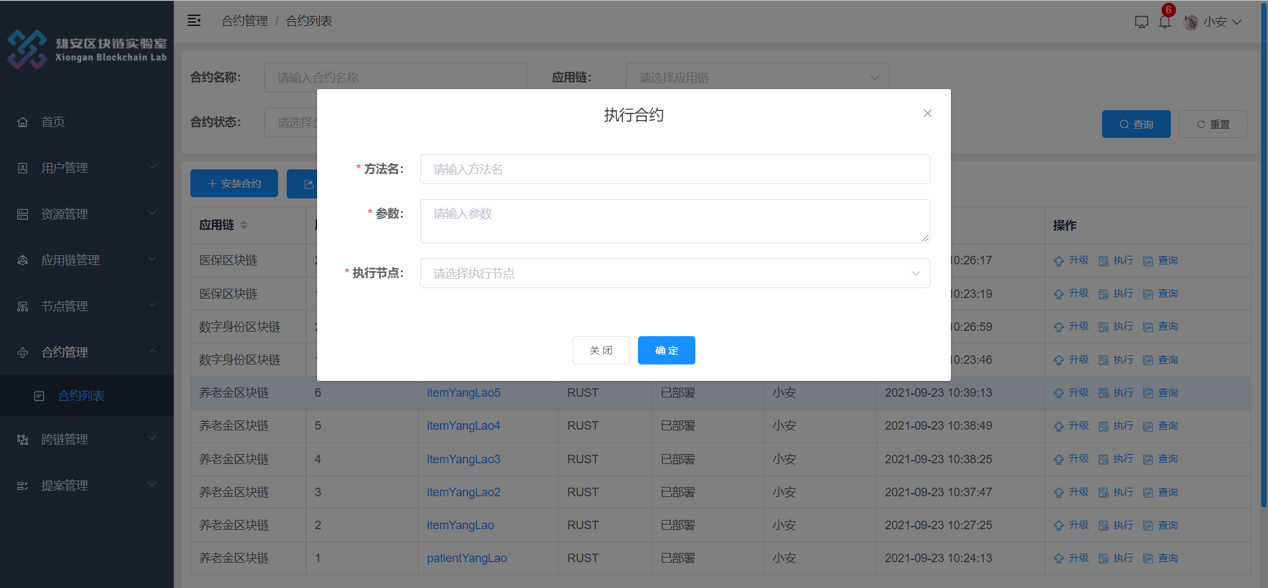
1. 安装合约。点击列表左上的安装合约按钮，打开弹窗，录入合约信息，上传合约文件，点击确定，完成安装合约的操作。



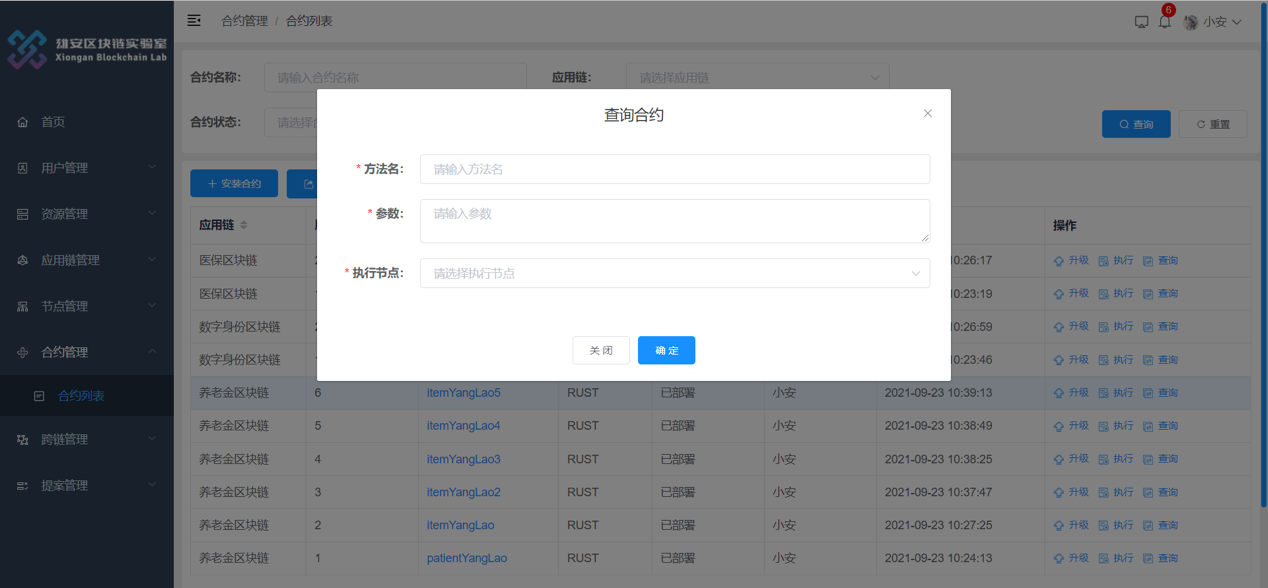
1. 升级合约。点击合约列表操作列的升级按钮，打开弹窗，默认展示该合约当前信息，修改可编辑内容，根据实际情况上传合约文件，点击确定，完成合约升级，同时生成一条该合约的历史版本信息。



