区块链服务网络用户手册

（Blockchain-based Service Network User Manual）

Version 1.4.0

区块链服务网络发展联盟

目录

前言

区块链服务网络 BSN（“服务网络”或者“BSN”）是为开发者提供了一站式的区块链开发、部署和运行环境服务，整合了各大云服务商、框架商以及多个门户方的资源。因为涉及到编程开发、配置资源、部署应用、网关调用、以及密钥证书等多个方面，因此 BSN 官方门户(bsnbase.com)为开发者准备了这个手册，帮助开发者顺利掌握 BSN 的使用技巧，从而能够将 BSN 作为区块链开发和部署的首选工具。

服务网络将会提供三大类服务：许可链服务、公有链服务和跨链服务。因为国内法律法规原因，公有链服务只能在 BSN 国际官网（bsnbase.io）使用。但在本手册内，仍然给出了公有链服务的使用说明。许可链服务主要分为两大部分：一是如何通过 BSN 门户将智能合约部署到选定的多个公共城市节点上；二是如何让 BSN 外的业务系统通过城市节点网关连接到相应的智能合约，并进行数据交易处理。BSN 的“Interchain Communications Hub”（跨链通讯枢纽），集成了分布科技的 Poly Enterprise 和边界智能的 IRITA 两种基于中继链机制的跨链解决方案，实现了不同许可链、开放联盟链、公链间的跨链互操作，我们将持续集成实现所有适配到 BSN 中的区块链都可以互联互通。

如果有任何其他问题，可以在联系我们章节内找到客服和技术支持的联系方式，欢迎随时沟通。也可以访问 BSN 官方知识库，里面有大量涉及 BSN 技术细节的文档和视频教学。

1 BSN 介绍

1.1 简介

服务网络的设计和建设理念完全借鉴互联网：互联网是由 TCP/IP 协议将所有数据中心连接而形成的，服务网络是通过建立一套区块链运行环境协议将所有数据中心连接而组成。与互联网一样，服务网络也是跨云服务、跨门户、跨底层框架的全球性基础网络。

服务网络的直接参与方有三类：一是云服务商，通过安装免费的服务网络公共城市节点软件，将其云服务资源（CPU、存储和带宽）接入服务网络，并在服务网络上进行销售；二是区块链底层框架商（包括联盟链和公有链），根据服务网络底层框架适配标准将框架进行适配后，可以部署到服务网络，供开发者选择使用，其中公有链适用于海外 BSN 门户和城市节点；三是门户商，可以在已有的传统门户网站内，依托服务网络的区块链运行环境和提供的 API 接口快速并低成本地建立 BaaS(Blockchain as a Service)平台，并向自己的客户提供基于服务网络的区块链应用开发、部署和运行服务。服务网络是一个开放性网络，任何云服务商、底层框架商和门户商在符合服务网络协议标准的前提下，均可以自由选择加入或退出服务网络。

与互联网一样，服务网络的直接使用者是开发者和科技公司。区块链应用开发者可以通过任意一个服务网络门户，在全世界任何公共城市节点上购买以 TPS（Transactions Per Second）、存储量和流量作为计费标准的云资源，并选择任何已适配的底层框架，以极低的成本和极方便的操作进行区块链应用的开发、部署和运营。区块链应用的发布者只需将应用的所有记账节点部署到服务网络的一个或多个公共城市节点上，参与者即可通过任何一个公共城市节点网关进行几乎无成本的接入。在每个公共城市节点内，所有部署的应用共享服务器资源。对于高频应用，公共城市节点可以为其智能化地自动分配单独的高处理性能的记账节点；而对于低频应用，则可多个应用共享一个记账节点。这种资源共享的机制能使服务网络所提供的资源成本

降低至传统区块链云服务所需成本的二十分之一。

服务网络是一个信息化基础设施，如同家家户户无需自己打井吃水，而是通过在城市建立公共水厂享受统一的供水服务，从而降低社会成本。在服务网络上，区块链应用发布者和参与者均不需要再购买物理服务器或者云服务来搭建自己的区块链运行环境，而是使用服务网络提供统一的公共服务，并按需租用共享资源，从而大大降低发布者和参与者的成本。经调研，如需搭建一个传统联盟链局域网环境，根据目前中国主流云服务商的报价，每年最低成本也要人民币 10 万元左右。而通过服务网络，一个应用每年仅需人民币 2-3 千元即可成链并进入运行。这样将鼓励大量的中小微企业、甚至学生在内的个人通过服务网络进行创新、创业，从而促进区块链技术的快速发展和普及。总体而言，从传统区块链的孤立封闭架构发展到服务网络的资源共享架构，完全遵循了互联网从早期的众多封闭孤立局域网逐步扩展成为全球性互联互通设施的发展历程。服务网络可以被视为区块链互联网。

1.2 BSN 服务

前面提到，服务网络主要为区块链开发者提供一站式的 DApp 应用部署、运行和管理的环境。从具体的服务上来讲，BSN 为用户提供以下三种服务：

1.2.1 许可链服务（Permissioned Services）

BSN 已经适配或正在适配国内绝大多数的主流联盟链框架，并提供许可链的共享节点服务、专有节点服务、IPFS 专网服务、开放联盟链、以及测试网服务。用户在 BSN 门户内，可以根据选择的底层框架，以及记账节点数，在 BSN 分布各地的公共城市节点上部署相应的联盟链应用。每个联盟链应用的记账节点数可以最多 40 个，并且可以分布在基于不同云服务的公共城市节点中。用户仅需要上传智能合约，并进行相应配置即可轻松完成联盟链应用的部署。这样的服

务模式可以让开发者将精力和时间花费在业务创新和智能合约编写上，所有与环境搭建、系统维护、应用部署、节点传输和网络配置的工作均由 BSN 统一完成。

BSN 许可链共享节点服务的定价策略是根据发布的应用每个记账节点的三资源要素制定的。这三要素为：TPS、存储和数据流量。其中 TPS 和存储在 BSN 官网内是预付费，而数据流量根据实际发生额进行后付费。这样的定价策略是为了能够将资源费降到最低，为用户提供最好的服务。根据目前 BSN 官网计算器显示，一个三个记账节点的 Fabric 应用，每个记账节点支持 10TPS、10GB 硬盘的情况，每月的费用仅为 140 元人民币。BSN 专有节点服务的定价策略根据服务选择的云平台主机配置，硬盘，数据流量制定，其中主机配置、硬盘是预付费，数据流量根据实际发生额后付费；IPFS 专网服务根据用户选择的存储空间实行预付费，数据流量同样根据实际使用额后付费；开放联盟链在 2021 年全年免费；跨链服务根据实际发生的跨链调用次数进行后付费；测试网提供的许可链服务发布、IPFS 专网、预言机服务和跨链服务全部免费。

1.2.2 公有链服务（Permissionless Services）

首先强调一下，BSN 的公有链服务只适用于 BSN 国际门户(www.bsnbase.io)和海外公共城市节点。相对许可链服务的复杂性，BSN 的公有链服务则比较简单。BSN 的公有链服务主要是为公有链 DApp 的开发者提供覆盖众多公有链节点的统一接入服务。开发者可以在 BSN 国际官网选择月套餐，并通过所接入的城市节点，可以同时在所有 BSN 已经适配的公有链节点上部署 DApp 和处理业务。

公有链套餐包括免费套餐，每天可以有 2000 个请求(Requests)。收费套餐有每月 20 美金、100 美金和 500 美金三种，分别包括每日 4 万，25 万和 150 万请求。所有请求可以自由分配到任何公有链上。

BSN 的公有链服务只提供共享节点和接入环境，而对公有链自身的业务完全无关。在任何公有链上发布和运行 DApp 所产生的链

上交易费用，由开发者自行承担，与 BSN 无关。

1.2.3 跨链服务（Interchain Services）

BSN 的愿景是成为区块链互联网。未来会有数百万各类 DApp 在 BSN 上部署和运行，不论是公有链还是许可链，都可以类似互联网上网站之间，非常轻松地相互调用和发生交易。在这种目标下，跨链是 BSN 技术体系内非常核心的一部分。

BSN 的“Interchain Communications Hub”（跨链通讯枢纽）已进入商用，集成了分布科技的 Poly Enterprise 跨链解决方案，支持联盟链与联盟链、联盟链与 ETH 测试网及 NEO 测试网之间的相互跨链。基于 IRITA 的跨链解决方案也在适配中，预计将在下个迭代中推出商用版本。

跨链服务也已在测试网上线，集成了分布科技的 Poly Enterprise 和边界智能的 IRITA 两种基于中继链机制的跨链解决方案。欢迎大家在测试网进行体验试用并反馈问题和开发建议，我们将持续完善功能。

BSN 的“Interchain Communications Hub”（跨链通讯枢纽），实现了不同许可链、开放联盟链、公链间的跨链互操作，并将持续集成新的跨链解决方案，实现所有适配到 BSN 中的区块链都可以互联互通。

1.3 名词解释

- **城市节点**：是服务网络的基础网络运行单元，物理部署在很多城市的 IDC 机房或云资源内。城市节点在服务网络中承担数据信用责任，是服务网络的基础网络参与方，也是服务网络中的基础网络节点组织成员。多个城市节点即构成以区块链技术为基础的服务网络的基础网络联盟。
- **行业应用**：服务网络上运行的应用服务和在门户应用商店中应用产品的总称。

- **应用服务：**指在区块链服务网络中已经发布并运行的区块链应用，服务的用户可以通过服务网络门户申请参与使用。
- **应用产品：**指没有在区块链服务网络中发布和运行，仅在服务网络门户的应用商店内上架的区块链应用。产品销售后，由购买方在服务网络上发布后方可以投入使用。产品发布完成之后即成为服务，此时购买方成为应用发布者，并可以邀请其他用户参与。产品的设置是让某些开发者仅以开发和销售产品为商业模式，而不需要投入成本去运营自己的产品（对比互联网，产品开发者类似提供网站搭建的开发者，而服务发布者类似经营网站的运营者）。
- **服务发布者：**发布区块链应用服务，拥有该应用产权，可以进行服务的运营和商务拓展管理，对通过门户网站的服务参与者的加入进行审批和操作权限管理。服务可以为公开服务和非公开服务。非公开服务将在门户网站内无法浏览和搜索到，由服务发布者内部使用或者通过自己业务系统管理使用者。
- **服务参与者：**是使用应用服务的客户，可以使用发布方的业务系统或者使用自有业务系统接入相关区块链应用进行链上数据操作。
- **产品开发者：**是在服务网络门户的应用商店内发布应用产品信息的个人或企业开发者。
- **链下业务系统/BSN 外业务系统：**是应用服务发布者或服务参与者部署在区块链服务网络之外的业务系统；链下业务系统通过区块链服务网络提供的城市节点网关接口调用应用服务的智能合约获取或写入数据。

2 更新说明

发布时间	版本号	更新内容
2021/01/31	1.4.0	**国内门户** 1、BSN 中国官网（www.bsnbase.com）界面迭代优化与技术优化，提升用户体验； 2、推出开放联盟链文昌链（基于 COSMOS/IRISnet SDK）与泰安链（基于 FISCO BCOS）； 3、BSN IPFS 服务专网商用发布； 4、BSN ConsenSys Quorum 专有节点服务上线； 5、推出基于 Poly Enterprise 的跨链通信枢纽商用服务； 6、修复了部分 BUG，提升了系统的稳定性。 **国际门户** 1、BSN 国际官网（www.bsnbase.io）迭代优化与技术优化，提升用户体验； 2、推出基于 Poly Enterprise 的跨链通信枢纽商用服务； 3、修复了部分 BUG，增强了系统的稳定性。
2020/12/31	1.3.1	**国内门户** 1、BSN 中国官网（www.bsnbase.com）界面迭代优化与技术优化，提升用户体验； 2、优化账号注册和企业实名认证流程； 3、BSN 许可链测试网集成了微众基于联盟链的 Truora 预言机服务； 4、CITA 国密框架适配商用发布； 5、修复了部分 Bug，提升了系统的稳定性。
2020/10/31	1.3.0	**国内门户** 1、BSN 中国官网（www.bsnbase.com）界面迭代优化与技术优化，提升用户体验； 2、BSN 许可链测试网上线，为开发者提供免费的联盟链服务测试环境，支持： • Hyperledger Fabric R1、FISCO BCOS 国密、XuperChain 国密应用服务发布 • 跨链测试服务 • IPFS 测试服务 3、推出基于 Poly Enterprise 和 IRITA 的 BSN 跨链通信枢纽测试网服务； 4、完成 CITA 国密框架适配，并推出公测版本； 5、XuperChain 和 Hyperledger Fabric 国密框架适配商用发布； 6、节点网关 SDK 集成 Xuperchain、CITA 国密服务接口； 7、修复了部分 BUG，提升了系统的稳定性。 **国际门户** 1、BSN 国际官网（www.bsnbase.io）迭代优化与技术优化，提升用户体验； 2、完成与 ShareRing、Solana 及 Alogrand 公有链的适配，提供主网和测试网节

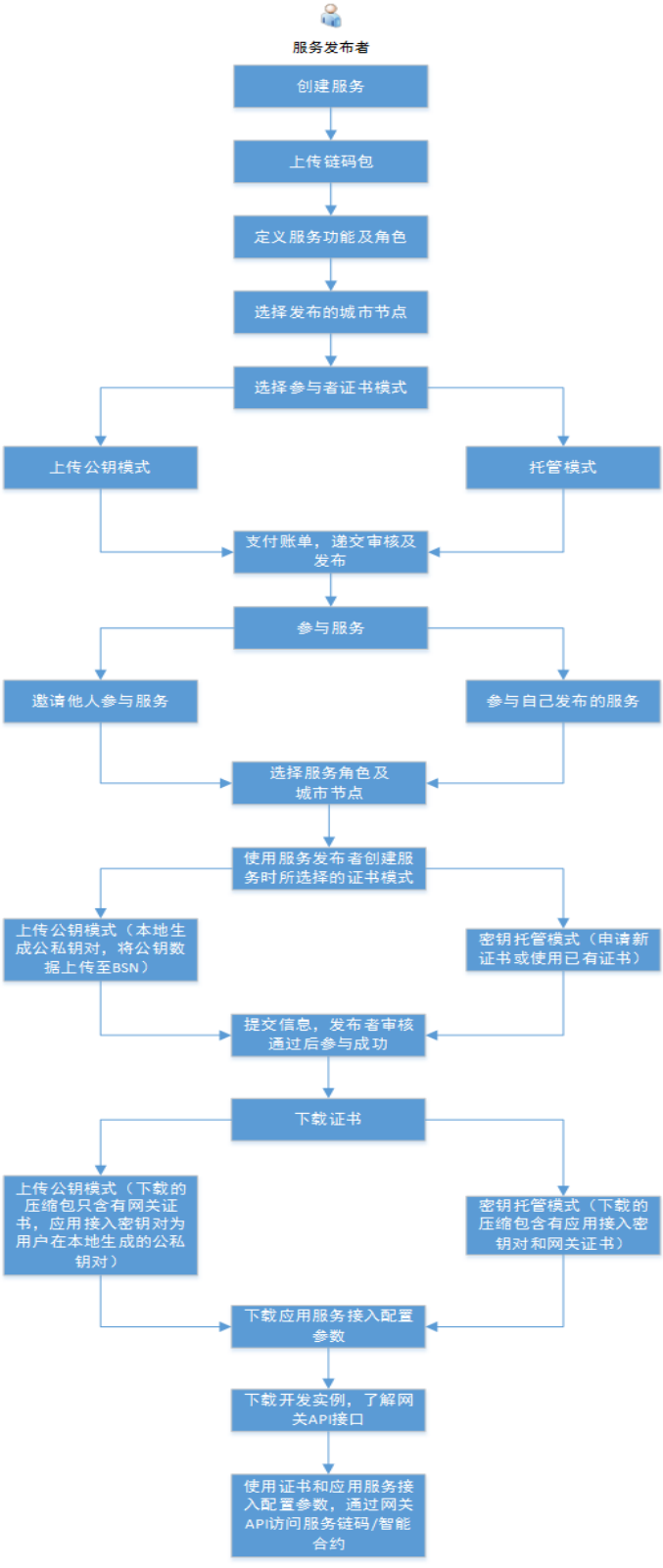
		点原生接口服务； 3、BSN 许可链测试网上线，为开发者提供免费的区块链服务测试环境，支持： • Hyperledger Fabric R1、FISCO BCOS K1 应用服务发布 • 跨链测试服务 4、推出基于 Poly Enterprise 和 IRITA 的 BSN 跨链通信枢纽测试网服务； 5、推出第三方网站接入 BSN 公有链服务管理 API； 6、修复了部分 BUG，增强了系统的稳定性。
2020/7/31	1.2.0	**国内版** 1.bsnbase.com 门户迭代优化，提升用户体验; 2.公共城市节点网关接口和 SDK 迭代优化，现有 API 接口增加 FISCO BCOS 上传公钥模式、链上事件订阅和通知机制相关 API； 3.FISCO BCOS 适配商用优化，实现应用服务商用计费计价； 4.完成百度 XuperChain 适配，并推出公测版本； 5.完成 Hyperledger Fabric 国密适配，并推出公测版本； 6.优化 BSN 公共城市节点运行监控机制； 7.推出 BSN 学生免费测试网; 8.修复了部分 BUG，提升了系统的稳定性。 **国际版** 1、全新的用户界面设计，提供易用的用户体验； 2、增加公有链主网和测试网节点服务，提供公有链节点原生接口接入服务。包含：Nervos、NEO、ETH、Tezos、EOS、IRISNET 等； 3、Permissioned Services 增加商用功能，支持 Hyperledger Fabirc 和 FISCO BCOS 两种框架； 4、更新 FISCO BCOS 节点支持 secp256 k1 密码算法； 5、增加 Permissioned Services 的应用服务发布费用支付、服务周期扣费、服务配置升级、流量周期扣费等功能，Permissionless Services 的套餐购买、套餐升级支持商用所需的计费计价功能； 6、用户中心增加了“我的账户”，方便用户更新信用卡信息、账单查询、账单支付以及账单下载等功能； 7、全新的在线文档，为开发者和门户使用者提供更流畅、易懂的帮助手册； 8、在 github 开源节点网关 SDK 及实例，https://github.com/bsnda； 9、修复了部分 BUG，增强了系统的稳定性；

2020/4/25	1.1.0	1.门户（bsnbase.com）面向公众访问者调整了登录前主页的栏目结构，增加了更多更全面的 BSN 产品、培训沙龙、赛事活动、媒体资讯、节点运行信息以及适配框架等相关内容； 2.应用服务发布支持多框架，新增适配 FISCO BCOS 并启动两个月的免费公测活动； 3.取消了内测期间对发布应用和每个应用记账节点的数量限制； 4.应用服务发布支持参与者接入身份密钥托管和公钥上传两种管理模式； 5.简化了应用服务发布的审批流程，并进一步提高服务部署的自动化程度，服务发布效率明显提升； 6.增加了应用服务配置升级功能； 7.支持非公开应用服务的基础信息修改； 8.应用服务参与者可在线更新接入身份证书和接入的城市节点； 9.应用服务支持停用、启用、卸载功能； 10.产品上架、应用服务发布、服务配置升级支持商用所需的计费计价功能； 11.用户中心增加了“我的账户”、“我的账单”、“我的发票”等商用相关功能，方便用户进行余额查询、账户充值、提现、账单支付、开发票等操作。支持银联云闪付、微信支付、企业网银、线下汇款等充值支付方式。 12.节点网关增加应用用户注册接口，链下系统可通过该接口为业务用户注册链上账户； 13.节点网关支持 Fabric 应用用户证书托管和非托管模式的用户注册，支持交易临时数据和链码事件注册、查询、通知和注销功能； 14.节点网关支持 FISCO BCOS 用户注册密钥托管、区块链交易处理和查询接口，支持应用区块信息查询； 15.优化节点网关交易限流、交易路由、负载均衡、流量日志等功能； 16.提供节点网关客户端 SDK（golang、java 版本），降低非托管密钥模式下应用业务系统交易报文封装的开发复杂性； 17.城市节点提供统一的适配多种区块链框架密钥算法的公私钥对生成、证书签发、数据签名和验签服务； 18.英文官网 www.bsnbase.io 发布。

3 快速入门

3.1 联盟链

3.1.1 共享节点服务



服务发布者可以在 BSN 门户中创建服务，创建服务时需要上传链码包、定义服务功能及角色、选择发布的城市节点及选择参与者证书模式（包括托管模式和上传公钥模式两种），完成这些操作以后需支付相关费用并由服务网络运营方进行审核和发布。

服务发布成功之后，发布者可以参与自己的服务或邀请其他用户参与服务，参与服务时需要选择角色、城市节点，并根据发布者设置的证书模式生成证书，经服务发布者审核通过之后参与成功。

参与服务成功之后可以下载链下业务系统接入指定节点网关的应用接入说明文件，并可以使用证书和应用服务接入配置参数通过网关 API 访问服务链码/智能合约。

3.1.2 专有节点服务

BSN 专有节点服务应用了 BSN 的多底层框架适配、虚拟化容器、自动化部署和节点网关等技术，为用户提供“开箱即用”的区块链专享云服务。用户可以在 BSN 门户快速创建自己专享的联盟链运行环

境，配置节点 CPU、内存、磁盘容量等参数；可以自主的进行节点管理、发布智能合约、访问节点数据、监控区块链运行状态等。专有节点不对底层框架的 API 做限制，开发者通过网关接入专有节点后，所有的接口开发者都可调用。

更多关于专有节点的介绍，请访问第六章《BSN 专有节点服务》。

3.2 教学视频

3.2.1 BSN 许可链

下面的视频通过一个案例演示了许可链服务的整个流程：从在 BSN 门户内发布一个应用服务，到 BSN 外的业务系统如何通过城市节点网关接入应用服务的智能合约。

https://www.bsnbase.com/static/tmpFile/Video_BSN.mp4

3.2.2 BSN 公有链

下面的视频演示了如何在 BSN 国际门户使用公有链产品进行公有链服务的发布以及 BSN 外系统接入的流程。

https://www.bsnbase.com/static/tmpFile/Video_BSN_PERMISSIONLESS_BLOCKCHAIN.mp4

3.3 文档

BSN 门户的直接用户是开发者。BSN 作为一个区块链应用开发、部署和运行的环境和工具，整体的使用还是比较复杂的。建议所有开发者，应该从学习文档和示例练手开始，在一到两天内就能够掌握 BSN 的使用。

我们提供的所有示例放在 Github 上，供大家方便使用以外，也希望有兴趣的开发者可以帮助我们优化和丰富示例，让更多的开发者能够更轻松的上手区块链开发。对示例有共享的开发者，我们会邮寄小礼品以及邀请参加 BSN 内部技术讨论会。

所有文档和示例链接，请访问第十三章《在线文档》。

4 账户注册

4.1 BSN 官方门户

- 打开网络地址：www.bsnbase.com；点击登录页面中的【注册】按钮进入注册申请页面：

- 根据页面中的提示填写注册申请信息并提交。注册申请用户默认注册为【个人用户】，如果在 BSN 上发布应用的费用需要开具企业发票，则建议申请成为【企业用户】，可以在完成注册后登录门户-用户中心-企业认证，提交企业实名制认证申请；
- 注册申请提交后将将收到系统发送的激活邮件，用户可点击**激活邮件**中的激活链接来设置登录密码进行激活；
- 帐户激活后即可登录系统使用；
- 申请企业实名认证：登录门户后，去用户中心-企业认证，提交企业实名制认证申请，提交后，由运营人员进行审核；

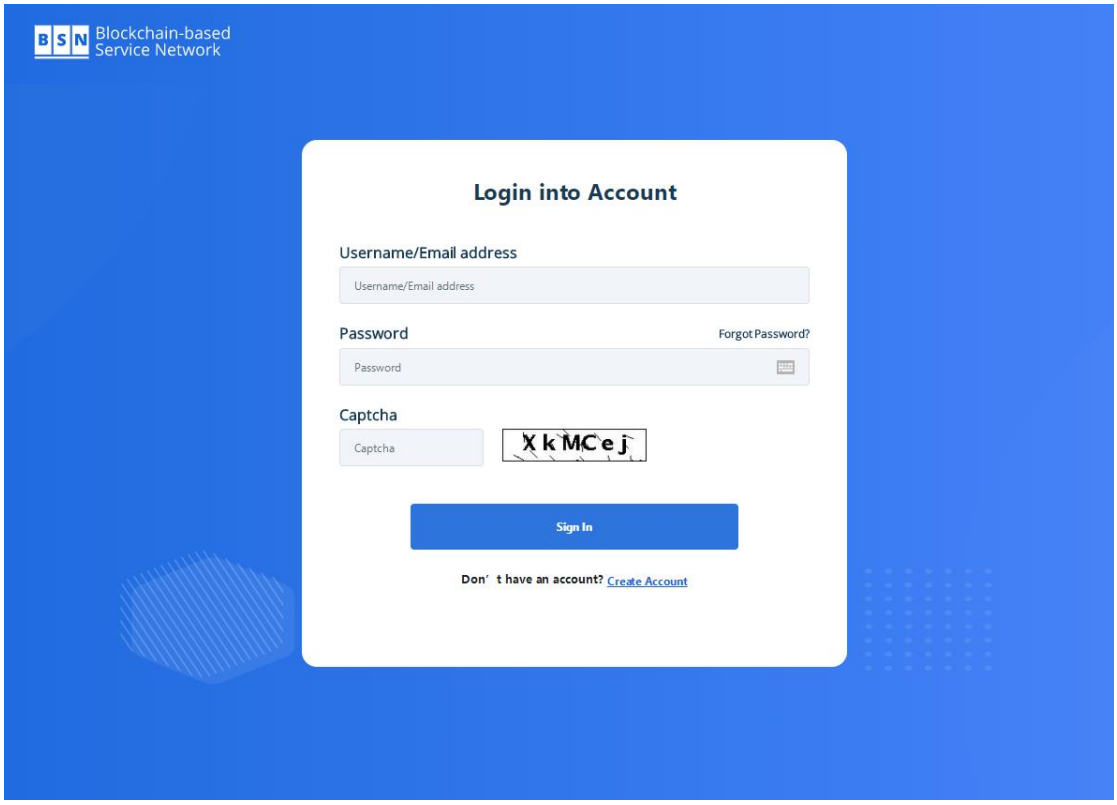

* 企业名称:
请输入企业名称
用户账号:
注册邮箱:
* 此邮箱和手机号用于您接收系统消息通知，如果您需要变更对接人员信息，请将需求发送邮件至support@bsnbase.com。
注册手机号:
* 国家:
中国
* 经营期限至
选择日期
长期
* 详细地址:
请填写您的详细地址
* 联系人姓名:
* 手机号:
* 邮箱:
* 统一社会信用代码:
请输入统一社会信用代码
* 上传企业证件:
上传营业执照图片;
清晰的图片可以让您更快通过审核;
推荐上传扫描文件，拍照请尽量降低反光;
请确保营业执照图片所有信息清晰可见，内容真实有效，无任何修改;
文件大小不超过10M，格式支持 .jpg .jpeg .bmp .png .pdf。
提交


企业认证
正在认证，请等待
企业名称:
统一社会信用代码:
温馨提示:
我们将会以邮件的方式通知您认证结果，请注意查收邮件。

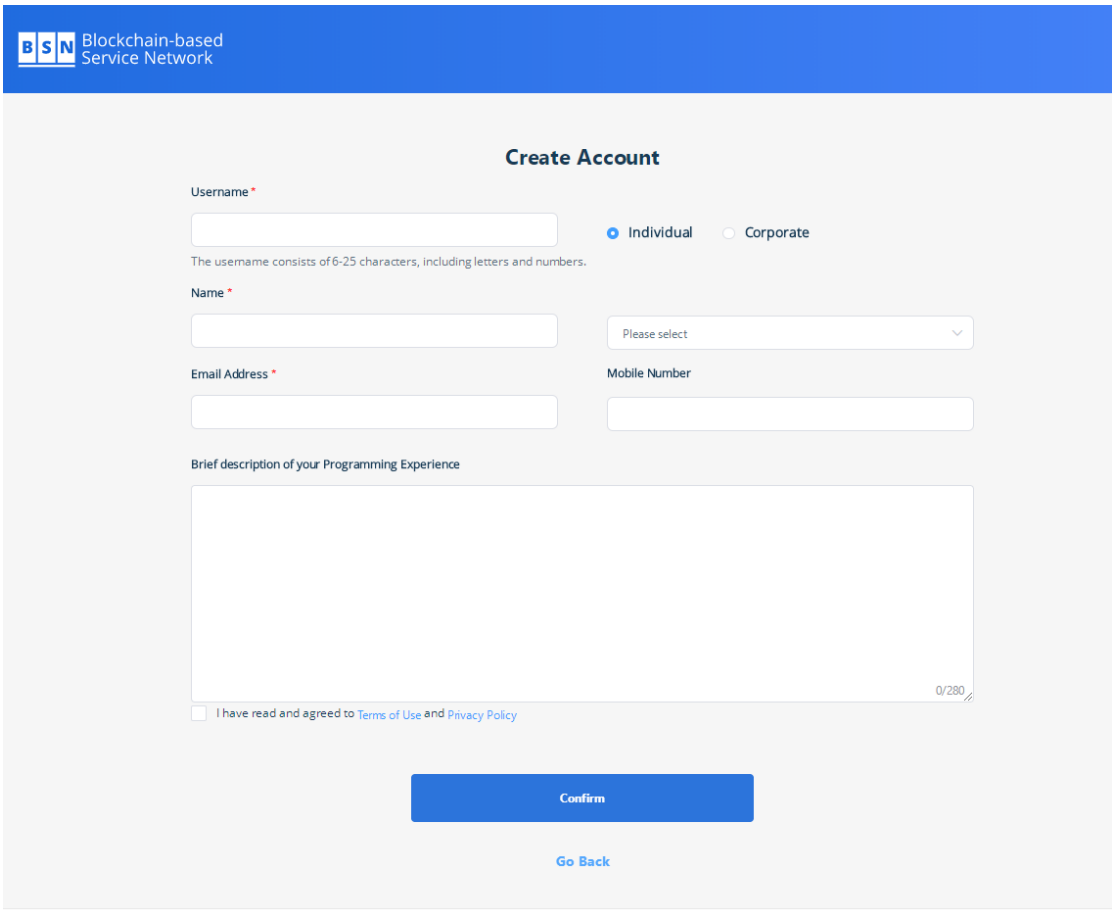
- 审核通过后，邮箱和手机会收到企业实名认证审核通过的通知。

4.2 BSN 国际门户

- 打开网络地址：www.bsnbase.io 进入 BSN 国际门户。并点击 Login，如下图：



- 在页面中点击【Register】进入注册页面，如下：



- 根据页面中的提示填写注册申请信息并提交。注册申请用户分为【个人用户】和【企业用户】两种，申请人需根据自己的用户类型任选一种申请注册。
- 注册申请提交后将将收到系统发送的激活邮件，用户可点击**激活邮件**中的激活链接来设置登录密码进行激活；
- 帐户激活后即可登录系统使用，BSN 国际门户不需要实名认证。

