WeBASE Documentation

发布 ***v3.1.0***

WeBankBlockchain

2023 年 **07** 月 **27** 日

平台介绍

WeBASE（WeBank Blockchain Application Software Extension） 是在区块链应用和FISCO BCOS节点之间搭建的一套通用组件。

- 本技术文档适用于WeBASE lab版本（适配FISCO BCOS v3.0），WeBASE 1.x版技术文档可跳转至[WeBASE master分支] 查看

重要： FISCO-BCOS 2.0与3.0对比、JDK版本、WeBASE及其他子系统的版本兼容说明！请查看

CHAPTER 1

什么是WeBASE

WeBASE（WeBank Blockchain Application Software Extension） 是在区块链应用和FISCO-BCOS节点之间搭建的一套通用组件。围绕交易、合约、密钥管理，数据，可视化管理来设计各个模块，开发者可以根据业务所需，选择子系统进行部署。WeBASE屏蔽了区块链底层的复杂度，降低开发者的门槛，大幅提高区块链应用的开发效率，包含节点前置、节点管理、交易链路，数据导出，Web管理平台等子系统。

1.1 建立区块链应用开发标准

WeBASE将区块链应用开发标准化，按照部署、配置、开发智能合约、开发应用层、在线运维管理五个步骤即可完成一个区块链应用的开发，详细开发流程请参阅 使用WeBASE开发区块链应用

1.2 设计原则

按需部署 WeBASE抽象应用开发的诸多共性模块，形成各类服务组件，开发者根据需要部署所需组件。

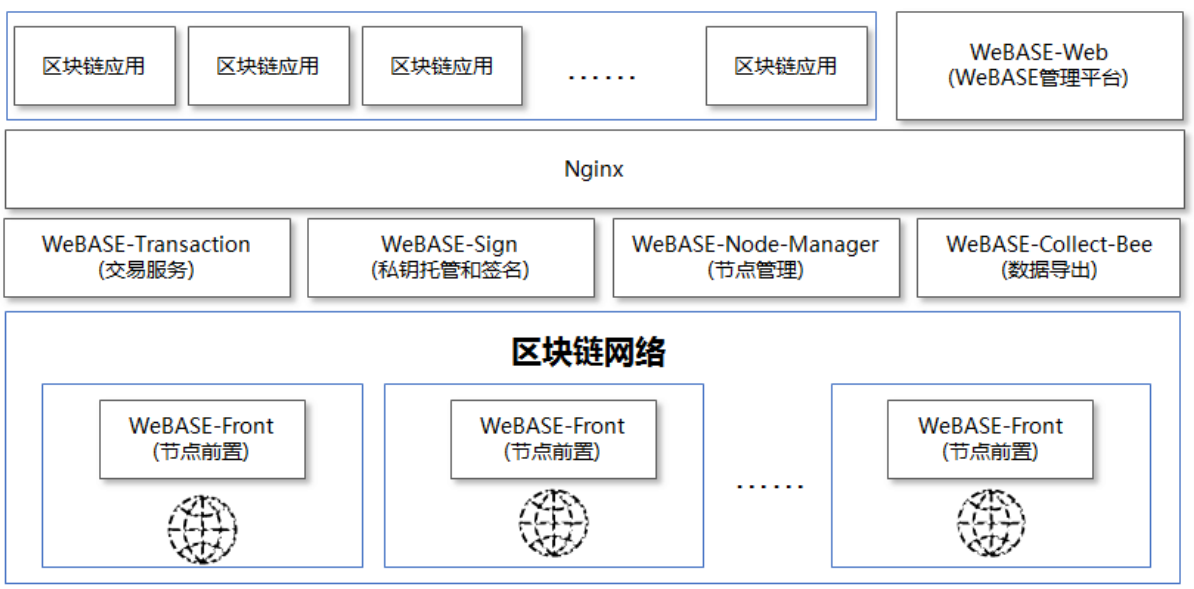
微服务 WeBASE采用微服务架构，基于Spring Boot框架，提供RESTful风格接口。

零耦合 WeBASE所有子系统独立存在，均可独立部署，独立提供服务。

可定制 前端页面往往带有自身的业务属性，因此WeBASE采用前后端分离的技术，便于开发者基于后端接口定制自己的前端页面。

1.3 整体架构

完整的部署架构如下，其中WeBASE-Front需要和区块链节点同机部署。



1.4 功能介绍

从可视化，智能合约，交易，数据四个维度设计各个中间件，各模块主要功能如下

1.5 各子系统简介

- **节点前置服务**

  集成FISCO BCOS JavaSDK，提供RESTful风格的接口，客户端可以使用http的形式和节点进行交互，内置内存数据库，采集节点健康度数据。内置web控制台，实现节点的可视化操作。

  Github地址

  Gitee地址

说明文档

- **节点管理服务**

  处理前端页面所有web请求，管理各个节点的状态，管理链上所有智能合约，对区块链的数据进行统计、分析，对异常交易的审计，私钥管理等。

  Github地址

  Gitee地址

  说明文档

- **WeBASE管理平台** 可视化操作平台，可基于此平台查看节点信息，开发智能合约等。

  Github地址

  Gitee地址

  说明文档

- **私钥托管和签名服务** 托管用户私钥，提供云端签名。

  Github地址

  Gitee地址

  说明文档

CHAPTER 2

版本及兼容

2.1 支持FISCO-BCOS 3.X.X系列版本

重要：FISCO-BCOS 2.0与3.0对比、JDK版本、WeBASE及其他子系统的版本兼容说明！请查看

2.1.1 WeBASE v3.1.0

WeBASE v3.1.0 版本支持 FISCO-BCOS 3.0.0及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS v3.4.0版本。

2.1.2 WeBASE v3.0.2

WeBASE v3.0.2 版本支持 FISCO-BCOS 3.0.0及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS v3.2.0版本。

2.1.3 WeBASE v3.0.1

WeBASE v3.0.1 版本支持 FISCO-BCOS 3.0.0及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 3.1.0版本。

2.1.4 WeBASE v3.0.0

WeBASE v3.0.0 版本支持 FISCO-BCOS 3.0.0及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 3.0.0版本。

2.1.5 WeBASE lab-rc2

WeBASE lab-rc2 版本支持 FISCO-BCOS 3.0.0-rc2及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 3.0.0-rc2版本。

2.1.6 WeBASE lab-rc1

WeBASE lab-rc1 版本支持 FISCO-BCOS 3.0.0-rc1版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 3.0.0-rc1版本。

2.2 支持FISCO-BCOS 2.X.X系列版本

2.2.1 WeBASE v1.5.3

WeBASE v1.5.3 版本支持 FISCO-BCOS 2.5.x及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 2.8.0版本。

2.2.2 WeBASE v1.5.2

WeBASE v1.5.2 版本支持 FISCO-BCOS 2.5.x及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 2.7.2版本。

2.2.3 WeBASE v1.5.1

WeBASE v1.5.1 版本支持 FISCO-BCOS 2.5.x及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 2.7.2版本。

2.2.4 WeBASE v1.5.0

WeBASE v1.5.0 版本支持 FISCO-BCOS 2.5.x及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 2.7.2版本。

2.2.5 WeBASE v1.4.3

WeBASE v1.4.3 版本支持 FISCO-BCOS 2.5.x及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 2.7.1版本。

2.2.6 WeBASE v1.4.2

WeBASE v1.4.2 版本支持 FISCO-BCOS 2.4.x及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 2.7.0版本。

2.2.7 WeBASE v1.4.1

WeBASE v1.4.1 版本支持 FISCO-BCOS 2.5.x及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 2.6.0版本。

2.2.8 WeBASE v1.4.0

WeBASE v1.4.0 版本支持 FISCO-BCOS 2.4.x, 2.5.x版本，推荐使用2.4.x版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用 FISCO BCOS 2.4.1+版本。

2.2.9 WeBASE v1.3.2

WeBASE v1.3.2版本支持FISCO-BCOS 2.4.x版本，暂未支持2.5.x版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用FISCO BCOS 2.4.1+版本。

2.2.10 WeBASE v1.3.1

WeBASE v1.3.1版本支持FISCO-BCOS 2.4.x版本，暂未支持2.5.x版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用FISCO BCOS 2.4.0+版本。

2.2.11 WeBASE v1.3.0

WeBASE v1.3.0版本支持FISCO-BCOS 2.0.0 - 2.4.x版本，暂未支持2.5.x版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用FISCO BCOS 2.3.0+版本。

2.2.12 WeBASE v1.2.4

WeBASE v1.2.4版本支持FISCO-BCOS 2.0.0-rc1，FISCO-BCOS 2.0.0-rc2，FISCO-BCOS 2.0.0-rc3，FISCO-BCOS 2.0.0及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用FISCO BCOS 2.3.0版本。

2.2.13 WeBASE v1.2.3

WeBASE v1.2.3版本支持FISCO-BCOS 2.0.0-rc1，FISCO-BCOS 2.0.0-rc2，FISCO-BCOS 2.0.0-rc3，FISCO-BCOS 2.0.0及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用FISCO BCOS 2.2.0版本。

2.2.14 WeBASE v1.2.2

WeBASE v1.2.2版本支持FISCO-BCOS 2.0.0-rc1，FISCO-BCOS 2.0.0-rc2，FISCO-BCOS 2.0.0-rc3，FISCO-BCOS 2.0.0及以上版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用FISCO BCOS 2.2.0版本。

2.2.15 WeBASE v1.2.1

WeBASE v1.2.1版本支持FISCO-BCOS 2.0.0-rc1，FISCO-BCOS 2.0.0-rc2，FISCO-BCOS 2.0.0-rc3，FISCO-BCOS 2.0.0，FISCO-BCOS v2.1.0。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用FISCO BCOS 2.1.0版本。

2.2.16 WeBASE v1.2.0

WeBASE v1.2.0版本支持FISCO-BCOS 2.0.0-rc1，FISCO-BCOS 2.0.0-rc2，FISCO-BCOS 2.0.0-rc3，FISCO-BCOS 2.0.0，FISCO-BCOS v2.1.0。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用FISCO BCOS 2.1.0版本。

2.2.17 WeBASE V1.1.0

WeBASE V1.1.0版本支持FISCO-BCOS 2.0.0-rc1，FISCO-BCOS 2.0.0-rc2，FISCO-BCOS 2.0.0-rc3，FISCO-BCOS 2.0.0，FISCO-BCOS v2.1.0。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用FISCO BCOS 2.1.0版本。

2.2.18 WeBASE V1.0.0

支持FISCO-BCOS 2.0.0-rc1，FISCO-BCOS 2.0.0-rc2，FISCO-BCOS 2.0.0-rc3。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS 推荐使用FISCO BCOS 2.0.0-rc3版本。

2.3 支持FISCO-BCOS 1.3.X系列版本

WeBASE V0.5.X版本支持FISCO-BCOS 1.3.X系列版本。WeBASE 子系统推荐使用下表的版本搭配，FISCO-BCOS推荐使用FISCO BCOS 1.3.8 Release。

服务搭建详见Release tag代码里的README.md

CHAPTER 3

快速入门搭建

在区块链应用开发阶段建议用户使用快速入门搭建。在快速入门搭建模式，开发者只需要搭建节点和节点前置服务(WeBASE-Front)，就可以通过WeBASE-Front的合约编辑器进行合约的编辑，编译，部署，调试。

重要：FISCO-BCOS 2.0与3.0对比、JDK版本、WeBASE及其他子系统的版本兼容说明！请查看

3.1 节点搭建

节点搭建的方法建议使用build_chain。

3.1.1 Liquid支持

如果使用的`liquid`合约的链并在WeBASE-Front的合约IDE中编译Liquid合约，要求**手动**在WeBASE-Front所在主机配置Liquid环境

配置好Liquid环境后，需要重启WeBASE-Front

3.2 节点前置服务搭建

前提条件

备注：部署出现问题请查看问题记录。

1. 下载安装包

```
wget https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/releases/
↪download/v3.1.0/webase-front.zip
```

2. 解压

```
unzip webase-front.zip && cd webase-front
```

3. 拷贝sdk证书文件（build_chain的时候生成的）

   将节点所在目录nodes/${ip}/sdk下的所有文件拷贝到当前conf目录，供SDK与节点建立连接时使用

   - 拷贝命令可使用cp -r nodes/${ip}/sdk/* ./conf/
   - 证书为以下两种之一：

     非国密：ca.crt、sdk.crt、sdk.key

     国密：sm_ca.crt、sm_sdk.crt、sm_sdk.key、sm_ensdk.crt、sm_ensdk.key

4. 修改配置

```
vi conf/application.yml
```

```
sdk:
  useSmSsl: false  # sdk连接节点是否使用国密ssl
  peers: ['127.0.0.1:20200','127.0.0.1:20201'] # 节点ip和rpc端口
```

5. 服务启停

   返回根目录，服务启停命令：

```
启动：  bash start.sh
停止：  bash stop.sh
检查：  bash status.sh
```

启动成功将出现如下日志：

```
...
        Application() - main run success...
```

3.3 状态检查

成功部署后，可以根据以下步骤**确认各个子服务是否启动成功**

3.3.1 1. 检查各子系统进程

通过ps命令，检查节点与节点前置的进程是否存在

- 包含：节点进程nodeXX，节点前置进程webase.front

检查方法如下，若无输出，则代表进程未启动，需要到webase-front/log中查看日志的错误信息，并根据错误提示或根据WeBASE-Front常见问题进行错误排查

检查节点进程

```
$ ps -ef | grep node
```

输出如下，此处部署了两个节点node0, node1

```
root     29977     1  1 17:24 pts/2    00:02:20 /root/fisco/webase/webase-deploy/
↪nodes/127.0.0.1/node1/../fisco-bcos -c config.ini
root     29979     1  1 17:24 pts/2    00:02:23 /root/fisco/webase/webase-deploy/
↪nodes/127.0.0.1/node0/../fisco-bcos -c config.ini
```

检查节点前置webase-front的进程

```
$ ps -ef | grep webase.front
```

输出如下

```
root      31805      1  0 17:24 pts/2    00:01:30 /usr/local/jdk/bin/java -Djdk.tls.
↪namedGroups=secp256k1 ... conf/:apps/*:lib/* com.webank.webase.front.Application
```

3.3.2 2. 检查进程端口

通过netstat命令，检查节点与节点前置的端口监听情况

检查方法如下，若无输出，则代表进程端口监听异常，需要到webase-front/log中查看日志的错误信息，并根据错误提示或根据WeBASE-Front常见问题进行错误排查

检查节点channel端口(默认为20200)是否已监听

```
$ netstat -anlp | grep 20200
```

输出如下

```
tcp        0      0 0.0.0.0:20200           0.0.0.0:*               LISTEN      ␣
↪29069/fisco-bcos
```

检查webase-front端口(默认为5002)是否已监听

```
$ netstat -anlp | grep 5002
```

输出如下

```
tcp6       0      0 :::5002                 :::*                    LISTEN      ␣
↪2909/java
```

3.3.3 3. 检查服务日志

日志中若出现报错信息，可根据信息提示判断服务是否异常，也可以参考并根据错误提示或根据WeBASE-Front常见问题进行错误排查

- 如果节点进程**已启用**且端口**已监听**，可跳过本章节
- 如果节点前置异常，如检查不到进程或端口监听，则需要webase-front/log中查看日志的错误信息
- 如果检查步骤出现检查不到进程或端口监听等异常，或者前置服务无法访问，可以按以下顺序逐步检查日志:
  - 检查webase-front/log中查看节点前置日志的错误信息，如果无错误，且日志最后出现application run success字样则代表运行成功
  - 检查nodes/127.0.0.1/nodeXXX/log中的节点日志

查看运行成功日志：webase-front运行成功后会打印日志main run success，可以通过搜索此关键字来确认服务正常运行。

如，检查webase-front日志，其他webase服务可进行类似操作

```
$ cd webase-front
$ grep -B 3 "main run success" log/WeBASE-Front.log
```

输出如下:

```
2020-12-09 15:47:25.355 [main] INFO  ScheduledAnnotationBeanPostProcessor() - No␣
↪TaskScheduler/ScheduledExecutorService bean found for scheduled processing
2020-12-09 15:47:25.378 [main] INFO  TomcatEmbeddedServletContainer() - Tomcat␣
↪started on port(s): 5002 (http)
```

(下页继续)

(续上页)

```
2020-12-09 15:47:25.383 [main] INFO  Application() - Started Application in 6.983␣
↪seconds (JVM running for 7.768)
2020-12-09 15:47:25.383 [main] INFO  Application() - main run success...
```

启动失败或无法使用时，欢迎到WeBASE-Front提交Issue或到技术社区共同探讨

- 提交Issue或讨论问题时，可以在issue中配上自己的**环境配置**，**操作步骤**，**错误现象**，**错误日志**等信息，方便社区用户快速定位问题

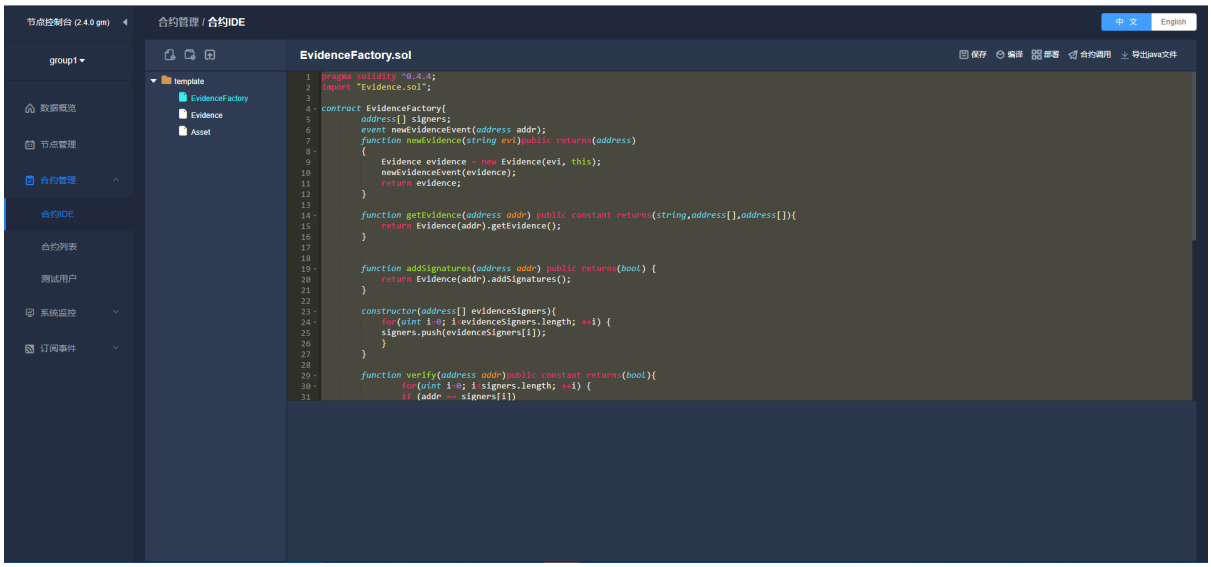
3.4 访问

访问 http://{deployIP}:{frontPort}/WeBASE-Front，示例：

```
```
http://localhost:5002/WeBASE-Front
```
```

注：若服务启动后无异常，但仍然无法访问，可以检查服务器的网络安全策略：

- **开放节点前置端口**：如果希望通过浏览器(Chrome Safari或Firefox)直接访问webase-front节点前置的页面，则需要开放节点前置端口`frontPort`（默认5002）



CHAPTER 4

一键部署

重要： FISCO-BCOS 2.0与3.0对比、JDK版本、WeBASE及其他子系统的版本兼容说明！请查看

一键部署可以在 **同机** 快速搭建WeBASE管理台环境，方便用户快速体验WeBASE管理平台。

一键部署会搭建：节点（FISCO-BCOS 3.0+）、管理平台（WeBASE-Web）、节点管理子系统（WeBASE-Node-Manager）、节点前置子系统（WeBASE-Front）、签名服务（WeBASE-Sign）。其中，节点的搭建是可选的，可以通过配置来选择使用已有链或者搭建新链。一键部署架构如下：

4.1 前提条件

4.1.1 检查环境

平台要求

推荐使用CentOS 7.2+, Ubuntu 16.04及以上版本, 一键部署脚本将自动安装`openssl, curl, wget, git, nginx, dos2unix`相关依赖项。

其余系统可能导致安装依赖失败，可自行安装`openssl, curl, wget, git, nginx, dos2unix`依赖项后重试

检查Java

推荐JDK8-JDK13版本，使用OracleJDK安装指引：

```
java -version
```

注意：不要用*sudo*执行安装脚本

检查mysql

MySQL-5.6或以上版本：

```
mysql --version
```

- Mysql安装部署可参考数据库部署

检查Python

使用Python3.6或以上版本：

```
python --version
# python3时
python3 --version
```

如已安装python3，也可通过`python3 --version`查看，在运行脚本时，使用`python3`命令即可

- Python3安装部署可参考*Python*部署

PyMySQL部署（Python3.6+）

Python3.6及以上版本，需安装`PyMySQL`依赖包

- CentOS

```
sudo yum -y install python36-pip
sudo pip3 install PyMySQL
```

- Ubuntu

```
sudo apt-get install -y python3-pip
sudo pip3 install PyMySQL
```

CentOS或Ubuntu不支持pip命令的话，可以使用以下方式：

```
git clone https://github.com/PyMySQL/PyMySQL
cd PyMySQL/
python3 setup.py install
```

4.1.2 检查服务器网络策略

网络策略检查：

- **开放WeBASE管理平台端口**：检查webase-web管理平台页面的端口`webPort`(默认为5000)在服务器的网络安全组中是否设置为**开放**。如，云服务厂商如腾讯云，查看安全组设置，为webase-web开放5000端口。**若端口未开放，将导致浏览器无法访问WeBASE服务页面**
- 开放节点前置端口：如果希望通过浏览器直接访问webase-front节点前置的页面，则需要开放节点前置端口`frontPort`（默认5002）；由于节点前置直连节点，**不建议对公网开放节点前置端口**，建议按需开放

4.2 拉取部署脚本

获取部署安装包：

```
wget https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/releases/download/
↪v3.1.0/webase-deploy.zip
```

解压安装包：

```
unzip webase-deploy.zip
```

进入目录：

```
cd webase-deploy
```

4.3 修改配置

① mysql数据库需提前安装，已安装直接配置即可，还未安装请参看数据库部署；

② 修改配置文件（vi common.properties）；

③ 一键部署支持使用已有链或者搭建新链。通过参数if.exist.fisco配置是否使用已有链，以下配置二选一即可：

- 当配置yes时，需配置已有链的sdk证书路径sdk.dir。路径下存放sdk证书，包含以下两种之一：

  非国密：ca.crt、sdk.crt、sdk.key

  国密：sm_ca.crt、sm_sdk.crt、sm_sdk.key、sm_ensdk.crt、sm_ensdk.key

- 当配置no时，需配置节点fisco版本和节点安装个数，未修改时搭建的新链默认两个群组

如果不使用一键部署搭建新链，可以参考FISCO BCOS官方文档搭建 FISCO BCOS部署流程；

- 如果使用的liquid合约的链（fisco.wasm=1），并在WeBASE管理台或WeBASE-Front的合约IDE中编译Liquid合约，要求**手动**在WeBASE-Front所在主机配置Liquid环境后，才能通过WeBASE编译Liquid合约
- 启用链的权限管理功能时（fisco.auth=1），建链后会在webase-deploy/nodes/ca目录生成一个链管理员私钥（包含国密与非国密），需要在WeBASE管理台的权限管理页面中**导入该私钥**才能进行权限管理。

④ 服务端口不能小于1024

```
[common]
# WeBASE子系统的最新版本（v3.0版本）
webase.web.version=v3.1.0
webase.mgr.version=v3.1.0
webase.sign.version=v3.0.2
webase.front.version=v3.1.0

# 节点管理子系统mysql数据库配置
mysql.ip=localhost
mysql.port=3306
mysql.user=dbUsername
mysql.password=dbPassword
mysql.database=webasenodemanager

# 签名服务子系统mysql数据库配置
sign.mysql.ip=localhost
sign.mysql.port=3306
sign.mysql.user=dbUsername
sign.mysql.password=dbPassword
sign.mysql.database=webasesign

# 节点前置子系统h2数据库名
front.h2.name=webasefront

# WeBASE管理平台服务端口
web.port=5000
# 节点管理子系统服务端口
```

(下页继续)

(续上页)

```
mgr.port=5001
# 节点前置子系统端口
front.port=5002
# 签名服务子系统端口
sign.port=5004

# SDK连接加密类型 （0: ECDSA SSL, 1: 国密SSL）
encrypt.type=0

# 是否使用已有的链（yes/no）
if.exist.fisco=no

# 搭建新链时需配置[if.exist.fisco=no]
# 节点监听Ip
node.listenIp=127.0.0.1
# 节点p2p端口
node.p2pPort=30300
# 节点rpc端口
node.rpcPort=20200
# FISCO-BCOS版本（v3.0.0或以上版本）
fisco.version=v3.4.0
# 搭建节点个数（默认两个）
node.counts=nodeCounts
# 是否搭建Liquid合约链（Solidity和Liquid合约需要二选一，默认Solidity
# 如果使用Liquid，要求在webase-front所在主机配置Liquid环境才能在WeBASE中编译合约
# [0: solidity, 1: liquid]
fisco.wasm=0
# 是否启用链的权限管理，默认不启用
# 如果启用权限，一键部署自动在'webase-deploy/nodes/ca'目录生成一个随机的管理员私钥（包含国密与非国密）
# [0: disabled, 1: enable]
fisco.auth=0

# 使用已有链时需配置[if.exist.fisco=yes]
# 已有链节点rpc端口列表
node.rpcPeers=['127.0.0.1:20200','127.0.0.1:20201']
# sdk目录，需包含已有链SSL所需的证书（如: ca.crt,sdk.crt,sdk.key）
sdk.dir=/data/app/nodes/127.0.0.1/sdk
```

4.4 部署

- 执行installAll命令，部署服务将自动部署FISCO BCOS节点，并部署 WeBASE 中间件服务，包括签名服务（sign）、节点前置（front）、节点管理服务（node-mgr）、节点管理前端（web）

备注：

- 部署脚本会拉取相关安装包进行部署，需保持网络畅通
- 首次部署需要下载编译包和初始化数据库，重复部署时可以根据提示不重复操作
- 部署过程中出现报错时，可根据错误提示进行操作，或根据本文档中的常见问题进行排查
- **不要用sudo执行脚本**，例如`sudo python3 deploy.py installAll`（sudo会导致无法获取当前用户的环境变量如JAVA_HOME）

```
# 部署并启动所有服务
python3 deploy.py installAll
```

部署完成后可以看到`deploy has completed`的日志：

```
$ python3 deploy.py installAll
...
============================================================
              _    _      _______  ___  _____ _____
             | |  | |     | ___ \/ _ \/  ___|  ___|
             | |  | | ___| |_/ / /_\ \ `--.| |__
             | |/\| |/ _ | ___ |  _  |`--. |  __|
             \  /\  |  __| |_/ | | | /\__/ | |___
              \/  \/ \___\____/\_| |_\____/\____/
...
...
============================================================
==============      deploy  has completed     ==============
============================================================
==============    webase-web version  v...        ========
==============    webase-node-mgr version  v...   ========
==============    webase-sign version  v...       ========
==============    webase-front version  v...      ========
============================================================
```

- 服务部署后，需要对各服务进行启停操作，可以使用以下命令：

```
# 一键部署
部署并启动所有服务          python3 deploy.py installAll
停止一键部署的所有服务      python3 deploy.py stopAll
启动一键部署的所有服务      python3 deploy.py startAll
# 各子服务启停
启动FISCO-BCOS节点:       python3 deploy.py startNode
停止FISCO-BCOS节点:       python3 deploy.py stopNode
启动WeBASE-Web:           python3 deploy.py startWeb
停止WeBASE-Web:           python3 deploy.py stopWeb
启动WeBASE-Node-Manager: python3 deploy.py startManager
停止WeBASE-Node-Manager: python3 deploy.py stopManager
启动WeBASE-Sign:          python3 deploy.py startSign
停止WeBASE-Sign:          python3 deploy.py stopSign
启动WeBASE-Front:         python3 deploy.py startFront
停止WeBASE-Front:         python3 deploy.py stopFront
```

4.5 状态检查

成功部署后，可以根据以下步骤**确认各个子服务是否启动成功**

4.5.1 检查各子系统进程

通过ps命令，检查各子系统的进程是否存在

- 包含：节点进程nodeXX，节点前置进程webase.front，节点管理服务进程webase.node.mgr，节点管理平台webase-web的nginx进程，以及签名服务进程webase.sign

检查方法如下，若无输出，则代表进程未启动，需要到该子系统的日志中检查日志错误信息，并根据错误提示或本文档的常见问题进行排查

- 检查节点进程，此处部署了两个节点node0, node1

```
$ ps -ef | grep node
```

输出如下

```
root      29977      1  1 17:24 pts/2    00:02:20 /root/fisco/webase/webase-deploy/
↪nodes/127.0.0.1/node1/../fisco-bcos -c config.ini
root      29979      1  1 17:24 pts/2    00:02:23 /root/fisco/webase/webase-deploy/
↪nodes/127.0.0.1/node0/../fisco-bcos -c config.ini
```

- 检查节点前置webase-front的进程

```
$ ps -ef | grep webase.front
```

输出如下

```
root      31805      1  0 17:24 pts/2    00:01:30 /usr/local/jdk/bin/java -Djdk.tls.
↪namedGroups=secp256k1 ... conf/:apps/*:lib/* com.webank.webase.front.Application
```

- 检查节点管理服务webase-node-manager的进程

```
$ ps -ef  | grep webase.node.mgr
```

输出如下

```
root      4696      1  0 17:26 pts/2    00:00:40 /usr/local/jdk/bin/java -Djdk.tls.
↪namedGroups=secp256k1 ... conf/:apps/*:lib/* com.webank.webase.node.mgr.
↪Application
```

- 检查webase-web对应的nginx进程

```
$ ps -ef | grep nginx
```

输出如下

```
root      5141      1  0 Dec08 ?        00:00:00 nginx: master process /usr/sbin/
↪nginx -c /root/fisco/webase/webase-deploy/comm/nginx.conf
```

- 检查签名服务webase-sign的进程

```
$ ps -ef  | grep webase.sign
```

输出如下

```
root      30718      1  0 17:24 pts/2    00:00:19 /usr/local/jdk/bin/java ... conf/
↪:apps/*:lib/* com.webank.webase.sign.Application
```

4.5.2 检查进程端口

通过netstat命令，检查各子系统进程的端口监听情况。

检查方法如下，若无输出，则代表进程端口监听异常，需要到该子系统的日志中检查日志错误信息，并根据错误提示或本文档的常见问题进行排查

- 检查节点channel端口(默认为20200)是否已监听

```
$ netstat -anlp | grep 20200
```

输出如下

```
tcp        0      0 0.0.0.0:20200           0.0.0.0:*               LISTEN      ␣
↪29069/fisco-bcos
```

- 检查webase-front端口(默认为5002)是否已监听

```
$ netstat -anlp | grep 5002
```

输出如下

```
tcp6       0      0 :::5002                 :::*                    LISTEN      ␣
↪2909/java
```

- 检查webase-node-mgr端口(默认为5001)是否已监听

```
$ netstat -anlp | grep 5001
```

输出如下

```
tcp6       0      0 :::5001                 :::*                    LISTEN      ␣
↪14049/java
```

- 检查webase-web端口(默认为5000)在nginx是否已监听

```
$ netstat -anlp | grep 5000
```

输出如下

```
tcp        0      0 0.0.0.0:5000            0.0.0.0:*               LISTEN      ␣
↪3498/nginx: master
```

- 检查webase-sign端口(默认为5004)是否已监听

```
$ netstat -anlp | grep 5004
```

输出如下

```
tcp6       0      0 :::5004                 :::*                    LISTEN      ␣
↪25271/java
```

4.5.3 检查服务日志

各子服务的日志路径如下：

```
|-- webase-deploy # 一键部署目录
|--|-- log # 部署日志目录
|--|-- webase-web # 管理平台目录
|--|--|-- log # 管理平台日志目录
|--|-- webase-node-mgr # 节点管理服务目录
|--|--|-- log # 节点管理服务日志目录
|--|-- webase-sign # 签名服务目录
|--|--|-- log # 签名服务日志目录
|--|-- webase-front # 节点前置服务目录
|--|--|-- log # 节点前置服务日志目录
|--|-- nodes # 一件部署搭链节点目录
|--|--|-- 127.0.0.1
|--|--|--|-- node0 # 具体节点目录
|--|--|--|--|-- log # 节点日志目录
```

备注：当前节点日志路径为一键部署搭链的路径，使用已有链请在相关路径查看日志

日志目录中包含`{XXX}.log`全量日志文件和`{XXX}-error.log`错误日志文件

- 通过日志定位错误问题时，可以结合`.log`全量日志和`-error.log`错误日志两种日志信息进行排查。，如查询WeBASE-Front日志，则打开`WeBASE-Front-error.log`可以快速找到错误信息，根据错误查看`WeBASE-Front.log`的相关内容，可以看到错误日志前后的普通日志信息

检查服务日志有无错误信息

- 如果各个子服务的进程**已启用**且端口**已监听**，可直接访问下一章节访问*WeBASE*
- 如果上述检查步骤出现异常，如检查不到进程或端口监听，则需要按日志路径进入**异常子服务**的日志目录，检查该服务的日志
- 如果检查步骤均无异常，但服务仍无法访问，可以分别检查部署日志`deployLog`，节点前置日志`frontLog`, 节点管理服务日志`nodeMgrLog`进行排查：
  - 检查webase-deploy/log中的**部署日志**，是否在部署时出现错误
  - 检查webase-deploy/webase-front/log中的**节点前置日志**，如果最后出现`application run success`字样则代表运行成功
  - 检查webase-deploy/webase-node-mgr/log或webase-deploy/webase-sign/log中的日志
  - 检查webase-deploy/nodes/127.0.0.1/nodeXXX/log中的节点日志

搜索日志

通过查看日志可以检查服务的运行状态，我们可以进入各子服务的日志路径，通过`grep`检查日志文件，以此判断服务是否正常运行

- **查看运行成功日志**：WeBASE子服务运行成功后均会打印日志`main run success`，可以通过搜索此关键字来确认服务正常运行。

如，检查webase-front日志，其他WeBASE服务可进行类似操作

```
$ cd webase-front
$ grep -B 3 "main run success" log/WeBASE-Front.log
```

输出如下：

```
2020-12-09 15:47:25.355 [main] INFO  ScheduledAnnotationBeanPostProcessor() - No␣
↪TaskScheduler/ScheduledExecutorService bean found for scheduled processing
2020-12-09 15:47:25.378 [main] INFO  TomcatEmbeddedServletContainer() - Tomcat␣
↪started on port(s): 5002 (http)
2020-12-09 15:47:25.383 [main] INFO  Application() - Started Application in 6.983␣
↪seconds (JVM running for 7.768)
2020-12-09 15:47:25.383 [main] INFO  Application() - main run success...
```

- **查看报错日志**：出现异常时，可以搜索关键字`ERROR`进行检查

如，检查webase-front错误日志，其他WeBASE服务可进行类似操作

```
$ cd webase-front
$ grep "ERROR" log/WeBASE-Front.log
```

输出如下

```
2020-12-09 09:10:42.138 [http-nio-5002-exec-1] ERROR ExceptionsHandler() - catch␣
↪frontException:  no active connection available network exception requset send␣
↪failed! please check the log file content for reasons.
2020-12-09 09:10:42.145 [http-nio-5002-exec-4] ERROR Web3ApiService() -␣
↪getBlockNumber fail.
```

如果出现错误日志，根据错误提示或本文档的常见问题进行排查

启动失败或无法使用时，欢迎到WeBASE提交Issue或到技术社区共同探讨。

- 提交Issue或讨论问题时，可以在issue中配上自己的**环境配置，操作步骤，错误现象，错误日志**等信息，方便社区用户快速定位问题

4.6 访问

WeBASE管理平台：

- 一键部署完成后，**打开浏览器（Chrome Safari或Firefox）访问**

```
http://{deployIP}:{webPort}
示例：http://localhost:5000
```

备注：

- 部署服务器IP和管理平台服务端口需对应修改，网络策略需开通
  - 使用云服务厂商的服务器时，需要开通网络安全组的对应端口。如开放webase使用的5000端口
- WeBASE管理平台使用说明请查看使用手册（获取WeBASE管理平台默认账号和密码，并初始化系统配置）
  - 默认账号为admin，默认密码为Abcd1234。首次登陆要求重置密码
  - 添加节点前置WeBASE-Front到WeBASE管理平台；一键部署时，节点前置与节点管理服务默认是同机部署，添加前置则填写IP为127.0.0.1，默认端口为5002。参考上文中common.properties的配置项front.port={frontPort}
- 检查节点前置是否启动，可以通过访问http://{frontIp}:{frontPort}/WeBASE-Front(默认端口5002)；访问前，确保服务端已对本地机器开放端口，如开放front的5002端口。（不建议节点前置的端口对公网开放访问权限，应对部分机器IP按需开放）

4.7 附录

4.7.1 1. Java环境部署

CentOS环境安装Java

注意：CentOS下OpenJDK无法正常工作，需要安装OracleJDK下载链接。

```
# 创建新的文件夹，安装Java 8或以上的版本，推荐JDK8-JDK13版本，将下载的jdk放在software目录
# 从Oracle官网(https://www.oracle.com/java/technologies/downloads/#java8)选择Java 8或以上的版本下载，例如下载jdk-8u301-linux-x64.tar.gz
$ mkdir /software

# 解压jdk
$ tar -zxvf jdk-8u301-linux-x64.tar.gz

#修改解压后文件的文件名
$mv jdk1.8.0_301 jdk-8u301

# 配置Java环境，编辑/etc/profile文件
$ vim /etc/profile

# 打开以后将下面三句输入到文件里面并保存退出
export JAVA_HOME=/software/jdk-8u301  #这是一个文件目录，非文件
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar

# 生效profile
$ source /etc/profile
```

(下页继续)

(续上页)

```
# 查询Java版本，出现的版本是自己下载的版本，则安装成功。
java -version
```

Ubuntu环境安装Java

```
# 安装默认Java版本(Java 8或以上)
sudo apt install -y default-jdk
# 查询Java版本
java -version
```

4.7.2 2. 数据库部署

① CentOS安装MariaDB

此处以**CentOS 7(x86_64)安装MariaDB 10.2**为例。*MariaDB*数据库是 MySQL 的一个分支，主要由开源社区在维护，采用 GPL 授权许可。*MariaDB*完全兼容 MySQL，包括API和命令行。MariaDB 10.2版本对应Mysql 5.7。其他安装方式请参考MySQL官网。

- CentOS 7 默认MariaDB为5.5版本，安装10.2版本需要按下文进行10.2版本的配置。
- 若使用CentOS 8则直接使用`sudo yum install -y mariadb*`即可安装MariaDB 10.3，并跳到下文的 启停 章节即可。

使用`vi`或`vim`创建新文件`/etc/yum.repos.d/mariadb.repo`，并写入下文的文件内容（参考MariaDB中科大镜像源修改进行配置）

- 创建repo文件

```
sudo vi /etc/yum.repos.d/mariadb.repo
```

- 文件内容，此处使用的是中科大镜像源

```
# MariaDB 10.2 CentOS repository list - created 2021-07-12 07:37 UTC
# http://downloads.mariadb.org/mariadb/repositories/
[mariadb]
name = MariaDB
baseurl = https://mirrors.ustc.edu.cn/mariadb/yum/10.2/centos7-amd64
gpgkey=https://mirrors.ustc.edu.cn/mariadb/yum/RPM-GPG-KEY-MariaDB
gpgcheck=1
```

- 更新yum源缓存数据

```
yum clean all
yum makecache all
```

- 安装`MariaDB 10.2`
- 如果已存在使用`sudo yum install -y mariadb*`命令安装的MariaDB，其版本默认为5.5版本，对应Mysql版本为5.5。新版本MariaDB无法兼容升级，需要先**卸载旧版本**的MariaDB，卸载前需要**备份**数据库内容，卸载命令可参考`yum remove mariadb`

```
sudo yum install MariaDB-server MariaDB-client -y
```

若安装时遇到错误"`Failed to connect to 2001:da8:d800:95::110: Network is unreachable`"，将源地址中的`mirrors.ustc.edu.cn`替换为`ipv4.mirrors.ustc.edu.cn`以强制使用 IPv4：

```
sudo sed -i 's#//mirrors.ustc.edu.cn#//ipv4.mirrors.ustc.edu.cn#g' /etc/yum.repos.
↪d/mariadb
```

详情参考MariaDB官网安装。

- 启停

```
启动: sudo systemctl start mariadb.service
停止: sudo systemctl stop  mariadb.service
```

- 设置开机启动

```
sudo systemctl enable mariadb.service
```

- 初始化

```
执行以下命令:
sudo mysql_secure_installation
以下根据提示输入:
Enter current password for root (enter for none):<-初次运行直接回车
Set root password? [Y/n] <- 是否设置root用户密码，输入y并回车或直接回车
New password: <- 设置root用户的密码
Re-enter new password: <- 再输入一次你设置的密码
Remove anonymous users? [Y/n] <- 是否删除匿名用户，回车
Disallow root login remotely? [Y/n] <-是否禁止root远程登录，回车
Remove test database and access to it? [Y/n] <- 是否删除test数据库，回车
Reload privilege tables now? [Y/n] <- 是否重新加载权限表，回车
```

② 授权访问和添加用户

- 使用root用户登录，密码为初始化设置的密码

```
mysql -uroot -p -h localhost -P 3306
```

- 授权root用户远程访问

```
mysql > GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '123456' WITH␣
↪GRANT OPTION;
mysql > flush PRIVILEGES;
```

- 创建test用户并授权本地访问

```
mysql > GRANT ALL PRIVILEGES ON *.* TO 'test'@localhost IDENTIFIED BY '123456'␣
↪WITH GRANT OPTION;
mysql > flush PRIVILEGES;
```

安全温馨提示:

- 例子中给出的数据库密码（123456）仅为样例，强烈建议设置成复杂密码
- 例子中root用户的远程授权设置会使数据库在所有网络上都可以访问，请按具体的网络拓扑和权限控制情况，设置网络和权限帐号

③ 测试连接和创建数据库

- 登录数据库

```
mysql -utest -p123456 -h localhost -P 3306
```

- 创建数据库

```
mysql > create database webasenodemanager;
```

④ Ubuntu安装mysql数据库

- 以root用户执行命令

```
apt-get install software-properties-common
sudo add-apt-repository 'deb http://archive.ubuntu.com/ubuntu trusty universe'
sudo apt-get update
sudo apt install mysql-server-5.6
sudo apt install mysql-client-5.6
```

- 执行mysql –version命令，若显示如下则安装成功

```
mysql  Ver 14.14 Distrib 5.6.16, for debian-linux-gnu (x86_64) using  EditLine␣
↪wrapper
```

4.7.3 3. Python部署

python版本要求使用python3.x, 推荐使用python3.6及以上版本

- CentOS

```
sudo yum install -y python36
sudo yum install -y python36-pip
```

- Ubuntu

```
// 添加仓库，回车继续
sudo add-apt-repository ppa:deadsnakes/ppa
// 安装python 3.6
sudo apt-get install -y python3.6
sudo apt-get install -y python3-pip
```

4.8 常见问题

4.8.1 1. Python命令出错

- SyntaxError报错

```
  File "deploy.py", line 62
    print helpMsg
                ^
SyntaxError: Missing parentheses in call to "print". Did you mean print(helpMsg)?
```

- 找不到fallback关键字

```
File "/home/ubuntu/webase-deploy/comm/utils.py", line 127, in getCommProperties
    value = cf.get('common', paramsKey,fallback=None)
TypeError: get() got an unexpected keyword argument 'fallback'
```

答：检查*Python*版本，推荐使用python3.6及以上版本

4.8.2 2. 使用Python3时找不到pymysql

```
Traceback (most recent call last):
...
ImportError: No module named 'pymysql'
```

答：需要安装PyMySQL，安装请参看 *pymysql*

4.8.3 3. 部署时某个组件失败，重新部署提示端口被占用问题

答：因为有个别组件是启动成功的，需先执行“python deploy.py stopAll”将其停止，再执行“python deploy.py installAll”部署全部。

4.8.4 4. 管理平台启动时Nginx报错

```
...
==============      WeBASE-Web      start...  ==============
Traceback (most recent call last):
...
Exception: execute cmd  error ,cmd : sudo /usr/local/nginx/sbin/nginx -c /data/app/
↪webase-deploy/comm/nginx.conf, status is 256 ,output is nginx: [emerg] open() "/
↪etc/nginx/mime.types" failed (2: No such file or directory) in /data/app/webase-
↪deploy/comm/nginx.conf:13
```

答：缺少/etc/nginx/mime.types文件，建议重装nginx。

4.8.5 5. 部署时数据库访问报错

```
...
checking database connection
Traceback (most recent call last):
  File "/data/temp/webase-deploy/comm/mysql.py", line 21, in dbConnect
    conn = mdb.connect(host=mysql_ip, port=mysql_port, user=mysql_user,␣
↪passwd=mysql_password, charset='utf8')
  File "/usr/lib64/python2.7/site-packages/MySQLdb/__init__.py", line 81, in␣
↪Connect
    return Connection(*args, **kwargs)
  File "/usr/lib64/python2.7/site-packages/MySQLdb/connections.py", line 193, in __
↪init__
    super(Connection, self).__init__(*args, **kwargs2)
OperationalError: (1045, "Access denied for user 'root'@'localhost' (using␣
↪password: YES)")
```

答：确认数据库用户名和密码

4.8.6 6. 节点sdk目录不存在

```
...
======= FISCO-BCOS sdk dir:/data/app/nodes/127.0.0.1/sdk is not exist. please␣
↪check! =======
```

答：确认节点安装目录下有没有sdk目录（企业部署工具搭建的链可能没有），如果没有，需手动创建”mkdir sdk”，并将节点sdk证书（ca.crt、sdk.key、sdk.crt）复制到该sdk目录，再重新部署。如果使用国密SSL，则在sdk目录里存放国密sdk证书（sm_ca.crt、sm_sdk.crt、sm_sdk.key、sm_ensdk.crt、sm_ensdk.key）。

4.8.7 7. 前置启动报错“nested exception is javax.net.ssl.SSLException”

```
...
nested exception is javax.net.ssl.SSLException: Failed to initialize the client-
↪side SSLContext: Input stream not contain valid certificates.
```

答：CentOS的yum仓库的OpenJDK缺少JCE(Java Cryptography Extension)，导致Web3SDK/Java-SDK无法正常连接区块链节点，因此在使用CentOS操作系统时，推荐使用*OracleJDK*。

4.8.8 8.前置启动报错“Processing bcos message timeout”

```
...
[main] ERROR SpringApplication() - Application startup failed
org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating␣
↪bean with name 'contractController': Unsatisfied dependency expressed through␣
↪field 'contractService'; nested exception is org.springframework.beans.factory.
↪UnsatisfiedDependencyException: Error creating bean with name 'contractService':␣
↪Unsatisfied dependency expressed through field 'web3jMap'; nested exception is␣
↪org.springframework.beans.factory.BeanCreationException: Error creating bean␣
↪with name 'web3j' defined in class path resource [com/webank/webase/front/config/
↪Web3Config.class]: Bean instantiation via factory method failed; nested␣
↪exception is org.springframework.beans.BeanInstantiationException: Failed to␣
↪instantiate [java.util.HashMap]: Factory method 'web3j' threw exception; nested␣
↪exception is java.io.IOException: Processing bcos message timeout
...
```

答：一些OpenJDK版本缺少相关包，导致节点连接异常。推荐使用*OracleJDK*。

4.8.9 9. 服务进程起来了，服务不正常

```
...
======= WeBASE-Node-Manager  starting . Please check through the log file (default␣
↪path:./webase-node-mgr/log/). =======
```

答：查看日志，确认问题原因。确认后修改重启，如果重启提示服务进程在运行，先执行“python deploy.py stopAll”将其停止，再执行“python deploy.py startAll”重启。

4.8.10 10. WeBASE-Web登录页面的验证码加载不出来

答：检查WeBASE-Node-Manager后台服务是否已启动成功。若启动成功，检查后台日志：

- 进入 `webase-node-mgr` 目录下，执行 `bash status.sh` 检查服务是否启动，如果服务没有启动，运行 `bash start.sh` 启动服务；
- 如果服务已经启动，按照如下修改日志级别
  - `webase-node-mgr/conf/application.yml`

```
#log config
logging:
  level:
    com.webank.webase.node.mgr: debug
```

  - `webase-node-mgr/conf/log/log4j2.xml`

```
<Loggers>
<Root level="debug">
  <AppenderRef ref="asyncInfo"/>
```

(下页继续)

(续上页)

```
  <AppenderRef ref="asyncErrorLog"/>
</Root>
</Loggers>
```

- 修改日志level后，重启服务 `bash stop.sh && bash start.sh`
- 重启服务后，检查日志文件 `log/WeBASE-Node-Manager.log`。
  - 检查是否有异常信息。如果有异常信息，根据具体的异常信息检查环境配置，或者通过搜索引擎进行排查。

4.8.11 11. WeBASE 国内镜像与CDN加速服务

答：WeBASE CDN 加速服务提供 WeBASE 各子系统安装包的下载服务，可参考国内镜像和CDN加速攻略

欢迎给*WeBASE*的文档提交 *Pull Request* 补充更多的 *Q&A*

CHAPTER 5

企业部署

5.1 1、企业部署

WeBASE四个服务的部署架构如下图：节点前置需要和区块链节点部署在同一台机器，签名服务可以和节点前置分开部署，也可以同机部署；节点管理和WeBASE管理平台可以同机部署，也可以分开部署。在企业生产环境，为了容灾往往会在多个节点上部署节点前置，也会部署多个签名服务、节点管理和WeBASE管理台。

重要：FISCO-BCOS 2.0与3.0对比、JDK版本、WeBASE及其他子系统的版本兼容说明！请查看

具体部署可以参考《WeBASE管理平台使用说明》中手动搭建部分。

5.2 2、使用手册

WeBASE管理平台的使用请查看使用手册。

CHAPTER 6

国内镜像和CDN加速攻略

本节为访问GitHub较慢的用户提供国内源码镜像与安装包下载地址，以及WeBASE文档加速访问介绍。

6.1 WeBASE及子系统源码及安装包

6.1.1 源码同步

WeBASE当前仓库源码位于https://github.com/WeBankBlockchain/WeBASE，每个新的版本发布会将代码合入master分支。

为了方便国内用户，我们同样在gitee上提供了镜像仓库https://giteee.com/Webank/WeBASE，每次新版本发布后，镜像仓库会同步GitHub上官方仓库的更新，如果从GitHub下载失败，请尝试使用Gitee镜像仓库。

WeBASE各子系统的Github代码仓库则是`https://github.com/WeBankBlockchain/` + `WeBASE-XXX`，对应的gitee仓库则是`https://gitee.com/WeBank/` + `WeBASE-XXX`

如WeBASE-Front的Github代码仓库为`https://github.com/WeBankBlockchain/WeBASE-Front`，Gitee代码仓库为`https://gitee.com/WeBank/WeBASE-Front`

6.1.2 一键部署与安装包

WeBASE每个新版本发布后，会在WeBASELargefilesGitHub的Releases中提供对应的WeBASE一键部署工具和对应安装包。

其中WeBASELargefiles提供webase-deploy一键部署工具（即WeBASE源码中`/deploy`文件夹），以及webase-front.zip, webase-node-mgr.zip, webase-sign.zip, webase-web.zip子系统的安装包。

同时提供以下国内镜像，**可加速下载安装包**：

```
https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/releases/download/
↪{release_version}/webase-{subsystem}.zip
```

其中`{release_version}`为`v1.x.x`格式，`{subsystem}`则是子系统名字。

支持下载`sign`, `front`, `node-mgr`, `web`子系统的zip安装包（全小写），暂不支持webase-transaction的安装包下载。

注：可视化部署中通过国内镜像手动下载*docker*镜像，文件名则是*docker-fisco-webase.tar*或国密版*docker-fisco-webase-gm.tar*

6.2 WeBASE文档镜像

WeBASE文档使用readthedocs管理，全部开源于https://webasedoc.readthedocs.io/zh_CN/lab/index.html，同样提供国内镜像文档https://fintech.webank.com/developer/docs/webase/index.html，由于网站资源更新周期安排，国内镜像文档更新会比*readthedocs*有所延迟。

每个版本发布会为上个版本的文档打Tag，新版本的文档会合入主干分支，文档由于会持续改进，所以是下个版本发布才打上个版本的tag。readthedocs文档支持下载PDF格式，方便用户使用。

6.3 举例：使用国内镜像进行一键部署

本节WeBASE 1.5.0为例进行一键部署，一键部署会默认使用国内镜像下载安装包，下面仅演示关键步骤，具体操作可参考WeBASE一键部署

6.3.1 下载WeBASE一键部署工具

```
# 使用CDN下载
wget https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/releases/download/
↪v1.5.0/webase-deploy.zip

# 使用github下载
wget https://github.com/WeBankBlockchain/WeBASELargeFiles/releases/download/v1.5.0/
↪webase-deploy.zip
```

6.3.2 单独下载WeBASE子系统的安装包

WeBASE一键部署(webase-deploy)会自动下载子系统安装包，用户也可以手动下载安装包或编译源码得到安装包，并复制到webase-deploy目录下。

如需手动下载某一子系统的安装包，可以直接通过`wget`或者`curl -O`命令直接获取安装包。比如：

- 获取WeBASE-Node-Manager v1.4.1的安装包`webase-node-mgr.zip`

```
wget https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/releases/download/
↪v1.4.1/webase-node-mgr.zip
// 或
curl -#LO https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/releases/
↪download/v1.4.1/webase-node-mgr.zip
```

6.3.3 单独下载WeBASE的solc JS文件

WeBASE提供FISCO BCOS中使用的v0.4.25, v0.5.2, v0.6.10三个版本的solc JS编译文件，对应的国密版本则在版本号后加上`-gm`后缀

如需手动下载某一版本的的安装包，可以直接通过`wget`或者`curl -O`命令直接获取安装包。比如：

- 获取v0.4.25的国密版本solc JS编译文件

```
wget https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/download/solidity/
↪wasm/v0.4.25-gm.js
// 或
curl -#LO https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/download/
↪solidity/wasm/v0.4.25-gm.js
```

若通过源码编译获取安装包并用于一键部署工具，需要进行文件夹的重命名，参考下一章节。

6.4 举例：使用国内源码镜像编译WeBASE-Front

本节以WeBASE-Front子系统为例，从gitee镜像下载源码并编译，编译后的配置方法，请参考各子系统的安装部署文档。

6.4.1 下载源码

```
git clone https://gitee.com/WeBank/WeBASE-Front.git
```

6.4.2 编译源码

依据WeBASE-Front节点前置安装文档的环境要求进行配置，如jdk(oracle jdk8及以上)，配置gradle(4.10或以上)或直接使用gradlew脚本进行编译

```
// java版本
java -version
// 进入源码目录
cd WeBASE-Front
// 已有gradle时，使用gradle
gradle build -x test
// 不配置gradle，直接使用gradlew
chmod +x ./gradlew && ./gradlew build -x test
```

6.4.3 修改配置

编译完成后，将在当前目录得到dist文件夹

重命名dist包中的conf_template为conf后，并将节点的sdk证书复制到conf目录后，即可启动。

修改配置的具体方法，可参考WeBASE-Front部署

```
cd dist
mv conf_template conf
// 此处需要复制ca.crt, node.crt, node.key
cp -rf /fisco/nodes/127.0.0.1/sdk/* ./conf
bash start.sh
```

CHAPTER 7

WeBASE管理平台使用手册

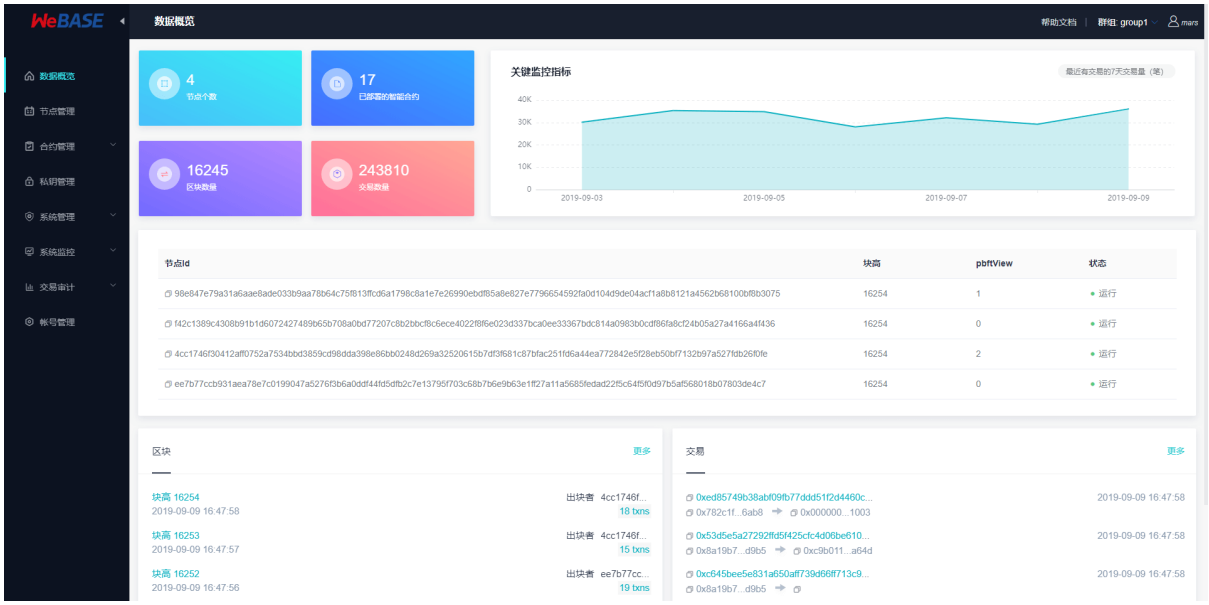
7.1 概览

7.1.1 基本描述

WeBASE管理平台是由四个WeBASE子系统组成的一套管理FISCO-BCOS联盟链的工具集。

7.1.2 主要功能

1. 区块链数据概览
2. 节点管理
3. 合约管理
4. 私钥管理
5. 系统管理
6. 系统监控
7. 交易审计
8. 订阅事件
9. 账号管理
10. 移动端管理台
11. 数据监控大屏



7.1.3 部署架构

这套管理工具主要由：节点前置，签名服务，节点管理，WeBASE管理平台四个WeBASE子系统构成。WeBASE四个服务的部署架构如下图：节点前置需要和区块链节点部署在同一台机器；节点管理和WeBASE管理平台可以同机部署，也可以分开部署。

7.2 使用前提

7.2.1 群组搭建

区块链浏览器展示的数据是从区块链上同步下来的。为了同步数据需要初始化配置（添加群组信息和节点信息），故在同步数据展示前需要用户先搭建好区块链群组。FISCO-BCOS 2.0提供了多种便捷的群组搭建方式。

1. 如果是开发者进行开发调试，建议使用build_chain。
2. 如果是开发企业级应用，建议使用企业部署工具FISCO-BCOS generator。

两者的主要区别在于build_chain为了使体验更好，搭建速度更快，辅助生成了群组内各个节点的私钥；但企业部署工具出于安全的考虑不辅助生成私钥，需要用户自己生成并设置。

Liquid支持

如果使用的`liquid`合约的链，并在WeBASE管理台或WeBASE-Front的合约IDE中编译Liquid合约，要求**手动**在WeBASE-Front所在主机配置Liquid环境，可参考WeBASE-Front节点前置文档中的Liquid配置或Liquid官方配置文档

配置好Liquid环境后，需要重启WeBASE-Front

7.2.2 WeBASE管理平台搭建

WeBASE管理平台分为四个部分：节点前置，签名服务，节点管理，WeBASE管理台。

当前版本我们提供了三种搭建方式：一键搭建、纯手动搭建各子系统、可视化部署。

1、一键搭建

适合同机部署，快速体验的情况使用。具体搭建流程参见安装文档。

2、手动搭建

2.1、签名服务搭建

签名服务使用Spring Boot的JAVA后台服务，具体搭建流程参见《签名服务安装说明》。

2.2、节点前置搭建

节点前置使用Spring Boot的JAVA后台服务，具体搭建流程参见《节点前置安装说明》。

2.3、节点管理搭建

节点管理使用Spring Boot的JAVA后台服务，具体搭建流程参见《节点管理安装说明》。

2.4、WeBASE管理平台

WeBASE管理台使用框架`vue-cli`，具体搭建流程参见《WeBASE管理平台安装说明》。

7.3 系统初始化配置

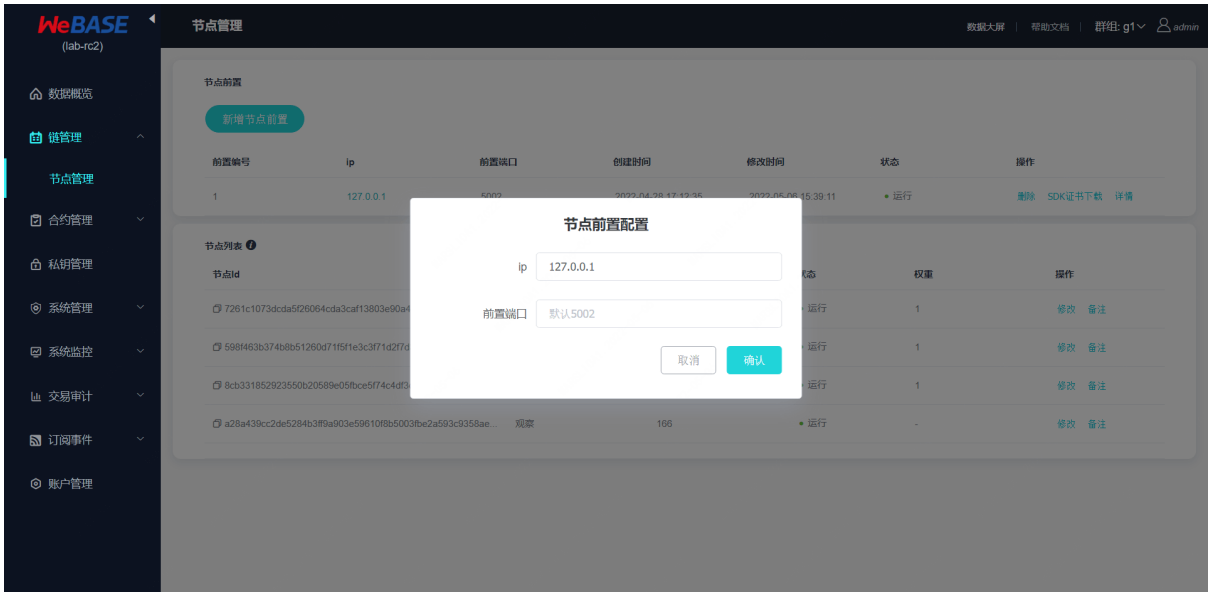
服务搭建成功后，可使用网页浏览器访问nginx配置的WeBASE管理台IP和端口(例如`127.0.0.1:5000`)，进入到管理平台页面。

管理平台默认用户为`admin`，默认密码为`Abcd1234`（第一次登陆成功后会要求重置密码，请按照密码标准设置一个更加安全的密码）。

7.3.1 添加节点前置

未初始化节点前置的管理平台，会引导去节点管理页面添加节点前置。

- 节点前置服务需要填写前置的IP与端口（默认为`127.0.0.1`和`5002`），机构名则根据实际自定义填写

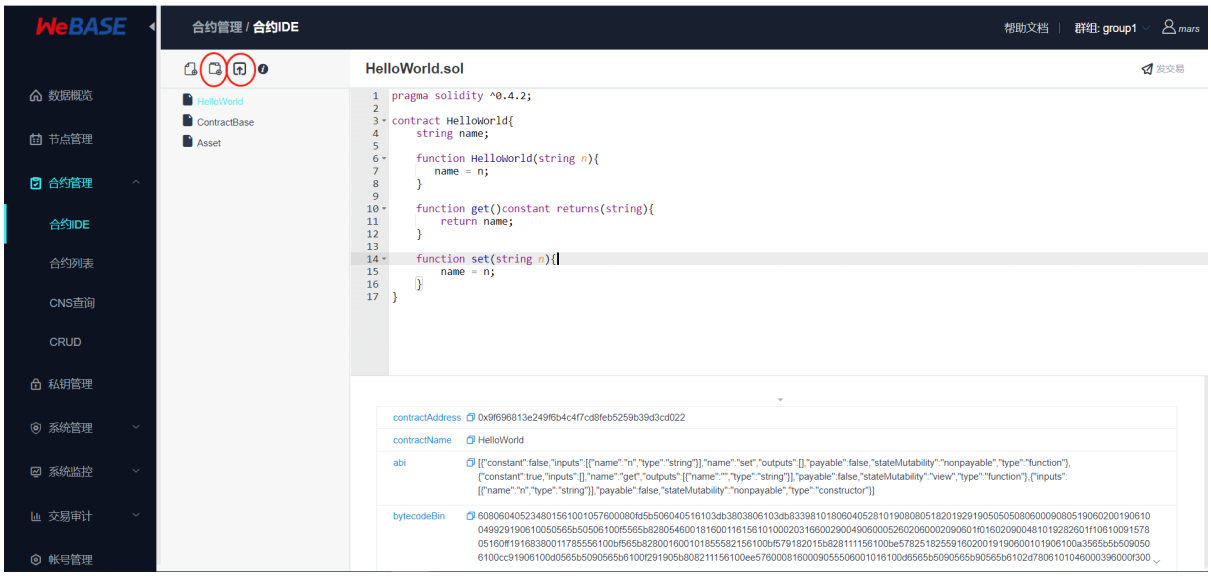


前置添加完成后，管理平台就会开始拉取群组信息和群组的区块信息。此时数据概览页面应该就有数据了。为了解析和审计区块数据，需要把相关的合约和用户导入到管理平台。具体看下面两个小节。

7.3.2 合约管理

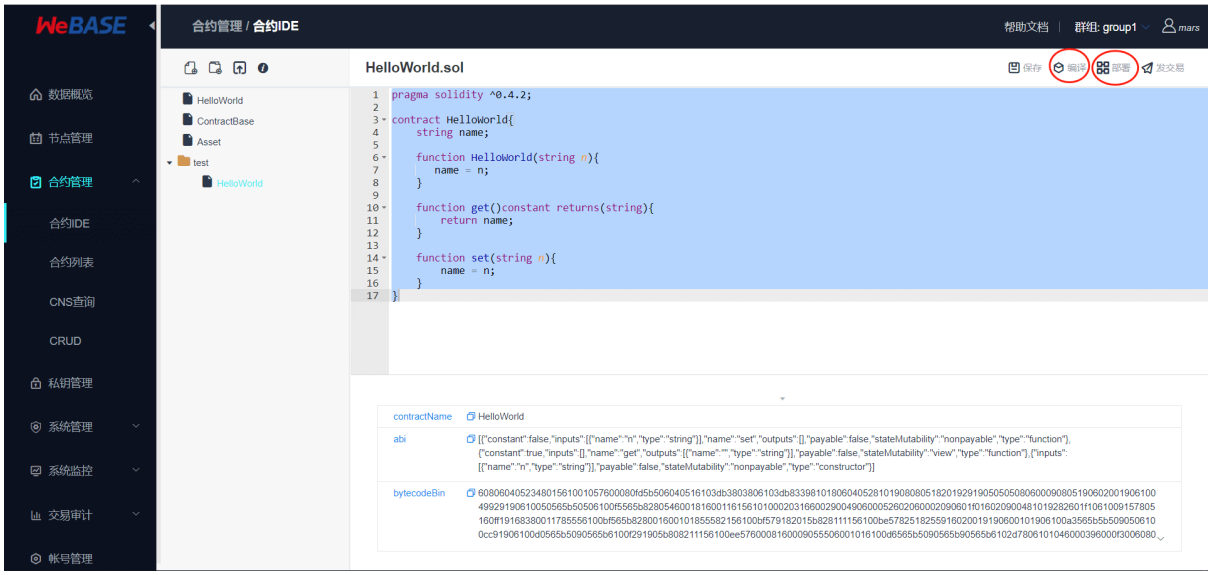
1、添加合约

管理平台提供两种添加合约的方式，一个是新建一个合约，一个是导入已有合约。同时合约编辑器还提供新建目录。用目录的形式管理合约，主要是为了解决同名合约引用的问题。合约添加完成后，需要编译保存。



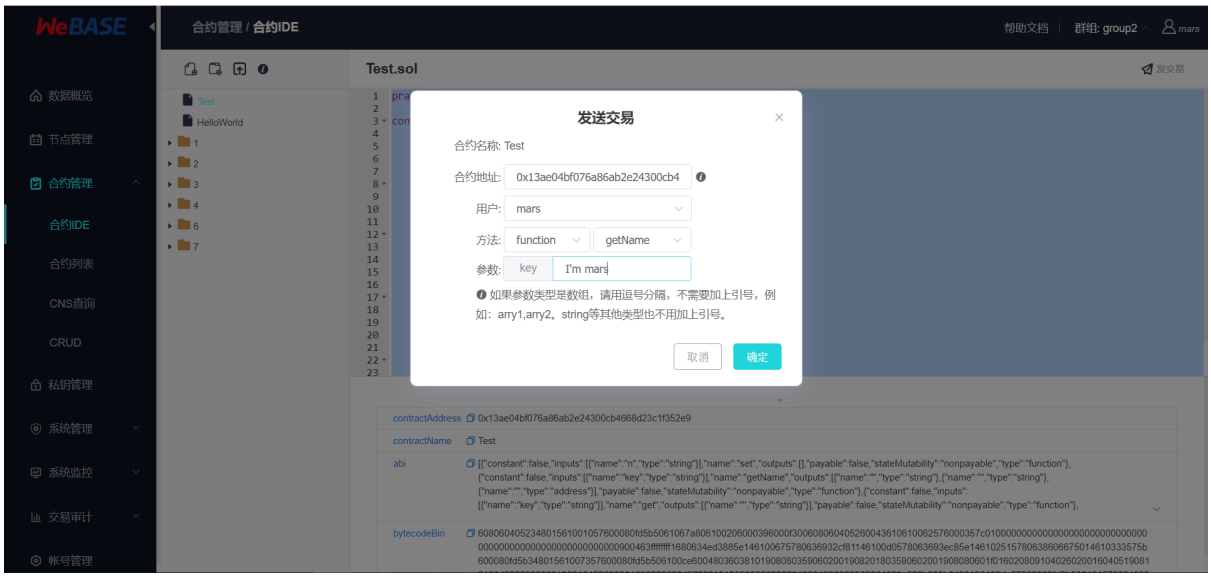
2、部署合约

合约编译时会自动保存合约内容，编译成功后可以执行合约部署。

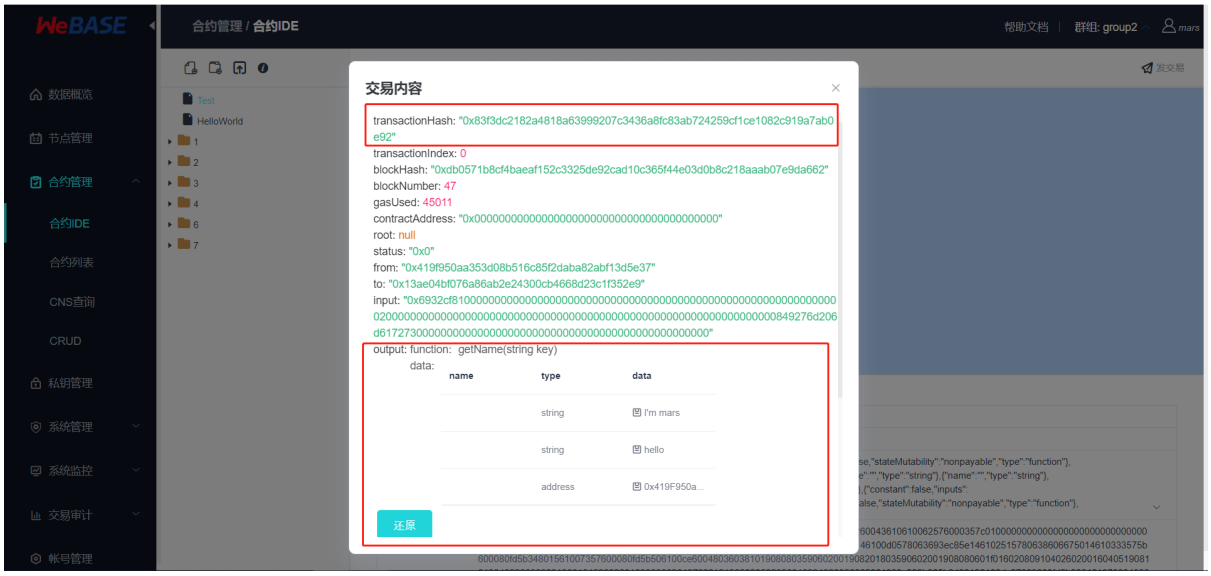


3、合约调用

在合约部署成功后，可以在合约IDE页面的右上角点击发交易，向合约发送交易进行合约调用。

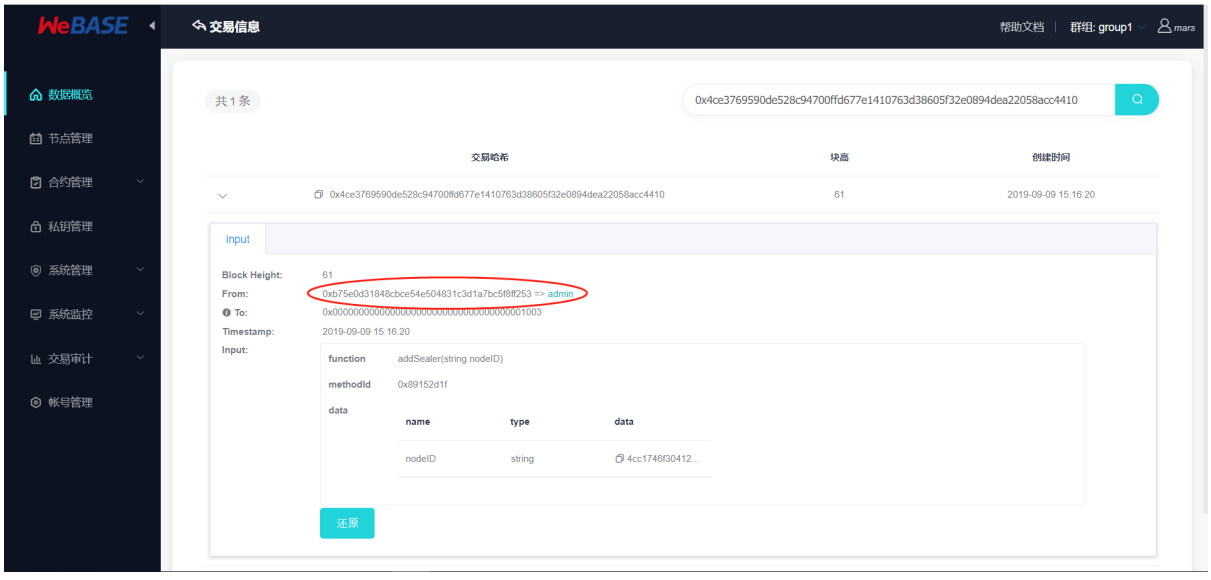


交易发送成功后，将返回交易回执。可以在数据概览-交易列表-更多中根据transactionHash搜索交易，通过交易解析和Event解析查看可视化的交易回执信息。具体操作方法参考下文的区块链数据概览章节中的交易解析与Event解析。(注：Liquid合约的交易暂未支持交易解析)


WeBASE
合约管理 / 合约IDE
帮助文档
群组: group2
交易内容
transactionHash: "0x83f3dc2182a4818a63999207c3436a8fc83ab724259cf1ce1082c919a7ab0e92"
transactionIndex: 0
blockHash: "0xdb0571b8cf4baeaf152c3325de92cad10c365f44e03d0b8c218aaab07e9da662"
blockNumber: 47
gasUsed: 45011
contractAddress: "0x0000000000000000000000000000000000000000"
root: null
status: "0x0"
from: "0x419f950aa353d08b516c85f2daba82abf13d5e37"
to: "0x13ae04bf076a86ab2e24300cb4668d23c1f352e9"
input: "0x6932cf810000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000000849276d206d617273000000000000000000000000000000000000000000000000"
output: function: getName(string key)
data:
name
type
data
string
I'm mars
string
hello
address
0x419F950a...
还原

7.3.3 私钥管理

私钥管理提供了新建私钥用户和导入公钥用户两种用户导入方式。第一种方式主要用于新建用户（私钥托管在签名服务中），在管理平台的合约管理中部署和调用合约。第二种方式主要用于把交易和用户关联起来。


WeBASE
交易信息
帮助文档
群组: group1
共 1 条
0x4ce3769590de528c94700ffd677e1410763d38605f32e0894dea22058acc4410
交易哈希
块高
创建时间
0x4ce3769590de528c94700ffd677e1410763d38605f32e0894dea22058acc4410
61
2019-09-09 15:16:20
input
Block Height: 61
From: 0xb75e0d31848cbce54e504831c3d1a7bc5f8ff253 => admin
To: 0x0000000000000000000000000000000000001003
Timestamp: 2019-09-09 15:16:20
Input:
function addSealer(string nodeID)
methodId 0x89152d1f
data
name
type
data
nodeID
string
4cc1746f30412...
还原

7.4 各模块的详细介绍

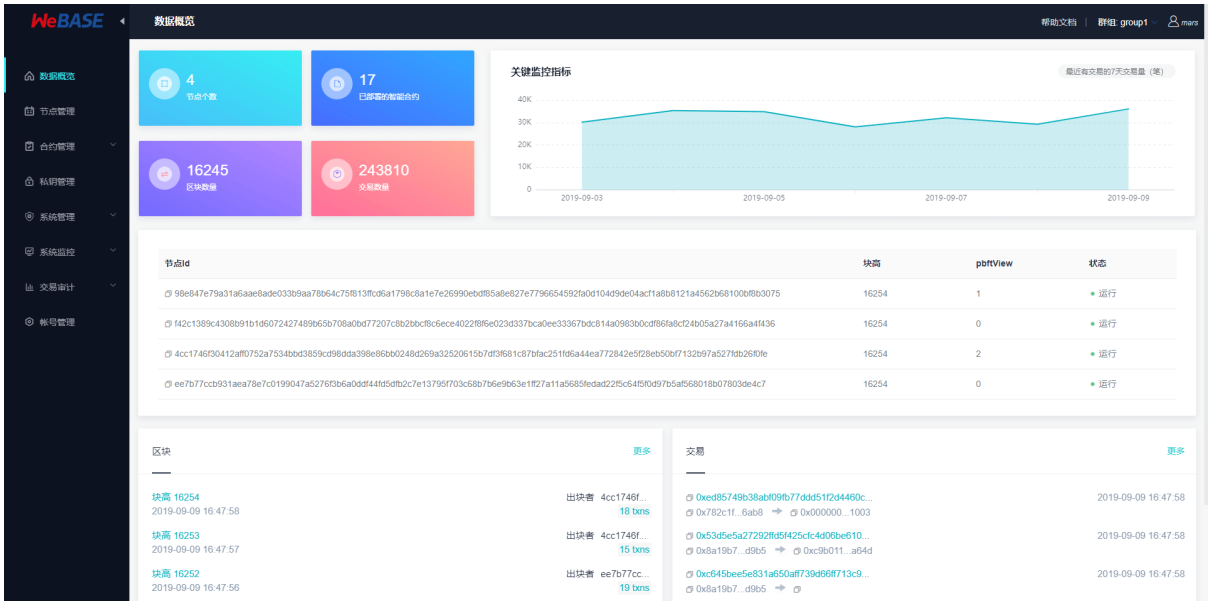
本小节概要介绍管理平台的各个模块，方便大家对WeBASE管理平台套件有一个整体的认识。这套工具集主要提供的管理功能有：

7.4.1 区块链数据概览

数据概览页面，展示了区块链的核心数据指标：节点个数，区块数量，交易数量，通过管理台部署的合约数量。关键监控指标：历史15天的交易量。

- 节点信息列表：展示了节点的ID，节点块高，节点view和运行状态；

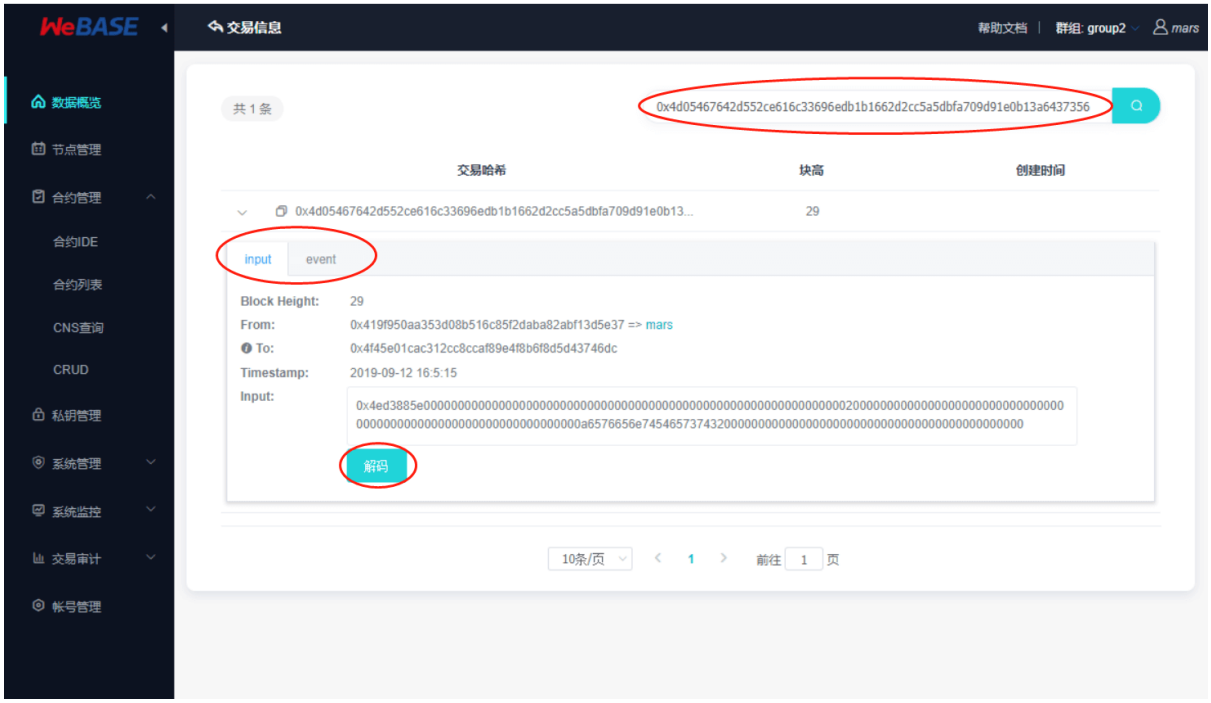
- 区块信息列表：展示了最近5个块的概览信息，点击更多可以查看更多历史区块；
- 交易信息列表：展示了最近5个交易的概览信息，点击更多可以查看更多历史交易；