1. 执行合约。合约安装成功之后，点击操作列执行按钮，打开弹窗，录入必填信息，点击确定，弹出执行结果通知。



1. 查询合约。查询合约的操作与执行合约类似，录入信息之后，点击确定，弹出查询结果通知。



1. 跨链合约。完成跨链注册之后，可以通过功能完成跨链合约操作。



### 跨链管理

#### 应用链列表

##### 功能简介

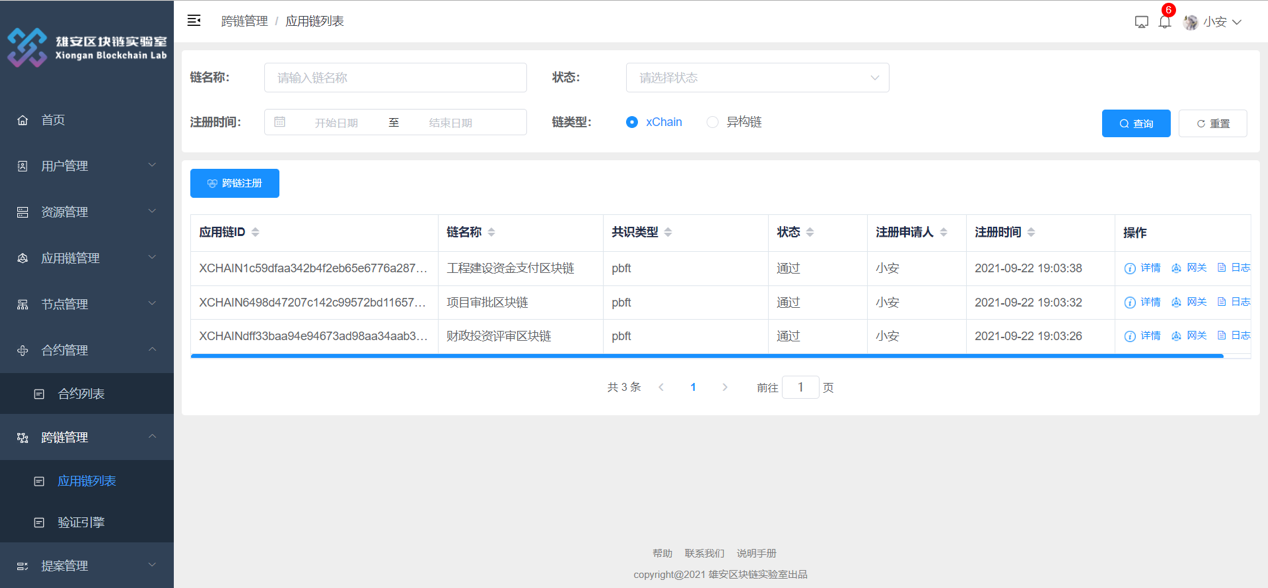
该功能主要是完成跨链注册和注册后应用链的信息展示，查看已注册的应用链的跨链事务、网关和网管日志信息。