5 许可链服务

5.1 概述

许可链服务是 BSN 提供的核心服务之一。其目的是让开发者可以轻松地在自己选择的公共城市节点上发布基于许可链底层框架的分布式应用（DApp）。许可链 DApp 和公有链 DApp 相比较，从架构设计、运行效率和智能合约编写等各方面，都要比公有链 DApp 灵活，并且具有很大的创新空间。但从开发角度，因为底层环境都需要自己搭建，而公有链底层环境是现成的，因此许可链 DApp 的开发和运维难度较高。BSN 为开发者解决的正是这个问题：BSN 将许可链底层环境已经搭建好，开发者只需要上传智能合约，选择部署的记账节点数和部署的城市节点，递交后，BSN 将完成整个链的搭建和运行。开发者的 BSN 外业务系统，可以通过城市节点网关，接入 DApp 进行数据业务处理。

虽然 BSN 将许可链 DApp 的开发难度进行了大幅度降低，但开发者仍然需要对以下三个方面进行深入了解，本章节后续主要围绕下面这三个方面进行详细说明：

第一是密钥证书体系：区块链应用本身就是基于加密算法的技术，因此对密钥证书要求较高。BSN 为了保证开发者的业务数据有行业内的最高安全保护机制，因此设置了两层密钥证书：应用接入密钥和用户交易密钥，并且每种证书可以通过两种方法生成：密钥托管模式和公钥上传模式。

第二是应用服务发布和参与流程：要建立一个许可链应用，首先要把这个链建立起来，并且部署上智能合约。这部分工作完全是在 BSN 官方网站（bsnbase.com）上实现，包括智能合约上传、证书模式选择、权限角色设定、记账节点配置、城市节点位置等等操作，最后需要上传或下载相应密钥，方便 BSN 外系统的接入。

第三就是 BSN 外业务系统接入：这里详细描述 BSN 外业务系统

的接入参数配置、SDK 使用、以及接入城市节点网关的 API 说明。API 说明包括了目前已适配的所有联盟链框架。

目前 BSN 许可链服务已推出共享节点服务和专有节点服务，以上是共享节点服务的介绍，关于专有节点服务的详情，可访问第六章《BSN 专有节点服务》。

5.2 BSN 密钥证书体系

应用发布者在 BSN 部署链码/智能合约后，发布者可以邀请参与方参与应用。参与方（包括发布者）的链下业务系统通过城市节点网关调用链码/智能合约并进行数据交易处理。在这个过程中，参与方需要两个密钥对来完成整个操作：应用接入密钥对和用户交易密钥对。应用发布者在 BSN 门户内发布应用服务时，要选择该应用服务是“密钥托管模式”还是“上传公钥模式”。密钥托管模式是指上述应用接入密钥对和用户交易密钥对均由 BSN 生成并托管起来，参与方下载或者直接调用相应接口使用即可。上传公钥模式是由参与方在本地生成公私钥对，并在 BSN 门户上传应用接入公钥，和通过城市节点网关接口将公钥证书申请文件上传登记生成用户交易证书。发布者一旦选用一种模式，该应用服务的所有密钥均要按照该模式进行，并且后期无法改变密钥模式。

1、 密钥托管模式下应用接入密钥对

在密钥托管模式下，应用接入密钥由参与方在 BSN 门户内生成，托管在 BSN 上，并下载私钥使用。请详见后续章节内的应用服务参加说明。

2、 密钥托管模式下用户交易密钥对

在密钥托管模式下，当参与方参与应用服务后，其链下业务系统可以调用相应节点网关接口 API 来为应用注册子用户账户并生成托管子用户交易密钥对。每个子用户可以用自己的托管密钥对参与交易。

参与方的业务系统连入节点网关时，使用每个子用户的 UserCode 去调用相应密钥对进行交易执行。在访问 Fabric 应用服务时，如果还没有注册子用户，节点网关会使用默认生成的子用户交易密钥对；在访问 FISCO BCOS 和 XuperChain 应用服务时，必须先调用网关 API 的用户注册接口生成子用户的密钥对。具体请详见 Fabric、FISCO BCOS 和 XuperChain 底层框架的密钥说明和相应接口描述。

3、 上传公钥模式下应用接入密钥对

在上传公钥模式下，应用服务参与方必须在本地生成应用接入密钥的公私钥对，在本地保存私钥，并在 BSN 门户内上传公钥文件。具体本地操作请参考本章节后附的《上传公钥模式下应用接入密钥生成方法说明》。具体如何上传公钥请参考后续章节的上传公钥内容。

4、 上传公钥模式下用户交易密钥对

在上传公钥模式下，应用服务参与方同理也必须先在本地生成自己的用户交易密钥的公私钥对，在本地保存私钥。然后通过城市节点网关的用户交易密钥证书登记接口，上传公钥证书申请文件进行证书登记。如果参与方的链下系统内有多个子用户，也可以为每个子用户生成各自的公私钥对并上传公钥证书申请文件进行登记。在本地生成公私钥对和公钥证书申请文件的方法可以参照网关 SDK 实例中的 API 说明。上传公钥的证书登记接口请参见本手册后续接口说明。

网关 SDK 实例下载地址：

https://github.com/BSNDA/PCNGateway-Go-SDK

https://github.com/BSNDA/PCNGateway-Java-SDK

https://github.com/BSNDA/PCNGateway-PY-SDK

https://github.com/BSNDA/PCNGateway-CSharp-SDK

在当前版本下，只有基于 Hyperledger Fabric 和 FISCO BCOS 的应用服务支持上传公钥模式，基于 Xuperchain 的应用服务暂时不

支持上传公钥模式。

5、 上传公钥模式下应用接入密钥生成方法说明

如果您参与的应用服务使用上传公钥模式，您需要在本地客户端自主生成应用接入密钥的公私钥对，在本地自行保存私钥，并在 BSN 门户上传公钥。建议使用 OpenSSL 最新版本生成密钥，如果应用服务的框架是 Fabric-secp256r1 请使用 **prime256v1** 密码算法标识，Fabric-sm2、FISCO-sm2 和 XuperChain-sm2 使用 **sm2**，FISCO-secp256k1 则使用 **secp256k1**，具体步骤如下：

准备工作：从 https://www.openssl.org/source/下载 openssl 最新版本；创建一个 data.txt 文件，里面输入一些测试字符串，如：Hello world

终端输入 openssl 进入 open ssl 命令行

OpenSSL>

输入命令 ecparam -name **prime256v1** -genkey -out key.pem 生成一个私钥文件 key.pem

OpenSSL> ecparam -name **prime256v1** -genkey -out key.pem

输入命令 ec -in key.pem -pubout -out pub.pem 用 key.pem 文件中的私钥生成一个公钥文件 pub.pem

OpenSSL> ec -in key.pem -pubout -out pub.pem

read EC key

writing EC key

输入命令 dgst -sha256 -sign key.pem -out signature.bin data.txt 用 key.pem 文件中的私钥对 data.txt 文件进行签名生成签名文件 signature.bin

OpenSSL> dgst -sha256 -sign key.pem -out signature.bin data.txt

输入命令 dgst -verify pub.pem -sha256 -signature signature.bin data.txt 用 pub.pem 文件中的公钥对 data.txt 和 signature.bin 文件进行签名验签

OpenSSL> dgst -verify pub.pem -sha256 -signature signature.bin data.txt

Verified OK

如果显示 Verified OK，输入命令 base64 -in signature.bin -out signature64.txt 将签名文件 signature.bin 转换成 base64 编码的 signature64.txt 文件

OpenSSL> base64 -in signature.bin -out signature64.txt

输入命令 pkcs8 -topk8 -inform PEM -in key.pem -outform PEM -nocrypt -out keypkcs8.pem 将 key.pem 文件中的私钥转为 pkcs8 格式

OpenSSL> pkcs8 -topk8 -inform PEM -in key.pem -outform PEM -nocrypt -out keypkcs8.pem

本地自行保存好 keypkcs8.pem 文件，并在 BSN 参与应用该服务时将 pub.pem、data.txt 和 signature64.txt 中的全部内容对应复制到上传公钥模式页面的公钥、测试数据和签名数据文本框中进行验证并提交。

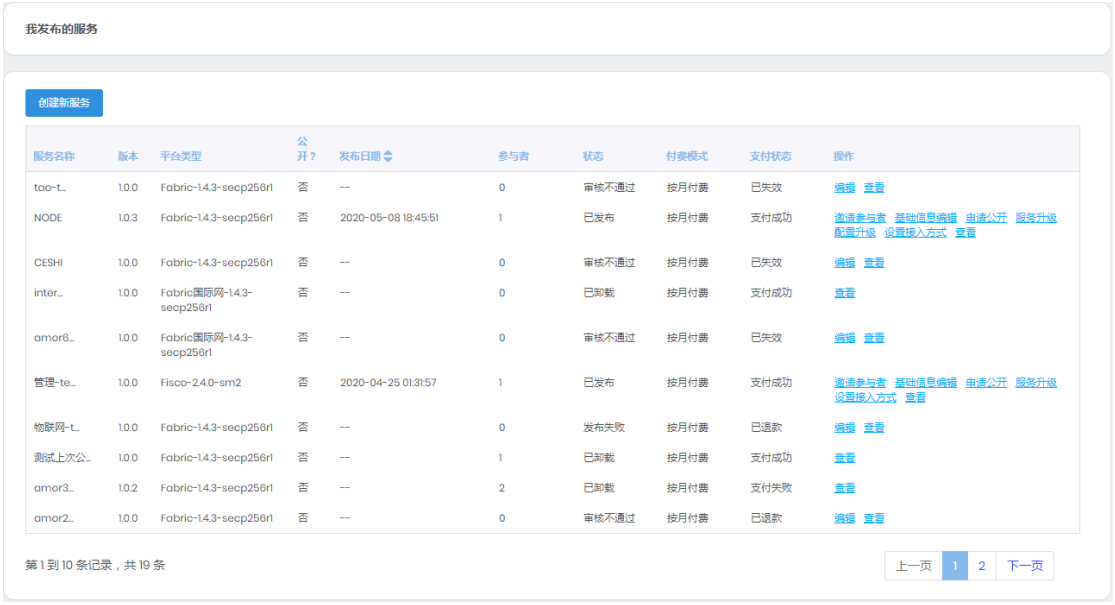
5.3 应用服务发布和参与

5.3.1 应用服务管理

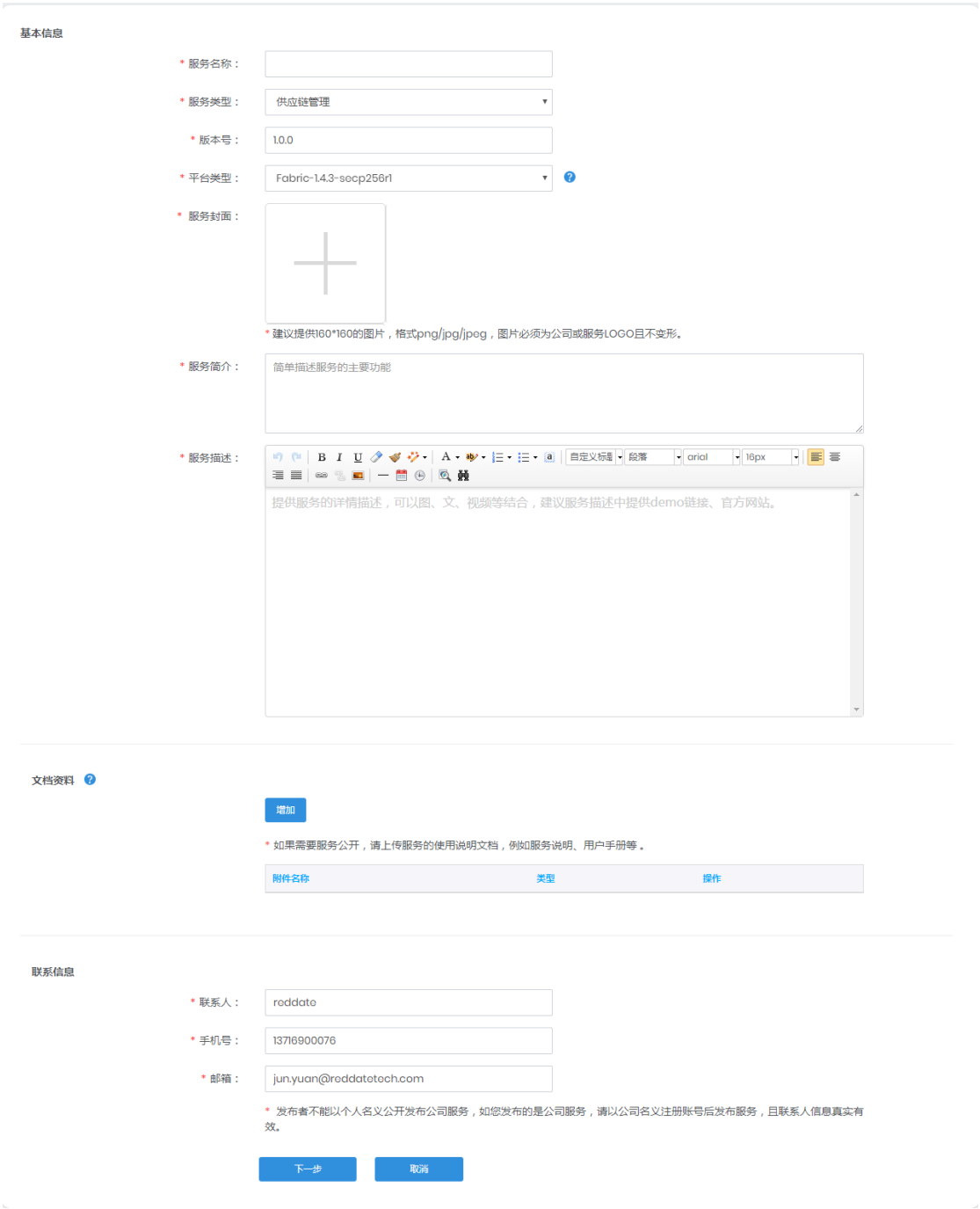
5.3.1.1 应用服务发布

服务发布的操作步骤如下：

- 创建新服务
  - 点击【联盟链服务】->【我发布的服务】页面发布服务；



  - 点击创建新服务，进入创建新服务页面，根据提示输入相应的信息。其中，在发布新服务时可以不上传文档资料，但若后续想要公开服务，则文档资料必须上传。



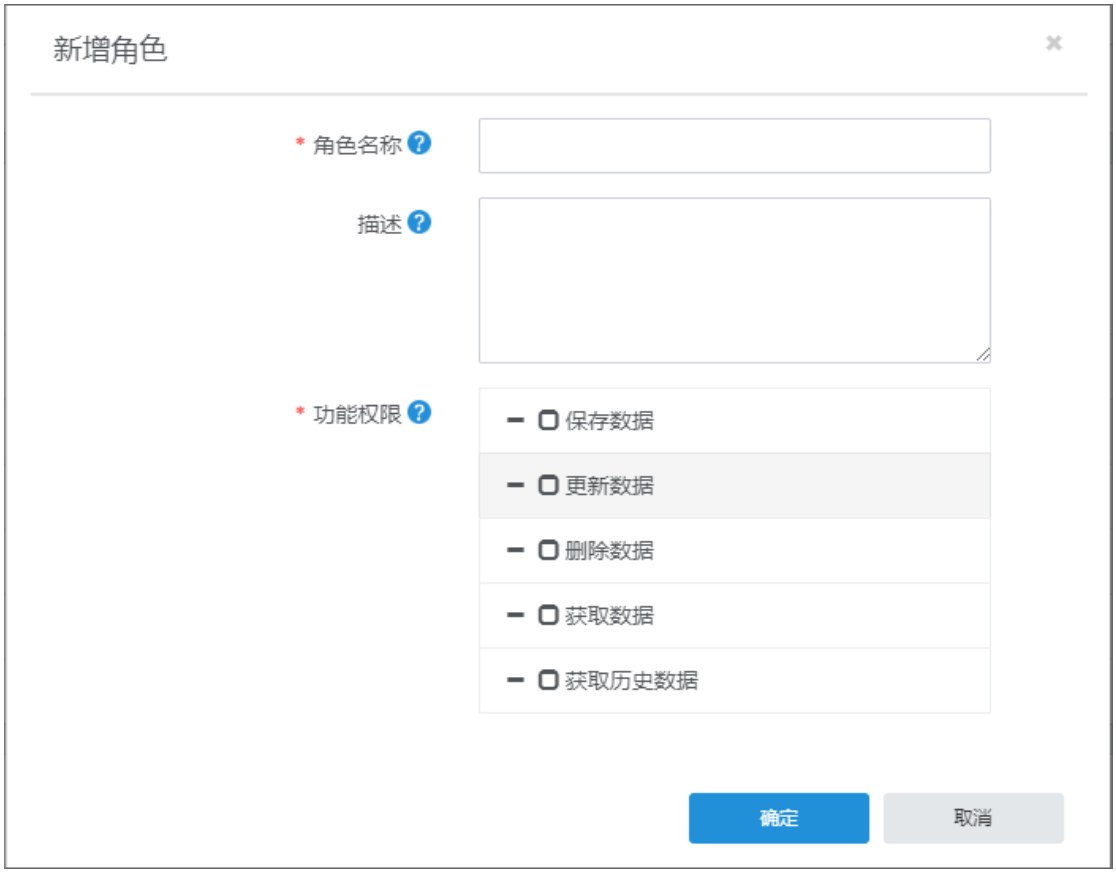
- 上传链码包
  - 一个服务可以上传多个链码包，用户可在上图中点击【下一步】开始上传链码包，如下图：

- 根据页面进行输入，并点击【确定】来完成链码包的上传。

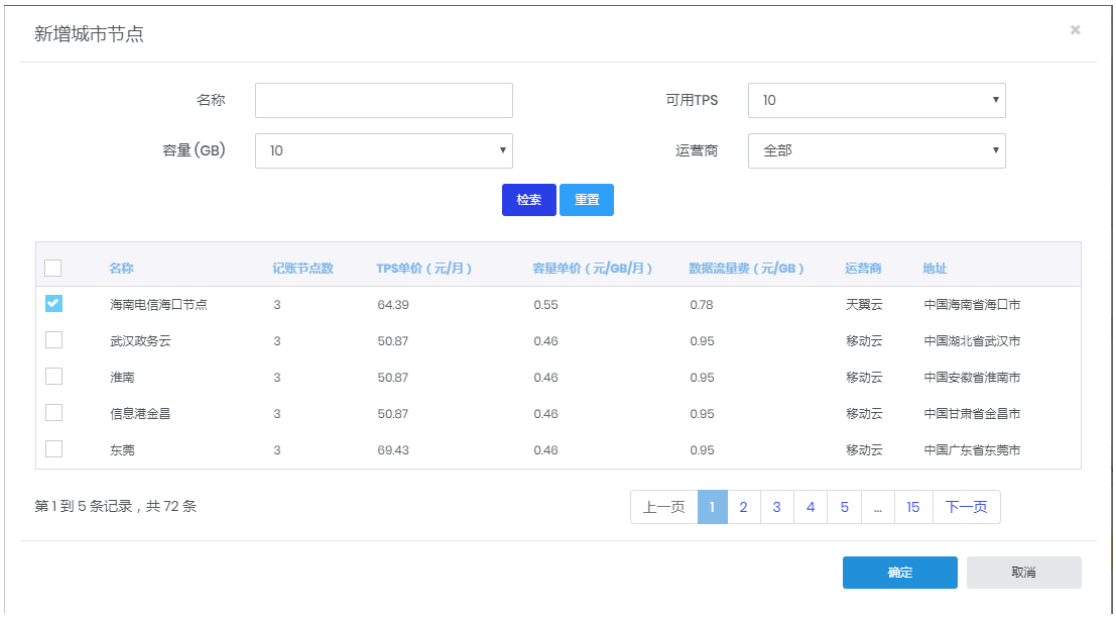
➢ 定义服务功能及角色

- 服务功能是指链码包包含的功能，服务发布完成以后，发布者通过功能和角色为服务参与者设置访问权限。如下图：

- 角色指整个服务包含的用户角色，用于为服务参与者设置权限，服务发布完成后可以修改定义角色如下图：



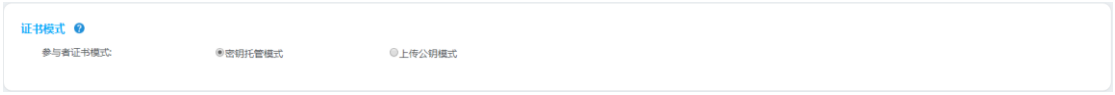
- 选择发布的城市节点
  - 指服务将部署在哪些城市节点上，服务发布完成以后，参与者从这些城市节点上参与该服务；服务发布完成以后可以修改；如下图：



- 发布者可以通过搜索来选择匹配的城市节点，勾选后点击【确定】完成选择；
- 发布者至少需要选择 3 个记账节点，建议这 3 个记账节点分布在不同的城市节点上，否则具有一定的安全风险。

➢ 选择参与者证书模式

- 证书模式用于指定服务参与者在使用服务时的“应用接入密钥”和“用户交易密钥”的生成方式为：密钥托管模式还是上传公钥模式，如下图：



➢ 支付账单、提交审核并发布

- 完成链码、功能、角色、城市节点、证书接入模式之后，继续点击【下一步】，进入到【服务账单】页面，如下：

- 点击【确定】后，系统生成账单，并提示用户支付资源使用费，用户确定以后，系统从用户的账户中进行扣款，扣款成功或失败都将进入服务审核环节。若服务扣款成功并审核通过，系统将对服务进行部署，部署成功之后服务发布完成。若服务扣款失败，服务对应的账单将保留 72 小时，过期失效。若仍想发布服务，需重新提交服务并支付；

- 如果用户账户内没有余额或余额不足，需要先进行充值。然后到【我的账单】或者【我发布的服务】进行支付；
- 服务的费用构成包括 TPS、容量、记账节点数（流量使用费根据服务的实际使用量按周扣款，服务发布时无须支付），按年支付时具有一定的折扣优惠；
- 注意：发布者默认不参与自己发布的服务，如需参与，需点击列表中的【参与应用】进行申请，并在【我发布的应用->服务参与管理】中自行审核。

5.3.1.2 邀请参与者

服务发布成功之后，发布者可以通过【邀请参与者】功能来邀请区块链服务网络中其它的用户来参与。

邀请参与者步骤如下：

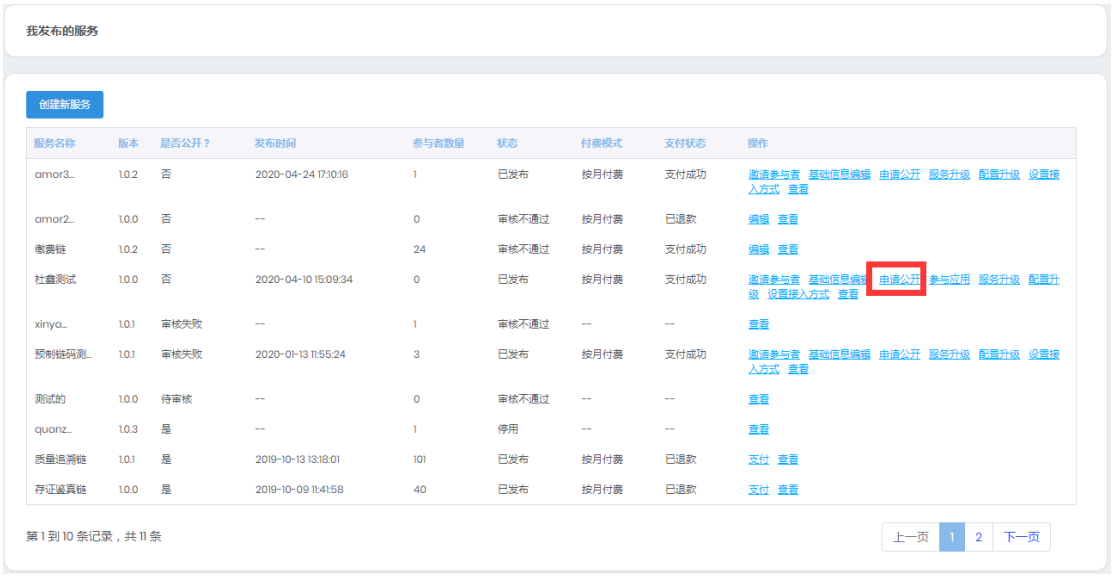
➢ 进入【联盟链服务->我发布的服务】页面，选择状态为【已发布】的服务，点击【邀请参与者】，显示如下：

➢ 点击【复制】即可复制邀请链接，并可以将邀请链接通过短信或微信等渠道发送给被邀请的用户；被邀请的用户点击链接并登录，即可在区块链服务网络门户中参与此服务。如果收到邀请的人员还不是服务网络的用户，则需要先进行注册，然后才能参加服务。

5.3.1.3 服务公开申请

服务发布者发布的服务默认是非公开服务，用户若要公开服务需要另行申请。操作步骤如下：

- 在【联盟链服务】->【我发布的服务】页面选择已发布且未公开的服务，点击【申请公开】，如下图：



- 在系统弹出的提示中点击【确定】即可申请公开服务，如下图：

- 用户申请公开服务后，需经运营人员审核通过方可公开，公开的服务将显示在应用商店中，其它用户可以自行申请参与。

5.3.1.4 服务升级

对于已经发布完成的服务，发布者可以通过服务升级功能来更新服务的链码和功能。服务升级不需要支付额外的费用，经运营系统审

核和部署以后完成升级。升级完成之前，原版本继续正常运行，升级完成后自动停用旧版本，并切换运行新版本。

服务升级的步骤如下：

- 在【联盟链服务】->【我发布的服务】页面选择已发布状态的服务并点击【服务升级】进入服务升级面面；
- 在服务升级页面中更新服务基本信息、链码信息和功能角色信息，并点击【确定】来申请服务升级；
- 经运营人员审核通过和部署成功后完成服务升级。

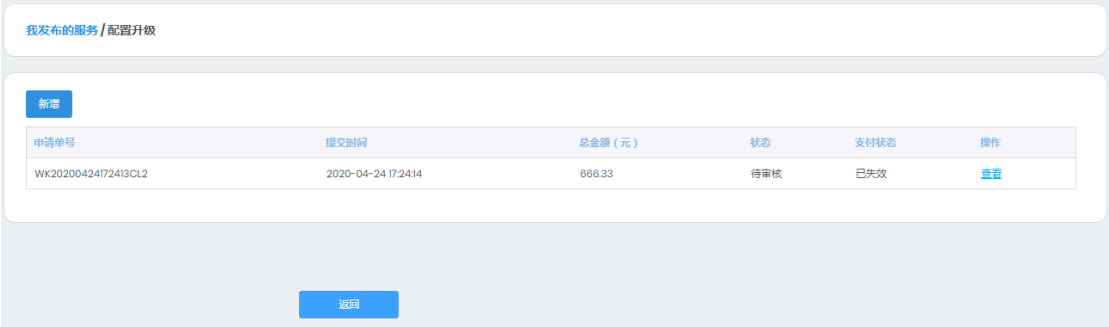
5.3.1.5 配置升级

发布者可以通过使用【配置升级】功能来升级服务的城市节点及城市节点中的 TPS、容量和记账节点。配置升级时需要支付相应的资源升级费用。

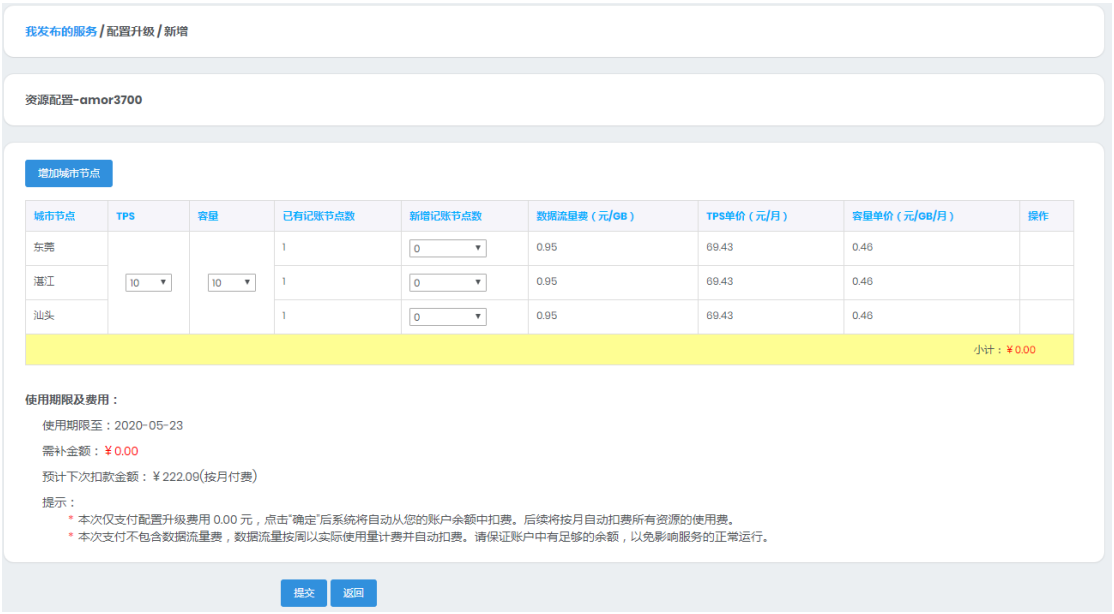
配置升级时不能删除原来的城市节点和下调原城市节点的配置，只能在原配置的基础上升级。