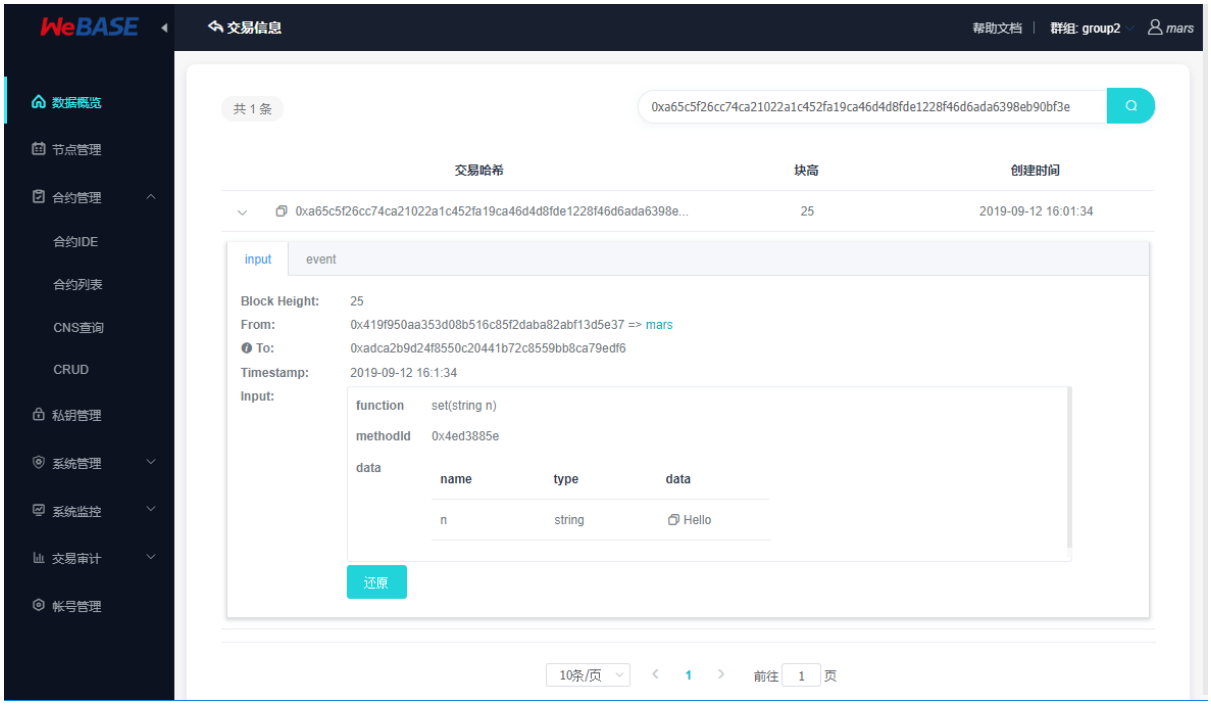
其中右下角的交易信息列表点击可跳入具体一条交易中查看交易详细信息：交易详细信息还包含了

- 交易解析：可以将交易返回的交易回执数据进行解析并可视化；
- Event解析：可以将交易返回的Event数据进行解析并可视化；

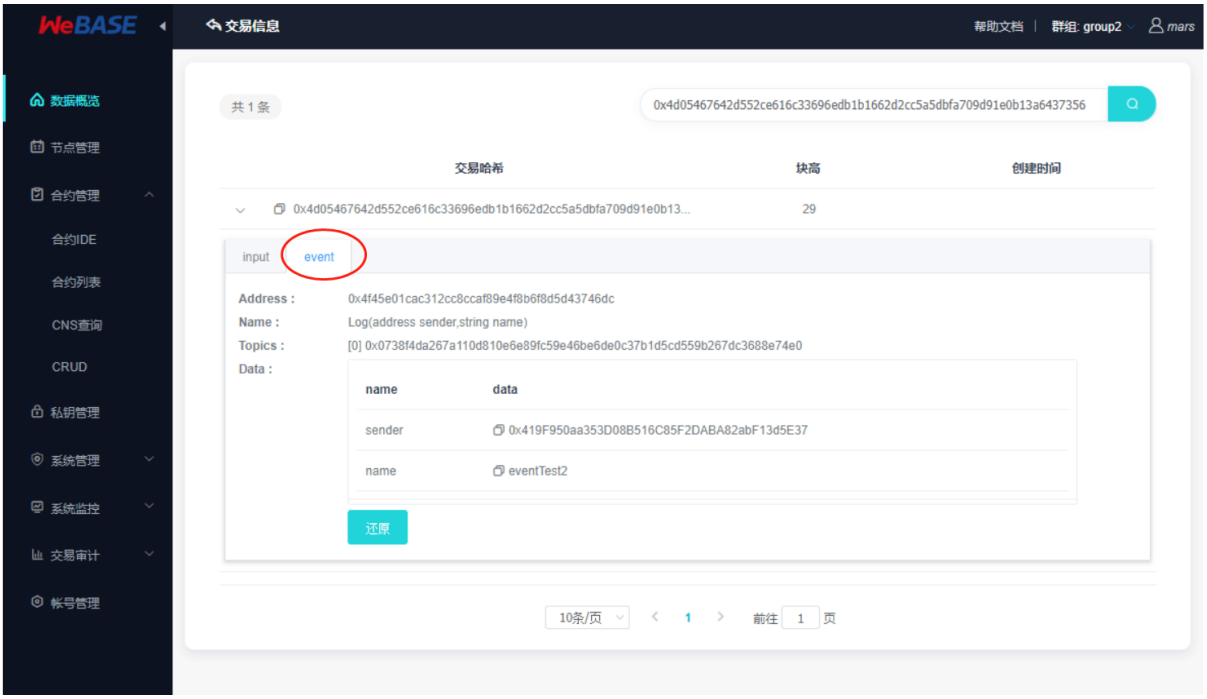
未解析的raw数据如下图所示：



进行交易解析后如下图所示：(注：Liquid合约的交易暂未支持交易解析)



同样的，Event数据解析后可以看到：(注：Liquid合约的交易暂未支持交易解析)



7.4.2 节点管理

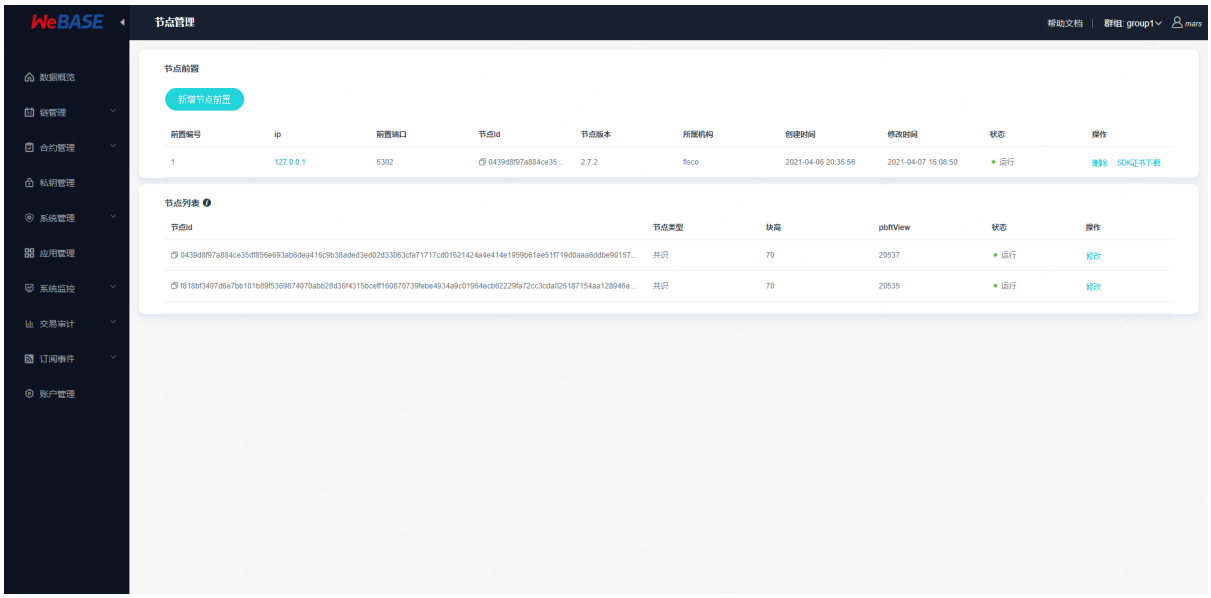
节点管理主要提供了前置列表、节点列表、修改节点共识状态的功能。

用户可以通过新增节点前置，把新的节点前置添加到前置列表。系统会默认拉取这些前置所在的群组和各个群组的节点信息。在节点列表中，用户可以修改节点的共识状态：共识节点、观察节点、游离节点。其中修改为游离节点相当于将节点移出群组，停止节点前务必先将节点设置为游离节点，否则将触发节点异常。

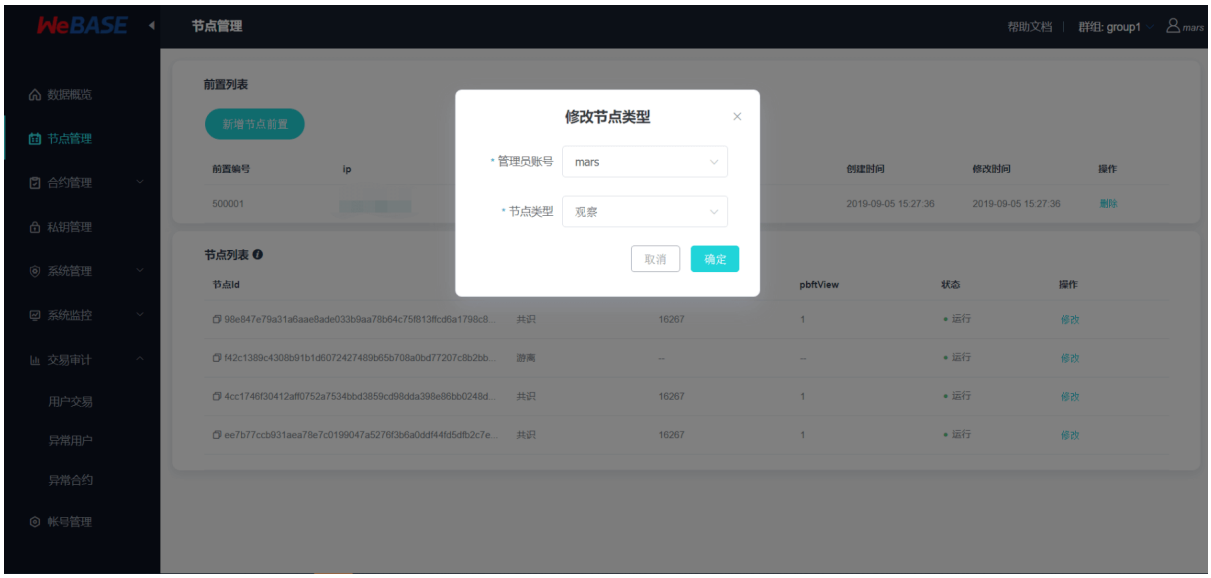
前置列表与节点管理：

- 前置列表：可以查看节点前置状态，导出前置的SDK证书zip包

- 节点管理：显示所有的共识/观察节点（无论运行或停止），以及正在运行的游离节点



修改节点共识状态：



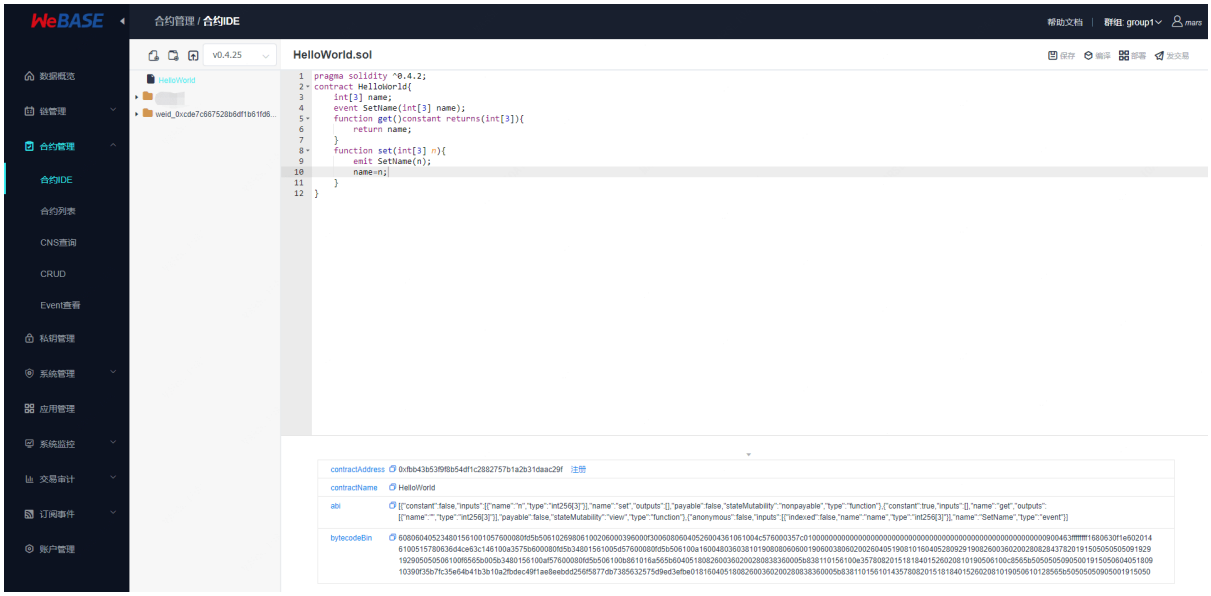
7.4.3 合约管理

合约管理提供了一个图形化的合约IDE环境、已部署的合约列表、合约仓库等功能。

图形化合约IDE提供了一整套的合约管理工具：新建合约，保存合约，编译合约，部署合约，调用合约接口。其中，新建合约可以通过编辑键入合约内容，也可以上传合约文件；编译合约后才可以部署合约；部署合约成功后，可以通过发送交易调用合约接口。具体操作步骤可以参考上一章节中系统初始化配置介绍。

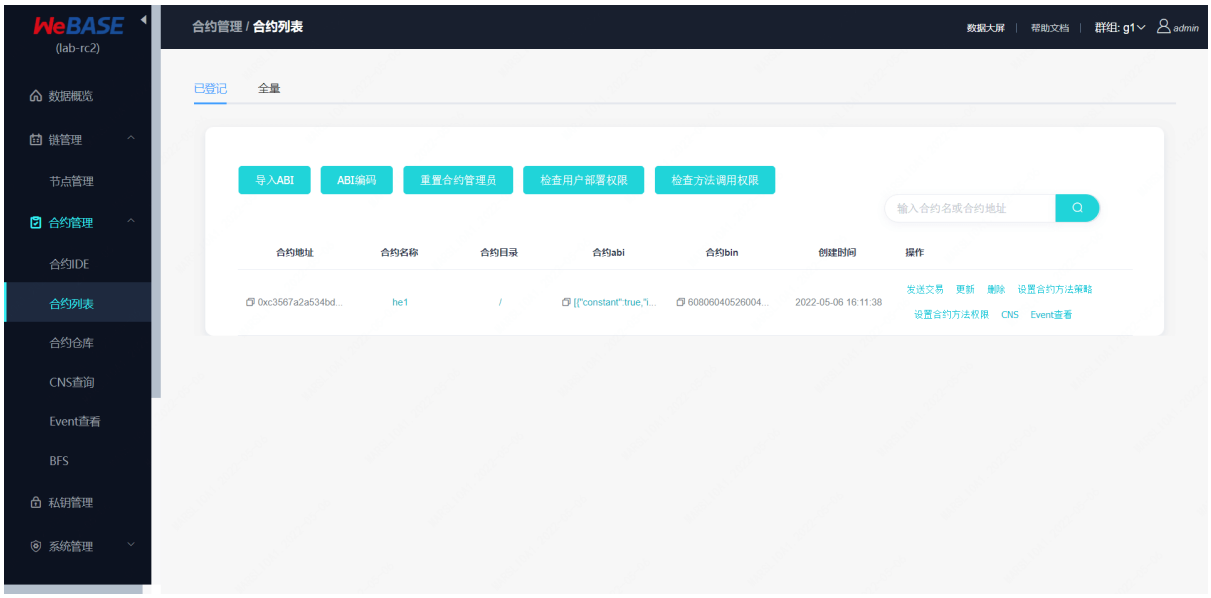
合约IDE：

- 进行Liquid合约编译需要在WeBASE-Front所在主机配置Liquid环境，可参考WeBASE-Front节点前置文档中的Liquid配置或参考Liquid环境配置进行配置后方可使用。
- 若当前群组属于Liquid群组（Wasm群组），合约IDE将自动切换至Liquid编译模式，并自动检查是否已在节点前置所在主机配置Liquid环境。

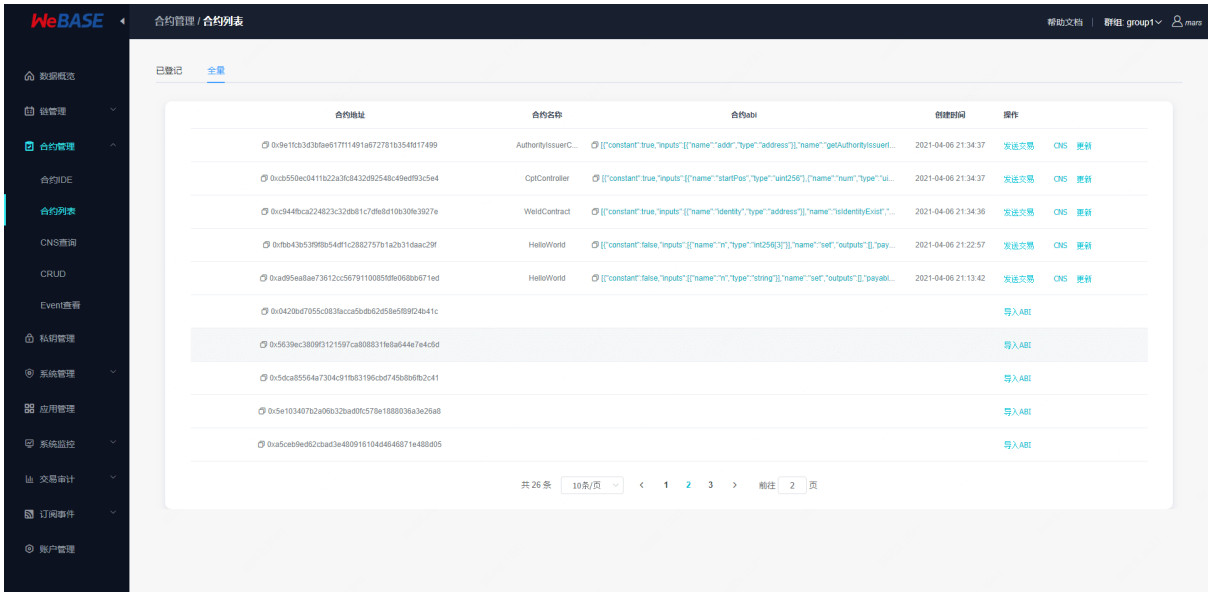


合约列表：包含WeBASE已登记合约与链上已部署的合约

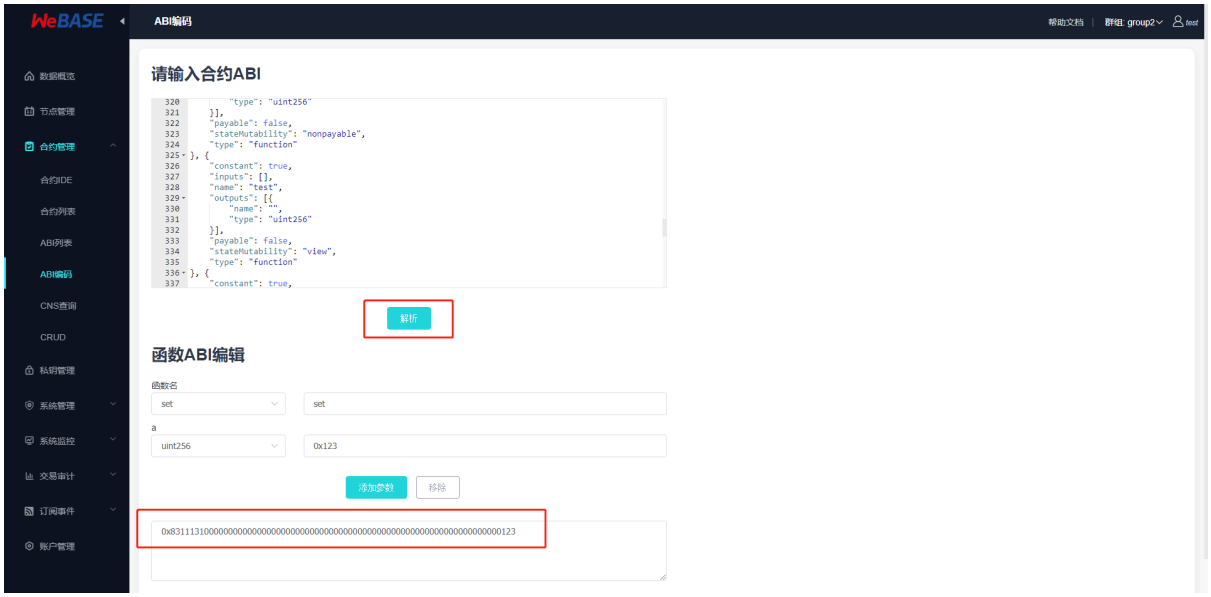
已登记合约：包含通过IDE部署的合约、导入ABI的合约



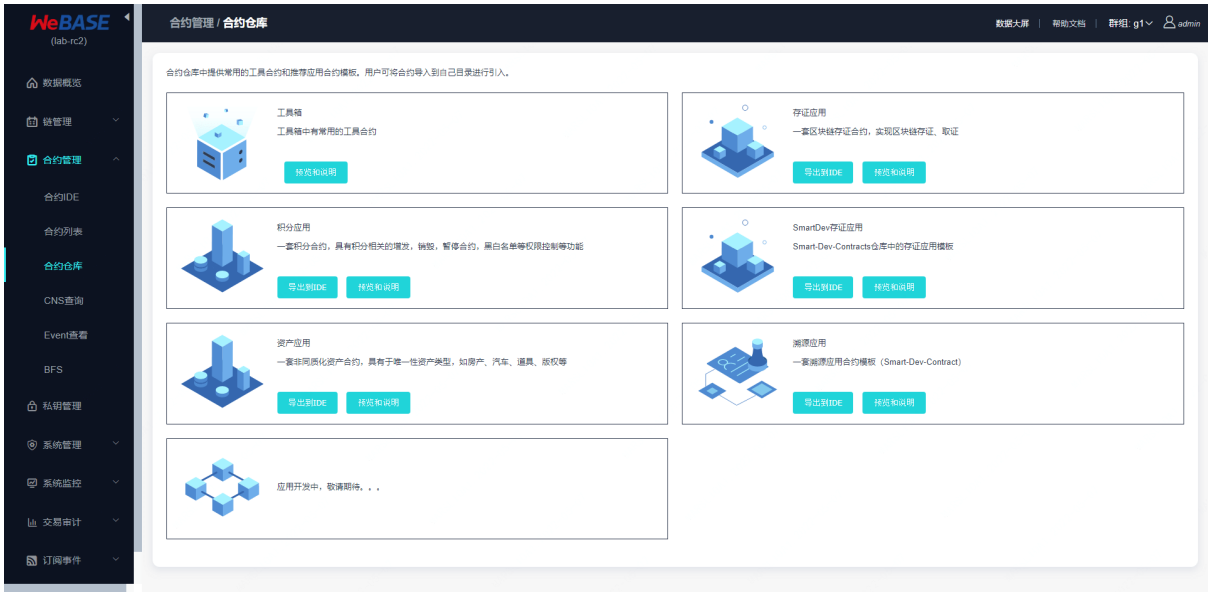
链上全量合约：包含通过其他平台部署到链上的合约与WeBASE已登记的合约（链上合约只有合约地址），可通过导入按钮，填入合约ABI导入到本地



ABI编码：支持对ABI的方法与入参进行编码



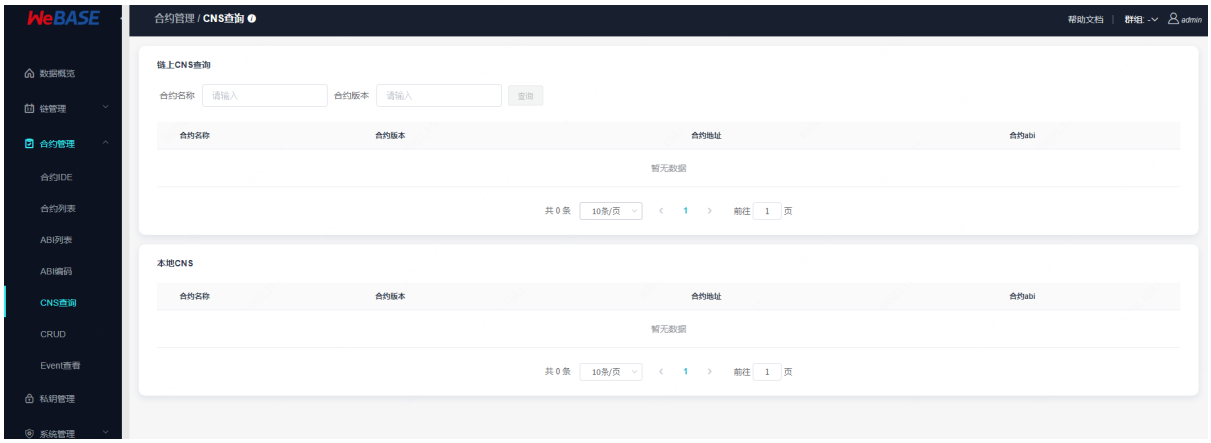
合约仓库：合约仓库是WeBASE整理社区贡献者所贡献的合约案例，其中包含基础的工具合约、存证合约、积分合约等。



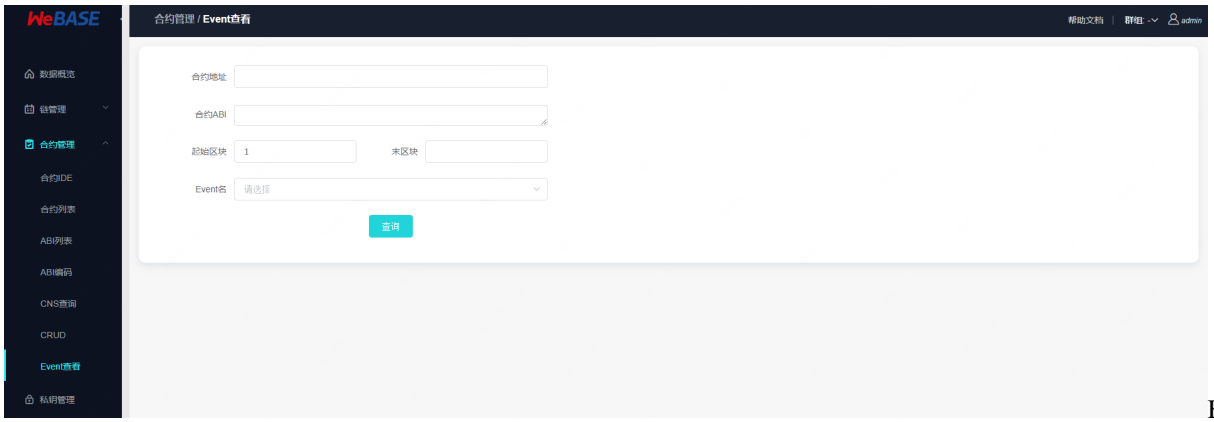
CNS查询：CNS（Contract Name Service）是通过提供链上合约名称与合约地址映射关系的记录及相应的查询功能，方便调用者通过记忆简单的合约名来实现对链上合约的调用。详情可查看FISCO-BCOS文档的 CNS方案

注册CNS后，CNS可以根据合约名和合约版本号查询CNS信息(合约名和合约版本号用英文冒号连接)。若缺失合约版本号，则返回所有符合合约名的合约信息。

- 需要在合约管理页面部署合约时**勾选CNS**，或合约列表页面中点击**CNS注册**，即可完成注册



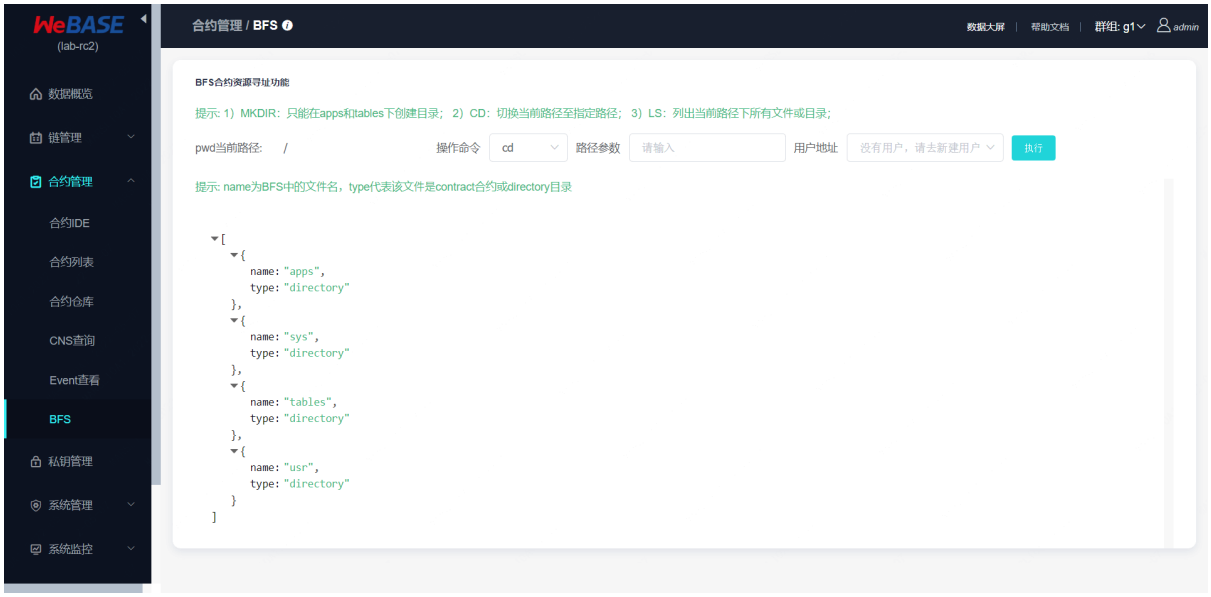
EventLog查看：支持输入合约地址和ABI、区块范围和Event名，即可查询并同步返回



EventLog

BFS：BFS（BBlockchain File System）是FISCO BCOS 3.0中新增的功能，可通过类似文件系统的方式查看链上所部署的合约，详情请参考BFS文档

- 部署的合约会在/apps目录下产生一个合约文件

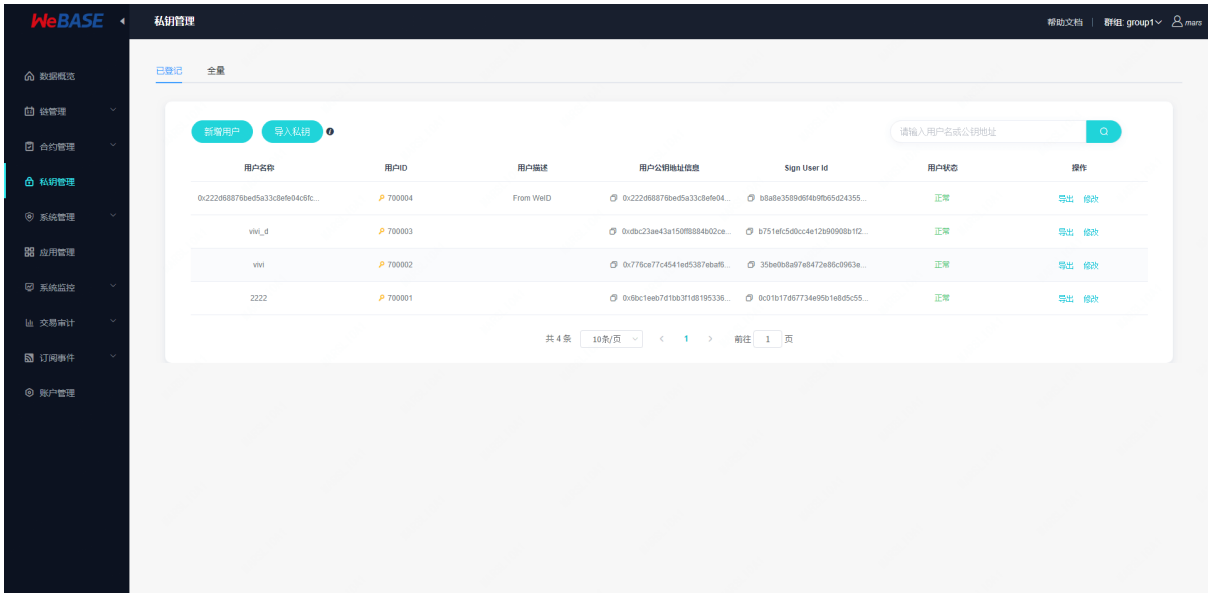


7.4.4 私钥管理

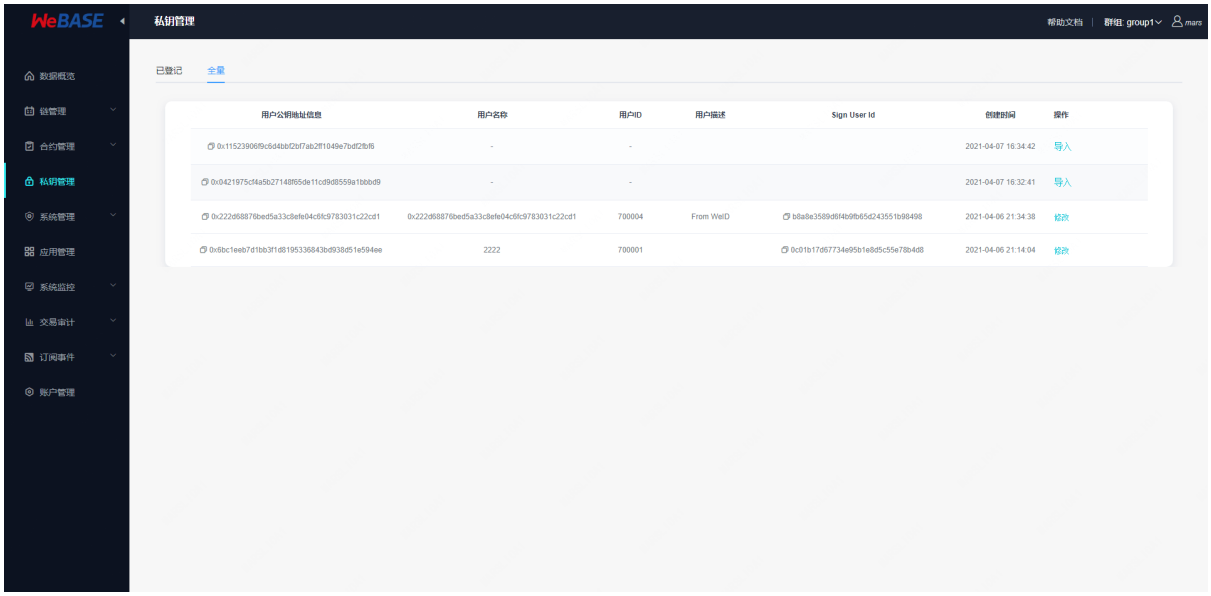
私钥管理包含新建私钥用户和新建公钥用户两个功能。在合约管理界面，可以看到合约部署和交易调用功能。这里的私钥管理可以新建私钥用户，私钥将托管在签名服务中，然后通过签名服务对合约部署和合约调用进行签名。注：外部账户可通过新建公钥账户导入，主要用于把交易和用户关联起来。

私钥管理：包含WeBASE本地已登记的私钥用户与链上全量私钥用户。

已登记私钥：包含本地创建的私钥与导入的私钥

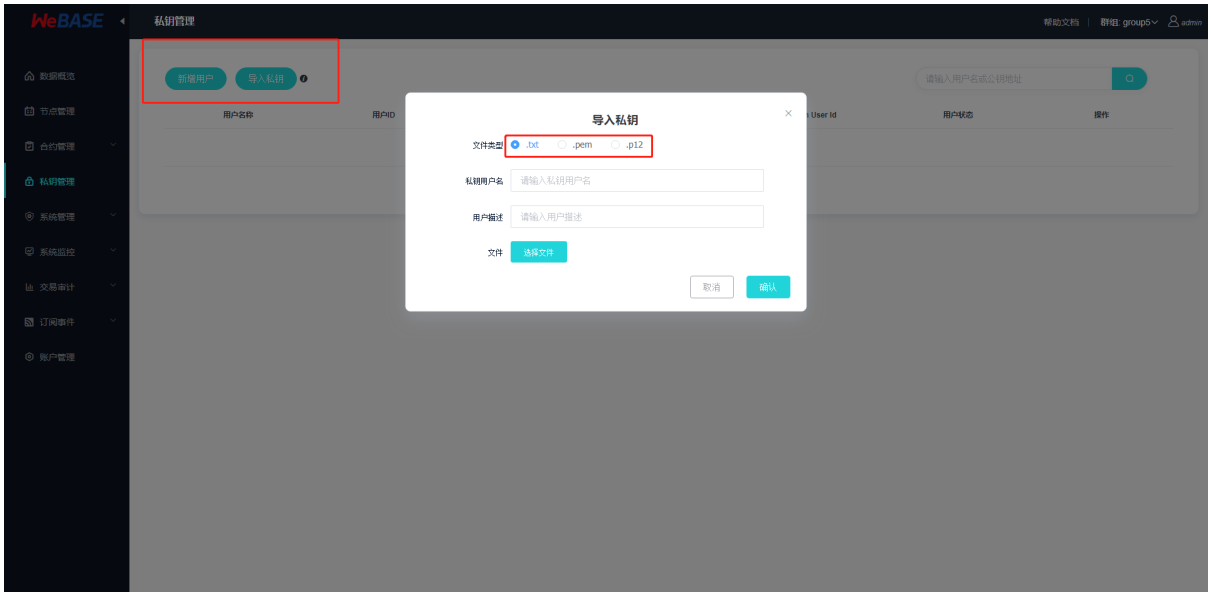


链上全量私钥：包含链上私钥和本地已登记的私钥，可通过导入按钮，作为公钥用户导入到本地

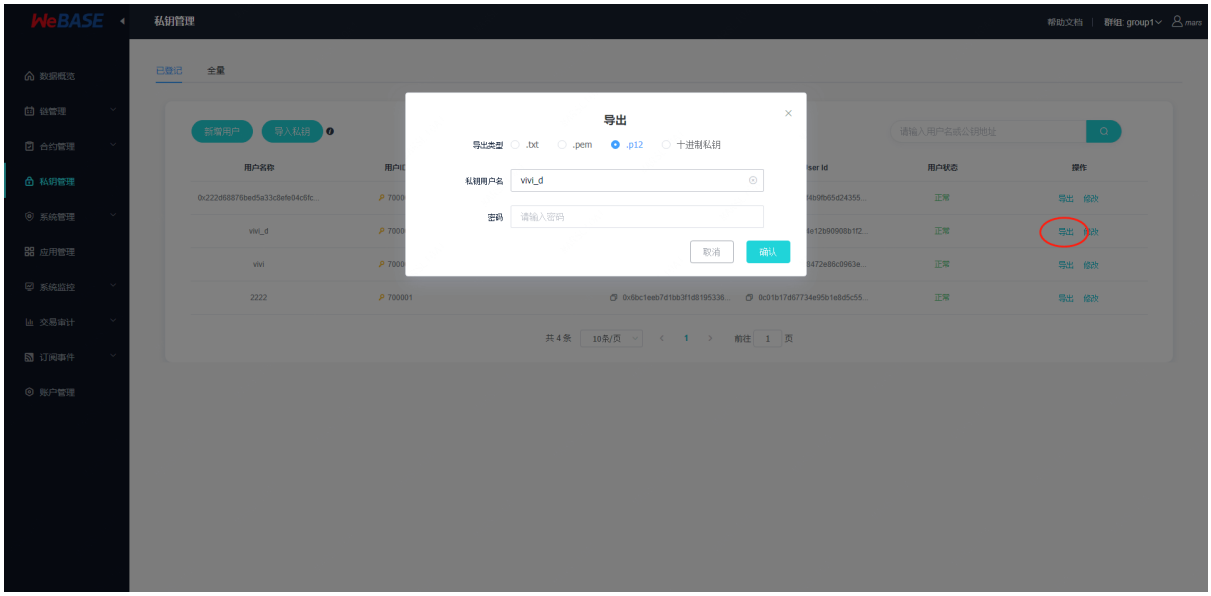


添加私钥用户：

导入私钥：支持导入.txt/.pem/.p12格式及明文的私钥，其中.txt私钥可由WeBASE-Front导出，.pem/.p12私钥可由console控制台导出。如果需要导入自定义私钥，可根据节点前置导出的.txt私钥，编辑其中的privateKey字段内容。



导出私钥：可以选中导出.txt/.pem/.p12/WeID等格式的私钥，其中WeID格式私钥为十进制明文私钥，txt则是十六进制明文私钥；在代码中加载私钥可以参考节点前置-私钥加载



7.4.5 系统管理

系统管理目前支持权限管理、系统配置管理、证书管理的功能。

权限管理：在FISCO BCOS3.0中，链上角色按照不同的权责可划分为三类：治理角色、合约管理员角色和用户角色，三种角色依次进行管理和被管理。详情可参考FISCO BCOS权限治理体系设计和FISCO BCOS权限治理使用指南

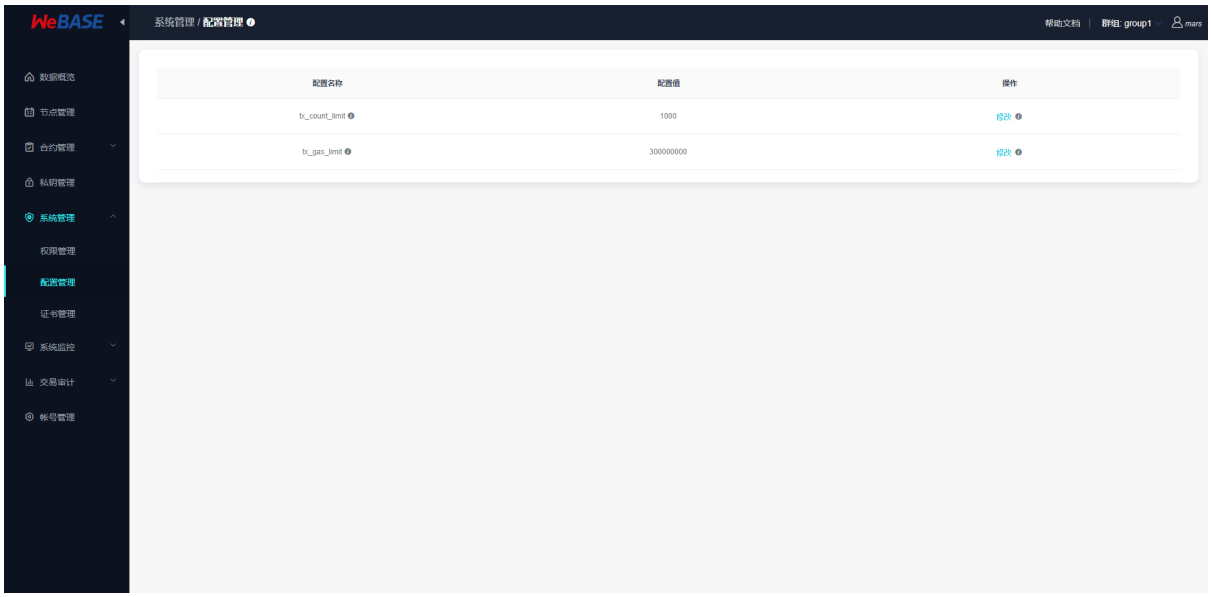
值得注意的是，在区块链初始化启动之前，在配置中必须**开启并设置好权限治理的配置**，才能正确启动权限治理模式。区块链启动后再配置将不起作用。详细方法参考FISCO BCOS权限治理使用指南

- 开启权限治理模式的主要要点是：将`is_auth_check`选项置为`true`，`auth_admin_account`初始委员会账户地址需要配置正确的地址。FISCO BCOS不同的节点部署模式，开启权限治理的方式略有不同
- 开启权限后，需要**在WeBASE权限管理页面中导入对应的链管理员私钥**

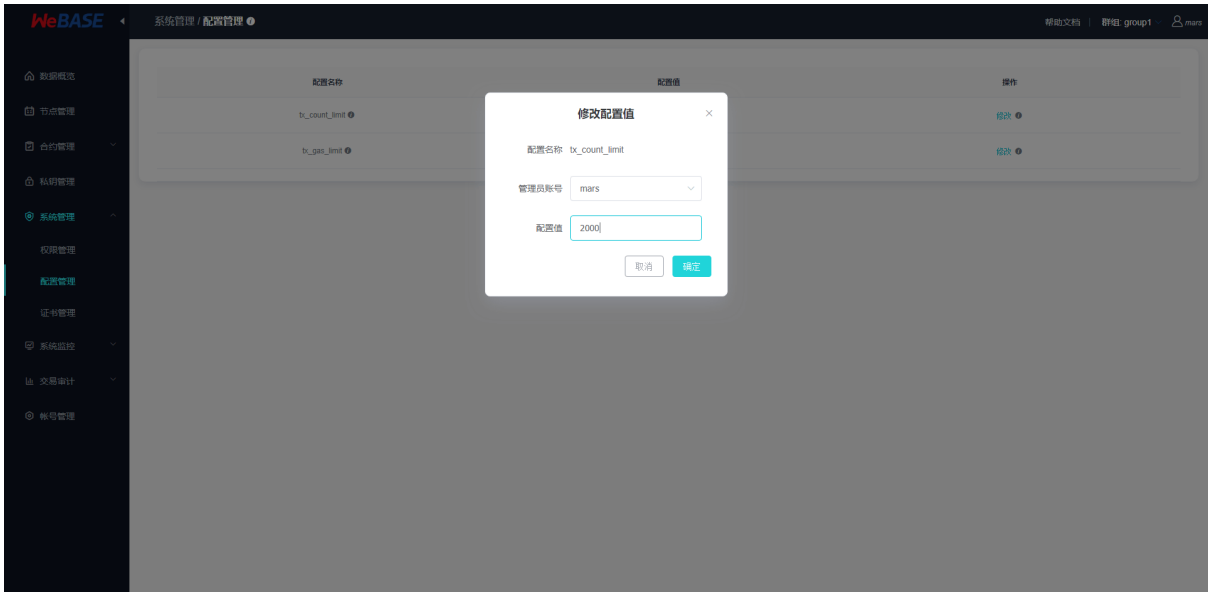
系统配置管理：系统属性包含FISCO-BCOS链的tx_count_limit和tx_gas_limit两种属性值的配置。注：一般不建议随意修改tx_count_limit和tx_gas_limit，如下情况可修改这些参数：

- 机器网络或CPU等硬件性能有限：调小tx_count_limit，降低业务压力；
- 业务逻辑太复杂，执行区块时gas不足：调大tx_gas_limit。

配置管理：



配置系统属性值：



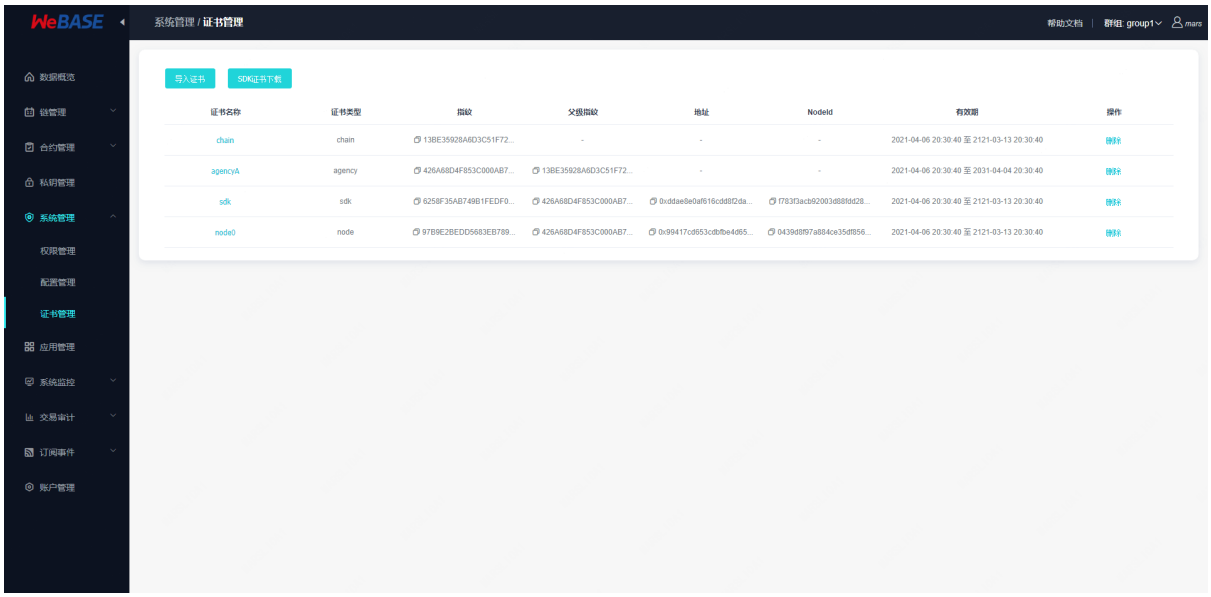
证书管理：支持导入和查看证书信息，包括查看Front对应节点的链证书、机构证书、节点证书，可查看证书内容、证书有效期、证书链关系等信息；

- 证书链关系可通过比对父证书指纹与证书指纹查找；
- 平台将默认加载所有Front的证书，需要在Webase-Front配置文件中配置nodePath节点路径；

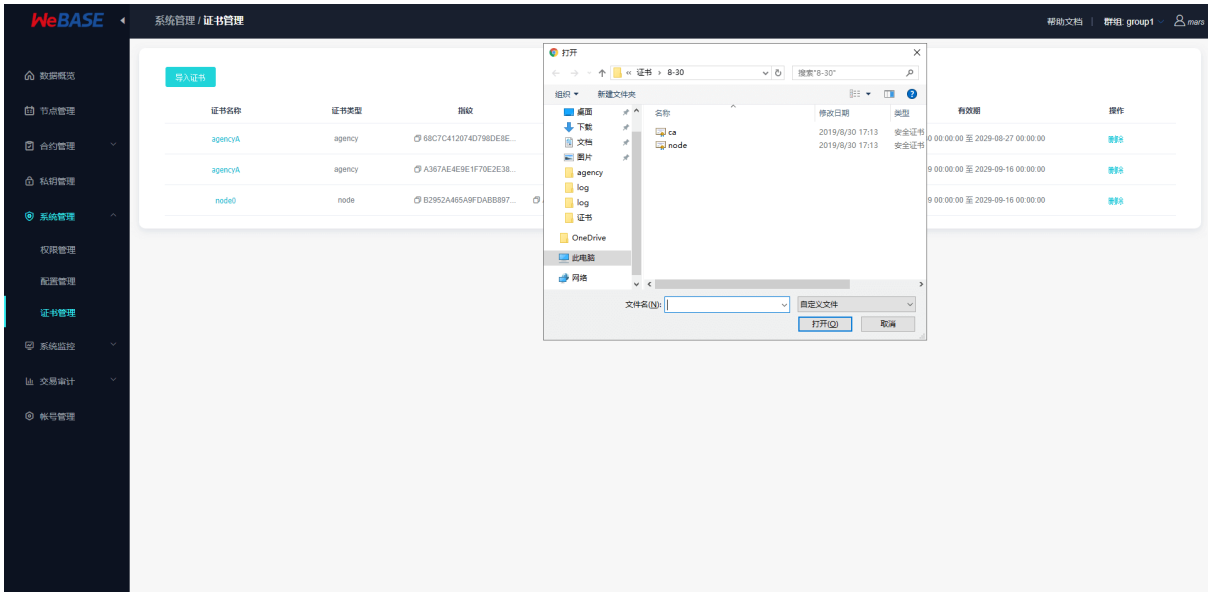
FISCO-BCOS证书说明可以参考FISCO-BCOS使用手册的证书说明

证书列表：

- 支持导出SDK证书：v1.5.0后支持导出节点前置的SDK证书zip包，可用于连接节点



导入证书：



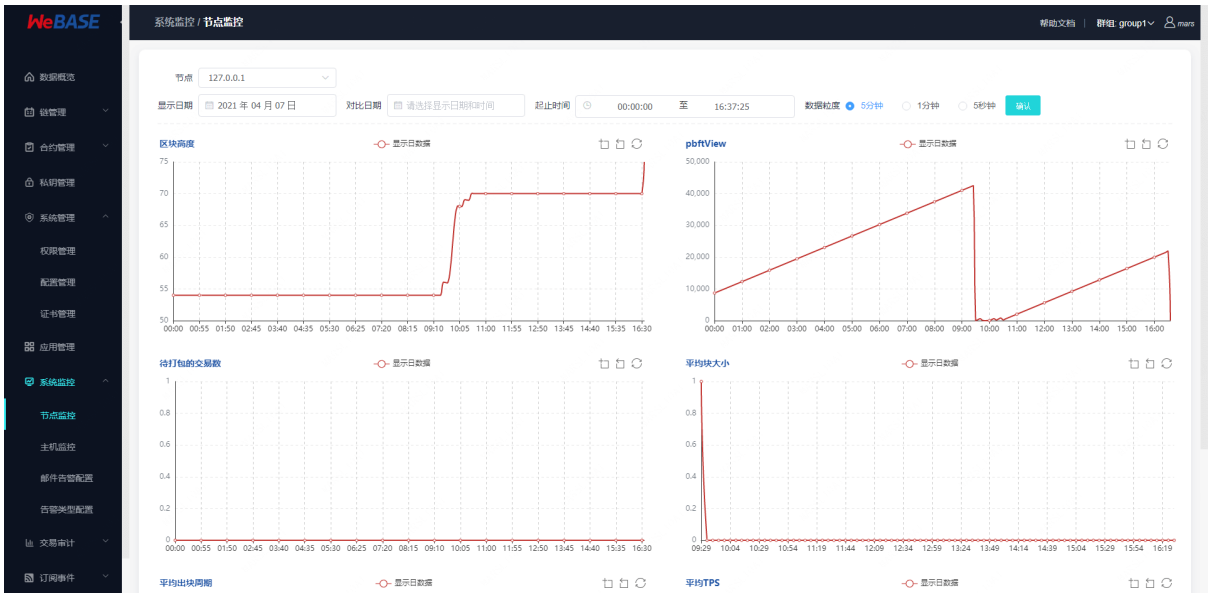
7.4.6 系统监控

系统监控包含了**监控**与**异常告警**两部分；

监控主要包括节点监控和主机监控，可以选择节点、时间范围等条件进行筛选查看：

- 节点监控主要有区块高度，pbftview，待打包交易；
- 主机监控主要有主机的CPU，内存，网络和硬盘IO；

节点监控：



主机监控：

异常告警部分主要包括**邮件服务配置**和**告警类型配置**:

邮件服务配置:

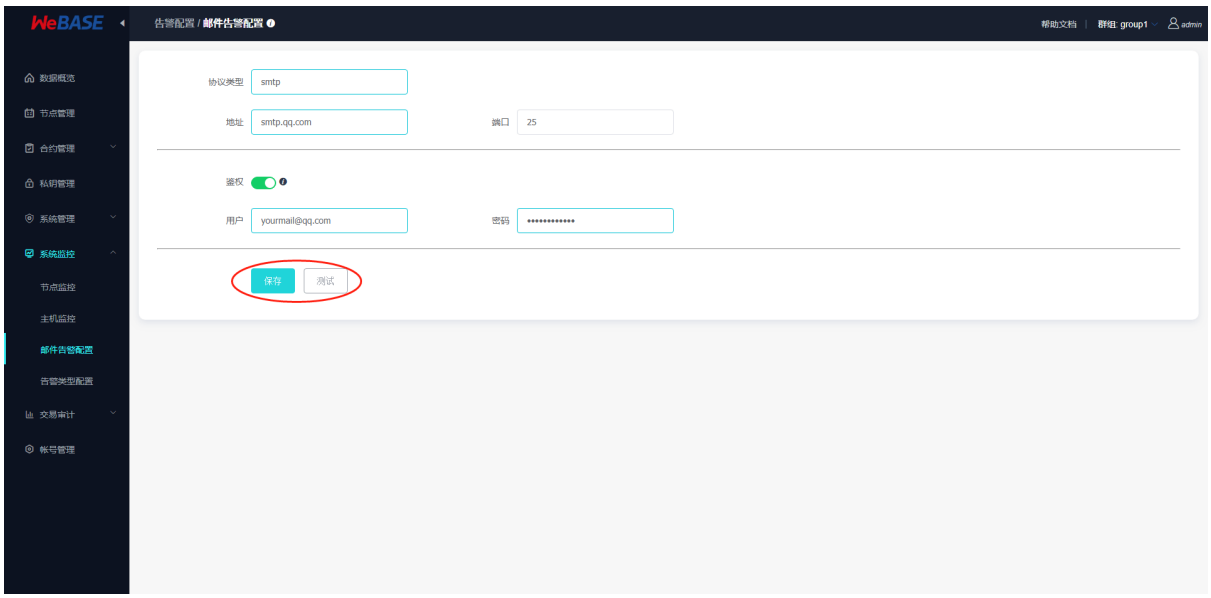
如何配置邮件服务可查看本文档末尾的附录-配置邮件服务指南

可配置邮件告警所用到的邮件服务器相关参数，包含邮件协议类型protocol、邮件服务器地址host、服务使用端口port、用户邮箱地址username、用户邮箱授权码password；鉴权选项包含Authentication验证开关authentication（默认开启）；

- 邮件告警的邮箱协议类型默认使用SMTP协议，使用25默认端口，默认使用username/password进行用户验证，目前仅支持通过TLS/SSL连接邮件服务器;
- 目前仅支持更新原有的邮件服务器配置，不支持新增配置;

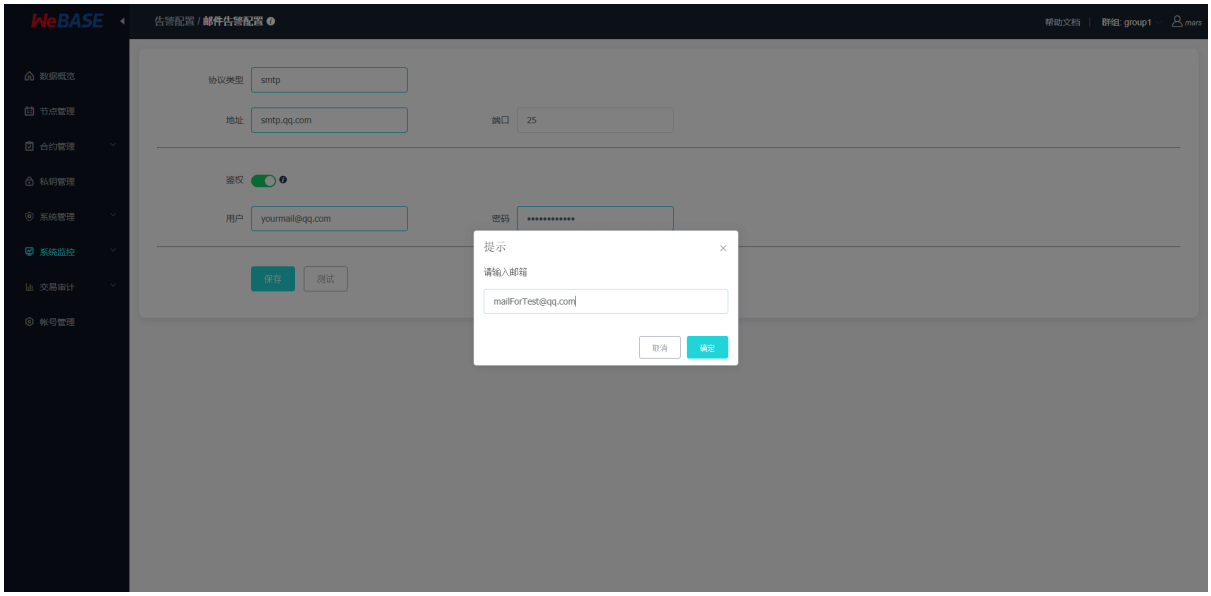
使用测试功能前，需要到“告警类型配置”中，在左上角**开启邮件服务总开关**;

注：邮件告警功能需要确保邮件服务器配置正确；务必使用测试按键，向指定的邮箱地址发送测试邮件并查收邮件。如果配置错误，将发送测试邮件失败，指定邮箱将收不到测试邮件;



邮件服务配置测试:

以当前表单中输入的配置值发送测试邮件（无论是否已保存，都以表单中当前的值为配置发送测试邮件）；**需要提前开启邮件服务开关**;



告警类型配置（告警邮件配置）：

包含了告警类型的配置，告警日志的查看；可配置告警类型的参数值，包含告警邮件标题ruleName，告警邮件内容alertContent，告警邮件发送时间间隔alertIntervalSeconds（单位：秒），上次告警时间lastAlertTime，目标告警邮箱地址userList，是否启用该类型的邮件告警enable，告警等级alertLevel等；

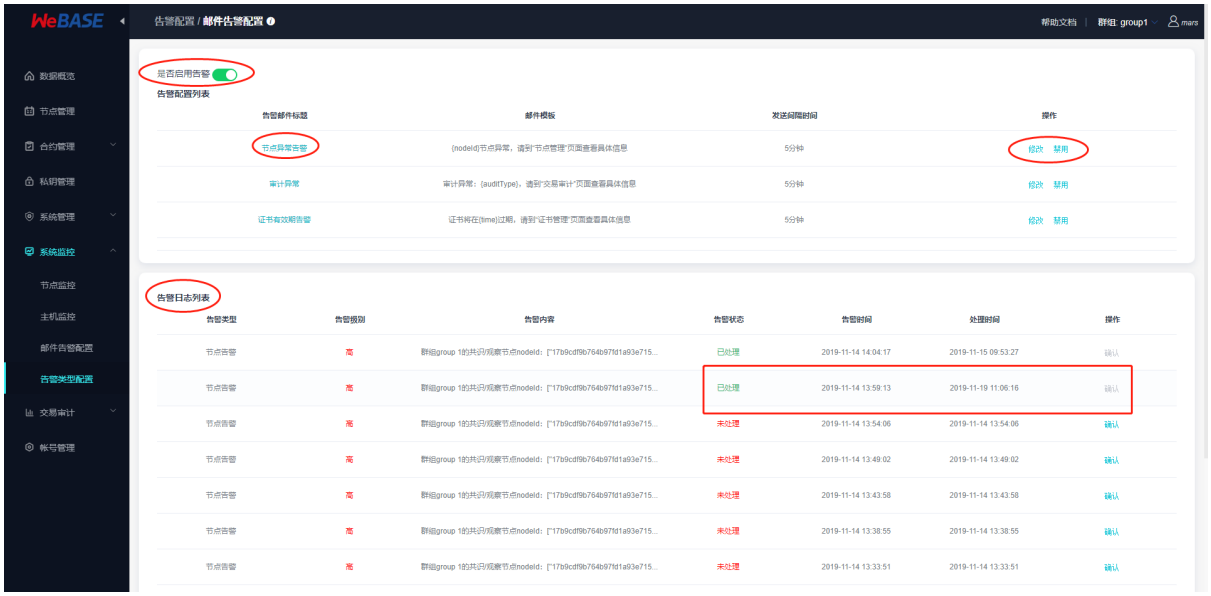
- 包含了节点状态告警、审计告警、证书有效期告警三种；
- 目前仅支持更新原有的三个邮件告警的配置，不支持新增配置；
- 需要先在左上角**开启邮件服务**才可以开启各个类型的邮件告警以及发送测试邮件；

包含了不同告警类型的配置，左上角可以开启邮件服务（作为告警邮件的全局开关），点击告警标题可查看详细配置内容；

下方则是告警日志的内容，可查看告警邮件的具体内容；告警项已处理后，可以点击确认键确认已消除异常；

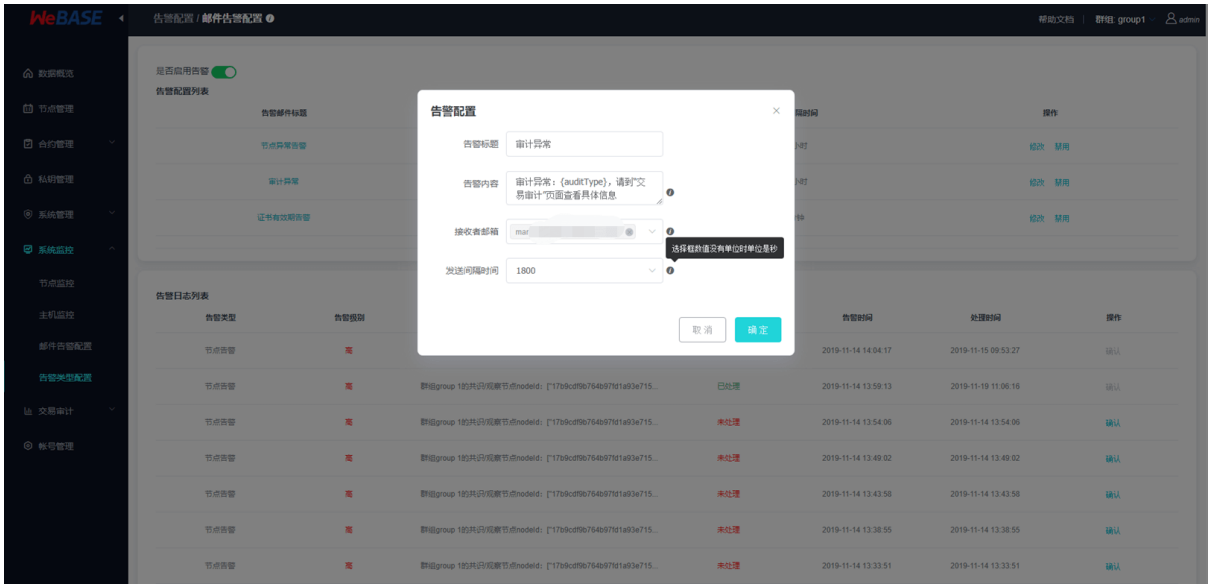
其中在WeBASE-Node-Manager的配置文件application.yml的constant可以配置定时任务定时监控节点状态、审计状态、证书有效期的频率，监控到异常状态时将触发邮件告警，发送告警邮件到联系人邮箱，同时按配置的间隔时间定时重复发送告警邮件，直到异常状态消除；

注：定时任务的频率为检查系统是否异常的频率，而配置不同的告警类型中的告警时间间隔是发送告警邮件的频率，如，设置检查频率为1h，配置的告警频率为6h，那么，系统会每小时检查一次系统状态，若出现异常，在定时任务检查到异常时，距离上次告警邮件超过6小时，则会发送一次告警邮件。



点击修改可以修改配置项的值，启用/禁用不同类型的告警，修改配置后不需要重启即可生效；

注：修改告警内容时，大括号{}以及里面的变量名不可去除，否则无法正常发送告警邮件。

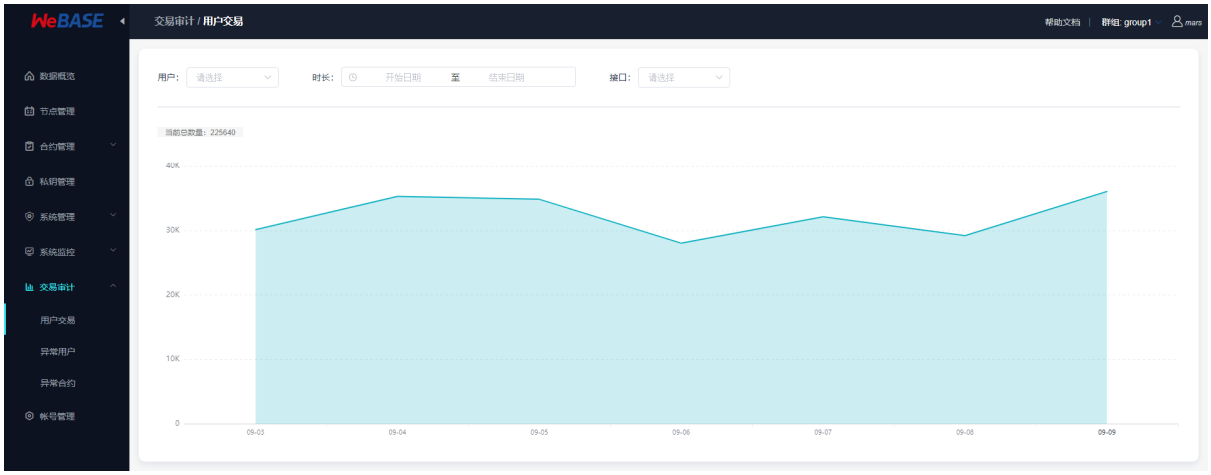


7.4.7 交易审计

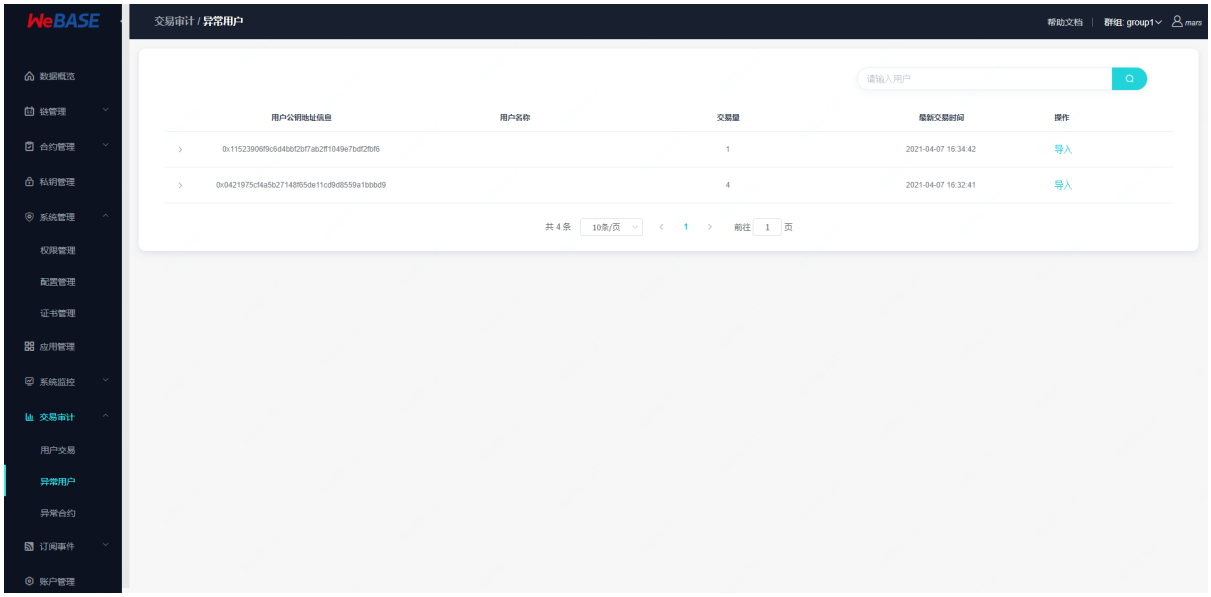
联盟链中各个机构按照联盟链委员会制定的规章在链上共享和流转数据。这些规章往往是字面的，大家是否遵守缺乏监管和审计。因此为了规范大家的使用方式，避免链的计算资源和存储资源被某些机构滥用，急需一套服务来辅助监管和审计链上的行为。交易审计就是结合上面的区块链数据，私钥管理和合约管理三者的数据，以区块链数据为原材料，以私钥管理和合约管理为依据做的一个综合性的数据分析功能。交易审计提供可视化的去中心化合约部署和交易监控、审计功能，方便识别链资源被滥用的情况，为联盟链治理提供依据。

交易审计主要指标：

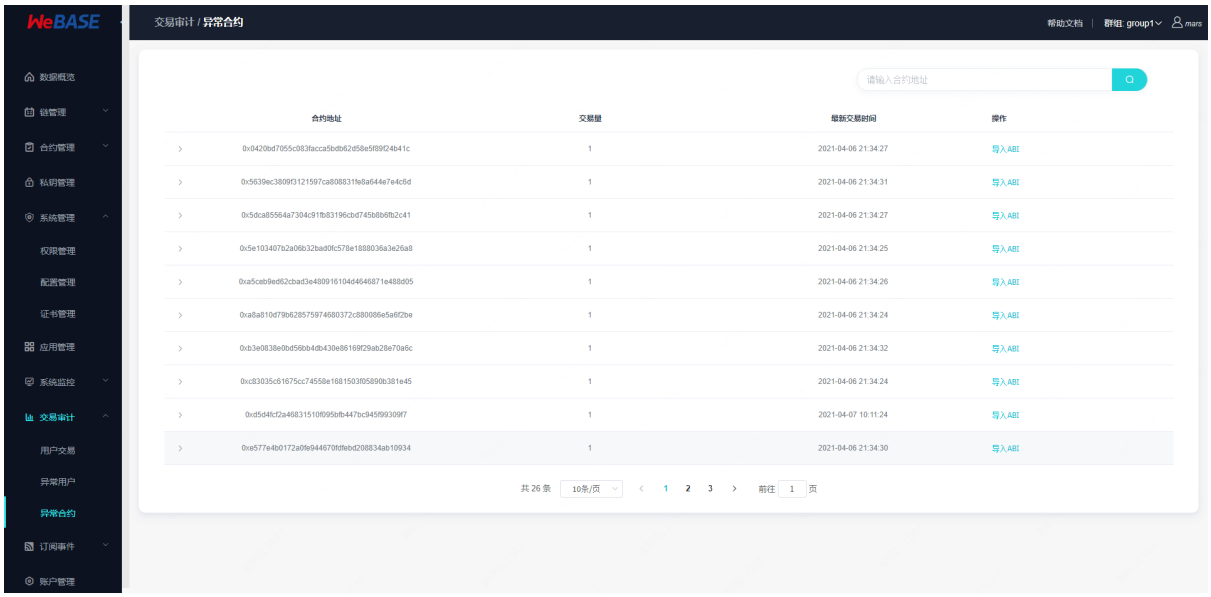
用户交易审计：可以指定用户、时间范围、交易接口进行筛选查看交易



异常用户审计：



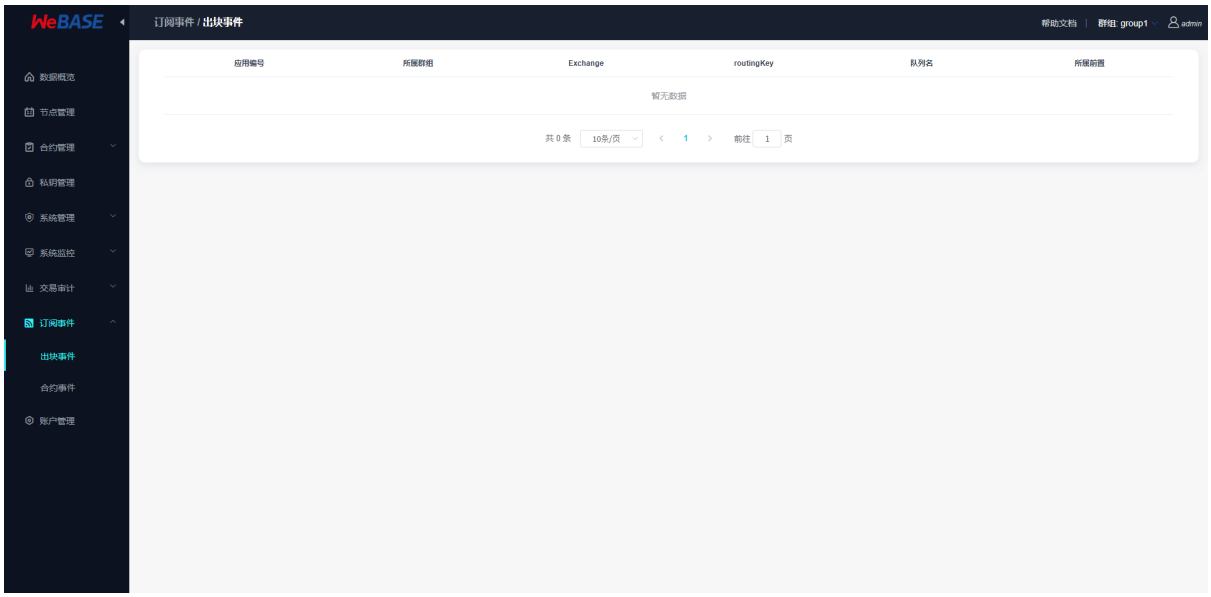
异常合约审计：



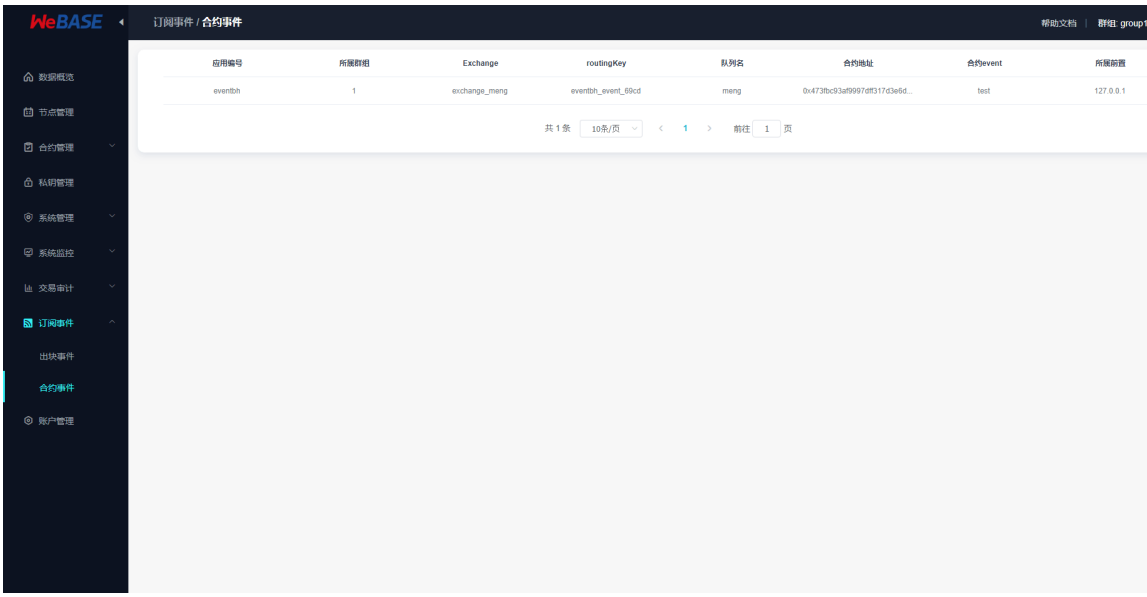
7.4.8 订阅事件

订阅事件管理：可查看前置中已订阅的链上事件通知，包括出块事件列表和合约Event事件列表。详情请参考节点前置-链上事件订阅和通知

出块事件列表：



合约Event事件列表：



7.4.9 账号管理

账号管理提供管理台登陆账号的管理功能。管理台用户分为三种角色：

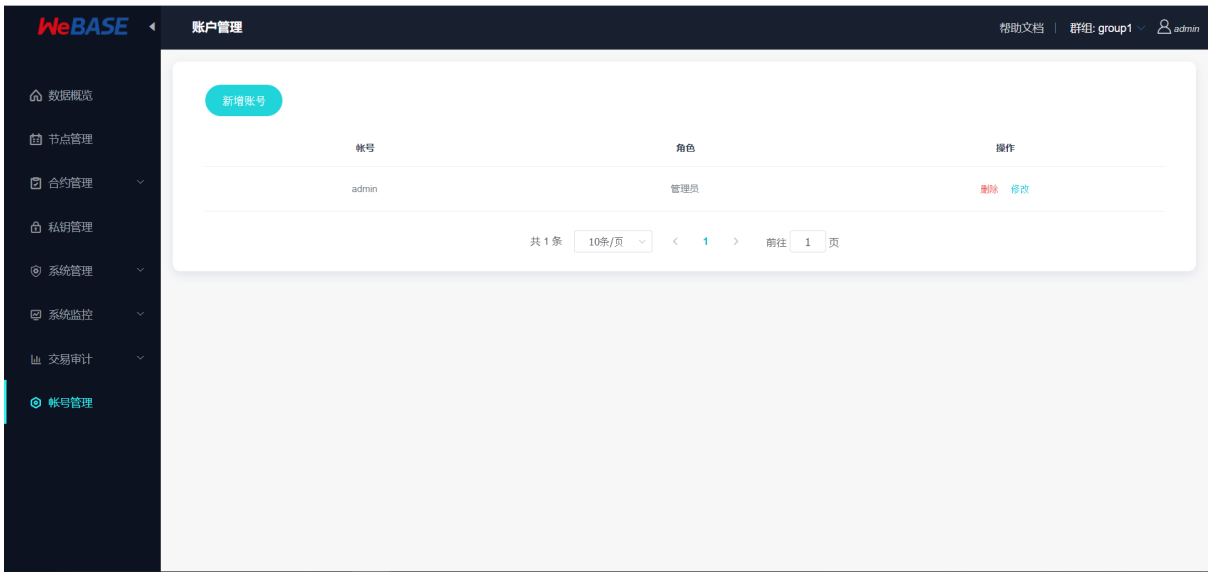
- 普通用户，只有查看权限；
- 管理员用户，拥有管理平台的读写权限；
- 开发者用户，拥有开发者自身的合约和私钥用户的读写权限，数据概览权限；

  开发者模式默认关闭。如需开启此功能，可以在WeBASE-Node-Manager配置文件application.yml中修改developerModeEnable为true，然后重启服务。

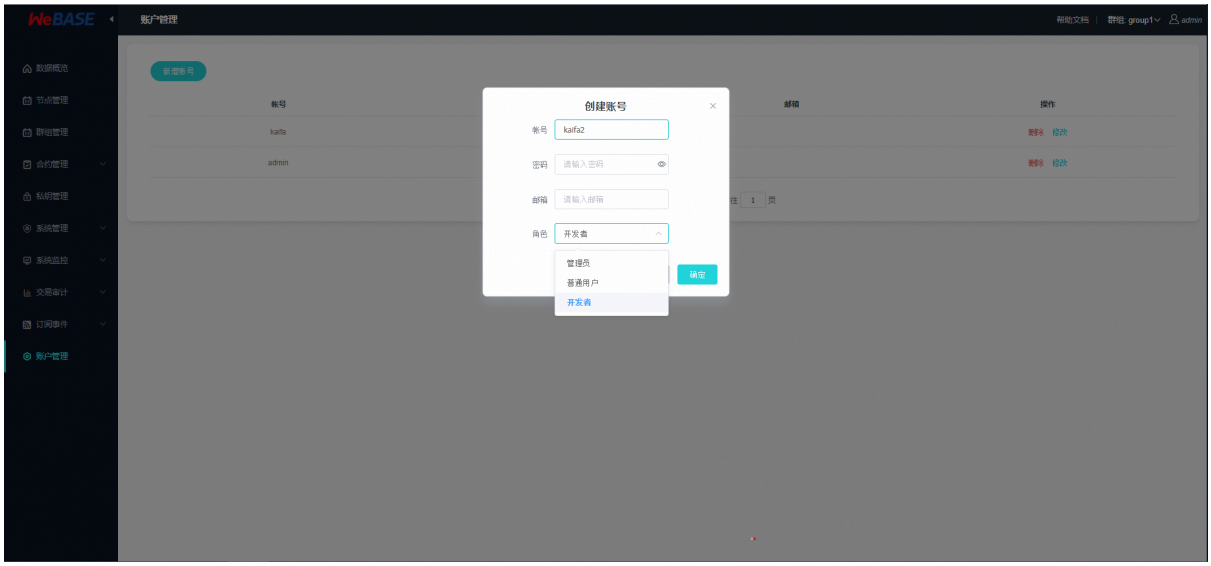
```
###common
developerModeEnable: false
isDeleteInfo: true
transRetainMax: 10000
deleteInfoCron: "0 0/1 * * * ?"
statisticsTransDailyCron: "0 0/1 * * * ?"
resetGroupListCycle: 600000
groupInvalidGrayscaleValue: 1M    # y:year, M:month, d:day of month, h:hour, m:minute, n:forever valid
notSupportFrontIp:
```

注：此处账号与私钥管理的私钥用户为两种不同的概念，账号用于管理台权限控制，私钥用户为区块链账户。

账号管理：



添加登陆账号并指定账号类型：



7.4.10 移动端管理台

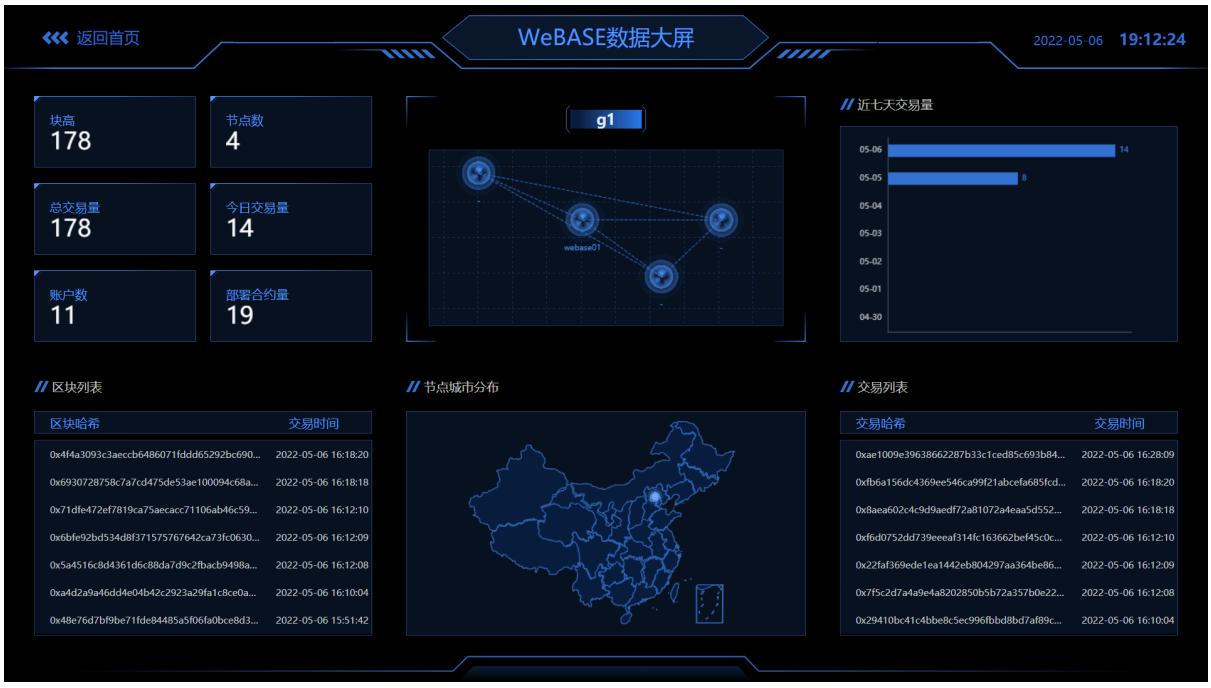
移动端管理台：支持区块链数据概览、链上合约、链上用户、节点列表、区块列表和交易列表的展示

- 在移动端设备访问WeBASE时将自动切换到移动端管理台页面

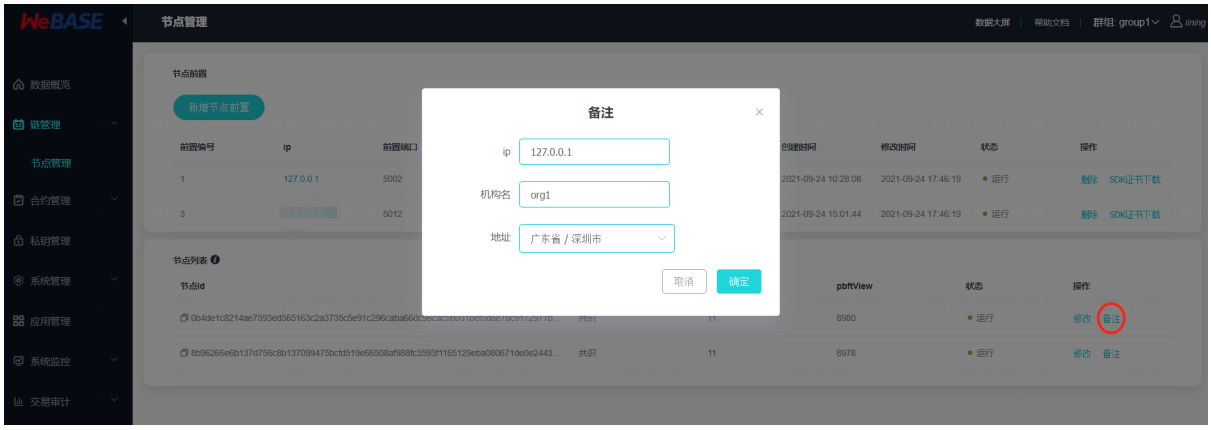
7.4.11 数据监控大屏

数据监控大屏页面的入口位于WeBASE管理台的左上角，点击“数据大屏”即可进入数据监控大屏，适用于企业级控制中心需要全局监控链状态数据的场景。

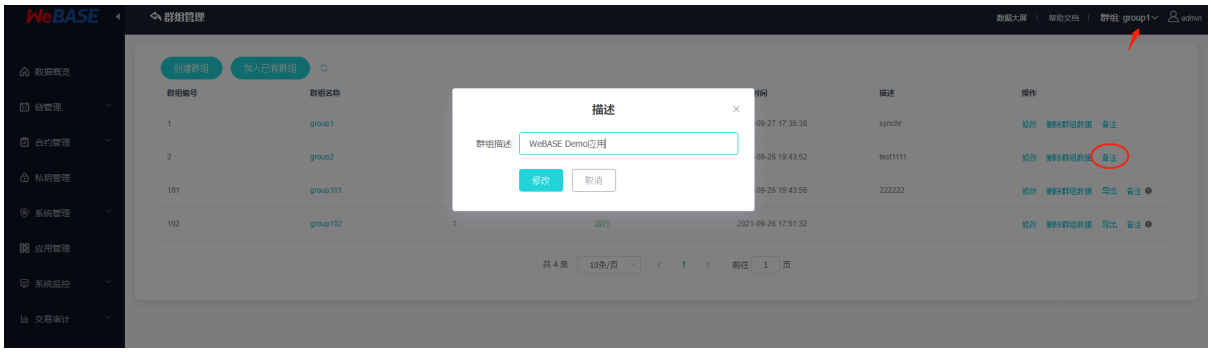
- 数据大屏每次仅展示单个群组的数据，并定时访问后台刷新数据。



在“节点管理”中，可以点击节点列表中的“备注”按钮，为数据大屏中的节点配置IP地址、机构名与城市。



在右上角的“群组管理”中，可以点击群组列表的“备注”按钮，为数据大屏中的群组配置群组应用名（标题）



7.5 升级兼容说明

WeBASE-Front升级至最新版，可查看节点前置升级说明，请结合WeBASE-Front Changelog进行阅读

WeBASE-Node-Manager升级至最新版，可查看节点管理服务升级说明，请结合WeBASE-Node-Manager Changelog进行阅读

WeBASE-Sign升级至最新版，可查看签名服务升级说明，请结合WeBASE-Sign Changelog进行阅读

7.6 附录

7.6.1 配置邮件服务指南

请先阅读本文档中管理平台使用手册的各模块的详细介绍-系统监控-邮件服务配置

问：邮件服务怎么用?