##### 前置条件

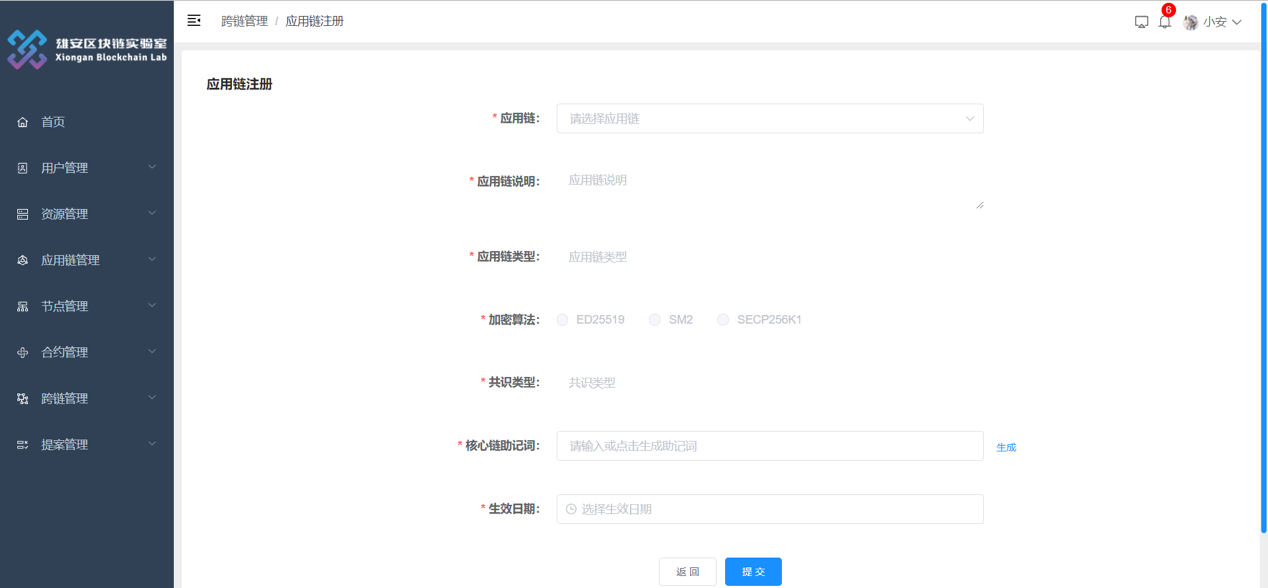
组织管理员登入系统。

##### 操作说明

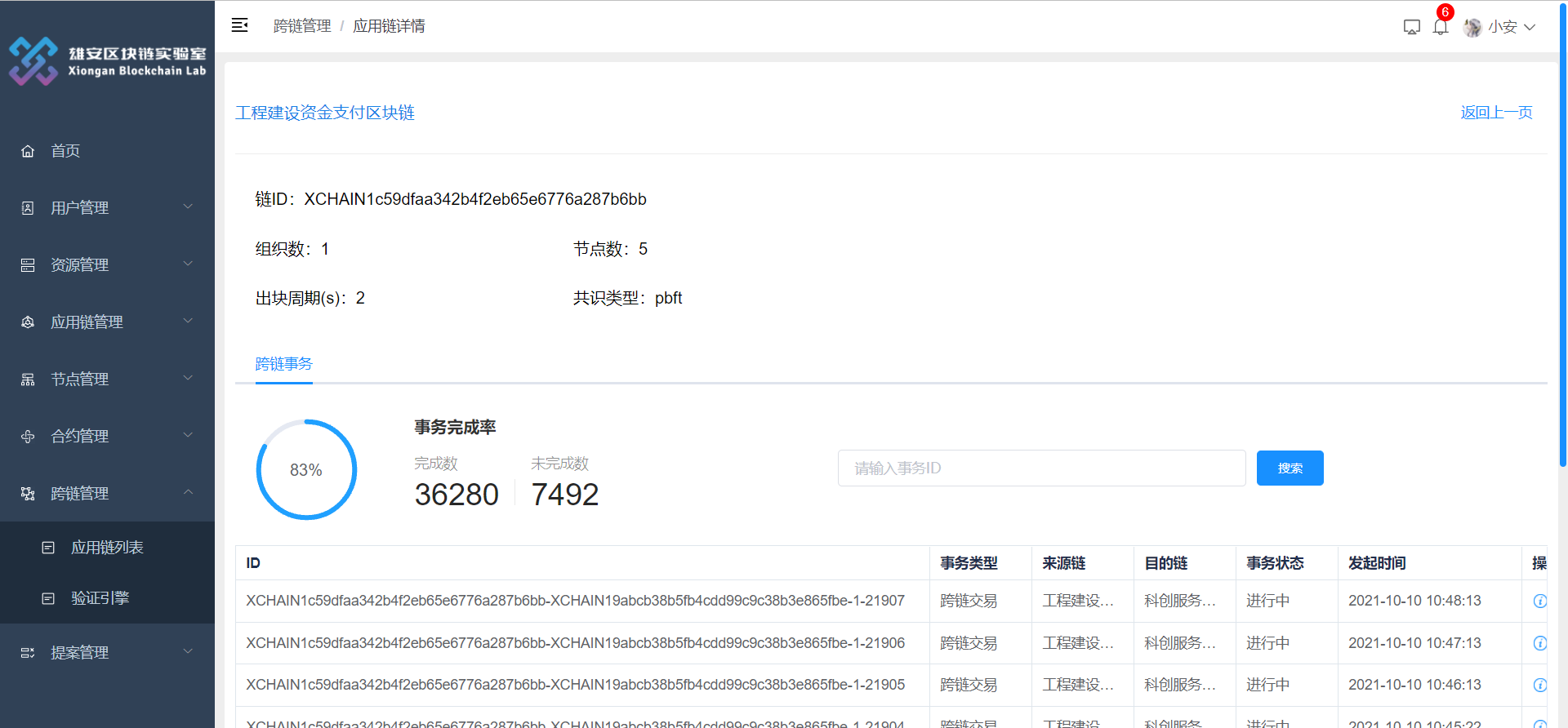
1. 页面默认展示全部已完成跨链注册的应用链信息。输入查询条件，点击查询按钮，列表显示对应的应用链信息。点击重置按钮，清空查询条件，列表展示全部已注册的应用链信息。