配置升级的操作步骤如下：

- 在【联盟链服务->我发布的服务】中选择已启用状态的服务，点击【配置升级】进入配置升级列表页面，如下：



- 点击【新增】创建配置升级申请单，并在页面中点击【增加城市节点】来增加服务发布的城市节点，如下：



- 点击【提交】按钮来提交配置升级申请，提交时系统提示需要支付相应的配置升级费用，发布者确认以后系统生成配置升级账单，并从用户个人（或企业）账户中扣款，无论扣款成功或失败配置升级申请都将进入审核流程，并等待运营人员审核。若扣款成功并审核通过，系统将进行配置升级处理，处理成功以后完成配置升级；若扣款失败则账单将保留 72 小时，过期失效，若仍想升级配置，则需要重新申请；

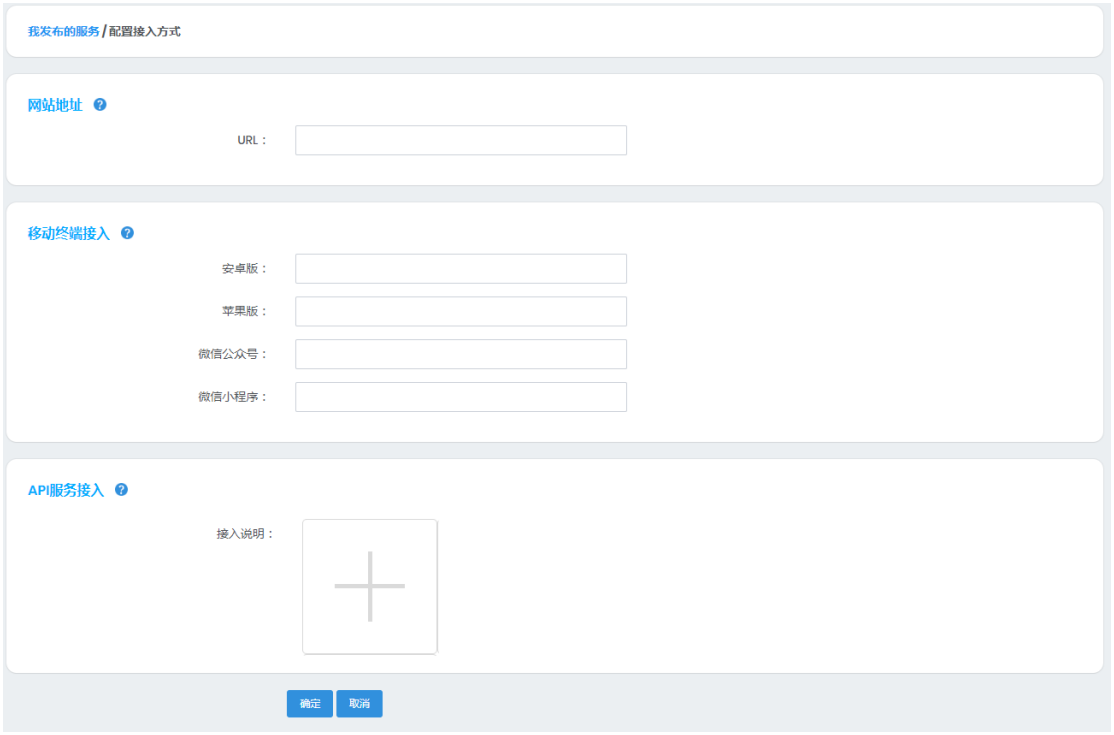
  注意：配置升级时支付的费用为升级费用，即补足升级前配置和升级后配置在账单周期内剩余时间的差额。升级成功以后，下个周期起将按新配置进行周期扣款。

5.3.1.6 设置接入方式

可以设置启用的服务的接入方式，包括网站地址、移动终端接入、API 服务接入三部分。其中，网站地址用户设置服务的访问地址；移动终端接入用于设置服务的移动版本、微信公众号和微信小程序的下载和访问地址；API 服务接入用于上传 API 服务的接入说明。服务的接入方式的设置是让服务发布者为参与者提供详尽的应用服务的信息和文档。

设置接入方式的步骤如下：

- 在【联盟链服务->我发布的服务】页面中选择已启用的服务，并点击【设置接入方式】进入设置接入方式页面，如下：



- 根据页面中的输入提示输入相应的内容并点击【确定】即可完成设置。

5.3.1.7 停用和启用

服务发布成功之后，系统将根据支付周期为服务生成资源使用账单，并根据服务实际使用的流量情况按周生成流量使用账单。当账单生成时，系统将从用户的账户中自动扣款，若扣款失败且账单超过 72 小时未支付则系统将自动停用服务（在服务的停用期间，由于其仍然占用服务资源，系统仍会按周期生成资源使用账单）。服务停用以后，链上数据无法访问。

已停用的服务可以在【我发布的应用->我发布的服务】页面进行启用，但启用时需补缴服务停用期间产生的所有欠费账单。启用服务时，无须重新审核。

5.3.1.8 应用服务运行监控

应用服务运行监控主要负责收集、处理、汇总系统中的运行状态数据，并以可视化方式将系统的实时量化数据呈现出来。

- 在【联盟链服务->我发布的服务】页面中，选择状态为【已发布】的服务，点击【查看->运行信息】可以查看当前服务的运行信息，页面如下：

基本信息 链码及部署 服务角色 审批记录 接入方式 运行信息 评论/询问 历史版本

缴费链
1.0.0
供应链管理
北京红枣科技有限公司
使用区块链技术实现缴费信息的发布、检索和交易，提升缴费业务的结算效率和降低错误率。

接入用户数
24

城市节点(3)： 杭州 长沙 泉州

资源类型	运行参数	负载状态
TPS	10	轻度
容量	10 G	轻度

10 块
9 交易数
3 记账节点数
3 链码数

记账节点

节点名称	运行状态	容量（已用/总容量）
peer1.hangzhounode.bsnbase.com	运行中	142336GB/10GB
peer1.quanzhounode.bsnbase.com	运行中	4812GB/10GB
peer1.changshanode.bsnbase.com	运行中	4608GB/10GB

区块Hash

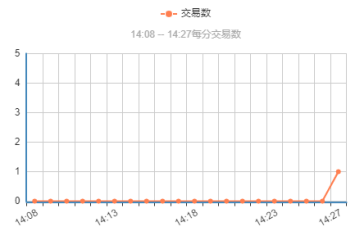
HashCode	上链时间
fa5b0f08732b6d33...	2019-11-01 14:27:23
0a8ad9e11d1ba8ea...	2019-11-01 09:29:59
6cae3ddab85dead7...	2019-11-01 08:57:10
e41fc321877fa767...	2019-10-31 17:33:40
a252abb830b39251...	2019-10-22 15:19:19
ca306e7a0aa2c88f...	2019-10-12 17:50:07
738fe1204f70489a...	2019-10-12 17:47:21
763ebb3c1683160d...	2019-10-12 17:46:26
3d8c3a85fbe2d519...	2019-10-08 13:57:43

流量使用情况（最近30天）

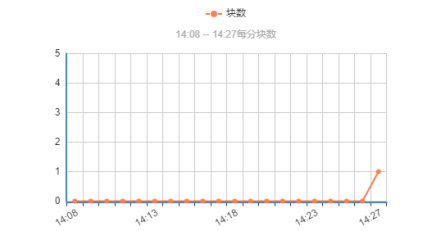
日期	城市节点	流量总计

交易数/s（TPS）

交易数/Min（TPM）



块/Min



返回

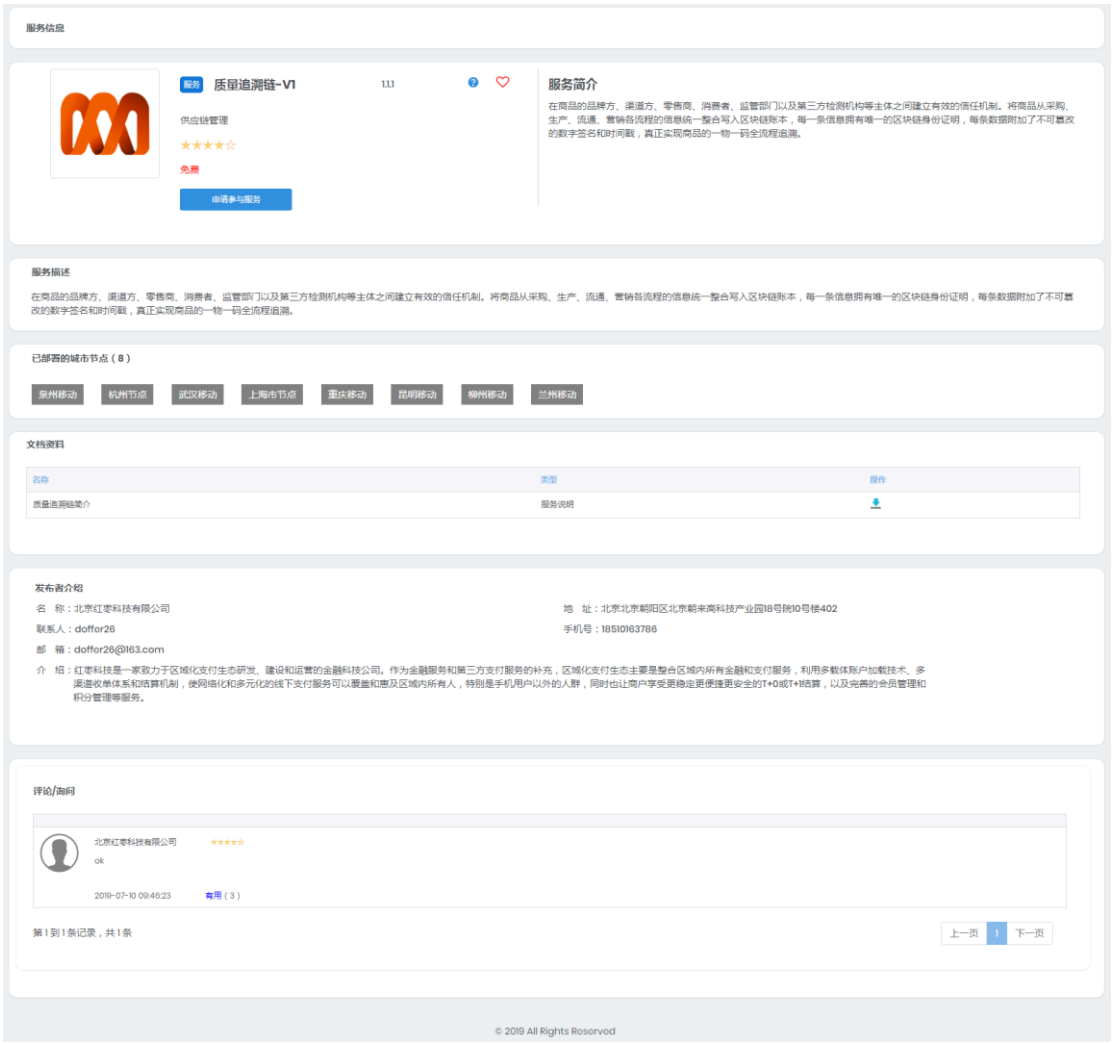
5.3.2 应用服务参与

5.3.2.1 应用服务参与

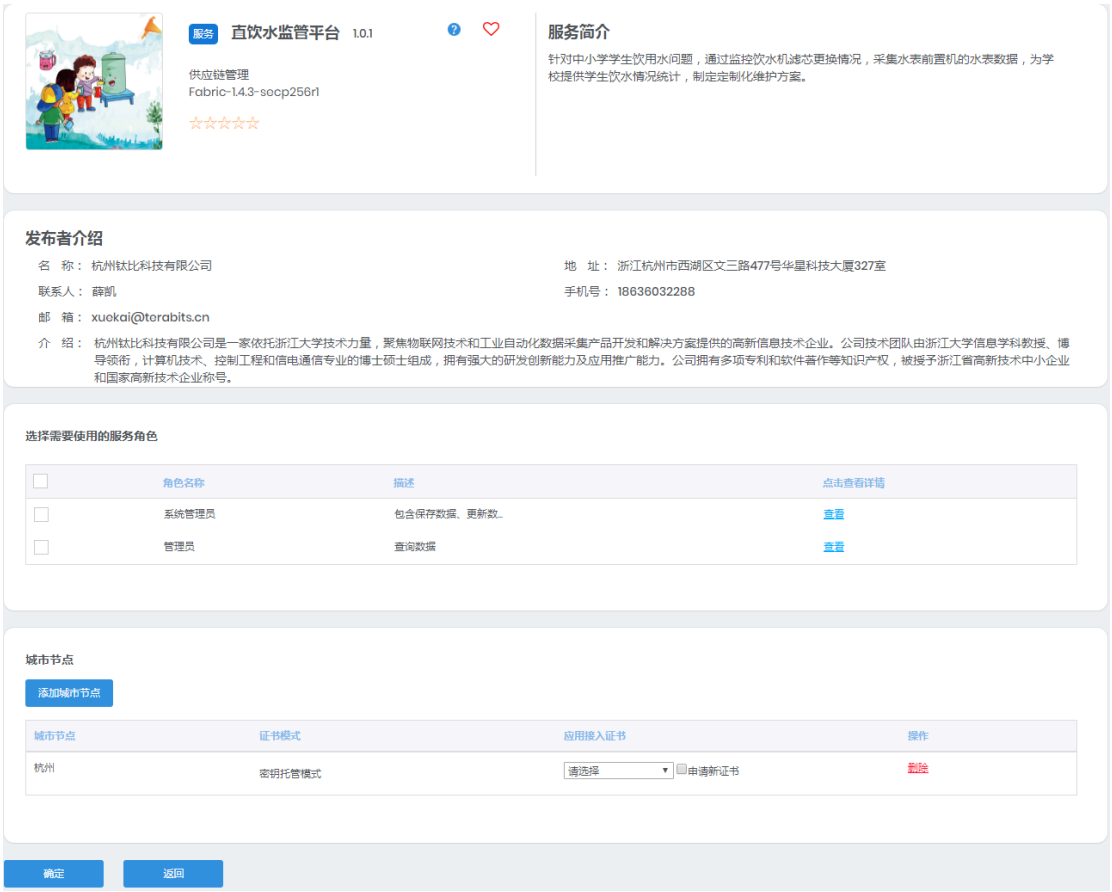
1、 应用服务参与

用户实名认证通过后，即可参与区块链服务网络中所有已发布并公开的服务。操作步骤如下：

- 在【联盟链服务】->【我参与的服务】页面或【应用商店】，选择一个未参与的应用去参与:

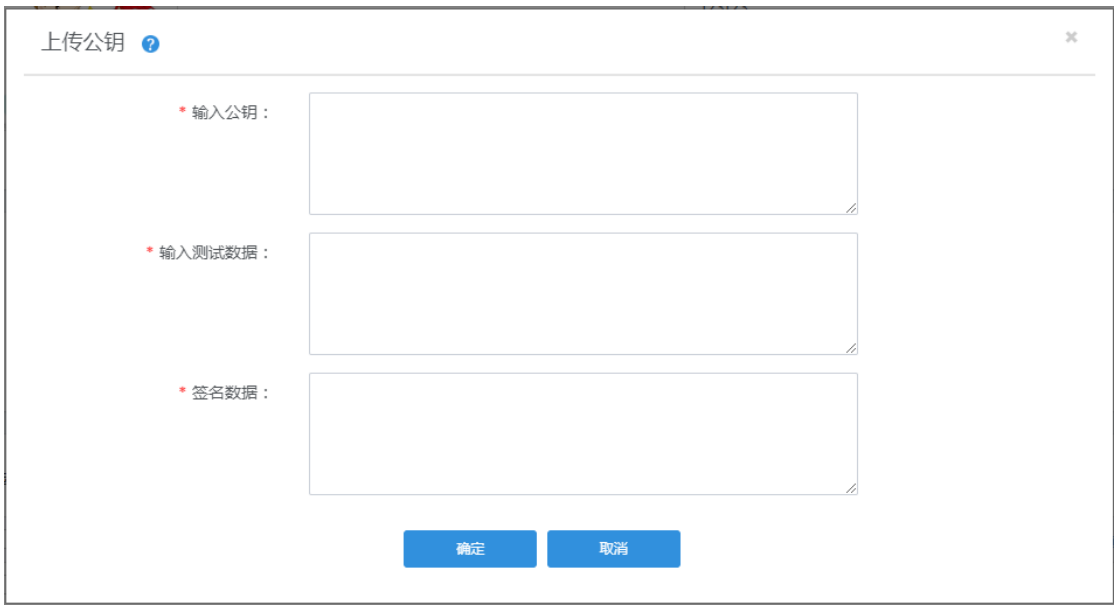


- 点击【申请参与服务】，进入选择应用角色及城市节点页面：



- 选择需要使用的应用角色和城市节点（角色是指用户参与服务所需使用的功能；城市节点是指用户从哪个城市来参与该服务）。根据服务发布者的设置，应用参与时有两种证书模式：上传公钥模式和密钥托管模式，其中：

  上传公钥模式：需要输入公钥、测试数据和签名数据，公私钥的生成可以点击【查看公/私钥生成说明】进行查看；



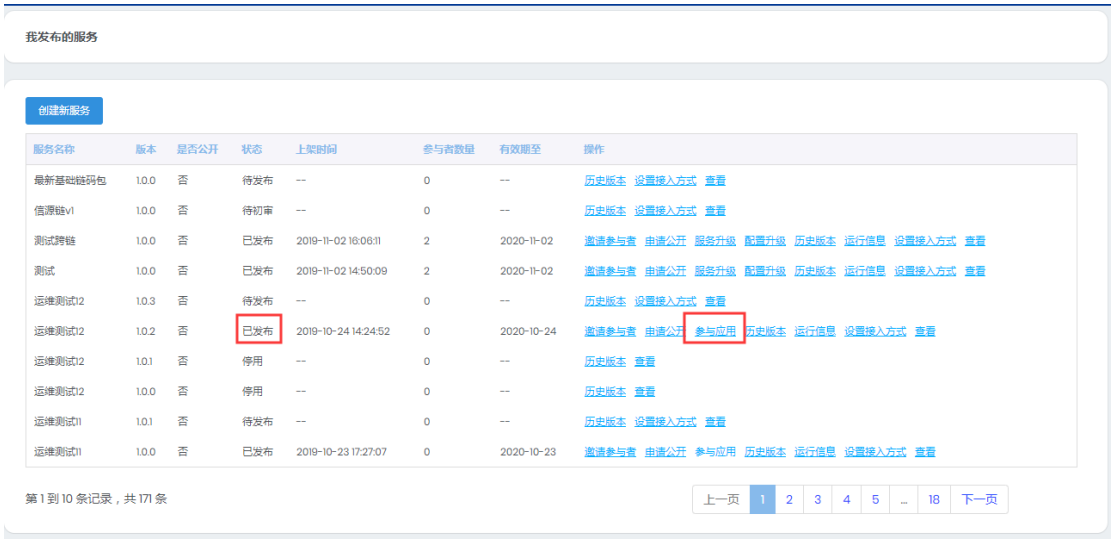
密钥托管模式：可以选择该城市节点上已有的证书或者申请一个新的证书。

- 点击【确定】向服务发布者发起参与申请，并等待服务发布者审核；
- 服务发布者对服务参与申请审核通过后，服务参与信息的状态更新为“已确认”。至此，参与应用服务成功。

注意：非公开的服务无法自由申请参与，只能通过发布者主动邀请其他用户参与。

2、 参与自己发布的服务

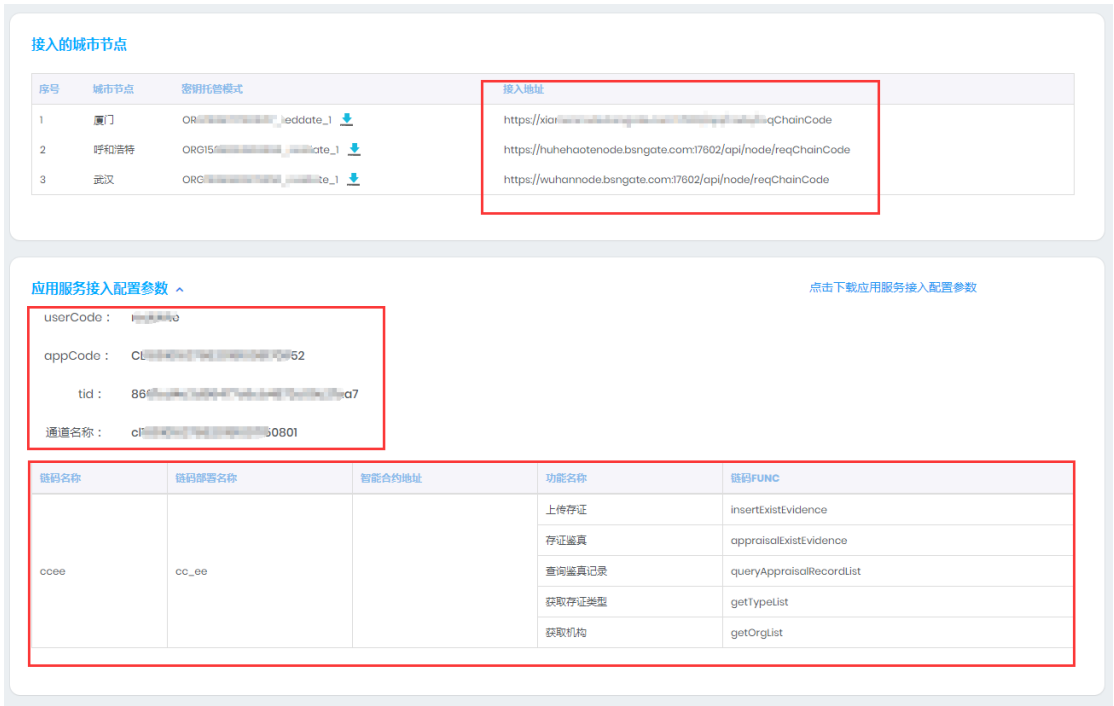
服务发布后，发布者默认不参与自己发布的服务，如需参与，则可以在【我发布的服务】列表中点击【参与应用】来参与自己发布的服务（具体步骤同上述应用服务参与步骤），如下图：



5.3.2.2 应用服务接入参数查看

用户参与服务成功以后，可以进入服务的查看页面来查看应用服务的接入参数，包括：城市节点接入地址和应服务接入配置参数，步骤如下：

- 进入【联盟链服务->我参与的服务】页面，选择“已确认”状态的服务，点击【查看】进入查看页面；

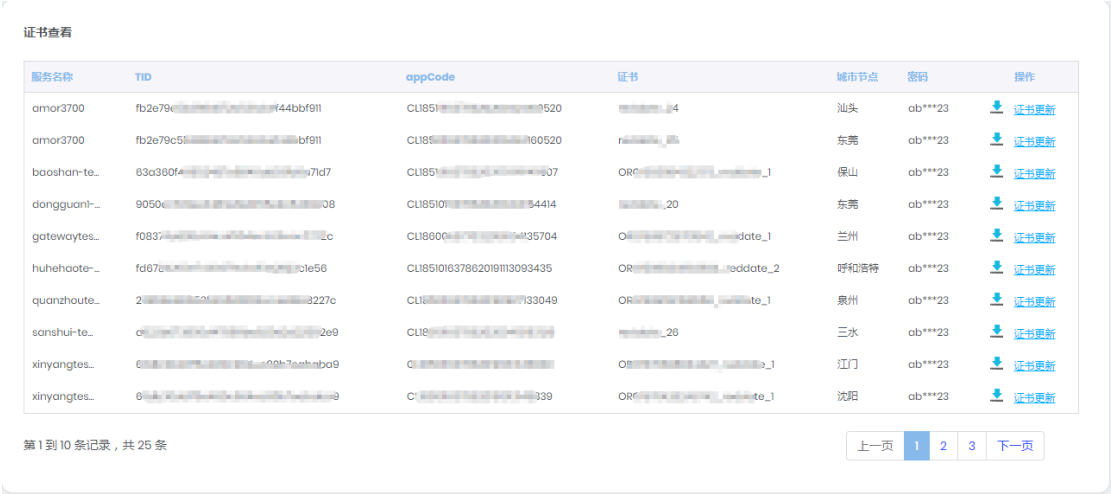


5.3.3 证书下载和更新

用户参与服务时，需要一个应用接入密钥对。根据服务发布者的设置，参与者可以使用密钥托管模式或者上传公钥模式来参与服务。服务参与成功之后用户可以下载证书或进行证书更新。

1、 证书下载步骤如下：

- 点击【联盟链服务->我的身份证书->我托管的证书】进入密钥托管模式下的证书查看页面，如下：

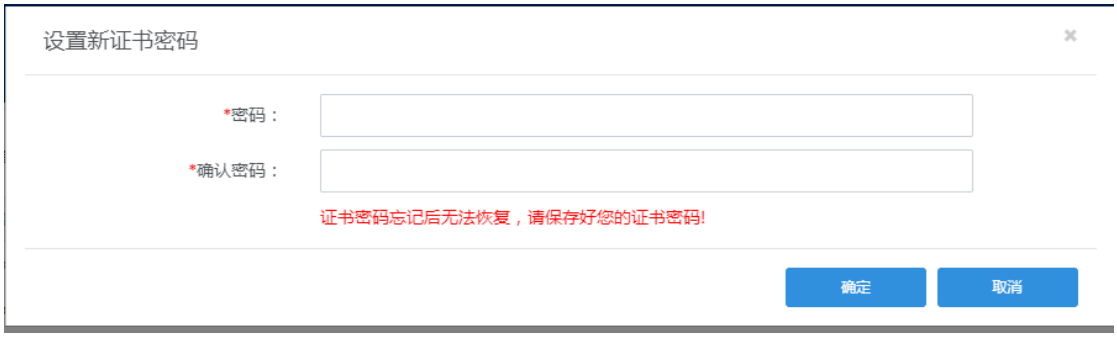


- 点击证书下载，并输入证书生成时设置的密码即可下载证书。

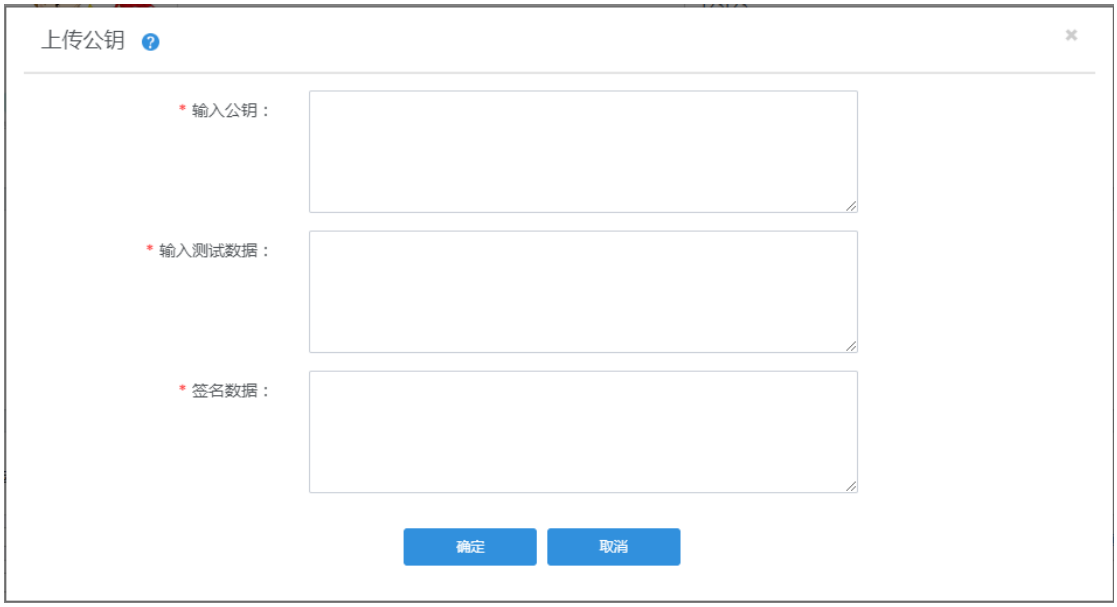
注意：由于使用上传公钥模式参与服务时，用户只需上传公钥，私钥由用户自行保管，所以无须下载证书。

2、 证书更新步骤如下：

- 点击【联盟链服务->我的身份证书->我托管的证书】或【联盟链服务->我的身份证书->我上传的证书】进入证书页面，点击【证书更新】来更新证书；如果是托管的证书，则更新时需要输入证书的新密码，如下：