答：在后台搭建邮件服务（邮箱服务器），用于后台监控到系统异常情况时，发送告警邮件到指定邮箱，方便运维；

下面介绍具体的使用方法：

邮件服务所使用的邮箱服务器：

1. 企业可使用自行搭建的邮箱服务器；
2. 普通用户可以使用QQ邮箱、网易邮箱等第三方邮箱；

开通邮箱服务

163邮箱开通邮箱服务：

- 登陆邮箱后，在邮箱的设置中找到包含SMTP的设置项；

- 勾选IMAP/SMTP和POP3/SMTP，初次开启时，会提醒用户设置**授权码**，并进行手机安全验证；
- 设置授权码后，勾选IMAP, POP3, SMTP开启全部服务；

QQ邮箱开通邮箱服务：

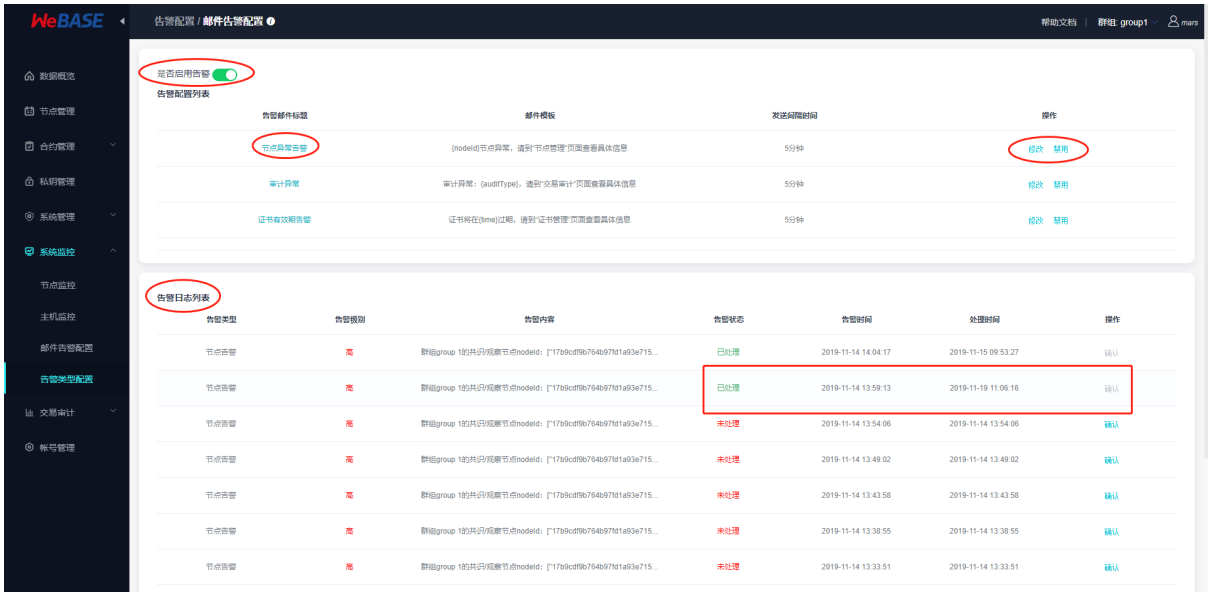
- 登陆邮箱后，在邮箱的设置账户中找到POP3/IMAP/SMTP/Exchange/CardDAV/CalDAV服务项；

- 开启POP3/SMTP服务和IMAP/SMTP服务并按照指引进行手机安全验证并设置授权码，；

配置邮件服务

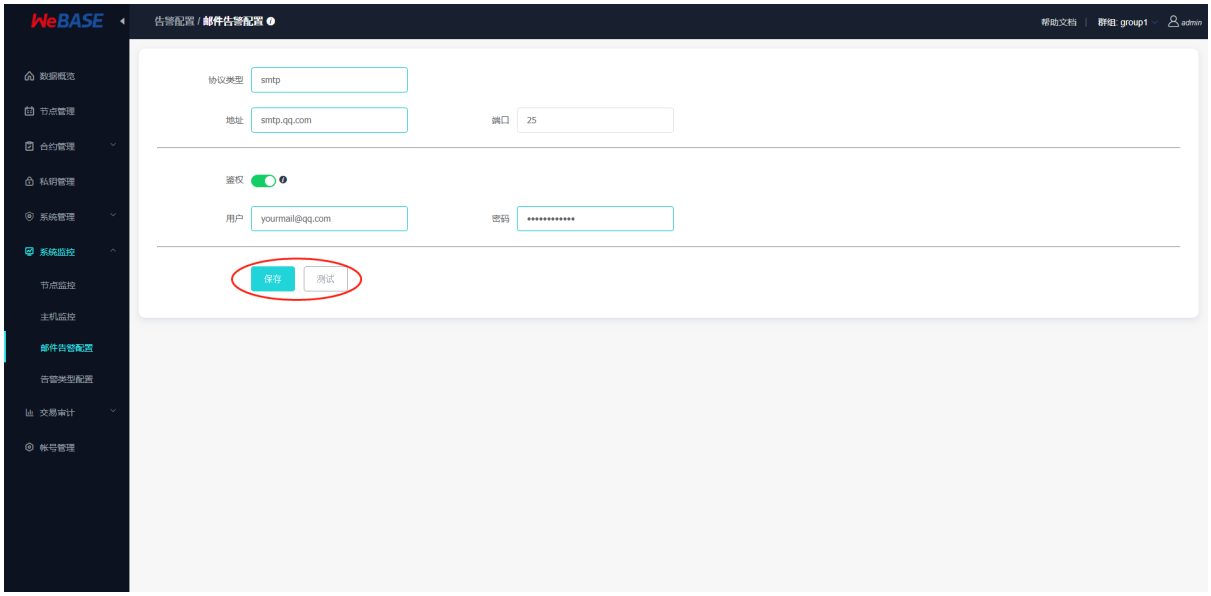
记下所设置的**授权码**，授权码即邮件服务中用到的"密码"，按照本文档各模块介绍-系统监控进行配置:

第一步，进入"告警类型配置"中，点击左上角"启用告警"以**开启邮件服务开关**

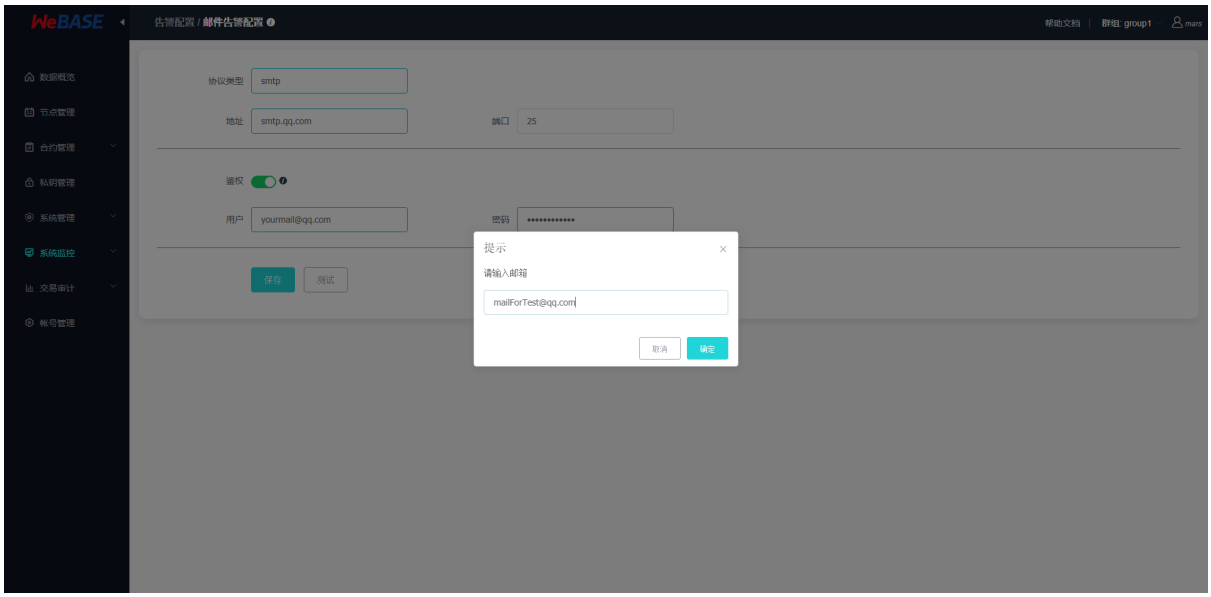


第二步，进入"邮件告警配置"，配置邮件服务

- 因为Node-Manager仅使用邮箱服务器的发件服务，因此**协议类型**填写smtp（IMAP/POP3均为收件服务协议）；
- 邮箱服务器填写smtp.xx.com，端口号默认为25即可启用邮件服务；如需使用其他端口如465则需要开放WeBASE-Node-Manager所在服务器的相应端口限制；
- 用户名填写邮箱地址，密码填写上文设置的**授权码**；



配置完成后，点击“测试”后，输入接收测试邮件的邮箱地址，测试成功即可“保存”邮件服务的配置；



注意事项

“邮件告警配置”中填写的端口默认为25，在不同服务器环境和不同邮箱所需的端口号有所差异，如果需要开启SSL进行邮箱安全验证则需要开通服务器防火墙相应的端口号。

目前已知的包含：

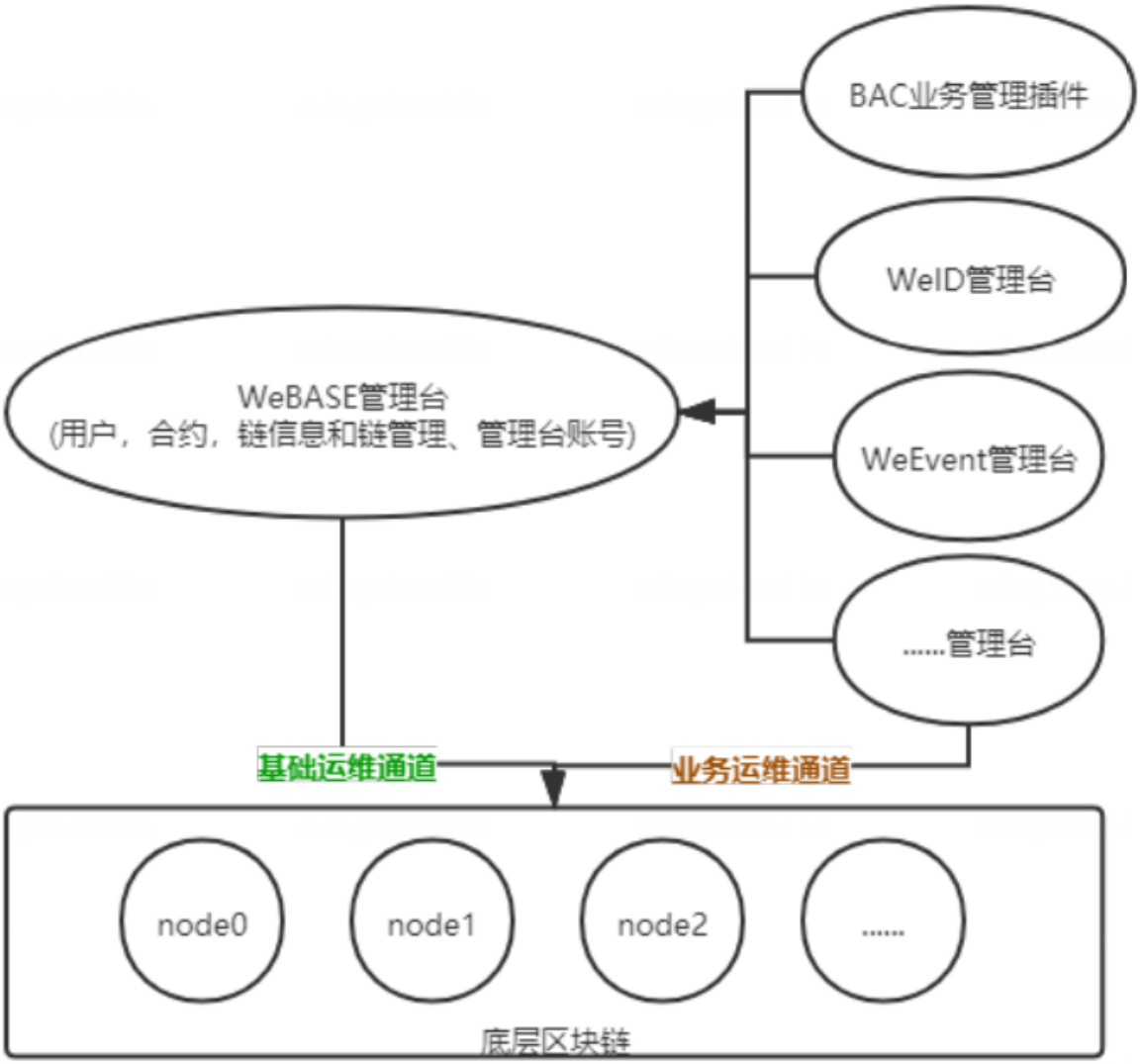
- SMTP协议：默认使用25端口(非SSL)，SSL默认465端口(SSL)或587端口(TLS)
- POP3/IMAP协议：因为邮箱服务使用的是发邮件功能，未用到POP3或IMAP收件协议，此处仅作端口说明：其中POP3默认110端口(非SSL)和995端口(SSL)，IMAP默认143端口(非SSL)和993端口(SSL)
- 126邮箱的SSL端口除了587，还可尝试994；在阿里云下25端口被禁用，请尝试587端口或其他端口；

CHAPTER 8

WeBASE应用管理

8.1 功能介绍

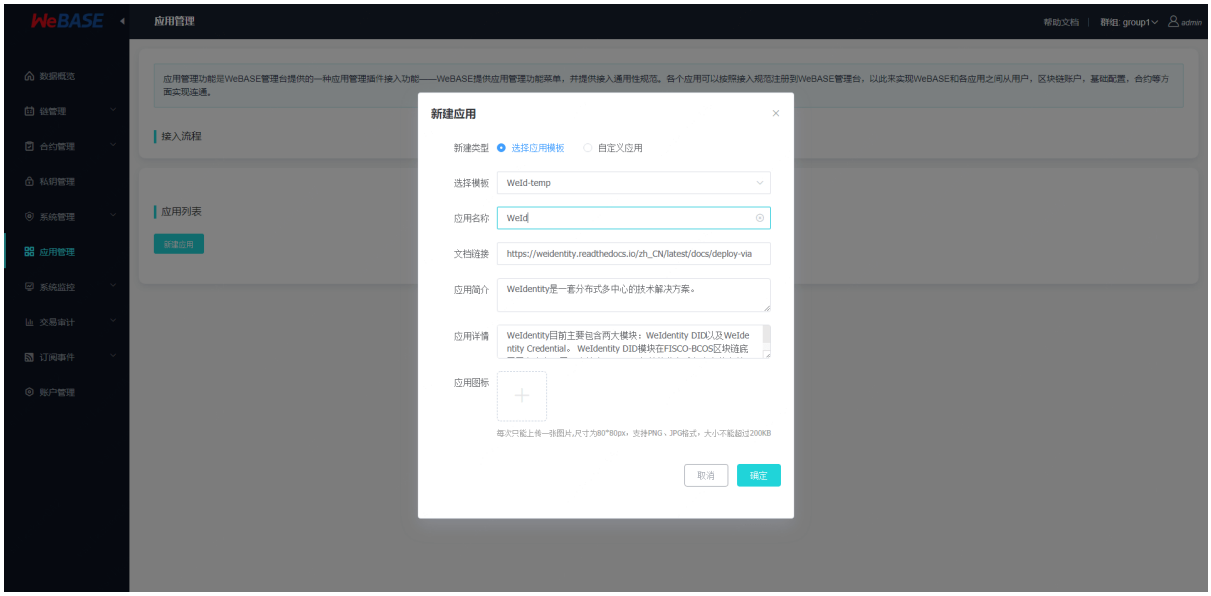
应用管理是WeBASE管理台提供的一种第三方应用接入功能。WeBASE作为底层运维平台，已经有了底层运维基础能力。各个应用可以利用这些基础能力来开发自己的运维管理台。这些可以利用的基础能力主要包括四个方面：1、链信息和链运维（权限，配置等）；2、合约；3、链的私钥账号；4、管理账号（登录态）。



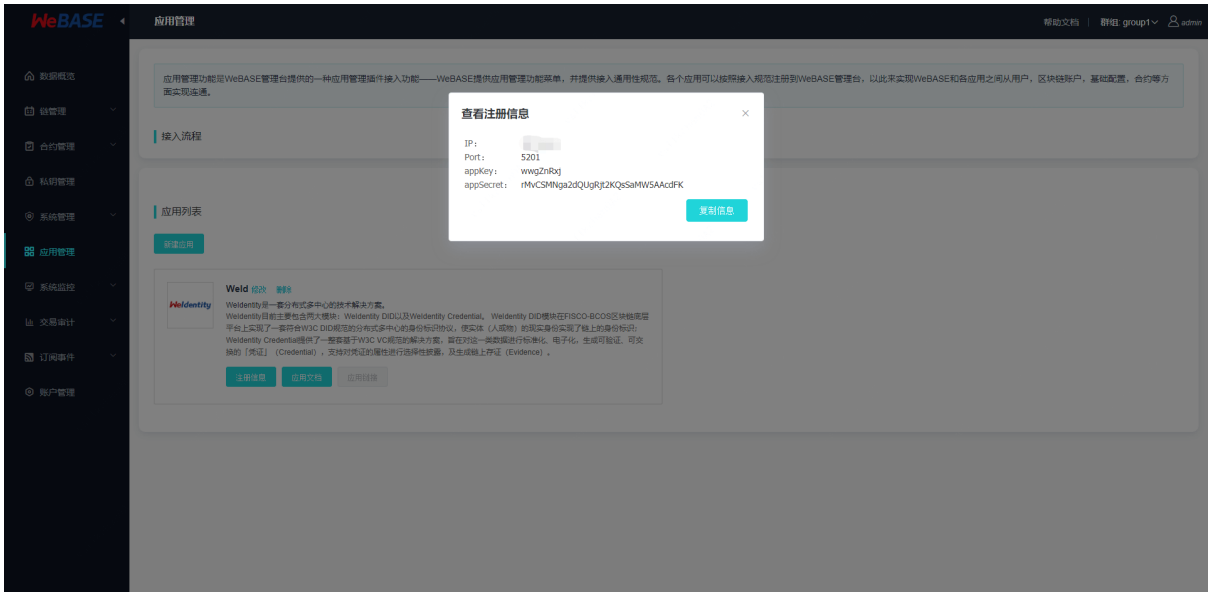
这样的主要好处是：

1. 各应用的进程管理还是自我管理，避免WeBASE过于笨重
2. WeBASE提供的是规范，方便其他应用参考打通应用与WeBASE的联系
3. 如果有扩展，WeBASE也方便提供API来实现
4. 不破坏各应用自身的完整性

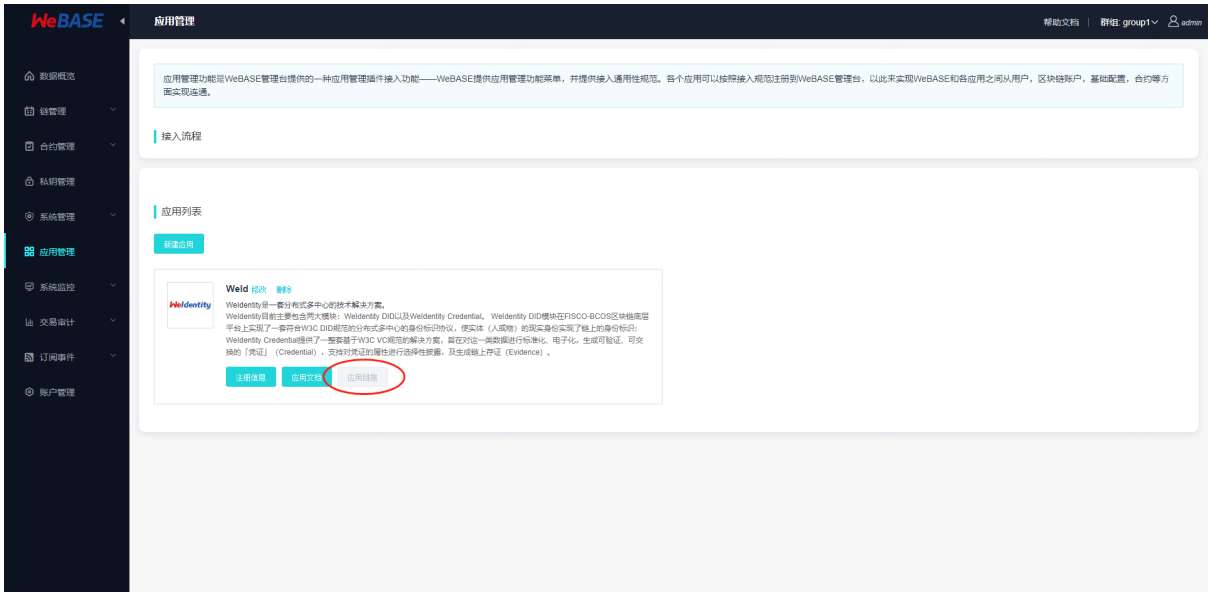
管理台新增了应用管理菜单。新增应用有两种方式，一种是选择已有应用模板——目前仅支持WeID；另外一种是自定应用：



新增应用后，会生成应用相关的注册信息，为应用分配的`appKey`（应用Key）和`appSecret`（应用密码，应用自己保存，不要暴露），WeBASE的`IP`为内网IP，访问不了的话需对应修改：



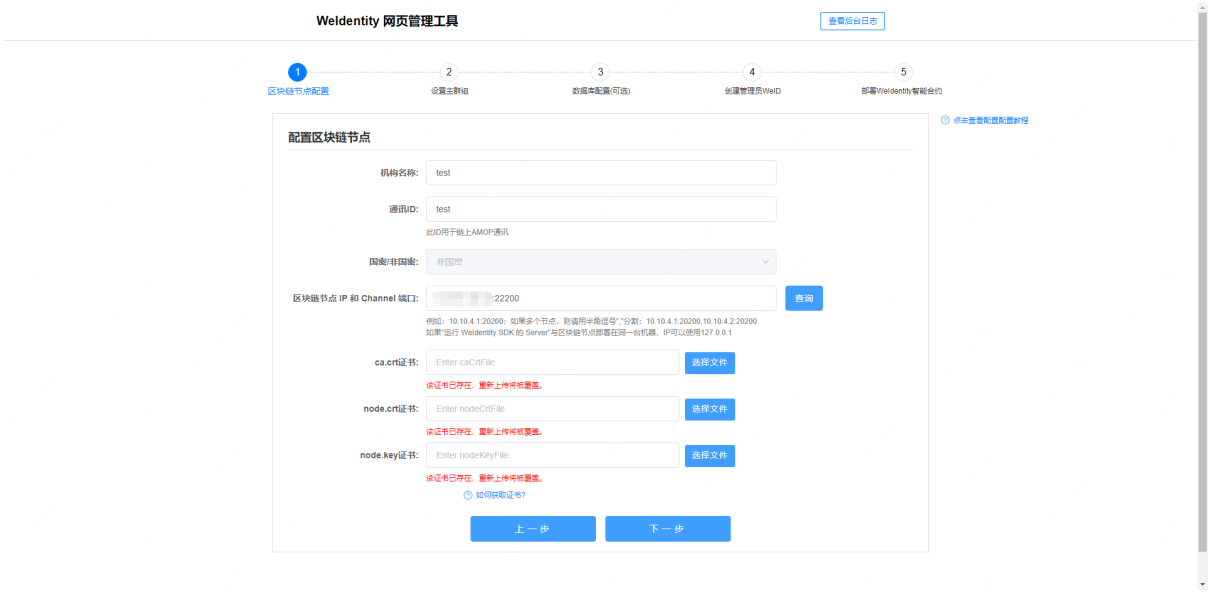
第三方应用未向WeBASE进行注册时，WeBASE管理台不可以通过应用链接跳转到应用服务：



第三方应用接入时，使用注册信息向WeBASE进行注册，调用相关OPEN API。第三方应用接入请参考【应用接入说明】。以下以 WeIdentity 可视化页面接入为例：

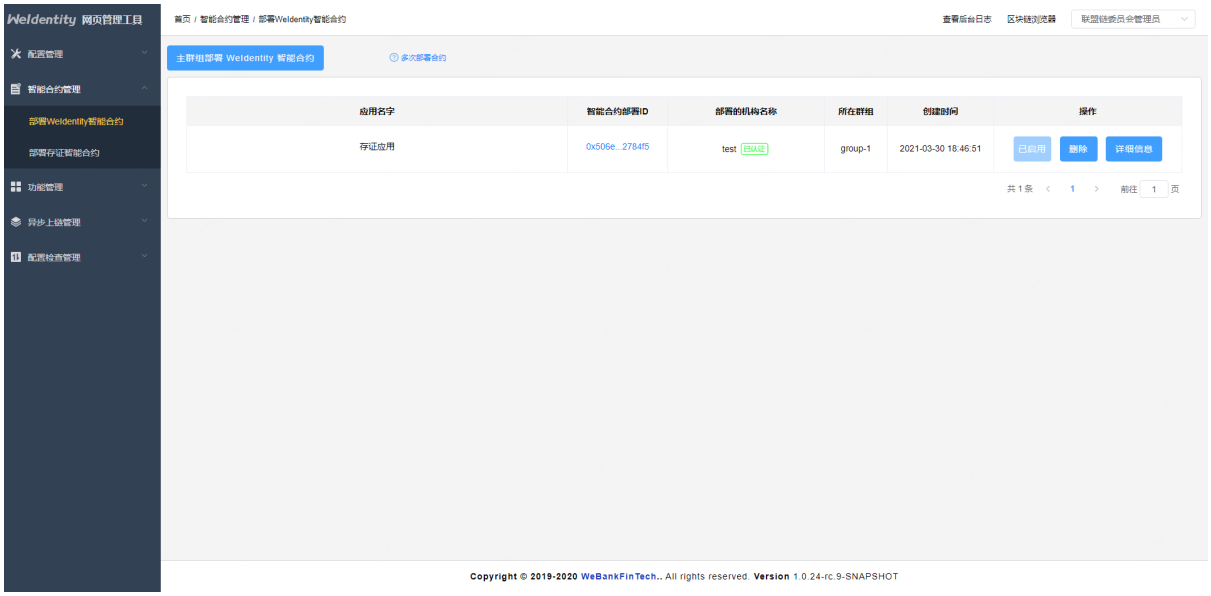
选择 WeID + WeBASE集成模式，从WeBASE复制注册信息：

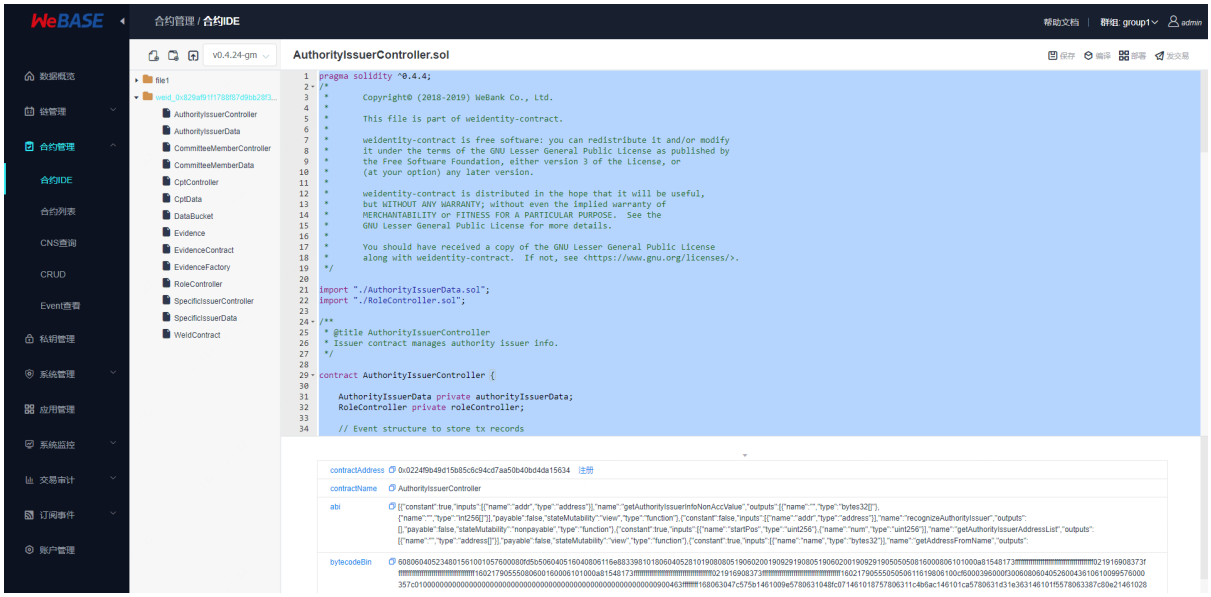
点击下一步时，WeID会向WeBASE进行注册并调用相关API，从WeBASE获取群组、节点、证书等相关信息：



可以选择自动创建公私钥，或选择从WeBASE同步公私钥用户：

部署WeID之后，WeID会将合约相关信息通过API导入WeBASE:

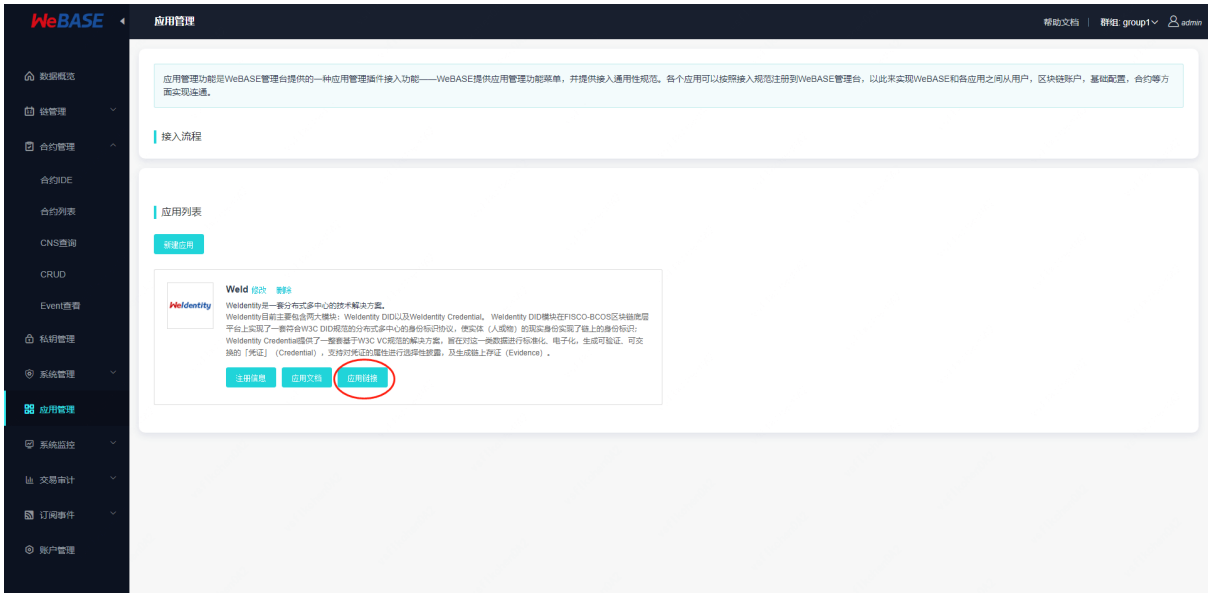




第三方应用向WeBASE进行注册后，在WeBASE管理台可以通过应用链接跳转到应用服务：

注册后WeBASE将和应用间保持心跳。如果应用状态变成不能访问，则应用链接会置灰，变成不可跳

转。



8.2 接入说明

应用管理是WeBASE管理台提供的一种第三方应用接入功能。其他应用可以通过WeBASE通用API来开发自己的运维管理台。接入的步骤如下：

1. 通过WeBASE管理平台获得注册信息，并通过API向WeBASE注册服务。
2. 通过WeBASE提供的基础能力API和WeBASE连通。

8.2.1 应用集成SDK

SDK简介

WeBASE-APP-SDK是应用集成SDK，提供调用WeBASE-Node-Manager的接口，方便WeBASE管理台接入第三方应用。 从`v1.5.1`开始，提供应用集成SDK，方便应用接入。接口API可以查看WeBASE通用API。调用之前需要添加依赖和初始化应用信息。

- `v1.5.1`及其以上版本，应用配置AppConfig的属性`isTransferEncrypt`需和WeBASE-Node-Manager的配置文件`/conf/application.yml`下的配置`constant.isTransferEncrypt`相同，默认为`true`。
- 如果`v1.5.0`需要使用SDK，应用配置AppConfig的属性`isTransferEncrypt`需设置为`false`。`v1.5.1`及其以上版本新增的接口调用不了。
- `1.5.6`以上版本，既支持WeBASE 1.5.x及以上版本，同时支持WeBASE 3.1.0及以上版本（WeBASE 3.0.2版本暂未支持webase-app-sdk）

添加依赖

- 添加 SDK 的依赖，以Gradle为例

```
repositories {
    maven { url "http://maven.aliyun.com/nexus/content/groups/public/" }
}
dependencies {
```

(下页继续)

(续上页)

```
    implementation 'com.webank:webase-app-sdk:1.5.6-SNAPSHOT'
    implementation 'org.bouncycastle:bcprov-jdk15on:1.67'
    implementation 'org.apache.commons:commons-lang3:3.8.1'
    implementation 'com.squareup.okhttp3:okhttp:4.8.1'
    implementation 'com.fasterxml.jackson.core:jackson-databind:2.14.2'
    implementation 'com.fasterxml.jackson.datatype:jackson-datatype-jdk8:2.14.2'
    implementation 'com.fasterxml.jackson.datatype:jackson-datatype-jsr310:2.14.2'
    implementation 'com.fasterxml.jackson.module:jackson-module-parameter-names:2.
↪14.2'
    implementation 'org.projectlombok:lombok:1.18.12'
    annotationProcessor 'org.projectlombok:lombok:1.18.12'
    implementation 'org.apache.logging.log4j:log4j-api:2.20.0'
    implementation 'org.apache.logging.log4j:log4j-core:2.20.0'
    implementation 'org.apache.logging.log4j:log4j-slf4j-impl:2.20.0'
    implementation 'org.slf4j:slf4j-api:1.7.36'
}
```

配置说明

- 应用配置

```
public class AppConfig {
    // 节点管理服务地址
    private String nodeManagerUrl;
    // 应用Key
    private String appKey;
    // 应用密码
    private String appSecret;
    // 是否加密传输
    private boolean isTransferEncrypt;
}
```

- Http请求配置

```
public class HttpConfig {
        // 连接超时（默认30s）
    private int connectTimeout;
    // 读取超时（默认30s）
    private int readTimeout;
    // 写超时（默认30s）
    private int writeTimeout;
}
```

调用示例

完整示例请查看SDK示例。

```
public class ClientTest {

    // WeBASE-Node-Manager的url
    private static String url = "http://localhost:5001";
    private static String appKey = "RUPCNAsd";
    private static String appSecret = "65KiXNxUpPywVwQxM7SFsMHsKmCbpGrQ";
    private static boolean isTransferEncrypt = true;

    private static AppClient appClient = null;

    public static void main(String[] args) {
```

(下页继续)

(续上页)

```
        try {
            initClient();
            appRegister();
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.exit(0);
    }

    public static void initClient() {
        // 未设置httpConfig时，默认http连接均为30s
        HttpConfig httpConfig = new HttpConfig(30, 30, 30);
        appClient = new AppClient(url, appKey, appSecret, isTransferEncrypt,␣
↪httpConfig);
        System.out.println("testInitClient:" + JacksonUtil.objToString(appClient));
    }

    public static void appRegister() throws Exception {
        try {
            ReqAppRegister req = new ReqAppRegister();
            req.setAppIp("127.0.0.1");
            req.setAppPort(5001);
            req.setAppLink("https://127.0.0.1:5001/");
            appClient.appRegister(req);
            System.out.println("appRegister end.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

8.2.2 签名

使用SDK时，SDK会自动填充。

签名说明

第三方应用从WeBASE管理平台获取注册信息**WeBASE的IP和端口、为应用分配的appKey（应用Key）和appSecret（应用密码，应用自己保存，不要暴露）**，向WeBASE发送请求时，需要使用应用分配的appSecret对请求进行签名。WeBASE收到请求后，根据appKey查询应用对应的appSecret，使用相同规则对请求进行签名验证。只有在验证通过后，WeBASE才会对请求进行相应的处理。

- 每个URL请求需带以下三个参数:

签名规则

使用MD5对timestamp、appKey加密并转大写得到签名值signature

```
public static String md5Encrypt(long timestamp, String appKey, String appSecret) {
        try {
            String dataStr = timestamp + appKey + appSecret;
            MessageDigest m = MessageDigest.getInstance("MD5");
            m.update(dataStr.getBytes("UTF8"));
            byte s[] = m.digest();
            String result = "";
            for (int i = 0; i < s.length; i++) {
```

(下页继续)

(续上页)

```
                result += Integer.toHexString((0x000000FF & s[i]) | 0xFFFFFF00).
↪substring(6);
            }
            return result.toUpperCase();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

示例

- 参数值:
- 签名后的 signature 为

```
EEFD7CD030E6B311AA85B053A90E8A31
```

8.3 管理实例

8.3.1 基于区块链的实体身份标识及可信数据交换解决方案

WeIdentity + WeBASE集成模式

git地址：https://github.com/WeBankBlockchain/WeIdentity-Build-Tools

文档地址：https://weidentity.readthedocs.io/zh_CN/latest/docs/deploy-via-web.html

8.3.2 基于FISCO BCOS 从0-1的供应链支付结算案例

FISCO BCOS Supply Chain Payment Settlement Demo created by Shanghai JiuYu Software Systems Co,Ltd.

由上海久誉软件系统有限公司研发的针对基于 FISCO BCOS 的供应链支付结算案例。

git地址：https://github.com/jiuyu-software/supply-chain-demo

文档地址：https://github.com/jiuyu-software/supply-chain-demo/blob/master/README.md

8.3.3 基于 FISCO BCOS实现的电子存证平台案例

由杭州亦笔科技有限公司开发的针对基于 FISCO BCOS 的区块链电子存证平台案例。

git地址：https://github.com/YibiOpen/evidence-chain-demo

文档地址：https://github.com/YibiOpen/evidence-chain-demo/blob/master/README.md

8.4 接口说明

8.4.1 1 应用管理模块

1.1 应用注册

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/appRegister?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：POST
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/appRegister?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31
```

```
{
  "appIp": "127.0.0.1",
  "appPort": 8080,
  "appLink": "http://127.0.0.1:8080/sample"
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "Success",
    "data": {}
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

8.4.2 2 系统账号信息模块

2.1 查询帐号列表

查询系统登录账号列表，不返回密码

传输协议规范

- 网络传输协议：使用HTTP协议

- 请求地址：**/api/accountList?appKey={appKey}&signature={signature}&timestamp={timestamp}&pageNumber={pa**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/accountList?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31&pageNumber=1&
↪pageSize=10&account=
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "account": "testAccount",
            "roleId": 100001,
            "roleName": "visitor",
            "roleNameZh": "访客",
            "loginFailTime": 0,
            "accountStatus": 1,
            "description": null,
            "createTime": "2019-03-04 15:11:44",
            "modifyTime": "2019-03-04 15:18:47"
        },
        {
            "account": "admin",
            "roleId": 100000,
            "roleName": "admin",
            "roleNameZh": "管理员",
            "loginFailTime": 0,
            "accountStatus": 2,
            "description": null,
            "createTime": "2019-02-14 17:33:50",
            "modifyTime": "2019-02-14 17:45:53"
        }
    ],
    "totalCount": 2
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

2.2 查询角色列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**api/roleList?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/api/roleList?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "totalCount": 2,
    "data": [
        {
            "roleId": 100000,
            "roleName": "admin",
            "roleNameZh": "管理员",
            "roleStatus": 1,
            "description": null,
            "createTime": "2019-02-14 17:33:50",
            "modifyTime": "2019-02-14 17:33:50"
        },
        {
            "roleId": 100001,
            "roleName": "visitor",
            "roleNameZh": "访客",
            "roleStatus": 1,
            "description": null,
            "createTime": "2019-02-14 17:33:50",
            "modifyTime": "2019-02-14 17:33:50"
        }
    ]
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
```

(下页继续)

(续上页)

```
    "data": {}
}
```

2.3 新增帐号

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/accountAdd?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/accountAdd?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31
```

```
{
    "account": "testAccount",
    "accountPwd": "3f21a8490cef2bfb60a9702e9d2ddb7a805c9bd1a263557dfd51a7d0e9dfa93e
↪",
    "roleId": 100001,
    "email": "test@xxx.com"
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "account": "testAccount",
        "roleId": 100001,
        "roleName": "visitor",
        "roleNameZh": "访客",
        "loginFailTime": 0,
        "accountStatus": 1,
        "description": null,
            "email": "test@xxx.com",
        "createTime": "2019-03-04 15:11:44",
        "modifyTime": "2019-03-04 15:11:44"
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

2.4 更新密码

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/passwordUpdate?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/passwordUpdate?
↪timestamp=1614928857832&appKey=fdsf78aW&
↪signature=EEFD7CD030E6B311AA85B053A90E8A31
```

```
{
    "account": "admin ",
    "oldAccountPwd":
↪"dfdfgdg490cef2bfb60a9702erd2ddb7a805c9bd1arrrewefd51a7d0etttfa93e",
    "newAccountPwd":
↪"3f21a8490cef2bfb60a9702e9d2ddb7a805c9bd1a263557dfd51a7d0e9dfa93e"
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success"
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

8.4.3 3 链信息模块

3.1 查询基本信息

获取链基本信息，包括链版本、是否使用国密、是否使用国密SSL连接、WeBASE版本等。

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/api/basicInfo?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/api/basicInfo?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "encryptType": 1,
        "sslCryptoType": 1,
        "fiscoBcosVersion": "2.7.2 gm",
        "webaseVersion": "v1.5.0"
    },
    "attachment": null
}
```

3.2 查询群组列表

默认只返回groupStatus为1的群组ID，可传入groupStatus筛选群组 (1-normal, 2-maintaining, 3-conflict-genesisi, 4-conflict-data)

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/groupList?appKey={appKey}&signature={signature}&timestamp={timestamp}&groupStatus={grou**

- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/groupList?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31&groupStatus=
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "totalCount": 1,
    "data": [
        {
            "groupId":1,
            "groupName":"group1",
            "groupStatus":1,
            "nodeCount":4,
            "latestBlock":0,
            "transCount":0,
            "createTime":"2020-05-07 16:32:02",
            "modifyTime":"2020-05-08 10:50:13",
            "description":"synchronous",
            "groupType":1
        }
    ]
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

3.3 查询所有节点列表

查询所有节点列表，用于获取所有节点信息，包括自动同步的节点信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/nodeList?appKey={appKey}&signature={signature}&timestamp={timestamp}&pageNumber={pageN**

- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/nodeList?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31&pageNumber=1&
↪pageSize=10
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "totalCount": 1,
    "data": [
        {
            "nodeId":
↪"5942fe2460c1a6329b8ebf5bc5a9ca9bd02ee944e8dac297742982de5920e7211489950ac4ac5eb1bd7219dcd773f8
↪",
            "nodeName": "127.0.0.1_10303",
            "groupId": 1,
            "nodeIp": "127.0.0.1",
            "p2pPort": 10303,
            "description": null,
            "blockNumber": 133,
            "pbftView": 5852,
            "nodeActive": 1,
            "createTime": "2019-02-14 17:47:00",
            "modifyTime": "2019-03-15 11:14:29"
        }
    ]
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

3.4 查询节点信息

查询具体某个节点信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/nodeInfo?appKey={appKey}&signature={signature}&timestamp={timestamp}&groupId={groupId}&**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/nodeInfo?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31&groupId=1&
↪nodeId=5942fe2460c1a6329b8ebf5bc5a9ca9bd02ee944e8dac297742982de5920e7211489950ac4ac5eb1bd7219dc
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "nodeId":
↪"5942fe2460c1a6329b8ebf5bc5a9ca9bd02ee944e8dac297742982de5920e7211489950ac4ac5eb1bd7219dcd773f8
↪",
        "nodeName": "127.0.0.1_10303",
        "groupId": 1,
        "nodeIp": "127.0.0.1",
        "p2pPort": 10303,
        "description": null,
        "blockNumber": 133,
        "pbftView": 5852,
        "nodeActive": 1,
        "createTime": "2019-02-14 17:47:00",
        "modifyTime": "2019-03-15 11:14:29"
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

3.5 查询前置节点列表

查询前置及其对应节点信息列表，包括前置ip（对应节点Ip），节点端口等。

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/frontNodeList?appKey={appKey}&signature={signature}&timestamp={timestamp}&groupId={grou**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/frontNodeList?
↪timestamp=1614928857832&appKey=fdsf78aW&
↪signature=EEFD7CD030E6B311AA85B053A90E8A31
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "frontId": 500001,
            "frontIp": "127.0.0.1",
            "frontPort": 5002,
            "agency": "aa",
            "groupList": [
                1,
                2
            ],
            "frontVersion": "v1.4.0",
            "signVersion": "v1.4.0",
            "clientVersion": "2.5.0 gm",
            "supportVersion": "2.5.0",
            "createTime": "2019-06-04 20:49:42",
            "modifyTime": "2019-06-04 20:49:42",
            "status": 1,
            "runType": 1,
            "agencyId": 1,
            "agencyName": "AgencyA",
            "hostId": 1,
            "hostIndex": 0,
            "imageTag": "v2.5.0",
            "containerName": "rootfisconode0",
            "jsonrpcPort": 8545,
            "p2pPort": 30300,
            "channelPort": 20200,
            "chainId": 1,
            "chainName": "default_chain"
```

(下页继续)

(续上页)

```
        }
    ],
    "totalCount": 1
}
```

- 失败：

```
{
   "code": 102000,
   "message": "system exception",
   "data": {}
}
```

3.6 查询sdk证书信息

返回sdk证书文件名和内容，根据需要保存。根据底层是否使用国密和是否使用国密SSL连接，自动识别返回国密还是非国密证书。

- 非国密证书：非国密链或国密链未使用国密SSL连接
- 国密证书：国密链且使用国密SSL连接

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/sdkCert?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/sdkCert?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "name": "sdk.key",
            "content": "-----BEGIN PRIVATE KEY-----
↪\nMIGEAgEAMBAGByqGSM49AgEGBSuBBAAKBG0wawIBAQQg4mtXDYHIhSPJwmy+CBVZ\nfZM7cpSimSxTKsbtPEWqYZKhRAN
↪----END PRIVATE KEY-----\n"
```

(下页继续)

(续上页)


```
        },
        {
            "name": "ca.crt",
            "content": "-----BEGIN CERTIFICATE-----
↪\nMIIBsTCCAVagAwIBAgIJAM+BzxI9PhQhMAoGCCqGSM49BAMCMDUxDjAMBgNVBAMM\nBWNoYWluMRMwEQYDVQQKDApmaXN
↪qfFlIGIdOj8vmOzywsqaNQME4wHQYDVR0OBBYEFPd7x8U8\nnHjZ5YWuzyKQh9SZ7p5bMB8GA1UdIwQYMBaAFPd7x8U8nHj
↪jJna1+nPKBVB+lxnfK3TpN/QH4V0sS1VW\nkpOFMVU=\n-----END CERTIFICATE-----\n"
        },
        {
            "name": "sdk.crt",
            "content": "-----BEGIN CERTIFICATE-----
↪\nMIIBeTCCAR+gAwIBAgIJAPYOLyl8weZiMAoGCCqGSM49BAMCMDgxEDAOBgNVBAMM\nB2FnZW5jeUExEzARBgNVBAoMCmZ
↪Pg28vUh1OycM6m58SB0uYWHCPi\n-----END CERTIFICATE-----\n-----BEGIN CERTIFICATE----
↪-
↪\nMIIBcTCCARegAwIBAgIJAKA09HPrLxBIMAoGCCqGSM49BAMCMDUxDjAMBgNVBAMM\nBWNoYWluMRMwEQYDVQQKDApmaXN
↪----END CERTIFICATE-----\n-----BEGIN CERTIFICATE-----
↪\nMIIBsTCCAVagAwIBAgIJAM+BzxI9PhQhMAoGCCqGSM49BAMCMDUxDjAMBgNVBAMM\nBWNoYWluMRMwEQYDVQQKDApmaXN
↪qfFlIGIdOj8vmOzywsqaNQME4wHQYDVR0OBBYEFPd7x8U8\nnHjZ5YWuzyKQh9SZ7p5bMB8GA1UdIwQYMBaAFPd7x8U8nHj
↪jJna1+nPKBVB+lxnfK3TpN/QH4V0sS1VW\nkpOFMVU=\n-----END CERTIFICATE-----\n"
        }
    ],
    "attachment": null
}
```


- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

8.4.4 4 私钥管理模块

说明：私钥传输时以Base64加密传输，示例如下：

```
String hexPrivateKey = CryptoSuite.createKeyPair().getHexPrivateKey();
// 加密
String privateKey = Base64.getEncoder().encodeToString(hexPrivateKey.getBytes());
// 解密
hexPrivateKey = new String(Base64.getDecoder().decode(privateKey));
```

4.1 新增私钥用户

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/newUser?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/newUser?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31
```

```
{"groupId":"1","userName":"bob","account":"admin","description":""}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "userId": 700001,
        "userName": "bob",
        "account": "admin",
        "groupId": 1,
        "publicKey":
↪"04d7e3391db028aba558e7111702b42d90db9fe3b3b47f19a9dd9ff3b7fd216e9f19adc4643b40c86cab266edbf947
↪",
        "privateKey":
↪"OGNlZDA1OTUyM2NmYjgzYThjYjk1MmMxMTM1YmQwZDViNzAzNmZkNWM2ZDExNmFhMjMwZWZkNGFmYzcwMWVjMQ==
↪",
        "userStatus": 1,
        "chainIndex": null,
        "userType": 1,
        "address": "0xc1c7f7e5916d1cf787924128bca0cb5b34622952",
        "signUserId": "6c804064630e46f587de3d905e82295a",
        "appId": "1",
        "hasPk": 1,
        "description": "",
        "createTime": "2021-03-07 16:52:29",
        "modifyTime": "2021-03-07 16:52:29"
    },
    "attachment": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

4.2 查询用户列表

返回用户信息列表，不返回私钥内容，查询私钥需调用4.3。

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/userList?appKey={appKey}&signature={signature}&timestamp={timestamp}&pageNumber={pageN**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/userList?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31&pageNumber=1&
↪pageSize=10&groupId=1
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "userId": 700001,
            "userName": "bob",
            "account": "admin",
            "groupId": 1,
            "publicKey":
↪"04d7e3391db028aba558e7111702b42d90db9fe3b3b47f19a9dd9ff3b7fd216e9f19adc4643b40c86cab266edbf947
↪",
            "privateKey": null,
            "userStatus": 1,
            "chainIndex": null,
            "userType": 1,
            "address": "0xc1c7f7e5916d1cf787924128bca0cb5b34622952",
            "signUserId": "6c804064630e46f587de3d905e82295a",
            "appId": "1",
            "hasPk": 1,
            "description": "",
            "createTime": "2021-03-07 16:52:29",
            "modifyTime": "2021-03-07 16:52:29"
        }
    ],
    "totalCount": 1
}
```

- 失败：

```
{
    "code": 102000,
```

(下页继续)

(续上页)

```
    "message": "system exception",
    "data": {}
}
```

4.3 查询用户信息

返回具体用户信息，包含私钥内容（Base64加密后的私钥内容）。

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/userInfo?appKey={appKey}&signature={signature}&timestamp={timestamp}&userId={userId}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/userInfo?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31&userId=700001
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "userId": 700001,
        "userName": "bob",
        "account": "admin",
        "groupId": 1,
        "publicKey":
↪"04d7e3391db028aba558e7111702b42d90db9fe3b3b47f19a9dd9ff3b7fd216e9f19adc4643b40c86cab266edbf947
↪",
        "privateKey":
↪"OGNlZDA1OTUyM2NmYjgzYThjYjk1MmMxMTM1YmQwZDViNzAzNmZkNWM2ZDExNmFhMjMwZWZkNGFmYzcwMWVjMQ==
↪",
        "userStatus": 1,
        "chainIndex": null,
        "userType": 1,
        "address": "0xc1c7f7e5916d1cf787924128bca0cb5b34622952",
        "signUserId": "6c804064630e46f587de3d905e82295a",
        "appId": "1",
        "hasPk": 1,
```

(下页继续)

(续上页)


```
        "description": "",
        "createTime": "2021-03-07 16:52:29",
        "modifyTime": "2021-03-07 16:52:29"
    },
    "attachment": null
}
```


- 失败：


```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```


4.4 导入私钥用户

导入私钥，其中私钥字段用Base64加密

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/importPrivateKey?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/importPrivateKey?
↪timestamp=1614928857832&appKey=fdsf78aW&
↪signature=EEFD7CD030E6B311AA85B053A90E8A31
```


```
{
    "privateKey":
↪"Yjk4YzM3Y2EzNTMxMzNiOWI2MWUwOTMxODhmOTk2NTc2MGYxMTBhMTljNTI2MmY3NTczMDVkNThlOGM3ZWNlMA==
↪",
    "groupId": 1,
    "description": "test",
    "userName": "alice",
    "account": "admin"
}
```


返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success"
}
```

- 失败：

```
{
    "code": 201031,
    "message": "privateKey decode fail",
    "data": null
}
```

4.5 导入.pem私钥

可导入控制台所生成的私钥.pem文件

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/importPem?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/importPem?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31
```

```
{
    "pemContent":"-----BEGIN PRIVATE KEY-----
↪\nMIGEAgEAMBAGByqGSM49AgEGBSuBBAAKBG0wawIBAQQgC8TbvFSMA9y3CghFt51/
↪\nXmExewlioX99veYHOV7dTvOhRANCAASZtMhCTcaedNP+H7iljbTIqXOFM6qm5aVs\nfM/
↪yuDBK2MRfFbfnOYVTNKyOSnmkY+xBfCR8Q86wcsQm9NZpkmFK\n-----END PRIVATE KEY-----\n",
    "groupId": "1",
    "description": "test",
    "userName": "user2",
    "account": "admin"
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success"
}
```

- 失败：

```
{
    "code": 201232,
    "message": "Pem file format error, must surrounded by -----XXXXX PRIVATE KEY-----",
    "data": null
}
```

4.6 导入.p12私钥

可导入控制台生成的私钥.p12文件

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/importP12?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：POST
- 请求头：Content-type: **form-data**
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/importP12?timestamp=1614928857832&appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31
```

使用form-data传参

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success"
}
```

- 失败：（p12文件的密码错误）

```
{
    "code": 201236,
    "message": "p12's password not match",
    "data": null
}
```

4.7 导入公钥用户

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/importPublicKey**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/importPublicKey?
↪timestamp=1614928857832&appKey=fdsf78aW&
↪signature=EEFD7CD030E6B311AA85B053A90E8A31
```

```
{
    "userName": "rose",
    "publicKey":
↪"0x98c4e9896dfa062c7555ede0f1509bda90668902ee9a3b382a3941869d3d69026ece966e1afe9f9de41c2e762750c
↪",
    "groupId": 1,
    "description": "test",
    "account": "admin"
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "userId": 700003,
        "userName": "rose",
        "account": "admin",
        "groupId": 1,
        "publicKey":
↪"0x98c4e9896dfa062c7555ede0f1509bda90668902ee9a3b382a3941869d3d69026ece966e1afe9f9de41c2e762750c
↪",
```

(下页继续)

(续上页)


```
        "privateKey": null,
        "userStatus": 1,
        "chainIndex": null,
        "userType": 1,
        "address": "0xd5eefc7e9df47f17ee8da8639078ac5da934a782",
        "signUserId": null,
        "appId": null,
        "hasPk": 2,
        "description": "sdfa",
        "createTime": "2021-03-12 18:39:39",
        "modifyTime": "2021-03-12 18:39:39"
    },
    "attachment": null
}
```


- 失败：


```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```


8.4.5 5 合约管理模块

5.1 合约原文保存接口

导入应用合约原文信息，按版本导入，WeBASE会存储到应用合约仓库表。当应用调用合约地址保存接口（5.2）时，会从合约仓库表获取对应合约保存到合约表，在WeBASE管理平台展示。

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/contractSourceSave?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/contractSourceSave?
↪timestamp=1614928857832&appKey=fdsf78aW&
↪signature=EEFD7CD030E6B311AA85B053A90E8A31
```


```
{
        "contractList": [
                {
        "contractName": "HelloWorld",
        "contractSource":
↪"cHJhZ21hIHNvbGlkaXR5IF4wLjQuMjsNCmNvbnRyYWN0IEhlbGxvV29ybGR7DQogICAgc3RyaW5nIG5hbWU7DQogICAgZX
↪",
```


(下页继续)

(续上页)


```
        "contractAbi": "[{\"constant\":false,\"inputs\":[{\"name\":\"n\",\"type\":\
↪"string\"}],\"name\":\"set\",\"outputs\":[],\"payable\":false,\"stateMutability\
↪":\"nonpayable\",\"type\":\"function\"},{\"constant\":true,\"inputs\":[],\"name\
↪":\"get\",\"outputs\":[{\"name\":\"\",\"type\":\"string\"}],\"payable\":false,\
↪"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\":true,\"inputs\
↪":[{\"name\":\"v\",\"type\":\"bytes32[]\"}],\"name\":\"get\",\"outputs\":[{\
↪"name\":\"\",\"type\":\"address\"}],\"payable\":false,\"stateMutability\":\"view\
↪",\"type\":\"function\"},{\"anonymous\":false,\"inputs\":[{\"indexed\":false,\
↪"name\":\"name\",\"type\":\"string\"}],\"name\":\"SetName\",\"type\":\"event\"}]
↪",
        "bytecodeBin":
↪"6080604052348015610010576000080fd5b506104380610020600039600f3006080604052600436106100575760003
↪",
        "account": "admin"
                }
    ],
    "contractVersion": "1.0.0",
    "account": "admin"
}
```


返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

5.2 合约地址保存接口

应用自己部署完合约，导入相应合约地址。导入时，WeBASE会从应用合约仓库表获取对应合约保存到合约表，从而可以在WeBASE管理平台进行展示和调用。

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/contractAddressSave?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/contractAddressSave?
↪timestamp=1614928857832&appKey=fdsf78aW&
↪signature=EEFD7CD030E6B311AA85B053A90E8A31
```

```
{
    "groupId": "1",
    "contractName": "hellos",
    "contractPath": "/",
    "contractVersion": "1.0.0",
    "contractAddress": "0x651a8e7899084019f82a93da880d77470bacf6d7"
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

8.4.6 6 其他接口

6.1 查询数据库信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/api/dbInfo?appKey={appKey}&signature={signature}&timestamp={timestamp}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/api/dbInfo?timestamp=1614928857832&
↪appKey=fdsf78aW&signature=EEFD7CD030E6B311AA85B053A90E8A31
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "dbIp": "127.0.0.1",
        "p2pPort": 3306,
        "dbUser": test,
        "dbPwd":
↪"Yjk4YzM3Y2EzNTMxMzNiOWI2MWUwOTMxODhmOTk2NTc2MGYxMTBhMTljNTI2MmY3NTcz=="
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

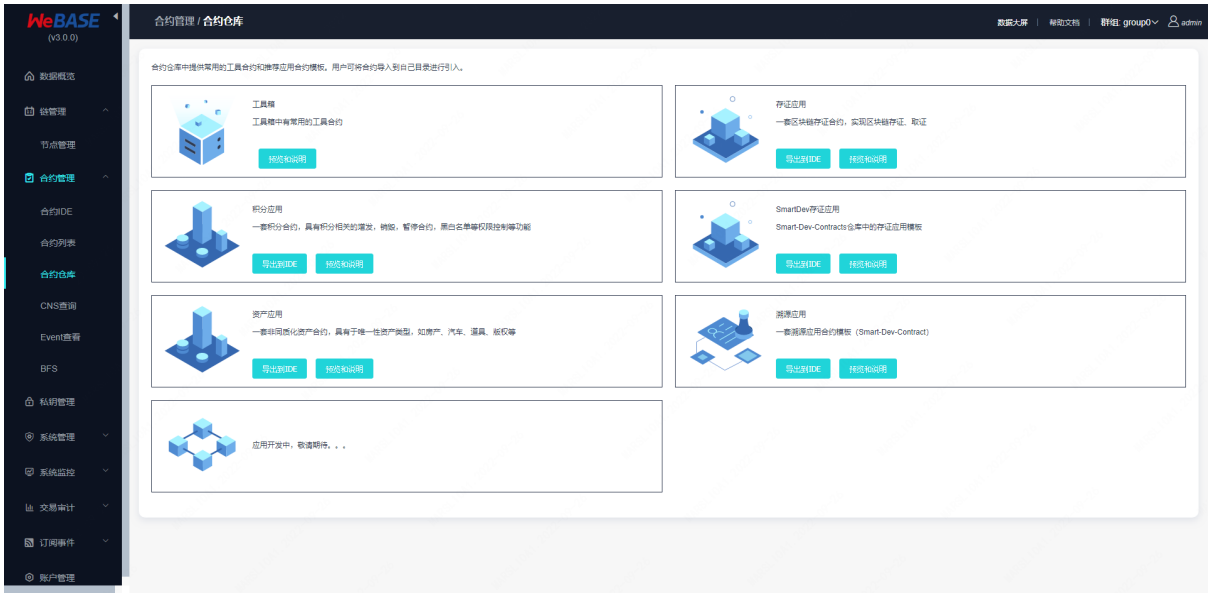
8.4.7 附录

1. 返回码信息列表

CHAPTER 9

WeBASE合约仓库

合约仓库内置于WeBASE-Front与WeBASE管理平台中。其中提供常用的工具合约和推荐应用合约模板，用户可将合约导入到自己目录进行引入。



9.1 合约仓库贡献者

9.1.1 工具合约

Address

Address library

Address contract usage methods can refer to the Points contract warehouse。

1. Detect whether the address is a contract
2. Detect if the address is 0x0

String library

String library

Provides common string-related operations, including copying, finding, replacing, and so on。

Roles

Role permissions control contracts

SafeMath

SafeMath library

A secure mathematical library that provides a safe addition, subtract, and divide。 The use of secure mathematical contracts can refer to the Points contract warehouse。

table

table library

BCOS CRUD uses the base library, which you can reference by re-contract calling CRUD.Table contract usage methods can be described by referring to the assat example in the bcos document https://fisco-bcos-documentation.readthedocs.io.

9.1.2 存证合约模板

简介

Evidence 示例合约，使用分层的智能合约结构：

- 工厂合约（EvidenceSignersData.sol），由存证各方事前约定，存储存证生效条件，并管理存证的生成。
- 存证合约（Evidence.sol），由工厂合约生成，存储存证id，hash和各方签名（每张存证一个合约）。

使用步骤：

1. 部署EvidenceSignersData合约，并在构造函数中指定存证生效条件（需要哪些机构进行认证确认）。
2. 存证时通过newEvidence接口在区块链上创建具体存证合约。
3. 解析newEvidence调用返回的receipt，将解析出来的存证合约地址保存在应用系统。
4. 仲裁等认证机构利用存证合约地址调用addSignatures来对存证进行确认。
5. 取证时利用存证合约地址调用getEvidence接口进行取证。

9.1.3 代理合约模板

本合约模板由深圳前海股权交易中心基于合约迭代升级的需要，研发合约应用开源实现参考，并基于拥抱开源的理念贡献给社区，包括合约接口层代理、合约数据层代理等主要功能。

本合约模板社区贡献者： [xiaomdong]https://github.com/xiaomdong

简介

本合约模板由深圳前海股权交易中心贡献，针对数据上链编写的通用代理存储合约。

代理合约利用solidity的fallback功能，包含EnrollProxy（代理合约），EnrollController（业务合约），EnrollStorage（存储合约）。

- 代理合约对外交互接口
- 业务合约实现业务逻辑
- 存储合约完成数据存储

EnrollProxy合约通过Fallback机制调用EnrollController合约的函数进行数据上链（通过EnrollProxy合约地址结合使用EnrollController合约的ABI，操作EnrollController合约的函数），其带来的优点包括：

- 区块链应用的业务层只与EnrollProxy合约进行交互，EnrollProxy合约不会升级，地址不会变化。
- 后续中业务或存储需求导致业务合约或存储合约需要升级，则升级EnrollController和EnrollStorage合约，达到数据、业务逻辑解耦的效果。

期待你一起完善合约模板中的权限控制逻辑

合约架构说明

```
EnrollProxy
        继承EnrollStorageStateful
        继承Proxy（继承Ownable）

EnrollController
        继承EnrollStorageStateful
        继承Ownable

EnrollStorageStateful
        包含成员enrollStorage，EnrollStorage合约实例

由于是继承的关系，EnrollProxy合约和EnrollController合约的存储空间排列是一样的，所以可通
过EnrollProxy执行fallback操作。

enrollStorage是EnrollStorageStateful合约中的成员，所以enrollStorage合约
与EnrollStorageStateful合约存储空间排布是不一样。
```

使用说明

1. 编译部署EnrollProxy，EnrollController，EnrollStorage合约。
2. 配置代理合约：
    1. 存储合约合约：调用EnrollProxy合约setStorage函数，参数为EnrollStorage合约地址。
    2. 配置业务合约：调用EnrollProxy合约upgradeTo函数，参数为：合约版本号，EnrollController合约地址。
3. 设置存储合约的代理地址：调用EnrollStorage合约setProxy函数，参数为EnrollProxy合约地址。

完成以上步骤后，就可以通过EnrollProxy合约地址，结合业务合约EnrollController合约的ABI，操作EnrollController合约的业务函数。

9.1.4 溯源合约模板

本合约模板由世纪鼎利科技股份有限公司基于家禽溯源场景，贡献合约应用开源实现参考，包括种类注册、家禽注册、状态变更、溯源查询等主要合约功能。

本合约模板社区贡献者： [fengqiao]https://github.com/fengqiao

简介

包含创建Traceability溯源类目、创建Goods溯源商品、更新溯源/商品状态、获取溯源/商品信息等
合约:

1. Goods 溯源商品
2. Traceability 商品溯源类目
3. TraceabilityFactory 溯源工厂类

9.1.5 资产合约

9.1.6 BAC002 合约规范

简介

BAC002 是区块链上定义非同质化资产的一种标准，可以用于唯一性资产类型，如房产、汽车、道具、版权等。，并可以做相应增发，销毁，暂停合约，黑白名单等权限控制。

三个基本元素

- description

  资产的具体描述

- shortName

  资产简称

- assetId

  资产编号

五个基本行为

- 发行

  调用合约的 deploy 方法，传入 description 和 shortName，即在区块链上发行指定名称的资产

- 转账

  调用 safeSendFrom 方法实现转账，调用 balance 方法可以查看自己的资产数量

- 增发

  调用 issueWithAssetURI 方法向资产地址增发指定资产编号和资产描述链接信息的资产。另外，可以通过 addIssuer 增加 有权限增发资产的人，也可以通过 renounceIssuer 方法移除增发权限

- 销毁

  调用 destroy 以及 destroyFrom 销毁自己地址下资产和特定地址下的资产

- 暂停

  遇到紧急状况，你可以调用 suspend 方法，暂停合约，这样任何人都不能调用 send 函数。故障修复后，可以调用 unSuspend 方法解除暂停。也可以通过 addSuspender 和 renounceSuspender 相应增加和移除暂停者权限

接口说明

- shortName()

  资产简称

- description()

  资产描述

- balance(address owner)

  返回 owner 的资产总数

- totalSupply()

  获得当前合约总的资产数目

- ownerOf(uint256 assetId)

  返回资产持有者的地址

- approve(address to, uint256 assetId)

  授予地址to具有指定资产的控制权

  - 此方法配合 getapproved 使用

- getApproved(uint256 assetId)

  获得资产授权的地址用户

  - 此方法配合 approve 使用，注意不要配合 setapprovealforall 方法使用

- setApprovalForAll(address operator, bool approved)

  授予地址operator具有自己所有资产的控制权

- isApprovedForAll(address owner, address operator)

  查询授权

- sendFrom(address from, address to, uint256 assetId, bytes memory data)

  安全转账，防止你转到错误的合约地址 (to如果是合约地址，必须实现接收接口 BAC002Holder 才可以接收转账)，并可以带转账备注

  - suspend 状态下无法执行此操作

- batchSendFrom(address from, address[] to, uint256[] assetId, bytes memory data)

  批量安全转账

  - suspend 状态下无法执行此操作
  - to 数组元素个数需要和 assetid 数组元素个数一致

- issueWithAssetURI(address to, uint256 assetId, string memory assetURI, bytes data)

  给地址 to 创建资产 assetId，data 是转账备注, assetURI 资产描述

- isIssuer(address account)

  检查account是否有增加资产的权限

- addIssuer(address account)

  使地址 account 拥有增加资产的权限

- renounceIssuer()

  移除增加资产的权限

- suspend()

  暂停合约

  - suspend 后无法进行 safesendfrom / sendfrom / safeBatchSendFrom 操作

- unSuspend()

  重启合约

  - 此方法配合 suspend 使用

- isSuspender(address account)

  是否有暂停合约权限

  - 此方法配合 addsuspender 使用

- addSuspender(address account)

  增加暂停权限者

  - 此方法配合 renouncesuspender / issuspender 放啊发使用

- renounceSuspender()

  移除暂停权限

- destroy(uint256 assetId, bytes data)

  减少自己的资产，data 是转账备注

  - 调用时，value 值需要小于等于目前自己的资产总量

- assetOfOwnerByIndex(address owner, uint256 index)

  根据索引 index 获取 owner 的资产 ID

- assetByIndex(uint256 index)

  根据索引 index 获取当前合约的资产 ID

9.1.7 积分合约模板

简介

BAC001 是一套区块链积分合约，可以积分相关的增发，销毁，暂停合约，黑白名单等权限控制。

四个基本元素

- description

  此积分的具体描述

- shortName

  积分简称

- minUnit

  积分最小单位

- totalAmount

  积分总数量

五个基本行为:

- 发行

  调用合约的 deploy 方法，传入你初始化的四个元素即可，即在区块链上发行了你指定总量和名称的积分。

  - 其中 minUnit 和 totalAmount 不能为负数或小数

- 转账

  调用 send 方法即可实现转账，之后调用 balance 方法可以查看自己的积分余额

- 增发

  调用 issue 方法特定地址增发积分， 并可以通过 addIssuer 增加有权限增发积分的人，也可以通过renounceIssuer 方法移除增发权限

- 销毁

  调用 destory 以及 destoryFrom 销毁自己地址下积分和特定地址下的积分

- 暂停

  遇到紧急状况，你可以调用 suspend 方法，暂停合约，这样任何人都不能调用 send 函数。故障修复后，可以调用 unSuspend 方法解除暂停。也可以通过 addSuspender 和 renounceSuspender 相应增加和移除暂停者权限

使用样例伪码

```
// 示例中Alice和Bob都为一个外部账号地址。

// 部署合约，即初始化积分，示例初始化信息如下：
// 积分描述：car points
// 积分简称 TTT
// 最小转账单位 1
// 发行总量10000000
// 默认合约部署者为第一个发行者
BAC001 bac001 = BAC001.deploy(web3j, credentials,
contractGasProvider, "car points", "TTT", BigInteger.valueOf(1), BigInteger.
↪valueOf(1000000)).send();
String contractAddress = bac001.getContractAddress();

// 增加积分发行者
bac001.addIssuer(Alice).send();

// 增发积分
bac001.issue(Alice, new BigInteger("10000"),"increase 10000 asset  ").send();

// 积分转账，以及积分转账备注 Owner -> Alice
bac001.send(Alice,new BigInteger("10000"),"dinner Points").send();

// 查询积分余额
assertEquals( bac001.balance(Alice).send().toString(),"30000");
```

接口说明

- totalAmount()

  返回积分总量

  - 这里的积分总量需要计算最小转账单位，所以实际返回值为 totalAmount * 10minUnit

- balance(address owner)

  返回owner的帐户的积分余额

- send(address to, uint256 value , string data)

  将数量为value的积分转入地址 to 并触发 transfer 事件, data 是转账备注

  - suspend 状态下无法进行此操作

  - 请避免 to 为自身进行操作

- sendFrom(address from,address to,uint256 value，string data))

  将地址 from 中的 value 数量的积分转入地址 to ，并触发 transfer 事件，data 是转账备注。

  - 方法的调用者可以不为 from， 此时需要预先进行 approve 授权
  - from 不能为调用者自身地址，否则会报错
  - suspend 状态下无法执行此操作

- safeSendFrom(address from, address to, uint256 value, string data)

  安全的将地址 from 中的 value 数量的积分转入地址 to (to如果是合约地址，必须实现接收接口 BAC001Holder 才可以接收转账) ，并触发 transfer 事件，data 是转账备注

  - suspend 状态下无法执行此操作

- safeBatchSend(address[] to, uint256[] values, string data)

  批量将自己账户下的积分转给 to 数组的地址， to 和 values 的个数要一致

  - suspend 状态下无法执行此操作

- approve(address spender,uint256 value)

  允许 spender 从自己账户提取限额 value 的积分

  - 此方法配合 sendfrom / safesendfrom 一起使用
  - 重复授权时，最终授权额度为最后一次授权的值

- allowance(address owner,address spender)

  返回 spender 可从 owner 提取的积分数量上限

  - 此方法配合 approve 一起使用

- increaseAllowance(address spender, uint256 addedValue)

  允许 spender 提取的积分上限在原有基础上增加 addedValue

  - 此方法配合 approve 使用

- decreaseAllowance(address spender, uint256 subtractedValue)

  允许 spender 提取的积分上限在原有基础上减少 subtractedValue

  - 此方法配合 approve 使用

- minUnit()

  积分最小单位

- shortName()

  积分简称

- description()

  积分描述

- destory(uint256 value， string data)

  减少自己的积分，data 是转账备注

  - 调用时，value 值需要小于等于目前自己的积分总量

- destroyFrom(address from, uint256 value， string data)

  减少地址 from 积分，data 是转账备注

  - 调用此方法时，需要配合 approve 进行使用

- issue(address to, uint256 value，string data)
  给地址 to 增加数量为 value 的积分，data 是转账备注
- isIssuer(address account)
  检查 account 是否有增加积分的权限
- addIssuer(address account)
  使地址 account 拥有增加积分的权限
- renounceIssuer()
  移除增加积分的权限
- suspend()
  暂停合约
  - suspend 后无法进行 send / safesendfrom / sendfrom / safeBatchSend / approves 操作
- unSuspend()
  重启合约
- suspended
  判断合约是否处于暂停状态
- isSuspender(address account)
  是否有暂停合约权限
  - 配合 suspend 方法一起使用
- addSuspender(address account)
  增加暂停权限者
  - 配合 suspend 方法一起使用
- renounceSuspender()
  移除暂停权限
  - 配合 suspend / addSuspender 方法使用

9.1.8 Evidence存证

简介

存证操作，上传、审批、修改、删除等，详情查看Smart-Dev Evidence Doc
合约:

1. EvidenceController 对外服务的唯一接口
2. EvidenceRepository 辅助合约，用于数据和逻辑分离
3. RequestRepository 辅助合约，用于数据和逻辑分离
4. Authentication 辅助合约，用于数据和逻辑分离

CHAPTER 10

WeBASE实训课程案例

WeBASE智能合约的实训课程案例集，可结合WeBASE完成实训课程与题目设计。

10.1 实训一：运行第一个智能合约

使用Solidity语言编写一个HelloWorld合约，合约包含一个string变量、一个get方法和一个set方法。

1. 构造函数初始化该变量为”Hello world!”
2. 提供get方法获取变量string的值
3. 提供set方法设置变量string的值

例：