1. 跨链注册。点击列表左上方跨链注册按钮，打开应用链注册页面，选择要注册的应用链，生成助记词，选择生效日期，点击提交按钮，完成跨链注册操作。



1. 跨链事务信息。点击操作列详情按钮，跳转至该链详情页面，页面展示了跨链事务的统计信息和列表信息，点击列表详情可以查看该事务的详细信息。



1. 跨链网关。点击操作列网关按钮，弹窗展示网关基本信息。



1. 网管日志。点击操作列日志按钮，弹窗展示网关运行日志。

#### 验证引擎

##### 功能简介

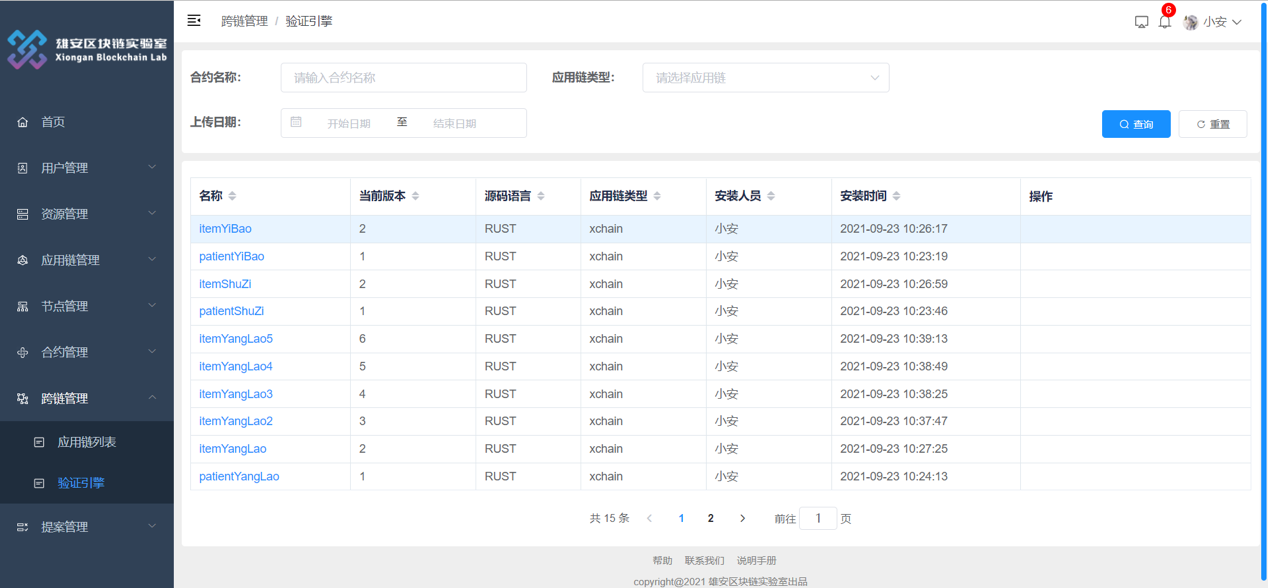
该功能展示了异构链跨链的验证引擎信息。

##### 前置条件

无