- 如果是上传的证书，则更新时需要重新输入公钥、测试数据和签名数据，并在测试通过后才能完成更新，如下：



5.3.4 应用产品管理

1. 应用产品上架

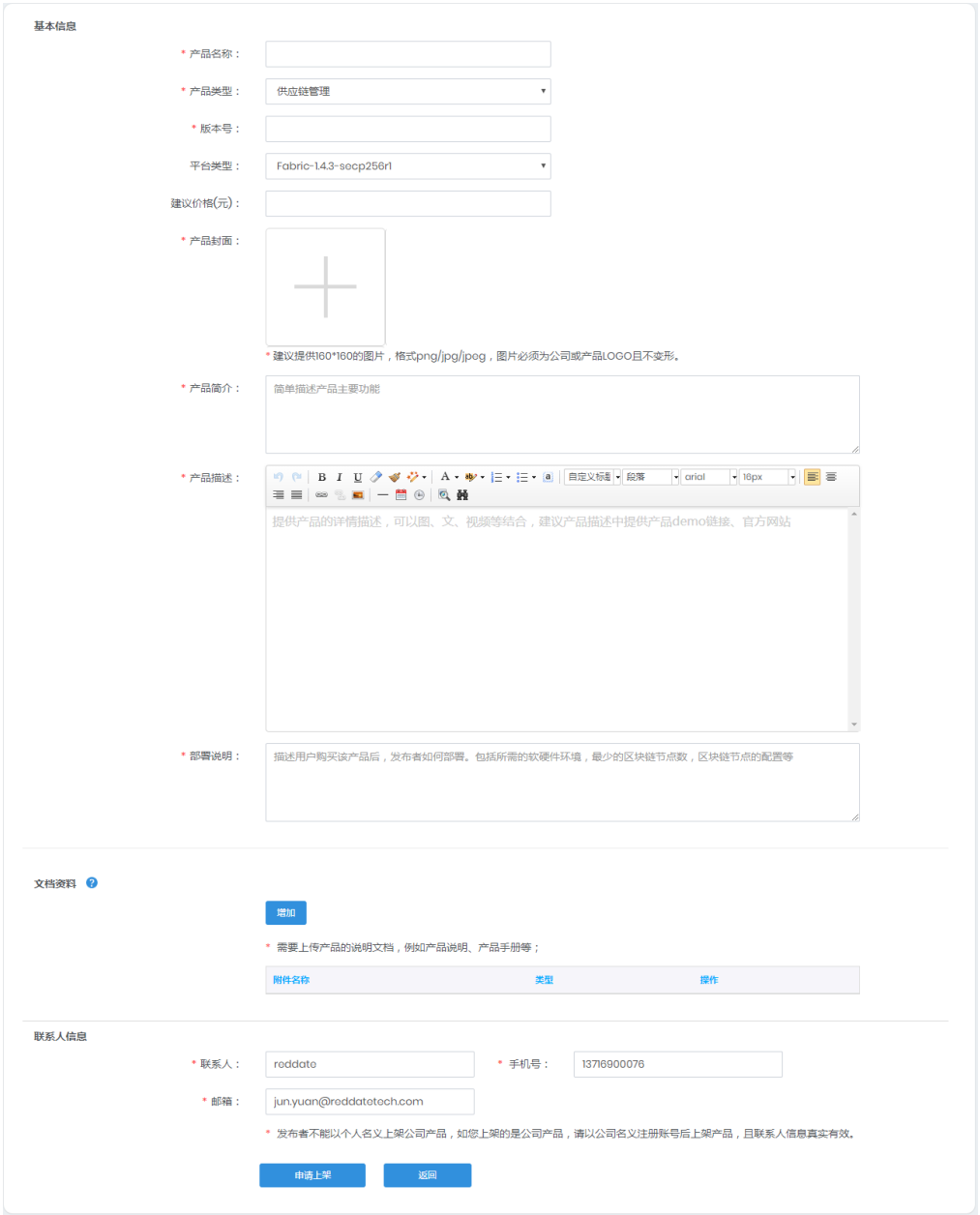
应用产品指没有在区块链服务网络中发布和运行，仅在服务网络门户的应用商店内上架的区块链应用。产品销售后，由购买方在服务网络上发布后方可以投入使用。产品发布完成之后即成为服务，此时购买方成为应用发布者，并可以邀请其他用户参与。产品的设置是让某些开发者仅以开发和销售产品为商业模式，而不需要投入成本去运营自己的产品（对比互联网，产品开发者类似提供网站搭建的开发者，而服务发布者类似经营网站的运营者）。

用户登录以后，可以申请上架产品。

应用产品上架操作步骤如下：

- 在【联盟链服务】->【我发布的产品】页面进行产品的发布：

- 点击创建新产品，进入创建新产品页面，根据提示输入相应的信



息：

- 点击【申请上架】，并确认支付上架费用，系统将生成账单并从用户账户中扣款。不论扣款成功或者失败，在【我发布的应用】->【我发布的产品】页面都显示该条新创建的产品，新创建的产品状态为“待审核”；

  若扣款成功并审核通过，则产品上架成功；若扣款失败，则账单保留 72 小时，过期失效，如要继续上架，还需重新申请；

- 产品上架后显示到应用商店中，用户可在【应用商店】查看到新上架的产品。

2. 应用产品下架

产品上架以后，系统将按周期生成账单，并从个人账户中扣款。若扣款失败，则账单保留 72 小时，过期失效。账单失效以后产品将被自动下架。

用户也可以通过页面中提供的【下架】功能来下架已上架的产品，产品下架以后可以修改产品信息并重新提交上架申请。

若重新提交上架申请时在账单周期之内，则审核通过后即可上架产品；若重新提交上架申请时超过了账单周期，则系统将重新生成账单，并在用户支付费用和产品审核通过后完成上架。

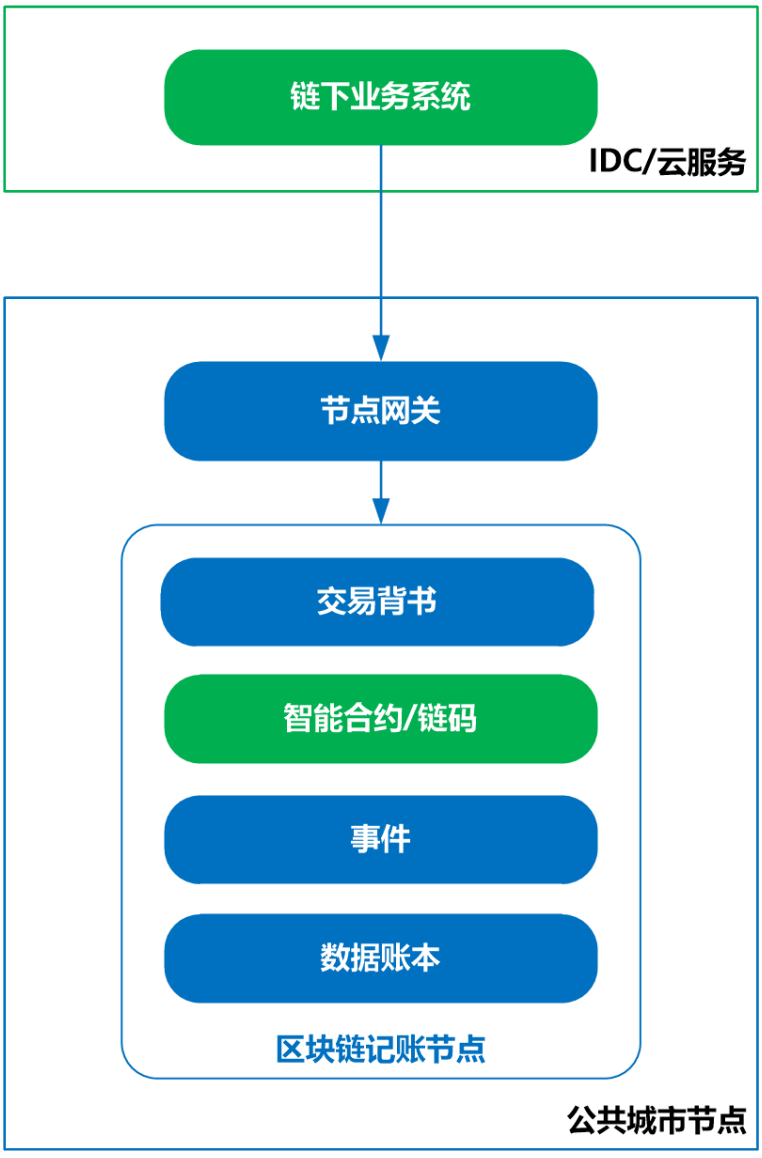
5.4 BSN 外系统接入指南

5.4.1 概述

区块链服务网络（BSN）是面向开发者提供基于区块链运行环境的基础设施网络，以互联网理念为客户提供公共区块链资源环境，帮助客户快速、低成本地开发、部署、运营和维护区块链应用（DApp）。

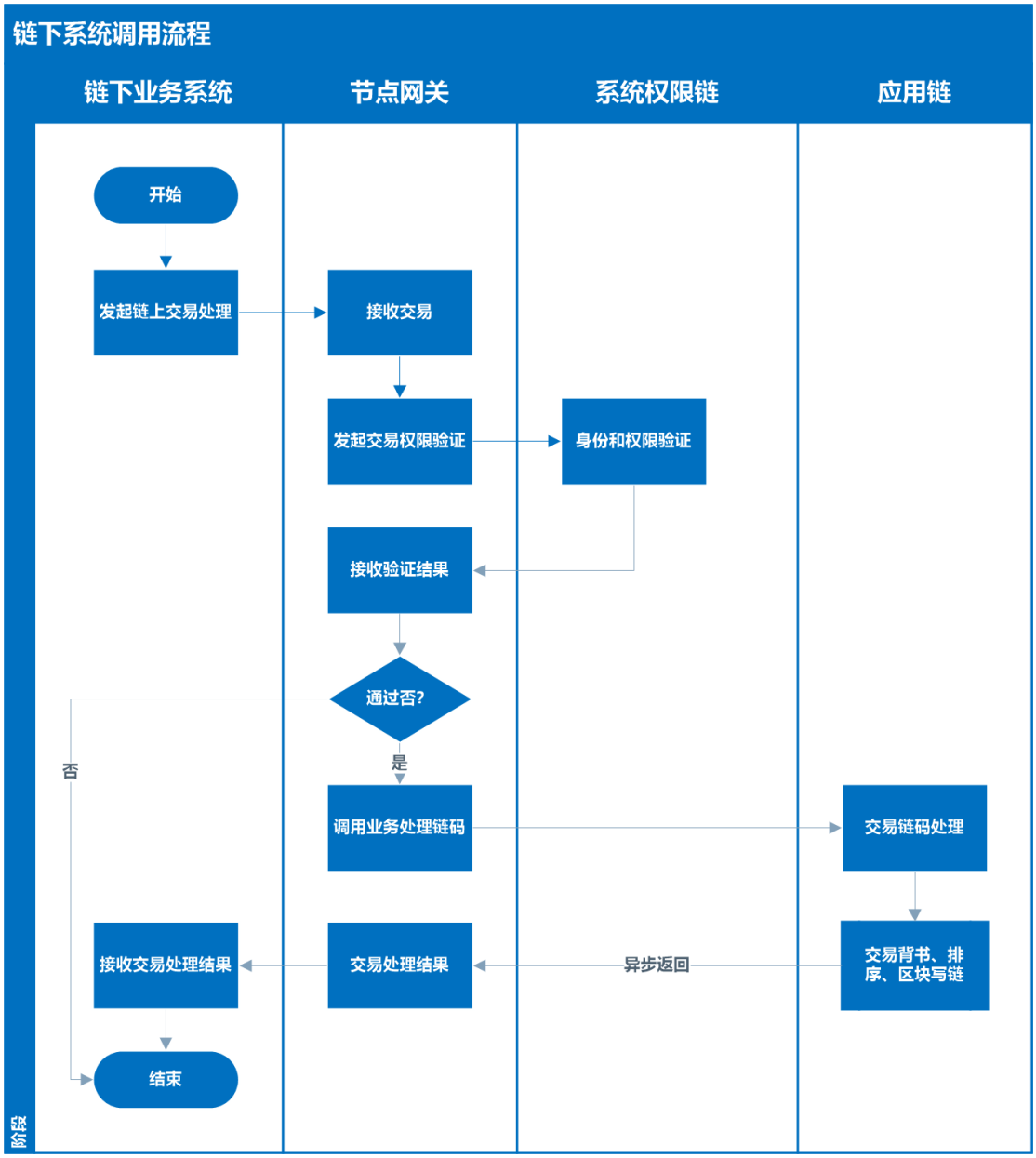
完整的区块链应用一般包含两部分：链上智能合约（应用服务链

码（DApp Service Chaincode））和链下业务系统/BSN 外业务系统（off-BSN system）。链下业务系统通过服务网络公共城市节点(Public City Nodes 或 PCN)网关(gateway)调用在城市节点上部署的应用服务链码进行链上交易处理、数据写链和数据查询等操作。服务发布者（Publisher）和参与方（Participant）可使用自己选择的 IDC 资源或云服务平台部署自己的链下业务系统，并通过互联网连接到城市节点网关访问应用服务链码。

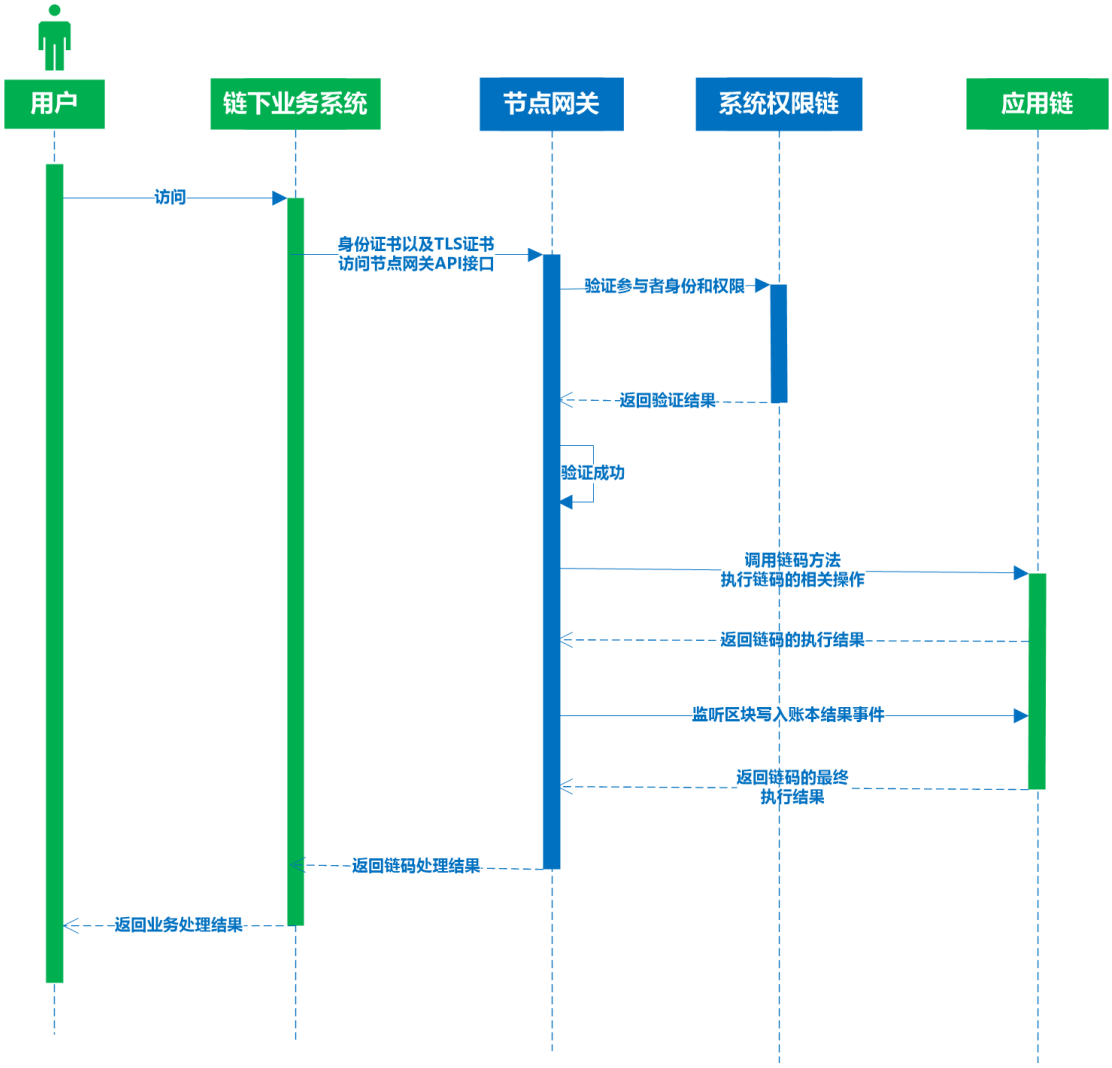


开发者需要自行开发应用的链下业务系统的业务逻辑，开发过程中可通过城市节点网关 API 来调用应用链码/智能合约执行交易事务处理、完成数据写链和数据查询等操作。

链下系统调用流程：



链下系统调用时序：



5.4.2 智能合约打包规范

链码（ChainCode）又称为智能合约，是用计算机语言描述合约条款、交易的条件、交易的业务逻辑等，通过调用智能合约实现交易的自动执行和对账本数据的操作。**一个区块链应用可以部署多个链码**，每个链码包含多个方法（Functions）。下面我们分别就 Hyperledger Fabric、FISCO BCOS 和 XuperChain 三个底层框架来描述在 BSN 上开发和部署智能合约的要求。

5.4.2.1 Hyperledger Fabric 链码包规范

Fabric 链码(智能合约)支持多种语言编写，包括 golang、java、node.js。每个链码程序都必须实现 Chaincode 接口，链码包含：Init、Invoke 和 Query 三个基本操作：

- **Init：**链码初始化的方法，在链码实例化或者升级的时候调用一次，以便链码可以执行任何必要的初始化，包括应用程序状态的初始化。
- **Invoke：**接收和处理链下业务系统调用事务处理提案，其参数包含调用的链码程序中函数的名称和具体业务处理数据参数，即在 Invoke 中根据不同的方法参数调用其他分支处理响应的业务。Invoke 可以简单的理解为链码方法的入口。
- **Query：**提供查询链码数据的方法，该方法只作为查询使用，不提供操作链上数据的操作。可在 Query 操作时调用，亦可在 Invoke 方法中作为某些方法的分支被调用。该方法可以不实现。

为实现自动化部署，提升部署效率，不同语言开发的链码在 BSN 发布时需按照以下规范要求来组织链码部署包。

1、 Golang

main 函数必须在项目中所有链码的上级或同级，打包路径为 main 函数所在同级文件夹，main 函数路径为基于 src 的引用路径。

例: bsnBaseCC 链码包（预置链码包）

bsnBaseCC

└─main.go

└─ChainCode/

└─models/

└─utils/

需在 bsnBaseCC/下进行打包（包名称无要求），main 函数路径（引用路径）为 bsnBaseCC。

例：github 上的 FabricBaseChaincode 链码包（预置链码包）

github.com

```
└─BSNDA
    └─FabricBaseChaincode
        └─chaincode
            └─go
                └─bsnBaseCC
                    └─main.go
                    └─ChainCode/
                    └─models/
                    └─utils/
```

需在 github.com/BSNDA/FabricBaseChaincode/chaincode/go/bsnBaseCC/下进行打包（包名称无要求），main 函数路径（引用路径）为 github.com/BSNDA/FabricBaseChaincode/chaincode/go/bsnBaseCC。说明：main.go 入口，ChainCode：链码；models：实体；utils：工具类

2、 Java

gradle 或 maven 构建的项目,项目中必须包含 build.gradle 或 pom.xml 文件。

例：bsnBaseCC 链码包（预置链码包）

```
bsnBaseCC
    └─build.gradle
    └─src
     └─main
```

```
└─java
  └─com.example.javacc
    └─javacc.java
```

需在 bsnBaseCC/下进行打包，zip 包名称无要求。

说明：src/main/java：项目目录，com.example.javacc：包名，javacc.java：链码信息

3、 NodeJs

必须在项目根目录下创建 package.json。需在 bsnBaseCC/下进行打包，zip 包名称无要求。

例：bsnBaseCC 链码包

```
bsnBaseCC
  └─marbles_chaincode.js
  └─package.json
```

说明：marbles_chaincode.js：链码

注：发布服务时，链码包打包时进入项目根目录进行打包，格式为.zip。

5.4.2.2 Hyperledger Fabric 预置链码包

预置链码包是我们为应用开发者提供对业务数据进行增删改查基本操作的链码（使用 Golang 语言编写）。应用开发者可以在此链码包的基础上根据应用业务需求进一步拓展链码功能。此链码支持存储的数据类型有字符串、整型、浮点型、集合（map、list）等。

下载地址:

https://github.com/BSNDA/FabricBaseChaincode

链码包功能如下：

1、 增加数据（set）

输入参数说明

baseKey：需要保存的唯一的主键标识

baseValue：保存的数据信息

例：{"baseKey":"str","baseValue":"this is string"}

其中 baseKey 是不能为空的字符串，baseValue 可以是任意类型的数据。如果 baseKey 已经存在，则直接返回已经存在，不能添加；如果不存在，则添加数据。

2、 修改数据（update）

输入参数说明

baseKey：需要修改的唯一的主键标识

baseValue：保存的数据信息

例：{"baseKey":"str","baseValue":"this is string"}

其中 baseKey 是不能为空的字符串，baseValue 可以是任意类型的数据。如果 baseKey 不存在，则无法更新，如果已经存在，则修改数据。

3、 删除数据（delete）

输入参数说明

baseKey：需要删除的唯一的主键标识的值

例："str"

其中 baseKey 的值不能为空，且必须存在，否则将无法删除。

4、 获取数据（get）

输入参数说明

baseKey：需要获取的唯一的主键标识的值

例："str"

其中 baseKey 的值不能为空，且必须存在，否则将无法获取到相应的信息。

5、 获取历史记录数据（getHistory）

输入参数说明

baseKey：需要获取的唯一的主键标识的值

例："str"

其中 baseKey 的值不能为空。响应结果：交易 Id（txId）、交易时间（txTime）、是否删除（isDelete）、交易信息（dataInfo）。

欢迎开发者们共享自己的通用链码作为服务网络预置链码包，与我们一起拓展服务网络的区块链应用支撑能力。

5.4.2.3 FISCO BCOS 智能合约包规范

为形成统一智能合约部署包规范，方便 BSN 运营方对各个应用进行安全扫描和审计以及方便运维人员在 BSN 上部署，针对智能合约部署包做出以下规定。

1、 Solidity 智能合约路径

单级文件夹里存放所有合约：包括应用合约、应用库和外部接口合约。合约引用方式都为：import “./XXXX.sol”

2、 合约部署文件说明（deploy.md）

deploy.md 主要是按照规范说明清楚合约如何初始化部署。

主要分为三个部分：

- 合约说明：规范合约说明，用于简单描述各个合约基本信息。
- 用户说明：规范用户说明，用于描述初始化部署过程中，各个交易签名用户的基本信息。
- 合约初始化步骤说明：规范合约初始化步骤说明，用于列举初始化部署的步骤，让运维人员能够按照文档完成部署。

3、 上传合约规范

上传链码包（智能合约包）时填写的链码名称（合约名称）要与主合约类名以及主合约的文件名一致。

例：BsnBaseContract 链码包（预置链码包）

BsnBaseContract

└─BsnBaseContract.sol

└─Table.sol

需在 BsnBaseContract/下进行打包，zip 包名称无要求。如主合约类名为 BsnBaseContract，主合约的文件名即为 BsnBaseContract.sol，链码名称（合约名称）必须填为 BsnBaseContract。

4、 BSN 适配 FISCO Solidity 版本说明

目前 BSN 适配的 FISCO BCOS 只支持 Solidity0.4.25 及以下版本。

5.4.2.4 FISCO BCOS 预置智能合约包

预置智能合约包选自 FISCO BCOS 提供的 Table.sol，可以为应用开发者提供对业务数据进行增删改查基本操作(Solidity 语言编写)。应用开发者可以在此智能合约包的基础上根据应用业务需求进一步

拓展合约功能。此合约支持存储的数据类型有 int256(int)、address 和 string，其中 string 不能超过 16MB。为了保证链上的性能所以没有对重复 base_id、base_key 做判断，需要链下业务系统自行处理;推荐一个 base_id 对应一个 base_key 和 base_value。

下载地址：

https://github.com/BSNDA/FISCOBaseContract

智能合约包功能如下：

1、 增加数据（insert）

输入参数说明

base_id：需要插入的主键标识

base_key：需要插入数据信息的 key

base_value:需要插入数据信息的 value

例：{"base_id":"1","base_key":1,"base_value":"this is string"}

其中 base_id、base_value 是不能为空的字符串，base_key 是 int256 类型的数据。

2、 修改数据（update）

输入参数说明

base_id：需要修改的主键标识

base_key：需要修改数据信息的 key

base_value：需要修改数据信息的 value

例 ： {"base_id":"1","base_key":"1","base_value":"this is string"}

其中 base_id、base_value 是不能为空的字符串，base_key 是

int256 类型的数据。如果 base_id 和 base_key 不存在，则无法更新，如果已经存在，则修改数据。

3、 删除数据（remove）

输入参数说明

base_id：需要删除的主键标识的值

base_key：需要删除数据信息的 key

例：{"base_id":"1","base_key":"1"}

其中 base_id 和 base_key 的值不能为空，且必须存在，否则将无法删除。

4、 获取数据（select）

输入参数说明

base_id：需要获取的主键标识的值

例：{"base_id":"1"}

其中 base_id 的值不能为空，且必须存在，否则将无法获取到相应的信息。

5.4.2.5 XuperChain 智能合约包规范

Xuperchain 链码(智能合约)支持多种语言编写，包括 c++、golang 等。每个链码程序都必须通过基类提供的 Initialize、PutObject、GetObject、DeleteObject 等基本操作来实现自己的业务逻辑。

为实现自动化部署，提升部署效率，不同语言开发的链码在 BSN 发布时需按照以下规范要求来组织链码部署包。

1、 C++

推荐使用 xuperchain 提供的 Xdev 开发工具进行项目初始化、编译、测试等功能。上传链码包（智能合约包）时只需将.wasm 文件压缩成 zip 包上传就行，zip 包名称无要求。

例: BsnBase 链码包（预制链码包）

BsnBase

├─src

├──bsnbase.cc

├──bsnbase.wasm

├─test

├──bsnbase.test.js

xdev.toml

说明：src 目录存放链码文件以及编译生成的 wasm 字节码文件，test 目录存放单测文件 。

注：发布服务时，链码包打包时进入项目根目录进行打包，格式为.zip。

5.4.2.6 XuperChain 预置智能合约包

该预置链码包采用 C++语言编写，依赖于 xuperchain 提供的头文件"xchain.h"以及使用 xdev 进行项目初始化、编译、测试等操作。

视频资源链接：

http://kb.bsnbase.com/webdoc/view/Pub2c908ad371c6396b01721851e1cc591c.html

文章资源链接：

https://xuperchain.readthedocs.io/zh/latest/development_manuals/XdevManual.html

包含增删改查基本数据操作，此链码支持存储的数据类型为 map。

下载地址：

https://github.com/BSNDA/XuperChainBaseContract

智能合约包功能如下：

1、 增加数据（insert_data）

输入参数说明

baseKey：需要保存的唯一的主键标识

baseValue：保存的数据信息

例：{"baseKey":"dev_001","baseValue":"hello"}

其中 baseKey 是不能为空的字符串，baseValue 可以是字符串类型的数据。如果 baseKey 已经存在直接返回已经存在，不能添加，如果不存在，则添加数据。

2、 修改数据（update_data）

输入参数说明

baseKey：需要修改的唯一的主键标识

baseValue：保存的数据信息

例：{"baseKey":"dev_001","baseValue":"hello world"}

其中 baseKey 是不能为空的字符串，baseValue 可以是字符串类型的数据。如果 baseKey 不存在，则无法更新，如果已经存在，则修改数据。

3、 删除数据（remove_data）

输入参数说明

baseKey：需要删除的唯一的主键标识的值

例：{"baseKey":"dev_001"}

其中 baseKey 的值不能为空，并且必须存在，否则将无法删除。

4、 获取数据（select_data）

输入参数说明

baseKey：需要获取的唯一的主键标识的值

例：{"baseKey":"dev_001"}

其中 baseKey 的值不能为空，并且必须存在，否则将无法获取到相应的信息。

5.4.2.7 CITA 智能合约包规范

为形成统一智能合约部署包规范，方便 BSN 运营方对各个应用进行安全扫描和审计以及方便运维人员在 BSN 上部署，针对智能合约部署包做出以下规定。

1、 Solidity 智能合约路径

单级文件夹里存放所有合约：包括应用合约、应用库和外部接口合约。合约引用方式都为：import “./XXXX.sol”

2、 合约部署文件说明（deploy.md）

deploy.md 主要是按照规范说明清楚合约如何初始化部署。

主要分为三个部分：

- 合约说明：规范合约说明，用于简单描述各个合约基本信息。
- 用户说明：规范用户说明，用于描述初始化部署过程中，各个交易签名用户的基本信息。

- 合约初始化步骤说明：规范合约初始化步骤说明，用于列举初始化部署的步骤，让运维人员能够按照文档完成部署。

3、 上传合约规范

上传链码包（智能合约包）时填写的链码名称（合约名称）要与主合约类名一致，建议主合约的文件名也一致。

例: CitaBsnBaseContract 链码包（预置链码包）

CitaBsnBaseContract

└─CitaBsnBaseContract.sol

需在 CitaBsnBaseContract/下进行打包，zip 包名称无要求。如主合约类名为 CitaBsnBaseContract，主合约的文件名即为 CitaBsnBaseContract.sol，链码名称（合约名称）必须填为 CitaBsnBaseContract。

4、 BSN 适配 CITA Solidity 版本说明

目前 BSN 适配的 CITA 只支持 Solidity0.4.25 及以下版本。

5.4.2.8 CITA 预置智能合约包

预置智能合约包选自 CITA 提供的 CitaBsnBaseContract.sol，可以为应用开发者提供对业务数据进行增删改查基本操作（Solidity 语言编写）。应用开发者可以在此智能合约包的基础上根据应用业务需求进一步拓展合约功能。此合约支持存储的数据类型有 bytes、bytes32。

下载地址：

https://github.com/BSNDA/CITABaseContract

智能合约包功能如下：

1、 增加数据（insert）

输入参数说明

baseKey：需要插入数据信息的 key

baseValue:需要插入数据信息的 value

例：

{"baseKey":"0000000000000000000000000000000000000000000000000000000000000008b",

"baseValue":"8b"}

其中 baseKey 是不能为空的 byte32。baseValue 可以是任意长度的 bytes。如果 baseKey 已经存在，则直接返回已经存在，不能添加；如果不存在，则添加数据。

2、 修改数据（update）

输入参数说明

baseKey：需要插入数据信息的 key

baseValue:需要插入数据信息的 value

例：

{"baseKey":"0000000000000000000000000000000000000000000000000000000000000008b",

"baseValue":"9a"}

其中 baseKey 是不能为空的 bytes32，baseValue 是 bytes 类型的数据。如果 baseKey 不存在，则无法更新，如果已经存在，则修改数据。

3、 删除数据（remove）

输入参数说明

baseKey：需要插入数据信息的 key

例：

{"baseKey":"0000000000000000000000000000000000000000000000000000000000000008b"}

其中 baseKey 的值不能为空，且必须存在，否则将无法删除。

4、 获取数据（retrieve）

输入参数说明

baseKey：需要插入数据信息的 key

例：

{"baseKey":"0000000000000000000000000000000000000000000000000000000000000008b"}

其中 baseKey 的值不能为空，且必须存在，否则将无法获取到相应的信息。

5、 根据 index 获取 baseKey

输入参数说明

index：需要获取的 baseKey 的序号

例：{"index":0}

其中 index 的值为 uint 类型的数据。

5.4.3 城市节点网关 Fabric API

城市节点网关部署在各个城市节点，用来接收链下业务系统的请求，使用应用接入密钥进行交易请求签名，向基于 Hyperledger Fabric 的应用的链码发起访问并返回链码的执行结果。城市节点网关的调用是通过向区块链服务网络的各个城市节点的网关服务发送 HTTP 请求来实现。节点网关负责验证用户身份信息、应用信息，通过用户身份信息和应用信息以及需要访问的链码、链码方法来传递链码参数、获取链码执行结果的服务通道。