```
pragma solidity >=0.4.25 <0.6.11;
contract HelloWorld {
string name;

constructor() public ...
function get() public ...
function set() public ...
}
```

10.1.1 实验步骤：

1）准备好区块链运行环境

使用FISCO BCOS搭建4节点的区块链，也可用系统自带的区块链。

2）编写智能合约

可使用系统自带的智能合约IDE编写智能合约

提交方式：

- 提交智能合约源码

3）编译部署智能合约

合约IDE进行编译、部署

提交方式：

- 部署成功后的交易回执截图
- 部署成功后的智能合约截图，截图应包含合约地址

4）向部署的智能合约发送交易

编写一个区块链应用程序，可以通过SDK连接区块链节点，并向智能合约发送交易。

提交方式：

- 提交初始化SDK连接节点源码代码
- 提交HelloWorld合约Java类截图
- 提交调用合约Java类set方法的源代码
- 通过Java调用HelloWorld合约set方法，将变量设置为“Hello From Java!”，在控制台输出交易哈希，截图并提交
- 通过合约IDE调用合约，获取string变量的值，其值应为“Hello From Java!”，截图返回结果并提交

5）通过区块链浏览器查看交易

发送的交易（交易哈希）可通过系统自带的区块链浏览器展示，确认在哪个区块中。

提交方式：

- 区块链浏览器中交易回执/交易详情的截图

10.1.2 参考答案：

1）智能合约：

- 要求constructor构造函数初始化string变量
- 要求get方法是查询交易view/constant/pure中的一种，返回string变量的值
- 要求set方法设置string变量

源码参考（实现方式不唯一）：

```
pragma solidity >=0.4.25 <0.6.11;
contract HelloWorld {
string name;
constructor() public
{
  name = "Hello, World!";
}
function get() public view returns (string memory){
    return name;
 }
function set(string memory n) public {
   name = n;
 }
}
```

2）调用端：

以Java语言为例，从Solidity智能合约，生成合约Java类，并传入调用set方法所需参数。

- 要求加载SDK的Client实例连接节点截图
- 要求合约Java类的源码截图，可通过工具将Solidity源码转为Java类
- 要求传入合约地址、SDK连接实例及私钥对来初始化/加载一个HelloWorld合约实例
- 要求调用合约实例的set方法，入参为"Hello sent from Java!"
- 要求输出/打印/记录调用set方法返回的交易回执，或打印交易回执的交易哈希

合约Java类参考：

```
package com.sx.demo;
// import内容，略

public class HelloWorld extends Contract {
    // BIN和ABI内容略
    public static final String[] BINARY_ARRAY = {""};
    public static final String BINARY = org.fisco.bcos.sdk.utils.StringUtils.
↪joinAll("", BINARY_ARRAY);
    public static final String[] SM_BINARY_ARRAY = {""};
    public static final String SM_BINARY = org.fisco.bcos.sdk.utils.StringUtils.
↪joinAll("", SM_BINARY_ARRAY);
    public static final String[] ABI_ARRAY = {""};
    public static final String ABI = org.fisco.bcos.sdk.utils.StringUtils.joinAll("
↪", ABI_ARRAY);

    public static final String FUNC_SET = "set";
    public static final String FUNC_GET = "get";

    protected HelloWorld(String contractAddress, Client client, CryptoKeyPair␣
↪credential) {
        super(getBinary(client.getCryptoSuite()), contractAddress, client,␣
↪credential);
    }

    public static String getBinary(CryptoSuite cryptoSuite) {
        return (cryptoSuite.getCryptoTypeConfig() == CryptoType.ECDSA_TYPE ?␣
↪BINARY : SM_BINARY);
    }

    public TransactionReceipt set(String n) {
        final Function function = new Function(
                FUNC_SET,
                Arrays.<Type>asList(new org.fisco.bcos.sdk.abi.datatypes.
↪Utf8String(n)),
                Collections.<TypeReference<?>>emptyList());
        return executeTransaction(function);
    }

    public String get() throws ContractException {
        final Function function = new Function(FUNC_GET,
                Arrays.<Type>asList(),
                Arrays.<TypeReference<?>>asList(new TypeReference<Utf8String>() {}
↪));
        return executeCallWithSingleValueReturn(function, String.class);
    }

    public static HelloWorld load(String contractAddress, Client client,␣
↪CryptoKeyPair credential) {
```

(下页继续)

(续上页)

```
        return new HelloWorld(contractAddress, client, credential);
    }
}
```

合约Java类调用源码参考，以单元测试代码为例（SDK连接节点端的Client实例初始化代码以截图为准）。

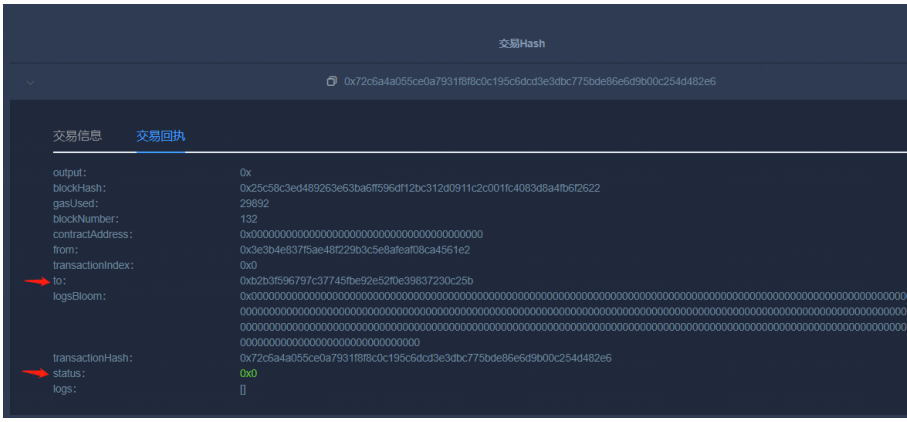
```
@RunWith(SpringRunner.class)
@SpringBootTest(classes = Application.class)
@WebAppConfiguration
public class RawServiceTest {

    // 获取SDK连接节点的Client实例
    @Autowired
    private Client client;

    @Test
    public void testHello() throws Exception {
        // 合约地址
        String contractAddress = "0xb2b3f596797c37745fbe92e52f0e39837230c25b";
        // 创建一个私钥对
        CryptoKeyPair keyPair = client.getCryptoSuite().createKeyPair();
        // 加载HelloWorld合约
        HelloWorld helloWorld = HelloWorld.load(contractAddress, client, keyPair);
        // 调用HelloWorld合约set方法
        TransactionReceipt receipt = helloWorld.set("Hello sent from Java!");
        // 打印交易回执的交易哈希
        System.out.println(receipt.getTransactionHash());
    }
}
```

3）区块链浏览器查看上链结果：

根据步骤2可以得到在Java端调用HelloWorld的set方法后，得到的交易哈希。拿到交易哈希后，可以通过区块链浏览器，查看交易哈希对应的交易回执的详情。

- 要求区块链浏览器上查询的交易哈希与上文调用的交易哈希一致、交易的被调用方to为上文部署的合约地址
- 要求交易回执的状态为成功（此处为0x0）

下图以WeBASE-Front为例，查看交易回执



10.2 实训二：实现积分转账合约

使用Solidity语言编写一个Asset积分合约，合约包含一个address变量记录发行者issuer、一

个mapping(address => uint256)变量balances记录各地址的余额、一个issue方法和一个send方法。

1. 构造函数初始化issuer变量为合约部署者
2. 提供issue方法给特定账户地址发行一定数量的积分，且只有部署合约的issuer才能调用isser方法
3. 提供send方法将调用方余额转账给接收方

例：

```
pragma solidity ^0.4.25;
contract Asset {
    address public issuer;
    mapping (address => uint) public balances;

    constructor() ...
    function issue() public  ...
    function send() public ...
}
```

10.2.1 实验步骤：

1）准备好区块链运行环境

使用FISCO BCOS搭建4节点的区块链，也可用系统自带的区块链。

2）编写智能合约

可使用系统自带的智能合约IDE编写智能合约

提交方式：

- 提交智能合约源码

3）编译部署智能合约

合约IDE进行编译、部署

提交方式：

- 提交合约部署成功后的交易回执截图
- 提交部署成功后的智能合约截图，截图应包含合约地址
- 调用合约，获取合约的issuer变量的值，提交截图
- 调用issue方法，给一个地址如Alice私钥发行100个积分，提交截图
- 调用合约，获取合约的balances变量中Alice的余额值，提交截图

4）向部署的智能合约发送交易

编写一个区块链应用程序，可以通过SDK连接区块链节点，并向智能合约发送交易。

提交方式：

- 提交Asset合约Java类截图
- 调用Asset的合约Java类send方法的源代码
- 通过Java调用Asset合约Java类的send方法，向issuer转账1个积分，在控制台输出交易哈希，截图并提交
- 通过Java调用Asset合约Java类，获取issuer的余额，在控制台输出issuer地址和余额值，截图并提交

- 通过合约IDE获取issuer余额，截图并提交

5）通过区块链浏览器查看交易

发送的交易（交易哈希）可通过系统自带的区块链浏览器展示，确认在哪个区块中。

10.2.2 参考答案：

1）准备好区块链运行环境

略

2） 编写智能合约

- 要求constructor设置issuer为合约部署者msg.sender
- 要求issue方法入参包含接收方address类型变量，以及发行数量uint类型
- 要求issue方法中校验该方法调用者msg.sender必须是issuer
- 要求issue方法给mapping变量中的接收方地址增加余额
- 要求send方法入参包含接收方address类型变量，以及发行数量uint类型
- 要求send方法在mapping中扣减发送方余额，且增加接收方余额
- 可选项：要求send方法在扣减发送方余额前，校验发送方余额是否满足

合约源码参考答案（实现方式不唯一）

```
pragma solidity ^0.4.25;

contract Asset {
    address public issuer;
    mapping (address => uint) public balances;
    constructor() {
        // 设置issuer为部署者
        issuer = msg.sender;
    }
    function issue(address receiver, uint amount) public {
        // 调用者必须是issuer
        if (msg.sender != issuer) return;
        // 增加余额
        balances[receiver] += amount;
    }
    function send(address receiver, uint amount) public {
        if (balances[msg.sender] < amount) return;
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
    }
}
```

3）编译部署智能合约

- 要求提交成功部署合约的截图，截图包含合约的ABI，BIN和部署得到的合约地址

4）向部署的智能合约发送交易

使用其他编程语言实现以上功能也可，需要截图源码与调用截图

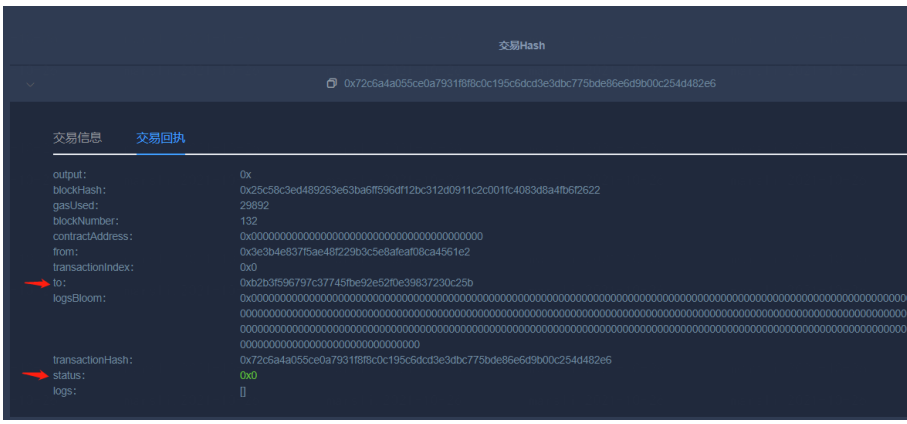
- 要求提交Asset合约Java类截图，可参考WeBASE导出Java类或导出Java项目
- 要求提交调用Asset的合约Java类send方法的源代码
- 要求提交调用Asset的send方法，向issuer转账1个积分，并在控制台输出该交易返回的交易回执，提交交易回执截图
- 要求提交调用Asset获取issuer的余额，输出issuer地址和余额值的截图
- 要求提交通过合约IDE获取issuer余额的截图

5）通过区块链浏览器查看交易

根据步骤2可以得到在Java端调用Asset的send方法后，得到的交易哈希。拿到交易哈希后，可以通过区块链浏览器，查看交易哈希对应的交易回执的详情。

- 要求区块链浏览器上查询的交易哈希与上文调用的交易哈希一致、交易的被调用方to为上文部署的合约地址
- 要求交易回执的状态为成功（此处为0x0）

下图以WeBASE-Front为例，查看交易回执



10.3 实训三：实现存证合约

使用Solidity语言实现存证合约，包含一个Evidence合约和一个EvidenceFactory合约。

题目合约内容：

Evidence合约

```
pragma solidity ^0.4.25;

contract EvidenceSignersDataABI
{
        function verify(address addr)public constant returns(bool){}
        function getSigner(uint index)public constant returns(address){}
        function getSignersSize() public constant returns(uint){}
}

contract Evidence{
    string evidence;
    address[] signers;
    address public factoryAddr;
    event addSignaturesEvent(string evi);
```

(下页继续)

(续上页)

```
    event newSignaturesEvent(string evi, address addr);
    event errorNewSignaturesEvent(string evi, address addr);
    event errorAddSignaturesEvent(string evi, address addr);
    event addRepeatSignaturesEvent(string evi);
    event errorRepeatSignaturesEvent(string evi, address addr);

    function CallVerify(address addr) public constant returns(bool) {
        return EvidenceSignersDataABI(factoryAddr)./*请在序号后填入 , 勿删除序号[1-1]*/
↪(addr);
    }

   constructor(string evi, address addr) {
        factoryAddr = /*请在序号后填入 , 勿删除序号[1-2]*/;
        if(CallVerify(tx.origin)){
            /*请在序号后填入 , 勿删除序号[1-3]*/
            newSignaturesEvent(evi,addr);
        }else{
            errorNewSignaturesEvent(evi,addr);
        }
    }

    function getEvidence() public constant returns(string,address[],address[]){
        uint length = EvidenceSignersDataABI(factoryAddr)./*请在序号后填入 , 勿删除序
号[1-4]*/;
         address[] memory signerList = /*请在序号后填入 , 勿删除序号[1-5]*/(length);
         for(uint i= 0 ;i<length ;i++) {
             signerList[i] = (EvidenceSignersDataABI(factoryAddr)./*请在序号后填入 ,
勿删除序号[1-6]*/(i));
         }
        return(/*请在序号后填入 , 勿删除序号[1-6]*/,signerList,signers);
    }

    function addSignatures() public returns(bool) {
        for(uint i= 0 ;i</*请在序号后填入 , 勿删除序号[1-7]*/.length ;i++) {
            if(tx.origin == signers[i]) {
                addRepeatSignaturesEvent(evidence);
                return /*请在序号后填入 , 勿删除序号[1-8]*/;
            }
        }
       if(CallVerify(tx.origin)) {
            signers./*请在序号后填入 , 勿删除序号[1-9]*/;
            addSignaturesEvent(evidence);
            return /*请在序号后填入 , 勿删除序号[1-10]*/;
       } else {
           errorAddSignaturesEvent(evidence,tx.origin);
           return /*请在序号后填入 , 勿删除序号[1-11]*/;
       }
    }

    function getSigners()public constant returns(address[])
    {
        uint length = EvidenceSignersDataABI(factoryAddr)./*请在序号后填入 , 勿删除序
号[1-12]*/;
         address[] memory signerList = /*请在序号后填入 , 勿删除序号[1-13]*/(length);
         for(uint i= 0 ;i<length ;i++) {
             signerList[i] = (EvidenceSignersDataABI(factoryAddr)./*请在序号后填入 ,
勿删除序号[1-14]*/);
         }
         return signerList;
    }
}
```

EvidenceFactory合约

```
pragma solidity ^0.4.25;
import "Evidence.sol";

contract EvidenceFactory{
    address[] signers;
        event newEvidenceEvent(address addr);

    constructor(address[] evidenceSigners){
        for(uint i=0; i<evidenceSigners.length; ++i) {
            signers./*请在序号后填入 ， 勿删除序号[2-1]*/;
                }
        }

    function newEvidence(string evi)public returns(address){
        Evidence evidence = /*请在序号后填入 ， 勿删除序号[2-2]*/;
        newEvidenceEvent(evidence);
        return evidence;
    }

    function getEvidence(address addr) public constant returns(string,address[],
↪address[]){
        return Evidence(addr)./*请在序号后填入 ， 勿删除序号[2-3]*/;
    }

    function addSignatures(address addr) public returns(bool) {
        return Evidence(addr)./*请在序号后填入 ， 勿删除序号[2-4]*/;
    }

    function verify(address addr)public constant returns(bool){
        for(uint i=0; i<signers.length; ++i) {
            if (addr == signers[i]) {
                return /*请在序号后填入 ， 勿删除序号[2-5]*/;
            }
        }
        return /*请在序号后填入 ， 勿删除序号[2-6]*/;
    }

    function getSigner(uint index)public constant returns(address){
        uint listSize = signers.length;
        if(index < listSize) {
            return /*请在序号后填入 ， 勿删除序号[2-7]*/;
        } else{
            return 0;
        }
    }

    function getSignersSize() public constant returns(uint){
        return signers./*请在序号后填入 ， 勿删除序号[2-8]*/;
    }

    function getSigners() public constant returns(address[]){
        return /*请在序号后填入 ， 勿删除序号[2-9]*/;
    }
}
```

10.3.1 实验步骤:

1）理解智能合约的功能

1. 分别描述Evidence合约和EvidenceFactory的功能
2. 分别描述Evidence合约中string evidence、address[] signers、address public factoryAddr的作用
3. 描述Evidence合约中EvidenceSignersDataABI合约的作用

2）完成智能合约空缺部分

将题目合约中的空缺部分/*待填入*/填补，可使用系统自带的智能合约IDE编写智能合约

提交方式：

- 提交智能合约源码，包含Evidence合约中1-1至1-14共14个空，以及EvidenceFactory合约中2-1至2-9共9个空。

3）编译部署智能合约

填写完合约空缺部分后，通过合约IDE编译合约。

- 同时，通过创建Alice私钥、Bob私钥和Caron私钥，以Alice、Bob、Caron私钥的地址为参数，部署合约

提交方式：

- 提交合约部署成功后的交易回执截图
- 提交部署成功后的智能合约截图，截图应包含合约地址
- 调用合约，获取合约的signer变量的值，提交截图（signer应该是Alice、Bob、Caron的地址）

4）向部署的智能合约发送交易

- 通过Alice私钥调用EvidenceFactory合约的newEvidence方法创建一个存证，提交交易回执截图，并记录Evidence的地址
- 调用EvidenceFactory合约getEvidence方法，传入上一步骤获得的Evidence，获取刚创建的存证，提交截图
- 通过Bob私钥调用EvidenceFactory合约addSignatures方法，提交交易回执截图
- 再调用EvidenceFactory合约getEvidence方法，获取存证中新增的Bob的签名，提交截图

5）编写应用程序调用合约

- 编写一个区块链应用程序，可以通过SDK连接区块链节点，并向智能合约发送交易。

提交方式：

- 提交EvidenceFactory合约Java类截图
- 加载Caron私钥，通过Caron的私钥调用EvidenceFactory合约Java类的addSignatures方法，给步骤4中创建的Evidence存证添加Caron的签名，在console控制台输出交易哈希，截图并提交
- 通过Java调用EvidenceFactory合约Java类的getEvidence方法，获取上述步骤所创建的存证，在控制台输出存证的地址和存证签名数，此时签名数应该为3（包含Alice、Bob、Caron的地址），截图并提交

10.3.2 参考答案：

1）理解智能合约的功能

1. 分别描述Evidence合约和EvidenceFactory的功能

Evidence合约记录存证内容和各方签名，EvidenceFactory合约用于管理Evidence存证的生成 (Evidence合约指出了用于保存存证即可，EvidenceFactory指出了用于生成或管理Evidence存证即可）

1. 分别描述Evidence合约中string evidence、address[] signers、address public factoryAddr的作用

evidence用于存储存证内容、signers用于存储存证签名者的签名、factoryAddr用于记录存证工厂合约的地址 （evidence指出存储存证即可、signers指出是保存签名即可、factoryAddr指出记录工厂合约或管理存证合约的合约地址即可）

1. 描述Evidence合约中EvidenceSignersDataABI合约的作用

用于调用EvidenceFactory合约中的verify，getSigner，getSignerSize方法 (指出EvidenceSignersDataABI用于获取签名者数据，或signer数据可得分、指出用于调用EvidenceFactory或获取EvidenceFactory签名者数据也可以得分、指出用于EvidenceFactory接口或签名者接口也可以得分。直接翻译verify，getSigner，getSignerSize的中文意思不得分)

2）完成智能合约空缺部分

- 将空缺的部分天上，并成功编译部署合约
- 若未成功部署，则提供参考合约

Evidence合约

```
pragma solidity ^0.4.25;

contract EvidenceSignersDataABI
{
        function verify(address addr)public constant returns(bool){}
        function getSigner(uint index)public constant returns(address){}
        function getSignersSize() public constant returns(uint){}
}

contract Evidence{

    string evidence;
    address[] signers;
    address public factoryAddr;

    event addSignaturesEvent(string evi);
    event newSignaturesEvent(string evi, address addr);
    event errorNewSignaturesEvent(string evi, address addr);
    event errorAddSignaturesEvent(string evi, address addr);
    event addRepeatSignaturesEvent(string evi);
    event errorRepeatSignaturesEvent(string evi, address addr);

    function CallVerify(address addr) public constant returns(bool) {
        return EvidenceSignersDataABI(factoryAddr).verify(addr);
    }

   constructor(string evi, address addr)  {
       factoryAddr = addr;
       if(CallVerify(tx.origin)){
           evidence = evi;
           signers.push(tx.origin);
           newSignaturesEvent(evi,addr);
```

(下页继续)

(续上页)

```
        }else{
            errorNewSignaturesEvent(evi,addr);
        }
    }

    function getEvidence() public constant returns(string,address[],address[]){
        uint length = EvidenceSignersDataABI(factoryAddr).getSignersSize();
         address[] memory signerList = new address[](length);
         for(uint i= 0 ;i<length ;i++) {
             signerList[i] = (EvidenceSignersDataABI(factoryAddr).getSigner(i));
         }
        return(evidence,signerList,signers);
    }

    function addSignatures() public returns(bool) {
        for(uint i= 0 ;i<signers.length ;i++) {
            if(tx.origin == signers[i]) {
                addRepeatSignaturesEvent(evidence);
                return true;
            }
        }
       if(CallVerify(tx.origin)) {
            signers.push(tx.origin);
            addSignaturesEvent(evidence);
            return true;
       } else {
           errorAddSignaturesEvent(evidence,tx.origin);
           return false;
       }
    }

    function getSigners()public constant returns(address[]) {
         uint length = EvidenceSignersDataABI(factoryAddr).getSignersSize();
         address[] memory signerList = new address[](length);
         for(uint i= 0 ;i<length ;i++) {
             signerList[i] = (EvidenceSignersDataABI(factoryAddr).getSigner(i));
         }
         return signerList;
    }
}
```

EvidenceFactory合约

```
pragma solidity ^0.4.25;
import "Evidence.sol";

contract EvidenceFactory{
    address[] signers;
        event newEvidenceEvent(address addr);

    constructor(address[] evidenceSigners){
        for(uint i=0; i<evidenceSigners.length; ++i) {
            signers.push(evidenceSigners[i]);
                }
        }

    function newEvidence(string evi)public returns(address) {
        Evidence evidence = new Evidence(evi, this);
        newEvidenceEvent(evidence);
        return evidence;
    }
```

(下页继续)

(续上页)

```
    function getEvidence(address addr) public constant returns(string,address[],
↪address[] {
        return Evidence(addr).getEvidence();
    }


    function addSignatures(address addr) public returns(bool) {
        return Evidence(addr).addSignatures();
    }


    function verify(address addr)public constant returns(bool){
        for(uint i=0; i<signers.length; ++i) {
            if (addr == signers[i]) {
                return true;
            }
        }
        return false;
    }

    function getSigner(uint index)public constant returns(address){
        uint listSize = signers.length;
        if(index < listSize) {
            return signers[index];
        } else {
            return 0;
        }

    }

    function getSignersSize() public constant returns(uint){
        return signers.length;
    }

    function getSigners() public constant returns(address[]){
        return signers;
    }
}
```

3）编译部署智能合约

若步骤2提交的合约未编译成功，可以使用参考合约进行部署，完成后续操作。

- 要求创建Alice私钥、Bob私钥和Caron私钥共三个用于存证的私钥
- 以Alice、Bob、Caron的地址为参数，部署合约
- 要求提交成功部署合约的截图，截图包含合约的ABI，BIN和部署得到的合约地址
- 调用合约，获取合约的signer变量的值，提交截图（signer应该是Alice、Bob、Caron的地址）

4）向部署的智能合约发送交易

- 要求通过Alice私钥调用EvidenceFactory合约的newEvidence方法创建一个存证，提交交易回执截图
- 要求调用EvidenceFactory合约getEvidence方法，传入上一步骤中newEvidence获得的Evidence的地址，获取刚创建的存证，提交截图
- 要求通过Bob私钥调用EvidenceFactory合约addSignatures方法，提交交易回执截图

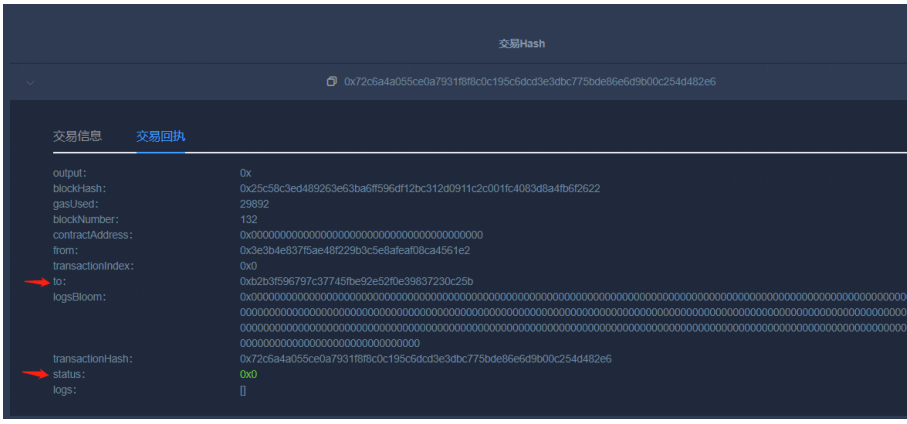
- 要求再调用EvidenceFactory合约getEvidence方法，获取存证中新增的Bob的签名，提交截图

5）编写应用程序调用合约

实现功能的编程语言不限，需要提交源码与调用结果的截图

- 提交EvidenceFactory合约Java类截图
- 加载Caron私钥，通过Caron的私钥调用EvidenceFactory合约Java类的addSignatures方法，给步骤4中创建的Evidence存证添加Caron的签名，在console控制台输出交易哈希，截图并提交
- 通过Java调用EvidenceFactory合约Java类的getEvidence方法，获取上述步骤所创建的存证，在控制台输出存证的地址和存证签名数，此时签名数应该为3（包含Alice、Bob、Caron的地址），截图并提交

下图以WeBASE-Front为例，查看交易回执



CHAPTER 11

使用WeBASE开发区块链应用

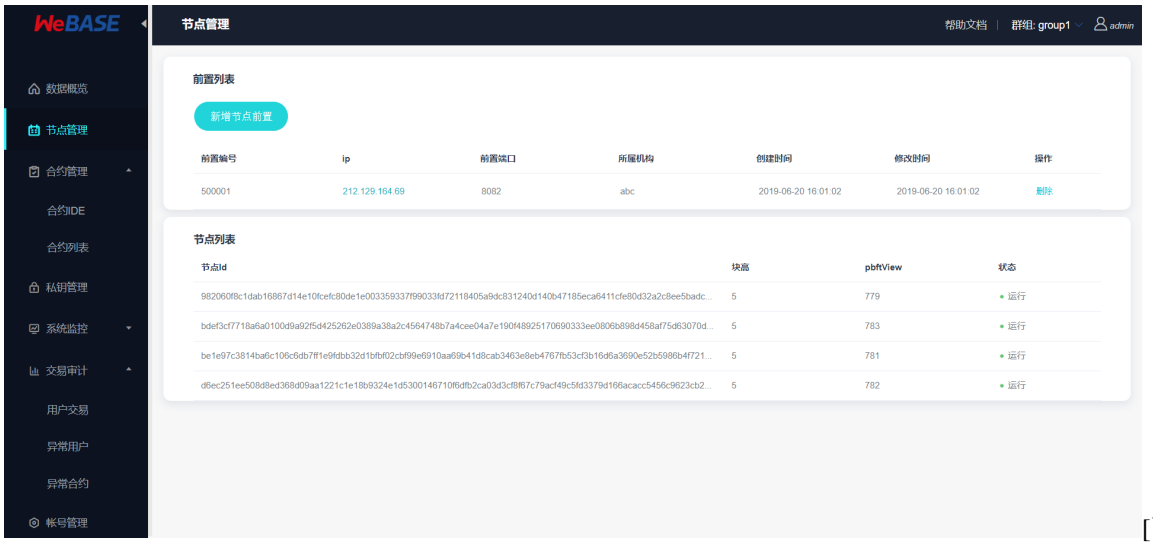
11.1 部署WeBASE

搭建WeBASE, 请参考快速部署

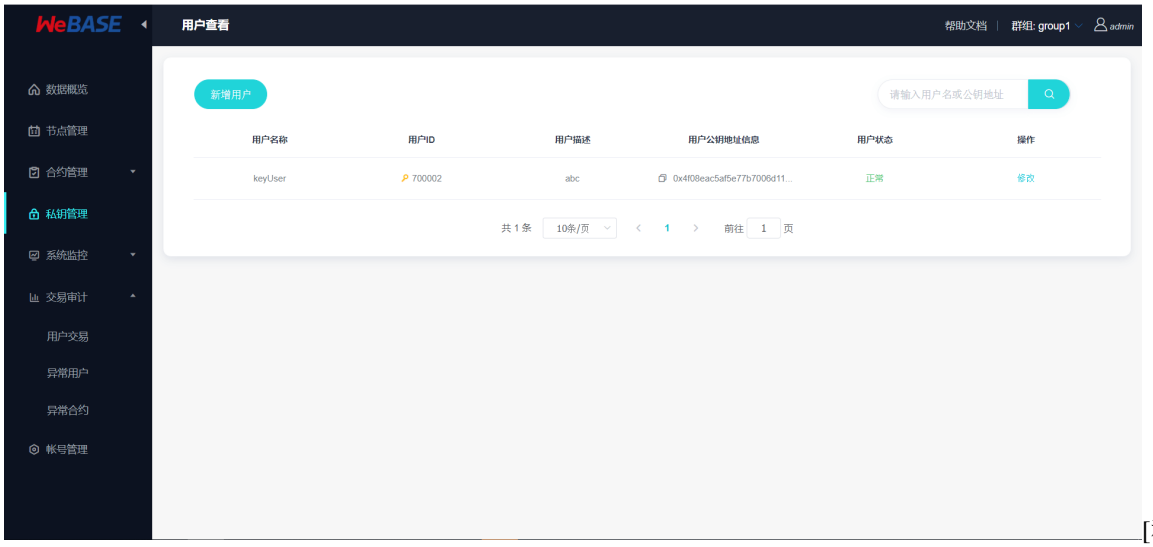
重要： FISCO-BCOS 2.0与3.0对比、JDK版本、WeBASE及其他子系统的版本兼容说明！请查看

11.2 登录WeBASE管理平台进行配置

安装WeBASE完成后，需要将节点信息添加到WeBASE平台中，这样WeBASE才可和节点进行通信。需要添加的信息包含节点信息，生成用户的私钥等。如下图所示:

- 节点信息: 

[节点]

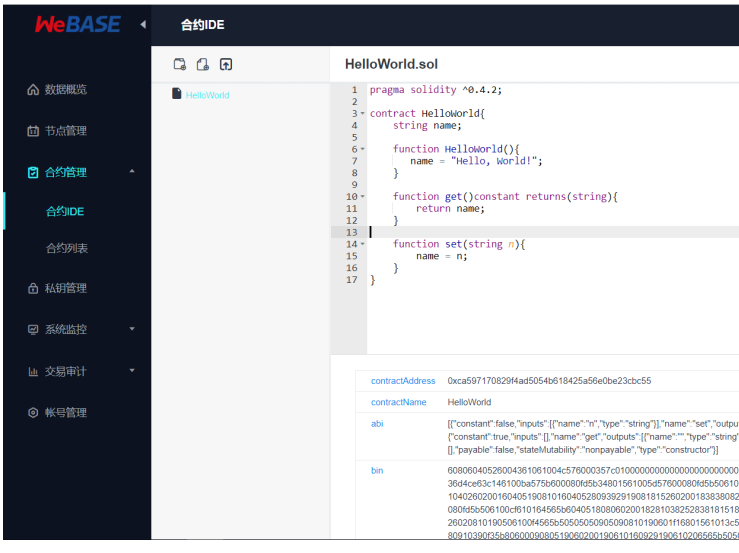


- 私钥用户：[私钥用户]

11.3 开发智能合约

以HelloWorld.sol为例

```
pragma solidity ^0.4.2;

contract HelloWorld{
    string name;

    function HelloWorld(){
       name = "Hello, World!";
    }

    function get()constant returns(string){
        return name;
    }

    function set(string n){
     name = n;
    }
}
```



- 通过智能合约IDE部署合约,并获取合约地址等信息,

约]

11.4 应用层开发

- 调用交易接口

请参考交易接口

从IDE中的输出信息，拷贝合约地址，合约名，方法名等信息，同时获取用户的公钥地址信息，调用交易接口。 具体代码请参考 HelloWorld范例

- 主要代码

```
application.yml

transactionUrl: http://127.0.0.1:5002/WeBASE-Front/trans/handle
groupId: 1
userAddress: "0x4f08eac5af5e77b7006d11bee94adba2f721def8"
useAes: true
contract.name: HelloWorld
contract.address: "0xca597170829f4ad5054b618425a56e0be23cbc55"
contract.funcName: set
contract.funcParam: "[\"abc\"]"
```

- TransactionService.java

```
@Slf4j
@Data
@Service
public class TransactionService {
    @Autowired
    private RestTemplate rest;
    @Value("${transactionUrl}")
    private String url;
    @Value("${userAddress}")
    private String user;
    @Value("${groupId}")
    private int groupId;
    @Value("${useAes}")
    private Boolean useAes;
    @Value("${contract.name}")
    private String contractName;
    @Value("${contract.address}")
    private String contractAddress;
    @Value("${contract.funcName}")
    private String funcName;
    @Value("${contract.funcParam}")
    private String funcParam;

    public void sendTransaction() {

        try {
            TransactionParam transParam = new TransactionParam();
            transParam.setGroupId(groupId);
            transParam.setContractAddress(contractAddress);
            transParam.setUseAes(useAes);
            transParam.setUser(user);
            transParam.setContractName(contractName);
            transParam.setFuncName(funcName);
            transParam.setFuncParam(JSONArray.parseArray(funcParam));
```

(下页继续)

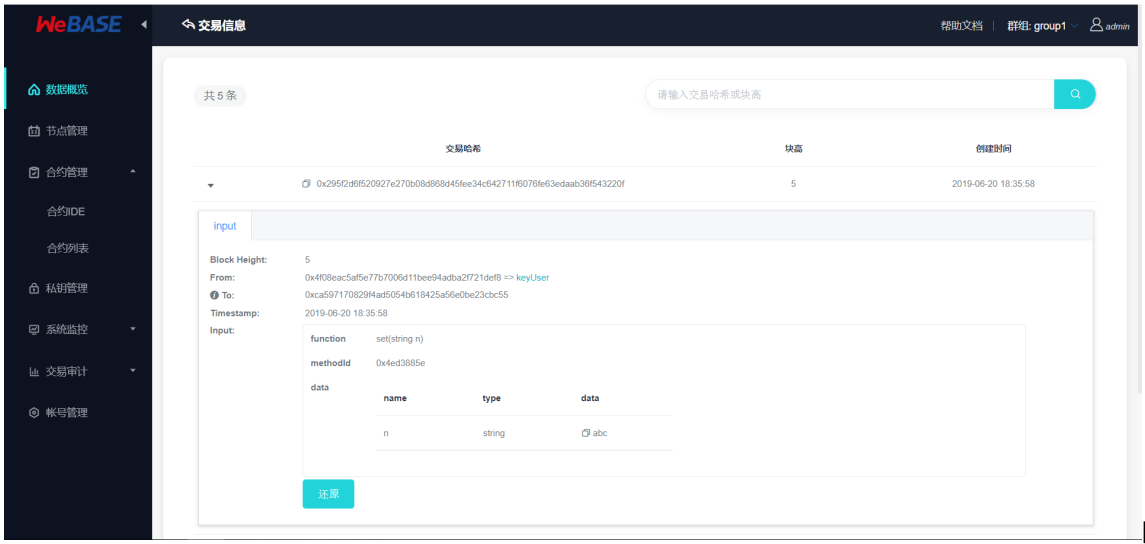
(续上页)

```
            log.info("transaction param:{}", JSON.toJSONString(transParam));
            Object rsp = rest.postForObject(url, transParam, Object.class);
            String rspStr = "null";
            if (Objects.nonNull(rsp)) {
                rspStr = JSON.toJSONString(rsp);
            }
            log.info("transaction result:{}", rspStr);
        } catch (Exception ex) {
            log.error("fail sendTransaction", ex);
        }
        System.exit(1);
    }
}
```

11.5 运维管理

应用层发布后，持续发送交易，可在WeBASE管理平台查看数据概览，节点监控，查看交易解析，交易审计等管理功能。

- 查看交易解析



[交易解析]

CHAPTER 12

节点前置服务

12.1 概要介绍

12.1.1 使用说明

WeBASE-Front是和FISCO-BCOS节点配合使用的一个子系统。此分支支持FISCO-BCOS 2.0以上版本，集成java-sdk，对接口进行了封装，可通过HTTP请求和节点进行通信。另外，具备可视化控制台，可以在控制台上开发智能合约，部署合约和发送交易，并查看交易和区块详情。还可以管理私钥，对节点健康度进行监控和统计。

WeBASE-Front使用方式有以下三种:

1、单独部署作为独立控制台使用，请参考部署说明。

2、结合WeBASE-Node-Manager和WeBASE-Web服务一起部署使用，请参考WeBASE安装部署。

3、结合WeBASE-Sign服务一起部署使用，调用WeBASE-Sign进行数据签名，再发送上链。此方式在方式1的基础上再部署WeBASE-Sign服务，然后需调用合约部署（结合WeBASE-Sign）接口、交易处理（结合WeBASE-Sign）接口进行合约部署和调用（需要在yml中配置sign的地址）。

12.2 部署说明

12.2.1 1. 前提条件

备注： Java推荐使用OracleJDK，可参考JDK配置指引（CentOS的yum仓库的OpenJDK缺少JCE(Java Cryptography Extension)，导致Web3SDK无法正常连接区块链节点）

重要： FISCO-BCOS 2.0与3.0对比、JDK版本、WeBASE及其他子系统的版本兼容说明！请查看

国密支持

WeBASE-Front v1.2.2+已支持 国密版FISCO-BCOS

- 在v1.5.0后，sdk将自动根据链的加密类型切换国密或非国密，自动根据链的SSL类型切换国密SSL

solidity v0.6.10支持

WeBASE-Front v1.4.2已支持solidity `v0.5.1`和`v0.6.10`

solidity v0.8.11支持

WeBASE-Front v3.0.1已支持solidity `v0.8.11`

Liquid支持

如果使用的`liquid`合约的链并在WeBASE-Front的合约IDE中编译Liquid合约，要求**手动**在WeBASE-Front所在主机配置Liquid环境

配置好Liquid环境后，需要重启WeBASE-Front

12.2.2 2. 拉取代码

执行命令：

```
git clone -b master-3.0 https://github.com/WeBankBlockchain/WeBASE-Front.git

# 若因网络问题导致长时间下载失败，可尝试以下命令
git clone -b master-3.0 https://gitee.com/WeBank/WeBASE-Front.git
```

进入目录：

```
cd WeBASE-Front
git checkout lab
```

12.2.3 3. 编译代码

使用以下方式编译构建，如果出现问题可以查看 常见问题解答 ：

方式一：如果服务器已安装Gradle，且版本为gradle-4.10至gradle-6.x版本

```
gradle build -x test
```

方式二：如果服务器未安装Gradle，或者版本不是gradle-4.10至gradle-6.x版本，使用gradlew编译

```
chmod +x ./gradlew && ./gradlew build -x test
```

构建完成后，会在根目录WeBASE-Front下生成已编译的代码目录dist。

12.2.4 4. 修改配置

（1）进入dist目录

```
cd dist
```

dist目录提供了一份配置模板conf_template：

```
根据配置模板生成一份实际配置conf。初次部署可直接拷贝。
例如：cp -r conf_template conf
```

（2）进入conf目录：

```
cd conf
```

注意：将节点所在目录nodes/${ip}/sdk下的所有文件拷贝到当前conf目录，供SDK与节点建立连接时使用（SDK根据application.yml中的useSmSsl判断是否使用国密SSL）

- 链的sdk目录在非国密时，包含ca.crt, sdk.crt, sdk.key，在国密时，包含sm_ca.crt,sm_sdk.crt,sm_sdk.key,sm_ensdk.crt,sm_ensdk.key
- 拷贝命令可使用cp nodes/${ip}/sdk/* ./conf/

（3）修改配置（根据实际情况修改）：

如果在企业部署中使用WeBASE-Front，必须配置下文中的keyServer，用于连接WeBASE-Sign服务

```
vi application.yml
```

```
spring:
  datasource:
    url: jdbc:h2:file:./h2/webasefront;DB_CLOSE_ON_EXIT=FALSE // 默认H2库
为webasefront，建议修改数据库存放路径
...
server:
  port: 5002                        // 服务端口
  context-path: /WeBASE-Front
sdk:
  useSmSsl: false  // 是否启用了国密SSL
  peers: ['127.0.0.1:20200','127.0.0.1:20201'] // 连接的节点（rpc节点）ip与端口，建议连接
所有rpc节点
  certPath: conf    // sdk证书的目录，默认为conf
  ...
constant:
  keyServer: 127.0.0.1:5004      // 密钥服务的IP和端口(WeBASE-Node-Manager服务或
者WeBASE-Sign服务，不同服务支持的接口不同)，如果作为独立控制台以下配置可选
  aesKey: EfdsW23D23d3df43           // aes加密key(16位) 如启用，各互联的子系统的加密key需
保持一致
  transMaxWait: 30                // 交易最大等待时间
...
```

12.2.5 5. 服务启停

返回到dist目录执行：

```
启动：bash start.sh
停止：bash stop.sh
检查：bash status.sh
```

备注：服务进程起来后，需通过日志确认是否正常启动，出现以下内容表示正常；如果服务出现异常，确认修改配置后，重启提示服务进程在运行，则先执行stop.sh，再执行start.sh。

启动成功将出现如下日志：

```
...
        Application() - main run success...
```

12.2.6 6. 访问控制台

```
http://{deployIP}:{frontPort}/WeBASE-Front
示例：http://localhost:5002/WeBASE-Front
```

- 部署服务器IP和服务端口需对应修改，网络策略需开通

- 基于可视化控制台，可以开发智能合约，部署合约和发送交易，并查看交易和区块详情。还可以管理私钥，对节点健康度进行监控和统计

12.2.7 7. 查看日志

在dist目录查看：

```
前置服务全量日志：tail -f log/WeBASE-Front.log
前置服务错误日志：tail -f log/WeBASE-Front.log
web3连接日志：tail -f log/web3sdk.log
```

12.3 接口说明

12.3.1 1. 合约接口

1.1. 发送abi接口

接口描述

根据abi内容判断合约是否已部署，未部署则生成对应abi文件

接口URL

http://localhost:5002/WeBASE-Front/contract/abiInfo

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "abiInfo": [
    {
      "anonymous": true,
      "constant": true,
      "inputs": [
        {
          "components": [
            null
          ],
          "dynamic": true,
          "indexed": true,
          "internalType": "string",
          "name": "string",
          "type": "string",
          "typeAsString": "string"
        }
      ],
```

(下页继续)

(续上页)


```
      "methodSignatureAsString": "string",
      "name": "string",
      "outputs": [
        {
          "components": [
            null
          ],
          "dynamic": true,
          "indexed": true,
          "internalType": "string",
          "name": "string",
          "type": "string",
          "typeAsString": "string"
        }
      ],
      "payable": true,
      "stateMutability": "string",
      "type": "string"
    }
  ],
  "address": "string",
  "contractBin": "string",
  "contractName": "string",
  "groupId": "string",
  "packageName": "string"
}
```


响应参数

1）数据格式

无

1.2. 合约部署接口（结合WeBASE-Sign）

接口描述

将合约部署到当前节点。此接口需结合WeBASE-Sign使用，通过调用WeBASE-Sign服务的签名接口让相关用户对数据进行签名，拿回签名数据再发送上链。需要调用此接口时，工程配置文件application.yml中的配置"keyServer"需配置WeBASE-Sign服务的ip和端口，并保证WeBASE-Sign服务正常和存在相关用户。

3.0.2及以后版本：

构造方法参数（funcParam）为String数组，每个参数都使用String字符串表示，多个参数以逗号分隔（参数为数组时同理），示例：

```
constructor(string s) -> ["aa,bb\"cc"]          // 双引号要转义
constructor(uint n,bool b) -> ["1","true"]
constructor(bytes b,address[] a) -> ["0x1a","[\
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE\",\
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9\"]"]
```

3.0.2以前的版本：

构造方法参数（funcParam）为JSON数组，多个参数以逗号分隔（参数为数组时同理），示例：

```
constructor(string s) -> ["aa,bb\"cc"]          // 双引号要转义
constructor(uint n,bool b) -> [1,true]
constructor(bytes b,address[] a) -> ["0x1a",[
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE",
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9"]]
```

查看*WeBASE-Front*通过本地私钥（测试用户）部署合约的接口（非*WeBASE-Sign*签名交易），可查看其他接口-合约部署接口（本地签名）

接口URL

http://localhost:5002/WeBASE-Front/contract/deployWithSign

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "abiInfo": [
    {}
  ],
  "bytecodeBin": "string",
  "contractBin": "string",
  "contractId": 0,
  "contractName": "string",
  "contractPath": "string",
  "contractSource": "string",
  "funcParam": [

  ],
  "groupId": "string",
  "signUserId": "string",
  "useAes": true,
  "user": "string",
  "version": "string",
  "isWasm": false
}
```

示例：

```
curl -X POST "http://localhost:5002/WeBASE-Front/contract/deployWithSign"  -H
↪"accept: */*" -H "Content-Type: application/json" -d "{ \"abiInfo\":[{\t\t\
↪"constant\": true,\t\t\"inputs\": [],\t\t\"name\": \"get\",\t\t\"outputs\": [
↪{\t\t\t\"name\": \"\",\t\t\t\"type\": \"string\"\t\t}],\t\t\"payable\": false,
↪\t\t\"stateMutability\": \"view\",\t\t\"type\": \"function\"\t}, {\t\t\"constant\
↪": false,\t\t\"inputs\": [{\t\t\t\"name\": \"n\",\t\t\t\"type\": \"string\"\t\t}
↪],\t\t\"name\": \"set\",\t\t\"outputs\": [],\t\t\"payable\": false,\t\t\
↪"stateMutability\": \"nonpayable\",\t\t\"type\": \"function\"\t}, {\t\t\"inputs\
↪": [],\t\t\"payable\": false,\t\t\"stateMutability\": \"nonpayable\",\t\t\"type\
↪": \"constructor\"\t}, {\t\t\"anonymous\": false,\t\t\"inputs\": [{\t\t\t\
↪"indexed\": false,\t\t\t\"name\": \"\",\t\t\t\"type\": \"string\"\t\t}],\t\t\
↪"name\": \"Test\",\t\t\"type\": \"event\"\t}],\"bytecodeBin\": \
↪"6080604052348015610010057600080fd5b506040805190810160405280600d81526020017f48656c6c6f2c20576f72
↪", \"groupId\": "group", \"signUserId\": \"ed54e13b0abf4c69b788bd83b8e3515e\"}"
```

(下页继续)

(续上页)

响应参数

1）数据格式

返回合约地址：

```
{
    "0x7571ff73f1a37ca07f678aebc4d8213e7ef5c266"
}
```

1.3. 合约部署接口(本地签名)

接口描述

此接口为WeBASE-Front使用**本地私钥（页面中的测试用户）进行签名**

将合约部署到当前节点。

3.0.2及以后版本：

构造方法参数（funcParam）为String数组，每个参数都使用String字符串表示，多个参数以逗号分隔（参数为数组时同理），示例：

```
constructor(string s) -> ["aa,bb\"cc"]          // 双引号要转义
constructor(uint n,bool b) -> ["1","true"]
constructor(bytes b,address[] a) -> ["0x1a","[\
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE\",\
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9\"]"]
```

3.0.2以前的版本：

构造方法参数（funcParam）为JSON数组，多个参数以逗号分隔（参数为数组时同理），示例：

```
constructor(string s) -> ["aa,bb\"cc"]          // 双引号要转义
constructor(uint n,bool b) -> [1,true]
constructor(bytes b,address[] a) -> ["0x1a",[
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE",
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9"]]
```

查看*WeBASE-Front*通过*WeBASE-Sign*部署合约的接口（非本地私钥签名交易），可查看合约接口-合约部署接口（结合WeBASE-Sign）

接口URL

http://localhost:5002/WeBASE-Front/contract/deploy

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "abiInfo": [
    {}
  ],
  "bytecodeBin": "string",
  "contractBin": "string",
  "contractId": 0,
  "contractName": "string",
  "contractPath": "string",
  "contractSource": "string",
  "funcParam": [

  ],
  "groupId": "string",
  "signUserId": "string",
  "useAes": true,
  "user": "string",
  "version": "string",
  "isWasm": false
}
```

示例：

```
curl -X POST "http://127.0.0.1:5002/WeBASE-Front/contract/deploy" -H "accept: */*"␣
↪-H "Content-Type: application/json" -d "{ \"abiInfo\": [{\"constant\":true,\
↪"inputs\":[],\"name\":\"get\",\"outputs\":[{\"name\":\"\",\"type\":\"string\"}],\
↪"payable\":false,\"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\
↪":false,\"inputs\":[{\"name\":\"n\",\"type\":\"string\"}],\"name\":\"set\",\
↪"outputs\":[],\"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\
↪"function\"},{\"inputs\":[],\"payable\":false,\"stateMutability\":\"nonpayable\",
↪\"type\":\"constructor\"}], \"bytecodeBin\": \
↪"60806040523480156100105760008fd5b506040805190810160405280600d81526020017f48656c6c6f2c20576f72
↪", \"funcParam\": [], \"groupId\": 1, \"user\": \
↪"0xe11dda6939ef947f9ef78f626e5c4fe0cbcbce1e\"}"
```

响应参数

1）数据格式

返回合约地址：

```
{
    "0x60aac015d5d41adc74217419ea77815ecb9a2192"
}
```

1.4. java转译接口

接口描述

将合约abi转成java文件

接口URL

http://localhost:5002/WeBASE-Front/contract/compile-java

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "abiInfo": [
    {
      "anonymous": true,
      "constant": true,
      "inputs": [
        {
          "components": [
            null
          ],
          "dynamic": true,
          "indexed": true,
          "internalType": "string",
          "name": "string",
          "type": "string",
          "typeAsString": "string"
        }
      ],
      "methodSignatureAsString": "string",
      "name": "string",
      "outputs": [
        {
          "components": [
            null
          ],
          "dynamic": true,
          "indexed": true,
          "internalType": "string",
          "name": "string",
          "type": "string",
          "typeAsString": "string"
        }
      ],
      "payable": true,
      "stateMutability": "string",
      "type": "string"
    }
  ],
  "address": "string",
  "contractBin": "string",
  "contractName": "string",
  "groupId": "string",
  "packageName": "string"
}
```

响应参数

1）数据格式 返回Java合约类源码

```
package com.webank;

import java.util.Arrays;
...

@SuppressWarnings("unchecked")
public class HelloWorld extends Contract {
    public static final String[] BINARY_ARRAY = {""};

    public static final String BINARY = org.fisco.bcos.sdk.utils.StringUtils.
↪joinAll("", BINARY_ARRAY);

    public static final String[] SM_BINARY_ARRAY = {""};

    public static final String SM_BINARY = org.fisco.bcos.sdk.utils.StringUtils.
↪joinAll("", SM_BINARY_ARRAY);

    public static final String[] ABI_ARRAY = {"[{\"name\":\"get\",\"type\":\
↪"function\",\"constant\":true,\"payable\":false,\"anonymous\":false,\
↪"stateMutability\":\"view\",\"inputs\":[],\"outputs\":[{\"name\":\"\",\"type\":\
↪"string\",\"indexed\":false,\"components\":null,\"typeAsString\":\"string\"}],\
↪"methodSignatureAsString\":\"get()\"},{\"name\":\"set\",\"type\":\"function\",\
↪"constant\":false,\"payable\":false,\"anonymous\":false,\"stateMutability\":\
↪"nonpayable\",\"inputs\":[{\"name\":\"n\",\"type\":\"string\",\"indexed\":false,\
↪"components\":null,\"typeAsString\":\"string\"}],\"outputs\":[],\
↪"methodSignatureAsString\":\"set(string)\"},{\"name\":null,\"type\":\
↪"constructor\",\"constant\":false,\"payable\":false,\"anonymous\":false,\
↪"stateMutability\":\"nonpayable\",\"inputs\":[],\"outputs\":null,\
↪"methodSignatureAsString\":\"null()\"}]"};

    public static final String ABI = org.fisco.bcos.sdk.utils.StringUtils.joinAll("
↪", ABI_ARRAY);

    public static final String FUNC_GET = "get";

    public static final String FUNC_SET = "set";

    protected HelloWorld(String contractAddress, Client client, CryptoKeyPair␣
↪credential) {
        super(getBinary(client.getCryptoSuite()), contractAddress, client,␣
↪credential);
    }

    public static String getBinary(CryptoSuite cryptoSuite) {
        return (cryptoSuite.getCryptoTypeConfig() == CryptoType.ECDSA_TYPE ?␣
↪BINARY : SM_BINARY);
    }

    public String get() throws ContractException {
        final Function function = new Function(FUNC_GET,
                Arrays.<Type>asList(),
                Arrays.<TypeReference<?>>asList(new TypeReference<Utf8String>() {}
↪));
        return executeCallWithSingleValueReturn(function, String.class);
    }

    public TransactionReceipt set(String n) {
        final Function function = new Function(
```

(下页继续)

(续上页)

```
                FUNC_SET,
                Arrays.<Type>asList(new org.fisco.bcos.sdk.abi.datatypes.
↪Utf8String(n)),
                Collections.<TypeReference<?>>emptyList());
        return executeTransaction(function);
    }

    public void set(String n, TransactionCallback callback) {
        final Function function = new Function(
                FUNC_SET,
                Arrays.<Type>asList(new org.fisco.bcos.sdk.abi.datatypes.
↪Utf8String(n)),
                Collections.<TypeReference<?>>emptyList());
        asyncExecuteTransaction(function, callback);
    }

    public String getSignedTransactionForSet(String n) {
        final Function function = new Function(
                FUNC_SET,
                Arrays.<Type>asList(new org.fisco.bcos.sdk.abi.datatypes.
↪Utf8String(n)),
                Collections.<TypeReference<?>>emptyList());
        return createSignedTransaction(function);
    }

    public Tuple1<String> getSetInput(TransactionReceipt transactionReceipt) {
        String data = transactionReceipt.getInput().substring(10);
        final Function function = new Function(FUNC_SET,
                Arrays.<Type>asList(),
                Arrays.<TypeReference<?>>asList(new TypeReference<Utf8String>() {}
↪));
        List<Type> results = FunctionReturnDecoder.decode(data, function.
↪getOutputParameters());
        return new Tuple1<String>(

                (String) results.get(0).getValue()
                );
    }

    public static HelloWorld load(String contractAddress, Client client,␣
↪CryptoKeyPair credential) {
        return new HelloWorld(contractAddress, client, credential);
    }

    public static HelloWorld deploy(Client client, CryptoKeyPair credential)␣
↪throws ContractException {
        return deploy(HelloWorld.class, client, credential, getBinary(client.
↪getCryptoSuite()), "");
    }
}
```

1.5. 保存合约接口

接口描述

支持前置的控制台保存合约信息

接口URL

http://localhost:5002/WeBASE-Front/contract/save

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "bytecodeBin": "string",
  "contractAbi": "string",
  "contractAddress": "string",
  "contractBin": "string",
  "contractId": 0,
  "contractName": "string",
  "contractPath": "string",
  "contractSource": "string",
  "groupId": "string",
  "version": "string"
}
```

响应参数

1）参数表

2）数据格式

```
{
    "id": 1,
    "contractPath": "/",
    "contractName": "HelloWorld",
    "contractStatus": 1,
    "groupId": 1,
    "contractSource":
↪"cHJhZ21hIHNvbGlkaXR5ID49MC40LjI0IDwwLjYuMTE7Cgpjb250cmFjdCBIZWxsb1dvcmxkIHsKICAgIHN0cmluZyBuYW
↪",
    "contractAbi": "[{\"constant\":true,\"inputs\":[],\"name\":\"get\",\"outputs\
↪":[{\"name\":\"\",\"type\":\"string\"}],\"payable\":false,\"stateMutability\":\
↪"view\",\"type\":\"function\"},{\"constant\":false,\"inputs\":[{\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"payable\":false,\
↪"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":[],\
↪"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"constructor\"}]",
    "contractBin": "60806040526004361061004c5760003569b80029",
    "bytecodeBin": "60806040523480156100105760080fd5b506029",
    "contractAddress": null,
    "deployTime": null,
    "description": null,
    "createTime": "2019-06-10 11:48:51",
    "modifyTime": "2019-06-10 15:31:29"
}
```

1.6. 删除合约接口

接口描述

支持前置的控制台通过群组编号和合约编号删除未部署的合约信息

接口URL

http://localhost:5002/WeBASE-Front/contract/{groupId}/{contractId}

调用方法

HTTP DELETE

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contract/group/1
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": null
}
```

1.7. 分页查询合约列表

接口描述

支持前置的控制台分页查询合约列表

接口URL

http://localhost:5002/WeBASE-Front/contract/contractList

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractAddress": "string",
  "contractName": "string",
  "contractPath": "string",
  "contractStatus": 0,
  "groupId": "string",
  "pageNumber": 0,
  "pageSize": 0
}
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "id": 2,
            "contractPath": "/",
            "contractName": "HeHe",
            "contractStatus": 1,
            "groupId": 1,
            "contractSource": "cHJhZ21hIHNvbGlkaXR5IICB9Cn0=",
            "contractAbi": "",
            "contractBin": "",
            "bytecodeBin": null,
            "contractAddress": null,
            "deployTime": null,
            "description": null,
            "createTime": "2019-06-10 16:42:50",
            "modifyTime": "2019-06-10 16:42:52"
        }
    ],
    "totalCount": 1
}
```

1.8. 合约是否被修改接口

接口描述

校验已部署的合约是否被修改了，返回true或false

接口URL

http://localhost:5002/WeBASE-Front/contract/ifChanged/{groupId}/{contractId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contract/ifChanged/group/10
```

响应参数

1）数据格式

```
true
```

1.9. 后台编译合约

接口描述

通过后台的solcJ对solidity合约进行编译，返回合约的BIN与ABI

接口URL

http://localhost:5002/WeBASE-Front/contract/contractCompile

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractName": "string",
  "groupId": "string",
  "solidityBase64": "string"
}
```

响应参数

1）参数表

2）数据格式

```
{
  "contractName": "HelloWorld",
  "contractAbi": "[{\"constant\":true,\"inputs\":[],\"name\":\"get\",\"outputs\":[
↪{\"name\":\"\",\"type\":\"string\"}],\"payable\":false,\"stateMutability\":\
↪"view\",\"type\":\"function\"},{\"constant\":false,\"inputs\":[{\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"payable\":false,\
↪"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":[],\
↪"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"constructor\"}]",
  "bytecodeBin":
↪"6080604052348015610010576000080fd5b506040805190810160405280600d81526020017f48656c6c6f2c20576f72
↪",
  "errors": ""
}
```


1.10. 多合约编译

接口描述

接口参数为合约文件压缩成zip并Base64编码后的字符串。合约文件需要放在同级目录压缩，涉及引用请使用"./XXX.sol"。可参考测试类ContractControllerTest的testMultiContractCompile()方法。国密和非国密编译的bytecodeBin不一样，以下以国密为例。

接口URL

http://localhost:5002/WeBASE-Front/contract/multiContractCompile

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式


```
{
  "contractZipBase64": "string",
  "groupId": "string"
}
```


响应参数

1）参数表

2）数据格式


```
[
    {
      "contractName": "HelloWorld",
      "contractAbi": "[{\"constant\":true,\"inputs\":[],\"name\":\"get\",\"outputs\
↪":[{\"name\":\"\",\"type\":\"string\"}],\"payable\":false,\"stateMutability\":\
↪"view\",\"type\":\"function\"},{\"constant\":false,\"inputs\":[{\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"payable\":false,\
↪"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"anonymous\":false,\
↪"inputs\":[{\"indexed\":false,\"name\":\"name\",\"type\":\"string\"}],\"name\":\
SetName\",\"type\":\"event\"}]",
```


(下页继续)

(续上页)

```
      "contractSource":
↪"cHJhZ21hIHNvbGlkaXR5IF4wLjQuMjsNCmNvbnRyYWN0IEhlbGxvV29ybGR7DQogICAgc3RyaW5nIG5hbWU7DQogICAgZX
↪",
      "bytecodeBin":
↪"608060405234801561001057600080fd5b506103738061002060003960000f3006080604052600436106100
↪"
    }
  ]
```

1.11. 获取全量合约列表（不包含abi/bin）

接口描述

根据群组编号和合约状态获取全量合约

接口URL

http://localhost:5002/WeBASE-Front/contract/contractList/all/light?groupId={groupId}&contractStatus={contractStatus}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contract/contractList/all/light?groupId=1&
↪contractStatus=2
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "id": 2,
            "contractPath": "/",
            "contractName": "HeHe",
            "contractStatus": 1,
            "groupId": "group",
            "contractSource": "cHJhZ21hIHNvbGlkaXR5IICB9Cn0=",
            "contractAddress": null,
            "deployTime": null,
            "description": null,
            "createTime": "2019-06-10 16:42:50",
```

(下页继续)

(续上页)

```
            "modifyTime": "2019-06-10 16:42:52"
        }
    ],
    "totalCount": 1
}
```

1.12. 根据id获取单个合约

接口描述

根据合约id获取单个合约

接口URL

http://localhost:5002/WeBASE-Front/contract/findOne/{contractId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contract/findOne/1
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "id": 1,
            "contractPath": "/",
            "contractName": "HeHe",
            "contractStatus": 1,
            "groupId": "group",
            "contractSource": "cHJhZ21hIHNvbGlkaXR5IICB9Cn0=",
            "contractAbi": "",
            "contractBin": "",
            "bytecodeBin": null,
            "contractAddress": null,
            "deployTime": null,
            "description": null,
            "createTime": "2019-06-10 16:42:50",
            "modifyTime": "2019-06-10 16:42:52"
```

(下页继续)

(续上页)

```
        }
    ]
}
```

1.13. 获取合约路径列表

接口描述

获取合约路径列表

接口URL

http://localhost:5002/WeBASE-Front/contract/findPathList/{groupId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contract/findPathList/group
```

响应参数

1）参数表

2）数据格式

```
[
  {
    "groupId": "group",
    "contractPath": "/",
    "createTime": null,
    "modifyTime": "2021-07-29 14:34:52"
  },
  {
    "groupId": "group",
    "contractPath": "Asset",
    "createTime": null,
    "modifyTime": "2021-07-27 10:43:59"
  },
  {
     "groupId": "group",
    "contractPath": "template",
    "createTime": "2021-07-20 11:31:18",
    "modifyTime": "2021-07-20 11:31:18"
  }
]
```

1.14. 保存合约路径

接口描述

保存合约路径

接口URL

http://localhost:5002/WeBASE-Front/contract/addContractPath

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contract/addContractPath
```

```
{
  "contractPath": "string",
  "groupId": "string"
}
```

响应参数

1）参数表

2）数据格式

```
{
  "groupId": "group",
  "contractPath": "test",
  "createTime": "2021-07-29 14:26:54",
  "modifyTime": "2021-07-29 14:26:54"
}
```

1.15. 删除合约路径

接口描述

删除合约路径（不删除目录下合约）

接口URL

http://localhost:5002/WeBASE-Front/contract/deletePath/{groupId}/{contractPath}

调用方法

HTTP DELETE

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contract/deletePath/group/test
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "data": {},
    "message": "success"
}
```

1.16. 根据合约路径批量删除合约

接口描述

根据合约路径批量删除合约

接口URL

http://localhost:5002/WeBASE-Front/contract/batch/{groupId}/{contractPath}

调用方法

HTTP DELETE

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contract/batch/group/test
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "data": {},
    "message": "success"
}
```

1.17. 注册cns接口

接口描述

注册cns

接口URL

http://localhost:5002/WeBASE-Front/contract/registerCns

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "abiInfo": [
    {
      "anonymous": true,
      "constant": true,
      "inputs": [
        {
          "components": [
            null
          ],
          "dynamic": true,
          "indexed": true,
          "internalType": "string",
          "name": "string",
          "type": "string",
          "typeAsString": "string"
        }
      ],
      "methodSignatureAsString": "string",
      "name": "string",
      "outputs": [
        {
          "components": [
            null
          ],
          "dynamic": true,
          "indexed": true,
          "internalType": "string",
          "name": "string",
```

(下页继续)

(续上页)


```
            "type": "string",
            "typeAsString": "string"
          }
        ],
        "payable": true,
        "stateMutability": "string",
        "type": "string"
      }
    ],
    "cnsName": "string",
    "contractAddress": "string",
    "contractName": "string",
    "contractPath": "string",
    "groupId": "string",
    "saveEnabled": true,
    "signUserId": "string",
    "userAddress": "string",
    "version": "string"
}
```


响应参数

1）参数表

2）数据格式


```
{
    "code": 0,
    "message": "success"
}
```


1.18. 根据合约地址获取cns信息

接口描述

根据合约地址获取cns信息，返回改合约地址最新的cns信息

接口URL

http://localhost:5002/WeBASE-Front/contract/findCns

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式


```
http://localhost:5002/WeBASE-Front/contract/findCns
```

```
{
  "groupId": "group",
  "contractAddress": "0xe46c1a681811ee78079b48a956ead6d9dd10bf6a"
}
```

响应参数

1）参数表

2）数据格式

```
{
  "code": 0,
  "message": "success"
  "data": {
    "groupId": "group",
    "contractPath": "/",
    "contractName": "Hello",
    "cnsName": "Hello",
    "version": "v0.4",
    "contractAddress": "0xcaff8fdf1d461b91c7c8f0ff2af2f79a80bc189e",
    "contractAbi":"[{\"constant\":true,\"inputs\":[],\"name\":\"get\",\"outputs\":[
↪{\"name\":\"\",\"type\":\"string\",\"type0\":null,\"indexed\":false}],\"type\":\
↪"function\",\"payable\":false,\"stateMutability\":\"view\"},{\"constant\":false,\
↪"inputs\":[{\"name\":\"n\",\"type\":\"string\",\"type0\":null,\"indexed\":false}
↪],\"name\":\"set\",\"outputs\":[],\"type\":\"function\",\"payable\":false,\
↪"stateMutability\":\"nonpayable\"},{\"constant\":false,\"inputs\":[{\"name\":\
↪"name\",\"type\":\"string\",\"type0\":null,\"indexed\":false}],\"name\":\
↪"SetName\",\"outputs\":null,\"type\":\"event\",\"payable\":false,\
↪"stateMutability\":null}]",
    "createTime": "2020-12-30 16:32:28",
    "modifyTime": "2020-12-30 16:32:28"
  }
}
```

1.19. 检测是否已配置Liquid环境

接口描述

通过cargo命令和liquid命令，检测WeBASE-Front所在主机是否已配置Liquid环境

接口URL

http://localhost:5002/WeBASE-Front/contract/liquid/check

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contract/liquid/check
```

响应参数

1）参数表

2）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": null
}
```

1.20. 编译liquid合约

接口描述

传入合约源码、编译liquid合约，并返回编译得到的abi和bin。

由于liquid合约类似于rust编译，耗时比solidity更长（3分钟左右），因此接口返回状态为“编译中”时，后台将异步执行编译任务，通过轮询/contract/liquid/compile/check接口可以获取最新的编译结果

接口URL

http://localhost:5002/WeBASE-Front/contract/liquid/compile

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
        "groupId": "group0",
        "contractName": "LiquidHelloWorld",
        "contractPath": "/",
        "contractSource":
↪"IyFbY2ZnX2F0dHIobm90KGZlYXR1cmUgPSAic3RkIiksIG5vX3N0ZCldCgp1c2UgbGlxdWlkOjpzdG9yYWdlOwp1c2UgbG
↪",
        "contractAbi": "[{\"inputs\":[],\"type\":\"constructor\"},{\"constant\
↪":true,\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"internalType\":\"string\",\
↪"type\":\"string\"}],\"type\":\"function\"},{\"conflictFields\":[{\"kind\":0,\
↪"path\":[],\"read_only\":false,\"slot\":0}],\"constant\":false,\"inputs\":[{\
↪"internalType\":\"string\",\"name\":\"name\",\"type\":\"string\"}],\"name\":\
↪"set\",\"outputs\":[],\"type\":\"function\"}]",
  "isWasm": true
}
```

响应参数

1）参数表

2）数据格式

状态为编译中时：（编译中时，后台将异步执行编译任务，通过轮询/contract/liquid/compile/check接口可以获取最新的编译结果）

```
{
  "code": 0,
  "message": "success"
  "data": {
    "groupId": "group",
    "contractPath": "",
    "contractName": "Hello",
    "status": "1",
    "bin": null,
    "abi": null,
    "createTime": "2020-12-30 16:32:28",
    "modifyTime": "2020-12-30 16:32:28"
  }
}
```

状态为编译成功时：

```
{
  "code": 0,
  "message": "success"
  "data": {
    "groupId": "group",
    "contractPath": "",
    "contractName": "Hello",
    "status": "2",
    "bin": "",  //bin过长，此处略
    "abi":"[{\"constant\":true,\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"name\
↪":\"\",\"type\":\"string\",\"type0\":null,\"indexed\":false}],\"type\":\
↪"function\",\"payable\":false,\"stateMutability\":\"view\"},{\"constant\":false,\
↪"inputs\":[{\"name\":\"n\",\"type\":\"string\",\"type0\":null,\"indexed\":false}
↪],\"name\":\"set\",\"outputs\":[],\"type\":\"function\",\"payable\":false,\
↪"stateMutability\":\"nonpayable\"},{\"constant\":false,\"inputs\":[{\"name\":\
↪"name\",\"type\":\"string\",\"type0\":null,\"indexed\":false}],\"name\":\
↪"SetName\",\"outputs\":null,\"type\":\"event\",\"payable\":false,\
↪"stateMutability\":null}]",
    "createTime": "2020-12-30 16:32:28",
    "modifyTime": "2020-12-30 16:32:28"
  }
}
```

1.20. 查询liquid合约编译进度

接口描述

根据群组ID，合约路径，合约名获取liquid合约的编译状态

接口URL

http://localhost:5002/WeBASE-Front/contract/liquid/compile/check

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
        "groupId": "group0",
        "contractName": "LiquidHelloWorld",
        "contractPath": "/"
}
```

响应参数

1）参数表

2）数据格式

状态为编译中时，轮询当前接口直到状态为编译成功、编译失败，status=1：

```
{
  "code": 0,
  "message": "success"
  "data": {
    "groupId": "group",
    "contractPath": "/",
    "contractName": "Hello",
    "status": "1",
    "bin": null,
    "abi": null,
    "createTime": "2020-12-30 16:32:28",
    "modifyTime": "2020-12-30 16:32:28"
  }
}
```

状态为编译成功时，status=2：

```
{
  "code": 0,
  "message": "success"
  "data": {
    "groupId": "group",
    "contractPath": "/",
    "contractName": "Hello",
    "status": "2",
    "bin": "",  //bin过长，此处略
    "abi":"[{\"constant\":true,\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"name\
↪":\"\",\"type\":\"string\",\"type0\":null,\"indexed\":false}],\"type\":\
↪"function\",\"payable\":false,\"stateMutability\":\"view\"},{\"constant\":false,\
↪"inputs\":[{\"name\":\"n\",\"type\":\"string\",\"type0\":null,\"indexed\":false}
↪],\"name\":\"set\",\"outputs\":[],\"type\":\"function\",\"payable\":false,\
↪"stateMutability\":\"nonpayable\"},{\"constant\":false,\"inputs\":[{\"name\":\
↪"name\",\"type\":\"string\",\"type0\":null,\"indexed\":false}],\"name\":\
↪"SetName\",\"outputs\":null,\"type\":\"event\",\"payable\":false,\
↪"stateMutability\":null}]",
    "createTime": "2020-12-30 16:32:28",
```

(下页继续)

(续上页)

```
    "modifyTime": "2020-12-30 16:32:28"
  }
}
```

12.3.2 2. 密钥接口

2.1. 创建私钥接口

接口描述

通过调用此接口获取公私钥对和对应账户信息

接口URL

http://localhost:5002/WeBASE-Front/privateKey?type={type}&userName={userName}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
// 本地用户
http://localhost:5002/WeBASE-Front/privateKey?type=0&userName=test
// 外部用户
http://localhost:5002/WeBASE-Front/privateKey?type=2&signUserId=0x123&appId=2
```

响应参数

1）数据格式 本地用户时：

```
{
  "groupId": "group",
  "address": "0x2007e1430f41f75c850464307c0994472bd92ee0",
  "publicKey":
↪"0x9bd35211855f9f8de22d8a8da7f30d35d62ab2c3d36ea5162008fcbb9faff4d83809f7033deb20049bf51e081105
↪",
  "privateKey": "42caa160cadcb635381b980ddd981171c862d3105981fe92d6db330f30615f21",
  "userName": "test",
  "type": 0,
  "signUserId": null, // 本地用户则为空
  "appId": null // 本地用户则为空
}
```

外部用户时（来自WeBASE-Sign）：

```
{
  "groupId": "group",
  "address": "0xef5afe7d9a7516cd36b5b2471a3fbb05d3e8a846",
  "publicKey":
↪"0461e78631ab8428c1be815a4543da8684db13cd2d9a0593e053184dbd29d08f38131e060bc8d1a1ef5f4290b26acc
↪",
  "privateKey": null, // 默认不返回
  "userName": null, //外部用户则为空
  "type": 2,
  "signUserId": "0x123",
  "appId": "1"
}
```

2.2. 导入私钥接口

接口描述

导入私钥信息，并返回对应的公钥及用户地址

接口URL

http://localhost:5002/WeBASE-Front/privateKey/import?privateKey={privateKey}&userName={userName}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/privateKey/import?
↪privateKey=8cf98bd0f37fb0984ab43ed6fc2dcdf58811522af7e4a3bedbe84636a79a501c&
↪userName=lili
```

响应参数

1）数据格式

```
{
  "address": "0x2e8ff65fb1b2ce5b0c9476b8f8beb221445f42ee",
  "publicKey":
↪"0x1c7073dc185af0644464b178da932846666a858bc492450e9e94c77203428ed54e2ce45679dbb36bfed714dbe055
↪",
  "privateKey": "8cf98bd0f37fb0984ab43ed6fc2dcdf58811522af7e4a3bedbe84636a79a501c",
  "userName": "lili",
  "type": 0,
  "appId": "string",
  "signUserId": "string"
}
```

2.3. 获取本地公私钥列表接口

接口描述

返回本地公私钥信息列表

接口URL

http://localhost:5002/WeBASE-Front/privateKey/localKeyStores

调用方法

HTTP GET

请求参数

1）参数表

无

2）数据格式

```
http://localhost:5002/WeBASE-Front/privateKey/localKeyStores
```

响应参数

1）数据格式

```
[
  {
    "address": "string",
    "appId": "string",
    "privateKey": "string",
    "publicKey": "string",
    "signUserId": "string",
    "type": 0,
    "userName": "string"
  }
]
```

2.4. 删除公私钥接口

接口描述

支持前置的控制台通过用户地址删除公私钥信息

接口URL

http://localhost:5002/WeBASE-Front/privateKey/{address}

调用方法

HTTP DELETE

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/privateKey/
↪0x2e8ff65fb1b2ce5b0c9476b8f8beb221445f42ee
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": null
}
```

2.5. 导入.pem私钥用户

接口描述

导入.pem格式的私钥

接口URL

http://localhost:5002/WeBASE-Front/privateKey/importPem

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/privateKey/importPem
```

```
{
  "groupId": "string",
  "pemContent": "string",
  "userName": "string"
}
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": null
}
```

2.6. 导入.p12私钥用户

接口描述

导入.p12格式的私钥

接口URL

http://localhost:5002/WeBASE-Front/privateKey/importP12

调用方法

HTTP POST | Content-type: form-data

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/privateKey/importP12
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": null
}
```

2.7. 导入私钥到WeBASE-Sign

接口描述

导入私钥到WeBASE-Sign，其中privateKey经过Base64加密

接口URL

http://localhost:5002/WeBASE-Front/privateKey/importWithSign

调用方法

HTTP POST | Content-type: application/json

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/privateKey/importWithSign
```

```
{
  "appId": "string",
  "groupId": "string",
  "privateKey": "string",
  "signUserId": "string"
}
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": null
}
```

2.8. 对哈希签名

接口描述

计算HASH和签名值

接口URL

http://localhost:5002/WeBASE-Front/privateKey/signMessageHash

调用方法

HTTP POST | Content-type: application/json

请求参数

1）参数表

2）数据格式

```
{
  "groupId": "string",
  "messageHash": "string",
  "signUserId": "string"
}
```

响应参数

1）数据格式

```
{
  "v": 0,
  "r": "0x2a76a45bcf1113615f796cc01b23c57f81f20ce79500080bb34c7994ed04de06",
  "s": "0x4f111eb37720e2618d8906368c825fd3cbe89b2781cb678efafb42399594a580",
  "p":
↪"0x4405f9d5d6ccff709b6543bc8ac24cbb649d3267a66db19ab8f85f3b884a8505f086c581490e7e50558879abde9c4
↪"
}
```

2.9. 导出pem私钥

接口描述

从本地或WeBASE-Sign导出pem私钥文件

接口URL

http://localhost:5002/WeBASE-Front/privateKey/exportPem

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/privateKey/exportPem
```

```
{
  "groupId": "string",
  "p12Password": "string",
  "signUserId": "string",
  "userAddress": "string"
}
```

响应参数

1）参数表

2）数据格式

```
headers: content-disposition: attachment;filename*=UTF-8''111_
↪0x0421975cf4a5b27148f65de11cd9d8559a1bbbd9.pem

{
-----BEGIN PRIVATE KEY-----
MIGNAgEAMBAGByqGSM49AgEGBSuBBAAKBHYwdAIBAQQg91Aha3x2UdpN2Sg5C5Wh
7Y8YwIYC5NTNtfQT1yp7hKWgBwYFK4EEAAqhRANCAAQ9SEdu1kLpLXVmayqax7N+
Pf/ATWx5jJIJIiQF6/BIiuORZrZb/M04F1xtGyVuQjQjbYVgyjNDUilj14OlZhXo
-----END PRIVATE KEY-----
}
```

2.10. 导出p12私钥

接口描述

从本地或WeBASE-Sign导出p12私钥文件

接口URL

http://localhost:5002/WeBASE-Front/privateKey/exportP12

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/privateKey/exportP12
```

```
{
  "groupId": "string",
  "p12Password": "string",
  "signUserId": "string",
  "userAddress": "string"
}
```

响应参数

1）参数表

2）数据格式

```
headers:  content-disposition: attachment;filename*=UTF-8''111_
↪0x0421975cf4a5b27148f65de11cd9d8559a1bbbd9.p12

{
// 二进制流
}
```

2.11. 获取WeBASE-Sign私钥用户信息

接口描述

从WeBASE-Sign获取私钥信息

接口URL

http://localhost:5002/WeBASE-Front/privateKey/userInfoWithSign

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/privateKey/userInfoWithSign?
↪signUserId=d1edf538748b4d899f251b5d746ec62f&returnPrivateKey=false
```

响应参数

1）参数表

2）数据格式

```
{
  "signUserId": "d1edf538748b4d899f251b5d746ec62f",
  "appId": "1",
  "address": "0x2b6e2f9d25bdeeca395bd743099ef50097043aaa",
  "publicKey":
↪"044ddb8e92cf9dd96c1767b5e9c7e55e749304e88e28a8ec4fb7059c3388590ce83f2786510898464dfeb79771656a5
↪",
  "privateKey": "",
  "description": null,
  "encryptType": 1
}
```

12.3.3 3. web3接口

3.1. 获取块高接口

接口描述

获取节点最新块高

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/blockNumber

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/blockNumber
```

响应参数

1）数据格式

```
232
```

3.2. 根据块高获取块信息接口

接口描述

通过块高获取块信息

接口URL

http://localhost:5002/WeBASE-Front/group/web3/blockByNumber/{blockNumber}

调用方法

HTTP GET

请求参数

1. 参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/blockByNumber/2
```