##### 操作说明

1. 输入查询条件，点击查询按钮，列表显示对应的合约信息。点击重置按钮，清空查询条件，列表展示全部合约信息。



### 提案管理

#### 提案列表

##### 功能简介

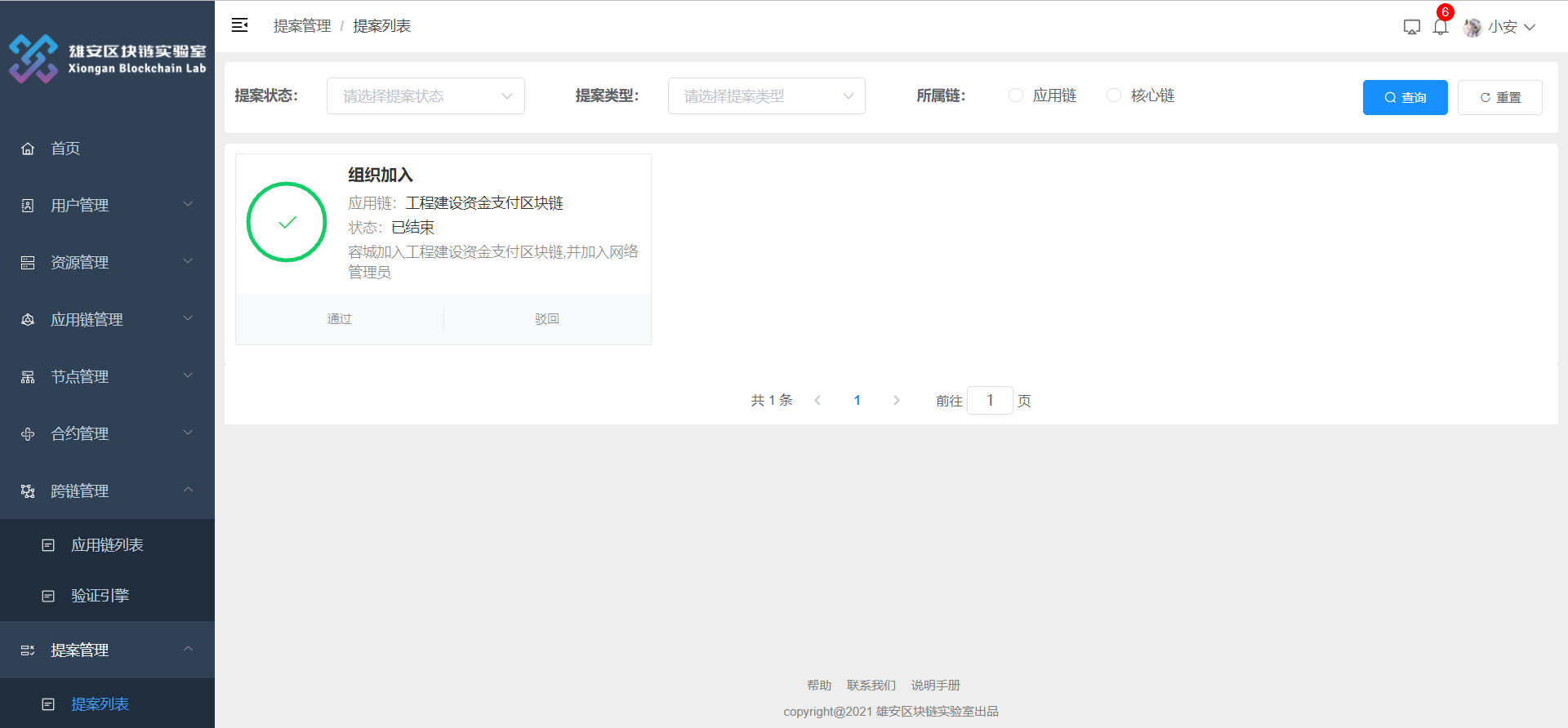
该功能收录了系统中全部提案信息，可以查提案状态，对提案进行投票。

##### 前置条件

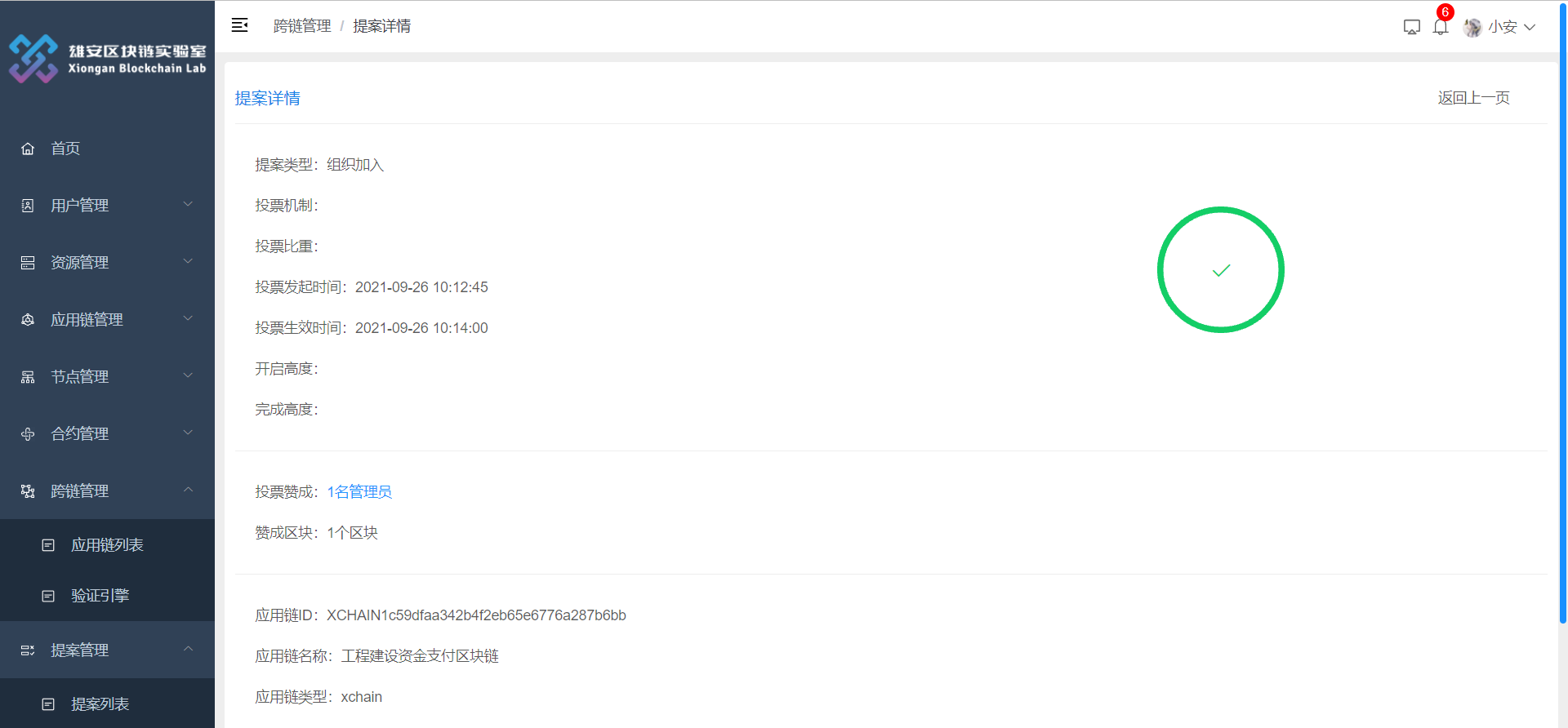
组织管理员登入系统。已经进行过需要生成提案并且投票的操作。例如组织加入应用链等。

##### 操作说明

1. 提案列表。页面默认展示已加入应用链的全部提案，最新生成的提案展示在列表最前。输入查询条件，点击查询，列表展示满足条件的提案信息。点击重置按钮，展示所有提案信息。



1. 提案投票。组织管理员可以对当前未投票的提案进行操作，点击通过或者驳回完成投票操作。
2. 提案详情。点击页面任意提案卡片，打开提案详情页面，展示提案的详细信息。



## 计费管理

### 功能简介

该功能收录了系统中的计费信息，可以查询消费充值记录。

### 操作说明

1. 应用链充值。管理员可以对应用链进行充值，页面显示当前余额，输入充值金额后点击确定进行充值。



1. 计费列表。页面默认展示当前用户已加入链的的全部计费记录，最新生成的记录展示在列表最前。输入查询条件，点击查询，列表展示满足条件的计费信息。点击重置按钮，展示所有计费信息。