5.4.3.1 应用接入签名算法

链下业务系统向城市节点网关发送交易请求时，需使用应用接入密钥对中的私钥对交易请求报文进行数字签名，节点网关接收到交易请求报文后，使用已上传或托管的密钥对中的公钥对该报文进行验签。只有在交易报文验签通过时，网关才会对交易请求报文进行后续的交易处理。具体签名算法如下：

1. 组装签名字符串

将请求参数按照文档参数表格中的顺序转换为字符串进行拼接，其中请求参数优先拼接 Header 中的 UserCode、AppCode；返回参数优先拼接 code、msg。然后再次按照文档参数顺序拼接 Body 中的参数。

2. 不同类型的转换格式

类型	规则	示例	结果
String	不转换	Abc	abc
Int/int64/long	十进制转换	-12	-12
Float	十进制转换，小数位参考备注	1.23	1.23
Bool	转换为“true”或者“false”	真	true
Array	按照参数顺序和类型拼接	[“abc”,”xyz”]	abcxyz
Map[key]value	按照顺序拼接 key 和 value	{“a”:1,”b”:2}	a1b2
Object	将对象内的属性按照文档循序依据上述格式转换	{“name”:”abc”,”secret”:”123456”}	abc123456

3. 签名规则

a) 针对使用 ECDSA（secp256r1）密钥算法的 Fabric 框架应用

- 哈希值计算：将按照上述规则转换后的待签名的字符串按照 UTF-8 编码做 SHA256 计算；
- 对哈希值获取签名：哈希值与私钥进行 ECDSA（secp256r1）加密签名计算；在部分语言的处理中（例如 C#、Java）如果使用 SHA256WithECDSA 进行签名则不需要上一步，该算法中已经取哈希计算。
- 将签名结果做 Base64 计算。

b) 针对使用国密算法的 Fabric 框架应用，使用以下规则

- 哈希值计算：将按照上述规则转换后的待签名的字符串按照 UTF-8 编码做 SM3 计算；
- 对哈希值获取签名：哈希值与私钥进行 SM2 加密签名计算；
- 将签名结果做 Base64 计算。

4. 示例

参数:
{"header":{"userCode":"user01","appCode":"app01"},"mac":"","body":{“userId”:”abc”,”list”:[“abc”,”xyz”]}}

结果：user01app01abcabcxyz

5.4.3.2 密钥证书模式

1. 密钥托管模式

如第五章描述，用户在参与应用时需要两个密钥对：应用接入密钥对和用户交易密钥对。在该模式下密钥对不需要用户在本地生成，而是由 BSN 生成并且托管，用户只需要从 BSN 门户下载使用即可。

- **应用接入密钥对：**用户在参与应用服务成功后，BSN 会生成一对与所参与应用的底层框架算法一致的公私密钥对托管在 BSN 中，

用户可以在 BSN 门户内我的证书列表下载该公私钥对，在其链下业务系统调用城市节点网关接口时需要使用该密钥对的私钥进行签名。节点网关将使用托管的公钥对签名进行验签。

- **用户交易密钥对：**是用户调用链码处理交易的身份。在密钥托管模式下，用户成功参与一个 Fabric 的应用服务后，BSN 将会创建一个默认的用户交易密钥对，用户使用自己的 UserCode 调用即可。如果用户因业务需要创建多个用户参与链上交易，可以调用注册用户接口生成一个子用户并将子用户的交易密钥对托管在城市节点中，在交易时使用子用户的 UserCode 即可进行交易处理。

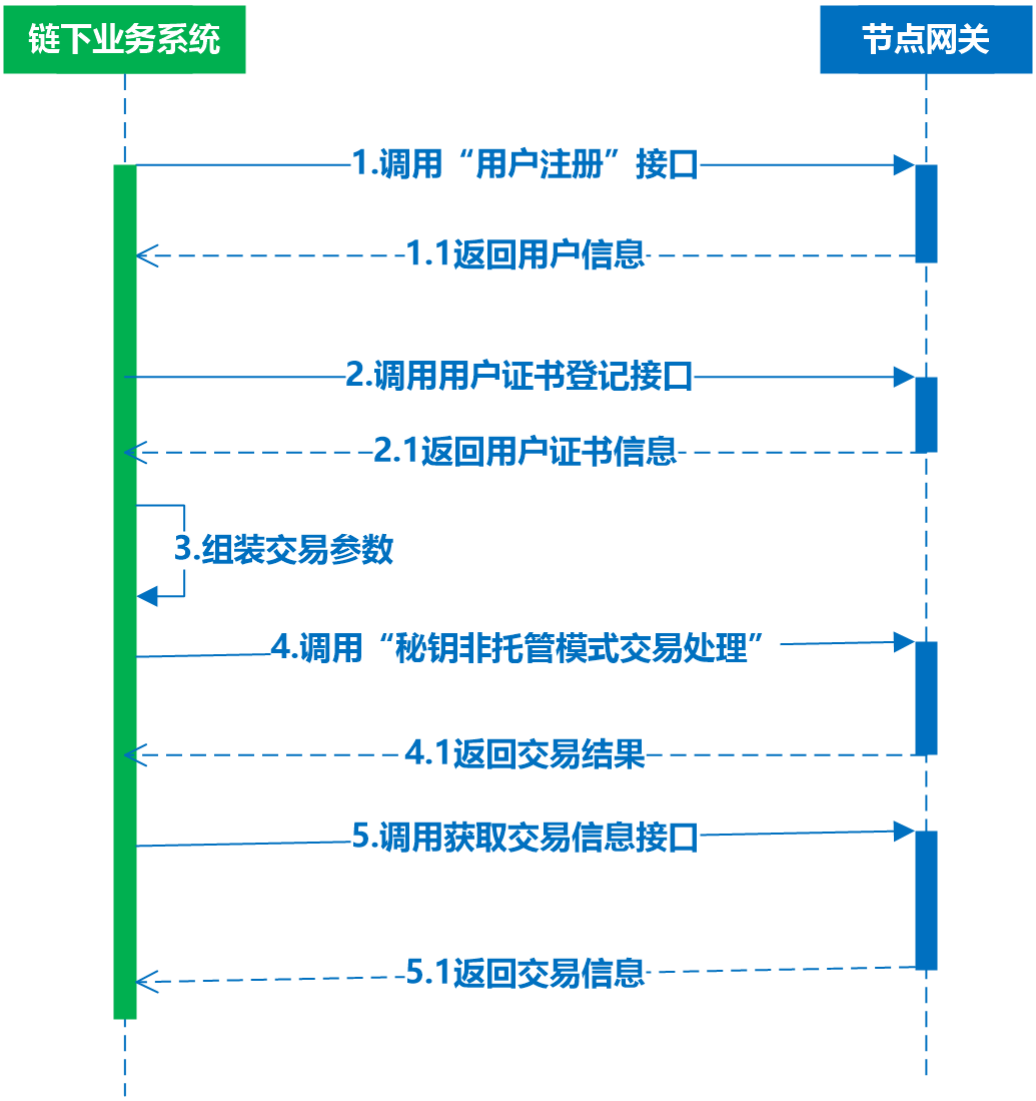
交易流程:

2. 上传公钥模式

用户在参与应用时需要两个密钥对：应用接入密钥对和用户交易密钥对。在该模式下用户需要自己在本地生成公私钥对，私钥本地保存，只需要将公钥或公钥证书申请文件上传 BSN。

- **应用接入密钥对：**用户在 BSN 门户参与应用服务时，需要在本地生成一对与该应用服务的底层框架算法一致的公私钥对，并且将公钥提交到 BSN 中。链下业务系统在调用城市节点网关接口时使用该密钥对的私钥进行交易签名。节点网关收到交易请求后使用用户上传的公钥对签名进行验签，以验证交易身份的合法性。
- **用户交易密钥对：**是用户调用链码处理交易的身份。在上传公钥模式下，用户需要在自己的链下业务系统中生成用户交易公私钥对，并使用公钥生成用户的公钥证书申请文件。链下业务系统通过城市节点网关的用户证书登记接口上传 BSN 进行登记，并获取由城市节点颁发的用户公钥证书。在调用用户登记证书接口之前，需要调用用户注册接口进行子用户账户的注册。

交易流程：



5.4.3.3 获取应用信息接口

调用该接口可以获取应用的基本信息，可以在上传公钥模式中使用。

1. 接口地址： https://节点网关地址/api/app/getAppInfo
2. 通讯方式： POST
3. 签名算法：不需要
4. 请求参数

序号	字段名	字段	类型	必填	备注

1	信息头	header	Map	是	
2	信息体	body	Map	否	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
示例：					
{"header":{"userCode":"USER00012020041519580108712 92","appCode":"app00012020041610 20152918451","tId":""},"mac":"","body":{}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：成功 -1：失败
2	响应信息	msg	String	是	
Body					
1	应用名	appName	String	是	
2	应用类型	appType	String	是	
3	应用密钥托管类型	caType	Int	是	1：托管 2：非托管
4	应用密钥类型	algorithmType	Int	是	1：SM2 2:ECDSA(secp256r1)
5	该城市 MSPID	mspId	String	是	
6	应用链名称	channelId	String	是	Fabric 对应 channelId，fisco 对应 groupId
示例					

{
"header": {
"code": 0,
"msg": "处理成功"
},
"mac":
"MEUCIQDE9zv0E/w4V/ILG6wUCFP08a7NDCAtX/IoZOcCyY4gIQIgUTYWsFTA1KE88gE6 452jKnnVBrhznGVOV2HPMCbNh8A=",
"body": {
"appName": "sdk 测试",

```
        "appType": "fabric",
        "caType": 2,
        "algorithmType": 2,
        "mspId": "OrgbNodeMSP",
        "channelId": "app00012020041610201529184 51"
    }
}
```


5.4.3.4 用户注册接口

在密钥托管和上传公钥两种模式下，当参与 Fabric 应用的用户需要为链下系统的子用户单独创建用户交易密钥证书时，需要先调用该接口注册本城市节点下的用户，用户的用户名为调用参数中的 name@appCode

1. 接口地址：

   https://节点网关地址/api/fabric/v1/user/register

2. 通讯方式： POST

3. 签名算法：详见 5.4.3.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	否	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	用户名	name	String	是	长度小于 20
2	用户密码	secret	String	否	密钥托管模式的应用可为空，上传公钥模式的应用为空时将返回随机密码
示例：					
{"header":{"userCode":"USER00012020041519580108712 92","appCode":"app00012020041610 20152918451","tId":""},"mac":"MEUCIQDCa3T1c8Fim3LFVfgvellC/wKWtFnyOl5FK7FXgdd FwIgGHXApypixu9RpkHl13z80ZYdVeyRObX7icU3XWk2+VI=","body":{"name":"user01","se cret":"123456"}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：成功 -1：失败
2	响应信息	msg	String	是	
Body					
1	用户名	name	String	是	长度小于 20
2	用户密码	secret	String	是	上传公钥模式应用请求参数密码为空时返回随机密码

示例


```
{
  "header": {
    "code": 0,
    "msg": "处理成功"
  },
  "mac":
"MEUCIQClfufMU8kRI1gMHIGqfWOh1iv2KIhS+H0dlUUdEuUrLQIgYJz98xp5w/KdVP6bJj
HhV2pZPTe9Cn4xcOrPV4E7ZsA=",
  "body": {
    "name": "user01",
    "secret": "123456"
  }
}
```


5.4.3.5 密钥托管模式下调用链码接口

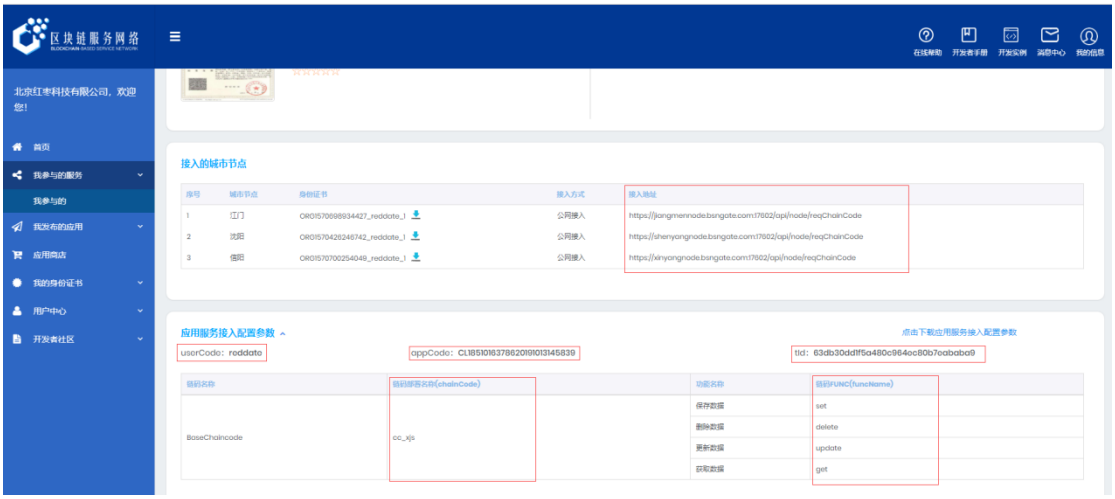
链下系统通过城市节点网关调用链码的交易处理函数时，需要按照接口说明在请求中加入相应的请求参数，调用节点网关后，节点网关会返回链码的执行结果。

1. 接口地址：

https://节点网关地址/api/fabric/v1/node/reqChainCode

该接口不等交易落块结果直接响应结果，可以调用“获取交易信息接口”根据交易 ID 查询落块结果；

注：用户参与服务成功后可以在服务详情页面查看并下载应用链下业务系统访问链上智能合约所需要的应用服务配置参数、节点网关地址和应用接入密钥对，如下图：



2. 通讯方式： POST

3. 签名算法：详见 5.4.3.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
3	用户与应用 ID	tId	String	否	
body					
1	用户名	userName	String	否	
	随机字符串	nonce	String	是	使用 base64 编码的 24 位随机 byte 数据
1	链码 Code	chainCode	String	是	
2	方法名称	funcName	String	是	
3	请求参数	args	String[]	否	
4	临时数据	transientData	Map<string,string>	否	
示例：					
{"header":{"userCode":"USER000120200416100930940741317141233920","tId":""},"mac":"MEQCICJpE1jfeJKtw/ZboVuKSLy2RmmSdkhrEVPGFJhm9IaIAiA/Qqs6RNz0ndSS4/AFSwBj7vC76Py1hXnqO5zMD9pNtA==","body":{"userName":"","nonc					

e":"lgH7Ozfv6npqg9D3pSbq9c6o+rAcpa5D","chainCode":"cc_app000120200416101714123392
0_00","funcName":"set","args":["{\"baseKey\":\"test2020048\",\"baseValue\":\"this is string
\"}"],"transientData":{}}}

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	块信息	blockInfo	**blockInfo**	否	code 不为 0 时为空
2	链码响应结果	ccRes	**ccRes**	否	code 不为 0 时为空
blockInfo					
1	交易 Id	txId	String	是	
2	块哈希	blockHash	String	否	同步接口返回块哈希
3	状态值	status	Int	是	详见交易状态描述
ccRes					
1	链码响应状态	ccCode	Int	是	200：成功 500：失败
2	链码响应结果	ccData	Str	否	具体链码响应的结果

示例

{
"header": {
"code": 0,
"msg": "处理成功"
},
"mac":
"MEUCIQCBtfO1AfYkoJ2hIlp8CfKK1iuhVEAYkPY8YFRAdvPJlAIgDjSqYgwlORJRyF6KZP
U/uC5Fx/DxXxu9VgKwU9+JhjU=",
"body": {
"blockInfo": {
"txId": "a144149150ee615a9d11c68485600f43dc2c3eb2a98d7b36de53a6b99e03c495",
"blockHash": "",
"status": 0
},
"ccRes": {
"ccCode": 200,
"ccData": "SUCCESS"

```
    }
  }
}
```

5.4.3.6 上传公钥模式下用户证书登记接口

当上传公钥模式应用的用户需要注册子用户时，在完成子用注册（6.3.2 用户注册接口）后，调用该接口上传公钥证书申请文件，获取一个由城市节点颁发的子用户证书。在密钥托管模式下调用该接口时将返回异常。

1. 接口地址： https://节点网关地址/api/fabric/v1/user/enroll
2. 通讯方式： POST
3. 签名算法：详见 5.4.3.1 应用接入签名算法
4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	否	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	用户名	name	String	是	注册时的用户名
2	用户密码	secret	String	是	注册时的密码
3	证书申请文件	csrPem	string	是	使用 ECDSA（secp256r1）算法生成的证书申请文件，证书 CN 为 name@appCode
示例： {"header":{"userCode":"USER00012020041519580108712920","appCode":"app00012020041610 20152918451","tId":""},"mac":"MEQCICQaYMzs+edIQkfpt5hoaSO5dWqcrY7Q75FYwyJo/B4r AiAQ10aEpdNATsZYHVcJJ4TxVCgY8XdQBBIyTAOqUmSjkw==","body":{"name":"user01", "secret":"123456","csrPem":"-----BEGIN CERTIFICATE REQUEST----- \nMIHoMIGQAgEAMC4xLDAqBgNVBAMMI3VzZXIwMUBhcHAwMDAxMjAyMDA0MT YxMDIw\nMTUyOTE4NDUxMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEnguk1xunmu U1bnKB\nam8QmeK6Geg/O6kL2D2ig85UMQTpG/sb9iYkduz8iC9SRnF9TvLiHuvJX2FGAOA Q\nK1Vz8aAAMAoGCCqGSM49BAMCA0cAMEQCIE19Iin91KlfEvfFIbxhF14enFHhtvOU\n5					

rK86huFiMMQAiBYXO4fJBq6eLGjaavR71O9fOvVZ5W7X+GQjIlQDuDgPQ==\n-----END CERTIFICATE REQUEST-----\n"}}

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：成功 -1：失败
2	响应信息	msg	String	是	
Body					
1	证书内容	cert	String	是	

示例

{
  "header": {
    "code": 0,
    "msg": "处理成功"
  },
  "mac":
"MEUCIQCE0gg5VHWsZluNKAV2+xOJANGnCkw6f9J4+mFT1TWz/gIgfu93jqzTzk0DU2lfMKnExcwVbgelWMLvLmwKplCXNBA=",
  "body": {
    "cert": "-----BEGIN CERTIFICATE-----
\nMIICvTCCAmSgAwIBAgIUcqn2HmCYmq/V2yKbnxuvc49KU00wCgYIKoZIzj0EAwIw\nTjELMAkGA1UEBhMCQ04xEDAOBgNVBAgTB0JlaWppbmcxDDAKBgNVBAoTA0JTTjEP\nMA0GA1UECxMGY2xpZW50MQ4wDAYDVQQDEwVic25jYTAgFw0yMDA0MjEwNTAzMDBa\nGA8yMTAwMDMyMTExMDQwMFowbDE8MA0GA1UECxMGY2xpZW50MA8GA1UECxMIb3Jn\nYm5vZGUwDgYDVQQLEwdic25iYXNlMAoGA1UECxMDY29tMSwwKgYDVQQDDCN1c2Vy\nMDFAYXBwMDAwMTIwMjAwNDE2MTAyMDE1MjkxODQ1MTBZMBMGByqGSM49AgEGCCqG\nSM49AwEHA0IABJ4LpNcbp5rlNW5ygWpvEJniuhnoPzupC9g9ooPOVDEE6Rv7G/Ym\nJHbs/IgvUkZxfU7y4h7ryV9hRgDgECtVc/Gjgf8wgfwwDgYDVR0PAQH/BAQDAgeA\nMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFG28toKRbzJTFa6v/xlIYr6S9EvaMB8G\nA1UdIwQYMBaAFAcI4H+kIs8vn94ZYYpkrd+5ldMKMIGbBggqAwQFBgcIAQSBjnsi\nYXR0cnMiOnsiaGYuQWZmaWxpYXRpb24iOiJvcmdibm9kZS5ic25iYXNlLmNvbSIs\nImhmLkVucm9sbG1lbnRJRCI6InVzZXIwMUBhcHAwMDAxMjAyMDA0MTYxMDIwMTUy\nOTE4NDUxIiwiaGYuVHlwZSI6ImNsaWVudCIsInJvbGUiOiJjbGllbnQifX0wCgYI\nKoZIzj0EAwIDRwAwRAIgLtlTps/DOHK8S3La7bnlChB+88b1Fko9bOAL36oAFPIC\nIHQPCC30MoTHIId/X3fC5IxNukssmlMnEuDX73zRL55/\n-----END CERTIFICATE-----\n"
  }
}

5.4.3.7 上传公钥模式下调用链码接口

当上传公钥模式应用的用户需要从链下系统向链上链码发起交易时，需要在本地组装交易报文，并调用该接口发起交易。

1. 接口地址： https://节点网关地址/api/fabric/v1/node/trans
2. 通讯方式： POST
3. 签名算法：详见 5.4.3.1 应用接入签名算法
4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	否	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	交易数据	transData	String	是	使用 base64 转码之后的交易数据

示例：

{"header":{"userCode":"USER000120200415195801087129 2","appCode":"app00012020041610 20152918451","tId":""},"mac":"MEUCIQCv8EZ2OqbSbI9xGGKX06Mquh+g+NhhbUoAJBbne mXdagIgNMF7W7ecu5uej9BpVx04qwJuVijbgcp3VYIcjDK0Z38=","body":{"transData":"Cq0K CrsJCpcBCAMaCwi9gPr0BRD0o+Z2IhxhcHAwMDAxMjAyMDA0MTYxMDIwMTUyOTE4N DUxKkBjM2M2NTIzOTU4YzM4MTExOTJiOGQzNThkZDI2MTdmMWIxNGNiNjYxZGU2Y jAyMmMxYTgyMjI2OWU4YThjNDhkOiYSJBIiY2NfYXBwMDAwMTIwMjAwNDE2MTAy MDE1MjkxODQ1MV8wMBKeCAqBCAoLT3JnYk5vZGVNU1AS8QctLS0tLUJFR0lOIENFUl RJRklDQVRFLS0tLS0KTUlJQ3ZUQ0NBbVNnQXdJQkFnSVVWanBGZTJFaERFaHJlOHBB VTh4bkd3dXhPbU13Q2dZSUtvWkl6ajBFQXdJdwpUakVMTUFrR0ExVUVCaE1DUTA0eEVE QU9CZ05WQkFnVEIwSmxhV3BwYm1jeEREQUtCZ05WQkFvVEEwSlRUakVQCk1BMEdB MVVFQ3hNR1kyeHBaVzUwTVE0d0RBWURWUVFERXdWaWMyNWpZVEFnRncweU1EQ TBNVGt3TkRNek1EQmEKR0E4eU1UQXdNRE15TVRFeE1EUXdNRm93YkRFOE1BMEdB MVVFQ3hNR1kyeHBaVzUwTUE4R0ExVUVDeE1JYjNKbgpZbTV2WkdVd0RnWURWUVF MRXdkaWMyNWlZWE5sTUFvR0ExVUVDeE1EWTI5dE1Td3dLZ1lEVlFRRERDTjBaWE4w Ck1ESkFZWEJ3TURBd01USXdNakF3TkRFMk1UQXlNREUxTWpreE9EUTFNVEJaTUJNR0 J5cUdTTTQ5QWdFR0NDcUcKU000OUF3RUhBMElBQk5YZmFMVW1wMXlJSFVMMXVK eEdwMDFQNHE5Zk81V2xFMFZtallYQmVMejBhYlhqSU96NwpYb29KcGRUS1ZkUUJaZzY rZkVPWmhudm1vbURXWjRpdTRhYWpnZjh3Z2Z3d0RnWURWUjBQQVFIL0JBUURBZ2V BCk1Bd0dBMVVkRXdFQi93UUNNQUF3SFFZRFZSME9CQllFRkZZRDg5emtkVllRbzZpUE

h3d2RJejNaQ1lSck1COEcKQTFVZEl3UVlNQmFBRkFjSTRIK2tJczh2bjk0WllZcGtyZCs1bGR
NS01JR2JCZ2dxQXdRRkJnY0lBUVNCam5zaQpZWFIwY25NaU9uc2lhR1l1UVdabWFXeHB
ZWFJwYjI0aU9pSnZjbWRpYm05a1pTNWljMjVpWVhObExtTnZiU0lzCkltaG1Ma1Z1Y205c2
JHMWxiblJKUkNJNkluUmxjM1F3TWtCaGNIQXdNREF4TWpBeU1EQTBNVFl4TURJd01U
VXkKT1RFNE5EVXhJaXdpYUdZdVZIbHdaU0k2SW1Oc2FXVnVkQ0lzSW5KdmJHVWlPaU
pqYkdsbGJuUWlmWDB3Q2dZSQpLb1pJemowRUF3SURSd0F3UkFJZ1ZZNi9jZ1NDTmpEN
kxwTXVaZEQzVWYvWko5c3FSUVVTR3hSQU9SeGZONThDCklFN0JHTDljOHRCcHJiVm
pYTldtQmpObWhqeUE3N0l3SW8rbUg1ZXp4R1B1Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0t
LS0KEhiQKmgB1IbwbgLAyoHXUNnjZSGOqBDheQMSbQprCmkIARIkEiJjY19hcHAwMDA
xMjAyMDA0MTYxMDIwMTUyOTE4NDUxXzAwGj8KA3NldAo4eyJiYXNlS2V5IjoidGVzd
DIwMjAwNDA0IiwiYmFzZVZhbHVlIjoidGhpcyBpcyBzdHJpbmcgIn0SRjBEAiB+mOUKY7f
RjcZ1/qc96YP9GGod3UK56jJaWaE4o3J90QIgeirrjyzL6zQLN89tv3jDpI7vxKChkGM9u8IEFiF
EGYo="}}

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	块信息	blockInfo	**blockInfo**	否	code 不为 0 时为空
2	链码响应结果	ccRes	**ccRes**	否	code 不为 0 时为空
blockInfo					
1	交易 Id	txId	String	是	
2	块哈希	blockHash	String	否	同步接口返回块哈希
3	状态值	status	Int	是	详见交易状态描述
ccRes					
1	链码响应状态	ccCode	Int	是	200：成功 500：失败
2	链码响应结果	ccData	Str	否	具体链码响应的结果

示例

{
  "header": {
    "code": 0,
    "msg": "处理成功"
  },
  "mac":
"MEQCICXNk40O+Gkqqe2XgoaxdOoIvDQe4RfLtwXkxjC7ce8TAiBLVu6PjOqWueVB3t4h7
REpNdcVf6L0qVzfdA1yovuc7g==",

```
  "body": {
    "blockInfo": {
      "txId": "c3c6523958c3811192b8d358dd2617f1b14cb661de6b022c1a822269e8a8c48d",
      "blockHash": "",
      "status": 0
    },
    "ccRes": {
      "ccCode": 200,
      "ccData": "SUCCESS"
    }
  }
}
```


5.4.3.8 获取交易信息接口

链下系统使用该接口可以根据交易 ID 获取交易信息。

1. 接口地址：

   https://节点网关地址/api/fabric/v1/node/getTransaction

2. 通讯方式： POST

3. 签名算法：详见 5.4.3.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	交易 Id	txId	String	是	
示例：					
{"header":{"userCode":"USER00012020041519580108712920","appCode":"app00012020041610 20152918451","tId":""},"mac":"MEUCIQDIbcNl+C1iBbXWGW3qjhf80IRgCgvJuyxx0WXU2v n2TAIgZgA020L2aXBtrdLsYEkYPyiOJ9+AFrXOEwfuzy8B4bE=","body":{"txId":"c3c652395 8c3811192b8d358dd2617f1b14cb661de6b022c1a822269e8a8c48d"}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	块 Hash	blockHash	String	是	
2	块号	blockNumber	Long	是	
3	交易状态	status	Int	是	详见交易状态描述
4	上链用户名	createName	String	是	
5	时间戳秒	timeSpanSec	Int64	是	时间戳的秒部分
6	时间戳纳秒	timeSpanNsec	Int64	是	时间戳的纳秒部分

示例


```
{
  "header": {
    "code": 0,
    "msg": "处理成功"
  },
  "mac":
"MEUCIQDUFw5pa4QJcEiQjYeLTl2L94HbsZbz7DArF+djgzWoTQIgU8u+dG6CcHwBZjuf9P
vhYdEFAa/ujwo8UAPbAmKxRq0=",
  "body": {
    "blockHash": "ab9366cf63881228863c884527fceefabc9ad2e375aa0bcbf71f17f75c7d3ff5",
    "blockNumber": 7,
    "status": 0,
    "createName": "test02@app000120200416102015291845 1",
    "timeSpanSec": 1587445821,
    "timeSpanNsec": 249139700
  }
}
```


5.4.3.9 获取块信息接口

数据上链后，链下业务系统调用城市节点网关该接口会获得当前交易所在块信息（body.blockInfo），状态值（body.blockInfo.status）和交易 Id（body.blockInfo.txId），如果状态值为 0 时，表示交易提交成功并且落块，可根据交易 Id 查询块信息。

1. 接口地址：

   https://节点网关地址/api/fabric/v1/node/getBlockInfo

2. 通讯方式： POST

3. 签名算法：详见 5.4.3.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	块号	blockNumber	Int64	否	不可同时为空
2	块哈希	blockHash	String	否	不可同时为空
3	交易 Id	txId	String	否	不可同时为空
示例：					
{"header":{"userCode":"USER000120200415195801087129 2","appCode":"app00012020041610 20152918451","tId":""},"mac":"MEUCIQCrGthrAvNalUsWEdnDxZkNXF4nCpXOxIFQdp1YY hGvugIgKvYql9Ex6RCcOhqt6coufNPH/QhtKYNeThWJ2rEL+4g=","body":{"blockNumber":6, "blockHash":"","txId":""}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	块 Hash	blockHash	String	是	
2	块号	blockNumber	Long	是	
3	上一个块 Hash	preBlockHash	String	是	
4	块大小	blockSize	Long	是	字节
5	当前块的交易数量	blockTxCount	Int	是	