响应参数

1）数据格式

```
{
  "number": 2,
  "hash": "0xf58f4f43b3761f4863ad366c4a7e2a812ed68df9f7bcad6b502fd544665e7625",
  "parentHash": "0x489ee0c00527879f7e2470bde7b62e9ea30fadb242bcbd9ba582d0dee4958e2f
↪",
  "logsBloom":
↪"0x00000000000000000000000000000000000000000000000000000000000000000000000000000000
↪",
  "transactionsRoot":
↪"0xade02313d3200f38dfc18cdc368241cf4c1cb7e72011edf847ec216efa43a99e",
  "receiptsRoot":
↪"0xb7c4b856c9335bd345c0b022057567a2223218e7a7dab822981467b29ac1e326",
  "dbHash": "0x8cbc3f2c0e35a71738909e3b388efa6697084b05badd3a3bd3c64f0575c78c15",
  "stateRoot": "0x8cbc3f2c0e35a71738909e3b388efa6697084b05badd3a3bd3c64f0575c78c15
↪",
  "sealer": "0x2",
  "sealerList": [

↪"06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b35067c1685d4343e6ad84b45
↪",

↪"65bc44d398d99d95a9d404aa16e4bfbc2f9ebb40f20439ddef8575a139dc3a80310cfc98a035bd59a67cc5f659f519
↪",

↪"95efafa5197796e7edf647191de83f4259d7cbb060f4bac5868be474037f49144d581c15d8aef9b07d78f18041a5f4
↪",

↪"fe57d7b39ed104b4fb2770ae5aad7946bfd377d0eb91ab92a383447e834c3257dec56686551d08178f2d5f40d9fad6
↪"
  ],
  "extraData": [],
  "gasLimit": "0",
  "gasUsed": "0",
  "timestamp": "1619424150450",
  "signatureList": [
    {
      "index": "0x1",
      "signature":
↪"0xa05b5d220e41051fa80f212884d9cdda3a8973a4ac2d2dc74e42db32b459e5f595a61e30266f0e8bf7edb065ff10
↪"
    },
    {
      "index": "0x3",
      "signature":
↪"0x48bb622e24b18f4eb601137d44b5d86e2c287be3aa849dce1768e87e75308ba69391e62f90d76bf84b9b6eb1509d
↪"
    },
    {
      "index": "0x2",
      "signature":
↪"0x3a3cc8c20c5cdbb6431ec1f749d94662670e87442debef119a4e6469ccb16a60a152d32754ab833efa513839b076
↪"
    }
  ],
  "transactions": [
    {
      "blockHash":
↪"0xf58f4f43b3761f4863ad366c4a7e2a812ed68df9f7bcad6b502fd544665e7625",
```

(下页继续)

(续上页)


```
      "blockNumber": 2,
      "from": "0x9d75e0ee66cfef16897b601624b60413d988ae7d",
      "gas": "0x419ce0",
      "hash": "0x69ced0162a0c3892e4eaa3091b831ac3aaeb772c062746b20891ceaf8a4fb429",
      "input":
↪"0x6080604052348015610010576000080fd5b506103e380610020600039600f3006080604052600436106100575760
↪",
      "nonce": "0x3460c30bd3e4e88a31d6d033b3a172859bf003258e9606fd63fb0d91f91f4e6",
      "to": "0x0000000000000000000000000000000000000000",
      "transactionIndex": "0x0",
      "value": "0x0",
      "gasPrice": "0x51f4d5c00",
      "blockLimit": "0x1f5",
      "chainId": "0x1",
      "groupId": "0x1",
      "extraData": "0x",
      "signature": {
        "r": "0x3416723318505669cca91689b213ff08ffb96d538210a0f691cfcfa9d529462b",
        "s": "0xd3642f19c228e2e86689de9efc19ecbb68378a6bb7c219984431e93d60c89124",
        "v":
↪"0xc7935c199b680452eb37911856282b9c820322fd5fdec8a06b48cc3df4e8ed7d3d66a5adcc134cca609146ec0aed
↪",
        "signature":
↪"0x3416723318505669cca91689b213ff08ffb96d538210a0f691cfcfa9d529462bd3642f19c228e2e86689de9efc19
↪"
      }
    }
  ]
}
```


3.3. 根据块hash获取块信息接口

接口描述

通过块hash获取块信息

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/blockByHash/{blockHash}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/blockByHash/
↪0xf58f4f43b3761f4863ad366c4a7e2a812ed68df9f7bcad6b502fd544665e7625
```

响应参数

1）数据格式


```
{{
  "number": 2,
  "hash": "0xf58f4f43b3761f4863ad366c4a7e2a812ed68df9f7bcad6b502fd544665e7625",
  "parentHash": "0x489ee0c00527879f7e2470bde7b62e9ea30fadb242bcbd9ba582d0dee4958e2f
↪",
  "logsBloom":
↪"0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
↪",
  "transactionsRoot":
↪"0xade02313d3200f38dfc18cdc368241cf4c1cb7e72011edf847ec216efa43a99e",
  "receiptsRoot":
↪"0xb7c4b856c9335bd345c0b022057567a2223218e7a7dab822981467b29ac1e326",
  "dbHash": "0x8cbc3f2c0e35a71738909e3b388efa6697084b05badd3a3bd3c64f0575c78c15",
  "stateRoot": "0x8cbc3f2c0e35a71738909e3b388efa6697084b05badd3a3bd3c64f0575c78c15
↪",
  "sealer": "0x2",
  "sealerList": [

↪"06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b35067c1685d4343e6ad84b45
↪",

↪"65bc44d398d99d95a9d404aa16e4bfbc2f9ebb40f20439ddef8575a139dc3a80310cfc98a035bd59a67cc5f659f519
↪",

↪"95efafa5197796e7edf647191de83f4259d7cbb060f4bac5868be474037f49144d581c15d8aef9b07d78f18041a5f4
↪",

↪"fe57d7b39ed104b4fb2770ae5aad7946bfd377d0eb91ab92a383447e834c3257dec56686551d08178f2d5f40d9fad6
↪"
  ],
  "extraData": [],
  "gasLimit": "0",
  "gasUsed": "0",
  "timestamp": "1619424150450",
  "signatureList": [
    {
      "index": "0x1",
      "signature":
↪"0xa05b5d220e41051fa80f212884d9cdda3a8973a4ac2d2dc74e42db32b459e5f595a61e30266f0e8bf7edb065ff10
↪"
    },
    {
      "index": "0x3",
      "signature":
↪"0x48bb622e24b18f4eb601137d44b5d86e2c287be3aa849dce1768e87e75308ba69391e62f90d76bf84b9b6eb1509d
↪"
    },
    {
      "index": "0x2",
      "signature":
↪"0x3a3cc8c20c5cdbb6431ec1f749d94662670e87442debef119a4e6469ccb16a60a152d32754ab833efa513839b076
↪"
    }
  ],
  "transactions": [
    {
      "blockHash":
↪"0xf58f4f43b3761f4863ad366c4a7e2a812ed68df9f7bcad6b502fd544665e7625",
```


(下页继续)

(续上页)


```
      "blockNumber": 2,
      "from": "0x9d75e0ee66cfef16897b601624b60413d988ae7d",
      "gas": "0x419ce0",
      "hash": "0x69ced0162a0c3892e4eaa3091b831ac3aaeb772c062746b20891ceaf8a4fb429",
      "input":
↪"0x6080604052348015610010576000fd5b506103e3806100206000396000f3006080604052600436106100575760
↪",
      "nonce": "0x3460c30bd3e4e88a31d6d033b3a172859bf003258e9606fd63fb0d91f91f4e6",
      "to": "0x0000000000000000000000000000000000000000",
      "transactionIndex": "0x0",
      "value": "0x0",
      "gasPrice": "0x51f4d5c00",
      "blockLimit": "0x1f5",
      "chainId": "0x1",
      "groupId": "0x1",
      "extraData": "0x",
      "signature": {
        "r": "0x3416723318505669cca91689b213ff08ffb96d538210a0f691cfcfa9d529462b",
        "s": "0xd3642f19c228e2e86689de9efc19ecbb68378a6bb7c219984431e93d60c89124",
        "v":
↪"0xc7935c199b680452eb37911856282b9c820322fd5fdec8a06b48cc3df4e8ed7d3d66a5adcc134cca609146ec0aed
↪",
        "signature":
↪"0x3416723318505669cca91689b213ff08ffb96d538210a0f691cfcfa9d529462bd3642f19c228e2e86689de9efc19
↪"
      }
    }
  ]
}
```


3.4. 获取块中交易个数接口

接口描述

根据块高获取该块中的交易个数

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/blockTransCnt/{blockNumber}

调用方法

HTTP GET

请求参数

1. 参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/blockTransCnt/100
```

响应参数

1）数据格式

```
2
```

3.5. 获取PbftView接口

接口描述

通过调用此接口获取PbftView

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/pbftView

调用方法

HTTP GET

请求参数

1. 参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/pbftView
```

响应参数

1）数据格式

```
{
    160565
}
```

3.6. 获取交易回执接口

接口描述

根据交易hash获取交易回执

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/transactionReceipt/{transHash}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/transactionReceipt/
↪0x69ced0162a0c3892e4eaa3091b831ac3aaeb772c062746b20891ceaf8a4fb429
```

响应参数

2）数据格式


```
{
  "transactionHash":
↪"0x69ced0162a0c3892e4eaa3091b831ac3aaeb772c062746b20891ceaf8a4fb429",
  "transactionIndex": "0x0",
  "root": "0x8cbc3f2c0e35a71738909e3b388efa6697084b05badd3a3bd3c64f0575c78c15",
  "blockNumber": "2",
  "blockHash": "0xf58f4f43b3761f4863ad366c4a7e2a812ed68df9f7bcad6b502fd544665e7625
↪",
  "from": "0x9d75e0ee66cfef16897b601624b60413d988ae7d",
  "to": "0x0000000000000000000000000000000000000000",
  "gasUsed": "316449",
  "contractAddress": "0xa8af0ee580d8af674a60341030ddbc45431bc235",
  "logs": [],
  "logsBloom":
↪"0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
↪",
  "status": "0x0",
  "statusMsg": "None",
  "input":
↪"0x608060405234801561001057600080fd5b506103e38061002060003960 00f30060806040526004361061005757600
↪",
  "output": "0x",
  "txProof": null,
  "receiptProof": null,
  "message": null,
  "statusOK": true
}
```


3.7. 根据交易hash获取交易信息接口

接口描述

> 根据交易hash获取交易信息

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/transaction/{transHash}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/1/web3/transaction/
↪0x69ced0162a0c3892e4eaa3091b831ac3aaeb772c062746b20891ceaf8a4fb429
```

响应参数

1）数据格式

```
{
  "blockHash": "0xf58f4f43b3761f4863ad366c4a7e2a812ed68df9f7bcad6b502fd544665e7625
↪",
  "blockNumber": 2,
  "from": "0x9d75e0ee66cfef16897b601624b60413d988ae7d",
  "gas": "4300000",
  "hash": "0x69ced0162a0c3892e4eaa3091b831ac3aaeb772c062746b20891ceaf8a4fb429",
  "input":
↪"0x6080604052348015610010576000080fd5b506103e380610020600039600f3006080604052600436106100575760
↪",
  "nonce": "0x3460c30bd3e4e88a31d6d033b3a172859bf003258e9606fd63fb0d91f91f4e6",
  "to": "0x0000000000000000000000000000000000000000",
  "transactionIndex": "0x0",
  "value": "0x0",
  "gasPrice": "22000000000",
  "blockLimit": "0x1f5",
  "chainId": "0x1",
  "groupId": "1",
  "extraData": "0x",
  "signature": {
    "r": "0x3416723318505669cca91689b213ff08ffb96d538210a0f691cfcfa9d529462b",
    "s": "0xd3642f19c228e2e86689de9efc19ecbb68378a6bb7c219984431e93d60c89124",
    "v":
↪"0xc7935c199b680452eb37911856282b9c820322fd5fdec8a06b48cc3df4e8ed7d3d66a5adcc134cca609146ec0aed
↪",
    "signature":
↪"0x3416723318505669cca91689b213ff08ffb96d538210a0f691cfcfa9d529462bd3642f19c228e2e86689de9efc19
↪"
  }
}
```

3.8. 获取合约二进制代码接口

接口描述

获取指定块高区块指定合约地址的二进制代码

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/code/{address}/{blockNumber}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/code/
↪0x0000000000000000000000000000000000000000/1
```

响应参数

1）数据格式

```
{
    0xxxx
}
```

3.9. 获取总交易数

接口描述

获取总交易数量

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/transaction-total

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/1/web3/transaction-total
```

响应参数

1）参数表

2）数据格式

```
{
    "txSum": 125,
    "blockNumber": 125,
    "blockNumberRaw": "0x7d",
    "txSumRaw": "0x7d"
}
```

3.10. 获取群组内的共识状态信息接口

接口描述

返回指定群组内的共识状态信息

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/consensusStatus

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/consensusStatus
```

响应参数

1）数据格式

```
{
  "baseConsensusInfo": {
    "nodeNum": "4",
    "consensusedBlockNumber": "233",
    "highestblockNumber": "232",
    "groupId": "1",
    "protocolId": "65544",
    "accountType": "1",
    "cfgErr": "false",
    "omitEmptyBlock": "true",
    "nodeId":
↪"fe57d7b39ed104b4fb2770ae5aad7946bfd377d0eb91ab92a383447e834c3257dec56686551d08178f2d5f40d9fad6
↪",
    "allowFutureBlocks": "true",
    "connectedNodes": "3",
    "currentView": "102303",
    "toView": "102303",
    "leaderFailed": "false",
    "highestblockHash":
↪"0x7f0885a7188bd5c1b3f8c182e00be0e63a3b1653b12752cf8501ddac995e6efc",
```

(下页继续)

(续上页)


```
    "leaderId": null,
    "leaderIdx": null,
    "node_index": "3",
    "node index": null,
    "max_faulty_leader": "1",
    "sealer.": [

↪"06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b35067c1685d4343e6ad84b45c
↪",

↪"65bc44d398d99d95a9d404aa16e4bfbc2f9ebb40f20439ddef8575a139dc3a80310cfc98a035bd59a67cc5f659f519e
↪",

↪"95efafa5197796e7edf647191de83f4259d7cbb060f4bac5868be474037f49144d581c15d8aef9b07d78f18041a5f43
↪",

↪"fe57d7b39ed104b4fb2770ae5aad7946bfd377d0eb91ab92a383447e834c3257dec56686551d08178f2d5f40d9fad61
↪"
    ]
  },
  "viewInfos": [
    {
      "nodeId":
↪"06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b35067c1685d4343e6ad84b45c
↪",
      "view": "102300"
    },
    {
      "nodeId":
↪"65bc44d398d99d95a9d404aa16e4bfbc2f9ebb40f20439ddef8575a139dc3a80310cfc98a035bd59a67cc5f659f519e
↪",
      "view": "102301"
    },
    {
      "nodeId":
↪"95efafa5197796e7edf647191de83f4259d7cbb060f4bac5868be474037f49144d581c15d8aef9b07d78f18041a5f43
↪",
      "view": "102302"
    },
    {
      "nodeId":
↪"fe57d7b39ed104b4fb2770ae5aad7946bfd377d0eb91ab92a383447e834c3257dec56686551d08178f2d5f40d9fad61
↪",
      "view": "102303"
    }
  ]
}
```


3.11. 获取群组列表接口

接口描述

返回群组列表

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/groupList

调用方法

HTTP GET

请求参数

1）参数表

无

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/groupList
```

响应参数

1）数据格式

```
[
  "1",
  "2"
]
```

3.12. 获取观察及共识节点列表

接口描述

返回指定群组内的共识节点和观察节点列表

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/groupPeers

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/groupPeers
```

响应参数

1）数据格式

```
[
↪"d822165959a0ed217df6541f1a7dd38b79336ff571dd5f8f85ad76f3e7ec097e1eabd8b03e4a757fd5a9fb0eea905a
↪",
↪"adfa2f9116d7ff68e0deb75307fa1595d636bf097ad1de4fb55cff00e4fef40b453abb30388aa2112bf5cd4c987afe
↪",
↪"552398be0eef124c000e632b0b76a48c52b6cfbd547d92c15527c2d1df15fab2bcded48353db22526c3540e4ab2027
↪",
↪"dde0bbf5eb3a731e6da861586e98e088e16e6fdd9afae2f2c213cead20a4f5eaa3910042b70d62266d2350d98a43c1
↪"
]
```

3.13. 获取群组内观察节点列表

接口描述

返回指定群组内的观察节点列表

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/observerList

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/observerList
```

响应参数

1）数据格式

```
[
↪"d822165959a0ed217df6541f1a7dd38b79336ff571dd5f8f85ad76f3e7ec097e1eabd8b03e4a757fd5a9fb0eea905a
↪"
]
```

3.14. 获取已连接的P2P节点信息

接口描述

返回指定群组内已连接的P2P节点信息

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/peers

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/peers
```

响应参数

1）数据格式

```
{
  "blockNumber": 0,
  "endPoint": "string",
  "genesisHash": "string",
  "groupNodeIDInfo": [
    {
      "group": "string",
      "nodeIDList": [
        "string"
      ]
    }
  ],
  "latestHash": "string",
  "nodeID": "string",
  "p2pNodeID": "string",
  "peers": [
    {
      "endPoint": "string",
      "groupNodeIDInfo": [
        {
          "group": "string",
          "nodeIDList": [
            "string"
          ]
        }
      ],
      "p2pNodeID": "string"
    }
  ]
}
```

3.15. 获取群组内正在处理的交易数

接口描述

获取群组内正在处理的交易数

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/pending-transactions-count

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/pending-transactions-count
```

响应参数

1）数据格式

```
0
```

3.16. 获取共识节点接口

接口描述

返回群组内共识节点列表

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/sealerList

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/sealerList
```

响应参数

1）数据格式

```
[

↪"d822165959a0ed217df6541f1a7dd38b79336ff571dd5f8f85ad76f3e7ec097e1eabd8b03e4a757fd5a9fb0eea905ac
↪",

↪"adfa2f9116d7ff68e0deb75307fa1595d636bf097ad1de4fb55cff00e4fef40b453abb30388aa2112bf5cd4c987afe2
↪",

↪"552398be0eef124c000e632b0b76a48c52b6cfbd547d92c15527c2d1df15fab2bcded48353db22526c3540e4ab2027
↪",

↪"dde0bbf5eb3a731e6da861586e98e088e16e6fdd9afae2f2c213cead20a4f5eaa3910042b70d62266d2350d98a43c1
↪"
]
```

3.17 区块/交易

接口描述

> 如果输入块高就返回区块信息，如果输入交易hash就返回交易信息

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/search?input={inputValue}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/search?input=123
```

响应参数

1）数据格式

```
{
  "number": 123,
  "hash": "0xa12a4a8d26a15fcff6f67acf94a3fb2c08d4476806009be40fe4026fd622419d",
  "parentHash": "0x898c6b1657d4eb26307b39f248e471498975eb820671c7540fa516e6c7b411e8
↪",
  "logsBloom":
↪"0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
↪",
  "transactionsRoot":
↪"0xa94acacd692cf3b23e19b429ce89c2016020b2264854fd609ba2b16f0c7d5f73",
  "receiptsRoot":
↪"0x9529b51c1f7ddd178ffd4a65d146b50cfc0beb405a808677dfddd694cf555a09",
```

(下页继续)

(续上页)


```
  "dbHash": "0x3047354b6776fd1f02911f2c4fc4aa1b215b3df53cc7504a71b5f3bfe12eca3a",
  "stateRoot": "0x3047354b6776fd1f02911f2c4fc4aa1b215b3df53cc7504a71b5f3bfe12eca3a
↪",
  "sealer": "0x1",
  "sealerList": [

↪"06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b35067c1685d4343e6ad84b45
↪",

↪"65bc44d398d99d95a9d404aa16e4bfbc2f9ebb40f20439ddef8575a139dc3a80310cfc98a035bd59a67cc5f659f519
↪",

↪"95efafa5197796e7edf647191de83f4259d7cbb060f4bac5868be474037f49144d581c15d8aef9b07d78f18041a5f4
↪",

↪"fe57d7b39ed104b4fb2770ae5aad7946bfd377d0eb91ab92a383447e834c3257dec56686551d08178f2d5f40d9fad6
↪"
  ],
  "extraData": [],
  "gasLimit": "0",
  "gasUsed": "0",
  "timestamp": "1621928850887",
  "signatureList": [
    {
      "index": "0x3",
      "signature":
↪"0x8549a8adddb9ff2d8d472b7a65ca414a7f10519f59e11244b9e92fed2e30f29cc272938a695ebb1ead579bdaed40
↪"
    },
    {
      "index": "0x1",
      "signature":
↪"0x0ff480d6c3fe5908c554653376492cba23cfc938bc0847298c9c8b4c140b417be1990d25c0239f9b18e24276e864
↪"
    },
    {
      "index": "0x2",
      "signature":
↪"0x50aa1513c390e5e61221418aa46f8dd45e3ce6e8c0a6fa5aa7565051756058959 0172b1e6cc64e9ba7ab45c964ff
↪"
    }
  ],
  "transactions": [
    {
      "blockHash":
↪"0xa12a4a8d26a15fcff6f67acf94a3fb2c08d4476806009be40fe4026fd622419d",
      "blockNumber": 123,
      "from": "0x1cd8126354aa951dc8ed083ed91fd31fad1f50d8",
      "gas": "0x5f5e100",
      "hash": "0xbe858390cf0bd94f9d17c6ab5e04a7d354f9482f2568f067bdc2529a0af324db",
      "input":
↪"0x25e85d160000000000000000000000000000000000000000000000000000000000000020000000000000000000000
↪",
      "nonce": "0x1479caf2ff5f1a6121beac93c85b59c36280ffe6a7f124cbc3d232021fa3307",
      "to": "0x0000000000000000000000000000000000001003",
      "transactionIndex": "0x0",
      "value": "0x0",
      "gasPrice": "0x1",
      "blockLimit": "0x26e",
      "chainId": "0x1",
      "groupId": "0x1",
```


(下页继续)

(续上页)


```
      "extraData": "0x",
      "signature": {
        "r": "0x36edc104cc5c2c5c992f9804bd3415378f93c18ebc9cdeb7794c08b1c95fd7ad",
        "s": "0xed555fe3a192acc5ea01951f7286f93cff3471e84731a2942968a2fd9a9a14bc",
        "v":
↪"0xd3a0fac51c66adf23bfccc6bb3aaa3d3623040912963ceb818ca2d8b1fb649c64bce64d99a8d9a39158b2c713101
↪",
        "signature":
↪"0x36edc104cc5c2c5c992f9804bd3415378f93c18ebc9cdeb7794c08b1c95fd7aded555fe3a192acc5ea01951f7286
↪"
      }
    }
  ]
}
```


3.18. 获取群组内同步状态信息

接口描述

获取群组内同步状态信息

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/syncStatus

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/syncStatus
```

响应参数

1）数据格式


```
{
  "isSyncing": "false",
  "protocolId": "65545",
  "genesisHash": "1caec77e0ff42af4f045af2ba5503c775c33430085b246374cdd23d1c9191173
↪",
  "nodeId":
↪"fe57d7b39ed104b4fb2770ae5aad7946bfd377d0eb91ab92a383447e834c3257dec56686551d08178f2d5f40d9fad6
↪",
  "blockNumber": "232",
  "latestHash": "7f0885a7188bd5c1b3f8c182e00be0e63a3b1653b12752cf8501ddac995e6efc",
  "knownHighestNumber": "232",
  "txPoolSize": "0",
```


(下页继续)

(续上页)


```
  "peers": [
    {
      "nodeId":
↪"06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b35067c1685d4343e6ad84b45c
↪",
      "genesisHash":
↪"1caec77e0ff42af4f045af2ba5503c775c33430085b246374cdd23d1c9191173",
      "blockNumber": "232",
      "latestHash":
↪"7f0885a7188bd5c1b3f8c182e00be0e63a3b1653b12752cf8501ddac995e6efc"
    },
    {
      "nodeId":
↪"65bc44d398d99d95a9d404aa16e4bfbc2f9ebb40f20439ddef8575a139dc3a80310cfc98a035bd59a67cc5f659f519e
↪",
      "genesisHash":
↪"1caec77e0ff42af4f045af2ba5503c775c33430085b246374cdd23d1c9191173",
      "blockNumber": "232",
      "latestHash":
↪"7f0885a7188bd5c1b3f8c182e00be0e63a3b1653b12752cf8501ddac995e6efc"
    },
    {
      "nodeId":
↪"95efafa5197796e7edf647191de83f4259d7cbb060f4bac5868be474037f49144d581c15d8aef9b07d78f18041a5f43
↪",
      "genesisHash":
↪"1caec77e0ff42af4f045af2ba5503c775c33430085b246374cdd23d1c9191173",
      "blockNumber": "232",
      "latestHash":
↪"7f0885a7188bd5c1b3f8c182e00be0e63a3b1653b12752cf8501ddac995e6efc"
    }
  ],
  "knownLatestHash":
↪"7f0885a7188bd5c1b3f8c182e00be0e63a3b1653b12752cf8501ddac995e6efc"
}
```


3.19. 刷新前置

接口描述

刷新前置的群组列表，功能与`groupList`类似

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/refresh

调用方法

HTTP GET

请求参数

1）参数表

无

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/refresh
```

响应参数

1）数据格式


```
{
    "blockNumber": 126,
    "genesisHash":
↪"0xed3350d191d23cbc609c98e920baa583403b9a02fa934df868e7f425cd72f5c3",
    "isSyncing": false,
    "latestHash":
↪"0x49ca6eb004f372c71ed900ec6992582cd107e4f3ea36aaa5a0a78829ebef1f14",
    "nodeId":
↪"d822165959a0ed217df6541f1a7dd38b79336ff571dd5f8f85ad76f3e7ec097e1eabd8b03e4a757fd5a9fb0eea905a
↪",
    "peers": [
        {
            "blockNumber": 126,
            "genesisHash":
↪"0xed3350d191d23cbc609c98e920baa583403b9a02fa934df868e7f425cd72f5c3",
            "latestHash":
↪"0x49ca6eb004f372c71ed900ec6992582cd107e4f3ea36aaa5a0a78829ebef1f14",
            "nodeId":
↪"552398be0eef124c000e632b0b76a48c52b6cfbd547d92c15527c2d1df15fab2bcded48353db22526c3540e4ab2027
↪"
        },
        {
            "blockNumber": 126,
            "genesisHash":
↪"0xed3350d191d23cbc609c98e920baa583403b9a02fa934df868e7f425cd72f5c3",
            "latestHash":
↪"0x49ca6eb004f372c71ed900ec6992582cd107e4f3ea36aaa5a0a78829ebef1f14",
            "nodeId":
↪"adfa2f9116d7ff68e0deb75307fa1595d636bf097ad1de4fb55cff00e4fef40b453abb30388aa2112bf5cd4c987afe
↪"
        },
        {
            "blockNumber": 126,
            "genesisHash":
↪"0xed3350d191d23cbc609c98e920baa583403b9a02fa934df868e7f425cd72f5c3",
            "latestHash":
↪"0x49ca6eb004f372c71ed900ec6992582cd107e4f3ea36aaa5a0a78829ebef1f14",
            "nodeId":
↪"dde0bbf5eb3a731e6da861586e98e088e16e6fdd9afae2f2c213cead20a4f5eaa3910042b70d62266d2350d98a43c1
↪"
        }
    ],
    "protocolId": 265,
    "txPoolSize": "0"
}
```


3.20. 获取区块的时间戳与交易量

接口描述

根据块高获取区块的时间戳与区块中的交易数

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/blockStat/{blockNumber}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/blockStat/5
```

响应参数

1）数据格式

a、成功:

```
{
  "blockNumber": 5,
  "timestamp": 1617715398371,
  "txCount": 1
}
```

3.21. 批量获取区块内交易回执接口

接口描述

根据块高批量获取区块交易回执

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/transactionReceipt/{transHash}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/transReceipt/batchByNumber/
↪{blockNumber}?start={start}&count={count}
```

响应参数

2）数据格式


```
{
  "blockNumber": "string",
  "contractAddress": "string",
  "from": "string",
  "gasUsed": "string",
  "hash": "string",
  "input": "string",
  "logEntries": [
    {
      "address": "string",
      "blockNumber": "string",
      "data": "string",
      "topic": [
        "string"
      ]
    }
  ],
  "message": "string",
  "output": "string",
  "receiptProof": [
    {
      "left": [
        "string"
      ],
      "right": [
        "string"
      ]
    }
  ],
  "status": 0,
  "statusOK": true,
  "to": "string",
  "transactionHash": "string",
  "transactionProof": [
    {
      "left": [
        "string"
      ],
      "right": [
        "string"
      ]
    }
  ],
  "version": "string"
}
```


12.3.4 4.链上信息接口

4.1. 查询链上信息

接口描述

获取链上信息

接口URL

http://localhost:5002/WeBASE-Front/chain

调用方法

HTTP GET

请求参数

1）参数表

2）入参示例

```
localhost:5002/WeBASE-Front/chain?beginDate=2019-03-13T00:00:00&endDate=2019-03-
↪13T14:34:22&contrastBeginDate=2019-03-13T00:00:00&contrastEndDate=2019-03-
↪13T14:34:22&gap=60&groupId=group
```

响应参数

1）参数表

2）出参示例

```
[
        {
                "metricType": "blockHeight",
                "data": {
                        "lineDataList": {
                                "timestampList": [
                                        1552406401042,
                                        1552406701001
                                ],
                                "valueList": [
                                        747309,
                                        747309
                                ]
                        },
                        "contrastDataList": {
                                "timestampList": [
                                        1552320005000,
                                        1552320301001
                                ],
                                "valueList": [
                                        null,
                                        747309
                                ]
                        }
                }
        },
        {
                "metricType": "pbftView",
                "data": {
                        "lineDataList": {
                                "timestampList": null,
                                "valueList": [
                                        118457,
                                        157604
```

(下页继续)

(续上页)

```
                                ]
                        },
                        "contrastDataList": {
                                "timestampList": null,
                                "valueList": [
                                        null,
                                        33298
                                ]
                        }
                }
        }
]
```

4.2. 分页查询链上信息

接口描述

获取链上信息

接口URL

http://localhost:5002/WeBASE-Front/chain/pagingQuery

调用方法

HTTP GET

请求参数

1）参数表

2）入参示例

```
localhost:5002/WeBASE-Front/chain/pagingQuery?groupId=1&pageNumber=1&pageSize=10
```

响应参数

```
{
  "code": 0,
  "message": "success",
  "data": [],
  "totalCount": 0
}
```

12.3.5 5. 交易接口

5.1. 交易处理接口（结合WeBASE-Sign）

接口描述

通过此接口对合约进行调用，前置根据调用的合约方法是否是“constant”方法区分返回信息，“constant”方法为查询，返回要查询的信息。非“constant”方法为发送数据上链，返回块hash、块高、交易hash等信息。

当合约方法为非“constant”方法，要发送数据上链时，此接口需结合WeBASE-Sign使用。通过调用WeBASE-Sign服务的签名接口让相关用户对数据进行签名，拿回签名数据再发送上链。需要调用此接口时，工程配置文件application.yml中的配置”keyServer”需配置WeBASE-Sign服务的ip和端口，并保证WeBASE-Sign服务正常和存在相关用户。

3.0.2及以后版本：

方法参数（funcParam）为String数组，每个参数都使用String字符串表示，多个参数以逗号分隔（参数为数组时同理），示例：

```
function set(string s) -> ["aa,bb\"cc"]         // 双引号要转义
function set(uint n,bool b) -> ["1","true"]
function set(bytes b,address[] a) -> ["0x1a","[\
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE\",\
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9\"]"]
```

3.0.2以前的版本：

方法入参（funcParam）为JSON数组，多个参数以逗号分隔（参数为数组时同理），示例：

```
function set(string s) -> ["aa,bb\"cc"]         // 双引号要转义
function set(uint n,bool b) -> [1,true]
function set(bytes b,address[] a) -> ["0x1a",[
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE",
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9"]]
```

查看*WeBASE-Front*通过本地私钥（测试用户）交易处理接口（非*WeBASE-Sign*签名交易），可查看其他接口-交易处理接口（本地签名）

接口URL

http://localhost:5002/WeBASE-Front/trans/handleWithSign

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "cnsName": "string",
  "contractAbi": [
    {}
  ],
  "contractAddress": "string",
  "contractName": "string",
  "contractPath": "string",
  "funcName": "string",
```

(下页继续)

(续上页)

```
  "funcParam": [
    {}
  ],
  "groupId": "string",
  "signUserId": "string",
  "useCns": true,
  "version": "string"
}
```

示例:

```
curl -X POST "http://localhost:5002/WeBASE-Front/trans/handleWithSign" -H "accept:␣
↪*/*" -H "Content-Type: application/json" -d "{\"groupId\":\"1\",\"signUserId\":\
↪"f4975519b0274e6ca8283650a7e1bc07\",\"contractName\":\"HelloWorld\",\
↪"contractPath\":\"/\",\"version\":\"\",\"funcName\":\"set\",\"funcParam\":[\"333\
↪"],\"contractAddress\":\"0xe10441d9179cf0424aae808b51bc85dcbbfe1447\",\
↪"contractAbi\":[{\"inputs\":[{\"internalType\":\"string\",\"name\":\"n\",\"type\
↪":\"string\"}],\"name\":\"set\",\"outputs\":[],\"stateMutability\":\"nonpayable\
↪",\"type\":\"function\",\"funcId\":2}],\"useAes\":false,\"useCns\":false,\
↪"cnsName\":\"\"}"
```

响应参数

a、正确查询交易返回值信息

```
{"Hi,Welcome!"}
```

b、正确发送数据上链返回值信息（交易收据）

```
{
  "transactionHash":
↪"0x69ced0162a0c3892e4eaa3091b831ac3aaeb772c062746b20891ceaf8a4fb429",
  "transactionIndex": "0x0",
  "root": "0x8cbc3f2c0e35a71738909e3b388efa6697084b05badd3a3bd3c64f0575c78c15",
  "blockNumber": "2",
  "blockHash": "0xf58f4f43b3761f4863ad366c4a7e2a812ed68df9f7bcad6b502fd544665e7625
↪",
  "from": "0x9d75e0ee66cfef16897b601624b60413d988ae7d",
  "to": "0x0000000000000000000000000000000000000000",
  "gasUsed": "316449",
  "contractAddress": "0xa8af0ee580d8af674a60341030ddbc45431bc235",
  "logs": [],
  "logsBloom":
↪"0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
↪",
  "status": "0x0",
  "statusMsg": "None",
  "input":
↪"0x608060405234801561001057600080fd5b506103e3806100206000396000f30060806040526004361061005757600
↪",
  "output": "0x",
  "txProof": null,
  "receiptProof": null,
  "message": null,
  "statusOK": true
}
```

5.2. 交易处理接口（本地签名）

接口描述

此接口为WeBASE-Front使用**本地私钥（页面中的测试用户）进行签名**

通过合约信息进行调用，前置根据调用的合约方法是否是“constant”方法区分返回信息，“constant”方法为查询，返回要查询的信息。非“constant”方法为发送数据上链，返回块hash、块高、交易hash等信息。

3.0.2及以后版本：

方法参数（funcParam）为String数组，每个参数都使用String字符串表示，多个参数以逗号分隔（参数为数组时同理），示例：

```
function set(string s) -> ["aa,bb\"cc"]         // 双引号要转义
function set(uint n,bool b) -> ["1","true"]
function set(bytes b,address[] a) -> ["0x1a","[\
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE\",\
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9\"]"]
```

3.0.2以前的版本：

方法入参（funcParam）为JSON数组，多个参数以逗号分隔（参数为数组时同理），示例：

```
function set(string s) -> ["aa,bb\"cc"]         // 双引号要转义
function set(uint n,bool b) -> [1,true]
function set(bytes b,address[] a) -> ["0x1a",[
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE",
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9"]]
```

查看*WeBASE-Front*通过*WeBASE-Sign*交易处理的接口（非本地私钥签名交易），可查看合约接口-交易处理接口（结合WeBASE-Sign）

接口URL

http://localhost:5002/WeBASE-Front/trans/handle

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "cnsName": "string",
  "contractAbi": [
    {}
  ],
  "contractAddress": "string",
  "contractName": "string",
  "contractPath": "string",
  "funcName": "string",
  "funcParam": [
    {}
  ],
  "groupId": "string",
  "useAes": true,
```

(下页继续)

(续上页)

```
  "useCns": true,
  "user": "string",
  "version": "string"
}
```

示例：

```
curl -l -H "Content-type: application/json" -X POST -d '{\"groupId\":\"1\",\"user\
↪":\"0x7bfa3539c8761978f0a2a1f7f240bde97819fb03\",\"contractName\":\"HelloWorld\",
↪\"contractPath\":\"/\",\"version\":\"\",\"funcName\":\"set\",\"funcParam\":[\
↪"333\"],\"contractAddress\":\"0xe10441d9179cf0424aae808b51bc85dcbbfe1447\",\
↪"contractAbi\":[{\"inputs\":[{\"internalType\":\"string\",\"name\":\"n\",\"type\
↪":\"string\"}],\"name\":\"set\",\"outputs\":[],\"stateMutability\":\"nonpayable\
↪",\"type\":\"function\",\"funcId\":2}],\"useAes\":false,\"useCns\":false,\
↪"cnsName\":\"\"}' http://10.0.0.1:5002/WeBASE-Front/trans/handle
```

传入合约abi:

```
{
        "groupId": "group",
        "user": "0x7bfa3539c8761978f0a2a1f7f240bde97819fb03",
        "contractName": "HelloWorld",
        "contractPath": "/",
        "version": "",
        "funcName": "set",
        "funcParam": ["333"],
        "contractAddress": "0xe10441d9179cf0424aae808b51bc85dcbbfe1447",
        "contractAbi": [{
                "inputs": [{
                        "internalType": "string",
                        "name": "n",
                        "type": "string"
                }],
                "name": "set",
                "outputs": [],
                "stateMutability": "nonpayable",
                "type": "function",
                "funcId": 2
        }],
        "useAes": false,
        "useCns": false,
        "cnsName": ""
}
```

响应参数

a、正确查询交易返回值信息

```
{"Hi,Welcome!"}
```

b、正确发送数据上链返回值信息（交易收据）

```
{
  "transactionHash":
↪"0x69ced0162a0c3892e4eaa3091b831ac3aaeb772c062746b20891ceaf8a4fb429",
  "transactionIndex": "0x0",
  "root": "0x8cbc3f2c0e35a71738909e3b388efa6697084b05badd3a3bd3c64f0575c78c15",
  "blockNumber": "2",
  "blockHash": "0xf58f4f43b3761f4863ad366c4a7e2a812ed68df9f7bcad6b502fd544665e7625
↪",
```

(下页继续)

(续上页)


```
  "from": "0x9d75e0ee66cfef16897b601624b60413d988ae7d",
  "to": "0x0000000000000000000000000000000000000000",
  "gasUsed": "316449",
  "contractAddress": "0xa8af0ee580d8af674a60341030ddbc45431bc235",
  "logs": [],
  "logsBloom":
↪"0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
↪",
  "status": "0x0",
  "statusMsg": "None",
  "input":
↪"0x6080604052348015610010576000080fd5b506103e3806100206000396000f3006080604052600436106100575760
↪",
  "output": "0x",
  "txProof": null,
  "receiptProof": null,
  "message": null,
  "statusOK": true
}
```


5.3. 已签名交易发送

接口描述

发送已签名的交易上链，返回交易收据；可结合/trans/convertRawTxStr/local或/trans/convertRawTxStr/withSign接口组装已签名的交易

接口URL

http://localhost:5002/WeBASE-Front/trans/signed-transaction

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId": "string",
  "signedStr": "string",
  "sync": true
}
```

响应参数

1）数据格式

```
{
        "transactionHash":
↪"0xd1234cdb673447b654874d05d4ed98be89f3cb7c5a6c12bc51e1274a03af029c",
        "transactionIndex": "0x0",
        "root": "0xeecf2b6bbb028e38e704ca88fe1d9f5e3d43197b32c3336d3273d86fb3a135f0
↪",
        "blockNumber": "0xa9",
        "blockHash":
↪"0x906bdf0747ba68c03e6cf73463db62fdbd8ab8debc149f890475775d7c6a8af7",
        "from": "0x7bfa3539c8761978f0a2a1f7f240bde97819fb03",
        "to": "0xe10441d9179cf0424aae808b51bc85dcbbfe1447",
        "gasUsed": "0x710b",
        "contractAddress": "0x0000000000000000000000000000000000000000",
        "logs": [],
        "logsBloom":
↪"0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
↪",
        "status": "0x0",
        "statusMsg": "None",
        "input":
↪"0x3590b49f000000000000000000000000000000000000000000000000000000000000002000000000000000000000000
↪",
        "output": "0x",
        "txProof": null,
        "receiptProof": null,
        "message": "Success",
        "statusOK": true
}
```

5.4. 已编码查询交易发送

接口描述

发送已编码的查询交易，返回合约的返回值；其中encodeStr字段可通过/trans/encodeFunction接口获取合约函数的编码值

接口URL

http://localhost:5002/WeBASE-Front/trans/query-transaction

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractAbi": "string",
  "contractAddress": "string",
  "encodeStr": "string",
  "funcName": "string",
```

(下页继续)

(续上页)

```
  "groupId": "string",
  "userAddress": "string"
}
```

响应参数

Object返回类型

```
{"Hi,Welcome!"}
```

5.5. Hash计算

接口描述

计算HASH和签名值

接口URL

http://localhost:5002/WeBASE-Front/trans/signMessageHash

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId": "string",
  "hash": "string",
  "signUserId": "string",
  "user": "string"
}
```

响应参数

1）数据格式

```
{
  "v": 0,
  "r": "0x2a76a45bcf1113615f796cc01b23c57f81f20ce79500080bb34c7994ed04de06",
  "s": "0x4f111eb37720e2618d8906368c825fd3cbe89b2781cb678efafb42399594a580",
  "p":
↪"0x4405f9d5d6ccff709b6543bc8ac24cbb649d3267a66db19ab8f85f3b884a8505f086c581490e7e50558879abde9c4
↪"
}
```

5.6. 获取签名后的交易体编码值（结合WeBASE-Sign）

接口描述

构造交易体RawTransaction并将交易体编码，通过传入的`signUserId`签名服务的用户ID，使用对应的私钥对交易提进行签名后，返回已签名的交易体编码值（十六进制字符串）

签名后的交易的编码值可以直接通过`/trans/signed-transaction`接口提交到链上

接口URL

http://localhost:5002/WeBASE-Front/trans/convertRawTxStr/withSign

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "cnsName": "string",
  "contractAbi": [
    {}
  ],
  "contractAddress": "string",
  "contractName": "string",
  "contractPath": "string",
  "funcName": "string",
  "funcParam": [

  ],
  "groupId": "string",
  "signUserId": "string",
  "useCns": true,
  "version": "string"
}
```

响应参数

```
0xf9012da002c1442b6ce974291ec317db9859e8310de46f6636d105f19fee67f10ac60b2f018405f5e10082029e94e10
```

5.7. 获取签名后的交易体编码值（本地签名）

接口描述

构造交易体RawTransaction并将交易体编码，并通过传入的`user`地址的私钥对交易提进行签名后，返回已签名的交易体编码值（十六进制字符串）

签名后的交易的编码值可以直接通过`/trans/signed-transaction`接口提交到链上

接口URL

http://localhost:5002/WeBASE-Front/trans/convertRawTxStr/local

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "cnsName": "string",
  "contractAbi": [
    {}
  ],
  "contractAddress": "string",
  "contractName": "string",
  "contractPath": "string",
  "funcName": "string",
  "funcParam": [
    {}
  ],
  "groupId": "string",
  "useAes": true,
  "useCns": true,
  "user": "string",
  "version": "string"
}
```

示例：

```
{
  "cnsName": "string",
  "contractAbi": [
    {}
  ],
  "contractAddress": "string",
  "contractName": "string",
  "contractPath": "string",
  "funcName": "string",
  "funcParam": [

  ],
  "groupId": "string",
  "useAes": true,
  "useCns": true,
  "user": "string",
  "version": "string"
}
```

响应参数

1）数据格式

```
0xf9012da001071041dddc1b3c553b48c0fbefecc07f3812f5ce4004d47708f1c3342844db018405f5e10082029d94e10
```

5.8. 获取合约函数的编码值

接口描述

构造合约函数的编码值，适用于查询交易，合约函数的编码值可以直接通过/trans/query-transaction接口提交到链上

接口URL

http://localhost:5002/WeBASE-Front/trans/encodeFunction

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractAbi": [
    {}
  ],
  "funcName": "string",
  "funcParam": [

  ],
  "groupId": "string"
}
```

示例：

响应参数

1）数据格式

```
0x299f7f9d
```

12.3.6 6. 系统管理接口

使用FISCO BCOS v2.5.0 与 WeBASE-Front v1.4.1 (及)以上版本将使用预编译合约中的ChainGovernance接口(从本章节接口*6.13*开始)，详情可参考FISCO BCOS基于角色的权限控制

6.1. 查询系统配置接口

接口描述

根据群组id获取系统配置SystemConfig的list列表，目前只支持tx_count_limit, tx_gas_limit两个参数。

接口URL

http://localhost:5002/WeBASE-Front/sys/config/list?groupId={groupId}&pageSize={pageSize}&pageNumber={pageNumbe

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/sys/config/list?groupId=1&pageSize=5&
↪pageNumber=1
```

响应参数

1）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "id": 6,
            "groupId": 1,
            "fromAddress": "0x",
            "configKey": "tx_gas_limit",
            "configValue": "300000000"
        },
        {
            "id": 5,
            "groupId": 1,
            "fromAddress": "0xd0b56b4ce0e8ce5e064f896d9ad1c16b2603f285",
            "configKey": "tx_count_limit",
            "configValue": "10002"
        }
    ],
    "totalCount": 2
}
```

6.2. 设置系统配置接口

接口描述

系统配置管理员设置系统配置，目前只支持tx_count_limit, tx_gas_limit两个参数。

接口URL

http://localhost:5002/WeBASE-Front/sys/config

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "configKey": "string",
  "configValue": "string",
  "fromAddress": "string",
  "groupId": "string",
  "signUserId": "string"
}
```

示例：

```
curl -l -H "Content-type: application/json" -X POST -d '{"groupId": 1, "fromAddress
↪": "0xd5bba8fe456fce310f529edecef902e4b63129b1", "configKey": "tx_count_limit",
↪"configValue": "1001"}' http://10.0.0.1:5002/WeBASE-Front/sys/config
```

响应参数

a、成功：

```
{
    "code": 0,
    "message": "success"
}
```

b、失败：

```
{
    "code": -50000,
    "message": "permission denied"
}
```

6.3. 查询节点接口（节点管理）

接口描述

获取节点的list列表，列表包含节点id，节点共识状态。

注：接口返回所有的共识/观察节点（无论运行或停止），以及正在运行的游离节点

接口URL

http://localhost:5002/WeBASE-Front/precompiled/consensus/list?groupId={groupId}&pageSize={pageSize}&pageNumber=

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/precompiled/consensus/list?groupId=group&
↪pageSize=5&pageNumber=1
```

响应参数

1）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "nodeId":
↪"13e0f2b94cbce924cc3737385a38587939e809fb786c4fc34a6ba3ea97693bccfa173b352ac41f1dbb97e9e4910ccb
↪",
            "nodeType": "sealer"
        },
        {
            "nodeId":
↪"bce4b2269c25c2cdba30155396bfe90af08c3c08cff205213477683117e4243ebe26588479519e636a5d5d93545cab
↪",
            "nodeType": "sealer"
        },
        {
            "nodeId":
↪"e815cc5637cb8c3274c83215c680822e4a0110d0a8315fcf03e43e8e3944edd758c8b173c4e0076f599aa1f853fa20
↪",
            "nodeType": "sealer"
        }
    ],
    "totalCount": 3
}
```

6.4. 设置节点共识状态接口（节点管理）

接口描述

节点管理相关接口，可用于节点三种共识状态的切换。分别是共识节点sealer, 观察节点observer, 游离节点remove

接口URL

http://localhost:5002/WeBASE-Front/precompiled/consensus

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fromAddress": "string",
  "groupId": "string",
  "nodeId": "string",
  "nodeType": "string",
  "signUserId": "string",
  "weight": 0
}
```

示例：

```
curl -l -H "Content-type: application/json" -X POST -d '{"groupId": 1, "fromAddress
↪": "0xd5bba8fe456fce310f529edecef902e4b63129b1", "configKey": "tx_count_limit",
↪"configValue": "1001"}' http://10.0.0.1:5002/WeBASE-Front/sys/config
```

响应参数

a、成功：

```
{
    "code": 0,
    "message": "success"
}
```

b、失败：

```
{
    "code": -50000,
    "message": "permission denied"
}
```

12.3.7 7. Abi管理接口

7.1. 获取Abi信息

接口描述

根据abiId获取abi信息

接口URL

http://localhost:5002/WeBASE-Front/abi/{abiId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/abi/{abiId}
```

响应参数

1）数据格式

a、成功：

```
{
  "code": 0,
  "message": "success",
  "data": {
    "abiId": 1,
    "groupId": 1,
    "contractName": "TTT",
    "contractAddress": "0x3214227e87bccca63893317febadd0b51ade735e",
    "contractAbi": "[{\"constant\":false,\"inputs\":[{\"name\":\"n\",\"type\":\
↪"string\"}],\"name\":\"set\",\"outputs\":[],\"payable\":false,\"stateMutability\
↪":\"nonpayable\",\"type\":\"function\"},{\"constant\":true,\"inputs\":[],\"name\
↪":\"get\",\"outputs\":[{\"name\":\"\",\"type\":\"string\"}],\"payable\":false,\
↪"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\":false,\"inputs\
↪":[{\"name\":\"n\",\"type\":\"string\"}],\"name\":\"setSender\",\"outputs\":[],\
↪"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\
↪"anonymous\":false,\"inputs\":[{\"indexed\":false,\"name\":\"name\",\"type\":\
↪"string\"}],\"name\":\"SetName\",\"type\":\"event\"},{\"anonymous\":false,\
↪"inputs\":[{\"indexed\":false,\"name\":\"\",\"type\":\"uint256[2]\"}],\"name\":\
↪"EventList\",\"type\":\"event\"},{\"anonymous\":false,\"inputs\":[{\"indexed\
↪":false,\"name\":\"sender\",\"type\":\"address\"}],\"name\":\"SetSender\",\"type\
↪":\"event\"}]",
    "contractBin": "60806040526004361061005757600035...",
    "createTime": "2020-05-18 10:59:02",
    "modifyTime": "2020-05-18 10:59:02"
  }
}
```

7.2. 获取Abi信息分页列表

接口描述

获取abi信息的列表

接口URL

http://localhost:5002/WeBASE-Front/abi/list/{groupId}/{pageNumber}/{pageSize}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/abi/list/{groupId}/{pageNumber}/{pageSize}
```

响应参数

1）数据格式

a、成功：


```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "abiId": 1,
      "groupId": 1,
      "contractName": "TTT",
      "contractAddress": "0x3214227e87bccca63893317febadd0b51ade735e",
      "contractAbi": "[{\"constant\":false,\"inputs\":[{\"name\":\"n\",\"type\":\
↪"string\"}],\"name\":\"set\",\"outputs\":[],\"payable\":false,\"stateMutability\
↪":\"nonpayable\",\"type\":\"function\"},{\"constant\":true,\"inputs\":[],\"name\
↪":\"get\",\"outputs\":[{\"name\":\"\",\"type\":\"string\"}],\"payable\":false,\
↪"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\":false,\"inputs\
↪":[{\"name\":\"n\",\"type\":\"string\"}],\"name\":\"setSender\",\"outputs\":[],\
↪"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\
↪"anonymous\":false,\"inputs\":[{\"indexed\":false,\"name\":\"name\",\"type\":\
↪"string\"}],\"name\":\"SetName\",\"type\":\"event\"},{\"anonymous\":false,\
↪"inputs\":[{\"indexed\":false,\"name\":\"\",\"type\":\"uint256[2]\"}],\"name\":\
↪"EventList\",\"type\":\"event\"},{\"anonymous\":false,\"inputs\":[{\"indexed\
↪":false,\"name\":\"sender\",\"type\":\"address\"}],\"name\":\"SetSender\",\"type\
↪":\"event\"}]",
      "contractBin":
↪"60806040526004361061005757600357c0100000000000000000000000000000000000000000000000000000000090
↪..29",
      "createTime": "2020-05-18 10:59:02",
      "modifyTime": "2020-05-18 10:59:02"
    }
  ],
  "totalCount": 1
}
```


7.3. 导入已部署合约的abi

接口描述

将其他平台已部署的合约导入到本平台进行管理

接口URL

http://localhost:5002/WeBASE-Front/abi

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "abiId": 0,
  "contractAbi": [
    {}
  ],
  "contractAddress": "string",
  "contractName": "string",
  "groupId": "string"
}
```

响应参数

1）数据格式

成功：

```
{
    "code": 0,
    "message": "success"
}
```

7.4. 修改已导入的合约abi

接口描述

更新已导入的合约abi内容

接口URL

http://localhost:5002/WeBASE-Front/abi

调用方法

HTTP PUT

请求参数

1）参数表

2）数据格式

```
{
  "abiId": 0,
  "contractAbi": [
    {}
  ],
  "contractAddress": "string",
  "contractName": "string",
  "groupId": "string"
}
```

响应参数

1）数据格式

成功：

```
{
    "code": 0,
    "message": "success"
}
```

7.5. 修改合约abi

接口描述

删除合约abi

接口URL

http://localhost:5002/WeBASE-Front/abi/{abiId}

调用方法

HTTP DELETE

请求参数

1）参数表

2）数据格式

```
{
    "abiId": 1
}
```

响应参数

1）数据格式

成功：

```
{
    "code": 0,
    "message": "success"
}
```

12.3.8 8. 其他接口

8.1. 查询是否使用国密

接口描述

获取WeBASE-Front的`encryptType`，即是否使用国密版；

接口URL

http://localhost:5002/WeBASE-Front/{groupId}/web3/encrypt

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/group/web3/encrypt
```

响应参数

1）数据格式

a、成功：

```
{
    1 // 1: 国密版，0: 非国密
}
```

8.2. 查询WeBASE-Front版本

接口描述

获取WeBASE-Front的版本号

接口URL

http://localhost:5002/WeBASE-Front/version

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/version
```

响应参数

1）数据格式

a、成功：

```
v1.4.0
```

8.3. 查询前置连接的WeBASE-Sign版本

接口描述

获取WeBASE-Front的所连接的WeBASE-Sign的版本号

接口URL

http://localhost:5002/WeBASE-Front/version/sign

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/version/sign
```

响应参数

1）数据格式

a、成功：

```
v1.4.0
```

8.4. 查询前置包含的solidity v0.6.10文件

接口描述

获取WeBASE-Front的本地conf/solcjs中包含的solidity 0.6.10的js文件列表

如需要使用*solidity 0.6.10*，则需要手动下载*CDN*中*solidity*的*v0.6.10.js*（国密*v0.6.10-gm.js*），并在*WeBASE-Front*的*conf*文件夹中创建*solcjs*文件夹，并将*js*文件复制到该文件夹。

注：使用*webase-front.zip*安装包直接部署则不需要手动放置*js*文件

接口URL

http://localhost:5002/WeBASE-Front/solc/list

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/solc/list
```

响应参数

1）数据格式

a、成功：

```
{
    "code":0,
    "message":"success",
    "data":["v0.6.10.js","v0.6.10-gm.js"]
}
```

12.3.9 9. 合约仓库

9.1. 获取合约仓库列表

接口描述

返回合约仓库信息列表

接口URL

http://localhost:5002/WeBASE-Front/contractStore/getContractStoreList

调用方法

HTTP GET

请求参数

1）参数表

无

2）数据格式

```
http://localhost:5002/WeBASE-Front/contractStore/getContractStoreList
```

响应参数

1）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "storeId": 1,
      "storeName": "工具箱",
      "storeName_en": "Toolbox",
      "storeType": "1",
      "storeIcon": "toolboxId",
      "storeDesc": "工具箱中有常用的工具合约",
      "storeDetail": "工具箱中有常用的工具合约",
      "storeDesc_en": "Toolbox Contract suite",
      "storeDetail_en": "Toolbox Contract suite",
      "createTime": "2021-01-20 18:02:10",
      "modifyTime": "2021-01-20 18:02:10"
    },
    {
      "storeId": 2,
      "storeName": "存证应用",
      "storeName_en": "Evidence",
      "storeType": "2",
      "storeIcon": "evidenceId",
      "storeDesc": "一套区块链存证合约",
      "storeDetail": "一套区块链存证合约",
      "storeDesc_en": "Evidence Contract suite",
      "storeDetail_en": "Evidence Contract suite",
      "createTime": "2021-01-20 18:02:10",
      "modifyTime": "2021-01-20 18:02:10"
    },
```

(下页继续)

(续上页)

```
    {
      "storeId": 3,
      "storeName": "积分应用",
      "storeName_en": "Points",
      "storeType": "3",
      "storeIcon": "pointsId",
      "storeDesc": "一套积分合约",
      "storeDetail": "一套积分合约",
      "storeDesc_en": "Points Contract suite",
      "storeDetail_en": "Points Contract suite",
      "createTime": "2021-01-20 18:02:10",
      "modifyTime": "2021-01-20 18:02:10"
    }
  ]
}
```

9.2. 根据仓库编号获取仓库信息

接口描述

返回合约仓库信息

接口URL

http://localhost:5002/WeBASE-Front/contractStore/getContractFolderById/{storeId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contractStore/getContractStoreById/1
```

响应参数

1）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": {
    "storeId": 1,
    "storeName": "工具箱",
    "storeName_en": "Toolbox",
    "storeType": "1",
    "storeIcon": "toolboxId",
    "storeDesc": "工具箱中有常用的工具合约",
    "storeDetail": "工具箱中有常用的工具合约",
```

(下页继续)

(续上页)

```
    "storeDesc_en": "Toolbox Contract suite",
    "storeDetail_en": "Toolbox Contract suite",
    "createTime": "2021-01-20 18:02:10",
    "modifyTime": "2021-01-20 18:02:10"
  }
}
```

9.3. 根据仓库编号获取合约文件夹信息

接口描述

返回合约文件夹信息

接口URL

http://localhost:5002/WeBASE-Front/contractStore/getContractFolderById/{storeId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contractStore/getFolderItemListByStoreId/2
```

响应参数

1）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "contractFolderId": 2,
      "storeId": 2,
      "contractFolderName": "Evidence",
      "contractFolderDesc": "Evidence",
      "contractFolderDetail": "Evidence",
      "contractFolderDesc_en": "Evidence",
      "contractFolderDetail_en": "Evidence",
      "createTime": "2021-01-20 18:02:10",
      "modifyTime": "2021-01-20 18:02:10"
    }
  ]
}
```

9.4. 根据合约文件夹编号获取合约文件夹信息

接口描述

返回合约文件夹信息

接口URL

http://localhost:5002/WeBASE-Front/contractStore/getContractFolderById/{contractFolderId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contractStore/getContractFolderById/2
```

响应参数

1）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": {
    "contractFolderId": 2,
    "storeId": 2,
    "contractFolderName": "Evidence",
    "contractFolderDesc": "Evidence",
    "contractFolderDetail": "Evidence",
    "contractFolderDesc_en": "Evidence",
    "contractFolderDetail_en": "Evidence",
    "createTime": "2021-01-20 18:02:10",
    "modifyTime": "2021-01-20 18:02:10"
  }
}
```

9.5. 根据文件夹编号获取合约列表

接口描述

返回合约信息列表

接口URL

http://localhost:5002/WeBASE-Front/contractStore/getContractItemByFolderId/{folderId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contractStore/getContractItemByFolderId/2
```

响应参数

1）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "contractId": 4,
      "contractFolderId": 2,
      "contractName": "Evidence",
      "contractDesc": "Evidence",
      "contractSrc":
↪"cHJhZ21hIHNvbGlkaXR5IF4wLjQuNDsKY29udHJhY3QgRXZpZGVuY2VTaWduZXJzRGF0YUFCSXsgZnVuY3Rpb24gdmVyaW
↪",
      "contractDesc_en": "Evidence",
      "createTime": "2021-01-20 18:02:10",
      "modifyTime": "2021-01-20 18:02:10"
    },
    {
      "contractId": 5,
      "contractFolderId": 2,
      "contractName": "EvidenceSignersData",
      "contractDesc": "EvidenceSignersData",
      "contractSrc":
↪"cHJhZ21hIHNvbGlkaXR5IF4wLjQuNDsKaW1wb3J0ICJFdmlkZW5jZS5zb2wiOwoKY29udHJhY3QgRXZpZGVuY2VTaWduZX
↪",
      "contractDesc_en": "EvidenceSignersData",
      "createTime": "2021-01-20 18:02:10",
      "modifyTime": "2021-01-20 18:02:10"
    }
  ]
}
```

9.6. 根据合约编号获取合约信息

接口描述

返回合约信息

接口URL

http://localhost:5002/WeBASE-Front/contractStore/getContractItemById/{contractId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/contractStore/getContractItemById/2
```

响应参数

1）数据格式


```
{
  "code": 0,
  "message": "success",
  "data": {
    "contractId": 2,
    "contractFolderId": 1,
    "contractName": "LibSafeMathForUint256Utils",
    "contractDesc": "LibSafeMathForUint256Utils",
    "contractSrc":
↪"LyoKICogQ29weXJpZ2h0IDIwMTQtMjAxOSB0aGUgb3JpZ2luYWwgYXV0aG9yIG9yIGF1dGhvcnMuCiAqCiAqIExpY2Vuc2V
↪",
    "contractDesc_en": "LibSafeMathForUint256Utils",
    "createTime": "2021-01-20 18:02:10",
    "modifyTime": "2021-01-20 18:02:10"
  }
}
```


12.3.10 10. 证书管理

10.1. 查询节点证书接口

接口描述

获取Front对应节点的Fisco证书和sdk证书（包含链证书、机构证书和节点证书）的内容；

需要在项目配置文件中`constant-nodePath`配置Front连接节点的绝对路径；

注：

> 接口只返回了证书的文本(Base64编码)，未包含开头与结尾以及换行的格式文本； 如需将文本保存为一个证书文件，需要加上开头“——BEGIN CERTIFICATE——\n”和结尾“\n——END CERTIFICATE——\n”；

接口URL

http://localhost:5002/WeBASE-Front/cert

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/cert
```

响应参数

1）数据格式

a、成功:


```
{
    "node":
↪"MIICOTCCASGgAwIBAgIJAKHsAYI3TsAOMA0GCSqGSIb3DQEBCwUAMDgxEDAOBgNV\nBAMMB2FnZW5jeUExEzARBgNVBAoM
↪LB7tELOiyCiIEhLScprb1LjvDDt2RDGjGjAYMAkGA1UdEwQCMAAwCwYD\nVR0PBAQDAgXgMA0GCSqGSIb3DQEBCwUAA4IBA
↪cnNhq5HVObbWxzfu7gn3+IN\nyQEeqdbGdzlu1EDcaMgAz6p2W3+FG/tmx/
↪yrNza29cYekWRL44OT5LOUPEKrJ4bJ\neOBRY4QlwZPFmM0QgP7DoKxHXldRopkmvqT4pbW51hWvPgj7KrdqwbVWzuWQuI3
↪",
    "chain":
↪"MIIDPTCCAiWgAwIBAgIJAMfvnu4d5fHdMA0GCSqGSIb3DQEBCwUAMDUxDjAMBgNV\nBAMMBWNoYWluMRMwEQYDVQQKDApma
↪S60cxvFS4tBLyfT0QKPLW1g3ZgMND\n03hrWp1FAnvE9htsDEgqvNLD5hKWaYcUhjQMq0WttiP/
↪vPxkwwJkZhzWhXpdSxMR\nqKVX4BppnkT0ICm84jYSyJdNFjKvfWlBIptIfFuTUDMT+XqF/
↪Ct756JksiUwKZRW\neRAVcYzFM4u4ZuKeaept/8Bv8Z/
↪RlJzGI57qj5BELeA0meUagq2WoCgJrPyvbO0b\nLwogFWS4kEjv20IIdj3fTqeJlooEXtPnuegunSMQB6aIh2im74FfJ3sHu
↪VYI16Mx+8OoGBD5koTpK8B+/
↪aedsUCAwEAAaNQME4w\nHQYDVR0OBBYEFLTg2FsUFekx9XjIi01BrDpo0aPIMB8GA1UdIwQYMBaAFLTg2FsU\nFekx9XjIi0
↪d2rSRg3hNqOyycPYtdVK1YXEj4Xm91qgL8An3Kui8\njSq1S9+PstGvyh14YUw43Y1VtEPGpNVTvDtkxQ/
↪8rs1sGHbqUxshgFMbqruxp7WH\ns0fxgn5COHEnRC4jQn02wZAk8pIjFVZLkhqdIYBtC35buHr17mXNL0S4H5cJxzPN\nk3
↪bZ6qDDq0BcduhtKiYVPiVw9z3moLuwDb0QDM59zCexpcz\nb/w7K4lIxWqpD5tbpKSmj/
↪3v5TCqez0Mim8/j4q29bP913KQrngnVCdCezOsPWIH\nDDoihgeRQHuz1VuGGZ259Hc=",
    "agency":
↪"MIIDADCCAeigAwIBAgIJAJUF2Dp1a9U6MA0GCSqGSIb3DQEBCwUAMDUxDjAMBgNV\nBAMMBWNoYWluMRMwEQYDVQQKDApma
↪bEpjEXsu2cXH0D6BHZk+wvuxG6\nezXWq5MYjCw3fQiSRWkDYoxzWgulkRyYROF1xoZeNGQssReFmCgP+pcQwRxjcq8z\nI
↪EE3tsJ0ae3ax6zixCT66aV49S27cMcisS+XKP/
↪q\nEVPxhO7SUjnzZY69MgZzNSFxCzIbapnlmYAOS26vIT0taSkoKXmIsYssga45XPwI\n7YBVCc/
↪34kHzW9xrNjyyThMWOgDsuBqZN9xvapGSQ82Lsh7ObN0dZVUCAwEAAaMQ\nMA4wDAYDVR0TBAUwAwEB/
↪zANBgkqhkiG9w0BAQsFAAOCAQEAu3aHxJnCZnICpHbQ\nv1Lc5tiXtAYE9aEP5cxb/
↪cO14xY8dS+t0wiLIvyrE2aTcgImzr4BYNBm1XDt5suc\nMpzha1oJytGv79M9/WnI/
↪BKmgUqTaaXOV2Ux2yPX9SadNcsD9/IbrV0b/
↪hlsPd6M\nK8w7ndowvBgopei+A1NQY6jTDUKif4RxD4u5HZFWUu7pByNLFaydU4qBKVkucXOq\nxmWoupL5XrDk5o490kiz/
↪Zgufqtb4w6oUr3lrQASAbFB3lID/
↪P1ipi0DwX7kZwVX\nECDLYvr+eX6GbTClzn0JGuzqV4OoRo1rrRv+0tp1aLZKpCYn0Lhf6s1iw/
↪kCeM2O\nnP9l2Q=="
}
```


b、失败:


```
{
    "code": 201231,
    "message": "Cert file not found, please check cert path in config",
    "data": "FileNotFound, node cert(node.crt) path prefix error"
}
```

10.2. 获取明文SDK证书与私钥

接口描述

获取Front使用的sdk证书（包含链证书、sdk证书和sdk私钥）的内容

接口URL

http://localhost:5002/WeBASE-Front/cert/sdk

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/cert/sdk
```

响应参数

1）参数表

1）数据格式

a、成功：

```
{
  "sdk.key": "-----BEGIN PRIVATE KEY-----
↪\nMIGEAgEAMBAGByqGSM49AgEGBSuBBAAKBG0wawIBAQQgxqr/d/VgQ0fAr/
↪KvyAeW\nJ6bD1tqxZ5gYOdfIJiK7WOmhRANCAAT3g/OsuSAD2I/dKLWnZTbMGQ8l9WnkD/
↪wr\npyoiQkMy1qI5/3Sj4WFKGcVu9vhsd0nLoP+y1QttYKM0m5QGcuhP\n-----END PRIVATE KEY---
↪--\n",
  "ca.crt": "-----BEGIN CERTIFICATE-----
↪\nMIIBsDCCAVagAwIBAgIJAPwQ7ISyofOIMAoGCCqGSM49BAMCMDUxDjAMBgNVBAMM\nBWNoYWluMRMwEQYDVQQKDApmaXN
↪Hz/Q2SAin5bMnE1nOFMB8GA1UdIwQYMBaAFBBSyZi8k/Hz/
↪Q2SAin5bMnE1nOF\nMAwGA1UdEwQFMAMBAf8wCgYIKoZIzj0EAwIDSAAwRQIgEpuPZypVImOtDty9p50X\njeD4wdgzHXpd
↪----END CERTIFICATE-----\n",
  "sdk.crt": "-----BEGIN CERTIFICATE-----
↪\nMIIBeDCCAR+gAwIBAgIJAJoEtSMUsa8HMAoGCCqGSM49BAMCMDgxEDAOBgNVBAMM\nB2FnZW5jeUExEzARBgNVBAoMCmZp
↪OsuSAD2I/dKLWnZTbMGQ8l9WnkD/wrpyoiQkMy1qI5/
↪3Sj\n4WFKGcVu9vhsd0nLoP+y1QttYKM0m5QGcuhPoxowGDAJBgNVHRMEAjAAMAsGA1Ud\nDwQEAwIF4DAKBggqhkjOPQQD
↪B2a\n+vrHMm6NwtliRAIgRH4gSF0XLmpVOEO21bJFDGWm9siIX0cnj0R3kNGZcB4=\n-----END␣
↪CERTIFICATE-----\n-----BEGIN CERTIFICATE-----
↪\nMIIBcTCCARegAwIBAgIJANrOZ+FrVNpIMAoGCCqGSM49BAMCMDUxDjAMBgNVBAMM\nBWNoYWluMRMwEQYDVQQKDApmaXN
↪OrPsMc9CrrYBsWdwOGhdx\nfNTJA1ss+vngjrhAmWHczvbh+E1WOlDGzpCumeqjEDAOMAwGA1UdEwQFMAMBAf8w\nCgYIKo
↪qHg0e8BG9ptEv7Do8caOPj33F+yOQ==\n-----END CERTIFICATE-----\n-----BEGIN␣
↪CERTIFICATE-----
↪\nMIIBsDCCAVagAwIBAgIJAPwQ7ISyofOIMAoGCCqGSM49BAMCMDUxDjAMBgNVBAMM\nBWNoYWluMRMwEQYDVQQKDApmaXN
↪Hz/Q2SAin5bMnE1nOFMB8GA1UdIwQYMBaAFBBSyZi8k/Hz/
↪Q2SAin5bMnE1nOF\nMAwGA1UdEwQFMAMBAf8wCgYIKoZIzj0EAwIDSAAwRQIgEpuPZypVImOtDty9p50X\njeD4wdgzHXpd
↪----END CERTIFICATE-----\n"
}
```

10.3. 获取SDK证书与私钥压缩包

接口描述

获取Front使用的sdk证书（包含链证书、sdk证书和sdk私钥）的zip压缩包

接口URL

http://localhost:5002/WeBASE-Front/cert/sdk/zip

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/cert/sdk/zip
```

响应参数

1）参数表

1）数据格式

a、成功：

```
headers:  content-disposition: attachment;filename*=UTF-8''conf.zip

{
    // 二进制流
}
```

12.3.11 11. 预编译权限管理

11.1. 查询链是否开启权限

接口描述

通过接口查询链是否开启权限治理功能

注：

权限治理功能需要在启动链时进行配置

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/base/queryChainHasAuth

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/precntauth/authmanager/base/queryChainHasAuth?
↪groupId=g1
```

响应参数

1）数据格式

a、成功1：

```
true
```

b、成功2：

```
false
```

11.2. 查询链环境

接口描述

通过接口查询链是liquid/还是solidity环境

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/base/queryExecEnvIsWasm

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/precntauth/authmanager/base/queryExecEnvIsWasm?
↪groupId=g1
```

响应参数

1）数据格式

a、成功1：

```
true
```

b、成功2：

```
false
```

11.3.查询治理委员信息(everyone可访问)

接口描述

通过接口查询链的治理委员信息

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/everyone/cmtInfo

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5002/WeBASE-Front/precntauth/authmanager/everyone/cmtInfo?
↪groupId=g1
```

响应参数

1）数据格式

a、成功：

```
[
  {
    "governorList": [
      {
        "governorAddress": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
        "weight": 1
      },
      {
        "governorAddress": "0x36c10bfbce3b6550ed92a5ebbb9a44e052bfd285",
        "weight": 2
      }
    ],
    "participatesRate": 100,
```

(下页继续)

(续上页)

```
    "winRate": 90
  }
]
```

b、失败:

```
{
  "code": 500,
  "errorMessage": "get Client failed, e: The group not exist, please check the␣
↪groupID, groupID: g3"
}
```

11.4. 查询合约管理员信息(everyone可访问)

接口描述

通过接口查询某合约的管理员信息

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/everyone/contract/admin

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractAddr": "0xB47fd49b0f1Af2Fce3a1824899b60C2b6A29B851",
  "groupId": "g1"
}
```

响应参数

1）数据格式

a、成功:

```
0x489877b18f93353c67d252c1b8f4b745d41c2107
```

b、失败1，查询群组不存在:

```
{
  "code": 500,
  "errorMessage": "get Client failed, e: The group not exist, please check the␣
↪groupID, groupID: g3"
}
```

失败2，查询合约地址错误:

```
{
  "code": 19,
  "errorMessage": "Call address error"
}
```

失败3，查询合约地址不存在：

```
0x0000000000000000000000000000000000000000
```

11.5. 查询合约函数访问权限(everyone可访问)

接口描述

通过接口查询某用户对某合约函数的访问权限

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/everyone/contract/method/auth

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractAddr": "0xB47fd49b0f1Af2Fce3a1824899b60C2b6A29B851",
  "func": "set",
  "groupId": "g1",
  "userAddress": "0x489877b18f93353c67d252c1b8f4b745d41c2107"
}
```

响应参数

1）数据格式

a、成功：

```
true
```

b、失败1，查询群组不存在：

```
{
  "code": 500,
  "errorMessage": "get Client failed, e: The group not exist, please check the␣
↪groupID, groupID: g3"
}
```

失败2，查询合约地址错误：

```
{
  "code": 19,
  "errorMessage": "Call address error"
}
```

11.6. 查询合约部署权限(everyone可访问)

接口描述

通过接口查询全局合约部署权限

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/everyone/deploy/type

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5002/WeBASE-Front/precntauth/authmanager/everyone/deploy/type?
↪groupId=g1
```

响应参数

1）数据格式

a、成功，可部署:

```
0
```

b、失败1，查询群组不存在:

```
{
  "code": 500,
  "errorMessage": "get Client failed, e: The group not exist, please check the␣
↪groupID, groupID: g0"
}
```

11.7. 查询单一提案信息(everyone可访问)

接口描述

通过接口查询某个提案信息

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/everyone/proposalInfo

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId": "g1",
  "proposalId": 1
}
```

响应参数

1）数据格式

a、成功，可部署：

```
[
  {
    "resourceId": "0xc0523dbdd94ba27e14b0336d799489340ca24cdf",
    "proposer": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
    "proposalType": 31,
    "blockNumberInterval": 604809,
    "status": 2,
    "agreeVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
    ],
    "againstVoters": [],
    "statusString": "finished",
    "proposalTypeString": "resetAdmin"
  }
]
```

b、失败1，查询提案不存在：

```
[
  {
    "resourceId": "0x0000000000000000000000000000000000000000",
    "proposer": "0x0000000000000000000000000000000000000000",
    "proposalType": 0,
    "blockNumberInterval": 0,
    "status": 0,
    "agreeVoters": [],
    "againstVoters": [],
    "statusString": "unknown",
    "proposalTypeString": "unknown"
  }
]
```

11.8. 查询提案总数(everyone可访问)

接口描述

通过接口查询某个提案信息

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/everyone/proposalInfo

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5002/WeBASE-Front/precntauth/authmanager/everyone/
↪proposalInfoCount?groupId=g1
```

响应参数

1）数据格式

a、成功：

```
5
```

b、失败1，查询提案不存在：

```
{
  "code": 500,
  "errorMessage": "get Client failed, e: The group not exist, please check the␣
↪groupID, groupID: g0"
}
```

11.9. 查询提案列表(everyone可访问)

接口描述

通过接口查询某群组提案列表

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/everyone/proposalInfo

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5002/WeBASE-Front/precntauth/authmanager/everyone/proposalInfoList
```

响应参数

1）数据格式

a、成功：

```
[
  {
    "resourceId": "0x02a6340ef1d8a780f0ddf01dde9141cc09f678d6",
    "proposer": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
    "proposalType": "setWeight",
    "blockNumberInterval": 604825,
    "status": "notEnoughVotes",
    "agreeVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
    ],
    "againstVoters": [],
    "proposalId": 5
  },
  {
    "resourceId": "0x36c10bfbce3b6550ed92a5ebbb9a44e052bfd285",
    "proposer": "0x36c10bfbce3b6550ed92a5ebbb9a44e052bfd285",
    "proposalType": "setWeight",
    "blockNumberInterval": 604812,
    "status": "failed",
    "agreeVoters": [
      "0x36c10bfbce3b6550ed92a5ebbb9a44e052bfd285"
    ],
    "againstVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
    ],
    "proposalId": 4
  },
  {
    "resourceId": "0x0000000000000000000000000000000000010001",
    "proposer": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
    "proposalType": "setRate",
    "blockNumberInterval": 604811,
    "status": "finished",
    "agreeVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
    ],
    "againstVoters": [],
    "proposalId": 3
  },
  {
    "resourceId": "0x36c10bfbce3b6550ed92a5ebbb9a44e052bfd285",
    "proposer": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
    "proposalType": "setWeight",
    "blockNumberInterval": 604810,
    "status": "finished",
    "agreeVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
```

(下页继续)

(续上页)


```
    ],
    "againstVoters": [],
    "proposalId": 2
  },
  {
    "resourceId": "0xc0523dbdd94ba27e14b0336d799489340ca24cdf",
    "proposer": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
    "proposalType": "resetAdmin",
    "blockNumberInterval": 604809,
    "status": "finished",
    "agreeVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
    ],
    "againstVoters": [],
    "proposalId": 1
  }
]
```


11.10. 查询用户全局部署权限(everyone可访问)

接口描述

通过接口查询用户全局部署权限

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/everyone/usr/deploy

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式


```
{
  "groupId": "g1",
  "userAddress": "0x489877b18f93353c67d252c1b8f4b745d41c2107"
}
```


响应参数

1）数据格式

a、成功：


```
true
```

11.11. 设置合约的访问权限类型(admin可访问)

接口描述

通过接口设置合约的访问权限类型

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/admin/method/auth/type

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "authType": 1,
  "contractAddr": "4721d1a77e0e76851d460073e64ea06d9c104194",
  "fromAddress": "0xe88ff54644de54fa32ac845c05ed2b7d5677c078",
  "func": "set",
  "groupId": "group0",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

1）数据格式

a、成功:

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

11.12. 设置某用户对合约的访问权限(admin可访问)

接口描述

通过接口设置某用户对合约函数的访问权限

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/admin/method/auth/set

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractAddr": "0xB47fd49b0f1Af2Fce3a1824899b60C2b6A29B851",
  "fromAddress": "0x489877b18f93353c67d252c1b8f4b745d41c2107",
  "func": "set",
  "groupId": "g1",
  "isOpen": true,
  "signUserId": "string",
  "userAddress": "0x489877b18f93353c67d252c1b8f4b745d41c2107"
}
```

响应参数

1）数据格式

a、成功：

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

11.13. 设置某合约的管理员(committee可访问)

接口描述

通过接口设置某合约的管理员

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/committee/contract/admin

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractAddr": "4721d1a77e0e76851d460073e64ea06d9c104194",
  "fromAddress": "0x70da1da76e0e423ec582ec866fae749af67ec4c0",
  "groupId": "group0",
  "newAdmin": "0x70da1da76e0e423ec582ec866fae749af67ec4c0",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

1）数据格式

a、成功：

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

11.14. 设置全局部署权限(committee可访问)

接口描述

通过接口设置全局部署类型

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/committee/deploy/type

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "deployAuthType":1,
  "fromAddress": "",
  "groupId": "group0",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

1）数据格式

a、成功：

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

11.15. 设置治理委员账户(committee可访问)

接口描述

通过接口设置治理委员(新增/更新/删除)

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/committee/governor

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "accountAddress": "0xe88ff54644de54fa32ac845c05ed2b7d5677c078",
  "fromAddress": "",
  "groupId": "group0",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31",
  "weight": 5
}
```

响应参数

1）数据格式

a、成功:

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

11.16. 设置治理阈值(committee可访问)

接口描述

通过接口设置治理阈值

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/committee/rate

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fromAddress": "string",
  "groupId": "group0",
  "participatesRate": 51,
  "signUserId": "5db5a98aef544650aa3864f4cb27af31",
  "winRate": 51
}
```

响应参数

1）数据格式

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

11.17. 对提案投票(committee可访问)

接口描述

通过接口设置对提案进行投票

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/committee/proposal/vote

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "agree": true,
  "fromAddress": "string",
  "groupId": "group0",
  "proposalId": 55,
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

1）数据格式

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

11.18.撤销提案(committee可访问)

接口描述

通过接口设置撤销某提案

接口URL

http://localhost:5002/WeBASE-Front/precntauth/authmanager/committee/proposal/revoke

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fromAddress": "string",
  "groupId": "group0",
  "proposalId": 55,
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

1）数据格式

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

12.3.12 12. 预编译合约管理

12.1. 创建BFS路径

接口描述

通过接口创建BFS

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/bfs/create

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fromAddress": "",
  "groupId": "group0",
  "path": "/apps/test9",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

12.2. 查询BFS路径

接口描述

通过接口查询BFS

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/bfs/query

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId": "group0",
  "path": "/apps"
}
```

响应参数

```
[
  "test",
  "test1"
]
```

12.3. 通过contractName查询合约信息

接口描述

通过groupId和contractName查询合约信息

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/cns/queryCnsByName

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractName": "HelloWorld",
  "groupId": "group0"
}
```

响应参数


```
[
  {
    "name": "HelloWorld",
    "version": "1.0",
    "address": "4721d1a77e0e76851d460073e64ea06d9c104194",
    "abi": "[{\"inputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\
↪"constructor\"},{\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"internalType\":\
↪"string\",\"name\":\"\",\"type\":\"string\"}],\"stateMutability\":\"view\",\
↪"type\":\"function\"},{\"inputs\":[{\"internalType\":\"string\",\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"stateMutability\":\
↪"nonpayable\",\"type\":\"function\"}]"
  },
  {
    "name": "HelloWorld",
    "version": "2.0",
    "address": "4721d1a77e0e76851d460073e64ea06d9c104194",
    "abi": "[{\"inputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\
↪"constructor\"},{\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"internalType\":\
↪"string\",\"name\":\"\",\"type\":\"string\"}],\"stateMutability\":\"view\",\
↪"type\":\"function\"},{\"inputs\":[{\"internalType\":\"string\",\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"stateMutability\":\
↪"nonpayable\",\"type\":\"function\"}]"
  }
]
```


12.4. 通过contractName和version查询合约信息

接口描述

通过groupId、contractName、version查询合约信息

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/cns/queryCnsByNameVersion

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式


```
{
  "contractName": "HelloWorld",
  "groupId": "group0",
  "version": "1.0"
}
```

响应参数

```
{
  "address": "0x4721d1a77e0e76851d460073e64ea06d9c104194",
  "abi": "[{\"inputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\
↪"constructor\"},{\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"internalType\":\
↪"string\",\"name\":\"\",\"type\":\"string\"}],\"stateMutability\":\"view\",\
↪"type\":\"function\"},{\"inputs\":[{\"internalType\":\"string\",\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"stateMutability\":\
↪"nonpayable\",\"type\":\"function\"}]"
}
```

12.5. 通过contractName/groupId/version查询合约地址

接口描述

通过contractName/groupId/version参数查询合约地址

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/cns/queryCnsByNameVersion

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractName": "HelloWorld",
  "groupId": "group0",
  "version": "1.0"
}
```