## Restful接口命令

* 添加组织并设置组织管理员

URL /member/orgs

Method POST

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"orgId":"",

"orgAdminAddress":""

"parentOrgFullId":""

}

* 撤销组织

URL /member/orgs/{orgFullId}

Method DELETE

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

}

}

* 冻结组织

URL /member/orgs/{orgFullId}

Method PUT

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"type":1 freeze 2 unfreeze

}

* 组织解冻（同上）
* 修改组织管理员

URL /member/orgs/{orgFullId}/orgadmin

Method PUT

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"newAdminAddress":""

}

* 添加网络管理员

URL /member/nwadmins

Method POST

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"address":""

}

* 删除网络管理员

URL /member/nwadmins/{address}

Method DELETE

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

}

* 获取组织详情

URL /member/orgs/{orgFullId}

Method GET

* 获取组织详情列表

URL /member/orgs/

Method GET

* 创建组织自定义角色

URL /member/orgs/{orgFullId}/roles

Method POST

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"roleId":"",

"baseRoleId":"",

}

* 撤销角色

URL /member/orgs/{orgFullId}/roles/{roleId}

Method DELETE

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

}

}

* 查看角色详情

URL /member/roles/{roleFullId}

Method GET

获取角色详情列表

URL /member/roles/{roleFullId}

Method GET

* 添加账号

URL /member/accounts

Method POST

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"address":"",

"orgFullId":"",

"roleId":""

}

* 撤销账号

URL /member/accounts/{address}

Method DELETE

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

}

}

冻结账号\账号解除冻结\给账号添加或删除角色

URL /member/accounts/{address}

Method PUT

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"roleId":"",

"type":0 removerole 1 addrole 2 freeze 3 unfreeze

}

* 查看账号详情

URL /member/accounts/{address}

Method GET

* 获取账号详情列表

URL /member/accounts

Method GET

* 添加或修改权限

URL /member/permissions/{resource}

Method PUT

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"policy":"",

"expElements":[

{

"OrgExps":[

],

"RoleIds":[

]

}

]

}

* 获取权限列表

URL /member/permissions

Method GET

* 获取单个权限

URL /member/permissions/{resource}

Method GET

* 检查账号是否拥有权限

URL /member/checkpermission/{address}/{resource}

Method GET

* 投票

URL /gov/proposals/{proposalId}

Method PUT

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"pass":"true/false",

}

* 获取单个提案

URL /gov/proposals/{proposalId}

Method GET

* 获取所有提案

URL /gov/proposals

Method GET

* 创建Validator

URL /poa/validators

Method POST

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"pubkey":"",

"power":"",

"moniker":"",

"identity":"",

"website":"",

"security\_contact":"",

"details":""

}

* 修改Validator

URL /poa/validators/{validatorAddress}

Method PUT

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"power":""

}

* 查询指定高度的Validator

URL /poa/historical\_info/{height}

Method GET

* 查询PoA模块的参数

URL /poa/parameters

Method GET

获取指定Validator

URL /poa/validators/{validatorAddr}

Method GET

获取所有Validator

URL /poa/validators

Method GET

* 修改参数

URL /params/{subspace}/{key}

Method PUT

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"value":""

}

* 上传合约

URL /wasm/code

Method POST

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"wasm\_bytes":"",

"execute\_perm":"",

"language":"",

"label":"",

"policy":""

}

* 部署合约

URL /wasm/code/{codeId}

Method POST

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"admin":"",

"init\_msg":"",

"contract\_name":""

}

* 执行合约

URL /wasm/contract/{contractName}

Method POST

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"exec\_msg":""

}

* 升级合约

URL /wasm/contract/{contractName}/code

Method PUT

Content-type application/json

Param {

"base\_req":{

"from":"",

"chain\_id":"",

"password":""

},

"admin":"",

"code\_id":"",

"migrate\_msg":""

}

* 查询已上传合约

URL /wasm/code

Method GET

* 查询合约信息

URL /wasm/contract/{contractName}

Method GET

* 调用合约内部查询方法

URL /wasm/contract/{contractName}/smart/{query}

Method GET

Param {

"encoding":"base64"

}

## 出错与故障恢复

当有节点宕机或者其他原因停止运行，此时只需在相应服务器中执行lsof -i:26656命令找到对应的pid然后kill掉此进程：kill -9 pid，最后重新启动xcd服务：xcd start。