6	交易详情	transactions	[]TransactionData	是	交易详情
TransactionData					
1	交易 Id	txId	String	是	
2	交易状态	Status	Int	是	详见交易状态描述
3	交易提交者	createName	String	是	
4	交易时间戳秒	timeSpanSec	Int64	是	
5	交易时间戳纳秒	timeSpanNsec	Int64	是	

示例


```
{
  "header": {
    "code": 0,
    "msg": "处理成功"
  },
  "mac":
"MEUCIQC8nfFnHw4sEYJmaSTT1xepxMGgomxyJtt0ysyGgPB0AwIgfuuiegdGEbBi/2wmFCco39wmik3isLWtvnN9ZmJDTdk=",
  "body": {
    "blockHash": "fc83c306677925efee540b4d7b7ca73e06f144cae34c706f1101d6b395ada2da",
    "blockNumber": 6,
    "preBlockHash":
"93c86551d812229274e144093cd4bd17dacb35bc6a01779930e11f43f886bf34",
    "blockSize": 7020,
    "blockTxCount": 1,
    "transactions": [
      {
        "txId": "a8639f3a796267e048d475b00fe7646a4524f1c20d71880e19708821177b7bdb",
        "status": 0,
        "createName": "test02@app000120200416102015291845l",
        "timeSpanSec": 1587271285,
        "timeSpanNsec": 26436800
      }
    ]
  }
}
```


5.4.3.10 获取最新账本信息接口

获取当前链上应用最新的账本信息，包含块 Hash、上一个块 Hash、当前块的高度等信息。

1. 接口地址：

https://节点网关地址/api/fabric/v1/node/getLedgerInfo

2. 通讯方式： POST

3. 签名算法：详见 5.4.3.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
示例：					
{"header":{"userCode":"USER000120200415195801087129 2","appCode":"app00012020041610 20152918451","tId":""},"mac":"MEQCID7Z3J2PiRDOx7JasRamBZRTAHXj1XAG1K/DUkzJE wuiAiBIY5p3H2kArE7OuYLOgEqMHl15Xgj5Voi5zVPGhyU/+w==","body":{}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	块 Hash	blockHash	String	是	
2	块高	height	Long	是	
3	上一个块 Hash	preBlockHash	String	是	
示例					

{
"header": {
"code": 0,
"msg": "处理成功"
},
"mac":
"MEUCIQC4PhYTBNyt1rSeBeZTdOly42CxILVgK1b/RlieA33G1gIgeodoEa5Ou0X4uWc/VGp 0n6NKByhXlBbo22FME4xQ8aw=",
"body": {
"blockHash": "ab9366cf63881228863c884527fceefabc9ad2e375aa0bcbf71f17f75c7d3ff5",
"height": 8,

```
      "preBlockHash":
"fc83c306677925efee540b4d7b7ca73e06f144cae34c706f1101d6b395ada2da"
   }
}
```


5.4.3.11 链码事件注册接口

参与的应用服务中，如果需要根据链码事件触发链下业务系统进行后续业务处理时，可调用该接口注册需要监听的链码事件。

1. 接口地址：

   https://节点网关地址/api/fabric/v1/chainCode/event/register

2. 通讯方式： POST

3. 签名算法：详见 5.4.3.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
Header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
Body					
1	链码 Code	chainCode	String	是	
2	链码事件 key	eventKey	String	是	
3	链码事件通知地址	notifyUrl	String	是	接收监听到的链码事件的地址
4	附加参数	attachArgs	String	否	
示例：					
{"header":{"appCode":"CL20191107112252","userCode":"lessing"},"body":{"attachArgs":"name=张三 &age=20","chainCode":"cc_bsn_test_00","eventKey":"test01","notifyUrl":"http://192.168.6.128:8080/api/event/notifyUrl"},"mac":"MEUCIQCjzPr4KZVild2Vm5YgcunOXTh9mQK2QfWcRnYCk+jOzgIgDW6oHca7/249M43p2ElwiMNbuejdwAnyW5OwiMqiWCQ="}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
Header					
1	响应标识	code	int	是	0：注册成功 -1：注册失败
2	响应信息	msg	String	是	
Body					
1	事件编码	eventId	String	是	
示例					
{ "header": { "code": 0, "msg": "注册链码事件成功" }, "body": { "eventId": "bd3391deedbe44a7ad5b7f80ce59abfa" }, "mac": "MEQCIENLpj2R9mRL100vcMXs0X5rwfSjB/U7kMg+76GjEPNJAiBlUo/Eyj49uXTPrzRW0m4rJ0NQIkZnDMPbyalxojXwrA=="}					

5.4.3.12 块事件注册接口

参与的应用服务中，如果需要根据块事件触发链下业务系统进行后续业务处理时，可调用该接口注册需要监听的块事件。

1. 接口地址：

https://节点网关地址

/api/fabric/v1/chainCode/event/blockRegister

2. 通讯方式： POST

3. 签名算法：详见 5.4.3.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
Header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	

Body					
1	事件通知地址	notifyUrl	String	是	接收监听到的块事件的地址
2	附加参数	attachArgs	String	否	
示例：					
{"header":{"userCode":"USER000120200710164124351616 3","appCode":"app00012021011914 11238426266","tId":""},"mac":"MEUCIQClsjKy/ee1qaYrItzCO1bMfjs0g0kPu8+YOCjbk3rPRA IgSfeyYvfeoh8QciZPG4fZQepaiyh7PmmWjYzFSqylT/c=","body":{"chainCode":"","eventKey": "","notifyUrl":"http://192.168.6.78:58011/v1/fabric/test","attachArgs":"a=1"}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
Header					
1	响应标识	code	int	是	0：注册成功 -1：注册失败
2	响应信息	msg	String	是	
Body					
1	事件编码	eventId	String	是	
示例					


```
{
   "header": {
      "code": 0,
      "msg": "success"
   },
   "mac":
"MEUCIQC6PKsSqfkQGLrqi2vMpZzBP5beLhyP+fXVr8S5aqhaagIgaEtAnsuiubibYoYZzQ/8a
GYErzm5rtU8Oj952OuHgCo=",
   "body": {
      "eventId": "002f0e1f0b0f4331ab541461547a38d6"
   }
}
```


5.4.3.13 链码以及出块事件查询接口

用于查询已注册的链码以及出块事件监听列表。

1. 接口地址：

https://节点网关地址/api/fabric/v1/chainCode/event/query

2. 通讯方式： POST

3. 签名算法：详见 5.4.3.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	否	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
示例：					
{"header":{"appCode":"CL20191107112252","userCode":"lessing"},"body":{},"mac":"MEQCIAnJxvuKVe0u/bG0VYCjM3g3ctxTYIWkejYp462okNlcAiBcOTGvAkF7xErL2w1PiwgfFjIu3Sszgyfzym/pEwRGxA=="}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	[]body	是	事件列表
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：查询成功 -1：查询失败
2	响应信息	msg	String	是	
body					
1	事件编码	eventId	String	是	
2	链码事件 key	eventKey	String	否	块事件时为空
3	链码事件通知地址	notifyUrl	String	是	
4	附加参数	attachArgs	String	否	
5	创建时间	createTime	String	是	
6	城市编码	orgCode	String	是	
7	用户唯一标识	userCode	String	是	
8	应用唯一标识	appCode	String	是	
9	链码 Code	chainCode	String	否	块事件时为空
10	事件类型	eventType	String	否	块事件返回”block”,链码事件为空

示例
{ "header": { "code": 0, "msg": "查询链码事件成功" }, "body": [{ "eventKey": "test001", "notifyUrl": "http://192.168.6.128:8080/api/event/notifyUrl", "attachArgs": "a=123\u0026b=456", "eventId": "945ee631d26140118963ad3104c81713", "createTime": "2019-11-18 14:22:59", "orgCode": "ORG1571365934172", "userCode": "lessing", "appCode": "CL20191107112252", "chainCode": "cc_bsn_test_00" }, { "eventKey": "test002", "notifyUrl": "http://192.168.6.128:8080/api/event/notifyUrl", "attachArgs": "hahahhahhahahahah", "eventId": "346617a493d84c6d8512b8dddad87811", "createTime": "2019-11-18 14:29:28", "orgCode": "ORG1571365934172", "userCode": "lessing", "appCode": "CL20191107112252", "chainCode": "cc_bsn_test_00" }, { "eventKey": "test01", "notifyUrl": "http://192.168.6.128:8080/api/event/notifyUrl", "attachArgs": "name=张三 \u0026age=20", "eventId": "bd3391deedbe44a7ad5b7f80ce59abfa", "createTime": "2019-11-19 10:52:15", "orgCode": "ORG1571365934172", "userCode": "lessing", "appCode": "CL20191107112252", "chainCode": "cc_bsn_test_00" }], "mac": "MEQCIEYXFMa8dfBrjy/s9H5JAoFIrjROJBiw+7/daELUbF5eAiA7a6HvqqbOpv6vlkunHGxCB1o5DoeuJFD0FM6kLoU34Q=="}

5.4.3.14 链码以及出块事件注销接口

注销已注册的链码或者块事件监听。

1. 接口地址：

   https://节点网关地址/api/fabric/v1/chainCode/event/remove

2. 通讯方式： POST

3. 签名算法：详见 5.4.3.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	否	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	事件编码	eventId	String	是	

示例： {"header":{"appCode":"CL20191107112252","userCode":"lessing"},"body":{"eventId":"bd3391deedbe44a7ad5b7f80ce59abfa"},"mac":"MEQCIE3/CLG5LxZZN7En7LZvzthajwxHzpvDduXSsw4Tb1JFAiAXGJ4WVtyCKbtCasQGofCkge8NOgZDNPgJIdTCtCi2SQ=="}

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：注销成功 -1：注销失败
2	响应信息	msg	String	是	
示例					
{ "header": { "code": 0, "msg": "注销链码事件成功" }, "body": null, "mac": "MEUCIQCaTFLliY7pPjkwcmSsLXOth7k9bQj9Sblq+1nMVjkFAAIgUsizFO+f1+dxU3/hPxjf/+na4qG6aQFftJIWGtMhlVI="}					

5.4.3.15 链码以及块事件通知报文信息接口

该接口是由链下业务系统侧实现。城市节点网关在监听到注册的链码或者块事件触发后调用该接口将事件执行结果通知给链下业务系统。

链下业务系统在收到城市节点网关通知后应返回一个包含 success 的字符串，否则城市节点网关将尝试分别在第 3、12、27、48 秒后再次发起通知，共计 5 次。

1. 通讯方式： POST

2. 签名算法：详见 5.4.3.1 应用接入签名算法

3. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	否	
3	签名值	mac	String	是	

header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	链码 Code	chainCode	String	否	块事件通知时为空
2	城市编码	orgCode	String	是	
3	注册的链码事件 key	eventKey	String	否	
4	注册的链码事件 ID	eventId	String	是	
5	注册的链码事件参数	attachArgs	String	否	注册时填写的附加参数
6	监听到的链码事件 key	eventName	String	否	链码中设置的事件名称，块事件通知时为空
7	当前链码的交易 ID	txId	String	否	块事件通知时为空
8	监听到的链码事件值	payload	String	否	
9	当前交易块高	blockNumber	Long	是	
10	响应随机码	nonceStr	String	是	业务平台可根据该值判断是否接收过通知，同一个业务的多次通知中，该字符串不变
11	上一个块哈希	previousHash	String	否	链码事件通知时为空
示例： 链码事件通知 {"header":{"userCode":"lessing","appCode":"CL20191107112252"},"body":{"chainCode":"cc_bsn_test_00","orgCode":"ORG1571365934172","eventKey":"test:\\S{32}","eventId":"2964a0f60b3e460f834618b3664af2da","attachArgs":"abc=123211","eventName":"test:1234567812345678123456781234567812345678","txId":"32fc105681820fa556b8a460efc1e43a47daa864b959ea1753abb4640f2dce49","payload":"","blockNumber":74,"nonceStr":"522c8061b5e84837bad72ca08c6a353f"},"mac":"MEQCIDU4tROyjLtvD1b8TTbWWAlCPuUbmdPAEUXwRRgVn7kIAiA58je5u/7xDuRPcgeUWL3nB9mouUGQ6dGKJMmD7Jm08g=="} 块事件通知： {"header":{"userCode":"USER000120200710164124351616 3","appCode":"app000120210119141123842626 6"},"body":{"orgCode":"ORG20200411141716923 60","eventId":"8746bb9a1e854c9f8b3710f5a63f7c59","attachArgs":"a=1","previousHash":"022281f6089e3684501251775166b6b0afd18a176ec98a835cb5d09aff0d4950","blockNumber":12,"nonceStr":"79a7baa26c854caeb2e2e7abc0b7f07e"},"mac":"MEUCIQDiZrwf8fKG/3fuaVrsfTN3BKmLx+qnnEuuSaHfvIBbMQIgS+1qHKXeVR24WXwOGu3Nze/tLLziQ0LkjXaueYu0ctM="}					

5.4.3.16 调用链码接口的交易状态描述

在密钥托管和上传公钥两种模式下，链下业务系统请求城市节点网关的调用链码接口时，节点网关返回的交易处理状态说明具体如下：

序号	状态码	备注
1.	0	成功

		等待响应结果时，落块的最终状态 不等待响应结果：链码的处理状态
2.	-1	等待落块结果超时
3.	1	空的提交数据
4.	2	异常的响应
5.	3	错误的提交信息
6.	4	错误的创建者签名
7.	5	无效的“背书人”交易
8.	6	无效的事务配置
9.	7	不支持的交易响应
10.	8	错误的交易 ID
11.	9	重复的交易 ID
12.	10	背书失败
13.	13	未知的交易类型
14.	14	找不到目标链码
15.	17	链码过期
16.	18	链码版本冲突
17.	254	无效的交易
18.	255	其他原因的无效交易

5.4.4 城市节点网关 FISCO BCOS API

城市节点网关部署在各个城市节点，接收应用系统的请求，使用托管的用户身份信息，向基于 FISCO BCOS 的应用的智能合约发起访问并返回智能合约的执行结果。节点网关的调用是通过向区块链服务的各个城市节点的网关服务发送 HTTP 请求来实现。节点网关负责验证用户身份信息和应用信息，通过用户身份信息和应用信息以及需要访问的智能合约、智能合约方法来传递智能合约参数、获取智能合约执行结果的服务通道。

5.4.4.1 应用接入签名算法

链下业务系统向城市节点网关发送交易请求时，需使用应用接入密钥对中的私钥对交易请求报文进行数字签名，节点网关接收到交易请求报文后，使用已上传或托管的密钥对中的公钥对该报文进行验签。只有在交易报文验签通过时，网关才会对交易请求报文进行后续的交易处理。具体签名算法如下：

1. 组装签名字符串

将请求参数按照文档参数表格中的顺序转换为字符串进行拼接，其中请求参数优先拼接 Header 中的 UserCode、AppCode；返回参数优先拼接 code、msg，然后再次按照文档参数顺序拼接 Body 中的参数。

2. 不同类型的转换格式

类型	规则	示例	结果
String	不转换	abc	abc
Int/int64/long	十进制转换	-12	-12
Float	十进制转换，小数位参考备注	1.23	1.23
Bool	转换为“true”或者“false”	真	true
Array	按照参数顺序和类型拼接	[“abc”,”xyz”]	abcxyz
Map[key]value	按照顺序拼接 key 和 value	[“a”:1,”b”:2]	a1b2
Object	将对象内的属性按照文档循序依据上述格式转换	{“name”:”abc”,”secret”:”123456”}	abc123456

a) 针对使用 ECDSA（secp256k1）密钥算法的 FISCO BCOS 框架应用

- 哈希值计算：将按照上述规则转换后的待签名的字符串按照 UTF-8 编码做 SHA256 计算；
- 对哈希值获取签名：哈希值与私钥进行 ECDSA（secp256k1）加密签名计算；在部分语言的处理中（例如 C#、Java）如果使用 SHA256WithECDSA 进行签名则不需要上一步，该算法中已经取哈希计算。
- 将签名结果做 Base64 计算。

b) 针对使用国密算法的 FISCO BCOS 框架应用，使用以下规则

- 哈希值计算：将按照上述规则转换后的待签名的字符串按照

UTF-8 编码做 SM3 计算；

- 对哈希值获取签名：哈希值与私钥进行 SM2 加密签名计算；
- 将签名结果做 Base64 计算。

3. 示例

参数:

{"header":{"userCode":"user01","appCode":"app01"},"mac":"","body":{“userId”:”abc”,”list”:[“abc”,”xyz”]}}

结果：user01app01abcabcxyz

5.4.4.2 密钥证书模式

1. 密钥托管模式

如第五章描述，用户在参与应用时需要两个密钥对：应用接入密钥对和用户交易密钥对。在该模式下密钥对不需要用户在本地生成，而是由 BSN 生成并且托管，用户只需要从 BSN 门户下载使用即可。

- **应用接入密钥对：**用户在参与应用服务成功后，BSN 会生成一对与所参与应用的底层框架算法一致的公私密钥对托管在 BSN 中，用户可以在 BSN 门户内我的证书列表下载该公私钥对。在其链下业务系统调用城市节点网关接口时需要使用该密钥对的私钥进行交易签名。节点网关使用托管的公钥对签名进行验签。
- **用户交易密钥对：**是用户调用智能合约处理交易的身份。在密钥托管模式下，用户参与应用服务成功后，其链下业务系统可以调用城市节点网关的注册用户接口生成一个链上用户账户并将用户的交易密钥对托管在城市节点中，在交易时使用用户的 UserId 即可进行交易处理。

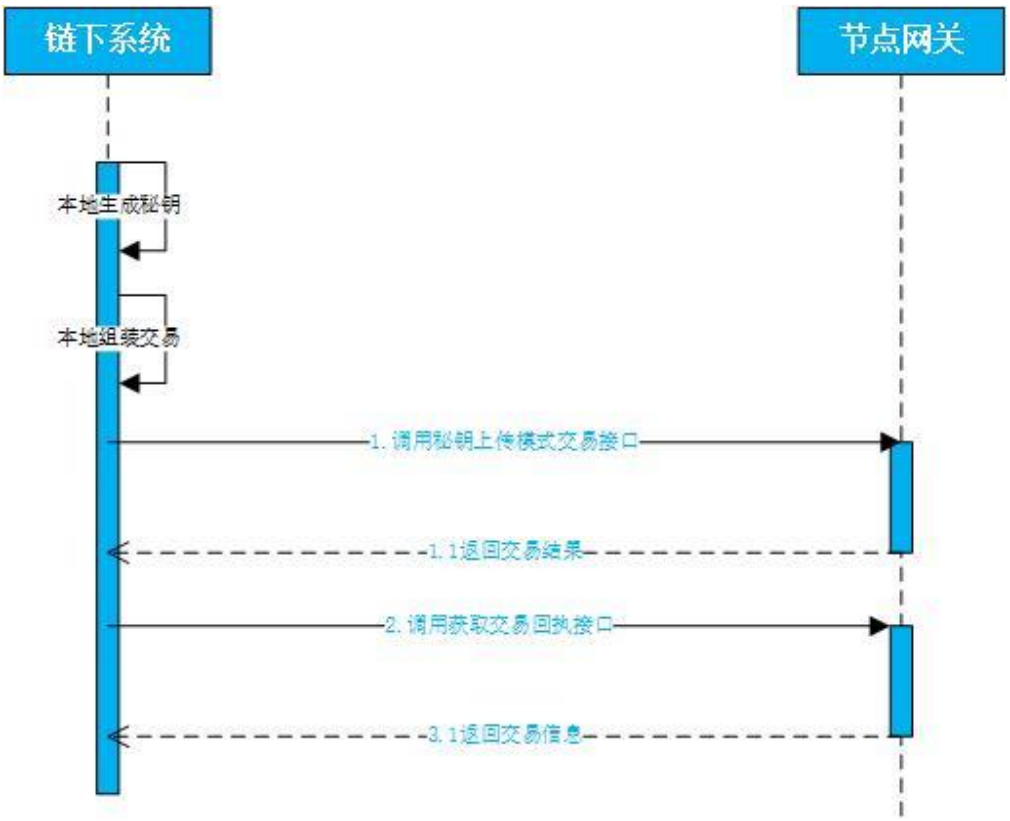
交易流程:

2. 上传公钥模式

用户在参与应用时需要两个密钥对：应用接入密钥对和用户交易密钥对。在该模式下用户需要自己在本地生成公私钥对，私钥本地保存。

- **应用接入密钥对**：用户在 BSN 门户参与应用服务时，需要在本地生成一对与该应用服务的底层框架算法一致的公私钥对，并且将公钥提交到 BSN 中。链下业务系统在调用城市节点网关接口时使用该密钥对的私钥进行交易签名。节点网关收到交易请求后使用用户上传的公钥对签名进行验签，以验证交易身份的合法性。
- **用户交易密钥对**：是用户调用链码处理交易的身份。在上传公钥模式下，用户需要在自己的链下业务系统中生成用户交易公私钥对，在参与交易时，使用自己生成的公私钥对交易数据进行本地签名。然后将签名后的数据上传即可。

交易流程：



5.4.4.3 获取应用信息接口

调用该接口可以获取应用的基本信息，可以在上传公钥模式中使用。

1. 接口地址： https://节点网关地址/api/app/getAppInfo
2. 通讯方式： POST
3. 签名算法：不需要
4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	否	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					

示例：					
{"header":{"userCode":"USER000120200415195801087129 2","appCode":"app00012020041610 20152918451","tId":""},"mac":"","body":{}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：成功 -1：失败
2	响应信息	msg	String	是	
Body					
1	应用名	appName	String	是	
2	应用类型	appType	String	是	
3	应用密钥托管类型	caType	Int	是	1：托管 2：非托管
4	应用密钥类型	algorithmType	Int	是	1：SM2 2:ECDSA(secp256r1) 3:ECDSA(secp256k1)
5	该城市 MSPID	mspId	String	是	
6	应用链名称	channelId	String	是	Fabric 对应 channelId， fisco 对应 groupId

示例


```
{
  "header": {
    "code": 0,
    "msg": "处理成功"
  },
  "mac":
"MEUCIQDE9zv0E/w4V/ILG6wUCFP08a7NDCAtX/IoZOcCyY4gIQIgUTYWsFTA1KE88gE6
452jKnnVBrhznGVOV2HPMCbNh8A=",
  "body": {
    "appName": "sdk 测试",
    "appType": "fabric",
    "caType": 2,
    "algorithmType": 2,
    "mspId": "OrgbNodeMSP",
    "channelId": "app00012020041610201529184 51"
  }
}
```

5.4.4.4 注册用户接口

用户参与一个 FISCO BCOS 的应用服务成功后，需要通过链下业务系统调用本接口生成进行链上交易处理的用户账户和账户地址。

1. 接口地址：

   https://节点网关地址/api/fiscobcos/v1/user/register

2. 通讯方式： POST

3. 签名算法：详见 5.4.4.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	用户名	userId	String	是	注册的用户名
示例： { "header":{"appCode":"CL188103887322019090211431 4","userCode":"newuser"}, "body": { “userId”:”abc” }, "mac":"MEQCIBRhaM2szckWl9N9qcqnaYXOXGQw7SfII9DlRvxcI3YVAiBt4XeNs+EUjhBNSr3IjLRPZucsuGHxfjt9RiaNIQS8cA=="}					
签名值：					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功

					-1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	用户信息	data	[]string	否	code 不为 0 时为空
data					
1	用户编号	userId	String	是	
2	用户地址	userAddress	String	是	
示例					


```
{
   "header": {
      "code": 0,
      "msg": "处理成功"
   },
   "mac":
"MEQCIEI5VKMyJUXls2Hf8TLoPXjZLT4/L2wyXoddgTnZdqRsAiBxEBMeCOZ8M97OCRU
AMZNMcL974vhzjOS/tk8/wbgbsA==",
   "body": {
      "userId": "100003",
      "userAddress": "0x14647a48303b5e1c77934583883ebc327ba3b297"
   }
}
```

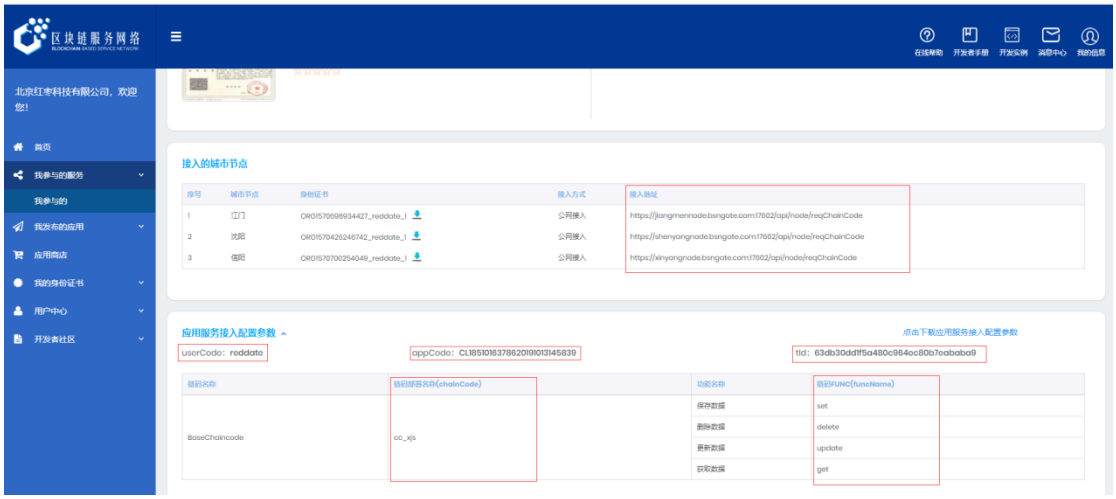

5.4.4.5 密钥托管模式下调用智能合约接口

链下业务系统在连接节点网关时，需要按照接口说明在请求中加入相应的请求参数，调用节点网关以后，节点网关会返回智能合约的执行结果。

1. 接口地址：

https://节点网关地址/api/fiscobcos/v1/node/reqChainCode

注：用户参与服务成功后可以在服务详情页面查看并下载应用链下业务系统开发所需要的应用服务配置参数、节点网关地址和应用接入密钥对，如下图：



2. 通讯方式： POST

3. 签名算法：详见 5.4.4.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
Body					
1	用户 ID	userId	String	是	通过用户注册接口注册的用户的 id
2	合约名称	contractName	String	是	
3	方法名称	funcName	String	是	
4	方法参数	funcParam	string	否	将 array 格式的参数转为 json 字符串赋值
示例：					
{"header":{"appCode":"cl00062020031819265736775 72","userCode":"USER000620200318195 1281835816"},"body":{"contractName":"HelloWorld","userId":"100003","funcName":"set","fun cParam":"[\"abc\"]"},"mac":"MEUCIQDTFe2Gerdf7YJrG1a1Yt99M0ZQ3T1lGpsXdNmFV7Wu TgIgSkZ19abUhAJbMrJMBoD8N7f26xhpQRuR4vNAfY7EEbs="}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 是可为 null
body					
1	调用类型	constant	Bool	否	
2	查询信息	queryInfo	String	否	Constant 为 true 时，该字段有值
3	交易 hash	txId	string	否	Constant 为 false 时，该字段有值并且有效
4	块哈希	blockHash	String	否	Constant 为 false 时，该字段有值并且有效
5	块号	blockNumber	Int	否	Constant 为 false 时，该字段有值并且有效
6	Gas 使用值	gasUsed	Int	否	Constant 为 false 时，该字段有值并且有效
7	交易状态	status	String	否	Constant 为 false 时，该字段有值并且有效,0x0 表示交易成功,状态值参考交易回执状态
8	发送者的地址	from	String	否	Constant 为 false 时，该字段有值并且有效
9	接收者的地址	To	String	否	Constant 为 false 时，该字段有值并且有效
10	输入	Input	String	否	Constant 为 false 时，该字段有值并且有效
11	输出	output	String	否	Constant 为 false 时，该字段有值并且有效
示例					

5.4.4.6 密钥上传模式下调用智能合约接口

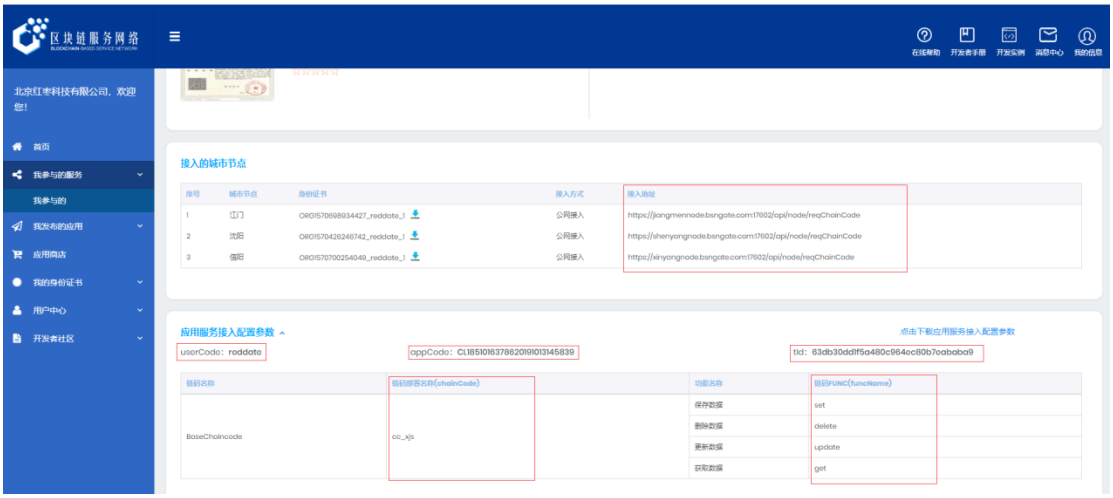
链下业务系统在连接节点网关时，需要按照接口说明在请求中加入相应的请求参数，调用节点网关以后，节点网关会返回智能合约的

执行结果。在上传公钥模式的交易中，链上交易的私钥由用户自己生成和保存，然后由客户端在本地进行上链数据的组装和签名，将签名后的数据上传至节点网关，网关将数据转发至相应的区块链节点发起交易请求。在该模式中组装数据时需要用到合约的 ABI 以及合约地址等信息，其中合约 ABI 在开发合约时编译合约得到，合约地址可以在应用的详情页面获取。在网关的 SDK 中已经实现了该上链数据的组装方法，直接调用即可。