响应参数

```
0x4721d1a77e0e76851d460073e64ea06d9c104194
```

12.6. 注册合约

接口描述

通过接口注册合约信息

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/cns/reqAddressInfoByNameVersion

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "abiData": "[{\"inputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\
↪"constructor\"},{\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"internalType\":\
↪"string\",\"name\":\"\",\"type\":\"string\"}],\"stateMutability\":\"view\",\
↪"type\":\"function\"},{\"inputs\":[{\"internalType\":\"string\",\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"stateMutability\":\
↪"nonpayable\",\"type\":\"function\"}]",
  "contractAddress": "4721d1a77e0e76851d460073e64ea06d9c104194",
  "contractName": "HelloWorld",
  "contractVersion": "1.0",
  "fromAddress": "",
  "groupId": "group0",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

12.7. 查询共识节点列表

接口描述

通过接口查询共识节点列表

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/consensus/list

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId": "group0",
  "pageNumber": 1,
  "pageSize": 5
}
```

响应参数

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "nodeId":
↪"6447e978505cafd05fc99b731d8fdff31fb07a3c6e9679054fb1880ae6f58aeff638eacfe082d54adca93086c2986b
↪",
      "nodeType": "sealer",
      "weight": 1
    },
    {
      "nodeId":
↪"b14bd4a225db308da3f395c69f12ce06f191ff19941d52eebf30cfb5fc979422ad086fedb0378fdcfbcb4630416e71
↪",
      "nodeType": "sealer",
      "weight": 1
    },
    {
      "nodeId":
↪"848883c435d5c7e32da7744ffb0659538995994a42c24ec7da81a2fd58cd28e76fbaaf603b81f9134d22f57d112cdb
↪",
      "nodeType": "sealer",
      "weight": 1
    },
    {
      "nodeId":
↪"5007b294c7aadd22d62e0c5e33bae14ee6ec0230ebd34df23f29f0330272f6021fd3a8f2b4a4789f1e2fe7fbc8581c
↪",
      "nodeType": "sealer",
      "weight": 1
    }
  ],
  "totalCount": 4
}
```

12.8. 修改共识节点类型

接口描述

通过接口查询共识节点列表

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/consensus/manage

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fromAddress": "0x2abd2fc35c4553b1f1aa6cf70a4e6ef30b4d53a2",
  "groupId": "group0",
  "nodeId":
↪"5007b294c7aadd22d62e0c5e33bae14ee6ec0230ebd34df23f29f0330272f6021fd3a8f2b4a4789f1e2fe7fbc8581c
↪",
  "nodeType": "observer",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31",
  "weight": 1
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

12.9. 建表

接口描述

通过接口插入新的表结构

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/kvtable/reqCreateTable

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fromAddress": "string",
  "groupId": "group0",
  "keyFieldName": "myKey",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31",
  "tableName": "test_table",
  "valueFields": [
    "valueIsData"
  ]
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

12.10. 写表

接口描述

通过接口在表插入数据

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/kvtable/reqSetTable

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fieldNameToValue": {
    "key1": "hi",
    "key2": "hello",
    "key3": "how are u"
  },
  "fromAddress": "string",
  "groupId": "group0",
  "key": "myKey",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31",
  "tableName": "test_table"
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

12.11. 读表

接口描述

通过接口在表读取数据

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/kvtable/reqGetTable

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId": "group0",
  "key": "myKey",
  "tableName": "test_table"
}
```

响应参数

```
{
  "key2": "hello",
  "key1": "hi",
  "key3": "how are u"
}
```

12.12. 获取群组系统配置

接口描述

通过接口读取某个群组的系统配置

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/sys/config/list

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5002/WeBASE-Front/precntauth/precompiled/sys/config/list?
↪groupId=group0
```

响应参数

```
[
  {
    "groupId": "group0",
    "configKey": "tx_count_limit",
    "configValue": "10"
  },
  {
    "groupId": "group0",
    "configKey": "tx_gas_limit",
    "configValue": "300000002"
  }
]
```

12.13. 设置群组系统配置

接口描述

通过接口设置某个群组的系统配置

接口URL

http://localhost:5002/WeBASE-Front/precntauth/precompiled/sys/config/list

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "configKey": "tx_count_limit",
  "configValue": "5",
  "fromAddress": "string",
```

(下页继续)

(续上页)

```
  "groupId": "group0",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

12.3.13 附录

1. 返回码信息列表

Precompiled Service API 错误码

3.调用接口遇到问题&解决方式

Q1：合约方法入参byte32类型， java 通过HTTP+JSON调用，对应java bean 该如何对应数据类型？ 报错

```
{
    "code":201153,
    "data":null,
    "errorMessage":"unable to create instance of        type:org.fisco.bcos.sdk.
↪abi.datatypes.generated.Bytes32"
}
```

A1：java bean对应还是由String 存储，而非byte[]。 java工程引用如下依赖

```
<dependency>
        <groupId>org.fisco-bcos.java-sdk</groupId>
        <artifactId>fisco-bcos-java-sdk</artifactId>
        <version>2.7.2</version>
</dependency>
```

java 代码处理

```
String username = "hello";
Bytes32 bytes32 = CommonUtils.utf8StringToBytes32(username);
String bytes32Str = Numeric.toHexString(bytes32.getValue());
```

注意：Numeric 工具类是fisco-bcos-java-sdk 自带的，CommonUtils 是WEBASE Front 里面的

12.4 升级说明

WeBASE-Front升级的兼容性说明，请结合WeBASE-Front Changelog进行阅读

WeBASE-Front升级的必须步骤：

1. 备份已有文件或数据，下载新的安装包（可参考安装包下载）
2. 采用新的安装包，并将旧版本yml已有配置添加到新版本yml中；可通过`diff aFile bFile`命令对比新旧yml的差异

3. bash stop.sh && bash start.sh重启

12.5 附录

12.5.1 1. 安装问题

1.1 Java部署

CentOS环境安装Java

注意：CentOS下OpenJDK无法正常工作，需要安装OracleJDK下载链接。

```
# 创建新的文件夹，安装Java 8或以上的版本，将下载的jdk放在software目录
# 从Oracle官网(https://www.oracle.com/technetwork/java/javase/downloads/index.html)选择Java 8或以上的版本下载，例如下载jdk-8u201-linux-x64.tar.gz
$ mkdir /software

# 解压jdk
$ tar -zxvf jdk-8u201-linux-x64.tar.gz

# 配置Java环境，编辑/etc/profile文件
$ vim /etc/profile

# 打开以后将下面三句输入到文件里面并保存退出
export JAVA_HOME=/software/jdk-8u201  #这是一个文件目录，非文件
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar

# 生效profile
$ source /etc/profile

# 查询Java版本，出现的版本是自己下载的版本，则安装成功。
java -version
```

Ubuntu环境安装Java

```
  # 安装默认Java版本(Java 8或以上)
  sudo apt install -y default-jdk
  # 查询Java版本
  java -version
```

1.2 Gradle部署

此处给出简单步骤，供快速查阅。更详细的步骤，请参考官网。

（1）从官网下载对应版本的Gradle安装包，并解压到相应目录

```
mkdir /software/
unzip -d /software/ gradleXXX.zip
```

（2）配置环境变量

```
export GRADLE_HOME=/software/gradle-4.9
export PATH=$GRADLE_HOME/bin:$PATH
```

（3）查看版本

```
gradle -version
```

12.5.2 2. 常见问题

- 1：执行shell脚本报错误"permission denied"或格式错误

```
赋权限：chmod + *.sh
转格式：dos2unix *.sh
```

- 2：eclipse环境编译源码失败，错误提示如下：

```
...
/data/temp/WeBASE-Front/src/main/java/com/webank/webase/front/performance/
↪PerformanceService.java:167: error: cannot find symbol
        log.info("begin sync performance");
        ^
  symbol:   variable log
  location: class PerformanceService
Note: /data/temp/WeBASE-Front/src/main/java/com/webank/webase/front/contract/
↪CommonContract.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
100 errors

> Task :compileJava FAILED

FAILURE: Build failed with an exception.
...
```

答：问题是不能编译Lombok注解，修改build.gradle文件，将以下代码的注释加上

```
 //annotationProcessor 'org.projectlombok:lombok:1.18.6'
```

- 3：节点运行一段时间后新增了一个群组，前置查不到新群组的信息。

  答：调用 http://{ip}:{port}/WeBASE-Front/1/web3/refresh 方法，即可手动更新。

- 4：升级1.0.2版本时，数据库报错：

```
Caused by: org.h2.jdbc.JdbcSQLException: NULL not allowed for column "TYPE";␣
↪SQL statement:
alter table key_store_info add column type integer not null [23502-197]
        at org.h2.message.DbException.getJdbcSQLException(DbException.
↪java:357) ~[h2-1.4.197.jar:1.4.197]
        at org.h2.message.DbException.get(DbException.java:179) ~[h2-1.4.197.
↪jar:1.4.197]
        at org.h2.message.DbException.get(DbException.java:155) ~[h2-1.4.197.
↪jar:1.4.197]
```

  答：将H2数据库删除（在h2目录下），或者配置新数据库名，在 application.yml 文件中的配置如下：

```
spring:
  datasource:
    url: jdbc:h2:file:./h2/webasefront;DB_CLOSE_ON_EXIT=FALSE // 默认H2库
为webasefront
...
```

- 5：日志报以下错误信息：

```
2019-08-08 17:29:05.505 [pool-11-thread-1] ERROR TaskUtils
↪$LoggingErrorHandler() - Unexpected error occurred in scheduled task.
org.hyperic.sigar.SigarFileNotFoundException: 没有那个文件或目录
        at org.hyperic.sigar.FileSystemUsage.gather(Native Method) ~[sigar-1.6.
↪4.jar:?]
        at org.hyperic.sigar.FileSystemUsage.fetch(FileSystemUsage.java:30) ~
↪[sigar-1.6.4.jar:?]
        at org.hyperic.sigar.Sigar.getFileSystemUsage(Sigar.java:667) ~[sigar-
↪1.6.4.jar:?]
```

答：监控目录不存在，需配置节点所在磁盘目录，在 application.yml 文件中的配置如下：

```
...
constant:
  monitorDisk: /              // 要监控的磁盘目录，配置节点所在目录（如：/home）
...
```

- 6：启动报错“nested exception is javax.net.ssl.SSLException”：

```
...
nested exception is javax.net.ssl.SSLException: Failed to initialize the client-
↪side SSLContext: Input stream not contain valid certificates.
```

答：CentOS的yum仓库的OpenJDK缺少JCE(Java Cryptography Extension)，导致Web3SDK/Java-SDK无法正常连接区块链节点，因此在使用CentOS操作系统时，推荐使用*OracleJDK*。

- 7：启动报错“Processing bcos message timeout”

```
...
[main] ERROR SpringApplication() - Application startup failed
org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating␣
↪bean with name 'contractController': Unsatisfied dependency expressed through␣
↪field 'contractService'; nested exception is org.springframework.beans.factory.
↪UnsatisfiedDependencyException: Error creating bean with name 'contractService':␣
↪Unsatisfied dependency expressed through field 'web3jMap'; nested exception is␣
↪org.springframework.beans.factory.BeanCreationException: Error creating bean␣
↪with name 'web3j' defined in class path resource [com/webank/webase/front/config/
↪Web3Config.class]: Bean instantiation via factory method failed; nested␣
↪exception is org.springframework.beans.BeanInstantiationException: Failed to␣
↪instantiate [java.util.HashMap]: Factory method 'web3j' threw exception; nested␣
↪exception is java.io.IOException: Processing bcos message timeout
...
```

答：一些OpenJDK版本缺少相关包，导致节点连接异常。推荐使用*OracleJDK*。

- 8：启动失败，日志却没有异常

```
================================================================================================
Starting Server com.webank.webase.front.Application Port 5002 ....................
↪...........[Failed]. Please view log file (default path:./log/).
Because port 5002 not up in 20 seconds.Script finally killed the process.
================================================================================================
```

答：确认机器是否满足硬件要求。机器性能过低会导致服务端口一定时间内没起来，脚本会自动杀掉进程。可以尝试手动修改dist目录下的start.sh脚本，将启动等待时间设置久一点（默认600，单位：秒），然后启动。

```
...
startWaitTime=600
...
```

- 9：启动报错SSLContext: null

答：确保conf/目录下包含sdk证书； 若使用的是v1.5.0以前的版本，则需要保证ca.crt, node.crt, node.key；其中node.crt, node.key为sdk.crt, sdk.key复制并重命名得到；若使用v1.5.0及以上版本，则需要复制链的sdk目录下的所有文件(ca.crt, sdk.crt, sdk.key及gm文件夹)到前置服务的conf目录

12.5.3 3. 使用说明

测试用户管理

3.1. 导入私钥

支持txt文件和pem文件导入测试用户的私钥信息

导入.txt私钥内容格式示例：

```
{
  "address":"0x06f81c8e1cb59b5df2cdeb87a212d17fba79aad7",
  "publicKey":
↪"0x4b1041710a4427dc1c0d542c8f0fd312d92b0d03a989f512d1f8d3cafb851967f3592df0035e01fa63b2626165d0
↪",
  "privateKey":"71f1479d9051e8d6b141a3b3ef9c01a7756da823a0af280c6bf62d18ee0cc978",␣
↪// 十六进制
  "userName":"111",
  "type":0  // type为0，不可修改
}
```

其中用户类型为0代表用户为WeBASE-Front的本地私钥用户，导入的私钥均为该类型；

导入.pem私钥内容示例：

```
-----BEGIN PRIVATE KEY-----
MIGEAgEAMBAGByqGSM49AgEGBSuBBAAKBG0wawIBAQQgC8TbvFSMA9y3CghFt51/
XmExewlioX99veYHOV7dTvOhRANCAASZtMhCTcaedNP+H7iljbTIqXOFM6qm5aVs
fM/yuDBK2MRfFbfnOYVTNKyOSnmkY+xBfCR8Q86wcsQm9NZpkmFK
-----END PRIVATE KEY-----
```

其中pem文件开头的-----BEGIN PRIVATE KEY-----\n和结尾的\n-----END PRIVATE KEY-----\n格式不可更改，否则将读取pem文件失败

3.2. 导出私钥

目前仅支持导出测试用户的txt格式私钥

Java中如何使用导出的私钥

以上文中的私钥加载：

基于javasdk的私钥加载：

```
@Test
public void testCrypto() {
    // 1-国密，0-ECDSA
    CryptoSuite cryptoSuite = new CryptoSuite(1);
    CryptoKeyPair keyPair = cryptoSuite.createKeyPair(
↪"e843a542a7a8240f9c9e418b9517c2c8f4dc041a11a44e614a3b026c3588c188");
    System.out.println("privateKey: " + keyPair.getHexPrivateKey());
    System.out.println("address: " + keyPair.getAddress());
    System.out.println("publicKey: " + keyPair.getHexPublicKey());
}
```

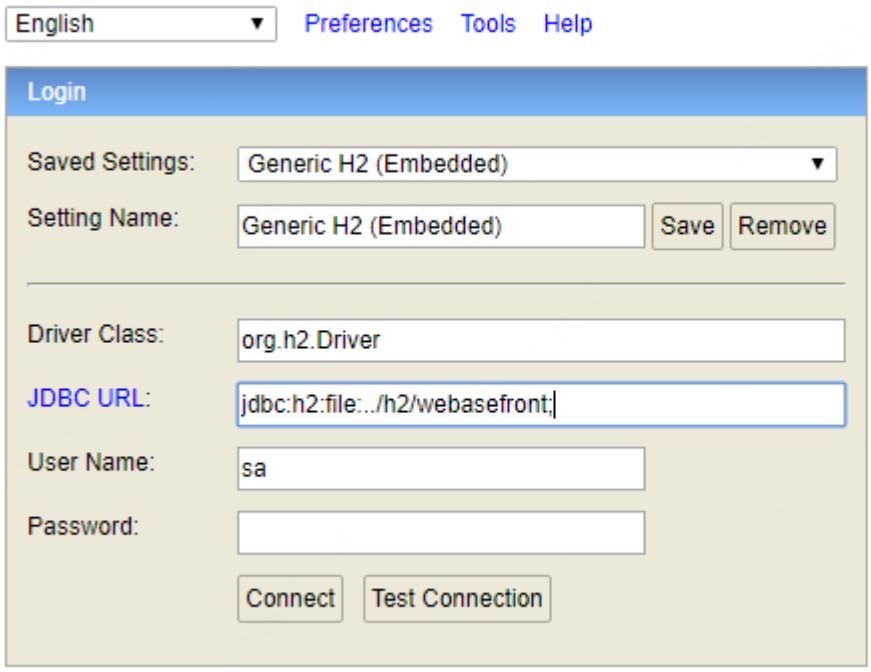
基于web3sdk的私钥加载：

```
@Test
public void loadPrivateKeyTest() {
  CryptoSuite
  String privateKey =
↪"71f1479d9051e8d6b141a3b3ef9c01a7756da823a0af280c6bf62d18ee0cc978";
  Credentials credentials = GenCredential.create(privateKey);
  // private key 实例
  BigInteger privateKeyInstance = credentials.getEcKeyPair().getPrivateKey();
  System.out.println(Numeric.toHexStringNoPrefix(privateKeyInstance));
  // public key 实例
  BigInteger publicKeyInstance = credentials.getEcKeyPair().getPublicKey();
  System.out.println(Numeric.toHexString(publicKeyInstance));
  // address 地址
  String address = credentials.getAddress();
  System.out.println(address);
}
```

访问h2数据库

WeBASE-Front采用 JPA + H2数据库 的方式保存数据

- 源码查看各个**数据表**的内容：需要通过查看WeBASE-Front源码的各个包中带有`@Entity`注解的entity实体类；如，查看私钥数据表`KeyStoreInfo`则查看该文件`com.webank.webase.front.keystore.entity.KeyStoreInfo.java`
- 通过H2控制台**连接H2数据库**：

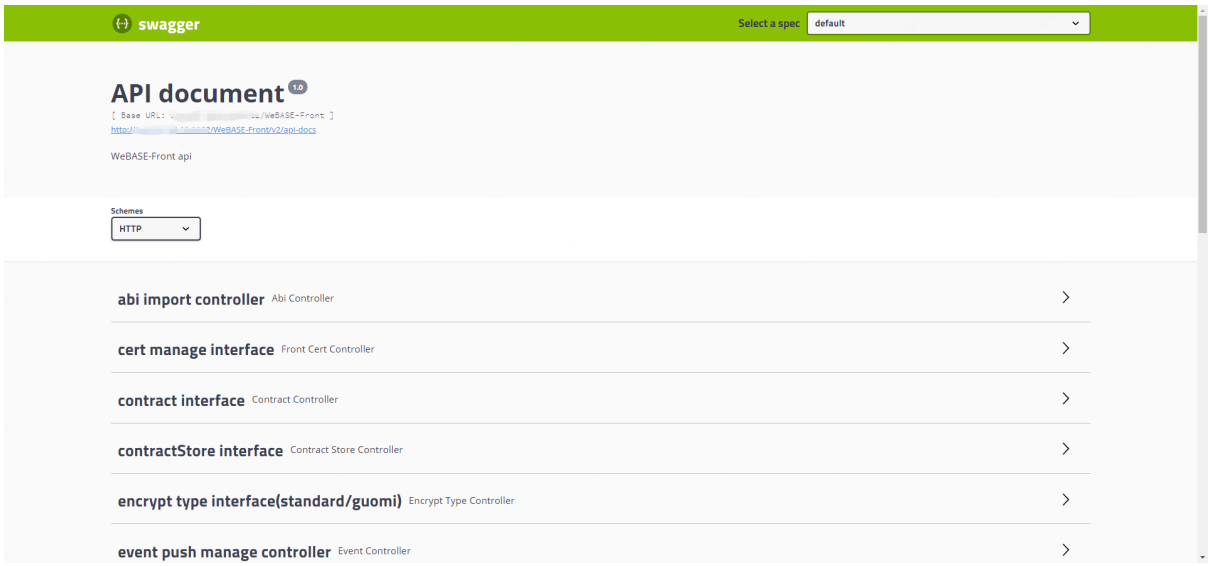


- 同机H2访问：可以通过浏览器打开`localhost:5002/WeBASE-Front/console`，以默认配置为例填入连接参数
  - `JDBC URL`应填入`file:../h2/webasefront;`，与前置服务的application.yml中配置的`spring.datasource.url`对应
  - 若未设置用户名与密码，则默认用户名为`sa`，密码为空
- 服务端H2访问：

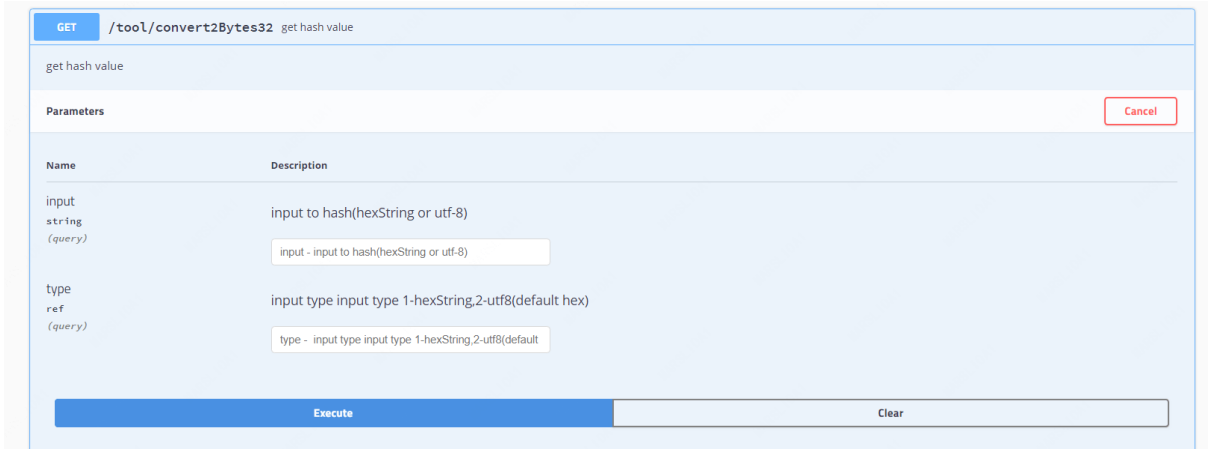
- 修改前置服务的application.yml中的`spring.h2.console.settings.web-allow-others`设为`true`，允许远端访问H2控制台
- 重启前置服务
- 访问`{ip}:{port}/WeBASE-Front/console`，参数填入方法同上

使用swagger

节点前置搭配了swagger，可用于直接调试接口，通过访问`{ip}:5002/WeBASE-Front/swagger-ui.html`即可访问前置的swagger页面



在swagger页面中选中一个接口后，点击“Try it out”既可以开始调用了，输入框将提示入参的格式



两阶段交易

在v1.5.2后，WeBASE-Front中丰富了组装交易的接口，包括了本地签名组装交易接口`/trans/convertRawTxStr/local`和通过WeBASE-Sign组装交易接口`/trans/convertRawTxStr/withSign`，下面以本地签名举例（接口的具体入参可参考对应接口文档）。

本地签名组装交易需要填入的参数包含合约地址、函数名及函数入参、群组ID和WeBASE-Front的私钥用户地址等，如下所示

```
{
    "user":"0x2db346f9d24324a4b0eac7fb7f3379a2422704db",
```

(下页继续)

(续上页)

```
    "contractName":"HelloWorld",
    "contractAddress":"dasdfav23rf213vbcdvadf3bcdf2fc23rqde",
    "funcName":"set",
    "contractAbi":[{"constant":true,"inputs":[],"name":"getVersion","outputs":[{
↪"name":"","type":"string"}],"payable":false,"stateMutability":"view","type":
↪"function"},{"constant":true,"inputs":[],"name":"getStorageCell","outputs":[{
↪"name":"","type":"string"},{"name":"","type":"string"}],"payable":false,
↪"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":
↪"n","type":"string"}],"name":"setVersion","outputs":[],"payable":false,
↪"stateMutability":"nonpayable","type":"function"},{"inputs":[{"name":"storageHash
↪","type":"string"},{"name":"storageInfo","type":"string"}],"payable":false,
↪"stateMutability":"nonpayable","type":"constructor"}],
    "funcParam":["Hi,Welcome!"],
    "groupId" :"1",
    "useCns": false
}
```

调用组装交易接口后，接口将返回签名后的交易体编码值（在测试接口时，可以通过节点前置的Swagger直接发起请求）：

```
0xf9012da001071041dddc1b3c553b48c0fbefecc07f3812f5ce4004d47708f1c3342844db018405f5e10082029d94e10
```

通过已签名交易发送接口`/trans/signed-transaction`将交易编码值发到链上，接口将返回交易回执，可根据交易回执的`status`判断交易是否成功

注：若发起的是查询交易，除了上述接口的两阶段调用方法之外，还可以使用合约函数的编码值接口`/trans/encodeFunction`获取`encodedFunction`值。 根据入参要求调用已编码查询交易发送接口`/trans/query-transaction`，即可发送查询交易，接口将返回查询的返回值。

12.5.4 4. 支持链上事件订阅和通知

在某些业务场景中，应用层需要实时获取链上的事件，如出块事件、合约Event事件等。应用层通过WeBASE连接节点后，**由于无法和节点直接建立长连接**，难以实时获取链上的消息。

支持通过**消息队列**(Message Queue)来获取WeBASE-Front(v1.2.3+)的链上事件的消息推送

目前支持出块事件与智能合约Event事件的事件Push通知，大致流程为：

1. WeBASE-Front连接到MQ-Server(目前支持RabbitMQ-Server);
2. WeBASE-Front接收节点的事件Push后，如出块通知，WeBASE-Front将出块消息发送到消息队列中;
3. 区块链应用连接MQ-Server，获取消息队列中待消费的消息，即可获得事件通知;

下面介绍如何搭建RabbitMQ的消息队列服务与WeBASE-Front的配置方法

12.5.5 5. 配置文件解析

- 1. 配置文件解析

12.6 Liquid配置

12.6.1 安装rust

```
# 结尾追加export使用镜像源
$ vi /etc/profile

export RUSTUP_DIST_SERVER=https://mirrors.ustc.edu.cn/rust-static
export RUSTUP_UPDATE_ROOT=https://mirrors.ustc.edu.cn/rust-static/rustup


# 此命令将会自动安装 rustup，rustup 会自动安装 rustc 及 cargo
$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

Rust is installed now. Great!

To get started you may need to restart your current shell.
This would reload your PATH environment variable to include
Cargo's bin directory ($HOME/.cargo/bin).

# 执行一次source使环境变量生效
$ source $HOME/.cargo/env
```

检查版本

检查rustc和cargo版本，确认安装成功

```
$ rustc --version

$ cargo --version
```

此外需要安装以下工具链组件：

```
$ rustup toolchain install nightly-2021-06-23 --component rust-src rustc-dev llvm-
↪tools-preview
$ rustup default nightly-2021-06-23
$ rustup target add wasm32-unknown-unknown
```

> 如果当前网络无法访问Rustup官方镜像，请参考 Rustup 镜像安装帮助 更换镜像源。
> https://mirrors.tuna.tsinghua.edu.cn/help/rustup/

cargo 更换镜像源

```
# 编辑cargo配置文件，若没有则新建
$ vim $HOME/.cargo/config

[source.crates-io]
registry = "https://github.com/rust-lang/crates.io-index"
replace-with = 'ustc'
[source.ustc]
registry = "git://mirrors.ustc.edu.cn/crates.io-index"
```

确保配置 cmake 环境，Linux可以通过以下命令安装：

要求安装 cmake 3.12及以上版本，gcc 7及以上版本

```
# CentOS请执行下面的命令
$ sudo yum install cmake3
# Ubuntu请执行下面的命令
$ sudo apt install cmak
```

如果centos的yum资源无cmake3，则需要手动下载cmake3进行配置

以下载cmake 3.21.3版本为例，到cmake官网下载包后，解压到目录如/data/home/webase目录，并修改/etc/profile，设置cmake环境变量

```
$ vi /etc/profile

export CMAKE3_HOME=//data/home/webase/cmake-3.21.3-linux-x86_64
export PATH=$PATH:$CMAKE3_HOME/bin

# 环境变量生效
source /etc/profile
```

安装 cargo-liquid

安装前，如果使用的是centos，执行下文命令以确保依赖符合要求，具体可参考Cargo-Liquid issue 14

```
#请确保cmake版本大于3.12
#请参考下述命令使用gcc7
$ sudo yum install -y epel-release centos-release-scl
$ sudo yum install -y devtoolset-7

# 启用devtool
$ source /opt/rh/devtoolset-7/enable

# 参考下述命令使用要求版本的rust工具链
$ rustup toolchain install nightly-2021-06-23 --component rust-src rustc-dev llvm-
↪tools-preview
$ rustup default nightly-2021-06-23
```

确保上述工具版本符合要求后，并执行以上命令尝试安装：

cargo-liquid 是用于辅助开发 Liquid 智能合约的命令行工具，在终端中执行以下命令安装：

```
# 通过gitee 安装
$ cargo install --git https://gitee.com/WeBankBlockchain/cargo-liquid --tag v1.0.0-
↪rc2 --force

# 通过github安装
$ cargo install --git https://github.com/WeBankBlockchain/cargo-liquid --tag v1.0.
↪0-rc2 --force
```

开始安装后：

```
Updating git repository `https://gitee.com/WeBankBlockchain/cargo-liquid`
Installing cargo-liquid v1.0.0-rc2 (https://gitee.com/WeBankBlockchain/cargo-
↪liquid?tag=v1.0.0-rc2#5da4da65)
Updating `git://mirrors.ustc.edu.cn/crates.io-index` index
Fetch [=======>                 ]  34.20%, 5.92MiB/s
```

如果下载crates失败，可重新执行cargo install命令**重试**下载

执行成功后

```
Compiling wabt v0.10.0
Finished release [optimized] target(s) in 1m 33s
```

(下页继续)

(续上页)

```
Installing /data/home/webase/.cargo/bin/cargo-liquid
Installing /data/home/webase/.cargo/bin/liquid-analy
Installed package `cargo-liquid v1.0.0-rc2 (https://gitee.com/WeBankBlockchain/
↪cargo-liquid?tag=v1.0.0-rc2#5da4da65)` (executables `cargo-liquid`, `liquid-
↪analy`)
```

至此liquid依赖安装完成

wasm-opt优化.wasm文件大小

安装 Binaryen（可选，推荐安装，优化编译）

Binaryen 项目中包含了一系列 Wasm 字节码分析及优化工具，其中如 wasm-opt 等工具会在 Liquid 智能合约的构建过程中使用。请参考其官方文档。https://github.com/WebAssembly/binaryen#building

除根据官方文档的编译安装方式外，

- Ubuntu下可通过 `sudo apt install binaryen` 下载安装（如使用Ubuntu，则系统版本不低于20.04）
- 其他操作系统可参照此处查看是否可直接通过包管理工具安装 https://pkgs.org/download/binaryen
- Mac下可直接通过 `brew install binaryen` 下载安装binaryen。

下面介绍CentOS安装方式：

```
# 下载其rpm包
$ wget https://download-ib01.fedoraproject.org/pub/epel/7/x86_64/Packages/b/
↪binaryen-104-1.el7.x86_64.rpm
# 安装rpm包
$ sudo rpm -ivh binaryen-104-1.el7.x86_64.rpm
```

12.6.2 例子：HelloWorld

以简单的 HelloWorld 合约为示例，帮助读者快速建立对 Liquid 合约的直观认识。

```
#![cfg_attr(not(feature = "std"), no_std)]

use liquid::storage;
use liquid_lang as liquid;

#[liquid::contract]
mod hello_world {
    use super::*;

    #[liquid(storage)]
    struct HelloWorld {
        name: storage::Value<String>,
    }

    #[liquid(methods)]
    impl HelloWorld {
        pub fn new(&mut self) {
            self.name.initialize(String::from("Alice"));
        }

        pub fn get(&self) -> String {
            self.name.clone()
        }
```

(下页继续)

(续上页)

```
        pub fn set(&mut self, name: String) {
            self.name.set(name)
        }
    }

    #[cfg(test)]
    mod tests {
        use super::*;

        #[test]
        fn get_works() {
            let contract = HelloWorld::new();
            assert_eq!(contract.get(), "Alice");
        }

        #[test]
        fn set_works() {
            let mut contract = HelloWorld::new();

            let new_name = String::from("Bob");
            contract.set(new_name.clone());
            assert_eq!(contract.get(), "Bob");
        }
    }
}
```

CHAPTER 13

节点管理服务

13.1 概要介绍

13.1.1 1. 功能说明

WeBASE-Node-Manager可以是处理前端页面所有web请求，管理各个节点的状态，管理链上所有智能合约，对区块链的数据进行统计、分析，对异常交易的审计，私钥管理等，含有如下功能模块：

安装详情可查看下一章节的WeBASE-Node-Manager部署说明

13.2 部署说明

13.2.1 1. 前提条件

13.2.2 2. 注意事项

重要： FISCO-BCOS 2.0与3.0对比、JDK版本、WeBASE及其他子系统的版本兼容说明！请查看

- Java推荐使用OracleJDK，JDK配置指引
- 在服务搭建的过程中，如碰到问题，请查看 常见问题解答
- 安全提示： 强烈建议设置复杂的数据库登录密码，且严格控制数据操作的权限和网络策略

通过WeBASE-Sign私钥管理 WeBASE-Node-Manager v1.3.0及以上版本将通过WeBASE-Sign进行私钥管理，即使用WeBASE-Node-Manager v1.3.0+的版本需要同步安装WeBASE-Sign v1.3.0，详情可参考升级文档进行阅读

13.2.3 3. 拉取代码

执行命令：

```
git clone -b master-3.0 https://github.com/WeBankBlockchain/WeBASE-Node-Manager.git

# 若因网络问题导致长时间下载失败，可尝试以下命令
git clone -b master-3.0 https://gitee.com/WeBank/WeBASE-Node-Manager.git
```

进入目录：

```
cd WeBASE-Node-Manager
git checkout lab
```

13.2.4 4. 编译代码

方式一：如果服务器已安装Gradle，且版本为gradle-4.10至gradle-6.x版本

```
gradle build -x test
```

方式二：如果服务器未安装Gradle，或者版本不是gradle-4.10至gradle-6.x版本，使用gradlew编译

```
chmod +x ./gradlew && ./gradlew build -x test
```

构建完成后，会在根目录WeBASE-Node-Manager下生成已编译的代码目录dist。

13.2.5 5. 数据库初始化

5.1 新建数据库

```
#登录MySQL:
mysql -u ${your_db_account} -p${your_db_password}  例如：mysql -u root -p123456
#新建数据库:
CREATE DATABASE IF NOT EXISTS {your_db_name} DEFAULT CHARSET utf8 COLLATE utf8_
↪general_ci;
```

5.2 修改脚本配置

进入数据库脚本目录

```
cd  dist/script
```

修改数据库连接信息：

```
修改数据库名称：sed -i "s/webasenodemanager/${your_db_name}/g" webase.sh
修改数据库用户名：sed -i "s/defaultAccount/${your_db_account}/g" webase.sh
修改数据库密码：sed -i "s/defaultPassword/${your_db_password}/g" webase.sh
```

例如：将数据库用户名修改为root，则执行：

```
sed -i "s/defaultAccount/root/g" webase.sh
```

5.3 运行数据库脚本

执行命令：bash webase.sh ${dbIP} ${dbPort} 如：

```
bash webase.sh 127.0.0.1 3306
```

13.2.6 6. 服务配置及启停

6.1 服务配置修改

（1）回到dist目录，dist目录提供了一份配置模板conf_template：

```
根据配置模板生成一份实际配置conf。初次部署可直接拷贝。
例如：cp conf_template conf -r
```

（2）修改服务配置：

```
修改服务端口：sed -i "s/5001/${your_server_port}/g" conf/application.yml
修改数据库IP：sed -i "s/127.0.0.1/${your_db_ip}/g" conf/application.yml
修改数据库端口：sed -i "s/3306/${your_db_port}/g" conf/application.yml
修改数据库名称：sed -i "s/webasenodemanager/${your_db_name}/g" conf/application.yml
修改数据库用户：sed -i "s/defaultAccount/${your_db_account}/g" conf/application.yml
修改数据库密码：sed -i "s/defaultPassword/${your_db_password}/g" conf/application.yml
```

6.2 服务启停

在dist目录下执行：

```
启动：bash start.sh
停止：bash stop.sh
检查：bash status.sh
```

备注：服务进程起来后，需通过日志确认是否正常启动，出现以下内容表示正常；如果服务出现异常，确认修改配置后，重启提示服务进程在运行，则先执行stop.sh，再执行start.sh。

```
...
        Application() - main run success...
```

6.3 查看日志

在dist目录查看：

```
全量日志：tail -f log/WeBASE-Node-Manager.log
错误日志：tail -f log/WeBASE-Node-Manager-error.log
```

13.3 接口说明

13.3.1 1 前置管理模块

1.1 新增节点前置信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/front/new**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/front/new
```

```
{
    "frontIp": "127.0.0.1",
    "frontPort": "5002"
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
        "code": 0,
        "message": "success",
        "data": {
                "frontId": 6,
                "nodeId":
↪"6e24cc5a21a41eab08636f8bbde0c93dd4a9ce4f16fa7fbaed0174872716dd1463d3ce822340d0f51badd2e43c9c10
↪",
                "frontIp": "127.0.0.1",
                "frontPort": 5022,
                "agency": null,
                "groupList": null,
                "clientVersion": null,
                "supportVersion": null,
                "frontVersion": "lab-rc1",
                "signVersion": "v2.0.0-lab",
                "createTime": null,
                "modifyTime": null,
                "status": 1,
                "runType": 0,
                "agencyId": null,
                "agencyName": null,
                "hostId": null,
                "hostIndex": null,
                "imageTag": null,
                "containerName": null,
                "jsonrpcPort": null,
                "p2pPort": null,
                "channelPort": null,
                "chainId": 0,
                "chainName": "default"
        },
        "attachment": null
}
```

- 失败：

```
{
    "code": 102000,
```

(下页继续)

(续上页)

```
    "message": "system exception",
    "data": {}
}
```

1.2 获取所有前置列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/front/find?frontId={frontId}&groupId={groupId}&frontStatus={frontStatus}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/front/find
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
        "code": 0,
        "message": "success",
        "data": [{
                "frontId": 2,
                "nodeId":
↪"5e4624ac3929babcba542088c0029d6a055f42e03f28b3c9f2c591c2fd0c0ce8c6f1f8bef92131f3acdcee3a4d5698
↪",
                "frontIp": "127.0.0.1",
                "frontPort": 5002,
                "agency": null,
                "groupList": null,
                "clientVersion": null,
                "supportVersion": null,
                "frontVersion": "string",
                "signVersion": "String",
                "createTime": "2021-12-01 12:49:32",
                "modifyTime": "2021-12-01 12:49:32",
                "status": 1,
                "runType": 0,
                "agencyId": null,
                "agencyName": null,
                "hostId": null,
                "hostIndex": null,
                "imageTag": null,
```

(下页继续)

(续上页)

```
                "containerName": null,
                "jsonrpcPort": null,
                "p2pPort": null,
                "channelPort": null,
                "chainId": 0,
                "chainName": "default"
        }],
        "totalCount": 1
}
```

- 失败：

```
{
   "code": 102000,
   "message": "system exception",
   "data": {}
}
```

1.3 删除前置信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/front/{frontId}**
- 请求方式：DELETE
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/front/500001
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "data": {},
    "message": "Success"
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

1.4 刷新前置信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/front/refresh**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

无

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/front/refresh
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "data": {},
    "message": "Success"
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

13.3.2 2 交易信息模块

2.1 查询交易信息列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：

```
/transaction/transList/{groupId}/{pageNumber}/{pageSize}?transactionHash=
↪{transactionHash}&blockNumber={blockNumber}
```

- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/transaction/transList/group/1/10?
↪transactionHash=0x303daa78ebe9e6f5a6d9761a8eab4bf5a0ed0b06c28764488e4716de42e1df01
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "transHash":
↪"0x158c20a61d5ee01f64f2fb2b7f752f834b6a64c1687ac2bfc541dcd8b261301c",
            "transFrom": "0x",
            "transTo": "92E730F449bbCE3af96dE1Fb9c9c44672DDccb66",
            "blockNumber": 204,
            "blockTimestamp": "2021-12-02 10:12:21",
            "statisticsFlag": 1,
            "createTime": "2021-12-02 10:12:37",
            "modifyTime": "2021-12-02 10:12:37"
        }
    ],
    "totalCount": 204
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

2.2 查询交易回执

2.2.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/transaction/transactionReceipt/{groupId}/{transHash}**
- 请求方式：GET
- 返回格式：JSON

2.2.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/transaction/transactionReceipt/group/
↪0x158c20a61d5ee01f64f2fb2b7f752f834b6a64c1687ac2bfc541dcd8b261301c
```

2.2.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": {
    "version": "0",
    "contractAddress": "",
    "gasUsed": "29999998452",
    "status": 0,
    "blockNumber": "204",
    "output": "0x",
    "transactionHash":
↪"0x158c20a61d5ee01f64f2fb2b7f752f834b6a64c1687ac2bfc541dcd8b261301c",
    "logEntries": [],
    "input":
↪"0x1e26fd33000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001",
    "from": "0x9097678eb34daeebb6e3528ed4f8715cdc5e4c09",
    "to": "92E730F449bbCE3af96dE1Fb9c9c44672DDccb66",
    "transactionProof": null,
    "receiptProof": null,
    "message": null,
    "statusOK": true,
    "hash": "0xf8f0fa250662d5403415600696fbeaba65035cf5d3642cb9787fc958d4d859cc"
  },
  "attachment": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
```

(下页继续)

(续上页)

```
    "data": {}
}
```

2.3 根据交易hash查询交易信息

2.3.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/transaction/transInfo/{groupId}/{transHash}**
- 请求方式：GET
- 返回格式：JSON

2.3.2 参数信息详情

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/transaction/transInfo/group/
↪0x158c20a61d5ee01f64f2fb2b7f752f834b6a64c1687ac2bfc541dcd8b261301c
```

2.3.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": {
    "version": 0,
    "hash": "0x158c20a61d5ee01f64f2fb2b7f752f834b6a64c1687ac2bfc541dcd8b261301c",
    "nonce":
↪"24220955266174890868924993188683552397136341299180306005285154057274733112 3",
    "blockLimit": 703,
    "to": "92E730F449bbCE3af96dE1Fb9c9c44672DDccb66",
    "from": "0x9097678eb34daeebb6e3528ed4f8715cdc5e4c09",
    "input":
↪"0x1e26fd3300000000000000000000000000000000000000000000000000000000000000000000000000000001",
    "chainID": "chain",
    "groupID": "group",
    "signature":
↪"0x9a43124ac99a39783b1765d0c3ab96de5e7766db44ce06119fde0a5f7357e05628cb9098c220ec7323fcf5bf84dda
↪",
    "importTime": 0,
    "transactionProof": null
  },
  "attachment": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

13.3.3 3 帐号管理模块

3.1 新增帐号

3.1.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/account/accountInfo**
- 请求方式：post
- 请求头：Content-type: application/json
- 返回格式：JSON

3.1.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/account/accountInfo
```

```
{
    "account": "testAccount",
    "accountPwd": "3f21a8490cef2bfb60a9702e9d2ddb7a805c9bd1a263557dfd51a7d0e9dfa93e
↪",
    "roleId": 100001,
    "email": "string"
}
```

3.1.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "account": "testAccount",
        "roleId": 100001,
        "roleName": "visitor",
        "roleNameZh": "访客",
        "loginFailTime": 0,
        "accountStatus": 1,
        "description": null,
```

(下页继续)

(续上页)

```
        "createTime": "2019-03-04 15:11:44",
        "modifyTime": "2019-03-04 15:11:44"
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

3.2 修改帐号

3.2.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/account/accountInfo**
- 请求方式：PUT
- 请求头：Content-type: application/json
- 返回格式：JSON

3.2.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/account/accountInfo
```

```
{
    "account": "testAccount",
    "accountPwd": "82ca84cf0d2ae423c09a214cee2bd5a7ac65c230c07d1859b9c43b30c3a9fc80
↪",
    "roleId": 100001,
    "email": "string"
}
```

3.2.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "account": "testAccount",
        "roleId": 100001,
```

(下页继续)

(续上页)

```
        "roleName": "visitor",
        "roleNameZh": "访客",
        "loginFailTime": 0,
        "accountStatus": 1,
        "description": null,
        "createTime": "2019-03-04 15:11:44",
        "modifyTime": "2019-03-04 15:11:44"
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

3.3 删除帐号

3.3.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/account/{account}**
- 请求方式：DELETE
- 返回格式：JSON

3.3.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/account/testAccount
```

3.3.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "data": {},
    "message": "Success"
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
```

(下页继续)

(续上页)

```
    "data": {}
}
```

3.4 查询帐号列表

3.4.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/account/accountList/{pageNumber}/{pageSize}?account={account}**
- 请求方式：GET
- 返回格式：JSON

3.4.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/account/accountList/1/10?account=
```

3.4.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "account": "testAccount",
            "roleId": 100001,
            "roleName": "visitor",
            "roleNameZh": "访客",
            "loginFailTime": 0,
            "accountStatus": 1,
            "description": null,
            "createTime": "2019-03-04 15:11:44",
            "modifyTime": "2019-03-04 15:18:47"
        },
        {
            "account": "admin",
            "roleId": 100000,
            "roleName": "admin",
            "roleNameZh": "管理员",
            "loginFailTime": 0,
            "accountStatus": 2,
            "description": null,
            "createTime": "2019-02-14 17:33:50",
            "modifyTime": "2019-02-14 17:45:53"
        }
```

(下页继续)

(续上页)

```
    ],
    "totalCount": 2
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

3.5 更新当前密码

3.5.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/account/passwordUpdate**
- 请求方式：put
- 请求头：Content-type: application/json
- 返回格式：JSON

3.5.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/account/passwordUpdate
```

```
{
    "oldAccountPwd":
↪"dfdfgdg490cef2bfb60a9702erd2ddb7a805c9bd1arrrewefd51a7d0etttfa93e ",
    "newAccountPwd":
↪"3f21a8490cef2bfb60a9702e9d2ddb7a805c9bd1a263557dfd51a7d0e9dfa93e"
}
```

3.5.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success"
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

3.6 获取登录验证码

3.6.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/account/pictureCheckCode**
- 请求方式：get
- 请求头：Content-type: application/json
- 返回格式：JSON

3.6.2 请求参数

1）入参表 无

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/account/pictureCheckCode
```

3.6.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "base64Image":
↪"iVBORw0KGgoAAAANSUhEUgAAAJsAAAA8CAIAAAD+Gl+NAAAC3ElEQVR42u3cO04DMRAG4DkER6CFhoNwCgoq6DgQh0DiVJ
↪GnGb8g9PN7wbXSRReUtQpEIYoCURSIokAUBaIQTV2+P798r3mfnJbB69nXvmAnjdHm9+8ZgruNjI31mUQrvWPpPpeuH+43n
↪ff3F+mVENJHVDrMeEUGB1W0mFGgLrhnVI9OL1QdZ58wzTnVIMcA3X3FXTQP6tjJ7dFuBjnm29JgXlGXbazoubcs63K23JgS
↪MQXXpKd1IWS6XQ42b7ZOoH/lIZfp9ff7YXCuJHvX709sj87Jb1kQ5Ichp/
↪XpPySmlra9hvCJAl3vr77qg8kO/
↪h2idU4eaRJTTCXzRo9QtSuOkBuOjckQVriWqC7MIldnX6jD1OfF25NwNUP679X50n+4qljH2rMBJvNYY5QefDrWspZ4xRSx
↪AlRL0OJ1OgShcwyvVotKjjLGmD2vSOXptGc3J63iSaBFUUx+oDmVSilf4/
↪nWieMHURLRWsomrUnhvcOVEdRW8tyIIkuj9OtKmVUDSC01NUh9ozr2ZDdU+5jRPvIFHmEsUy3eVMeqP37jtwVk7uyCgkrcU
↪2F4kCjnCLZxPqqI1Hot/mmMPVKDNgXdRflnMso9I07Aqc/
↪abl+PWMOsispvmaRZVL3ru3vmF7EkdT+NWUq0kr5FO4Vl3U31iB0GL865UAV/Czj8X/
↪tOUoxdhN9hyChqqQ7RpTghClEUiKJAFKIQBSpEF0aFKETX7cRsW1q6JyEoHh0qZABWfDqdnDPi5rGPSqflHLK06OBKiM7O1e
↪d/aH4oBfftm+5EiAAAAAElFTkSuQmCC",
        "token": "eyJhbGciOiJIUzI1NiJ9.
↪eyJqdGkiOiIzOGM0NjlhNC1kMTg3LTQyZDQtYWM1YS02OWU0OWM5MjMxNTkiLCJpYXQiOjE1NjAyNDY3MzksInN1YiI6ImU
↪FJYRZJSAhFjvO_P4AjMO6bnoOZJiu-AOSdO9ikb-30M"
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

3.7 登录接口

3.7.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/account/login?checkCode={checkCode}**
- 请求方式：get
- 请求头：Content-type: application/json;token:{token}
- 返回格式：JSON

3.7.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/account/login?checkCode=aege
```

```
{
    "account": "admin",
    "accountPwd": "Abcd1234"
}
```

1.1.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "data": {
        "accountStatus": 2,
        "roleName": "admin",
        "account": "admin"
    },
    "message": "success"
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

13.3.4 4 区块管理模块

4.1 查询区块列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/block/blockList/{groupId}/{pageNumber}/{pageSize}}?pkHash={pkHash}&blockNumber={blockNumber**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/block/blockList/group/1/10?pkHash=
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "pkHash":
↪"0xa6a8e4154fc9c5dbd19922fcb7a77e4018f7004c5681d9d5faf1e3f5723c2053",
            "blockNumber": 206,
            "blockTimestamp": "2021-12-02 14:43:51",
            "transCount": 1,
            "sealerIndex": 0,
            "sealer":
↪"2d5053131f5c0a906fec99380ae797b505e697e1ec0e1f2b59a85ae7f28cb3313986e4e627090d8fdff0c545b82e47
↪",
            "createTime": "2021-12-02 14:44:16",
            "modifyTime": "2021-12-02 14:44:16"
        },
        ......
    ],
    "totalCount": 207
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

4.2 根据块高查询区块信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/block/blockByNumber/{groupId}/{blockNumber}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/block/blockByNumber/group/1
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": {
    "number": 1,
    "version": 0,
    "hash": "0x60d1eacd2a279c343c74b673ad30fc6c6a528fb8434b90cc276cc239bf998835",
    "logsBloom": null,
    "receiptsRoot":
↪"0x96e73cf059095f36cd0fed8bd3ce15550de892eae269598b7ba9e1721c8d05a9",
    "stateRoot":
↪"0xd212274a9fdaf3c59a9733f0607b510456d5f395ae0a19c391cd277ff54c2872",
    "sealer": 0,
    "sealerList": [

↪"0x2d5053131f5c0a906fec99380ae797b505e697e1ec0e1f2b59a85ae7f28cb3313986e4e627090d8fdff0c545b82e
↪",

↪"0x44f760aa2f9058d2dd85a578197d2bfdeac7ef8db1dec39386854e1d66bce4077ce3cae1e4779658b3ebcf929c12
↪"
    ],
    "extraData": "0x",
    "gasUsed": "6677",
    "timestamp": 1637909172326,
    "parentInfo": [
      {
        "blockNumber": 0,
        "blockHash":
↪"0xe093202d1b18d96ac52700c37af54cd370f06d5062d31232baa3d9ac324d6a57"
      }
    ],
    "signatureList": [
```

(下页继续)

(续上页)


```
      {
        "signature":
↪"0x1695e0ae6ca37a5fd3d6dfd5c4f18223ec509caa89a485f58cc2675b9147538d1bd381ff5e1b07169639ad1865d1
↪",
        "sealerIndex": 0
      },
      {
        "signature":
↪"0x8d05b9be2cd01006e249b96c10261e560bf4994144e3808c64186b67001f70675a7d00244d4bc78b6c2369206d96
↪",
        "sealerIndex": 1
      }
    ],
    "consensusWeights": [
      1,
      1
    ],
    "transactions": [
      {
        "version": 0,
        "hash": "0x29fc394aab9f6b850cf670e4ffcfa07f749d65a96fdaf4a1734d08078716df2a
↪",
        "nonce":
↪"628963865034161222069339779084488493077824905060891430374114608099078683365",
        "blockLimit": 500,
        "to": "",
        "from": "0x",
        "input":
↪"0x60806040523480156100105760008fd5b506101488061002060000396000f300608060405260043610610c5760
↪",
        "chainID": "chain",
        "groupID": "group",
        "signature":
↪"0x01ca79d793e424fa2cd54ef8b8967cfdae72b7560d9ff9236bff5d833946571573b87172963acda9a88f119d932d
↪",
        "importTime": 0,
        "transactionProof": null
      }
    ],
    "txsRoot": "0xc329ee0d01e17507035ef1e507a77484f0191744f9d28ad252ed33c9147cfa05"
  },
  "attachment": null
}
```


- 失败：


```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```


13.3.5 5 合约管理模块

5.1 查询合约列表

传输协议规范

- 网络传输协议：使用HTTP协议

- 请求地址：**/contract/contractList/**
- 请求方式：POST
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/contractList
```

```
{
  "account": "string",
  "contractAddress": "string",
  "contractName": "string",
  "contractPath": "string",
  "contractStatus": 0,
  "groupId": "string",
  "pageNumber": 0,
  "pageSize": 0
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "contractId": 200002,
            "contractPath": "hellos",
            "contractVersion": null,
            "contractName": "hellos",
            "contractStatus": 2,
            "groupId": "group",
            "contractType": 0,
            "contractSource": "cHJhZ21hIHNvbGlkaXgICAJbmFtZSA9IG47CiAgICB9Cn0=",
            "contractAbi": "[\"payable\":false,\"stateMutability\":\"nonpayable\",\
↪"type\":\"constructor\"}]",
            "contractBin": "6080604052348015610010576000800004d4c",
            "bytecodeBin":
↪"60806040526004361061004c576000398de7e4ddf5fdc9ccbcfd44565fed695cd960b0029",
            "deployTime": "2019-06-11 18:11:33",
            "description": null,
            "account": "admin",
            "createTime": "2019-06-05 16:40:40",
            "modifyTime": "2019-06-11 18:11:33"
        }
    ],
    "totalCount": 1
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

5.2 查询合约信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/contract/{contractId}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/200001
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "contractId": 200002,
        "contractPath": "hellos",
        "contractVersion": null,
        "contractName": "hellos",
        "contractStatus": 2,
        "groupId": "group",
        "contractType": 0,
        "contractSource": "cHJhZ21hIHNvbGlkaXgICAJbmFtZSA9IG47CiAgICB9Cn0=",
        "contractAbi": "[\"payable\":false,\"stateMutability\":\"nonpayable\",\
↪"type\":\"constructor\"}]",
        "bytecodeBin":
↪"608060405260043610610 4c576000398de7e4ddf5fdc9ccbcfd44565fed695cd960b0029",
        "contractBin": "6080604052348015610010576000800 04d4c",
        "deployTime": "2019-06-11 18:11:33",
        "description": null,
        "account": "admin",
        "createTime": "2019-06-05 16:40:40",
        "modifyTime": "2019-06-11 18:11:33"
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

5.3 部署合约

3.0.2及以后版本：

构造方法参数（funcParam）为String数组，每个参数都使用String字符串表示，多个参数以逗号分隔（参数为数组时同理），示例：

```
constructor(string s) -> ["aa,bb\"cc"]          // 双引号要转义
constructor(uint n,bool b) -> ["1","true"]
constructor(bytes b,address[] a) -> ["0x1a","[\
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE\",\
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9\"]"]
```

3.0.2以前的版本：

构造方法参数（funcParam）为JSON数组，多个参数以逗号分隔（参数为数组时同理），示例：

```
constructor(string s) -> ["aa,bb\"cc"]          // 双引号要转义
constructor(uint n,bool b) -> [1,true]
constructor(bytes b,address[] a) -> ["0x1a",[
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE",
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9"]]
```

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/contract/deploy**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/deploy
```

```
{
        "groupId": "group",
        "contractBin":
↪"6080604052600436106100c4576000357c0100000000000000000000000000000000000000000000000000000000900
↪",
        "bytecodeBin":
↪"6080604052348015610010576000080fd5b506040805190810160405280600d81526020017f48656c6c6f2c20576f72
↪",
```

(下页继续)

(续上页)


```
        "contractAbi": "[{\"constant\":true,\"inputs\":[],\"name\":\"get\",\
↪"outputs\":[{\"name\":\"\",\"type\":\"string\"}],\"payable\":false,\
↪"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\":false,\"inputs\
↪":[{\"name\":\"n\",\"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\
↪"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\
↪"inputs\":[],\"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\
↪"constructor\"}]",
        "contractSource":
↪"cHJhZ21hIHNvbGlkaXR5ID49MC40LjI0IDwwLjYuMTE7Cgpjb250cmFjdCBIZWxsb1dvcmxkIHsKICAgIHN0cmluZyBuYW
↪",
        "user": "0xdccae56cef725605d0fa1e00fd553074a74091c5",
        "contractName": "HelloWorld",
        "contractId": 200306,
        "contractPath": "/",
        "account": "admin"
}
```


返回参数

1）出参表

2）出参示例

- 成功：


```
{
    "code": 0,
    "message": "success",
    "data": {
        "contractId": 200008,
        "contractPath": "Hi",
        "contractVersion": null,
        "contractName": "HeHe",
        "contractStatus": 2,
        "groupId": "group",
        "contractType": null,
        "contractSource": "cHJhZ21hIHNvbGlkaXR5IF4wLjQuM0=",
        "contractAbi": "[]",
        "bytecodeBin":
↪"60806040526004361061004c576000357c010274c87bff322ea2269b80029",
        "contractBin": "6080604052348015610010057629",
        "contractAddress": "0xa2ea2280b3a08a3ae2e1785dff09561e13915fb2",
        "deployTime": "2019-06-11 18:58:33",
        "description": null,
        "account": "admin",
        "createTime": null,
        "modifyTime": null,
        "deployAddress": "0xb783d7c2cd45d8c678093d5f6f1d61855e260e67",
                "deployUserName": "frank"
    }
}
```


- 失败：


```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

5.4 发送交易

3.0.2及以后版本：

方法参数（funcParam）为String数组，每个参数都使用String字符串表示，多个参数以逗号分隔（参数为数组时同理），示例：

```
function set(string s) -> ["aa,bb\"cc"]         // 双引号要转义
function set(uint n,bool b) -> ["1","true"]
function set(bytes b,address[] a) -> ["0x1a","[\
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE\",\
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9\"]"]
```

3.0.2以前的版本：

方法入参（funcParam）为JSON数组，多个参数以逗号分隔（参数为数组时同理），示例：

```
function set(string s) -> ["aa,bb\"cc"]         // 双引号要转义
function set(uint n,bool b) -> [1,true]
function set(bytes b,address[] a) -> ["0x1a",[
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE",
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9"]]
```

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/contract/transaction**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/transaction
```

```
{
    "groupId": "group",
    "user":"0xdccae56cef725605d0fa1e00fd553074a74091c5",
    "contractName":"HelloWorld",
    "funcName":"set",
    "funcParam":["gwes"],
    "contractAbi": [{"constant":true,"inputs":[],"name":"get","outputs":[{"name":"
↪","type":"string"}],"payable":false,"stateMutability":"view","type":"function"},{
↪"constant":false,"inputs":[{"name":"n","type":"string"}],"name":"set","outputs
↪":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"inputs
↪":[],"payable":false,"stateMutability":"nonpayable","type":"constructor"}],
    "contractId":200306,
    "contractAddress":"0x4d1cbcc47b2558d818b9672df67f22f9a9645c87"
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
        "code": 0,
        "message": "success",
        "data": [{
                "value": 0,
                "bitSize": 16,
                "typeAsString": "int16"
        }],
        "attachment": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

5.5 根据包含bytecodeBin的字符串查询合约

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/contract/findByPartOfBytecodeBin**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/findByPartOfBytecodeBin
```

```
{
    "groupId": "group",
    "partOfBytecodeBin": "abc123455dev"
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "contractId": 200002,
        "contractName": "Ok",
        "groupId": 2,
        "chainIndex": null,
        "contractType": 0,
        "contractSource": "cHJhZ21hIDQoNCg0KfQ==",
        "contractAbi": "[]",
        "contractBin": "60606040526000357c01000000000029",
        "bytecodeBin": "123455",
        "contractAddress": "0x19146d3a2f138aacb97ac52dd45dd7ba7cb3e04a",
        "deployTime": null,
        "contractVersion": "v6.0",
        "description": null,
        "account": "admin",
        "createTime": "2019-04-15 21:14:40",
        "modifyTime": "2019-04-15 21:14:40"
    }
}
```


- 失败：


```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```


5.6. 保存合约接口

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/contract/save**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例


```
{
    "groupId": "group",
    "contractName": "HeHe",
    "contractPath": "/",
    "contractSource": "cHJhZ21hIHNvbGlkaXR5IF4wLjQuMjsn0=",
    "contractAbi": "[]"
    "contractBin": "6080604052600436106100c576000357c0100000000000000000000000000029
↪",
    "bytecodeBin": "60806040523480156100010572269b80029",
```


(下页继续)

(续上页)

```
    "contractId": 1,
    "account": "admin",
    "contractAddress": "string"
}
```

返回参数

1）出参表

5.7 获取Abi信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/abi/{abiId}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/abi/1
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": {
    "abiId": 1,
    "groupId": 1,
    "contractName": "TTT",
    "contractAddress": "0x3214227e87bccca63893317febadd0b51ade735e",
    "contractAbi": "[{\"constant\":false,\"inputs\":[{\"name\":\"n\",\"type\":\
↪"string\"}],\"name\":\"set\",\"outputs\":[],\"payable\":false,\"stateMutability\
↪":\"nonpayable\",\"type\":\"function\"},{\"constant\":true,\"inputs\":[],\"name\
↪":\"get\",\"outputs\":[{\"name\":\"\",\"type\":\"string\"}],\"payable\":false,\
↪"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\":false,\"inputs\
↪":[{\"name\":\"n\",\"type\":\"string\"}],\"name\":\"setSender\",\"outputs\":[],\
↪"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\
↪"anonymous\":false,\"inputs\":[{\"indexed\":false,\"name\":\"name\",\"type\":\
↪"string\"}],\"name\":\"SetName\",\"type\":\"event\"},{\"anonymous\":false,\
↪"inputs\":[{\"indexed\":false,\"name\":\"\",\"type\":\"uint256[2]\"}],\"name\":\
↪"EventList\",\"type\":\"event\"},{\"anonymous\":false,\"inputs\":[{\"indexed\
↪":false,\"name\":\"sender\",\"type\":\"address\"}],\"name\":\"SetSender\",\"type\
↪":\"event\"}]",
```

(下页继续)

(续上页)

```
    "contractBin": "6080604052600436106100575760003357...",
    "account": "admin",
    "createTime": "2020-05-18 10:59:02",
    "modifyTime": "2020-05-18 10:59:02"
  }
}
```

5.8 获取Abi信息分页列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/abi/list/{groupId}/{pageNumber}/{pageSize}?account={account}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/abi/list/group/1/5?account=
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "abiId": 1,
      "groupId": "group",
      "account": "admin",
      "contractName": "TTT",
      "contractAddress": "0x3214227e87bccca63893317febadd0b51ade735e",
      "contractAbi": "[{\"constant\":false,\"inputs\":[{\"name\":\"n\",\"type\":\
↪"string\"}],\"name\":\"set\",\"outputs\":[],\"payable\":false,\"stateMutability\
↪":\"nonpayable\",\"type\":\"function\"},{\"constant\":true,\"inputs\":[],\"name\
↪":\"get\",\"outputs\":[{\"name\":\"\",\"type\":\"string\"}],\"payable\":false,\
↪"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\":false,\"inputs\
↪":[{\"name\":\"n\",\"type\":\"string\"}],\"name\":\"setSender\",\"outputs\":[],\
↪"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\
↪"anonymous\":false,\"inputs\":[{\"indexed\":false,\"name\":\"name\",\"type\":\
↪"string\"}],\"name\":\"SetName\",\"type\":\"event\"},{\"anonymous\":false,\
↪"inputs\":[{\"indexed\":false,\"name\":\"\",\"type\":\"uint256[2]\"}],\"name\":\
↪"EventList\",\"type\":\"event\"},{\"anonymous\":false,\"inputs\":[{\"indexed\
↪":false,\"name\":\"sender\",\"type\":\"address\"}],\"name\":\"SetSender\",\"type\
↪":\"event\"}]",
```

(下页继续)

(续上页)

```
      "contractBin":
↪"6080604052600436106100575760003
↪..29",
      "createTime": "2020-05-18 10:59:02",
      "modifyTime": "2020-05-18 10:59:02"
    }
  ],
  "totalCount": 1
}
```

5.9. 导入已部署合约的abi

将其他平台已部署的合约导入到本平台进行管理

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/abi**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
{
    "groupId": "group",
    "abiId": 1,
    "account": "admin",
    "contractAddress": "0x3214227e87bccca63893317febadd0b51ade735e",
    "contractName": "HelloWorld",
    "contractAbi": [{
        "constant": false,
        "inputs": [{
            "name": "n",
            "type": "string"
        }],
        "name": "set",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    }, {
        "constant": true,
        "inputs": [],
        "name": "get",
        "outputs": [{
            "name": "",
            "type": "string"
        }],
        "payable": false,
```

(下页继续)

(续上页)

```
        "stateMutability": "view",
        "type": "function"
    }, {
        "anonymous": false,
        "inputs": [{
            "indexed": false,
            "name": "name",
            "type": "string"
        }],
        "name": "SetName",
        "type": "event"
    }]
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success"
}
```

5.10. 修改已部署合约的abi

将其他平台已部署的合约导入到本平台进行管理

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/abi**
- 请求方式：PUT
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
{
    "abiId": 1,
    "groupId": "group",
    "contractAddress": "0x3214227e87bccca63893317febadd0b51ade735e",
    "contractName": "HelloWorld",
```

(下页继续)

(续上页)

```
    "contractAbi": [{"constant":false,"inputs":[{"name":"n","type":"string"}],"name
↪":"set","outputs":[],"payable":false,"stateMutability":"nonpayable","type":
↪"function"},{"constant":true,"inputs":[],"name":"get","outputs":[{"name":"","type
↪":"string"}],"payable":false,"stateMutability":"view","type":"function"},{
↪"anonymous":false,"inputs":[{"indexed":false,"name":"name","type":"string"}],
↪"name":"SetName","type":"event"}]
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success"
}
```

5.11. 获取全量合约列表（不包含abi/bin）

接口描述

根据群组编号和合约状态获取全量合约

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/contract/contractList/all/light?groupId={groupId}&contractStatus={contractStatus}**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/contractList/all/light?
↪groupId=group&contractStatus=2
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "id": 2,
            "contractPath": "/",
            "contractName": "HeHe",
            "contractStatus": 1,
            "groupId":"group",
            "contractSource": "cHJhZ21hIHNvbGlkaXR5IICB9Cn0=",
            "contractAddress": null,
            "deployTime": null,
            "description": null,
            "createTime": "2019-06-10 16:42:50",
            "modifyTime": "2019-06-10 16:42:52"
        }
    ],
    "totalCount": 1
}
```


5.12. 保存合约路径

接口描述

保存合约路径

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/contract/contractPath**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/contractPath
```

响应参数

1）参数表

2）数据格式


```
{
    "code": 0,
    "message": "success",
```


(下页继续)

(续上页)

```
    "data": 1
}
```

5.13. 删除合约路径并批量删除合约

接口描述

删除合约路径并批量删除合约

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/contract/batch/path**
- 请求方式：DELETE
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/batch/path
```

```
{
    "groupId": "group",
    "contractPath": test
}
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "message": "success"
}
```

5.14 获取合约与导入ABI列表（分页）

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/abi/list/all/{groupId}/{pageNumber}/{pageSize}?account={account}**
- 请求方式：GET

- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/abi/list/all/group/1/5
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "contractId": 200032,
      "contractPath": "/",
      "contractVersion": null,
      "contractName": "AAA",
      "account": "admin",
      "contractStatus": 2,
      "groupId": "group",
      "contractType": 0,
      "contractSource": null,
      "contractAbi": "[{\"constant\":false,\"inputs\":[{\"name\":\"n\",\"type\":\
↪"string\"}],\"name\":\"set\",\"outputs\":[],\"payable\":false,\"stateMutability\
↪":\"nonpayable\",\"type\":\"function\"},{\"constant\":true,\"inputs\":[],\"name\
↪":\"get\",\"outputs\":[{\"name\":\"\",\"type\":\"string\"}],\"payable\":false,\
↪"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":[],\"payable\
↪":false,\"stateMutability\":\"nonpayable\",\"type\":\"constructor\"},{\
↪"anonymous\":false,\"inputs\":[{\"indexed\":false,\"name\":\"\",\"type\":\
↪"int256\"}],\"name\":\"test\",\"type\":\"event\"}]",
      "contractBin":
↪"60806040526004361061004c576000357c0100000000000000000000000000000000000000000000000000000000900
↪",
      "bytecodeBin":
↪"60806040523480156100105760080fd5b5061030f80610020600039600f3006080604052600436106100c576000
↪",
      "contractAddress": "0xab9F30F9827e1C16Bf62d557c3626AeAd9E85502",
      "deployTime": "2021-12-06 11:31:34",
      "description": null,
      "createTime": "2021-12-06 11:31:34",
      "modifyTime": "2021-12-06 11:31:34",
      "deployAddress": "0x9097678eb34daeebb6e3528ed4f8715cdc5e4c09",
      "deployUserName": "new_test",
      "abiId": 26
    },
    ...
  ],
  "totalCount": 2
}
```

5.15. 导入合约仓库到IDE

接口描述

保存多个合约

传输协议规范

- 网络传输协议：HTTP协议
- 请求地址：**/contract/copy**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/copy
```

```
{
        "contractItems": [{
                "contractName": "Evidence",
                "contractSource":
↪"cHJhZ21hIHNvbGlkaXR5IF4wLjQuNDsKY29udHJhY3QgRXZpZGVuY2VTaWduZXJzRGF0YUFCSXsgZnVuY3Rpb24gdmVyaW
↪"
        }, {
                "contractName": "EvidenceSignersData",
                "contractSource":
↪"cHJhZ21hIHNvbGlkaXR5IF4wLjQuNDsKaW1wb3J0ICJFdmlkZW5jZS5zb2wiOwoKY29udHJhY3QgRXZpZGVuY2VTaWduZX
↪"
        }],
        "contractPath": "Evidence1",
        "groupId": "1",
        "account": "mars"
}
```

响应参数

1）参数表

2）数据格式

```
{
    "code": 0,
    "message": "success",
    "data": 2
}
```

5.16. 获取合约仓库列表

接口描述

返回合约仓库信息列表

接口URL

http://localhost:5001/WeBASE-Node-Manager/warehouse/list

调用方法

HTTP GET

请求参数

1）参数表

无

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/warehouse/list
```

响应参数

1）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "id": 1,
      "warehouseName": "工具箱",
      "warehouseNameEn": "Toolbox",
      "type": "1",
      "warehouseIcon": "toolboxId",
      "description": "工具箱中有常用的工具合约",
      "warehouseDetail": "工具箱中有常用的工具合约",
      "descriptionEn": "Toolbox Contract suite",
      "warehouseDetailEn": "Toolbox Contract suite",
      "createTime": "2021-01-20 18:02:10",
      "modifyTime": "2021-01-20 18:02:10"
    },
    {
      "id": 2,
      "warehouseName": "存证应用",
      "warehouseNameEn": "Evidence",
      "type": "2",
      "warehouseIcon": "evidenceId",
      "description": "一套区块链存证合约",
      "warehouseDetail": "一套区块链存证合约",
      "descriptionEn": "Evidence Contract suite",
      "warehouseDetailEn": "Evidence Contract suite",
      "createTime": "2021-01-20 18:02:10",
```

(下页继续)

(续上页)

```
      "modifyTime": "2021-01-20 18:02:10"
    },
    {
      "id": 3,
      "warehouseName": "积分应用",
      "warehouseNameEn": "Points",
      "type": "3",
      "warehouseIcon": "pointsId",
      "description": "一套积分合约",
      "warehouseDetail": "一套积分合约",
      "descriptionEn": "Points Contract suite",
      "warehouseDetailEn": "Points Contract suite",
      "createTime": "2021-01-20 18:02:10",
      "modifyTime": "2021-01-20 18:02:10"
    }
  ]
}
```

5.17. 根据仓库编号获取仓库信息

接口描述

返回合约仓库信息

接口URL

http://localhost:5001/WeBASE-Node-Manager/warehouse?warehouseId={warehouseId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/warehouse?warehouseId=1
```

响应参数

1）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": {
    "id": 1,
    "warehouseName": "工具箱",
    "warehouseNameEn": "Toolbox",
    "type": "1",
    "warehouseIcon": "toolboxId",
```

(下页继续)

(续上页)

```
    "description": "工具箱中有常用的工具合约",
    "warehouseDetail": "工具箱中有常用的工具合约",
    "descriptionEn": "Toolbox Contract suite",
    "warehouseDetailEn": "Toolbox Contract suite",
    "createTime": "2021-01-20 18:02:10",
    "modifyTime": "2021-01-20 18:02:10"
  }
}
```

5.18. 根据仓库编号获取合约文件夹信息

接口描述

返回合约文件夹信息

接口URL

http://localhost:5001/WeBASE-Node-Manager/warehouse/folder/list?warehouseId={warehouseId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/warehouse/folder/list?warehouseId=1
```

响应参数

1）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "id": 2,
      "folderName": "Evidence",
      "folderDesc": "Evidence",
      "folderDetail": "Evidence",
      "folderDescEn": "Evidence",
      "folderDetailEn": "Evidence",
      "createTime": "2021-01-20 18:02:10",
      "modifyTime": "2021-01-20 18:02:10"
    }
  ]
}
```

5.19. 根据合约文件夹编号获取合约文件夹信息

接口描述

返回合约文件夹信息

接口URL

http://localhost:5001/WeBASE-Node-Manager/warehouse/folder?folderId={folderId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/warehouse/folder?folderId=2
```

响应参数

1）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": {
    "id": 2,
    "folderName": "Evidence",
    "folderDesc": "Evidence",
    "folderDetail": "Evidence",
    "folderDescEn": "Evidence",
    "folderDetailEn": "Evidence",
    "createTime": "2021-01-20 18:02:10",
    "modifyTime": "2021-01-20 18:02:10"
  }
}
```

5.20. 根据文件夹编号获取合约列表

接口描述

返回合约信息列表

接口URL

http://localhost:5001/WeBASE-Node-Manager/warehouse/item/list?folderId={folderId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/warehouse/item/list?folderId=2
```

响应参数

1）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "contractId": 4,
      "contractFolderId": 2,
      "contractName": "Evidence",
      "contractDesc": "Evidence",
      "contractSrc":
↪"cHJhZ21hIHNvbGlkaXR5IF4wLjQuNDsKY29udHJhY3QgRXZpZGVuY2VTaWduZXJzRGF0YUFCSXsgZnVuY3Rpb24gdmVyaW
↪",
      "contractDescEn": "Evidence",
      "createTime": "2021-01-20 18:02:10",
      "modifyTime": "2021-01-20 18:02:10"
    },
    {
      "contractId": 5,
      "contractFolderId": 2,
      "contractName": "EvidenceSignersData",
      "contractDesc": "EvidenceSignersData",
      "contractSrc":
↪"cHJhZ21hIHNvbGlkaXR5IF4wLjQuNDsKaW1wb3J0ICJFdmlkZW5jZS5zb2wiOwoKY29udHJhY3QgRXZpZGVuY2VTaWduZX
↪",
      "contractDescEn": "EvidenceSignersData",
      "createTime": "2021-01-20 18:02:10",
      "modifyTime": "2021-01-20 18:02:10"
    }
  ]
}
```

5.21. 根据合约编号获取合约信息

接口描述

返回合约信息

接口URL

http://localhost:5001/WeBASE-Node-Manager/warehouse/item?contractId={contractId}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/warehouse/item?contractId=2
```

响应参数

1）数据格式

```
{
  "code": 0,
  "message": "success",
  "data": {
    "contractId": 2,
    "contractFolderId": 1,
    "contractName": "LibSafeMathForUint256Utils",
    "contractDesc": "LibSafeMathForUint256Utils",
    "contractSrc":
↪"LyoKICogQ29weXJpZ2h0IDIwMTQtMjAxOSB0aGUgb3JpZ2luYWwgYXV0aG9yIG9yIGF1dGhvcnMuCiAqCiAqIExpY2Vuc2V
↪",
    "contractDescEn": "LibSafeMathForUint256Utils",
    "createTime": "2021-01-20 18:02:10",
    "modifyTime": "2021-01-20 18:02:10"
  }
}
```

5.22. 查询合约管理者列表

根据群组ID和合约地址，返回在WeBASE拥有私钥的合约部署者或链委员（若链委员非空）私钥用户

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/contract/listManager/{groupId}/{contractAddress}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/listManager/group/
↪0xab9F30F9827e1C16Bf62d557c3626AeAd9E85502
```

返回参数

1）出参表

2）出参示例

- 成功：


```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "userId": 700003,
      "userName": "new_test",
      "account": "test",
      "groupId": "group",
      "publicKey":
↪"04b27022e80712f4def30d008a390cbe3503100ef54236bea3d4ca9a7fe95c8aafc2a401dd344605fa8d6b52533b28
↪",
      "privateKey": null,
      "userStatus": 1,
      "chainIndex": null,
      "userType": 1,
      "address": "0x9097678eb34daeebb6e3528ed4f8715cdc5e4c09",
      "signUserId": "05281cdc3c894a2da852af6b2ad3d919",
      "appId": "group",
      "hasPk": 1,
      "description": "",
      "createTime": "2021-11-30 16:18:03",
      "modifyTime": "2021-11-30 16:18:03"
    }
  ],
  "attachment": null
}
```


- 失败：


```
{
    "code": 202542,
    "message": "No private key of contract manager address in webase"
}
```


5.23. 检测是否已配置Liquid环境

通过cargo命令和liquid命令，检测WeBASE-Front所在主机是否已配置Liquid环境

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/contract/liquid/check/{frontId}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/liquid/check/10001
```

响应参数

1）参数表

2）数据格式

```
{
  "code": 0,
  "message": "success"
}
```

5.24. 编译liquid合约

传入合约源码、编译liquid合约，并返回编译得到的abi和bin。

由于liquid合约类似于rust编译，耗时比solidity更长（3分钟左右），因此接口返回状态为“编译中”时，后台将异步执行编译任务，通过轮询/contract/liquid/compile/check接口可以获取最新的编译结果

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/contract/liquid/compile**
- 请求方式：POST
- 返回格式：JSON

请求参数

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/liquid/compile
```

```
{
    "groupId": "group0",
        "contractName": "LiquidHelloWorld",
        "contractPath": "/",
        "contractSource":
↪"IyFbY2ZnX2F0dHIobm90KGZlYXR1cmUgPSAic3RkIiksIG5vX3N0ZCldCgp1c2UgbGlxdWlkOjpzdG9yYWdlOwp1c2UgbG
↪",
        "contractAbi": "[{\"inputs\":[],\"type\":\"constructor\"},{\"constant\
↪":true,\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"internalType\":\"string\",\
↪"type\":\"string\"}],\"type\":\"function\"},{\"conflictFields\":[{\"kind\":0,\
↪"path\":[],\"read_only\":false,\"slot\":0}],\"constant\":false,\"inputs\":[{\
↪"internalType\":\"string\",\"name\":\"name\",\"type\":\"string\"}],\"name\":\
↪"set\",\"outputs\":[],\"type\":\"function\"}]",
    "isWasm": true,
    "contractId": 1,
    "account": "admin",
    "frontId": 10001
}
```

响应参数

1）参数表

2）数据格式

状态为编译中时： （编译中时，后台将异步执行编译任务，通过轮询/contract/liquid/compile/check接口可以获取最新的编译结果）

```
{
  "code": 0,
  "message": "success"
  "data": {
    "groupId": "group",
    "contractPath": "",
    "contractName": "Hello",
    "status": "1",
    "bin": null,
    "abi": null,
    "createTime": "2020-12-30 16:32:28",
    "modifyTime": "2020-12-30 16:32:28"
  }
}
```

状态为编译成功时（后台将自动更新合约IDE中的合约内容，包括ABI，BIN等）：

```
{
  "code": 0,
  "message": "success"
  "data": {
    "groupId": "group",
    "contractPath": "",
    "contractName": "Hello",
    "status": "2",
    "bin": "",  //bin过长，此处略
    "abi":"[{\"constant\":true,\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"name\
↪":\"\",\"type\":\"string\",\"type0\":null,\"indexed\":false}],\"type\":\
↪"function\",\"payable\":false,\"stateMutability\":\"view\"},{\"constant\":false,\
↪"inputs\":[{\"name\":\"n\",\"type\":\"string\",\"type0\":null,\"indexed\":false}
↪],\"name\":\"set\",\"outputs\":[],\"type\":\"function\",\"payable\":false,\
↪"stateMutability\":\"nonpayable\"},{\"constant\":false,\"inputs\":[{\"name\":\
↪"name\",\"type\":\"string\",\"type0\":null,\"indexed\":false}],\"name\":\
↪"SetName\",\"outputs\":null,\"type\":\"event\",\"payable\":false,\
↪"stateMutability\":null}]",
    "createTime": "2020-12-30 16:32:28",
    "modifyTime": "2020-12-30 16:32:28"
  }
}
```

5.25. 查询liquid合约编译进度

根据群组ID，合约路径，合约名获取liquid合约的编译状态

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/contract/liquid/compile/check**
- 请求方式：POST

- 返回格式：JSON

请求参数

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/contract/liquid/compile/check
```

```
{
    "groupId": "group0",
        "contractName": "LiquidHelloWorld",
        "contractPath": "/",
        "contractSource":
↪"IyFbY2ZnX2F0dHIobm90KGZlYXR1cmUgPSAic3RkIiksIG5vX3N0ZCldCgp1c2UgbGlxdWlkOjpzdG9yYWdlOwp1c2UgbG
↪",
        "contractAbi": "[{\"inputs\":[],\"type\":\"constructor\"},{\"constant\
↪":true,\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"internalType\":\"string\",\
↪"type\":\"string\"}],\"type\":\"function\"},{\"conflictFields\":[{\"kind\":0,\
↪"path\":[],\"read_only\":false,\"slot\":0}],\"constant\":false,\"inputs\":[{\
↪"internalType\":\"string\",\"name\":\"name\",\"type\":\"string\"}],\"name\":\
↪"set\",\"outputs\":[],\"type\":\"function\"}]",
    "isWasm": true,
    "contractId": 1,
    "account": "admin",
    "contractAddress": "string",
    "frontId": 10001
}
```

响应参数

1）参数表

2）数据格式

状态为编译中时，轮询当前接口直到状态为编译成功、编译失败，status=1：

```
{
  "code": 0,
  "message": "success"
  "data": {
    "groupId": "group",
    "contractPath": "/",
    "contractName": "Hello",
    "status": "1",
    "bin": null,
    "abi": null,
    "createTime": "2020-12-30 16:32:28",
    "modifyTime": "2020-12-30 16:32:28"
  }
}
```

状态为编译成功时，status=2（后台将自动更新合约IDE中的合约内容，包括ABI，BIN等）：

```
{
  "code": 0,
  "message": "success"
  "data": {
    "groupId": "group",
    "contractPath": "/",
```

(下页继续)

(续上页)

```
    "contractName": "Hello",
    "status": "2",
    "bin": "",  //bin过长，此处略
    "abi":"[{\"constant\":true,\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"name\
↪":\"\",\"type\":\"string\",\"type0\":null,\"indexed\":false}],\"type\":\
↪"function\",\"payable\":false,\"stateMutability\":\"view\"},{\"constant\":false,\
↪"inputs\":[{\"name\":\"n\",\"type\":\"string\",\"type0\":null,\"indexed\":false}
↪],\"name\":\"set\",\"outputs\":[],\"type\":\"function\",\"payable\":false,\
↪"stateMutability\":\"nonpayable\"},{\"constant\":false,\"inputs\":[{\"name\":\
↪"name\",\"type\":\"string\",\"type0\":null,\"indexed\":false}],\"name\":\
↪"SetName\",\"outputs\":null,\"type\":\"event\",\"payable\":false,\
↪"stateMutability\":null}]",
    "createTime": "2020-12-30 16:32:28",
    "modifyTime": "2020-12-30 16:32:28"
  }
}
```

13.3.6 6 服务器监控相关

6.1 发送测试邮件

使用当前的邮件服务配置，向指定的邮箱地址发送测试邮件，如果配置错误将发送失败；

注：需要确保配置正确才能使用后续的邮件告警功能；返回成功信息后，需要用户到自己的邮箱查看是否收到邮测试邮件；

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/alert/mail/test/{toMailAddress}**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/alert/mail/test/yourmail@qq.com
```

```
{
    "host": "smtp.qq.com",
    "port": 25,
    "username": "yourmail@qq.com",
    "password": "yourpassword",
    "protocol": "smtp",
    "defaultEncoding": "UTF-8",
    "authentication": 1,
    "starttlsEnable": 1,
    "starttlsRequired": 0,
    "socketFactoryPort": 465,
```

(下页继续)

(续上页)

```
    "socketFactoryClass": "javax.net.ssl.SSLSocketFactory",
    "socketFactoryFallback": 0,
    "timeout": 5000,
    "connectionTimeout": 5000,
    "writeTimeout": 5000
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": null
}
```

- 失败：

```
{
    "code": 202080,
    "message": "Send mail error, please check mail server configuration.",
    "data": "Failed messages: javax.mail.SendFailedException: No recipient␣
↪addresses"
}
```

6.2 获取告警类型配置

获取单个告警配置的内容；告警类型配置是对不同告警类型下的不同内容，包含告警邮件标题ruleName，告警邮件内容alertContent，告警邮件发送时间间隔alertIntervalSeconds，上次告警时间lastAlertTime，目标告警邮箱地址列表userList，是否启用该类型的邮件告警enable，告警等级alertLevel等；

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/alert/{ruleId}**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/alert/1
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "ruleId": 1,
        "ruleName": "节点异常告警",
        "enable": 0,
        "alertType": 1,
        "alertLevel": 1,
        "alertIntervalSeconds": 3600,
        "alertContent": "{nodeId}节点异常，请到\"节点管理\"页面查看具体信息",
        "contentParamList": "[\"{nodeId}\"]",
        "description": null,
        "createTime": "2019-10-29 20:02:30",
        "modifyTime": "2019-10-29 20:02:30",
        "isAllUser": 0,
        "userList": "[\"targetmail@qq.com\"]",
        "lastAlertTime": null
    }
}
```

6.3 获取全部告警类型配置列表

返回所有的告警类型配置

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/alert/list**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/alert/list
```

返回参数

1）出参表

2）出参示例

- 成功：


```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "ruleId": 1,
            "ruleName": "节点异常告警",
            "enable": 0,
            "alertType": 1,
            "alertLevel": 1,
            "alertIntervalSeconds": 3600,
            "alertContent": "{nodeId}节点异常，请到\"节点管理\"页面查看具体信息",
            "contentParamList": "[\"{nodeId}\"]",
            "description": null,
            "createTime": "2019-10-29 20:02:30",
            "modifyTime": "2019-10-29 20:02:30",
            "isAllUser": 0,
            "userList": "[\"targetmail@qq.com\"]",
            "lastAlertTime": null
        },
        {
            "ruleId": 2,
            "ruleName": "审计异常",
            "enable": 0,
            "alertType": 2,
            "alertLevel": 1,
            "alertIntervalSeconds": 3600,
            "alertContent": "审计异常：{auditType}，请到\"交易审计\"页面查看具体信息",
            "contentParamList": "[\"{auditType}\"]",
            "description": null,
            "createTime": "2019-10-29 20:02:30",
            "modifyTime": "2019-10-29 20:02:30",
            "isAllUser": 0,
            "userList": "[\"targetmail@qq.com\"]",
            "lastAlertTime": null
        },
        {
            "ruleId": 3,
            "ruleName": "证书有效期告警",
            "enable": 0,
            "alertType": 3,
            "alertLevel": 1,
            "alertIntervalSeconds": 3600,
            "alertContent": "证书将在{time}过期，请到\"证书管理\"页面查看具体信息",
            "contentParamList": "[\"{time}\"]",
            "description": null,
            "createTime": "2019-10-29 20:02:30",
            "modifyTime": "2019-10-29 20:02:30",
            "isAllUser": 0,
            "userList": "[\"targetmail@qq.com\"]",
            "lastAlertTime": null
        }
    ]
}
```


6.4 更新告警类型配置

更新告警类型配置的内容；目前仅支持更新原有的三个邮件告警的配置，不支持新增配置；

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/alert**
- 请求方式：PUT
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/mailServer/config
```

```
{
    "ruleId": 3,
    "ruleName": "证书有效期告警",
    "enable": 0,
    "alertType": 3,
    "alertIntervalSeconds": 1800,
    "alertContent": "证书将在{time}过期，请到\"证书管理\"页面查看具体信息",
    "userList": "[\"targetmail@qq.com\"]",
    "alertLevel": 1
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "ruleId": 3,
        "ruleName": "证书有效期告警",
        "enable": 0,
        "alertType": 3,
        "alertLevel": 1,
        "alertIntervalSeconds": 1800,
        "alertContent": "证书将在{time}过期，请到\"证书管理\"页面查看具体信息",
        "contentParamList": "[\"{time}\"]",
        "description": null,
        "createTime": "2019-10-29 20:02:30",
        "modifyTime": "2019-11-07 10:35:03",
        "isAllUser": 0,
        "userList": "[\"targetmail@qq.com\"]",
        "lastAlertTime": null
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

6.5 开启/关闭 告警类型

修改告警类型配置中的enable，0-关闭，1-开启；

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/alert/toggle**
- 请求方式：PUT
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/alert/toggle
```

```
{
   "ruleId": 3,
   "enable": 1
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "ruleId": 3,
        "ruleName": "证书有效期告警",
        "enable": 1,
        "alertType": 3,
        "alertLevel": 1,
        "alertIntervalSeconds": 1800,
        "alertContent": "证书将在{time}过期，请到\"证书管理\"页面查看具体信息",
        "contentParamList": "[\"{time}\"]",
        "description": null,
        "createTime": "2019-10-29 20:02:30",
```

(下页继续)

(续上页)

```
        "modifyTime": "2019-11-07 10:35:03",
        "isAllUser": 0,
        "userList": "[\"targetmail@qq.com\"]",
        "lastAlertTime": null
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

6.6 获取出块监控信息

获取出块周期、块大小、平均TPS的监控数据

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/stat?groupId={groupId}&beginDate={beginDate}&endDate={endDate}&contrastBeginDate={contrastBe**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/stat?groupId=1&
↪gap=60eginDate=1617811200000&endDate=1617871955000&contrastBeginDate=&
↪contrastEndDate=
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
        "code": 0,
        "message": "success",
        "data": [{
                "metricType": "blockSize",
                "data": {
                        "lineDataList": {
                                "timestampList": [1617866162706, 1617866462706,␣
↪1617866762706, 1617867062706, 1617867362706, 1617867662706, 1617867962706,␣
↪1617868262706, 1617868562706, 1617868862706, 1617869162706, 1617869462706,␣
↪1617869762706, 1617870062706, 1617870362706, 1617870664184, 1617870964184],(下页继续)
```

(续上页)


```
                                "valueList": [null, null, null, null, null, null,␣
↪null, null, null, null, null, null, null, null, null, null, null]
                        },
                        "contrastDataList": {
                                "timestampList": [],
                                "valueList": []
                        }
                }
        }, {
                "metricType": "blockCycle",
                "data": {
                        "lineDataList": {
                                "timestampList": null,
                                "valueList": [null, null, null, null, null, null,␣
↪null, null, null, null, null, null, null, null, null, null, null]
                        },
                        "contrastDataList": {
                                "timestampList": null,
                                "valueList": []
                        }
                }
        }, {
                "metricType": "tps",
                "data": {
                        "lineDataList": {
                                "timestampList": null,
                                "valueList": [null, null, null, null, null, null,␣
↪null, null, null, null, null, null, null, null, null, null, null]
                        },
                        "contrastDataList": {
                                "timestampList": null,
                                "valueList": []
                        }
                }
        }],
        "attachment": null
}
```


13.3.7 7 审计相关模块

7.1 获取用户交易监管信息列表

7.1.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/monitor/userList/{groupId}**
- 请求方式：GET
- 返回格式：JSON

7.1.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/monitor/userList/300001
```

7.1.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "userName": "SYSTEMUSER",
            "userType": 0,
            "groupId": null,
            "contractName": null,
            "contractAddress": null,
            "interfaceName": null,
            "transType": null,
            "transUnusualType": null,
            "transCount": null,
            "transHashs": null,
            "transHashLastest": null,
            "createTime": null,
            "modifyTime": null
        },
        {
            "userName": "asdf",
            "userType": 0,
            "groupId": null,
            "contractName": null,
            "contractAddress": null,
            "interfaceName": null,
            "transType": null,
            "transUnusualType": null,
            "transCount": null,
            "transHashs": null,
            "transHashLastest": null,
            "createTime": null,
            "modifyTime": null
        }
    ]
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

7.2 获取合约方法监管信息列表

7.2.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/monitor/interfaceList/{groupId}?userName={userName}**
- 请求方式：GET
- 返回格式：JSON

7.2.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/monitor/interfaceList/300001
```

7.2.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "userName": "SYSTEMUSER",
            "userType": 0,
            "groupId": null,
            "contractName": null,
            "contractAddress": null,
            "interfaceName": null,
            "transType": null,
            "transUnusualType": null,
            "transCount": null,
            "transHashs": null,
            "transHashLastest": null,
            "createTime": null,
            "modifyTime": null
        },
        {
            "userName": "asdf",
            "userType": 0,
            "groupId": null,
            "contractName": null,
            "contractAddress": null,
            "interfaceName": null,
            "transType": null,
            "transUnusualType": null,
            "transCount": null,
            "transHashs": null,
            "transHashLastest": null,
            "createTime": null,
            "modifyTime": null
        }
```

(下页继续)

(续上页)

```
    ]
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

7.3 获取交易hash监管信息列表

7.3.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/monitor/transList/{groupId}**
- 请求方式：GET
- 返回格式：JSON

7.3.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/monitor/transList/300001?
↪userName=0x5d97f8d41638a7b1b669b70b307bab6d49df8e2c&interfaceName=0x4ed3885e
```

7.3.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "groupId": 300001,
        "userName": "0x5d97f8d41638a7b1b669b70b307bab6d49df8e2c",
        "interfaceName": "0x4ed3885e",
        "totalCount": 1,
        "transInfoList": [
            {
                "transCount": 1,
                "time": "2019-03-13 15:41:56"
            }
        ]
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

7.4 获取异常用户信息列表

7.4.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/unusualUserList/{groupId}/{pageNumber}/{pageSize}?userName={userName}**
- 请求方式：GET
- 返回格式：JSON

7.4.2 参数信息详情

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/monitor/unusualUserList/300001/1/10?
↪userName=
```

7.4.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "userName": "0x08b52f85638a925929cf62a3ac77c67415012c24",
            "transCount": 1,
            "hashs":
↪"0x43b50faa3f007c22cf5dd710c3561c5cde516e01a55b5b4acffd7d94cf61fc57",
            "time": "2019-03-13 22:28:29"
        }
    ],
    "totalCount": 1
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

7.5 获取异常合约信息列表

7.5.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/unusualContractList/{groupId}/{pageNumber}/{pageSize}?contractAddress={contractAddress}**
- 请求方式：GET
- 返回格式：JSON

7.5.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/monitor/unusualContractList/300001/1/10?
↪contractAddress=
```

7.5.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "totalCount": 1,
    "data": [
        {
            "contractName": "0x00000000",
            "contractAddress": "0x0000000000000000000000000000000000000000",
            "transCount": 3,
            "hashs":
↪"0xc87e306db85740895369cc2a849984fe544a6e9b0ecdbd2d898fc0756a02a4ce",
            "time": "2019-03-13 15:41:56"
        }
    ]
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

13.3.8 8 群组信息模块

8.1 获取群组概况

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/group/general/{groupId}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/group/general/group
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": {
    "groupId": "group",
    "orgCount": 0,
    "nodeCount": 2,
    "contractCount": 51,
    "transactionCount": 249,
    "latestBlock": 249
  },
  "attachment": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

8.2 获取所有群组列表

默认只返回groupStatus为1的群组ID，可传入groupStatus筛选群组 (1-normal, 2-maintaining, 3-conflict-genesisi, 4-conflict-data)

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/group/all**

- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

无

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/group/all
http://127.0.0.1:5001/WeBASE-Node-Manager/group/all/{groupStatus}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "groupId": "group",
      "groupName": "group",
      "groupStatus": 1,
      "nodeCount": 2,
      "latestBlock": 0,
      "transCount": 0,
      "createTime": "2021-12-06 11:21:55",
      "modifyTime": "2021-12-06 17:12:06",
      "description": "synchronous",
      "groupType": 1,
      "encryptType": 0,
      "groupTimestamp": null,
      "nodeIdList": null,
      "chainId": 0,
      "chainName": ""
    }
  ],
  "totalCount": 1
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

8.3 查询每日交易数据

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/group/transDaily/{groupId}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/group/transDaily/group
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "day": "2021-11-29",
      "groupId": "group",
      "transCount": 75
    },
    ...
  ],
  "attachment": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

8.4 刷新群组列表

刷新节点管理服务的群组列表，检查本地群组数据与链上群组数据是否有冲突，检查多个节点之间的创世块是否一致，并从链上拉取最新的群组列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/group/update**

- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/group/update
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {}
}
```

8.5 获取所有群组列表（包含异常群组）

返回所有群组，包含正常运行、维护中、脏数据冲突、创世块冲突4种状态的群组

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/group/all/invalidIncluded**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表 | 序号 | 输入参数 | 类型 | 必填 | 备注 | |——|————-|————|——|————————-| | 1 | pageSize | Integer | 是 | 每页记录数 | | 2 | pageNumber | Integer | 是 | 当前页码 |

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/group/all/invalidIncluded/{pageNumber}/
↪{pageSize}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "groupId": "group",
      "groupName": "group",
      "groupStatus": 1,
      "nodeCount": 2,
      "latestBlock": 0,
      "transCount": 0,
      "createTime": "2021-12-06 11:21:55",
      "modifyTime": "2021-12-06 17:12:06",
      "description": "synchronous",
      "groupType": 1,
      "encryptType": 0,
      "groupTimestamp": null,
      "nodeIdList": null,
      "chainId": 0,
      "chainName": ""
    }
  ],
  "totalCount": 1
}
```


- 失败：


```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```


8.6 删除群组所有数据

删除指定群组编号的群组的所有数据，包含节点数据、交易数据、交易审计数据等等。

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/group/{groupId}**
- 请求方式：DELETE
- 返回格式：JSON

请求参数

1）入参表 | 序号 | 输入参数 | 类型 | 必填 | 备注 | |——|————-|————|——|————————-| | 1 | groupId | String | 是 | 群组编号 |

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/group/{groupId}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": null,
  "attachment": null
}
```

- 失败：

```
{
    "code": 202006,
    "message": "invalid group id"
}
```

8.7 配置群组备注信息

配置群组的备注信息，用于数据大屏中大标题的展示。（默认备注为"synchronous"）

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/group/description**
- 请求方式：PUT
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/group/description
```

{ "description": "溯源存证应用", "groupId": "group" }

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": 1,
```

(下页继续)

(续上页)

```
  "attachment": null
}
```

8.8 获取单个群组详细信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/group/detail/{groupId}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/group/detail/{groupId}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": {
    "groupId": 1,
    "groupName": "group1",
    "groupStatus": 1,
    "nodeCount": 4,
    "latestBlock": 0,
    "transCount": 0,
    "createTime": "2021-07-29 14:44:59",
    "modifyTime": "2021-09-29 16:42:05",
    "description": "溯源存证应用",
    "groupType": 1,
    "groupTimestamp": null,
    "nodeIdList": null,
    "chainId": 0,
    "chainName": "default",
    "encryptType": 0
  },
  "attachment": null
}
```

13.3.9 9 节点管理模块

9.1 查询节点列表

查询WeBASE本地保存的节点列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/node/nodeList/{groupId}/{pageNumber}/{pageSize}?nodeName={nodeName}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/node/nodeList/300001/1/10?nodeName=
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "totalCount": 1,
    "data": [
        {
            "nodeId":
↪"06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b35067c1685d4343e6ad84b45
↪",
            "nodeName": "1_
↪06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b35067c1685d4343e6ad84b45c
↪",
            "groupId": 1,
            "nodeIp": "127.0.0.1",
            "p2pPort": null,
            "description": null,
            "blockNumber": 589,
            "pbftView": 11,
            "nodeActive": 1,
            "createTime": "2021-07-29 14:44:59",
            "modifyTime": "2021-09-29 16:17:38",
            "city": "430500",
            "agency": "GZ"
        },
    ]
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

9.2 查询节点信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/node/nodeInfo/{groupId}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/node/nodeInfo/1
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "nodeId":
↪"06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b35067c1685d4343e6ad84b45c
↪",
        "nodeName": "1_
↪06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b35067c1685d4343e6ad84b45c
↪",
        "groupId": "group",
        "nodeIp": "127.0.0.1",
        "p2pPort": null,
        "description": null,
        "blockNumber": 589,
        "pbftView": 11,
        "nodeActive": 1,
        "createTime": "2021-07-29 14:44:59",
        "modifyTime": "2021-09-29 16:17:38",
        "city": "430500",
        "agency": "GZ"
    },
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

9.3 查询群组下的节点ID列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/node/nodeIdList/{groupId}/{pageNumber}/{pageSize}**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

**

```
http://127.0.0.1:5001/WeBASE-Node-Manager/node/nodeIdList/group
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": [

↪"fe57d7b39ed104b4fb2770ae5aad7946bfd377d0eb91ab92a383447e834c3257dec56686551d08178f2d5f40d9fad6
↪",

↪"65bc44d398d99d95a9d404aa16e4bfbc2f9ebb40f20439ddef8575a139dc3a80310cfc98a035bd59a67cc5f659f519
↪",

↪"95efafa5197796e7edf647191de83f4259d7cbb060f4bac5868be474037f49144d581c15d8aef9b07d78f18041a5f4
↪",

↪"06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b35067c1685d4343e6ad84b45
↪"
  ],
  "attachment": null
}
```

9.4 配置节点备注信息

可配置节点的IP、机构、城市信息，可用于数据监控大屏展示

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/node/description**
- 请求方式：PUT
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/node/description
```

{ “agency”: “org1”, “city”: “GZ”, “nodeId”: “06269e130f8220ebaa78e67832df0de6b4c5ee3f1b14e64ab2bae26510a4bcf997454b3
“nodeIp”: “127.0.0.1” }

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success"
}
```

9.5 查询节点的城市列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/node/city/list**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

无

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/node/city/list
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "city": "110100", // 此处为城市ID
      "count": 2
    },
    {
      "city": "430500",
      "count": 1
    },
    {
      "city": "440300",
      "count": 1
    }
  ],
  "attachment": null
}
```

13.3.10 10 角色管理模块

10.1 查询角色列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**role/roleList**
- 请求方式：GET
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/role/roleList?groupId=300001&pageNumber=&
↪pageSize=&roleId=&roleName=
```

返回参数

1）出参表

2）出参示例

- 成功：


```
{
    "code": 0,
    "message": "success",
    "totalCount": 2,
    "data": [
        {
            "roleId": 100000,
            "roleName": "admin",
            "roleNameZh": "管理员",
            "roleStatus": 1,
            "description": null,
            "createTime": "2019-02-14 17:33:50",
            "modifyTime": "2019-02-14 17:33:50"
        },
        {
            "roleId": 100001,
            "roleName": "visitor",
            "roleNameZh": "访客",
            "roleStatus": 1,
            "description": null,
            "createTime": "2019-02-14 17:33:50",
            "modifyTime": "2019-02-14 17:33:50"
        }
    ]
}
```


- 失败：


```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```


13.3.11 11 用户管理模块

11.1 新增私钥用户

11.1.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/user/userInfo**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

11.1.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/user/userInfo
```

```
{
  "account": "admin",
  "description": "string",
  "groupId": "group",
  "userName": "frank",
  "userType": 0
}
```

11.1.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": {
    "userId": 700006,
    "userName": "frank",
    "account": "admin",
    "groupId": "group",
    "publicKey":
↪"04aa23e6fcc6099c622d7cc9da0ab30019c6a349bfd71d69c83f7cf65a00a5ff9ed5e6fa0ab12c21cc46f0af0aed361
↪",
    "privateKey": null,
    "userStatus": 1,
    "chainIndex": null,
    "userType": 0,
    "address": "0xd24488a4d51661694f0c31f2cc688d1670f51d29",
    "signUserId": "c5a76d781e124c7db7279af39819ed2f",
    "appId": "group",
    "hasPk": 1,
    "description": "string",
    "createTime": "2021-12-06 17:53:53",
    "modifyTime": "2021-12-06 17:53:53"
  },
  "attachment": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

11.2 绑定公钥用户

11.2.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/user/bind**
- 请求方式：POST

- 请求头：Content-type: application/json
- 返回格式：JSON

11.2.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/user/userInfo
```

```
{
  "account": "admin",
  "description": "string",
  "groupId": "group",
  "publicKey":
↪"04aa23e6fcc6099c622d7cc9da0ab30019c6a349bfd71d69c83f7cf65a00a5ff9ed5e6fa0ab12c21cc46f0af0aed361
↪",
  "userName": "alex",
  "userType": 0
}
```

11.2.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "userId": 700007,
        "userName": "asdfvw",
        "groupId": 300001,
        "publicKey":
↪"0x4189fdacff55fb99172e015e1adb96dc77b0cae1619b1a41cc360777bee6682fcc9752d8aabf144fbf610a3057fd4
↪",
        "userStatus": 1,
        "userType": 1,
        "address": "0x40ec3c20b5178401ae14ad8ce9c9f94fa5ebb86a",
        "hasPk": 1,
        "description": "sda",
        "account": "admin",
        "createTime": "2019-03-15 18:00:27",
        "modifyTime": "2019-03-15 18:00:27"
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

11.3 修改用户备注

11.3.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/user/userInfo**
- 请求方式：PUT
- 请求头：Content-type: application/json
- 返回格式：JSON

11.3.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/user/userInfo
```

```
{
    "userId": "700006",
    "description": "newDescription"
}
```

11.3.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": {
    "userId": 700006,
    "userName": "frank",
    "account": "admin",
    "groupId": "group",
    "publicKey":
↪"04aa23e6fcc6099c622d7cc9da0ab30019c6a349bfd71d69c83f7cf65a00a5ff9ed5e6fa0ab12c21cc46f0af0aed361
↪",
    "privateKey": null,
    "userStatus": 1,
    "chainIndex": null,
    "userType": 0,
    "address": "0xd24488a4d51661694f0c31f2cc688d1670f51d29",
    "signUserId": "c5a76d781e124c7db7279af39819ed2f",
    "appId": "group",
    "hasPk": 1,
    "description": "string",
    "createTime": "2021-12-06 17:53:53",
    "modifyTime": "2021-12-06 19:13:17"
  },
  "attachment": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

11.4 导入私钥用户

可在页面导入WeBASE-Front所导出的私钥txt文件
其中私钥字段用Base64加密

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/user/import**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表
2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/user/import
```

```
{
    "privateKey":
↪"OGFmNWIzMzNmYTc3MGFhY2UwNjdjYTY3ZDRmMzE4MzU4OWRmOThkMjVjYzEzZGFlMGJmODhkYjhlYzVhMDcxYw==
↪",
    "groupId": "group",
    "description": "密钥拥有者",
    "userName": "user1",
    "account": "admin",
    "userId": 0,
    "userType": 0
}
```

返回参数

1）出参表
2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success"
}
```

- 失败：

```
{
    "code": 201031,
    "message": "privateKey decode fail",
    "data": null
}
```

11.5 导入.pem私钥

可导入控制台所生成的私钥.pem文件

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/user/importPem**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/user/importPem
```

```
{
    "pemContent":"-----BEGIN PRIVATE KEY-----
↪\nMIGEAgEAMBAGByqGSM49AgEGBSuBBAAKBG0wawIBAQQgC8TbvFSMA9y3CghFt51/
↪\nXmExewlioX99veYHOV7dTvOhRANCAASZtMhCTcaedNP+H7iljbTIqXOFM6qm5aVs\nfM/
↪yuDBK2MRfFbfnOYVTNKyOSnmkY+xBfCR8Q86wcsQm9NZpkmFK\n-----END PRIVATE KEY-----\n",
    "groupId": "group",
    "description": "密钥拥有者",
    "userName": "user2",
    "account": "admin",
    "userType": 0
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success"
}
```

- 失败：

```
{
    "code": 201232,
    "message": "Pem file format error, must surrounded by -----XXXXX PRIVATE KEY---
↪--"",
    "data": null
}
```

11.6 导入.p12私钥

可导入控制台生成的私钥.p12文件

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/user/importP12**
- 请求方式：POST
- 请求头：Content-type: **form-data**
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/user/importP12
```

使用form-data传参

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success"
}
```

- 失败：（p12文件的密码错误）

```
{
    "code": 201236,
    "message": "p12's password not match",
    "data": null
}
```

11.7 导出明文私钥

可在页面导出WeBASE-Front所导出的私钥txt文件

其中私钥字段用Base64加密

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/user/export/{userId}**
- 请求方式：POST
- 请求头：Content-type: form-data
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/user/export/300001
```

返回参数

1）出参表

2）出参示例

- 成功：


```
{
        "code": 0,
        "message": "success",
        "data": {
                "userId": 700003,
                "userName": "frank",
                "account": "admin",
                "groupId": "group",
                "publicKey":
↪"04d01115d548e7561b15c38f004d734633687cf4419620095bc5b0f47070afe85aa9f34ffdc815e0d7a8b64537e17b
↪",
                "privateKey":
↪"MDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwYw==
↪",
                "userStatus": 1,
                "chainIndex": null,
                "userType": 1,
                "address": "0xdbc23ae43a150ff8884b02cea117b22d1c3b9796",
                "signUserId": "b751efc5d0cc4e12b90908b1f2670258",
                "appId": "group",
                "hasPk": 1,
                "description": "",
                "createTime": "2021-04-06 21:24:12",
                "modifyTime": "2021-04-06 21:24:12"
        },
        "attachment": null
}
```

11.8 导出.pem私钥

导出pem格式的私钥文件

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/user/exportPem**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/user/exportPem
```

返回参数

1）出参表

2）出参示例

- 成功：

```
-----BEGIN PRIVATE KEY-----
MIGNAgEAMBAGByqGSM49AgEGBSuBBAAKBHYwdAIBAQQgnAXS1DYA90nML3Kge4Qd
IgMXiQ9cojmRgyjo1BLYXOqgBwYFK4EEAAqhRANCAATLPOzgUzNbo6UeCAjYv2++
FwlBmT1Sa7goXELaazyJEJLbAlAFGB6qvjdA9m2nx5+rTmfGoSuQK9T2hC/vWJfq
-----END PRIVATE KEY-----
```

11.9 导出.p12私钥

导出pem格式的私钥文件

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/user/exportP12**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/user/exportP12
```

返回参数

1）出参表

2）出参示例

- 成功：

```
// 二进制流
```

13.3.12 12 合约方法管理模块

12.1 新增合约方法

12.1.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/method/add**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

12.1.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/method/add
```

```
{
  "groupId": "string",
  "methodList": [
    {
      "abiInfo": "string",
      "methodId": "string",
      "methodType": "string"
    }
  ]
}
```

1.1.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

12.2 根据方法编号查询

12.1.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/method/findById/{groupId}/{methodId}**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

12.1.2 请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/method/findById/group/add
```

1.1.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "methodId": "methodIasdfdtttttt",
        "groupId": 2,
        "abiInfo": "fsdabiTestfd232222",
```

(下页继续)

(续上页)

```
        "methodType": "function",
        "createTime": "2019-04-16 16:59:27",
        "modifyTime": "2019-04-16 16:59:27"
    }
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

13.3.13 13 系统管理模块

系统管理中的权限管理接口

- 使用FISCO BCOS v2.5.0 与 WeBASE-Node-Manager v1.4.1 (及)以上版本将使用预编译合约中的ChainGovernance接口(本章节接口*13.14*开始)，详情可参考FISCO BCOS基于角色的权限控制
- 使用低于FISCO BCOS v2.5.0 与 WeBASE-Node-Manager v1.4.1版本，则使用接口13.1至13.4接口

13.1获取系统配置

根据群组id获取系统配置SystemConfig的list列表，目前只支持tx_count_limit, tx_gas_limit两个参数。

13.1.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/sys/config/list**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

13.1.2 请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/sys/config/list?groupId=1&pageSize=10&
↪pageNumber=1
```

13.1.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "id": 6,
            "groupId": 1,
            "fromAddress": "0x",
            "configKey": "tx_gas_limit",
            "configValue": "300000000"
        },
        {
            "id": 5,
            "groupId": 1,
            "fromAddress": "0xd0b56b4ce0e8ce5e064f896d9ad1c16b2603f285",
            "configKey": "tx_count_limit",
            "configValue": "10002"
        }
    ],
    "totalCount": 2
}
```


13.2 设置系统配置

系统配置管理员设置系统配置，目前只支持tx_count_limit, tx_gas_limit两个参数。

13.2.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/sys/config**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

13.2.2 请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/sys/config
```


```
{
    "groupId": "group",
    "fromAddress": "0xd5bba8fe456fce310f529edecef902e4b63129b1",
    "configKey": "tx_count_limit",
    "configValue": "1001",
    "signUserId": "string"
}
```

13.2.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success"
 }
```

- 失败：

```
{
    "code": -50000,
    "message": "permission denied"
}
```

13.3 获取节点列表(节点管理)

获取节点的list列表，包含节点id，节点共识状态。

注：接口返回所有的共识/观察节点（无论运行或停止），以及正在运行的游离节点

13.3.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/precompiled/consensus/list**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

13.3.2 请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/precompiled/consensus/list?groupId=1&
↪pageSize=10&pageNumber=1
```

13.3.3 返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "nodeId":
↪"13e0f2b94cbce924cc3737385a38587939e809fb786c4fc34a6ba3ea97693bccfa173b352ac41f1dbb97e9e4910ccbe
↪",
            "nodeType": "sealer"
        },
        {
            "nodeId":
↪"bce4b2269c25c2cdba30155396bfe90af08c3c08cff205213477683117e4243ebe26588479519e636a5d5d93545cab
↪",
            "nodeType": "sealer"
        },
        {
            "nodeId":
↪"e815cc5637cb8c3274c83215c680822e4a0110d0a8315fcf03e43e8e3944edd758c8b173c4e0076f599aa1f853fa20
↪",
            "nodeType": "sealer"
        }
    ],
    "totalCount": 3
}
```


13.4 设置节点共识状态接口（节点管理）

节点管理相关接口，可用于节点三种共识状态的切换。分别是共识节点sealer, 观察节点observer, 游离节点remove

13.4.1 传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/precompiled/consensus**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

13.4.2 请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/precompiled/consensus
```


```
{
    "groupId": "group",
    "fromAddress": "0xd5bba8fe456fce310f529edecef902e4b63129b1",
    "nodeType": "remove",
    "nodeId":
↪"224e6ee23e8a02d371298b9aec828f77cc2711da3a981684896715a3711885a3177b3cf7906bf9d1b84e597fad1e00
↪",
    "signUserId": "string",
```


(下页继续)

(续上页)

```
    "weight": 0
}
```

13.4.3 返回参数

1）出参表

2）出参示例

- 成功：

```
[
    {
      "code": 0,
      "message": "success"
     }

]
```

- 失败：

```
{
    "code": -51000,
    "message": "nodeId already exist"
}
```

13.3.14 14 证书管理模块

14.1 获取证书列表接口

获取证书的list列表，返回的列表包含证书指纹、证书内容、证书名字、证书的父证书、证书对应nodeid（节点证书）、证书有效期

注：首次启动项目会自动拉取每一个Front的证书

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/cert/list**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/cert/list
```

返回参数

1）出参表

2）出参示例

- 成功：


```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "fingerPrint": "814D51FB7CBAB33676FE73E8FBBFECB3D3B1301A",
            "certName": "sdk",
            "content": "-----BEGIN CERTIFICATE-----
↪\nMIICOTCCASGgAwIBAgIJAKHsAYI3TsAOMA0GCSqGSIb3DQEBCwUAMDgxEDAOBgNV\nBAMMB2FnZW5jeUExEzARBgNVBAo
↪LB7tELOiyCiIEhLScprb1LjvDDt2RDGjGjAYMAkGA1UdEwQCMAAwCwYD\nVR0PBAQDAgXgMA0GCSqGSIb3DQEBCwUAA4IBA
↪cnNhq5HVObbWxzfu7gn3+IN\nyQEeqdbGdzlu1EDcaMgAz6p2W3+FG/tmx/
↪yrNza29cYekWRL44OT5LOUPEKrJ4bJ\neOBRY4QlwZPFmM0QgP7DoKxHXldRopkmvqT4pbW51hWvPgj7KrdqwbVWzuWQuI3
↪----END CERTIFICATE-----\n",
            "certType": "node",
            "publicKey":
↪"9efdad22886fdef16700725b9f311fab73a0d28d2da286ec9e70c4012001bd373b687832410c2ed7ff2c1eed10b3a2c
↪",
            "address": "5cb81b06ef0734fff99929c5deae4a5b25e800cc",
            "father": "EEBAAB2F674D05CF1EAD70367B4D2A928D894EF8",
            "validityFrom": 1562860800000,
            "validityTo": 1878220800000,
            "createTime": 1569686400000
        }
    ],
    "totalCount": 1
}
```


14.2 根据指纹获取证书接口

根据指纹获取单个证书

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/cert**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/cert?
↪fingerPrint=814D51FB7CBAB33676FE73E8FBBFECB3D3B1301A
```

返回参数

1）出参表

2）出参示例

- 成功：’


```
{
    "code": 0,
    "message": "success",
    "data": {
        "fingerPrint": "EEBAAB2F674D05CF1EAD70367B4D2A928D894EF8",
        "certName": "agencyA",
        "content": "-----BEGIN CERTIFICATE-----
↪\nMIIDADCCAeigAwIBAgIJAJUF2Dp1a9U6MA0GCSqGSIb3DQEBCwUAMDUxDjAMBgNV\nBAMMBWNoYWluMRMwEQYDVQQKDAp
↪bEpjEXsu2cXH0D6BHZk+wvuxG6\nezXWq5MYjCw3fQiSRWkDYoxzWgulkRyYROF1xoZeNGQssReFmCgP+pcQwRxjcq8z\nI
↪EE3tsJ0ae3ax6zixCT66aV49S27cMcisS+XKP/
↪q\nEVPxhO7SUjnzZY69MgZzNSFxCzIbapnlmYAOS26vIT0taSkoKXmIsYssga45XPwI\n7YBVCc/
↪34kHzW9xrNjyyThMWOgDsuBqZN9xvapGSQ82Lsh7ObN0dZVUCAwEAAaMQ\nMA4wDAYDVR0TBAUwAwEB/
↪zANBgkqhkiG9w0BAQsFAAOCAQEAu3aHxJnCZnICpHbQ\nv1Lc5tiXtAYE9aEP5cxb/
↪cO14xY8dS+t0wiLIvyrE2aTcgImzr4BYNBm1XDt5suc\nMpzha1oJytGv79M9/WnI/
↪BKmgUqTaaXOV2Ux2yPX9SadNcsD9/IbrV0b/
↪hlsPd6M\nK8w7ndowvBgopei+A1NQY6jTDUKif4RxD4u5HZFWUu7pByNLFaydU4qBKVkucXOq\nxmWoupL5XrDk5o490kiz
↪Zgufqtb4w6oUr3lrQASAbFB3lID/
↪P1ipi0DwX7kZwVX\nECDLYvr+eX6GbTClzn0JGuzqV4OoRo1rrRv+0tp1aLZKpCYn0Lhf6s1iw/
↪kCeM2O\nnP9l2Q==\n-----END CERTIFICATE-----\n",
        "certType": "agency",
        "publicKey": "",
        "address": "",
        "father": "",
        "validityFrom": 1562860800000,
        "validityTo": 1878220800000,
        "createTime": 1569686400000
    }
}
```


14.3 导入证书接口

导入保存证书文件

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/cert**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

14.3.2 请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/cert
```


```
{
    "content": "-----BEGIN CERTIFICATE-----
↪\nMIICOzCCASOgAwIBAgIJANJZtoFLZsGcMA0GCSqGSIb3DQEBCwUAMDgxEDAOBgNVBAMMB2FnZW5jeUExEzARBgNVBAoMCm
↪xNolQDBOYZX1E0XtUwMUp6Az2xbmSH/7S3sXJCwgHZrtoiKkcFLbss1TDk/UdUya4n/
↪dz4BcH3OzR2MvMHenA8kh4yaofJNsJeXFqPHAbI5+yUVK2+VK2hI0o=\n-----END CERTIFICATE----
↪-\n-----BEGIN CERTIFICATE-----
↪\nMIIDPTCCAiWgAwIBAgIJAKUGxOHHqV05MA0GCSqGSIb3DQEBCwUAMDUxDjAMBgNVBAMMBWNoYWluMRMwEQYDVQQKDApmaX
↪k5+B943CUPF/RDxZ8/7Q2v6Z+t+1v3Dc81aszMr/
↪8YyyCQWh0I3EdWyInsocZ2pBkjymetyE5VOSd+p7I8qc9PpHJKZjy2M9J5bePVjHJxleHP2u6I4QctjZoE8PJnZYT5Q0On0
↪KtyDKlyPTETXOFGMiLerWusXZxFgj0K0xhuXaNkbJI6AdhQnywgn755ugfBDzi24rfsk/
↪BkUf5DVitfWePh4C7zaCZIeTTr8whV3twE2BTv4LENdidxCVUHN1JBvZNGyHaH4gIbwtsZcCAwEAAaNQME4wHQYDVR0OBBYE
↪6LNJa8w2m+pAvC3ILIBcvpDe6lH3cMOz2HwCzFkKlT8Ji1HwsKPywx/
↪9fmO60RvEoPIBanziqONLb8HDUT0QHz3jgCTj46URM6hXIEhFwg4KekpzvqaLPRHHtoCrcebUAmySOlNvlwkSnxJnufp0zF
↪r21ZRMeu/
↪hNaUb1gOzP06NOB7NodOQ5MR7ItVXyN9rl3fABP3rUFIJ+Z11WmSldaCRCQMlEOkdD8LGFYVj4Q5fx06hcJlPd2arnxALWr
↪----END CERTIFICATE-----\n"
}
```


返回参数

1）出参表

2）出参示例

- 成功：


```
{
    "code": 0,
    "message": "success",
    "data": 2
}
```


- 失败：


```
{
    "code": 202060,
    "message": "cert handle error",
    "data": "Could not parse certificate: java.io.IOException: Empty input"
}
```


14.4 删除证书接口

根据证书指纹删除一个证书

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/cert**
- 请求方式：DELETE
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/cert
```

```
{
    "fingerPrint": "F588C511F5471860120F7BE8127DE026ADD8378C"
}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": 1
}
```

14.5. 获取明文SDK证书与私钥

获取Front使用的sdk证书（包含链证书、sdk证书和sdk私钥）的内容

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/cert/sdk//{frontId}**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）参数表

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/cert/sdk/1
```

响应参数

1）参数表

1）数据格式

a、成功：

```
{
  "sdk.key": "-----BEGIN PRIVATE KEY-----
↪\nMIGEAgEAMBAGByqGSM49AgEGBSuBBAAKBG0wawIBAQQgxqr/d/VgQ0fAr/
↪KvyAeW\nJ6bD1tqxZ5gYOdfIJiK7WOmhRANCAAT3g/OsuSAD2I/dKLWnZTbMGQ8l9WnkD/
↪wr\npyoiQkMy1qI5/3Sj4WFKGcVu9vhsd0nLoP+y1QttYKM0m5QGcuhP\n-----END PRIVATE KEY---
↪--\n",
  "ca.crt": "-----BEGIN CERTIFICATE-----
↪\nMIIBsDCCAVagAwIBAgIJAPwQ7ISyofOIMAoGCCqGSM49BAMCMDUxDjAMBgNVBAMM\nBWNoYWluMRMwEQYDVQQKDApmaXN
↪Hz/Q2SAin5bMnE1nOFMB8GA1UdIwQYMBaAFBBSyZi8k/Hz/
↪Q2SAin5bMnE1nOF\nMAwGA1UdEwQFMAMBAf8wCgYIKoZIzj0EAwIDSAAwRQIgEpuPZypVImOtDty9p50X\njeD4wdgzHXpd
↪----END CERTIFICATE-----\n",
  "sdk.crt": "-----BEGIN CERTIFICATE-----
↪\nMIIBeDCCAR+gAwIBAgIJAJoEtSMUsa8HMAoGCCqGSM49BAMCMDgxEDAOBgNVBAMM\nB2FnZW5jeUExEzARBgNVBAoMCmZp
↪OsuSAD2I/dKLWnZTbMGQ8l9WnkD/wrpyoiQkMy1qI5/
↪3Sj\n4WFKGcVu9vhsd0nLoP+y1QttYKM0m5QGcuhPoxowGDAJBgNVHRMEAjAAMAsGA1Ud\nDwQEAwIF4DAKBggqhkjOPQQD
↪B2a\n+vrHMm6NwtliRAIgRH4gSF0XLmpVOEO21bJFDGWm9siIX0cnj0R3kNGZcB4=\n-----END␣
↪CERTIFICATE-----\n-----BEGIN CERTIFICATE-----
↪\nMIIBcTCCARegAwIBAgIJANrOZ+FrVNpIMAoGCCqGSM49BAMCMDUxDjAMBgNVBAMM\nBWNoYWluMRMwEQYDVQQKDApmaXN
↪OrPsMc9CrrYBsWdwOGhdx\nfNTJA1ss+vngjrhAmWHczvbh+E1WOlDGzpCumeqjEDAOMAwGA1UdEwQFMAMBAf8w\nCgYIKo
↪qHg0e8BG9ptEv7Do8caOPj33F+yOQ==\n-----END CERTIFICATE-----\n-----BEGIN␣
↪CERTIFICATE-----
↪\nMIIBsDCCAVagAwIBAgIJAPwQ7ISyofOIMAoGCCqGSM49BAMCMDUxDjAMBgNVBAMM\nBWNoYWluMRMwEQYDVQQKDApmaXN
↪Hz/Q2SAin5bMnE1nOFMB8GA1UdIwQYMBaAFBBSyZi8k/Hz/
↪Q2SAin5bMnE1nOF\nMAwGA1UdEwQFMAMBAf8wCgYIKoZIzj0EAwIDSAAwRQIgEpuPZypVImOtDty9p50X\njeD4wdgzHXpd
↪----END CERTIFICATE-----\n"
}
```


14.6. 获取SDK证书与私钥压缩包

获取Front使用的sdk证书（包含链证书、sdk证书和sdk私钥）的zip压缩包

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/cert/sdk/zip/{frontId}**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）参数表

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/cert/sdk/zip/1
```

响应参数

1）参数表

1）数据格式

a、成功：

```
headers:  content-disposition: attachment;filename*=UTF-8''conf.zip

{
    // 二进制流
}
```

13.3.15 15 订阅事件管理

15.1 获取已订阅的出块事件列表

获取所有前置中已订阅的节点出块事件列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/event/newBlockEvent/list/{groupId}**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/event/newBlockEvent/list/{groupId}/
↪{pageNumber}/{pageSize}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "frontInfo": "127.0.0.1",
            "id": "8aba82b570f22a750170f22bcab90000",
            "eventType": 1,
            "appId": "app2",
            "groupId": 1,
            "exchangeName": "group001",
            "queueName": "user1",
            "routingKey": "app2_block_b63",
            "createTime": "2020-03-19 17:42:01"
```

(下页继续)

(续上页)

```
        }
    ],
    "totalCount": 1
}
```

15.2 获取已订阅的合约Event事件列表

获取所有前置中已订阅的合约Event事件列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/event/contractEvent/list/{groupId}**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/event/contractEvent/list/{groupId}/
↪{pageNumber}/{pageSize}
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "frontInfo": "127.0.0.1",
            "id": "8aba82b5708095af01708095e4f70001",
            "eventType": 2,
            "appId": "app1",
            "groupId": 1,
            "exchangeName": "group001",
            "queueName": "user1",
            "routingKey": "user1_event_app1",
            "fromBlock": "latest",
            "toBlock": "latest",
            "contractAddress": "0x657201d59ec41d1dc278a67916f751f86ca672f7",
            "contractAbi": "[{\"constant\":false,\"inputs\":[{\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"payable\":false,\
↪"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"constant\":true,\
↪"inputs\":[],\"name\":\"get\",\"outputs\":[{\"name\":\"\",\"type\":\"string\"}],\
↪"payable\":false,\"stateMutability\":\"view\",\"type\":\"function\"},{\
↪"anonymous\":false,\"inputs\":[{\"indexed\":false,\"name\":\"name\",\"type\":\
↪"string\"}],\"name\":\"SetName\",\"type\":\"event\"}]",
```

(下页继续)

(续上页)

```
            "topicList": "SetName(string)",
            "createTime": "2020-02-26 16:21:12"
        }
    ],
    "totalCount": 1
}
```

15.3. 获取历史区块EventLog

接口描述

同步获取历史区块中的EventLog

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/event/eventLogs/list**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）参数表

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/event/eventLogs/list
```

```
{
    "groupId": "group",
    "contractAbi": [],
    "contractAddress": "0x19fb54101fef551187d3a79ea1c87de8d0ce754e",
    "fromBlock": 1,
    "toBlock": 1,
    "topics": {
        "eventName": "SetName",
        "indexed1": {
            "type": "bool",
            "value": true
        },
        "indexed2": {
            "type": "string",
            "value": null
        }
    }
}
```

响应参数

1）数据格式

成功：


```
{
        "code": 0,
        "message": "success",
        "data": [{
                "log": {
                        "logIndex": 0,
                        "transactionIndex": 0,
                        "transactionHash":
↪"0x67c8d9a1bc62586b9feb0c8b1127bf0030f649771db3e3d0d99cd99209851ed8",
                        "blockHash":
↪"0x0ca880c70a3f24dc5e6052cca4dbb50d9aa0ec973474e07d82e2f5281c54e582",
                        "blockNumber": 71,
                        "address": "0xd5d4fcf2a46831510f095bfb447bc945f99309f7",
                        "data":
↪"0x00000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000
↪",
                        "topics": [
↪"0x4df9dcd34ae35f40f2c756fd8ac83210ed0b76d065543ee73d868aec7c7fcf02"]
                }, // `org.fisco.bcos.sdk.model.EventLog`, 可参考java sdk
                "data": ["123"]
        }, {
                "log": {
                        "logIndex": 0,
                        "transactionIndex": 0,
                        "transactionHash":
↪"0x7c27078b372cee951dd102edcbbeab6b3b799212e4337d6bfac7f723e487a1fb",
                        "blockHash":
↪"0x47f8456a656f226d5a4a7206790ae864ae4fb645bca625b87cf0378ecca7e742",
                        "blockNumber": 72,
                        "address": "0xd5d4fcf2a46831510f095bfb447bc945f99309f7",
                        "data":
↪"0x00000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000
↪",
                        "topics": [
↪"0x4df9dcd34ae35f40f2c756fd8ac83210ed0b76d065543ee73d868aec7c7fcf02"]
                },
                "data": ["333"]
        }],
        "totalCount": 1
}
```


15.4. 获取ABI与合约所有合约信息

接口描述

获取导入的ABI与IDE中已部署合约所有合约的地址、合约名字信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/event/listAddress/{groupId}**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5001/WeBASE-Node-Manager/event/listAddress/{groupId}
```

响应参数

1）数据格式

成功：

```
{
    "code": 0,
    "message": "success",
    "data": [{
        "type": "contract",
        "contractAddress": "0x88156d500422a542435616e5a1e9d2df44c7fc70",
        "contractName": "Hello3"
    }, {
        "type": "contract",
        "contractAddress": "0xc2b3b552258b6016f80a070c1aa91bf9e3c48c53",
        "contractName": "Hello3"
    }, {
        "type": "abi",
        "contractAddress": "0x7a754bb46418c93b4cec7dcc6fef0676ae6a1e32",
        "contractName": "Hello3"
    }]
}
```

15.5. 根据地址获取ABI与合约的合约信息

接口描述

根据合约地址、合约类型（abi或contract）获取导入的ABI与IDE中已部署合约的合约地址、合约名字信息

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址：**/event/contractInfo/{groupId}/{type}/{contractAddress}**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5001/WeBASE-Node-Manager/event/contractInfo/{groupId}/{type}/
↪{contractAddress}
```

响应参数

1）数据格式

成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "abiId": 1,
        "groupId": "group",
        "contractName": "Hello3",
        "contractAddress": "0x7a754bb46418c93b4cec7dcc6fef0676ae6a1e32",
        "contractAbi": "",
        "contractBin": "",
        "createTime": "2020-11-06 15:12:51",
        "modifyTime": "2020-11-06 15:12:51"
    }
}
```

13.3.16 16 配置接口

16.1 查询是否使用国密

获取WeBASE-Node-Manager是否使用国密版

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/config/encrypt**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/config/encrypt
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": 1
}
```

16.2 查询WeBASE-Node-Manager的版本

获取WeBASE-Node-Manager服务的版本号

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/config/version**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/config/version
```

返回参数

1）出参表

2）出参示例

- 成功：

```
v1.4.0
```

16.3 获取服务ip和端口

返回本服务ip和端口。

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/config/ipPort**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/config/ipPort
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "ip": "127.0.0.1",
        "port": 5001
    },
    "attachment": null
}
```

- 失败：

```
{
    "code": 102000,
    "message": "system exception",
    "data": {}
}
```

16.4 查询已部署合约是否支持修改

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/config/isDeployedModifyEnable**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/config/isDeployedModifyEnable
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
    "code": 0,
    "message": "success",
    "data": true
}
```

16.5 获取配置列表

查询后台保存的配置列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/config/list**
- 请求方式：POST
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://127.0.0.1:5001/WeBASE-Node-Manager/config/list?type=1
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "id": 1,
      "configName": "docker 镜像版本",
      "configType": 1,
      "configValue": "v2.5.0",
      "createTime": 1590577419000,
      "modifyTime": 1590577419000
    }
```

(下页继续)

(续上页)

```
  ]
}
```

13.3.17 17. 链上全量数据接口

17.1 查询链上全量私钥用户列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/external/account/list/all/{groupId}/{pageNumber}/{pageSize}**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/external/account/list/all/1/1/10
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
        "code": 0,
        "message": "success",
        "data": [{
                "userId": null,
                "userName": null,
                "account": null,
                "groupId": 1,
                "publicKey": null,
                "privateKey": null,
                "userStatus": null,
                "chainIndex": null,
                "userType": null,
                "address": "0x11523906f9c6d4bbf2bf7ab2ff1049e7bdf2fbf6",
                "signUserId": null,
                "appId": null,
                "hasPk": null,
                "description": null,
                "createTime": "2021-04-07 16:34:42",
                "modifyTime": "2021-04-07 16:34:42",
                "extAccountId": 4,
                "transCount": 1,
```

(下页继续)

(续上页)


```
                "hashs":
↪"0x6889e8ea793d6326026b2a32bab023183ccc7846d3bcb9bd1001e1f08fb892c5"
        },{
                "userId": 700001,
                "userName": "2222",
                "account": "mars",
                "groupId": 1,
                "publicKey":
↪"04299feb42f324521464c9503220efaeaae99093d92ef08e6f9c1f76e761c2a57422c5fe4dc721e049dfdc05ff8e9b
↪",
                "privateKey": null,
                "userStatus": 1,
                "chainIndex": null,
                "userType": 1,
                "address": "0x6bc1eeb7d1bb3f1d8195336843bd938d51e594ee",
                "signUserId": "0c01b17d67734e95b1e8d5c55e78b4d8",
                "appId": "1",
                "hasPk": 1,
                "description": "",
                "createTime": "2021-04-06 21:14:04",
                "modifyTime": "2021-04-06 21:14:04",
                "extAccountId": 1,
                "transCount": null,
                "hashs": null
        }],
        "totalCount": 2
}
```


17.2 查询链上全量合约列表

传输协议规范

- 网络传输协议：使用HTTP协议
- 请求地址： **/external/contract/list/all/{groupId}/{pageNumber}/{pageSize}?type={type}**
- 请求方式：GET
- 请求头：Content-type: application/json
- 返回格式：JSON

请求参数

1）入参表

2）入参示例

```
http://localhost:5001/WeBASE-Node-Manager/external/contract/list/all/1/1/10?type=1
```

返回参数

1）出参表

2）出参示例

- 成功：

```
{
        "code": 0,
        "message": "success",
        "data": [{
                "abiId": 6,
                "groupId": 1,
                "account": "admin",
                "contractName": "SpecificIssuerController",
                "contractAddress": "0xce1d576181e1d68899a3f2b86c8e274657c07fea",
                "contractAbi": "[{\"constant\":false,\"inputs\":[{\"name\":\
↪"typeName\",\"type\":\"bytes32\"},{\"name\":\"addr\",\"type\":\"address\"}],\
↪"name\":\"addIssuer\",\"outputs\":[],\"payable\":false,\"stateMutability\":\
↪"nonpayable\",\"type\":\"function\"},{\"constant\":true,\"inputs\":[{\"name\":\
↪"typeName\",\"type\":\"bytes32\"},{\"name\":\"addr\",\"type\":\"address\"}],\
↪"name\":\"isSpecificTypeIssuer\",\"outputs\":[{\"name\":\"\",\"type\":\"bool\"}],
↪\"payable\":false,\"stateMutability\":\"view\",\"type\":\"function\"},{\
↪"constant\":false,\"inputs\":[{\"name\":\"typeName\",\"type\":\"bytes32\"},{\
↪"name\":\"extraValue\",\"type\":\"bytes32\"}],\"name\":\"addExtraValue\",\
↪"outputs\":[],\"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\
↪"function\"},{\"constant\":false,\"inputs\":[{\"name\":\"typeName\",\"type\":\
↪"bytes32\"}],\"name\":\"registerIssuerType\",\"outputs\":[],\"payable\":false,\
↪"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"constant\":true,\
↪"inputs\":[{\"name\":\"typeName\",\"type\":\"bytes32\"}],\"name\":\
↪"getExtraValue\",\"outputs\":[{\"name\":\"\",\"type\":\"bytes32[]\"}],\"payable\
↪":false,\"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\":true,\
↪"inputs\":[{\"name\":\"typeName\",\"type\":\"bytes32\"}],\"name\":\
↪"isIssuerTypeExist\",\"outputs\":[{\"name\":\"\",\"type\":\"bool\"}],\"payable\
↪":false,\"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\":false,\
↪"inputs\":[{\"name\":\"typeName\",\"type\":\"bytes32\"},{\"name\":\"addr\",\
↪"type\":\"address\"}],\"name\":\"removeIssuer\",\"outputs\":[],\"payable\":false,
↪\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"constant\":true,\
↪"inputs\":[{\"name\":\"typeName\",\"type\":\"bytes32\"},{\"name\":\"startPos\",\
↪"type\":\"uint256\"},{\"name\":\"num\",\"type\":\"uint256\"}],\"name\":\
↪"getSpecificTypeIssuerList\",\"outputs\":[{\"name\":\"\",\"type\":\"address[]\"}
↪],\"payable\":false,\"stateMutability\":\"view\",\"type\":\"function\"},{\
↪"inputs\":[{\"name\":\"specificIssuerDataAddress\",\"type\":\"address\"},{\"name\
↪":\"roleControllerAddress\",\"type\":\"address\"}],\"payable\":false,\
↪"stateMutability\":\"nonpayable\",\"type\":\"constructor\"},{\"anonymous\":false,
↪\"inputs\":[{\"indexed\":false,\"name\":\"operation\",\"type\":\"uint256\"},{\
↪"indexed\":false,\"name\":\"retCode\",\"type\":\"uint256\"},{\"indexed\":false,\
↪"name\":\"typeName\",\"type\":\"bytes32\"},{\"indexed\":false,\"name\":\"addr\",\
↪"type\":\"address\"}],\"name\":\"SpecificIssuerRetLog\",\"type\":\"event\"}]",
                "contractBin": "",
                "createTime": "2021-04-06 21:34:38",
                "modifyTime": "2021-04-06 21:34:38",
                "extContractId": 800014,
                "deployAddress": "0x222d68876bed5a33c8efe04c6fc9783031c22cd1",
                "deployTxHash":
↪"0xd04b35d7741f0568546de4fc29a880fa0f3727beef544497ca5816fa8793fa13",
                "deployTime": 1617716070000,
                "transCount": 0,
                "hashs": null
        },{
                "abiId": null,
                "groupId": 1,
                "account": null,
                "contractName": null,
                "contractAddress": "0x5e103407b2a06b32bad0fc578e1888036a3e26a8",
                "contractAbi": null,
                "contractBin": "",
                "createTime": null,
                "modifyTime": null,
```


(下页继续)

(续上页)

```
                "extContractId": 800007,
                "deployAddress": "0x222d68876bed5a33c8efe04c6fc9783031c22cd1",
                "deployTxHash":
↪"0x4c6f026e72b69e9ed1f3685f24bc055f5ef1bdde4778ee684a96d25deba6b34b",
                "deployTime": 1617716065000,
                "transCount": 1,
                "hashs":
↪"0x4c6f026e72b69e9ed1f3685f24bc055f5ef1bdde4778ee684a96d25deba6b34b"
        }],
        "totalCount": 2
}
```

13.3.18 18. 预编译权限管理

18.1.查询治理委员信息(everyone可访问)

接口描述

通过接口查询链的治理委员信息

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/cmtInfo

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/cmtInfo?
↪groupId=group0
```

响应参数

1）数据格式

a、成功：

```
[
  {
    "governorList": [
      {
        "governorAddress": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
        "weight": 1
      },
      {
        "governorAddress": "0x36c10bfbce3b6550ed92a5ebbb9a44e052bfd285",
```

(下页继续)

(续上页)

```
        "weight": 2
      }
    ],
    "participatesRate": 100,
    "winRate": 90
  }
]
```

b、失败：

```
{
  "code": 500,
  "errorMessage": "get Client failed, e: The group not exist, please check the␣
↪groupID, groupID: g3"
}
```

18.2. 查询合约管理员信息(everyone可访问)

接口描述

通过接口查询某合约的管理员信息

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/contract/admin

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractAddr": "0xB47fd49b0f1Af2Fce3a1824899b60C2b6A29B851",
  "groupId": "g1"
}
```

响应参数

1）数据格式

a、成功：

```
0x489877b18f93353c67d252c1b8f4b745d41c2107
```

b、失败1，查询群组不存在：

```
{
  "code": 500,
  "errorMessage": "get Client failed, e: The group not exist, please check the␣
↪groupID, groupID: g3"
}
```

失败2，查询合约地址错误：

```
{
  "code": 19,
  "errorMessage": "Call address error"
}
```

失败3，查询合约地址不存在：

```
0x0000000000000000000000000000000000000000
```

18.5. 查询合约函数访问权限(everyone可访问)

接口描述

通过接口查询某用户对某合约函数的访问权限

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/contract/method/auth

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractAddr": "0xB47fd49b0f1Af2Fce3a1824899b60C2b6A29B851",
  "func": "set",
  "groupId": "g1",
  "userAddress": "0x489877b18f93353c67d252c1b8f4b745d41c2107"
}
```

响应参数

1）数据格式

a、成功：

```
true
```

b、失败1，查询群组不存在：

```
{
  "code": 500,
  "errorMessage": "get Client failed, e: The group not exist, please check the␣
↪groupID, groupID: g3"
}
```

失败2，查询合约地址错误：

```
{
  "code": 19,
  "errorMessage": "Call address error"
}
```

18.6. 查询合约部署权限(everyone可访问)

接口描述

通过接口查询全局合约部署权限

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/deploy/type

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/deploy/
↪type?groupId=g1
```

响应参数

1）数据格式

a、成功，可部署：

```
0
```

b、失败1，查询群组不存在：

```
{
  "code": 500,
  "errorMessage": "get Client failed, e: The group not exist, please check the␣
↪groupID, groupID: g0"
}
```

18.7. 查询单一提案信息(everyone可访问)

接口描述

通过接口查询某个提案信息

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/proposalInfo

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId": "g1",
  "proposalId": 1
}
```

响应参数

1）数据格式

a、成功，可部署：

```
[
  {
    "resourceId": "0xc0523dbdd94ba27e14b0336d799489340ca24cdf",
    "proposer": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
    "proposalType": 31,
    "blockNumberInterval": 604809,
    "status": 2,
    "agreeVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
    ],
    "againstVoters": [],
    "statusString": "finished",
    "proposalTypeString": "resetAdmin"
  }
]
```

b、失败1，查询提案不存在：

```
[
  {
    "resourceId": "0x0000000000000000000000000000000000000000",
    "proposer": "0x0000000000000000000000000000000000000000",
    "proposalType": 0,
    "blockNumberInterval": 0,
    "status": 0,
```

(下页继续)

(续上页)

```
    "agreeVoters": [],
    "againstVoters": [],
    "statusString": "unknown",
    "proposalTypeString": "unknown"
  }
]
```

18.8. 查询提案总数(everyone可访问)

接口描述

通过接口查询某个提案信息

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/proposalInfo

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/
↪proposalInfoCount?groupId=g1
```

响应参数

1）数据格式

a、成功:

```
5
```

b、失败1，查询提案不存在:

```
{
  "code": 500,
  "errorMessage": "get Client failed, e: The group not exist, please check the␣
↪groupID, groupID: g0"
}
```

18.9. 查询提案列表(everyone可访问)

接口描述

通过接口查询某群组提案列表

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/proposalInfo

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/
↪proposalInfoList
```

响应参数

1）数据格式

a、成功:

```
[
  {
    "resourceId": "0x02a6340ef1d8a780f0ddf01dde9141cc09f678d6",
    "proposer": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
    "proposalType": "setWeight",
    "blockNumberInterval": 604825,
    "status": "notEnoughVotes",
    "agreeVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
    ],
    "againstVoters": [],
    "proposalId": 5
  },
  {
    "resourceId": "0x36c10bfbce3b6550ed92a5ebbb9a44e052bfd285",
    "proposer": "0x36c10bfbce3b6550ed92a5ebbb9a44e052bfd285",
    "proposalType": "setWeight",
    "blockNumberInterval": 604812,
    "status": "failed",
    "agreeVoters": [
      "0x36c10bfbce3b6550ed92a5ebbb9a44e052bfd285"
    ],
    "againstVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
    ],
    "proposalId": 4
  },
  {
    "resourceId": "0x0000000000000000000000000000000000010001",
    "proposer": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
    "proposalType": "setRate",
    "blockNumberInterval": 604811,
    "status": "finished",
    "agreeVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
```

(下页继续)

(续上页)


```
    ],
    "againstVoters": [],
    "proposalId": 3
  },
  {
    "resourceId": "0x36c10bfbce3b6550ed92a5ebbb9a44e052bfd285",
    "proposer": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
    "proposalType": "setWeight",
    "blockNumberInterval": 604810,
    "status": "finished",
    "agreeVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
    ],
    "againstVoters": [],
    "proposalId": 2
  },
  {
    "resourceId": "0xc0523dbdd94ba27e14b0336d799489340ca24cdf",
    "proposer": "0x015577ab8c903adcf9b65433f16e574d6daf0559",
    "proposalType": "resetAdmin",
    "blockNumberInterval": 604809,
    "status": "finished",
    "agreeVoters": [
      "0x015577ab8c903adcf9b65433f16e574d6daf0559"
    ],
    "againstVoters": [],
    "proposalId": 1
  }
]
```


18.10. 查询用户全局部署权限(everyone可访问)

接口描述

通过接口查询用户全局部署权限

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/everyone/usr/deploy

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式


```
{
  "groupId": "g1",
  "userAddress": "0x489877b18f93353c67d252c1b8f4b745d41c2107"
}
```

响应参数

1）数据格式

a、成功：

```
true
```

18.11. 设置合约的访问权限类型(admin可访问)

接口描述

通过接口设置合约的访问权限类型

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/admin/method/auth/type

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "authType": 1,
  "contractAddr": "4721d1a77e0e76851d460073e64ea06d9c104194",
  "fromAddress": "0xe88ff54644de54fa32ac845c05ed2b7d5677c078",
  "func": "set",
  "groupId": "group0",
  "signUserId": ""
}
```

响应参数

1）数据格式

a、成功：

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

18.12. 设置某用户对合约的访问权限(admin可访问)

接口描述

通过接口设置某用户对合约函数的访问权限

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/admin/method/auth/set

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractAddr": "0xB47fd49b0f1Af2Fce3a1824899b60C2b6A29B851",
  "fromAddress": "0x489877b18f93353c67d252c1b8f4b745d41c2107",
  "func": "set",
  "groupId": "g1",
  "isOpen": true,
  "signUserId": "string",
  "userAddress": "0x489877b18f93353c67d252c1b8f4b745d41c2107"
}
```

响应参数

1）数据格式

a、成功：

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

18.13. 设置某合约的管理员(committee可访问)

接口描述

通过接口设置某合约的管理员

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/committee/contract/admin

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractAddr": "4721d1a77e0e76851d460073e64ea06d9c104194",
  "fromAddress": "0x70da1da76e0e423ec582ec866fae749af67ec4c0",
  "groupId": "group0",
  "newAdmin": "0x70da1da76e0e423ec582ec866fae749af67ec4c0",
  "signUserId": ""
}
```

响应参数

1）数据格式

a、成功:

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

18.14. 设置全局部署权限类型(committee可访问)

接口描述

通过接口设置全局部署类型

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/committee/deploy/type

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "deployAuthType":1,
  "fromAddress": "0x70da1da76e0e423ec582ec866fae749af67ec4c0",
  "groupId": "group0",
  "signUserId": ""
}
```

响应参数

1）数据格式

a、成功：

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

18.15. 设置治理委员账户(committee可访问)

接口描述

通过接口设置治理委员(新增/更新/删除)

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/committee/governor

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "accountAddress": "0xe88ff54644de54fa32ac845c05ed2b7d5677c078",
  "fromAddress": "0x70da1da76e0e423ec582ec866fae749af67ec4c0",
  "groupId": "group0",
  "signUserId": "",
  "weight": 5
}
```

响应参数

1）数据格式

a、成功：

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

18.16. 设置治理阈值(committee可访问)

接口描述

通过接口设置治理阈值

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/committee/rate

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fromAddress": "0x70da1da76e0e423ec582ec866fae749af67ec4c0",
  "groupId": "group0",
  "participatesRate": 51,
  "signUserId": "",
  "winRate": 51
}
```

响应参数

1）数据格式

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

18.17. 对提案投票(committee可访问)

接口描述

通过接口设置对提案进行投票

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/committee/proposal/vote

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "agree": true,
  "fromAddress": "0x70da1da76e0e423ec582ec866fae749af67ec4c0",
  "groupId": "group0",
  "proposalId": 55,
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

1）数据格式

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

18.18.撤销提案(committee可访问)

接口描述

通过接口设置撤销某提案

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/authmanager/committee/proposal/revoke

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fromAddress": "0x70da1da76e0e423ec582ec866fae749af67ec4c0",
  "groupId": "group0",
  "proposalId": 55,
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

1）数据格式

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

13.3.19 19. 预编译合约管理

19.1. 创建BFS路径

接口描述

通过接口创建BFS

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/bfs/create

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fromAddress": "0x2abd2fc35c4553b1f1aa6cf70a4e6ef30b4d53a2",
  "groupId": "group0",
  "path": "/apps/test9",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

19.2. 查询BFS路径

接口描述

通过接口查询BFS

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/bfs/query

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId": "group0",
  "path": "/apps"
}
```

响应参数

```
[
  "test",
  "test1"
]
```

19.3. 通过contractName查询合约信息

接口描述

通过groupId和contractName查询合约信息

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/cns/queryCnsByName

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractName": "HelloWorld",
  "groupId": "group0"
}
```

响应参数

```
[
  {
    "name": "HelloWorld",
    "version": "1.0",
    "address": "4721d1a77e0e76851d460073e64ea06d9c104194",
    "abi": "[{\"inputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\
↪"constructor\"},{\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"internalType\":\
↪"string\",\"name\":\"\",\"type\":\"string\"}],\"stateMutability\":\"view\",\
↪"type\":\"function\"},{\"inputs\":[{\"internalType\":\"string\",\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"stateMutability\":\
↪"nonpayable\",\"type\":\"function\"}]"
  },
  {
    "name": "HelloWorld",
    "version": "2.0",
    "address": "4721d1a77e0e76851d460073e64ea06d9c104194",
    "abi": "[{\"inputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\
↪"constructor\"},{\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"internalType\":\
↪"string\",\"name\":\"\",\"type\":\"string\"}],\"stateMutability\":\"view\",\
↪"type\":\"function\"},{\"inputs\":[{\"internalType\":\"string\",\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"stateMutability\":\
↪"nonpayable\",\"type\":\"function\"}]"
  }
]
```

19.4. 通过contractName和version查询合约信息

接口描述

通过groupId、contractName、version查询合约信息

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/cns/queryCnsByNameVersion

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractName": "HelloWorld",
  "groupId": "group0",
  "version": "1.0"
}
```

响应参数

```
{
  "address": "0x4721d1a77e0e76851d460073e64ea06d9c104194",
  "abi": "[{\"inputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\
↪"constructor\"},{\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"internalType\":\
↪"string\",\"name\":\"\",\"type\":\"string\"}],\"stateMutability\":\"view\",\
↪"type\":\"function\"},{\"inputs\":[{\"internalType\":\"string\",\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"stateMutability\":\
↪"nonpayable\",\"type\":\"function\"}]"
}
```

19.5. 通过contractName/groupId/version查询合约地址

接口描述

通过contractName/groupId/version参数查询合约地址

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/cns/queryCnsByNameVersion

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "contractName": "HelloWorld",
  "groupId": "group0",
  "version": "1.0"
}
```

响应参数

```
0x4721d1a77e0e76851d460073e64ea06d9c104194
```

19.6. 注册合约

接口描述

通过接口注册合约信息

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/cns/reqAddressInfoByNameVersion

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "abiData": "[{\"inputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\
↪"constructor\"},{\"inputs\":[],\"name\":\"get\",\"outputs\":[{\"internalType\":\
↪"string\",\"name\":\"\",\"type\":\"string\"}],\"stateMutability\":\"view\",\
↪"type\":\"function\"},{\"inputs\":[{\"internalType\":\"string\",\"name\":\"n\",\
↪"type\":\"string\"}],\"name\":\"set\",\"outputs\":[],\"stateMutability\":\
↪"nonpayable\",\"type\":\"function\"}]",
  "contractAddress": "4721d1a77e0e76851d460073e64ea06d9c104194",
  "contractName": "HelloWorld",
  "contractVersion": "1.0",
  "fromAddress": "",
  "groupId": "group0",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

19.7. 查询共识节点列表

接口描述

通过接口查询共识节点列表

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/consensus/list

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId": "group0",
  "pageNumber": 1,
  "pageSize": 5
}
```

响应参数

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "nodeId":
↪"6447e978505cafd05fc99b731d8fdff31fb07a3c6e9679054fb1880ae6f58aeff638eacfe082d54adca93086c2986b
↪",
      "nodeType": "sealer",
      "weight": 1
    },
    {
      "nodeId":
↪"b14bd4a225db308da3f395c69f12ce06f191ff19941d52eebf30cfb5fc979422ad086fedb0378fdcfbcb4630416e71
↪",
      "nodeType": "sealer",
      "weight": 1
    },
    {
      "nodeId":
↪"848883c435d5c7e32da7744ffb0659538995994a42c24ec7da81a2fd58cd28e76fbaaf603b81f9134d22f57d112cdb
↪",
      "nodeType": "sealer",
      "weight": 1
    },
    {
      "nodeId":
↪"5007b294c7aadd22d62e0c5e33bae14ee6ec0230ebd34df23f29f0330272f6021fd3a8f2b4a4789f1e2fe7fbc8581c
↪",
      "nodeType": "sealer",
      "weight": 1
    }
  ],
  "totalCount": 4
}
```

19.8. 修改共识节点类型

接口描述

通过接口查询共识节点列表

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/consensus/manage

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fromAddress": "0x2abd2fc35c4553b1f1aa6cf70a4e6ef30b4d53a2",
  "groupId": "group0",
  "nodeId":
↪"5007b294c7aadd22d62e0c5e33bae14ee6ec0230ebd34df23f29f0330272f6021fd3a8f2b4a4789f1e2fe7fbc8581c
↪",
  "nodeType": "observer",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31",
  "weight": 1
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

19.9. 建表

接口描述

通过接口插入新的表结构

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/kvtable/reqCreateTable

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fromAddress": "0x2abd2fc35c4553b1f1aa6cf70a4e6ef30b4d53a2",
  "groupId": "group0",
  "keyFieldName": "myKey",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31",
  "tableName": "test_table",
  "valueFields": [
    "valueIsData"
  ]
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

19.10. 写表

接口描述

通过接口在表插入数据

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/kvtable/reqSetTable

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "fieldNameToValue": {
    "key1": "hi",
    "key2": "hello",
    "key3": "how are u"
  },
  "fromAddress": "0x2abd2fc35c4553b1f1aa6cf70a4e6ef30b4d53a2",
  "groupId": "group0",
  "key": "myKey",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31",
  "tableName": "test_table"
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

19.11. 读表

接口描述

通过接口在表读取数据

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/kvtable/reqGetTable

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId": "group0",
  "key": "myKey",
  "tableName": "test_table"
}
```

响应参数

```
{
  "key2": "hello",
  "key1": "hi",
  "key3": "how are u"
}
```

19.12. 获取群组系统配置

接口描述

通过接口读取某个群组的系统配置

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/sys/config/list

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5001/WeBASE-Node-Manager/precntauth/precompiled/sys/config/list?
↪groupId=group0
```

响应参数

```
[
  {
    "groupId": "group0",
    "configKey": "tx_count_limit",
    "configValue": "10"
  },
  {
    "groupId": "group0",
    "configKey": "tx_gas_limit",
    "configValue": "300000002"
  }
]
```

19.13. 设置群组系统配置

接口描述

通过接口设置某个群组的系统配置

接口URL

http://localhost:5001/WeBASE-Node-Manager/precntauth/precompiled/sys/config/list

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "configKey": "tx_count_limit",
  "configValue": "5",
  "fromAddress": "0x2abd2fc35c4553b1f1aa6cf70a4e6ef30b4d53a2",
  "groupId": "group0",
  "signUserId": "5db5a98aef544650aa3864f4cb27af31"
}
```

响应参数

```
{
  "code" : 0,
  "message" : "success",
  "data" : "Success"
}
```

13.3.20 附录

1. 返回码信息列表

X01XXX为WeBASE-Front节点前置错误码，X02XXX为WeBASE-Node-Manager节点管理服务错误码，X03XXX为WeBASE-Sign签名服务错误码。

2. Precompiled Service说明

对预编译合约接口的使用有疑惑，可以查看FISCO BCOS的PreCompiledService API说明

查看预编译合约的solidity接口代码，可以查看FISCO BCOS的web3sdk precompile模块，如crud/TableFactory.sol:

```
pragma solidity ^0.4.2;

contract TableFactory {
    function createTable(string tableName, string key, string valueField) public␣
↪returns (int);
}
```

查看FISCO BCOS中实现的precompild合约列表、地址分配及源码：

Precompiled Service API 错误码

13.4 升级说明

WeBASE-Node-Manager升级的兼容性说明，请结合WeBASE-Node-Manager Changelog进行阅读

WeBASE-Node-Manager升级的必须步骤:

1. 备份已有文件或数据，下载新的安装包（可参考安装包下载）
2. 使用新的安装包，并将旧版本yml已有配置添加到新版本yml中；可通过`diff aFile bFile`命令对比新旧yml的差异
3. 查看节点管理服务升级文档中对应版本是否需要修改数据表，若不需要升级则跳过 3.1 若需要升级数据表，首先使用`mysqldump`命令备份数据库 3.2 按照升级文档指引，操作数据表
4. `bash stop.sh && bash start.sh`重启

13.5 附录

13.5.1 1. 安装问题

1.1 Java部署

CentOS环境安装Java

注意：CentOS下OpenJDK无法正常工作，需要安装OracleJDK下载链接。

```
# 创建新的文件夹，安装Java 8或以上的版本，将下载的jdk放在software目录
# 从Oracle官网(https://www.oracle.com/technetwork/java/javase/downloads/index.html)选择Java 8或以上的版本下载，例如下载jdk-8u201-linux-x64.tar.gz
$ mkdir /software

# 解压jdk
$ tar -zxvf jdk-8u201-linux-x64.tar.gz

# 配置Java环境，编辑/etc/profile文件
$ vim /etc/profile

# 打开以后将下面三句输入到文件里面并保存退出
export JAVA_HOME=/software/jdk-8u201  #这是一个文件目录，非文件
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar

# 生效profile
$ source /etc/profile

# 查询Java版本，出现的版本是自己下载的版本，则安装成功。
java -version
```

Ubuntu环境安装Java

```
  # 安装默认Java版本(Java 8或以上)
  sudo apt install -y default-jdk
  # 查询Java版本
  java -version
```

1.2 Gradle部署

此处给出简单步骤，供快速查阅。更详细的步骤，请参考官网。

（1）从官网下载对应版本的Gradle安装包，并解压到相应目录

```
mkdir /software/
unzip -d /software/ gradleXXX.zip
```

（2）配置环境变量

```
export GRADLE_HOME=/software/gradle-4.9
export PATH=$GRADLE_HOME/bin:$PATH
```

（3）查看版本

```
gradle -version
```

13.5.2 2. 常见问题及方案

一般问题

- 问：执行shell脚本报下面错误：

```
[app@VM_96_107_centos deployInputParam]$ bash start.sh
start.sh: line 2: $'\r': command not found
start.sh: line 8: $'\r': command not found
start.sh: line 9: $'\r': command not found
start.sh: line 10: $'\r': command not found
```

答：这是编码问题，在脚本的目录下执行转码命令：

```
dos2unix *.sh
```

数据库问题

- 问：服务访问数据库抛出异常：

```
The last packet sent successfully to the server was 0 milliseconds ago. The driver␣
↪has not received any packets from the server.
```

答：检查数据库的网络策略是否开通

```
下面以centos7为例：
查看防火墙是否开放3306端口：  firewall-cmd --query-port=3306/tcp
防火墙永久开放3306端口：firewall-cmd --zone=public --add-port=3306/tcp --permanent
重新启动防火墙：firewall-cmd --reload
```

- 问：执行数据库初始化脚本抛出异常：

```
ERROR 2003 (HY000): Can't connect to MySQL server on '127.0.0.1' (110)
```

答：MySQL没有开通该帐号的远程访问权限，登录MySQL，执行如下命令，其中TestUser改为你的帐号

```
GRANT ALL PRIVILEGES ON *.* TO 'TestUser'@'%' IDENTIFIED BY '此处为TestUser的密码' ␣
↪WITH GRANT OPTION;
```

WeBASE-Node-Manager服务搭建问题

- 问：执行构建命令`gradle build -x test`抛出异常：

```
A problem occurred evaluating root project 'WeBASE-Node-Manager'.
Could not find method compileOnly() for arguments [[org.projectlombok:lombok:1.18.
↪2]] on root project 'WeBASE-Node-Manager'.
```

答： 方法1、已安装的Gradle版本过低，升级Gradle版本到4.10以上即可。 方法2、直接使用命令：`./gradlew build -x test`，如果提示gradlew为非可执行文件，执行`chmod +x ./gradlew`再次执行build操作即可。

全量交易/全量合约/交易审计出现系统异常问题

- 问：在WeBASE页面进入合约管理/合约列表/全量或者交易审计/异常用户界面时，发生`WeBASE-Node-Manager`系统异常：

答：检查日志发现报错是由于数据库版本过高，需要禁用only_full_group_by。

登陆数据库 `mysql -u root -p` 返回:`Enter password:`

需要输入配置数据库时设置的密码。

然后查询数据库开启的规则:

```
SELECT @@GLOBAL.sql_mode;
SELECT @@SESSION.sql_mode;
```

返回如下:

```
STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO, NO_
↪AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION,ONLY_FULL_GROUP_BY
```

需要关闭其中的ONLY_FULL_GROUP_BY规则:

```
set @@GLOBAL.sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_
↪DIVISION_BY_ZERO, NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION';
set @@SESSION.sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_
↪DIVISION_BY_ZERO, NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION';
```

引号中为之前查询规则结果中除了ONLY_FULL_GROUP_BY以外的规则

若设置时报错，只需将引号中报错的规则删去并重新执行命令即可，然后重启WeBASE便可解决异常。

启动问题

- 问：启动Node-Manager进程后，后台日志显示`not found any front`：

答：此处为正常提示，表示后台没有可访问的节点前置。通过WeBASE-Web连接Node-Manager后台服务后，添加节点前置即可。

节点管理服务忘记密码

登录到WeBASE-Node-Manager中配置的Mysql数据库（默认为webasenodemanager）后，在`tb_account_info`中插入一条新的管理员账号`test`，密码`Abcd1234`

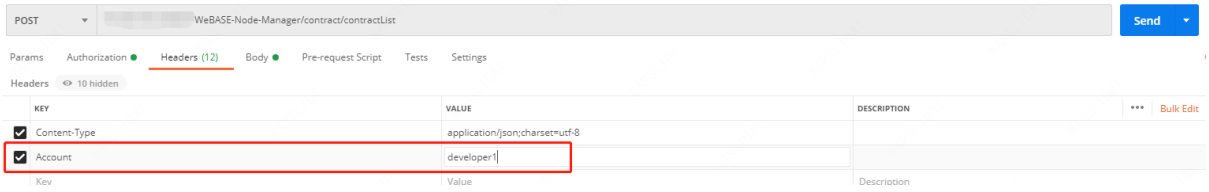
```
INSERT INTO tb_account_info (account,account_pwd,role_id,create_time,modify_
↪time)values('test', '$2a$10$F/aEB1iEx/FvVh0fMn6L/uyy.PkpTy8Kd9EdbqLGo7Bw7eCivpq.m
↪',100000,now(),now());
```