1. 接口地址：

   https://节点网关地址/api/fiscobcos/v1/node/trans

注：用户参与服务成功后可以在服务详情页面查看并下载应用链下业务系统开发所需要的应用服务配置参数、节点网关地址和应用接入密钥对，如下图：



2. 通讯方式： POST

3. 签名算法：详见 5.4.4.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一	userCode	String	是	

	标识				
2	应用唯一标识	appCode	String	是	
Body					
1	合约名称	contractName	String	是	
2	交易数据	transData	String	是	
3	合约地址	contractAddress	String	否	当用户权限为不校验权限且调用的合约方法为 constant 时将会调用该地址的合约，并且使用 contractAbi 字段的合约 ABI 解析交易数据
4	合约 ABI	contractAbi	String	否	当用户权限为不校验权限且调用的合约方法为 constant 时将会调用该地址的合约，并且使用 contractAbi 字段的合约 ABI 解析交易数据
示例：					
{"header":{"userCode":"USER000120200604232157969244 0","appCode":"app00012020060423 23057101002","tId":""},"mac":"MEUCIQCrjIeRVSt1uwFFGkr37bVM8pF0JgAWb40mKEBc5 HbpjgIgEzXRIgG+Q7obwuD2MY4EHo9sIsI1W71M+aQKOfAN3wU=","body":{"contractNam e":"BsnBaseContractk1","transData":"0xf9016fa008d8ebcb4b1f8205fd7883aa3ce9b9c84442407 0e55a3af6a5da5d7ee97d287385051f4d5c0083419ce07794866aefc204b8f8fdc3e45b908fd43d766 67d7f7680b8e4ebf3b24f0000000000000000000000000000000000000000000000000000000000000 0000600000000000000000000000000000000000000000000000000000000000000000000000000000500000000 0000000000000000000000000000000000000000000000000000000a000000000000000000000000 0000000000000000000000000000000000000000573303630340000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000000000000000000 0000000000002616100000000000000000000000000000000000000000000000000000000000000 018187801ba0324efc9e17f1d31d95535c5103083560560f836931945d03fb69acb6fd2046b5a05fa 6f574f83b3b753c9fe40649a08c2a497af1cd804e08fed2b153af20267f23", "contractAddress":"0xe2d0d414d436d8be9d52e2f40e6dd24a63faa638","contractAbi":"合约 ABI"}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	调用类型	constant	Bool	否	

2	查询信息	queryInfo	String	否	Constant 为 true 时，该字段有值
3	交易 hash	txId	string	否	Constant 为 false 时，该字段有值并且有效
4	块哈希	blockHash	String	否	Constant 为 false 时，该字段有值并且有效
5	块号	blockNumber	Int	否	Constant 为 false 时，该字段有值并且有效
6	Gas 使用值	gasUsed	Int	否	Constant 为 false 时，该字段有值并且有效
7	交易状态	status	String	否	Constant 为 false 时，该字段有值并且有效,0x0 表示交易成功,状态值参考交易回执状态
8	发送者的地址	from	String	否	Constant 为 false 时，该字段有值并且有效
9	接收者的地址	To	String	否	Constant 为 false 时，该字段有值并且有效
10	输入	Input	String	否	Constant 为 false 时，该字段有值并且有效
11	输出	output	String	否	Constant 为 false 时，该字段有值并且有效
示例					

5.4.4.7 获取交易回执接口

在智能合约执行一笔交易后，可以使用该接口根据交易的哈希值获取交易的回执信息。

1. 接口地址：

   https://节点网关地址/api/fiscobcos/v1/node/ getTxReceiptByTxHash

2. 通讯方式： POST

3. 签名算法：详见 5.4.4.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	

3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	交易哈希	txHash	string	是	
示例：					
{"header":{"appCode":"cl000620200318192657367 7572","userCode":"USER000620200318195 1281835816"},"body":{"txHash":"0x755f3e7833778f674e1b025f513f05722ba7248be43a3c9168 b880847814021a"},"mac":"MEYCIQCe6sI9zqspsy1bS6Ka9Q8O+pE7TEDWdsWj4UBSg6FM7 AIhAJrud/EoxnURQcDc47iwTdh7OdxJEJPE+raK9UaHjNaJ"}					
签名值：					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	交易回执信息	txId	string	否	code 不为 0 时为空
2	块哈希	blockHash	string		
3	块号	blockNumber	string		
4	使用 gas	gasUsed	Int		
5	发送者的地址	from	string		
6	接收者的地址	to	string		
7	合约地址	contractAddress	string		
示例					

{
"header": {
"code": 0,
"msg": "处理成功"
},
"mac":
"MEUCIQCUlhnvH9a4HN/YITf4OWgTuHmmz6qMEO89I4effHdcIwIgStdeb/dVplhn3/FoCjeS cVRyiEUhpkbze9bVm1gaXqs=",
"body": {
"blockHash":
"0x199eca276b60473dd65f8b36641684456694b419d89ef41b4953a9cdac848305",

```
            "gasUsed": 2154887,
            "blockNumber": 1,
            "txId":
"0x8ee0c68e222742b5b70878265d3fdbd3a8e0d549da42a298a4ae872ca4fbfd89",
            "contractAddress": "0x20453db36c492fa49da9fab1b80db7fa5f46b01e",
            "from": "0x08ac3132a6c7e6ca5a7fbaf0521bb8b6f370ed35",
            "to": "0x0000000000000000000000000000000000000000"
        }
}
```


5.4.4.8 获取交易信息接口

在智能合约执行一笔交易后，可以使用该接口根据交易的哈希获取交易的详细信息。

1. 接口地址：

https://节点网关地址/api/fiscobcos/v1/node/ getTxInfoByTxHash

2. 通讯方式： POST

3. 签名算法：详见 5.4.4.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	交易哈希	txHash	string	是	
示例：					
{"header":{"appCode":"cl000620200318192657367 7572","userCode":"USER000620200318195 1281835816"},"body":{"txHash":"0x755f3e7833778f674e1b025f513f05722ba7248be43a3c9168 b880847814021a"},"mac":"MEUCIQDDQudQBvHkI5tIpeTDGkQA+LPRMTA2k9u7hCZAYV obvQIgNseUfaVw8d/LxooPPWyQSo2O4EUt6wmEISgtnTcUO7k="}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	交易哈希	txId	String	是	
2	块哈希	blockHash	String	是	
3	块号	blockNumber	Int	是	
4	使用 gas	gasUserd	Int	是	
5	发送者的地址	from	String	是	
6	接收者的地址	to	String	是	
7	转移的值	value	Int	是	
8	交易的输入	input	String	是	

示例


```
{
    "header": {
        "code": 0,
        "msg": "处理成功"
    },
    "mac":
"MEQCIBMqntmqQqZXkBbrLhmXEcuOqTG4YWvlfGJmebzEDbzcAiAKKHut9MBShqpSAE
o8ts2+OBIRmEEbedjihix5FZZvrw==",
    "body": {
        "blockHash":
"0x199eca276b60473dd65f8b36641684456694b419d89ef41b4953a9cdac848305",
        "input":
"0x6080604052348015620000115760008 0fd5b5061100160008061010 0a81548173ffffffffffffff
ffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff160217905550600080905 49
06101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff1663c9
2a7801604080519081016040528060068152602001 7f745f6261736500000000000000000000
0000000000000000000000000000000815250604051826 3ffffffff167c0100000000000000000000
0000000000000000000000000000000028152600401620001019190620002 4a565b60206040
518083038160008 7803b1580156200011c57600080fd5b505af11580156200013157 3d6000803e3
d6000fd5b505050506040513d601f19601f820116820180604052506200015791908101906200 01
74565b50620002f4565b60006200016c8251620002a3565b905092915050565b600060208284031
215620001875760008 0fd5b60006200019784828501620001 5e565b91505092915050565b600062
0001ad8262000298565b808452620001c381602086016020860162000 2ad565b620001ce8162000
2e3565b602085010191505092915050565b6000601382527f626173655f6b65792c626173655f76
```

```
616c756500000000000000000000000000602083015260408201905091905056 5b600060078252
7f626173655f6964000000000000000000000000000000000000000000000000000000000060208301526040
8201905091905056 5b60006060820190508181036000830152620002668184620001a0565b905
0818103",
            "gasUsed": 100000000,
            "blockNumber": 1,
            "txId":
"0x8ee0c68e222742b5b70878265d3fdbd3a8e0d549da42a298a4ae872ca4fbfd89",
            "from": "0x08ac3132a6c7e6ca5a7fbaf0521bb8b6f370ed35",
            "to": "0x0000000000000000000000000000000000000000",
            "value": 0
        }
}
```


5.4.4.9 获取块信息接口

当数据落链并生成区块链块后，可以根据块号或者块的哈希查询响应的块的信息，其中块号和块哈希不能同时为空。当同时不为空时，优先查询块号对应的块信息。

1. 接口地址：

   https://节点网关地址/api/fiscobcos/v1/node/getBlockInfo

2. 通讯方式： POST

3. 签名算法：详见 5.4.4.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	块高	blockNumber	string	否	为空时 blockHash 不能为空
2	区块 hash	blockHash	String	否	为空时 blockNumber 不能为空

示例： { "header":{"appCode":"CL188103887322019090211431 4","userCode":"newuser"}, "body": { "blockNumber":22, "blockHash":"0xf27ff42d4be65329a1e7b11365e190086d92f9836168d0379e92642786db7ad e" }, "mac":"MEUCIC5/z5u+4sO7SyBi//nk/QJ8WzBQU4y6WWzZViHZ2XbrAiEAuK0+QKIgkdDM xeGwh18FJuPJm6TdmTzc0SaSMXfxOAI="}
签名值：

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	块哈希	blockHash	String	是	
2	块号	blockNumber	Int	是	
3	上一个块哈希	parentBlockHash	String	是	
4	块大小	blockSize	Int	是	
5	落块时间	blockTime	Int	是	毫秒格式时间戳
6		author	String	是	
7	交易信息	transactions	[]TransactionData	是	
TransactionData					
1	交易 Id	txId	String	是	
2	块哈希	blockHash	String	是	
3	块号	blockNumber	Int	是	
4	使用 gas	gasUsed	Int	是	
5	发送者的地址	from	String	是	
6	接收者的地址	to	String	是	
7	转移的值	value	Int	是	
8	交易的输入	input	String	是	
示例					

```
{
    "header": {
        "code": 0,
        "msg": "处理成功"
    },
    "mac":
"MEQCIHX8SuEn/sDiPscd5li3X1GdseyggAyC2o9L92FjhzrfAiBLyFW/rguLkqz/Lz62VtX3m7
Y1nHqcFqcNdM7Wq0wGLQ==",
    "body": {
        "blockHash":
"0x199eca276b60473dd65f8b36641684456694b419d89ef41b4953a9cdac848305",
        "blockNumber": 1,
        "parentBlockHash":
"0xa6886f12ee91470e35546432413ed372615f8d4c23fa82e8381b3e5b31219d4c",
        "blockSize": 0,
        "blockTime": 1587125168039,
        "transactions": [
            {
                "txId":
"0x8ee0c68e222742b5b70878265d3fdbd3a8e0d549da42a298a4ae872ca4fbfd89",
                "blockHash":
"0x199eca276b60473dd65f8b36641684456694b419d89ef41b4953a9cdac848305",
                "blockNumber": 1,
                "gasUsed": 100000000,
                "from": "0x08ac3132a6c7e6ca5a7fbaf0521bb8b6f370ed35",
                "to": "",
                "value": 0,
                "input":
"0x6080604052348015620000115760008fd5b5061100160008061010000a81548173ffffffffffffff
ffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff160217905550600080905549
06101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff1663c9
2a7801604080519081016040528060068152602001 7f745f626173650000000000000000000000
0000000000000000000000000000000000815250604051826 3ffffffff167c0100000000000000000000000
0000000000000000000000000000000000000028152600401620001019190620002 4a565b60206040
518083038160008780 3b1580156200011c57600080fd5b505af11580156200013157 3d6000803e3
d6000fd5b505050506040513d601f19601f8201168201806040525062000157919081019062000 1
74565b50620002f4565b60006200016c8251620002a3565b905092915050565b600060208284031
2156200018757600080fd5b60006200019784828501620001 5e565b9150509291505056 5b600062
0001ad8262000298565b808452620001c381602086016020860162000 2ad565b620001ce8162000
2e3565b602085010191505092915050565b6000601382527f626173655f6b65792c626173655f76
616c756500000000000000000000000000602083015260408201905091905056 5b600060078252
7f626173655f6964000000000000000000000000000000000000000000000000000000006020830152604
0820190509190505 65b60006060820190508181036000830152620002668184620001a0565b905
0818103602083015262000 27b8162000213565b905081810360408301520000 00"
```

```
            }
        ]
    }
}
```

5.4.4.10 获取应用内的块高接口

根据该接口可以获取应用的块高度。

1. 接口地址：

https://节点网关地址/api/fiscobcos/v1/node/getBlockHeight

2. 通讯方式： POST

3. 签名算法：详见 5.4.4.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
示例：					
{"header":{"appCode":"cl000620200318192657367757 2","userCode":"USER000620200318195 1281835816"},"body":{},"mac":"MEQCIHb2o7hb0apDukOQBXkZftETsizDBaftnHxO9A9ux5E tAiABuiFrVYPWT5FiU+Wd9HpXF/AJh0Yh2SXtL6h98m4eZw=="}					
签名值：					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败

2	响应信息	msg	String	否	code=0 时可为 null
body					
1	块高	data	string	否	code 不为 0 时为空
示例					


```
{
   "header": {
      "code": 0,
      "msg": "处理成功"
   },
   "mac":
"MEQCICtCOdv4ZL72M3WoA9nAei2P0/PpKjlgI0Y5qeuzg61uAiA9D3TcB/+b2RMuNwVq+X
0vgiglHfM5NBhoTJPR0gCPMA==",
   "body": {
      "data": "4"
   }
}
```


5.4.4.11 获取应用内的交易总数接口

根据该接口可以获取应用的交易总数。

1. 接口地址：

   https://节点网关地址/api/fiscobcos/v1/node/ getTxCount

2. 通讯方式： POST

3. 签名算法：详见 5.4.4.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
示例：					


```
{"header":{"appCode":"cl000620200318192657367 7572","userCode":"USER000620200318195
1281835816"},"body":{},"mac":"MEYCIQCUW4d1/x2cV45SiFgkgF8h73+/Q/PocgxNZe0qKT
```

WLsQIhAK4jcRmEVXSd+nYRdE375y7STKczKt38uYp3rU1sf7FS"}
签名值：

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	交易信息	data	string	否	code 不为 0 时为空

示例


```
{
   "header": {
      "code": 0,
      "msg": "处理成功"
   },
   "mac":
"MEQCIGgXINn3B9d/hC/ow0IJvi5eKDj59QbZRFdrCqcUeNCgAiApI4jkwhTY33qev1RwsJ3v
eDBKXokvIiSe3ck7SKlxmg==",
   "body": {
      "data":
"{\"txSum\":5,\"blockNumber\":5,\"txSumRaw\":\"0x5\",\"blockNumberRaw\":\"0x5\"}"
   }
}
```


5.4.4.12 获取块内的交易总数接口

可以在 FISCO BCOS 应用内根据块号查询块内的交易总数，其中块号不能为空。

1. 接口地址：

https:// 节点网关地址 /api/fiscobcos/v1/node/getTxCountByBlockNumber

2. 通讯方式： POST

3. 签名算法：详见 5.4.4.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	块号	blockNumber	string	是	
示例：					
{ "header":{"appCode":"CL188103887322019090211431 4","userCode":"newuser"}, "body": { "grouId":1, "blockNumber":22, }, "mac":"MEYCIQD0sDKn3rPs3xRGkpGxy2uh/TpYRLvjDML4AKwPbqi1eAIhAKlQ7ks86ni9cFag+jLmhEHYlSdF8/HrxBg30IIoBoOo"}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	块交易总数信息	data	string	否	code 不为 0 时为空
data					
示例					
{ "header": { "code": 0, "msg": "处理成功" },					

```
  "mac":
"MEUCIQCMFbVhfH9X8pJ1mNI3YpzKIBcXCpfmf2AniF/42ak9EwIgTWDEF+xW5l39ZDUn
DSSSc8Zv8J1glEf9izp16eW/Rn4=",
  "body": {
    "data": "1"
  }
}
```


5.4.4.13 链码事件注册接口

可以通过该接口注册 FISCO BCOS 的出块事件或者合约事件，当事件被触发时，系统将向注册的通知地址发送事件内容。

1. 接口地址： https://节点网关地址/api/fiscobcos/v1/event/register
2. 通讯方式： POST
3. 签名算法：详见 5.4.4.1 应用接入签名算法
4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	事件类型	eventType	String	是	1：出块事件 2：合约事件
2	合约地址	contractAddress	String	否	EventType 为 1 时可以为空，为 2 时，不可与 contractName 同时为空
3	合约名	contractName	String	否	EventType 为 1 时可以为空，为 2 时，不可与 contractAddress 同时为空
4	通知地址	notifyUrl	String	是	
5	附件参数	attachArgs	String	否	
示例：					
{"header":{"userCode":"USER000120200604232157969244 0","appCode":"app00012020 06042323057101002","tId":""},"mac":"MEUCIQCMP1ToZS5e8S94kYZ/8y5XfeyjRyUrPFpeI QMES3SGpQIgO8b6O8Kk/qpNTo1vbNTwyAYNaw6HBi9OkAH8Rp23j8s=","body":{"event					

Type":1,"contractAddress":"0x866aefc204b8f8fdc3e45b908fd43d76667d7f76","contractName":"BsnBaseContractk1","notifyUrl":"http://127.0.0.1:18080","attachArgs":"abc=123"}}

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	
body					
1	事件 Id	eventId	string	是	code 不为 0 时为空
data					
示例					


```
{
   "header": {
      "code": 0,
      "msg": "处理成功"
   },
   "mac":
"MEUCIQDYSTwYhh6EDHT5Z7ukcqXW9LMjZW6WPnrv8Xt14RuH2AIgIwa5K7NK4/TThzs8z6VfkpNNJU+dzAXeypFmfjkru88=",
   "body": {
      "eventId": "xxxxxxxxxxxxxxxxxxxxxxxxxxx"
   }
}
```


5.4.4.14 链码事件查询接口

该接口可以查询已经注册成功的链码事件的列表。

1. 接口地址：

   https://节点网关地址/api/fiscobcos/v1/event/query

2. 通讯方式： POST

3. 签名算法：详见 5.4.4.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
示例：					
{"header":{"userCode":"USER000120200604232157969244 0","appCode":"app000120200604 2323057101002","tId":""},"mac":"MEUCIQC2NTuUlsxQSWPpZwwhJK9zXEMaeYZC04Ar0P5T wyp5AQIgFvZrskasuLiYfOGxd1F9TCetWHIfENg8BCiYfNS1xGk=" }					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	
body					
1	出块事件	blockEvent	[]blockEvent	是	code 不为 0 时为空
2	合约事件	contractEvent	[]contractEvent	是	
blockEvent					
1	出块事件	eventId	string	是	code 不为 0 时为空
2	应用编号	appcode	String	是	
3	用户编号	userCode	String	是	
4	通知地址	notifyUrl	String	是	
5	附件参数	attachArgs	String	否	
6	创建时间	createTime	String	是	UTC 时间
contractEvent					
1	出块事件	eventId	string	是	
2	应用编号	appcode	String	是	
3	用户编号	userCode	String	是	
4	通知地址	notifyUrl	String	是	
5	附件参数	attachArgs	String	否	
6	创建时间	createTime	String	是	UTC 时间
7	合约地址	contractAddress	String	是	

示例


```
{
   "header": {
      "code": 0,
      "msg": "处理成功"
   },
   "mac":
"MEUCIQCQ/RjmlVkLKZw6jcLKBPh1BwK4EIQE001vUAKPVq1HTgIgXUQ7Bn+y8D8xQxYU
wtZOoh/bpteAPCUtKXZeAiN7cMU=",
   "body": {
      "blockEvent": [
         {
            "eventId": "ba537419953e4e219ceb0fe26ad5e125",
            "appCode": "app000120200604232305710100 2",
            "userCode": "USER000120200604232157969244 0",
            "notifyUrl": "http://127.0.0.1:18080",
            "attachArgs": "abc=123",
            "createTime": "0001-01-01 00:00:00.000 +0000 UTC"
         }
      ],
      "contractEvent": [
   {
            "eventId": "ba537419953e4e219ceb0fe26ad5e126",
            "appCode": "app000120200604232305710100 2",
            "userCode": "USER000120200604232157969244 0",
            "notifyUrl": "http://127.0.0.1:18080",
            "attachArgs": "abc=123",
            "createTime": "0001-01-01 00:00:00.000 +0000 UTC",
            “contractAddress”:” 0x866aefc204b8f8fdc3e45b908fd43d76667d7f76”
         }

      ]
   }
      }
```


5.4.4.15 链码事件取消接口

该接口可以取消已经注册成功的链码事件。

1. 接口地址：

https://节点网关地址/api/fiscobcos/v1/event/remove

2. 通讯方式： POST

3. 签名算法：详见 5.4.4.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	事件编号	eventId	string	是	
示例：					
{"header":{"userCode":"USER000120200604232157969244 0","appCode":"app00012020 06042323057101002","tId":""},"mac":"MEQCIARwCd9VEkIP6ISniKPnJRjyPJ/nqwf7kZw0r BmzSAxlAiATW7zcRM+yQ+JJTKxXV6L/S7zUT1r6Hctlb0ccAVTc7Q==","body":{"eventId" :"ba537419953e4e219ceb0fe26ad5e125"}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	
示例					


```
{
  "header": {
    "code": 0,
    "msg": "处理成功"
  },
  "mac":
"MEUCIQCjkDCMi1S38CbKFmd/zN6HpacIi798S3nE/bqO8gU7ggIgCGpX3DQk1Ulf4sevbKJ
6S2cDopHFJB1bAqDOvOarGFk=",
  "body": null
    }
```

5.4.4.16 链码事件通知报文信息

该接口是由链下业务系统侧实现。城市节点网关在监听到注册的链码事件触发后调用该接口将事件执行结果通知给链下业务系统。

链下业务系统在收到城市节点网关通知后应返回一个包含 success 的字符串，否则城市节点网关将尝试分别在第 3、12、27、48 秒后再次发起通知，共计 5 次。

1. 通讯方式： POST

2. 签名算法：详见 5.4.4.1 应用接入签名算法

3. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	事件 Id	eventId	String	是	
2	城市编码	orgCode	String	是	
3	附加参数	attachArgs	String	否	注册时填写的附加参数
4	随机字符串	nonceStr	String	是	业务平台可根据该值判断是否接收过通知，同一个业务的多次通知中，该字符串不变
5	事件类型	eventType	String	是	
6	事件数据	eventData	String	是	
示例：					
{"header":{"userCode":"USER000120200604232157969244 0","appCode":"app000120200604232305710100 2"},"body":{"eventId":"5b5b865f8dc94ae59d215cf26aa81d69","orgCode":"ORG2020041114171692360","appCode":"app000120200604232305710100 2","attachArgs":"abc=123","nonceStr":"52f080f27ff045eb87e21812d12cee40","eventType":1,"eventData":"{\"appId\":\"app000120200604232305710100 2\",\"blockNumber\":17,\"eventType\":1,\"groupId\":135}"},"mac":"MEUCIQD3Sp6xuI4DHy/GOb9z3nH6kQisEzfXvZ/Hn/mfZXIAOgIgYsISRfBKSJGt4FrmxETflfR4A8VenCZHvxthMFUWRkc="}					

5.4.4.17 调用智能合约接口的交易状态描述

在密钥托管和上传公钥两种模式下，链下业务系统请求城市节点网关的调用链码接口时，节点网关返回的交易处理状态说明具体如下：

status(十进制/十六进制)	message	含义
0(0x0)	None	正常
1(0x1)	Unknown	未知异常
2(0x2)	BadRLP	无效 RLP 异常
3(0x3)	InvalidFormat	无效格式异常
4(0x4)	OutOfGasIntrinsic	部署的合约长度超过 gas 限制/调用合约接口参数超过 gas 限制
5(0x5)	InvalidSignature	无效的签名异常
6(0x6)	InvalidNonce	无效 nonce 异常
7(0x7)	NotEnoughCash	cash 不足异常
8(0x8)	OutOfGasBase	调用合约的参数过长（RC 版本）
9(0x9)	BlockGasLimitReached	GasLimit 异常
10(0xa)	BadInstruction	错误指令异常
11(0xb)	BadJumpDestination	错误目的跳转异常
12(0xc)	OutOfGas	合约执行时 gas 不足 / 部署的合约长度超过最长上限
13(0xd)	OutOfStack	栈溢出异常
14(0xe)	StackUnderflow	栈下限溢位异常
15(0xf)	NonceCheckFail	nonce 检测失败异常
16(0x10)	BlockLimitCheckFail	blocklimit 检测失败异常
17(0x11)	FilterCheckFail	filter 检测失败异常
18(0x12)	NoDeployPermission	非法部署合约异常
19(0x13)	NoCallPermission	非法 call 合约异常
20(0x14)	NoTxPermission	非法交易异常
21(0x15)	PrecompiledError	precompiled 错误异常
22(0x16)	RevertInstruction	revert 指令异常
23(0x17)	InvalidZeroSignatureFormat	无效签名格式异常
24(0x18)	AddressAlreadyUsed	地址占用异常
25(0x19)	PermissionDenied	无权限异常
26(0x1a)	CallAddressError	被调用的合约地址不存在

5.4.5 城市节点网关 XuperChain API

5.4.5.1 应用接入签名算法

链下业务系统向城市节点网关发送交易请求时，需使用应用接入密钥对中的私钥对交易请求报文进行数字签名，节点网关接收到交易请求报文后，使用已上传或托管的密钥对中的公钥对该报文进行验签。只有在交易报文验签通过时，网关才会对交易请求报文进行后续的交易处理。具体签名算法如下：

1. 组装签名字符串

将请求参数按照文档参数表格中的顺序转换为字符串进行拼接，其中请求参数优先拼接 Header 中的 UserCode、AppCode；返回参数优先拼接 code、msg，然后再次按照文档参数顺序拼接 Body 中的参数。

2. 不同类型的转换格式

类型	规则	示例	结果
String	不转换	abc	abc
Int/int64/long	十进制转换	-12	-12
Float	十进制转换，小数位参考备注	1.23	1.23
Bool	转换为“true”或者“false”	真	true
Array	按照参数顺序和类型拼接	[“abc”,”xyz”]	abcxyz
Map[key]value	按照顺序拼接 key 和 value	[“a”:1,”b”:2]	a1b2
Object	将对象内的属性按照文档循序依据上述格式转换	{“name”:”abc”,”secret”:”123456”}	abc123456

3. 签名规则

- 哈希值计算：将需要签名的字符串按照 UTF-8 编码做 SM3 计算；

- 对哈希值获取签名：将哈希值与私钥进行 SM2 签名计算；
- 将签名结果做 Base64 计算。

4. 示例

参数:

{"header":{"userCode":"user01","appCode":"app01"},"mac":"","body":{“userId”:”abc”,”list”:[“abc”,”xyz”]}}

结果：user01app01abcabcxyz

5.4.5.2 密钥证书模式

1．密钥托管模式

如第五章描述，用户在参与应用时需要两个密钥对：应用接入密钥对和用户交易密钥对。在该模式下密钥对不需要用户在本地生成，而是由 BSN 生成并且托管，用户只需要从 BSN 门户下载使用即可。

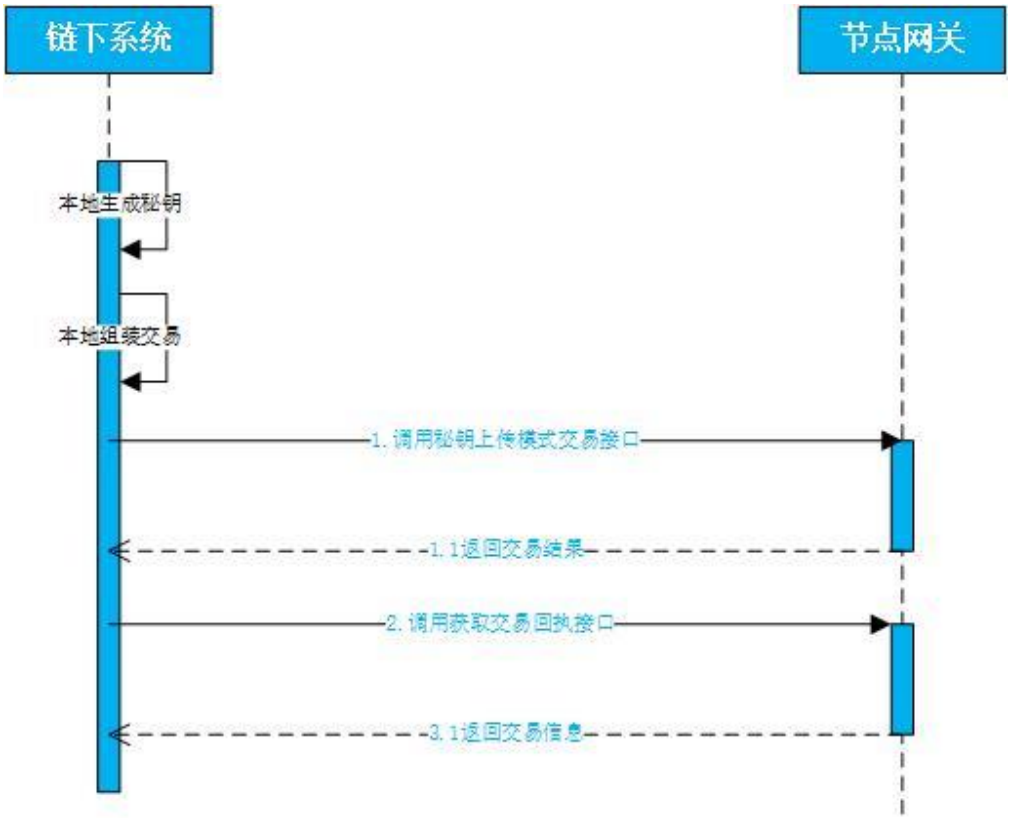
- **应用接入密钥对：**用户在参与应用服务成功后，BSN 会生成一对与所参与应用的底层框架算法一致的公私密钥对托管在 BSN 中，用户可以在 BSN 门户内我的证书列表下载该公私钥对。在其链下业务系统调用城市节点网关接口时需要使用该密钥对的私钥进行交易签名。节点网关使用托管的公钥对签名进行验签。
- **用户交易密钥对：**是用户调用智能合约处理交易的身份。在密钥托管模式下，用户参与应用服务成功后，其链下业务系统可以调用城市节点网关的注册用户接口生成一个链上用户账户并将用户的交易密钥对托管在城市节点中，在交易时使用用户的 UserId 即可进行交易处理。