关闭鉴权调用（联调）

在application.yml中 配 置constant.isUseSecurity为false即 可 禁 用WeBASE-Node-Manager的登录鉴权。

```
#constants
constant:
  ###http request
  # login's authorization whether enable, if false, default login as `admin` account
  isUseSecurity: false
```

- 免鉴权后，默认使用的是管理员用户admin（管理员用户可以看到所有用户的数据），可以跳过登陆页面，直接访问WeBASE-Web节点管理台的主页，如http://localhost:5000/#/home。
- 若需要指定用户进行接口调用，可以在请求的headers中增加Account字段，其值设置为节点管理服务的用户名，如获取开发者用户developer1对应数据（开发者用户只能看到自己所创建的数据）。



使用swagger

v1.5.2后，节点管理服务搭配了swagger，可用于直接调试接口，需要关闭鉴权后，访问 {ip}:5001/WeBASE-Node-Manager/swagger-ui.html 即可访问swagger页面

13.5.3 3. 配置文件解析

CHAPTER 14

WeBASE管理平台

14.1 概要介绍

14.1.1 功能说明

本项目是区块链中间件平台WeBASE管理平台，使用框架`vue-cli`。

支持FISCO-BCOS 3.0以上版本，支持群组和群组切换。具体功能有：

1. **区块链数据概览**，可以查看区块链的节点、区块、交易、合约信息。点击左上角交易信息和区块信息界面，可以跳转到区块或交易信息列表页，交易信息支持input解码和event解码。
2. **节点管理**，可以查看前置列表、节点列表、修改节点共识状态。可以查看链上的所有群组和节点，查看前置所在服务器状态相关信息，管理节点的共识状态。
3. **合约管理**，提供图形化合约IDE、查询已部署合约列表、合约CNS查询以及预编译合约的CRUD功能。编译、部署合约后该合约会被保存。
4. **私钥管理**，管理所有可以发交易的帐号，公钥用户是其他机构的帐号，无法在本机构发交易，可以通过手动绑定和自动同步获取。私钥用户为本机构发交易的用户。
5. **系统管理**，提供权限管理、系统配置管理、证书管理的功能。权限管理可以控制私钥用户的权限范围，证书管理可以查看链的相关证书。
6. **系统监控**，系统监控包含了节点监控、主机监控与异常告警。监控整条链所有机构所有用户发送交易行为，查看是否有异常用户和异常合约，并在异常状态下通过告警邮件通知运维管理员。
7. **交易审计**，主要监控整条链所有机构所有用户发送交易行为，查看是否有异常用户和异常合约。
8. **帐号管理**，只有admin帐号才能查看此功能，可以新增帐号（登录此系统帐号）、修改密码、修改账户邮箱等等。

14.1.2 国密支持

WeBASE-Web v1.2.2+已支持国密功能，使用WeBASE-Node-Manager v1.2.2，使用WeBASE-Front v1.2.2及以上版本

WeBASE-Web将根据WeBASE-Node-Manager的版本，自动在国密与非国密之间切换，合约编译、合约部署、调用合约、发送交易等功能均已支持国密。

14.1.3 solidity v0.5.2、v0.6.10、v0.8.11支持

WeBASE-Web v1.4.2+已支持solidity `v0.4.25`、`v0.5.2`、`v0.6.10`和`v0.8.11`，可在合约IDE中的左上角进行版本切换

14.2 部署说明

14.2.1 1. 依赖环境

14.2.2 2. 拉取代码

代码可以放在/data下面，执行命令：

```
git clone -b master-3.0 https://github.com/WeBankBlockchain/WeBASE-Web.git

# 若网络问题导致长时间无法下载，可尝试以下命令
git clone -b master-3.0 https://gitee.com/WeBank/WeBASE-Web.git
```

进入目录：

```
cd WeBASE-Web
```

2.1 下载solc-bin

solidity v0.8.11支持

WeBASE-Web v3.0.1已支持solidity `v0.8.11`

执行脚本get_solc_js.sh会自动下载solc-bin，即下面v0.4.25.js等文件。 在`WeBASE-Web/`目录中直接执行脚本get_solc_js.sh（（脚本与`dist`文件夹同级））

```
    bash ./get_solc_js.sh
```

等待脚本执行完成

- 如果执行不成功，请使用下面的命令：

注意：当且仅当`get_solc_js.sh`脚本执行失败才需要执行下面的命令

```
    curl -#L https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/download/
↪solidity/wasm/v0.4.25.js -o ./dist/static/js/v0.4.25.js
    curl -#L https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/download/
↪solidity/wasm/v0.4.25-gm.js -o ./dist/static/js/v0.4.25-gm.js
    curl -#L https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/download/
↪solidity/wasm/v0.5.2.js -o ./dist/static/js/v0.5.2.js
    curl -#L https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/download/
↪solidity/wasm/v0.5.2-gm.js -o ./dist/static/js/v0.5.2-gm.js
    curl -#L https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/download/
↪solidity/wasm/v0.6.10.js -o ./dist/static/js/v0.6.10.js
    curl -#L https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/download/
↪solidity/wasm/v0.6.10-gm.js -o ./dist/static/js/v0.6.10-gm.js
    curl -#L https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/download/
↪solidity/wasm/v0.8.11.js -o ./dist/static/js/v0.8.11.js
    curl -#L https://osp-1257653870.cos.ap-guangzhou.myqcloud.com/WeBASE/download/
↪solidity/wasm/v0.8.11-gm.js -o ./dist/static/js/v0.8.11-gm.js
```

执行完后检查dist/static/js是否下载完这些js文件。

14.2.3 4. 修改配置

在docs目录下有配置文件nginx.conf，修改完后替换安装的nginx的配置文件nginx.conf（这里nginx安装配置文件在/usr/local/nginx/conf下面，如果这里没找到，可以到/etc下寻找，如有权限问题，请加上sudo）。

- 修改配置：

```
# 修改服务器ip，也可以使用域名
sed -i "s%127.0.0.1%${your_ip}%g" docs/nginx.conf

# 修改WeBASE-Web服务端口（端口需要开通策略且不能被占用）
sed -i "s%5000%${your_port}%g" docs/nginx.conf

# 修改静态文件路径（文件需要有权限访问）
sed -i "s%/data/WeBASE-Web/dist%${your_file_dir}%g" docs/nginx.conf

# 节点管理服务ip和端口
sed -i "s%10.0.0.1:5001%${your_node_manager}%g" docs/nginx.conf
```

- 复制配置文件nginx.conf到默认配置目录中
- 也可以直接通过`nginx -c docs/nginx.conf`命令加载docs/nginx.conf配置

```
cp -rf docs/nginx.conf /usr/local/nginx/conf
```

备注： 如果服务器已有nginx，可在原配置文件nginx.conf增加一个server：

```
    upstream node_mgr_server{
        server 10.0.0.1:5001; # 节点管理服务ip和端口
    }
    server {
        listen       5000 default_server; # 前端端口（端口需要开通策略且不能被占用）
        server_name  127.0.0.1;           # 服务器ip，也可配置为域名
        location / {
            root   /data/WeBASE-Web/dist;    # 前端文件路径(文件需要有权限访问)
            index  index.html index.htm;
            try_files $uri $uri/ /index.html =404;
        }

        include /etc/nginx/default.d/*.conf;

        location /mgr {
            proxy_pass    http://node_mgr_server/;
            proxy_set_header                Host                        $host;
            proxy_set_header                X-Real-IP                 $remote_addr;
            proxy_set_header                X-Forwarded-For         $proxy_add_x_
↪forwarded_for;
        }
    }
```

14.2.4 5. 启动nginx

启动命令：

```
/usr/local/nginx/sbin/nginx # nginx在/usr/local目录下
```

检查nginx是否启动：

```
ps -ef | grep nginx
```

14.2.5 6. 访问页面

```
http://{deployIP}:{webPort}
示例：http://127.0.0.1:5000
```

备注：

- 部署服务器IP和管理平台服务端口需对应修改，网络策略需开通
- 默认账号密码：admin/Abcd1234
- WeBASE管理平台使用说明请查看使用手册

14.2.6 7. 查看日志

```
进程日志：tail -f logs/access.log
错误日志：tail -f logs/eror.log
```

14.3 附录

14.3.1 1 安装nginx

1.1 下载nginx依赖

在安装nginx前首先要确认系统中安装了gcc、pcre-devel、zlib-devel、openssl-devel。如果没有，请执行命令

```
yum -y install gcc pcre-devel zlib-devel openssl openssl-devel
```

执行命令时注意权限问题，如遇到，请加上sudo

1.2 下载nginx

nginx下载地址：https://nginx.org/download/（下载最新稳定版本即可） 或者使用命令：

```
wget http://nginx.org/download/nginx-1.9.9.tar.gz  (版本号可换)
```

将下载的包移动到/usr/local/下

1.3 安装nginx

1.3.1 解压

```
tar -zxvf nginx-1.9.9.tar.gz
```

1.3.2 进入nginx目录

```
cd nginx-1.9.9
```

1.3.3 配置

```
./configure --prefix=/usr/local/nginx
```

1.3.4 make

```
make
make install
```

1.3.5 测试是否安装成功

使用命令：

```
/usr/local/nginx/sbin/nginx -t
```

正常情况的信息输出：

```
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
```

1.3.6 nginx几个常见命令

```
/usr/local/nginx/sbin/nginx -s reload            # 重新载入配置文件
/usr/local/nginx/sbin/nginx -s reopen            # 重启 Nginx
/usr/local/nginx/sbin/nginx -s stop              # 停止 Nginx
ps -ef | grep nginx                              # 查看nginx进程
```

14.3.2 2 常见问题

2.1 出现“登录错误”怎么排查问题

登录时出现“登录错误”，请一一排查：

1. WeBASE-Node-Manager服务是否启动成功，
2. WeBASE-Node-Manager的数据库是否正常，
3. nginx代理是否存在错误。

2.2 登录页面的验证码加载不出来

- 进入 webase-node-mgr 目录下，执行 bash status.sh 检查服务是否启动，如果服务没有启动，运行 bash start.sh 启动服务；
- 如果服务已经启动，按照如下修改日志级别
  - webase-node-mgr/conf/application.yml

```
#log config
logging:
  level:
    com.webank.webase.node.mgr: debug
```

  - webase-node-mgr/conf/log/log4j2.xml

```
<Loggers>
<Root level="debug">
  <AppenderRef ref="asyncInfo"/>
  <AppenderRef ref="asyncErrorLog"/>
</Root>
</Loggers>
```

- 重启服务 `bash stop.sh && bash start.sh`
- 重启服务后，检查日志文件 `log/WeBASE-Node-Manager.log`。
  - 检查是否有异常信息。如果有异常信息，根据具体的异常信息检查环境配置，或者通过搜索引擎进行排查。

2.3 为什么输入正确的验证码显示验证码错误

登录验证码有效时间为五分钟，五分钟后验证码失效，登录会出现“验证码错误”。

2.4 交易解码解不出来

将该交易所属的合约上传到合约管理，并编译一次，下一笔调用合约的交易触发后即可成功解码。

2.5 交易审计异常交易和异常合约怎么消除

- 将发送交易的账户在私钥管理中添加成公钥用户，那么该用户所发的交易将审计成正常交易；
- 将部署该合约的账户在私钥管理中添加成公钥用户，那么该用户所部署的合约将审计成正常合约。

14.3.3 3. 二次开发

开发文档

CHAPTER 15

签名服务

15.1 概要介绍

15.1.1 功能介绍

本系统为签名服务子系统。功能：管理公私钥、对数据进行签名。

15.1.2 国密支持

WeBASE-Sign v1.2.2+已支持 国密版FISCO-BCOS，使用WeBASE v1.2.2及以上版本

本系统同时支持国密与非国密，分别提供了ECDSA与国密两类接口，可生成/管理ECDSA和国密公私钥用户，可对数据进行国密或非国密的签名

15.2 部署说明

15.2.1 1. 前提条件

备注：安装说明请参看 附录-1。

15.2.2 2. 拉取代码

执行命令：

```
git clone -b master-3.0 https://github.com/WeBankBlockchain/WeBASE-Sign.git

# 若因网络问题导致长时间下载失败，可尝试以下命令
git clone -b master-3.0 https://gitee.com/WeBank/WeBASE-Sign.git
```

进入目录：

```
cd WeBASE-Sign
```

15.2.3 3. 编译代码

方式一：如果服务器已安装Gradle，且版本为gradle-4.10至gradle-6.x版本

```
gradle build -x test
```

方式二：如果服务器未安装Gradle，或者版本不是gradle-4.10至gradle-6.x版本，使用gradlew编译

```
chmod +x ./gradlew && ./gradlew build -x test
```

构建完成后，会在根目录WeBASE-Sign下生成已编译的代码目录dist。

15.2.4 4. 数据库初始化

```
#登录MySQL:
mysql -u ${your_db_account} -p${your_db_password}  例如: mysql -u root -p123456
#新建数据库:
CREATE DATABASE IF NOT EXISTS {your_db_name} DEFAULT CHARSET utf8 COLLATE utf8_
↪general_ci;
```

15.2.5 5. 修改配置

（1）进入dist目录

```
cd dist
```

dist目录提供了一份配置模板conf_template:

```
根据配置模板生成一份实际配置conf。初次部署可直接拷贝。
例如: cp conf_template conf -r
```

（2）修改配置（根据实际情况修改）：

```
vi conf/application.yml
```

```
server:
  # 本工程服务端口，端口被占用则修改
  port: 5004
  context-path: /WeBASE-Sign

spring:
  datasource:
    # 数据库连接信息
    url: jdbc:mysql://127.0.0.1:3306/webasesign?serverTimezone=GMT%2B8&
↪useUnicode=true&characterEncoding=utf8
    # 数据库用户名
    username: "dbUsername"
    # 数据库密码
    password: "dbPassword"
    driver-class-name: com.mysql.cj.jdbc.Driver

constant:
  # aes加密key(16位)，如启用，各互联的子系统的加密key需保持一致
  aesKey: EfdsW23D23d3df43
```

15.2.6 5. 服务启停

在dist目录下执行：

```
启动：bash start.sh
停止：bash stop.sh
检查：bash status.sh
```

备注：服务进程起来后，需通过日志确认是否正常启动，出现以下内容表示正常；如果服务出现异常，确认修改配置后，重启提示服务进程在运行，则先执行stop.sh，再执行start.sh。

```
...
        Application() - main run success...
```

15.2.7 6. 查看日志

在dist目录查看：

```
全量日志：tail -f log/WeBASE-Sign.log
错误日志：tail -f log/WeBASE-Sign-error.log
```

15.3 接口说明

15.3.1 1. 新增用户接口

1.1. 新增ECDSA/国密用户接口

接口描述

根据传入的encryptType值，新增ECDSA或国密公私钥用户。

接口URL

http://localhost:5004/WeBASE-Sign/user/newUser?signUserId={signUserId}&appId={appId}&encryptType={encryptType}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5004/WeBASE-Sign/user/newUser?signUserId={signUserId}&appId=
↪{appId}&encryptType=0
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

ECDSA用户：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "signUserId": "user_111",
        "appId": "group_01",
        "address": "0x2df87ff79e8c85a318c00c82ee76e2581fbab0a8",
        "publicKey":
↪"0x1befc9824623dfc2f1541d2fc1df4bc445d9dd26816b0884e24628881d5bb572bf7dfd69520d540adc2d16d295df
↪",
        "privateKey": "",
        "description": null,
        "encryptType": 0
    }
}
```

国密用户：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "signUserId": "user_222",
        "appId": "group_02",
        "address": "0x0bc3465986845864fc1646dedf2dd892c0fe11be",
        "publicKey":
↪"0xd09d4efe3c127898186c197ae6004a9b40d7c7805fc7e31f7c4a835a4b9cf4148155cbd6dfcf3e5fd84acf1ea55c
↪",
        "privateKey": "",
        "description": null,
        "encryptType": 1
    }
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
    "code": 303001,
    "message": "user of this sign user id is already exists",
    "data": null
}
```

1.2. 导入私钥用户接口

接口描述

导入私钥到Sign，与新增私钥类似

接口URL

http://localhost:5004/WeBASE-Sign/user/newUser

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
http://localhost:5004/WeBASE-Sign/user/newUser
```

```
{
    //privateKey编码前原文为:␣
↪3d1a470b2e7ae9d536c69af1cc5edf7830ece5b6a97df0e9441bab9f7a77b131
  "privateKey":
↪"M2QxYTQ3MGIyZTdhZTlkNTM2YzY5YWYxY2M1ZWRmNzgzMGVjZTViNmE5N2RmMGU5NDQxYmFiOWY3YTc3YjEzMQ==
↪",
  "signUserId": "user_222",
  "appId": "app_222",
  "encryptType": 0
}
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

ECDSA用户：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "signUserId": "user_111",
        "appId": "group_01",
        "address": "0x2df87ff79e8c85a318c00c82ee76e2581fbab0a8",
        "publicKey":
↪"0x1befc9824623dfc2f1541d2fc1df4bc445d9dd26816b0884e24628881d5bb572bf7dfd69520d540adc2d16d295df
↪",
        "privateKey": "", //不返回私钥
        "description": null,
        "encryptType": 0
    }
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
    "code": 303001,
    "message": "user of this sign user id is already exists",
```

(下页继续)

(续上页)

```
    "data": null
}
```

15.3.2 2. 查询用户接口

2.1 根据userId查询用户

接口描述

根据用户编号查询用户信息。

接口URL

http://localhost:5004/WeBASE-Sign/user/{signUserId}/userInfo

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5004/WeBASE-Sign/user/{signUserId}/userInfo
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

ECDSA用户：

```
{
    "code": 0,
    "message": "success",
    "data": {
        "signUserId": "user_111",
        "appId": "group_01",
        "address": "0x2df87ff79e8c85a318c00c82ee76e2581fbab0a8",
        "publicKey":
↪"0x1befc9824623dfc2f1541d2fc1df4bc445d9dd26816b0884e24628881d5bb572bf7dfd69520d540adc2d16d295df
↪",
        "privateKey": "",
        "description": null,
        "encryptType": 0
    }
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
    "code": 303002,
    "message": "user does not exist",
    "data": null
}
```

15.3.3 3. 私钥用户管理接口

3.1. 停用私钥用户

接口描述

通过修改私钥用户的status状态值来停用私钥用户；停用后，其他接口将不返回被停用的私钥用户

接口URL

http://localhost:5004/WeBASE-Sign/user

调用方法

HTTP DELETE

请求参数

1）参数表

2）数据格式

```
http://localhost:5004/WeBASE-Sign/user
```

```
{
  "signUserId": "user_111"
}
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
    "code": 0,
    "message": "success"
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
    "code": 303002,
    "message": "user does not exist",
    "data": null
}
```

3.2. 清除私钥用户缓存

接口描述

私钥用户的缓存用于缓存私钥数据到内存中，提高私钥签名效率；此接口可删除所有用户缓存信息

接口URL

http://localhost:5004/WeBASE-Sign/user/all

调用方法

HTTP DELETE

请求参数

1）参数表

2）数据格式

```
http://localhost:5004/WeBASE-Sign/user/all
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
    "code": 0,
    "message": "success"
}
```

15.3.4 4. 用户列表接口

4.1. 根据appId查询用户列表（分页）

接口描述

根据传入的appId值，查询所有所有属于该appId的用户信息列表。

接口URL

http://localhost:5004/WeBASE-Sign/user/list/{appId}/{pageNumber}/{pageSize}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5004/WeBASE-Sign/user/list/group_01/1/5
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

ECDSA用户列表:

```
{
    "code": 0,
    "message": "success",
    "data": [
        {
            "signUserId": "user_111",
            "appId": "group_01",
            "address": "0x2df87ff79e8c85a318c00c82ee76e2581fbab0a8",
            "publicKey":
↪"0x1befc9824623dfc2f1541d2fc1df4bc445d9dd26816b0884e24628881d5bb572bf7dfd69520d540adc2d16d295df
↪",
            "privateKey": "",
            "description": null,
            "encryptType": 0
        }
    ],
    "totalCount": 1
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

15.3.5 5. 数据签名接口

5.1. ECDSA/国密数据签名接口

接口描述

指定用户通过ECDSA/国密SM2对数据进行签名。

接口URL

http://localhost:5004/WeBASE-Sign/sign

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
http://localhost:5004/WeBASE-Sign/sign
```

```
{
  "signUserId": "user_111",
  "encodedDataStr": "0xba001"
}
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
    "code": 0,
    "message": "success",
    "data": {
        "signDataStr":
↪"1c3f59a48593b66de4c57fe99f9c429811aa2dc9b495823cd99faa3e72b4a4d02e04bb7c3da6390a17adc00b0e7402
↪"
    }
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
    "code": 203009,
    "message": "encoded data string must be hex string",
    "data": null
}
```

5.1. ECDSA/国密对交易体哈希签名接口

接口描述

指定用户通过ECDSA/国密SM2对交易体的哈希进行签名。

接口URL

http://localhost:5004/WeBASE-Sign/sign/hash

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
http://localhost:5004/WeBASE-Sign/sign/hash
```

```
{
  "signUserId": "user_111",
  "encodedDataStr":
↪"0xa665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3"
}
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
    "code": 0,
    "message": "success",
    "data": {
        "signDataStr":
↪"1c3f59a48593b66de4c57fe99f9c429811aa2dc9b495823cd99faa3e72b4a4d02e04bb7c3da6390a17adc00b0e7402
↪"
    }
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
    "code": 203009,
    "message": "encoded data string must be hex string",
    "data": null
}
```

15.3.6 6. 其他接口

6.1. 查询WeBASE-Sign版本接口

接口描述

获取WeBASE-Sign的版本号

接口URL

http://localhost:5004/WeBASE-Sign/version

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5004/WeBASE-Sign/version
```

响应参数

1）数据格式

a、成功：

```
v1.4.0
```

15.3.7 附录

1. 返回码信息列表

15.4 升级说明

WeBASE-Sign升级的兼容性说明，请结合WeBASE-Sign Changelog进行阅读

WeBASE-Sign升级的必须步骤：

1. 备份已有文件或数据，下载新的安装包（可参考安装包下载）
2. 使用新的安装包，并将旧版本yml已有配置添加到新版本yml中；可通过`diff aFile bFile`命令对比新旧yml的差异
3. 查看签名服务升级文档中对应版本是否需要修改数据表，若不需要升级则跳过 3.1 若需要升级数据表，首先使用`mysqldump`命令备份数据库 3.2 按照升级文档指引，操作数据表
4. `bash stop.sh && bash start.sh`重启

15.5 附录

15.5.1 1. 安装问题

1.1 Java部署

此处给出简单步骤，供快速查阅。更详细的步骤，请参考官网。

① 安装包下载

从官网下载对应版本的java安装包，并解压到服务器相关目录

```
mkdir /software
tar -zxvf jdkXXX.tar.gz /software/
```

② 配置环境变量

- 修改/etc/profile

```
sudo vi /etc/profile
```

- 在/etc/profile末尾添加以下信息

```
JAVA_HOME=/nemo/jdk1.8.0_181
PATH=$PATH:$JAVA_HOME/bin
CLASSPATH==.:$JAVA_HOME/lib
export JAVA_HOME CLASSPATH PATH
```

- 重载/etc/profile

```
source /etc/profile
```

③ 查看版本

```
java -version
```

1.2. 数据库部署

此处以Centos安装*MariaDB*为例。*MariaDB*数据库是 MySQL 的一个分支，主要由开源社区在维护，采用 GPL 授权许可。*MariaDB*完全兼容 MySQL，包括API和命令行。其他安装方式请参考MySQL官网。

① 安装MariaDB

- 安装命令

```
sudo yum install -y mariadb*
```

- 启停

```
启动: sudo systemctl start mariadb.service
停止: sudo systemctl stop  mariadb.service
```

- 设置开机启动

```
sudo systemctl enable mariadb.service
```

- 初始化

```
执行以下命令:
sudo mysql_secure_installation
以下根据提示输入:
Enter current password for root (enter for none):<-初次运行直接回车
Set root password? [Y/n] <- 是否设置root用户密码，输入y并回车或直接回车
New password: <- 设置root用户的密码
Re-enter new password: <- 再输入一次你设置的密码
Remove anonymous users? [Y/n] <- 是否删除匿名用户，回车
```

(下页继续)

(续上页)

```
Disallow root login remotely? [Y/n] <-是否禁止root远程登录，回车
Remove test database and access to it? [Y/n] <- 是否删除test数据库，回车
Reload privilege tables now? [Y/n] <- 是否重新加载权限表，回车
```

② 授权访问和添加用户

- 使用root用户登录，密码为初始化设置的密码

```
mysql -uroot -p -h localhost -P 3306
```

- 授权root用户远程访问

```
mysql > GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '123456' WITH␣
↪GRANT OPTION;
mysql > flush PRIVILEGES;
```

- 创建test用户并授权本地访问

```
mysql > GRANT ALL PRIVILEGES ON *.* TO 'test'@localhost IDENTIFIED BY '123456'␣
↪WITH GRANT OPTION;
mysql > flush PRIVILEGES;
```

安全温馨提示:

- 例子中给出的数据库密码（123456）仅为样例，强烈建议设置成复杂密码
- 例子中root用户的远程授权设置会使数据库在所有网络上都可以访问，请按具体的网络拓扑和权限控制情况，设置网络和权限帐号

③ 测试连接和创建数据库

- 登录数据库

```
mysql -utest -p123456 -h localhost -P 3306
```

- 创建数据库

```
mysql > create database webasesign;
```

15.5.2 2. 常见问题

2.1 脚本没权限

- 执行shell脚本报错误”permission denied”或格式错误

```
赋权限：chmod + *.sh
转格式：dos2unix *.sh
```

15.5.3 3. 配置文件解析

- 1. 配置文件解析

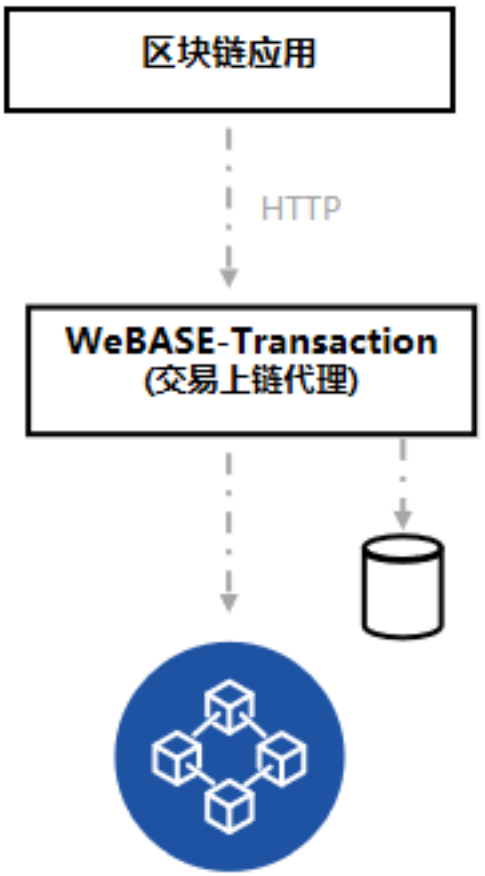
CHAPTER 16

交易服务

16.1 概要介绍

16.1.1 功能介绍

本系统为交易上链代理子系统。主要接收无状态交易请求，缓存到数据库中，再异步上链。本系统可大幅提升吞吐量，解决区块链的tps瓶颈。



架构图

主要功能：合约编译；交易请求处理，交易分为合约部署和普通的合约调用请求。

合约编译：上传合约文件zip压缩包（压缩包里的每个合约的文件名要和合约名一致，合约引用需使用“./xxx.sol”），返回合约编译信息。

合约部署：交易服务子系统会将合约部署请求信息缓存到数据库，通过轮询服务向节点发送交易请求，确保合约成功部署。

合约调用：分为无状态交易上链（非constant方法）和交易结果查询（constant方法）。无状态交易上链是交易服务子系统会将交易请求信息缓存到数据库，通过轮询服务向节点发送交易请求，确保交易成功上链。交易结果查询是交易服务子系统会同步向节点发送交易请求，返回结果。

交易上链数据签名支持以下三种模式：

- 本地配置私钥签名
- 本地随机私钥签名
- 调用WeBASE-Sign进行签名

本工程支持单机部署，也支持分布式任务多活部署（使用分布式任务的话需部署Zookeeper）。

安装详情可查看下一章节的WeBASE-Transaction部署说明

16.2 部署说明

16.2.1 1. 前提条件

重要：FISCO-BCOS 2.0与3.0对比、JDK版本、WeBASE及其他子系统的版本兼容说明！请查看

备注：

- Java推荐使用OracleJDK，可参考JDK配置指引（CentOS的yum仓库的OpenJDK缺少JCE(Java Cryptography Extension)，导致Web3SDK无法正常连接区块链节点）
- 安装说明请参看 安装示例，不使用分布式任务可以不部署ZooKeeper。
- 使用**国密SSL**需要修改`application.properties`中的`sdk.useSmSsl`配置为true（sdk将自动判断是否使用国密或非国密加密链数据，不再需要手动配置）
- 编译国密版智能合约在v1.3.1版本后，通过引入solcJ:0.4.25-rc1.jar，自动切换支持国密版智能合约的编译/部署/调用；（可自行切换solcJ-0.5.2）

16.2.2 2. 拉取代码

执行命令：

```
git clone -b lab https://github.com/WeBankBlockchain/WeBASE-Transaction.git

# 若因网络问题导致长时间下载失败，可尝试以下命令
git clone -b lab https://gitee.com/WeBank/WeBASE-Transaction.git
```

进入目录：

```
cd WeBASE-Transaction
```

16.2.3 3. 编译代码

使用以下方式编译构建，如果出现问题可以查看 常见问题解答

方式一：如果服务器已安装Gradle，且版本为gradle-4.10至gradle-6.x版本

```
gradle build -x test
```

方式二：如果服务器未安装Gradle，或者版本不是gradle-4.10至gradle-6.x版本，使用gradlew编译

```
chmod +x ./gradlew && ./gradlew build -x test
```

构建完成后，会在根目录WeBASE-Transaction下生成已编译的代码目录dist。

16.2.4 4. 修改配置

4.1 复制模板

进入编译目录dist：

```
cd dist
```

dist目录提供了一份配置模板conf_template：

根据配置模板生成一份实际配置conf。初次部署可直接拷贝。
例如：cp conf_template conf -r

4.2 复制证书

进入配置目录conf：

```
cd conf
```

将节点所在目录nodes/${ip}/sdk下的所有文件拷贝到当前conf目录（包括ca.crt, sdk.crt, sdk.key，国密下则包括smca.crt, smensdk.crt, smensdk.key, smsdk.crt, smensdk.key），供SDK与节点建立连接时使用。

4.3 修改配置

说明： 有注释的地方根据实际情况修改，完整配置项说明请查看 配置说明

```
vi application.properties
```

```
#####################################  Basic Configuration  ########################
↪##########
# 后台服务的版本
version=lab-rc2
# 工程服务端口，端口被占用则修改
server.port=5003
server.context-path=/WeBASE-Transaction
mybatis.mapper-locations=classpath:mapper/*.xml
logging.config=classpath:log4j2.xml

#################################### fisco-bcos javasdk Configuration #############
↪#####################
# sdk是否使用国密SSL（将自动判断是否使用国密加密）
sdk.useSmSsl=false
# fisco-bcos节点的ip列表，端口为channel端口
sdk.peers[0]=127.0.0.1:20200
sdk.peers[1]=127.0.0.1:20201
# sdk证书所在的相对路径，默认为conf
sdk.certPath=conf
# sdk线程池大小
sdk.threadPoolSize=50

#################################### constant Configuration ######################
↪###########
# WeBASE-Sign签名服务ip端口，使用本签名方式（signType=2）则对应修改
constant.signServer=127.0.0.1:5004
# 本地配置私钥进行签名，使用本签名方式（signType=0）则对应修改
constant.
↪privateKey=edf02a4a69b14ee6b1650a95de71d5f50496ef62ae4213026bd8d6651d030995
constant.cronTrans=0/1 * * * * ?
constant.requestCountMax=6
constant.selectCount=10
constant.intervalTime=600
constant.sleepTime=50
# 是否删除数据
constant.ifDeleteData=false
constant.cronDeleteData=0 0 1 * * ?
constant.keepDays=360
# 使用分布式任务部署多活（true-是，false-否）
constant.ifDistributedTask=false
```

(下页继续)

(续上页)

```
################################### elastic-job 分布式任务 ##########################
↪#########
# 部署多活的话需配置zookeeper，支持集群
job.regCenter.serverLists=127.0.0.1:2181
# zookeeper命名空间
job.regCenter.namespace=elasticjob-transaction
# 分片数（如多活3个的话可分成3片）
job.dataflow.shardingTotalCount=3

###################################        数据源配置        ########################
↪#########
# * 说明：本工程使用Sharding-JDBC分库分表，支持单一数据源，也支持多库多表。
# *       单库单表：配置单个数据源，将分库策略和分表策略注释或删除
# *       多库多表：配置多数据源，以群组分库，以年份分表，用户自定义每年分成几个表（注：分表策略的
路由字段不可修改[id,gmt_create]）
# * 样例：以两个数据源为例（数据库需事先创建），每张表根据年分表，每年再分成两个子表，
以2020和2021年的表为例

# 配置所有的数据源，如此处定义了ds0,ds1两个数据源，对应两个库
sharding.jdbc.datasource.names=ds0,ds1

# 定义数据源ds0，配置数据库连接信息
sharding.jdbc.datasource.ds0.type=com.alibaba.druid.pool.DruidDataSource
sharding.jdbc.datasource.ds0.driver-class-name=com.mysql.cj.jdbc.Driver
sharding.jdbc.datasource.ds0.url=jdbc:mysql://localhost:3306/webasetransaction0?
↪autoReconnect=true&useSSL=false&serverTimezone=GMT%2b8&useUnicode=true&
↪characterEncoding=UTF-8
sharding.jdbc.datasource.ds0.username=dbUsername
sharding.jdbc.datasource.ds0.password=dbPassword

## 定义数据源ds1，配置数据库连接信息
sharding.jdbc.datasource.ds1.type=com.alibaba.druid.pool.DruidDataSource
sharding.jdbc.datasource.ds1.driver-class-name=com.mysql.cj.jdbc.Driver
sharding.jdbc.datasource.ds1.url=jdbc:mysql://127.0.0.1:3306/webasetransaction1?
↪autoReconnect=true&useSSL=false&serverTimezone=GMT%2b8&useUnicode=true&
↪characterEncoding=UTF-8
sharding.jdbc.datasource.ds1.username=dbUsername
sharding.jdbc.datasource.ds1.password=dbPassword

# 定义数据库分片策略，如此处以群组id取模2来路由到ds0或ds1
# sharding.jdbc.config.sharding.default-database-strategy.inline.sharding-
↪column=group_id
# sharding.jdbc.config.sharding.default-database-strategy.inline.algorithm-
↪expression=ds$->{group_id % 2}
sharding.jdbc.config.sharding.default-database-strategy.inline.sharding-column=id
sharding.jdbc.config.sharding.default-database-strategy.inline.algorithm-
↪expression=ds$->{id % 2}

# 定义tb_deploy_transaction的分表策略，如此处以创建时间的年份和自增id取模2来路由到子表
sharding.jdbc.config.sharding.tables.tb_deploy_transaction.actual-data-nodes=ds$->
↪{0..1}.tb_deploy_transaction_$->{2021..2022}_$->{0..1}
sharding.jdbc.config.sharding.tables.tb_deploy_transaction.table-strategy.complex.
↪sharding-columns=id,gmt_create
sharding.jdbc.config.sharding.tables.tb_deploy_transaction.table-strategy.complex.
↪algorithm-class-name=com.webank.webase.transaction.config.
↪MyComplexShardingAlgorithm
sharding.jdbc.config.sharding.tables.tb_deploy_transaction.key-generator-column-
↪name=id

# 定义tb_stateless_transaction的分表策略，如此处以创建时间的年份和自增id取模2来路由到子表
```

(下页继续)

(续上页)

```
sharding.jdbc.config.sharding.tables.tb_stateless_transaction.actual-data-nodes=ds
↪$->{0..1}.tb_stateless_transaction_$->{2021..2022}_$->{0..1}
sharding.jdbc.config.sharding.tables.tb_stateless_transaction.table-strategy.
↪complex.sharding-columns=id,gmt_create
sharding.jdbc.config.sharding.tables.tb_stateless_transaction.table-strategy.
↪complex.algorithm-class-name=com.webank.webase.transaction.config.
↪MyComplexShardingAlgorithm
sharding.jdbc.config.sharding.tables.tb_stateless_transaction.key-generator-column-
↪name=id

sharding.jdbc.config.props.sql.show=false
```

16.2.5 5. 服务启停

返回到dist目录执行：

```
启动：bash start.sh
停止：bash stop.sh
检查：bash status.sh
```

备注：服务进程起来后，需通过日志确认是否正常启动，出现以下内容表示正常；如果服务出现异常，确认修改配置后，重启提示服务进程在运行，则先执行stop.sh，再执行start.sh。

```
...
        Application() - main run success...
```

16.2.6 6. 查看日志

在dist目录查看：

```
交易服务日志：tail -f log/transaction.log
web3连接日志：tail -f log/web3sdk.log
```

16.3 接口说明

16.3.1 1. 合约接口

1.1. 合约编译接口

接口描述

调用此接口编译合约。上传合约文件zip压缩包（压缩包里的每个合约的文件名要和合约名一致，合约引用需使用“./xxx.sol”），返回合约编译信息。

WeBASE-Transaction编译国密版智能合约，v1.3.1+版本已支持根据配置项的`encryptType`自动切换国密版soclJ jar包；

接口URL

http://localhost:5003/WeBASE-Transaction/contract/compile

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

压缩包文件

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "contractName": "HelloWorld",
      "contractBin": "xxx",
      "contractAbi": []
    }
  ]
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

1.2. 合约部署接口

接口描述

调用此接口发送合约部署相关信息，交易服务子系统会将合约部署请求信息缓存到数据库，通过轮询服务向节点发送交易请求，确保合约成功部署。

3.0.2及以后版本：

构造方法参数（funcParam）为String数组，每个参数都使用String字符串表示，多个参数以逗号分隔（参数为数组时同理），示例：

```
constructor(string s) -> ["aa,bb\"cc"]          // 双引号要转义
constructor(uint n,bool b) -> ["1","true"]
constructor(bytes b,address[] a) -> ["0x1a","[\
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE\",\
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9\"]"]
```

3.0.2以前的版本：

构造方法参数（funcParam）为JSON数组，多个参数以逗号分隔（参数为数组时同理），示例：

```
constructor(string s) -> ["aa,bb\"cc"]          // 双引号要转义
constructor(uint n,bool b) -> [1,true]
constructor(bytes b,address[] a) -> ["0x1a",[
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE",
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9"]]
```

接口URL

http://localhost:5003/WeBASE-Transaction/contract/deploy

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId":"group0",
  "uuidDeploy":"XXX",
  "signType":0,
  "contractBin":"0xXXXXX",
  "contractAbi":[],
  "funcParam":["hello"],
  "signUserId": "458ecc77a08c486087a3dcbc7ab5a9c3"
}
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": null
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

1.3. 合约地址查询接口

接口描述

根据群组编号和和部署业务流水号查询部署的合约地址。

接口URL

http://localhost:5003/WeBASE-Transaction/contract/address/{groupId}/{uuidDeploy}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5003/WeBASE-Transaction/contract/address/1/10001
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": "0xXXXXX"
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

1.4. 部署event查询接口

接口描述

根据群组编号和和部署业务流水号查询部署的合约的构造函数的event信息。

接口URL

http://localhost:5003/WeBASE-Transaction/contract/event/{groupId}/{uuidDeploy}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5003/WeBASE-Transaction/contract/event/1/10001
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": {
    "constructorEvent1": [
        "hello!"
    ],
    "constructorEvent": [
        "test",
        8
    ]
  }
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

1.5. 部署信息查询接口

接口描述

根据群组编号和和部署业务流水号查询部署的信息。

接口URL

http://localhost:5003/WeBASE-Transaction/contract/deployInfo/{groupId}/{uuidDeploy}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5003/WeBASE-Transaction/contract/deployInfo/1/10001
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": {
    "id": 1,
    "groupId": "group0",
    "uuidDeploy": "10001",
    "contractBin": "XXX",
    "contractAbi": "XXX"
    "contractAddress": "XXX",
    "funcParam": "[]",
    "signType": 0,
    "signUserId": "458ecc77a08c486087a3dcbc7ab5a9c3",
    "requestCount": 1,
    "handleStatus": 1,
    "transHash": "XXX",
    "receiptStatus": true,
    "gmtCreate": 1574853659000
  }
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

16.3.2 2. keystore接口

2.1. 查询账户地址接口

接口描述

查询本地配置私钥对应的账户地址 。

接口URL

http://localhost:5003/WeBASE-Transaction/key/address

调用方法

HTTP GET

请求参数

1）参数表

无

2）数据格式

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": {
    "0xfe12013103cf85f05b0862e5ef49da4fbdbd8f99"
  }
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

16.3.3 3. 交易接口

3.1. 交易请求接口

接口描述

调用此接口发送无状态交易请求，交易服务子系统会将交易请求信息缓存到数据库，通过轮询服务向节点发送交易请求，确保交易成功上链。当部署业务流水号为空时（即不是调用交易子系统部署合约），合约地址和abi不能为空。

3.0.2及以后版本：

方法参数（funcParam）为String数组，每个参数都使用String字符串表示，多个参数以逗号分隔（参数为数组时同理），示例：

```
function set(string s) -> ["aa,bb\"cc"]         // 双引号要转义
function set(uint n,bool b) -> ["1","true"]
function set(bytes b,address[] a) -> ["0x1a","[\
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE\",\
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9\"]"]
```

3.0.2以前的版本：

方法入参（funcParam）为JSON数组，多个参数以逗号分隔（参数为数组时同理），示例：

```
function set(string s) -> ["aa,bb\"cc"]         // 双引号要转义
function set(uint n,bool b) -> [1,true]
function set(bytes b,address[] a) -> ["0x1a",[
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE",
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9"]]
```

接口URL

http://localhost:5003/WeBASE-Transaction/trans/send

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId":"1",
  "uuidStateless":"XXX",
  "uuidDeploy":"XXX",
  "signType":0,
  "contractAddress":"0xXXXXX",
  "contractAbi":[],
  "funcName":"set",
  "funcParam":["hello"],
  "signUserId":"458ecc77a08c486087a3dcbc7ab5a9c3"
}
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": null
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

3.2. 交易查询接口

接口描述

调用此接口同步从节点查询交易信息。当部署业务流水号为空时（即不是调用交易子系统部署合约），合约地址和abi不能为空。

3.0.2及以后版本：

方法参数（funcParam）为String数组，每个参数都使用String字符串表示，多个参数以逗号分隔（参数为数组时同理），示例：

```
function set(string s) -> ["aa,bb\"cc"]         // 双引号要转义
function set(uint n,bool b) -> ["1","true"]
function set(bytes b,address[] a) -> ["0x1a","[\
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE\",\
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9\"]"]
```

3.0.2以前的版本：

方法入参（funcParam）为JSON数组，多个参数以逗号分隔（参数为数组时同理），示例：

```
function set(string s) -> ["aa,bb\"cc"]         // 双引号要转义
function set(uint n,bool b) -> [1,true]
function set(bytes b,address[] a) -> ["0x1a",[
↪"0x7939E26070BE44E6c4Fc759Ce55C6C8b166d94BE",
↪"0xce867fD9afa64175bb50A4Aa0c17fC7C4A3C67D9"]]
```

接口URL

http://localhost:5003/WeBASE-Transaction/trans/call

调用方法

HTTP POST

请求参数

1）参数表

2）数据格式

```
{
  "groupId":"1",
  "uuidDeploy":"XXX",
  "contractAbi":[],
  "funcName":"get",
  "funcParam":[]
}
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": [
    "hello"
  ]
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

3.3. 交易请求event查询接口

接口描述

根据群组编号和交易业务流水号查询交易请求的event信息。

接口URL

http://localhost:5003/WeBASE-Transaction/trans/event/{groupId}/{uuidStateless}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5003/WeBASE-Transaction/trans/event/1/20001
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": {
    "setEvent": [
        "test"
    ],
    "setEvent1": [
```

(下页继续)

(续上页)

```
        "test"
    ]
  }
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

3.4. 交易请求output查询接口

接口描述

根据群组编号和交易业务流水号查询交易请求的output信息。

接口URL

http://localhost:5003/WeBASE-Transaction/trans/output/{groupId}/{uuidStateless}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5003/WeBASE-Transaction/trans/output/1/20001
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": [
    "hello!"
  ]
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

3.5. 交易信息查询接口

接口描述

根据群组编号和交易业务流水号查询交易信息。

接口URL

http://localhost:5003/WeBASE-Transaction/trans/transInfo/{groupId}/{uuidStateless}

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5003/WeBASE-Transaction/trans/transInfo/1/20001
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": {
    "id": 1,
    "groupId": "group0",
    "uuidStateless": "20001",
    "uuidDeploy": "10001",
    "contractAbi": "XXX"
        "contractAddress": "XXX",
    "funcName": "set",
    "funcParam": "XXX",
    "signType": 0,
    "signUserId": "458ecc77a08c486087a3dcbc7ab5a9c3",
    "requestCount": 1,
    "handleStatus": 1,
    "transHash": "XXX",
    "transOutput": "0x",
```

(下页继续)

(续上页)

```
    "receiptStatus": true,
    "gmtCreate": 1574854118000
  }
}
```

b.异常返回结果示例（信息详情请参看附录1）

```
{
  "code": 103001,
  "message": "system error",
  "data": null
}
```

16.3.4 4. 其他接口

4.1. 获取EncryptType接口

接口描述

返回Transaction服务中web3sdk所使用的`encryptType`，0：标准，1：国密

接口URL

http://localhost:5003/WeBASE-Transaction/encrypt

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://127.0.0.1:5003/WeBASE-Transaction/encrypt
```

响应参数

1）参数表

2）数据格式

a.请求正常返回结果

```
{
  "code": 0,
  "message": "success",
  "data": 0
}
```

4.2. 查询WeBASE-Transaction版本接口

接口描述

获取WeBASE-Transaction的版本号

接口URL

http://localhost:5003/WeBASE-Transaction/version

调用方法

HTTP GET

请求参数

1）参数表

2）数据格式

```
http://localhost:5003/WeBASE-Transaction/version
```

响应参数

1）数据格式

a、成功：

```
v1.4.0
```

16.3.5 附录

1. 返回码信息列表

16.4 附录

16.4.1 1. 安装问题

1.1 Java部署

CentOS环境安装Java

注意：CentOS下OpenJDK无法正常工作，需要安装OracleJDK下载链接。

```
# 创建新的文件夹，安装Java 8或以上的版本，将下载的jdk放在software目录
# 从Oracle官网(https://www.oracle.com/technetwork/java/javase/downloads/index.html)选
择Java 8或以上的版本下载，例如下载jdk-8u201-linux-x64.tar.gz
$ mkdir /software

# 解压jdk
$ tar -zxvf jdk-8u201-linux-x64.tar.gz
```

(下页继续)

(续上页)

```
# 配置Java环境，编辑/etc/profile文件
$ vim /etc/profile

# 打开以后将下面三句输入到文件里面并保存退出
export JAVA_HOME=/software/jdk-8u201  #这是一个文件目录，非文件
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar

# 生效profile
$ source /etc/profile

# 查询Java版本，出现的版本是自己下载的版本，则安装成功。
java -version
```

Ubuntu环境安装Java

```
# 安装默认Java版本(Java 8或以上)
sudo apt install -y default-jdk
# 查询Java版本
java -version
```

1.2. 数据库部署

此处以Centos安装*MariaDB*为例。*MariaDB*数据库是 MySQL 的一个分支，主要由开源社区在维护，采用 GPL 授权许可。*MariaDB*完全兼容 MySQL，包括API和命令行。其他安装方式请参考MySQL官网。

① 安装MariaDB

- 安装命令

```
sudo yum install -y mariadb*
```

- 启停

```
启动: sudo systemctl start mariadb.service
停止: sudo systemctl stop  mariadb.service
```

- 设置开机启动

```
sudo systemctl enable mariadb.service
```

- 初始化

```
执行以下命令:
sudo mysql_secure_installation
以下根据提示输入:
Enter current password for root (enter for none):<-初次运行直接回车
Set root password? [Y/n] <- 是否设置root用户密码，输入y并回车或直接回车
New password: <- 设置root用户的密码
Re-enter new password: <- 再输入一次你设置的密码
Remove anonymous users? [Y/n] <- 是否删除匿名用户，回车
Disallow root login remotely? [Y/n] <-是否禁止root远程登录，回车
Remove test database and access to it? [Y/n] <- 是否删除test数据库，回车
Reload privilege tables now? [Y/n] <- 是否重新加载权限表，回车
```

② 授权访问和添加用户

- 使用root用户登录，密码为初始化设置的密码

```
mysql -uroot -p -h localhost -P 3306
```

- 授权root用户远程访问

```
mysql > GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '123456' WITH␣
↪GRANT OPTION;
mysql > flush PRIVILEGES;
```

- 创建test用户并授权本地访问

```
mysql > GRANT ALL PRIVILEGES ON *.* TO 'test'@localhost IDENTIFIED BY '123456'␣
↪WITH GRANT OPTION;
mysql > flush PRIVILEGES;
```

安全温馨提示:

- 例子中给出的数据库密码（123456）仅为样例，强烈建议设置成复杂密码
- 例子中root用户的远程授权设置会使数据库在所有网络上都可以访问，请按具体的网络拓扑和权限控制情况，设置网络和权限帐号

③ 测试连接和创建数据库

- 登录数据库

```
mysql -utest -p123456 -h localhost -P 3306
```

- 创建数据库

```
mysql > create database webasetransaction;
```

1.3. Zookeeper部署

此处给出简单步骤，供快速查阅。详情请参考官网。

（1）从官网下载对应版本的安装包，并解压到相应目录

```
mkdir /software
tar -zxvf zookeeper-XXX.tar.gz /software/
```

（2）配置和启动

ZooKeeper的安装包括单机模式安装，以及集群模式安装。具体步骤请参考官网说明:

- 集群部署
- 单机部署

16.4.2 2. 常见问题

2.1 脚本没权限

执行shell脚本报错误”permission denied”或格式错误

```
赋权限：chmod + *.sh
转格式：dos2unix *.sh
```

2.2 构建失败

“gradle build -x test”失败，不能编译Lombok注解：

```
...
/data/trans/webase-transcation/src/main/java/com/webank/webase/transaction/trans/
↪TransService.java:175: error: cannot find symbol
                        log.warn("save fail. contract is not deploed",␣
↪contractAddress);
                        ^
  symbol:   variable log
  location: class TransService
/data/trans/webase-transcation/src/main/java/com/webank/webase/transaction/trans/
↪TransService.java:183: error: cannot find symbol
                                log.warn("call fail. contractAddress:{} abi is not␣
↪exists", contractAddress);
                                ^
  symbol:   variable log
  location: class TransService
Note: /data/trans/webase-transcation/src/main/java/com/webank/webase/transaction/
↪util/ContractAbiUtil.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
100 errors

FAILURE: Build failed with an exception.
...
```

答： 修改 build.gradle文件，将以下代码的注释去掉

```
 //annotationProcessor 'org.projectlombok:lombok:1.18.2'
```

2.3 启动报错“nested exception is javax.net.ssl.SSLException”

```
...
nested exception is javax.net.ssl.SSLException: Failed to initialize the client-
↪side SSLContext: Input stream not contain valid certificates.
```

答：CentOS的yum仓库的OpenJDK缺少JCE(Java Cryptography Extension)，导致Web3SDK无法正常连接区块链节点，因此在使用CentOS操作系统时，推荐从OpenJDK网站自行下载。

2.4 启动报错“Processing bcos message timeout”

```
 [main] ERROR SpringApplication() - Application startup failed
org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating␣
↪bean with name 'scheduleService': Unsatisfied dependency expressed through field
↪'transService'; nested exception is org.springframework.beans.factory.
↪UnsatisfiedDependencyException: Error creating bean with name 'transService':␣
↪Unsatisfied dependency expressed through field 'web3jMap'; nested exception is␣
↪org.springframework.beans.factory.BeanCreationException: Error creating bean␣
↪with name 'web3j' defined in class path resource [com/webank/webase/transaction/
↪config/Web3Config.class]: Bean instantiation via factory method failed; nested␣
↪exception is org.springframework.beans.BeanInstantiationException: Failed to␣
↪instantiate [java.util.HashMap]: Factory method 'web3j' threw exception; nested␣
↪exception is java.io.IOException: Processing bcos message timeout
```

答：一些Oracle JDK版本缺少相关包，导致节点连接异常。推荐使用OpenJDK，从OpenJDK网站自行下载。

16.4.3 3. application.properties配置项说明

CHAPTER 17

WeBASE社区文章

17.1 区块链教程 | 使用WeBASE进行“两阶段交易”

作者：黎宁

作为一个友好的、功能丰富的区块链中间件平台， WeBASE致力于提高区块链开发者的运维与管理效率。在新近发布的 WeBASE v1.5.2 中，一大优化是提供了获取交易编码的接口，更方便用户使用”两阶段交易”。

“两阶段交易”是什么？“两阶段交易”是指分成两个步骤发送交易，即对交易编码并签名、将交易提交到链上这两个阶段：

- 第一阶段：构造并获取交易编码值，并通过私钥对交易编码值签名;
- 第二阶段：发送交易，也就是将已签名的编码值发送到链上。

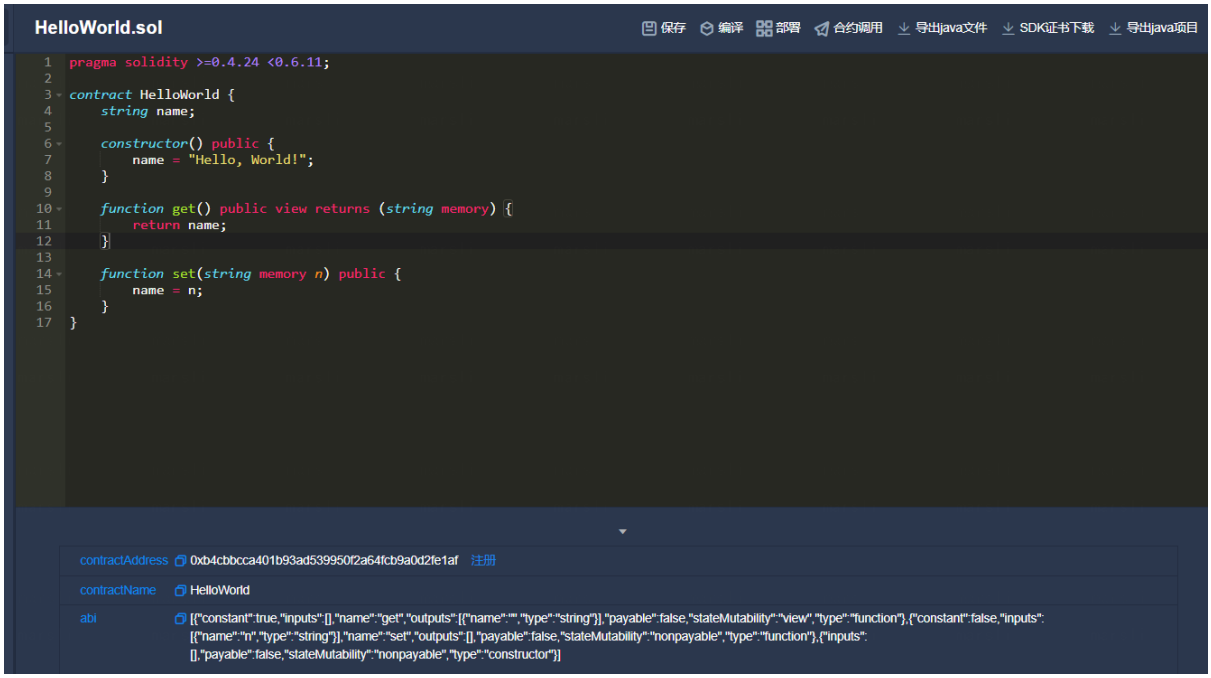
在WeBASE v1.5.2中，我们在WeBASE-Front节点前置服务中增加了获取交易编码值的功能。该接口可以返回未签名的交易编码值，也可以返回通过WeBASE-Front本地私钥或WeBASE-Sign私钥签名后的交易编码值。获得已签名的编码值后，用户直接调用前置服务的提交交易接口即可完成“两阶段交易”。

以下演示，我们通过WeBASE-Front节点前置服务接口获取交易编码值，并通过FISCO-BCOS Java-SDK对编码值进行签名，最后通过接口提交交易来加深对“两阶段交易”的了解。

17.1.1 |前期准备

部署HelloWorld合约

在发起交易之前，首先要确保在链上部署一个可调用的合约。这里以WeBASE-Front “合约仓库-工具合约”中的 “HelloWorld” 合约为例，部署一份 HelloWorld 合约。 我们在 WeBASE-Front 的合约IDE中编译一份 HelloWorld 合约并完成部署操作，如下图所示：



获得合约地址、合约ABI等信息后，我们根据 WeBASE-Front 的接口文档指引，调用获取交易编码接口。

查看接口文档

两阶段交易中，第一步交易编码并签名可以通过 WeBASE-Front 的 `/trans/convertRawTxStr/withSign` 接口构造一个已签名的交易体，接口文档简介如下：

5.6. 获取签名后的交易体编码值（结合WeBASE-Sign）

接口描述

构造交易体RawTransaction并将交易体编码，通过传入的 `signUserId` 签名服务的用户ID，使用对应的私钥对交值（十六进制字符串）

签名后的交易的编码值可以直接通过 `/trans/signed-transaction` 接口提交到链上

接口URL

http://localhost:5002/WeBASE-Front/trans/convertRawTxStr/withSign

调用方法

HTTP POST

值得一提的是，调用 `/trans/convertRawTxStr/withSign` 接口时：

- 如果传入了 signUserId 非空，则返回的交易体编码值是通过signUserId对应私钥签名后的交易体编码值。
- 如果传入的 signUserId 为空，则返回的是未签名的交易体编码值，开发者也可以通过JAVA-SDK用私钥对该值签名。

获取已签名的交易编码值后，就可以进行第二步的提交交易操作了。

在 WeBASE-Front 中，我们可以通过 `/trans/signed-transaction` 接口，将已签名的交易体编码值，完成交易上链并获得交易回执。

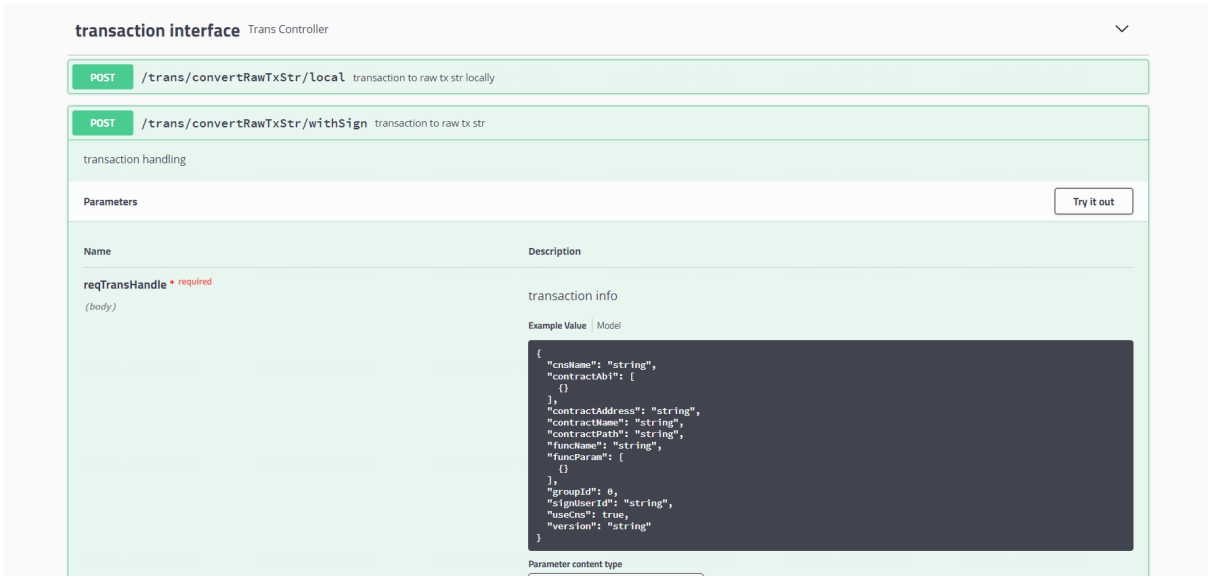
上述各个接口的调用方法都可以在 WeBASE-Front 的接口文档中找到（https://webasedoc.readthedocs.io/zh_CN/latest/docs/WeBASE-Front/interface.html）。

17.1.2 |结合WeBASE-Front接口进行“两阶段交易”

获取交易编码值

下面以 WeBASE-Sign 签名的获取交易编码接口(`/trans/convertRawTxStr/withSign`)为例，获取未签名的交易编码值。

我们可以访问 WeBASE-Front 的 Swagger 进行接口调用（如，http://localhost:5002/WeBASE-Front/swagger-ui.html），找到Swagger接口列表中的”transaction interface”交易接口一栏，点开 /trans/convertRawTxStr/withSign 即可。

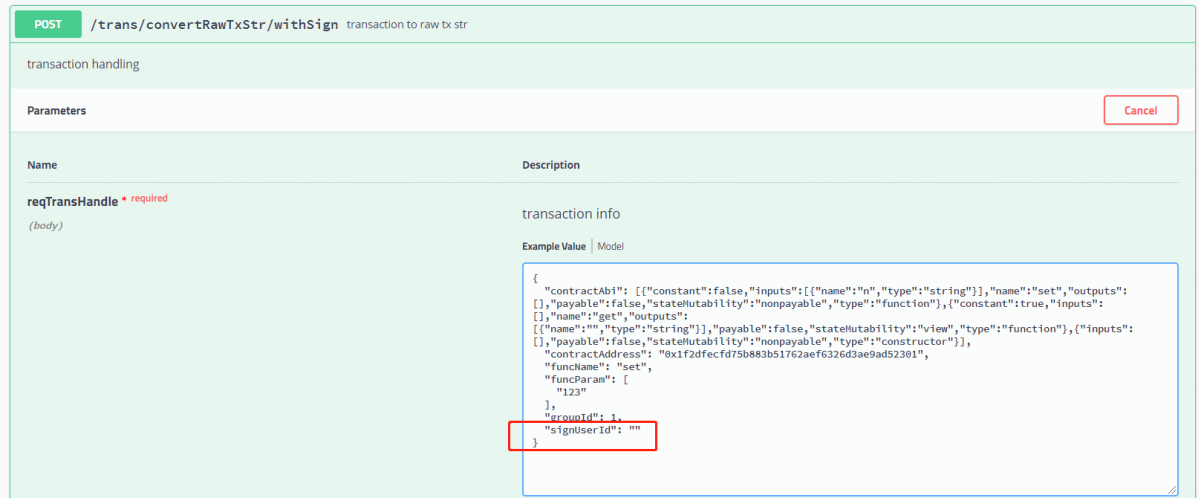


在文章开头我们提到，“两阶段交易”的第一阶段是交易编码并通过私钥对编码值签名。

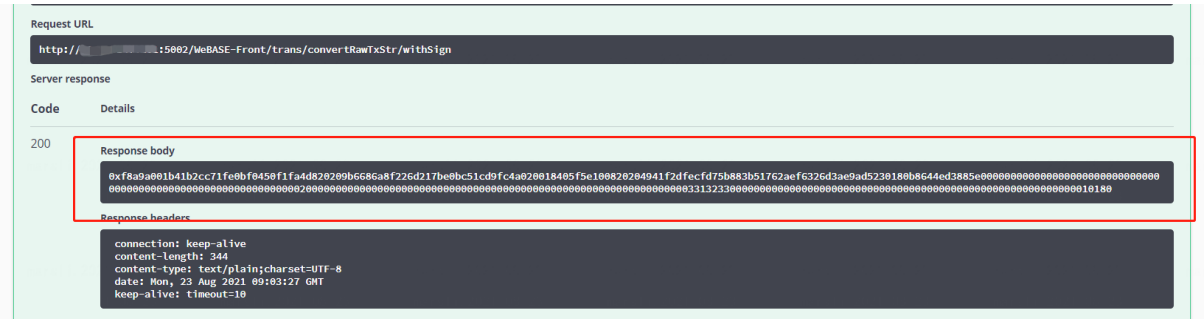
因此，我们调用接口时传入的 “signUserId” 为空字符串，接口将返回未签名的交易编码值，稍后我们再通过 Java-SDK 手动对编码值签名。在调用 /trans/convertRawTxStr/local 接口时同理，user地址字段为空字符串时也会返回未签名的交易编码值。

我们以调用HelloWorld合约的 “set” 方法为例，按接口文档填入对应参数。

首先，点开Swagger中的 /trans/convertRawTxStr/withSign 接口，再填入参数包括合约ABI、合约地址、函数名及函数入参、群组ID和WeBASE-Sign的私钥用户ID signUserId，点击”Try it out”输入参数，删除不必要的字段。注意，其中signUserId为空字符串。



点击”Execute”即可发起调用，获得未签名的交易编码值。接口返回值为：



拿到未签名的交易编码值之后，我们接下来通过 Java-SDK 对编码值进行签名。

对交易编码值签名

下面我们使用 FISCO-BCOS Java-SDK 加载私钥，对上文获取的未签名交易编码值进行签名操作，并根据 RawTransaction 交易体再次编码，得到最终签名后的交易编码值。

```
    public void testSign(TransactionEncoderService encoderService,
↪ RawTransaction rawTransaction) {
        // 未签名的交易编码值
        String encodedTransaction =
↪"0xf8a9a001b41b2cc71fe0bf0450f1fa4d820209b6686a8f226d217be0bc51cd9fc4a020018405f5e100820204941f
↪";
        // 私钥
        String privateKey = "0x123";
        // ECDSA 加密套件
        CryptoSuite cryptoSuite = new CryptoSuite(CryptoType.ECDSA_TYPE);
        // 对待签名的编码值作哈运算
        String hashMessageStr = cryptoSuite.hash(encodedTransaction);
        System.out.println("hashMessageStr: " + hashMessageStr);
        // 创建私钥对
        CryptoKeyPair myKeyPair = cryptoSuite.createKeyPair(privateKey);
        // 对交易编码值签名
        SignatureResult signedTx = cryptoSuite.sign(hashMessageStr, myKeyPair);

        // 获得最终签名后的交易编码值
```

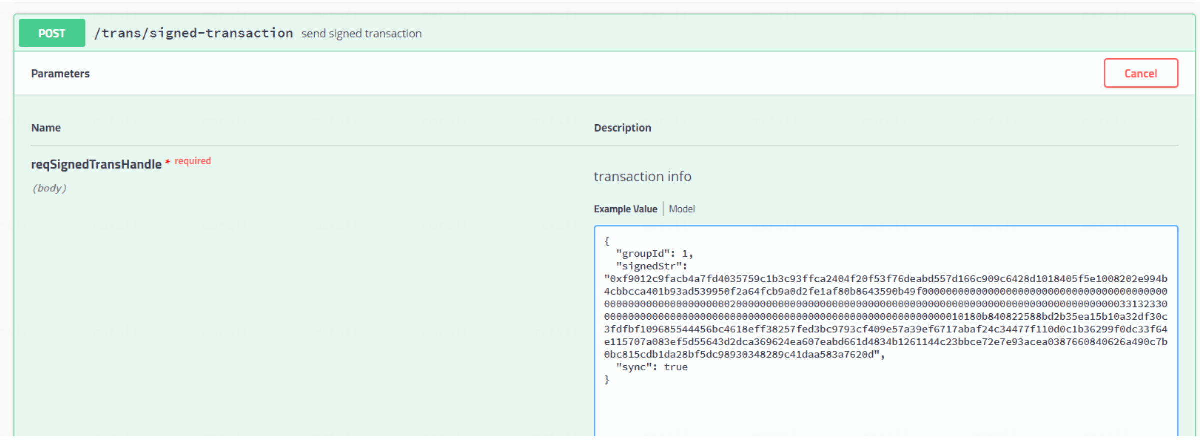
(下页继续)

(续上页)

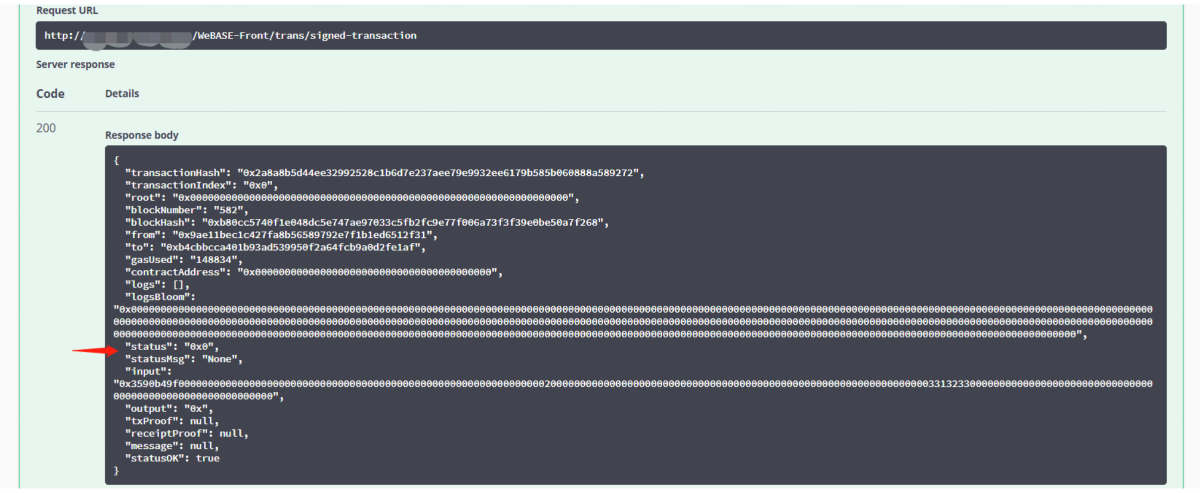
```
        byte[] signedTransaction = encoderService.encode(rawTransaction, signedTx);
        // 转十六进制字符串
        String signedTransactionStr = Numeric.toHexString(signedTransaction);
        System.out.println("signedTransactionStr: " + signedTransactionStr);
    }
```

提交交易

有了已签名的交易编码值后，我们可以调用 /trans/signed-transaction 接口，将该交易发到链上，获得交易回执。这里我们继续使用 Swagger 调用该接口。



提交请求后，接口返回了交易的回执。可以根据交易回执判断交易是否执行成功。



当看到返回的交易回执中显示 status 为 0x0，也就意味着交易执行成功了。

17.1.3 |交易编码接口源码解析

WeBASE-Front 源码中，位于 transaction 包里的 TransService 包含了对交易编码并签名的具体代码。

获取交易编码值

我们找到 createRawTxEncoded() 方法，该方法通过合约函数的ABI，合约函数的函数名 funcName 和合约函数入参 funcParam 等参数构造了 Function 实例，并通过FunctionEncoder 将 Function 实例进行编码得到字符串 encodedFunction（代码中的 cryptoSuite 是国密或非国密的加密套件，可用于计算哈希、创建私钥对、签名等）。

```
// 构造Function实例
Function function = new Function(funcName, contractFunction.getFinalInputs(),
        contractFunction.getFinalOutputs());
// 编码Function
FunctionEncoder functionEncoder = new FunctionEncoder(cryptoSuite);
String encodedFunction = functionEncoder.encode(function);
```

下面使用 convertRawTx2Str() 方法，该方法主要负责构造 RawTransaction 交易体。

构造 RawTransaction 需要传入一个随机数和从节点获取当前的 BlockLimit 值(避免重复提交交易)、合约地址和上文获得的 encodedFunction 等参数。

```
// 构造交易体
BigInteger randomId = new BigInteger(250, new SecureRandom());
BigInteger blockLimit = web3j.getBlockLimit();
RawTransaction rawTransaction =
    RawTransaction.createTransaction(randomId, Constants.GAS_PRICE,
        Constants.GAS_LIMIT, blockLimit, contractAddress, BigInteger.ZERO,␣
↪encodedFunction,
        new BigInteger(Constants.chainId), BigInteger.valueOf(groupId), "");
// 编码交易体RawTransaction
TransactionEncoderService encoderService = new TransactionEncoderService(cryptoSuite);
↪
byte[] encodedTransaction = encoderService.encode(rawTransaction, null);
```

对交易编码值签名

对交易编码值签名前，WeBASE-Front 中会根据传入的 user 字段和 isLocal 字段判断：

- 如果 user 字段为空，则将 encodedTransaction 转为十六进制后返回。该值就是第一阶段未签名的交易编码值。
- 如果 user 字段非空， isLocal 字段为 true，则 user 为 WeBASE-Front 本地的用户私钥，通过本地私钥对交易编码值 encodedTransaction 签名。注意，签名前还需对 encodedTransaction 进行一次哈希运算后再签名。
- 如果 user 字段非空， isLocal 字段为 false，则 user 为 WeBASE-Sign 托管私钥的signUserId，通过签名服务对交易体编码值 encodedTransaction 签名。注意，此处签名前没有对 encodedTransaction 进行哈希，而是直接转为十六进制发到签名服务，签名服务拿到该值后再做哈希运算并签名返回结果。

下面展示的代码为 isLocal 字段为 false，user 字段非空，其值为 signUserId 的交易编码值签名逻辑。

我们将交易编码值 encodedTransaction 转十六进制后，传到签名服务进行签名，得到了 String 格式的签名结果 signDataStr ，将签名结果反序列化，得到了签名结果 SignatureResult。同时，通过 TransactionEncoderService 将签名结果和上文构造的 RawTransaction 实例进行编码，最终可得到十六进制的已签名的交易编码值 signResultStr 。

```
// encodedTransaction转十六进制
String hashMessageStr = Numeric.toHexString(encodedTransaction);
// 通过WeBASE-Sign签名
EncodeInfo encodeInfo = new EncodeInfo(user, hashMessageStr);
String signDataStr = keyStoreService.getSignData(encodeInfo);
// 反序列化签名结果
SignatureResult signData = CommonUtils.stringToSignatureData(signDataStr,␣
↪cryptoSuite.cryptoTypeConfig);
// 加入签名结果，再次编码
byte[] signedMessage = encoderService.encode(rawTransaction, userSignResult);
// 转为十六进制
String signResultStr = Numeric.toHexString(signedMessage);
```

至此，获取交易编码，对交易编码签名交易体，并对编码值签名的过程就完成了。

值得一提的是，提交交易后获得的交易哈希 TransHash 值是通过对签名交易体编码值进行哈希计算得到的，有了交易哈希，也可以在提交交易后，直接根据交易哈希到链上查询交易回执。

```
// 通过CryptoSuite实例计算signResultStr的交易哈希值
String transHash = cryptoSuite.hash(signResultStr);
```

更多WeBASE社区文章，请参考本页面的latest分支

CHAPTER 18

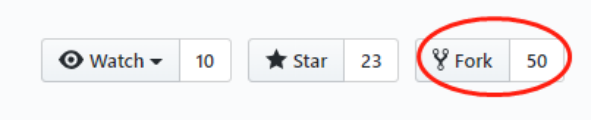
WeBASE贡献指南

欢迎，提前感谢你的帮助和支持！

如果你是第一次贡献，只需按照以下简单步骤操作即可。我们将以修改WeBASE-Node-Manager为例子给你介绍。

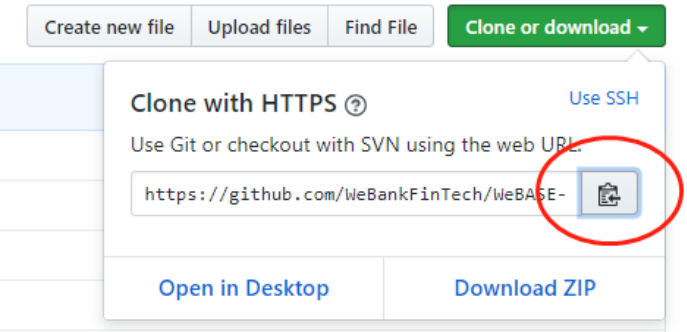
如果你的电脑上尚未安装 git, 请按照这个安装指引 进行安装。

18.1 Fork本代码仓库



点击图示中的按钮去 Fork 这个代码仓库。 这个操作会将代码仓库复制到你的账户名下。

18.2 Clone代码仓库



接下来，将复制后的代码仓库克隆到你的电脑上。点击图示中的绿色按钮，接着点击复制到剪切板按钮（将代码仓库地址复制下来）

随后打开命令行窗口，敲入如下 git 命令：

```
git clone "刚才复制的 url 链接"
```

“刚才复制的 url 链接”（去掉双引号）就是复制到你账户名下的代码仓库地址。获取这链接地址的方法请见上一步。

```
git clone https://github.com/"你的 Github 用户名"/WeBASE-Node-Manager.git
```

‘你的 Github 用户名’ 指的就是你的 Github 用户名。这一步，你将复制你账户名下的 WeBASE-Node-Manager 这个代码仓库克隆到你的本地电脑上。

18.3 代码修改

```
cd WeBASE-Node-Manager

vim XXX
```

18.4 Commit修改

```
git commit -am "一个伟大改进"
```

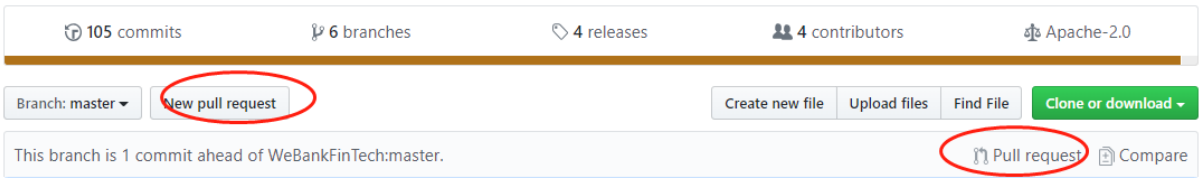
18.5 将改动 Push 到 GitHub

使用 `git push` 命令发布代码

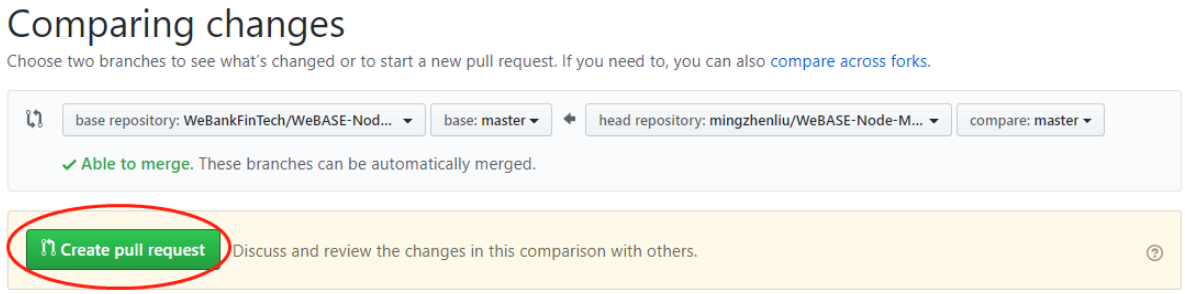
```
git push origin <分支的名称>
```

18.6 提出 Pull Request 将你的修改供他人审阅

前往 Github 你的代码仓库，你会看到一个 `Compare & pull request` 的按钮。点击该按钮。



接着再点击 `Create pull request` 按钮，正式提交 pull request。



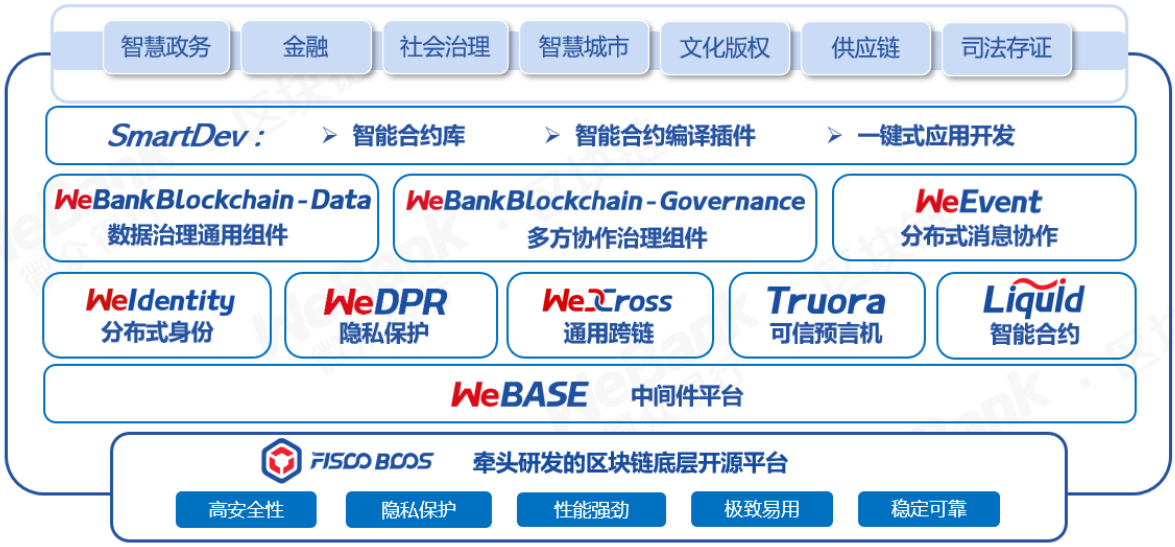
提交的改动经过审核，会合入到仓库。合并后，你会收到电子邮件通知。

CHAPTER 19

更多开源项目

All the project addresses participated and established by WeBank Blockchain are collected.

汇集了微众银行参与和建立的所有区块链项目地址。



19.1 FISCO-BCOS 适用于金融行业的区块链底层平台

git地址：https://github.com/FISCO-BCOS

gitee地址：https://gitee.com/FISCO-BCOS

文档地址： https://fisco-bcos-documentation.readthedocs.io/

19.2 WeBASE 区块链中间件平台

git地址：https://github.com/WeBankBlockchain/WeBASE

gitee地址：https://gitee.com/WeBank/WeBASE

文档地址：https://webasedoc.readthedocs.io/

19.3 Liquid 智能合约编程语言软件

git地址：https://github.com/WeBankBlockchain/liquid

gitee地址：https://gitee.com/WeBankBlockchain/liquid

文档地址： https://liquid-doc.readthedocs.io/

19.4 cargo-liquid Liquid智能合约辅助开发工具

https://github.com/WeBankBlockchain/liquid

https://gitee.com/WeBankBlockchain/cargo-liquid

CHAPTER 20

社区

20.1 加入微众银行区块链社区

关于微众区块链开源的动态，社区活动，欢迎关注“**微众区块链**”公众号，不定期还有开源周边相送！(^^)

社区助手微信 ID : WeBank_Blockchain

微众区块链小助手

扫一扫上面的二维码图案，加我微信