交易流程:

2. 上传公钥模式

用户在参与应用时需要两个密钥对：应用接入密钥对和用户交易密钥对。在该模式下用户需要自己在本地生成公私钥对，私钥本地保存。

- **应用接入密钥对：**用户在 BSN 门户参与应用服务时，需要在本地生成一对与该应用服务的底层框架算法一致的公私钥对，并且将公钥提交到 BSN 中。链下业务系统在调用城市节点网关接口时使用该密钥对的私钥进行交易签名。节点网关收到交易请求后使用用户上传的公钥对签名进行验签，以验证交易身份的合法性。
- **用户交易密钥对：**是用户调用链码处理交易的身份。在上传公钥模式下，用户需要在自己的链下业务系统中生成用户交易公私钥对，在参与交易时，使用自己生成的公私钥对交易数据进行本地签名。然后将签名后的数据上传即可。



5.4.5.3 获取应用信息接口

调用该接口可以获取应用的基本信息，可以在上传公钥模式中使用。

1. 接口地址： https://节点网关地址/api/app/getAppInfo
2. 通讯方式： POST
3. 签名算法：不需要
4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	否	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					

示例：					
{"header":{"userCode":"USER000120201020153939008609 0","appCode":"app00012020102210 38364886804","tId":""},"mac":"","body":{}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：成功 -1：失败
2	响应信息	msg	String	是	
Body					
1	应用名	appName	String	是	
2	应用类型	appType	String	是	
3	应用密钥托管类型	caType	Int	是	1：托管 2：非托管
4	应用密钥类型	algorithmType	Int	是	1：SM2 2:ECDSA(secp256r1)
5	该城市 MSPID	mspId	String	是	
6	应用链名称	channelId	String	是	Fabric 对应 channelId，fisco 对应 groupId

示例


```
{
    "header": {
        "code": 0,
        "msg": "success"
    },
    "mac":
"MEYCIQCkspyNHVdvZ/vrfY9coU7r7ktRdU2ZG4FtdrzQRnv8qgIhAMNbmOhClsH7DNl77D
DffDMs/Nh19gOywJuFPhWCkzXX",
    "body": {
        "appName": "xuperchiangongyao",
        "appType": "xuperchain",
        "caType": 2,
        "algorithmType": 1,
        "mspId": "",
        "channelId": "app000120201022103836488 6804"
    }
}
```

5.4.5.4 注册用户接口

用户参与一个 XuperChain 的应用服务成功后，需要通过链下业务系统调用本接口生成进行链上交易处理的用户账户和账户地址。

1. 接口地址：

https://节点网关地址/api/xuperchain/v1/user/register

2. 通讯方式： POST

3. 签名算法：详见 5.4.5.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	用户名	userId	String	是	注册的用户名
示例：					

<pre>{
"header": {
"appCode": "CL1881038873220190902114314",
"userCode": "newuser"
},
"body": {
"userId": "abc"
},
"mac": "MEYCIQDUCidxeWRLhPjrYhfJIhkiJS+67lwnV/K4j+fkvpsRZgIhALtJvPwORTB9T6HwqdyDMvIgFDMd5p5CX61+pd6KFfqZ"
}</pre>					

签名值：					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	用户编号	userId	string	是	
2	用户地址	userAddr	string	是	

示例


```
{
   "header": {
      "code": 0,
      "msg": "处理成功"
   },
   "mac":
"MEUCIBiWAmyojuNPUHeZAPV006TFq1HSbzDUXZQBNLITcOtMAiEAuQ1mfxEkSJvqRl
qqx6fRE8sO03woVxYSiJiuSDjEBx4=",
   "body": {
      "userId": " abc",
      " userAddr": "2CcmGvEFpC16MAn3MzheMy5eR4FqzaPcHV"
   }
}
```


5.4.5.5 密钥托管模式下调用智能合约接口

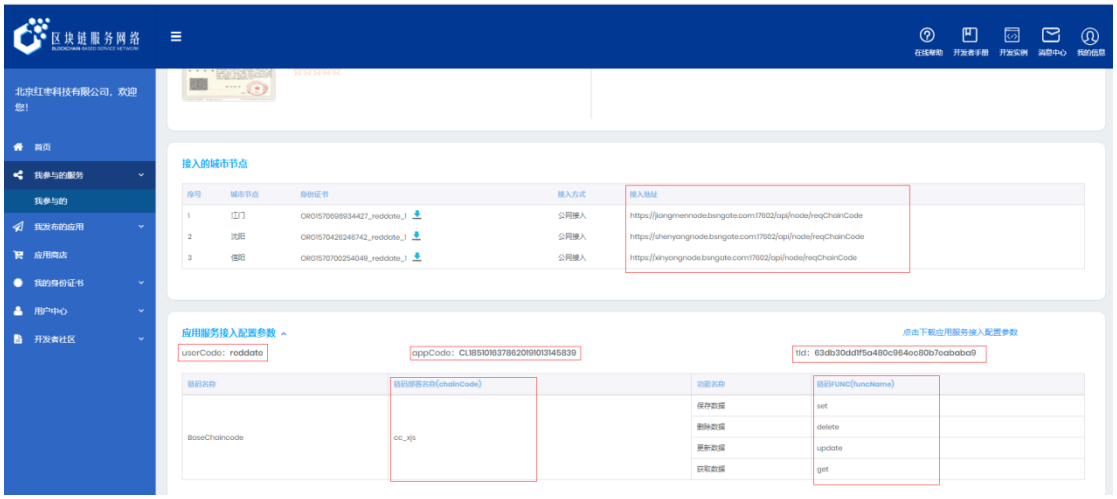
链下业务系统在连接节点网关时，需要按照接口说明在请求中加入相应的请求参数，调用节点网关以后，节点网关会返回智能合约的执行结果。

1. 接口地址：

   https://节点网关地址

   /api/xuperchain/v1/node/reqChainCode

注：用户参与服务成功后可以在服务详情页面查看并下载应用链下业务系统开发所需要的应用服务配置参数、节点网关地址和应用接入密钥对，如下图：



2. 通讯方式： POST

3. 签名算法：详见 5.4.5.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
Body					
1	用户编号	userId	String	是	
2	用户地址	userAddr	string	是	
3	合约名称	contractName	String	是	
4	方法名称	funcName	String	是	
5	方法参数	funcParam	string	否	字符串，参数转为 json 字符串
示例：					

```
{
    "header":{
        "appCode":" CL188103887322019090211431 4",
        "userCode":" newuser "
    },
    "body":{
```

```
                "userId":" abc",
                " userAddr": "2CcmGvEFpC16MAn3MzheMy5eR4FqzaPcHV",
                "contractName":"bsnbase",
                "funcName":"insert_data",
                "funcParam":"{\"baseKey\":\"dev_001\",\"baseValue\":\"aaron\"}"
            },

"mac":"MEYCIQCRQyABwBai79zbjl2CJsAnhrisO6kl4hzg/W42+oZgLAIhAOXpXJBcSOl3d4h
cgcs6AvA0F6qnRtU2WwIqNyC1LqCh"
        }
```


5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 是可为 null
body					
1	交易标识	txId	string		Invoke 合约，不为空
2	查询信息	queryInfo	String		Query 合约，不为空

示例


```
{
    "header": {
        "code": 0,
        "msg": "处理成功"
    },
    "mac":
"MEQCIFbBW4AEzdf1g35vCwFphQM3VyO22FiXAgpLdkJ5+IhyAiA38xOJSaiCj4DTcmBW
Xhn6niaGFTd/bmq7cXGO+xVupg==",
    "body": {
        "txId":
"0xd568dab2920a2c8a24883980da049269d0489dbba8fe04a20daa3762c5035dc0"
    }
}
```

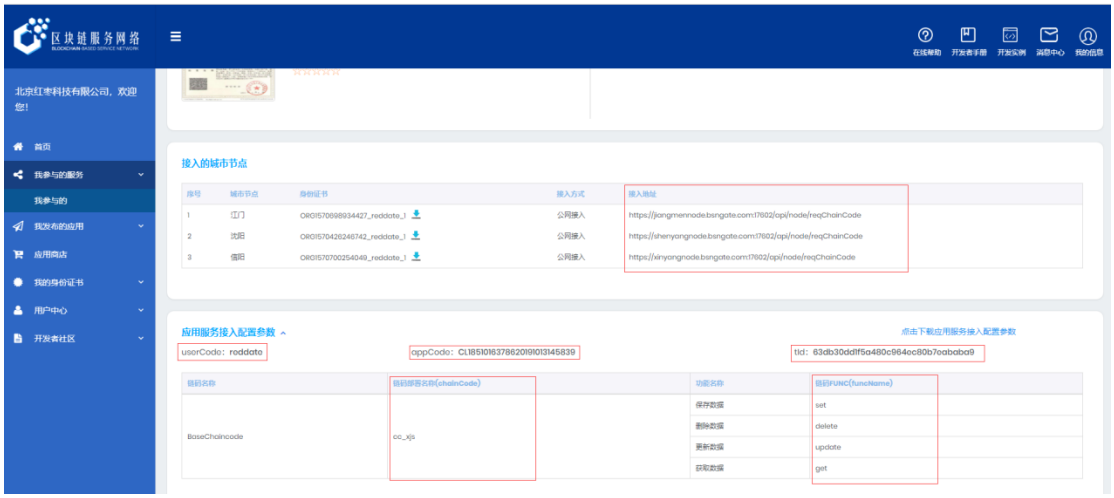
5.4.5.6 密钥上传模式下调用智能合约接口

链下业务系统在连接节点网关时，需要按照接口说明在请求中加入相应的请求参数，调用节点网关以后，节点网关会返回智能合约的执行结果。在上传公钥模式的交易中，链上交易的私钥由用户自己生成和保存，然后由客户端在本地进行上链数据的组装和签名，将签名后的数据上传至节点网关，网关将数据转发至相应的区块链节点发起交易请求。在网关的 SDK 中已经实现了该上链数据的组装方法，直接调用即可。

1. 接口地址：

https://节点网关地址/api/xuperchain/v1/node/trans

注：用户参与服务成功后可以在服务详情页面查看并下载应用链下业务系统开发所需要的应用服务配置参数、节点网关地址和应用接入密钥对，如下图：



2. 通讯方式： POST

3. 签名算法：详见 5.4.5.1 应用接入签名算法

5.4.5.7 获取交易信息接口

在智能合约执行一笔交易后，可以使用该接口根据交易的哈希获

取交易的详细信息。

1. 接口地址：

   https://节点网关地址/api/xuperchain/v1/node/getTxInfoByTxHash

2. 通讯方式： POST

3. 签名算法：详见 5.4.5.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	交易哈希	txHash	string	是	
示例：					

```
{
    "header":{
        "appCode":" CL1881038873220190902114314",
        "userCode":" newuser "
    },
    "body":{
        " txHash ":"
134d69f1a4d98cc6f52590cb338a5bc4316cacf5a6ad5c858de224a0ef1288b6"
    },

"mac":"MEUCIHiDrLC6Z82/tByyIkS6dj/pjCQjkqE7dN/RhB8M9xc4AiEAuXaa5beCMJz/2xKQ
1IeaMlaoAZOSfBOFqx6SnFqOw4w="
}
```

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功

					-1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	交易哈希	txId	String	是	
2	块哈希	blockId	String	是	
3	Tx 格式版本号	Version	String	是	
4	合约的调用请求	contractRequests	[]contractRequestData	是	
5	收到 tx 的时间戳	receivedTimestamp	long	是	
contractRequestData					
1	合约名称	contractName	String	是	
2	方法名称	methodName	String	是	
3	参数	args	string	是	
示例					


```
{
    "header": {
        "code": 0,
        "msg": "处理成功"
    },
    "mac": "",
    "body": {
        "txId": "134d69f1a4d98cc6f52590cb338a5bc4316cacf5a6ad5c858de224a0ef1288b6",
        "blockId":
"52e92335bf7711ac3cdd643672f1dfc2747555b5f27667b3d52802ecb1cfa11b",
        "version": 1,
        "contractRequests": [
        ],
        "receivedTimestamp": 1593092737000830499
    }
}
```


5.4.5.8 获取块信息接口

当数据落链并生成区块链块后，可以根据块高或者块的哈希查询响应的块的信息，其中块高和块哈希不能同时为空。当同时不为空时，优先查询块高对应的块信息。

1. 接口地址：

https://节点网关地址/api/xuperchain/v1/node/getBlockInfo

2. 通讯方式： POST

3. 签名算法：详见 5.4.5.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	块高	blockHeight	uInt64	否	不可同时为空
2	区块 hash	blockHash	String	否	不可同时为空

示例：


```
{
    "header":{
        "appCode":"CL188103887322019090211431 4",
        "userCode":"newuser"
    },
    "body":
    {
        "blockHeight":0,

"blockHash":"0xf27ff42d4be65329a1e7b11365e190086d92f9836168d0379e92642786db7ade"
    },

"mac":"MEQCIGzS59y62I8EtDXiHCs37Q2c82tv6fap8gdnoHqoSNvuAiAtejOiLGhUe5ZVvnJE
NazQzW4HUUgNHfsV/BEl3A3dHg=="
}
```


签名值：

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为

					null
body					
1	区块版本	Version	Int	是	
2	块哈希	blockId	String	是	
3	当前区块前置依赖区块 id	PreHash	String	是	
4	区块所在高度	height	int	是	
5	打包区块时间戳	timestamp	long	是	
6	交易内容	transactions	**[]**TransactionData	是	
7	区块包含的交易数	txCount	int	是	
8	当前区块的后继区块 id	nextHash	String	是	
TransactionData					
1	交易哈希	txId	String	是	
2	块哈希	blockId	String	是	
3	Tx 格式版本号	Version	String	是	
4	合约的调用请求	contractRequests	[]contractRequestData	是	
5	收到 tx 的时间戳	receivedTimestamp	long	是	
contractRequestData					
1	合约名称	contractName	String	是	
2	方法名称	methodName	String	是	
3	参数	args	string	是	

示例


```
{
    "header": {
        "code": 0,
        "msg": "处理成功"
    },
    "mac":
"MEUCIA1qT8akZGUMG8leQYNhGhJSZULpUkP4ZPJYq1aRWagbAiEAiLES1A0NYsWa5NwuiIAOY3I/W8iCpSASNv+TAj+rgy8=",
    "body": {
        "version": 1,
        "blockid":
"52e92335bf7711ac3cdd643672f1dfc2747555b5f27667b3d52802ecb1cfa11b",
        "preHash":
"974264986dbc45463b9d14c640bc89f4fa11ed258dea61a7d90d5f47505b4e14",
        "height": 10,
        "timestamp": 1594092777000880499,
        "transactions": [
```

```
                {
                    "txId":
"134d69f1a4d98cc6f52590cb338a5bc4316cacf5a6ad5c858de224a0ef1288b6",
                    "blockId":
"52e92335bf7711ac3cdd643672f1dfc2747555b5f27667b3d52802ecb1cfa11b",
                    "version": 1,
                    "contractRequests": null,
                    "receivedTimestamp": 0
                }
            ],
            "txCount": 1,
            "nextHash":
"6ede8695bb8437990fce0d9becdd012b5f02ee3873694b0a005037c0d919dde5"
        }
}
```


5.4.5.9 链码事件注册接口

参与的应用服务中，如果需要根据链码事件触发链下业务系统进行后续业务处理时，可调用该接口注册需要监听的链码事件。

1. 接口地址：

   https://节点网关地址/api/xuperchain/v1/event/register

2. 通讯方式： POST

3. 签名算法：详见 5.4.5.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
Header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
Body					
1	合约名称	contractName	String	是	
2	合约事件 key	eventKey	String	是	。
3	事件通知地址	notifyUrl	String	是	接收监听到的事件的地

					址
4	附加参数	attachArgs	String	否	
示例：					
{"header":{"userCode":"USER000120201020153939008609 0","appCode":"app00012020102210 38364886804","tId":""},"mac":"MEUCIEjMeYepitdvXJLndgf7l1ufJDlC8YBxGDX/s/gkT5BBA iEA7Wzbgfnlh/dUjEeY1FSfZJEfSyHyW5fyA5g+h+boBZA=","body":{"contractName":"cc_app xc_01","eventKey":"increase_event","notifyUrl":"http://192.168.7.141:8088/revNotify1","attach Args":"abc=123"}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
Header					
1	响应标识	code	int	是	0：注册成功 -1：注册失败
2	响应信息	msg	String	是	
Body					
1	事件编码	eventId	String	是	
示例					

```
{
    "header": {
        "code": 0,
        "msg": "success"
    },
    "mac":
"MEYCIQDFXWBPvvhqNUyBqOKL1Muq164wgL/FCUTd12AOAmak+QIhAL2vkO8KgZ74t
k++D4qqYWCrX52QfSlIeJTn/FXeLOh7",
    "body": {
        "eventId": "9476afd7414e4ad2bce853b4e8f44bd4"
    }
}
```

5.4.5.10 链码事件查询接口

用于查询已注册的链码事件监听列表。

1. 接口地址：

https://节点网关地址/api/xuperchain/v1/event/query

2. 通讯方式： POST

3. 签名算法：详见 5.4.5.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	否	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
示例：					
{"header":{"userCode":"USER000120201020153939008609 0","appCode":"app00012020102210 38364886804","tId":""},"mac":"MEUCIA+/chkZV6Q0ijOoJ40oqIXAfhvRtzbnPfGltOdhTag3Ai EA1XvoFYGyEJePR1vprGKTO6pE35qpwg6wmXzSQ2BqD40=","body":null}					

5.4.5.11 链码事件注销接口

注销已注册的链码事件监听。

1. 接口地址：

https://节点网关地址/api/xuperchain/v1/event/remove

2. 通讯方式： POST

3. 签名算法：详见 5.4.5.1 应用接入签名算法

5.4.5.12 链码事件通知报文信息

该接口是由链下业务系统侧实现。城市节点网关在监听到注册的链码事件触发后调用该接口将事件执行结果通知给链下业务系统。

链下业务系统在收到城市节点网关通知后应返回一个包含 success 的字符串，否则城市节点网关将尝试分别在第 3、12、27、48 秒后再次发起通知，共计 5 次。

1. 通讯方式： POST

2. 签名算法：详见 5.4.5.1 应用接入签名算法

3. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	事件 Id	eventId	String	是	
2	城市编码	orgCode	String	否	备用，目前为空
3	附加参数	attachArgs	String	否	注册时填写的附加参数
4	事件数据	Payload	String	是	
5	随机字符串	nonceStr	String	是	业务平台可根据该值判断是否接收过通知，同一个业务的多次通知中，该字符串不变

示例：


```
{
"header": {
    "userCode": "USER000120201020153939008609O",
    "appCode": "app000120201022103836488680 4"
},
"body": {
    "eventId": "f4c6912f5b0540399ff080ef798763fa",
    "orgCode": "",
    "attachArgs": "abc=123",
    "payload":
"{\"bcname\":\"app000120201022103836488680 4\",\"blockid\":\"cab4bf0b90b69ab4ca3e30dca7
d52365376ac277f417648bf03751d7b4d84952\",\"block_height\":18184,\"txs\":[{\"txid\":\"e638c
322a747b4f85ee7175ca7eec3d0d5ade3c970478b762d4b01c26035d951\",\"events\":[{\"contract\"
:\"cc_appxc_01\",\"name\":\"increase_event\",\"body\":\"12\"}]}]}",
    "nonceStr": "40a182e08f6e454daf8a2a0d042dbb44"
},
"mac":
"MEYCIQCcsoW0RJjLY4UX+yjWlSwRdYGaprxFyR/9F2b5gTQOUwIhAI98SanNujqgtjaAC/5
```

```
YxOxxLPwVMevPq/XLRpHhtrHi"
    }
```

5.4.5.13 调用智能合约接口的交易状态描述

在密钥托管和上传公钥两种模式下，链下业务系统请求城市节点网关的调用链码接口时，节点网关返回的交易处理状态说明具体如下：

状态码	message	含义
0	SUCCESS	正常
1	UNKNOW_ERROR	未知异常
2	CONNECT_REFUSE	连接拒绝
7	TX_NOT_FOUND_ERROR	无效的交易
8	TX_SIGN_ERROR	交易签名错误
9	BLOCKCHAIN_NOTEXIST	链不存在
10	VALIDATE_ERROR	验证失败
12	CONFIRM_BLOCK_ERROR	确认块失败
16	BLOCK_EXIST_ERROR	块已存在
17	ROOT_BLOCK_EXIST_ERROR	主链块已存在
19	TX_DUPLICATE_ERROR	交易重复
20	SERVICE_REFUSED_ERROR	服务拒绝
21	TXDATA_SIGN_ERROR	交易数据签名错误
34	RWACL_INVALID_ERROR	读写集权限无效
36	TX_VERSION_INVALID_ERROR	交易版本无效
38	ACCOUNT_CONTRACT_STATUS_ERROR	合约账号状态错误
40	TX_VERIFICATION_ERROR	交易验证错误

5.4.6 城市节点网关 CITA API

城市节点网关部署在各个城市节点，接收应用系统的请求，使用托管的用户身份信息，向基于 CITA 的应用的智能合约发起访问并返回智能合约的执行结果。节点网关的调用是通过向区块链服务的各个城市节点的网关服务发送 HTTP 请求来实现。节点网关负责验证用户身份信息和应用信息，通过用户身份信息和应用信息以及需要访问的智能合约、智能合约方法来传递智能合约参数、获取智能合约执行结果的服务通道。

5.4.6.1 应用接入签名算法

链下业务系统向城市节点网关发送交易请求时，需使用应用接入密钥对中的私钥对交易请求报文进行数字签名，节点网关接收到交易请求报文后，使用已上传或托管的密钥对中的公钥对该报文进行验签。只有在交易报文验签通过时，网关才会对交易请求报文进行后续的交易处理。具体签名算法如下：

1. 组装签名字符串

将请求参数按照文档参数表格中的顺序转换为字符串进行拼接，其中请求参数优先拼接 Header 中的 UserCode、AppCode；返回参数优先拼接 code、msg，然后再次按照文档参数顺序拼接 Body 中的参数。

2. 不同类型的转换格式

类型	规则	示例	结果
String	不转换	abc	abc
Int/int64/long	十进制转换	-12	-12
Float	十进制转换，小数位参考备注	1.23	1.23
Bool	转换为“true”或者“false”	真	true
Array	按照参数顺序和类型拼接	[“abc”,”xyz”]	abcxyz
Map[key]value	按照顺序拼接 key 和 value	[“a”:1,”b”:2]	a1b2
Object	将对象内的属性按照文档循序依据上述格式转换	{“name”:”abc”,”secret”:”123456”}	abc123456

针对使用国密算法的 CITA 框架应用，使用以下规则

- 哈希值计算：将按照上述规则转换后的待签名的字符串按照 UTF-8 编码做 SM3 计算；
- 对哈希值获取签名：哈希值与私钥进行 SM2 加密签名计算；

- 将签名结果做 Base64 计算。

3. 示例

参数:

{"header":{"userCode":"user01","appCode":"app01"},"mac":"","body":{“userId”:”abc”,”list”:[“abc”,”xyz”]}}

结果：user01app01abcabcxyz

5.4.6.2 密钥证书模式

1. 密钥托管模式

如第五章描述，用户在参与应用时需要两个密钥对：应用接入密钥对和用户交易密钥对。在该模式下密钥对不需要用户在本地生成，而是由 BSN 生成并且托管，用户只需要从 BSN 门户下载使用即可。

- **应用接入密钥对：**用户在参与应用服务成功后，BSN 会生成一对与所参与应用的底层框架算法一致的公私密钥对托管在 BSN 中，用户可以在 BSN 门户内我的证书列表下载该公私钥对。在其链下业务系统调用城市节点网关接口时需要使用该密钥对的私钥进行交易签名。节点网关使用托管的公钥对签名进行验签。
- **用户交易密钥对：**是用户调用智能合约处理交易的身份。在密钥托管模式下，用户参与应用服务成功后，其链下业务系统可以调用城市节点网关的注册用户接口生成一个链上用户账户并将用户的交易密钥对托管在城市节点中，在交易时使用用户的 UserId 即可进行交易处理。

交易流程:

5.4.6.3 获取应用信息接口

调用该接口可以获取应用的基本信息，可以在上传公钥模式中使用。

1. 接口地址： https://节点网关地址/api/app/getAppInfo
2. 通讯方式： POST
3. 签名算法：不需要
4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	否	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
示例：					

{"header":{"userCode":"USER00012020041519580108712 92","appCode":"app00012020041610 20152918451","tId":""},"mac":"","body":{}}

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：成功 -1：失败
2	响应信息	msg	String	是	
Body					
1	应用名	appName	String	是	
2	应用类型	appType	String	是	
3	应用密钥托管类型	caType	Int	是	1：托管 2：非托管
4	应用密钥类型	algorithmType	Int	是	1：SM2 2:ECDSA(secp256r1) 3:ECDSA(secp256k1)
5	该城市 MSPID	mspId	String	是	
6	应用链名称	channelId	String	是	Fabric 对应 channelId， fisco 对应 groupId

示例


```
{
  "header": {
    "code": 0,
    "msg": "处理成功"
  },
  "mac":
"MEUCIQDE9zv0E/w4V/ILG6wUCFP08a7NDCAtX/IoZOcCyY4gIQIgUTYWsFTA1KE88gE6
452jKnnVBrhznGVOV2HPMCbNh8A=",
  "body": {
    "appName": "sdk 测试",
    "appType": "fabric",
    "caType": 2,
    "algorithmType": 2,
    "mspId": "OrgbNodeMSP",
    "channelId": "app00012020041610201529184 51"
  }
}
```

5.4.6.4 注册用户接口

用户参与一个 CITA 的应用服务成功后，需要通过链下业务系统调用本接口生成进行链上交易处理的用户账户和账户地址。

1. 接口地址：

   https://节点网关地址/api/cita/v1/user/register

2. 通讯方式： POST

3. 签名算法：详见 5.4.6.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	用户名	userId	String	是	注册的用户名
示例： { "header":{"appCode":"app0001202010201541523083","userCode":"USER000120201090086090"}, "body": { "userId":"citatest" }, "mac":"MEQCIBRhaM2szckWl9N9qcqnaYXOXGQw7SfII9DlRvxcI3YVAiBt4XeNs+EUjhBNSr3IjLRPZucsuGHxfjt9RiaNIQS8cA=="} 签名值：					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					

1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	用户信息	data	[]string	否	code 不为 0 时为空
data					
1	用户编号	userId	String	是	
2	用户地址	userAddress	String	是	
示例					


```
{
  "header": {
    "code": 0,
    "msg": "处理成功"
  },
  "mac":
"MEQCIEI5VKMyJUXls2Hf8TLoPXjZLT4/L2wyXoddgTnZdqRsAiBxEBMeCOZ8M97OCRU
AMZNMcL974vhzjOS/tk8/wbgbsA==",
  "body": {
    "userId": "100003",
    "userAddress": "0x14647a48303b5e1c77934583883ebc327ba3b297"
  }
}
```


5.4.6.5 密钥托管模式下调用智能合约接口

链下业务系统在连接节点网关时，需要按照接口说明在请求中加入相应的请求参数，调用节点网关以后，节点网关会返回智能合约的执行结果。

1. 接口地址：

   https://节点网关地址/api/cita/v1/node/reqChainCode

2. 通讯方式： POST

3. 签名算法：详见 5.4.6.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	

header					
1	用户唯一标示	userCode	String	是	
2	应用唯一标识	appCode	String	是	
Body					
1	用户 ID	userId	String	是	用户通过 5.4.6.4 注册接口注册的用户的 id
2	合约名称	contractName	String	是	
3	方法名称	funcName	String	是	
4	方法参数	funcParam	string	否	将 array 格式的参数转为 json 字符串赋值
示例：					
{"header":{"appCode":"c1000620200318192657367757 2","userCode":"USER00062020031 81951281835816"},"body":{"contractName":"HelloWorld","userId":"100003","funcNa me":"set","funcParam":["abc"]},"mac":"MEUCIQDTFe2Gerdf7YJrG1a1Yt99MOZQ3T1lGpsX dNmFV7WuTgIgSkZ19abUhAJbMrJMBoD8N7f26xhpQRuR4vNAfY7EEbs="}					
签名值：					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 是可为 null
body					
1	交易 hash	txId	string	否	
2	交易状态	status	String	否	
3	查询数据	data	Object	否	合约查询方法时返回
示例					

5.4.6.6 获取交易回执接口

在智能合约执行一笔交易后，可以使用该接口根据交易的哈希值获取交易的回执信息。

1. 接口地址：

   https://节点网关地址/api/cita/v1/node/getTxReceiptByTxHash

2. 通讯方式： POST

3. 签名算法：详见 5.4.6.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	交易哈希	txHash	string	是	
示例：					
{"header":{"appCode":"c1000620200318192657367 7572","userCode":"USER00062020031 81951281835816"},"body":{"txHash":"0x755f3e7833778f674e1b025f513f05722ba7248be 43a3c9168b880847814021a"},"mac":"MEUCIQDTFe2Gerdf7YJrG1a1Yt99MOZQ3T1lGpsXdNmFV 7WuTgIgSkZ19abUhAJbMrJMBoD8N7f26xhpQRuR4vNAfY7EEbs="}					
签名值：					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	交易回执信息	transactionHash	string	否	code 不为 0 时为空
		transactionIndex	Int64		
	块哈希	blockHash	string		
	块号	blockNumber	long		
		cumulativeGasUsed	string		
		cumulativeQuotaUsed	long		

	使用 gas	gasUsed	string		
		quotaUsed	long		
		contractAddress	string		
		root	string		
		status	string		
		from	string		
		to	string		
		logsBloom	string		
		errorMessage	string		
		transactionIndexRaw	string		
		blockNumberRaw	string		
		cumulativeGasUsedRaw	string		
		cumulativeQuotaUsedRaw	string		
		gasUsedRaw	string		
		quotaUsedRaw	string		
		logs	[]log		
log					
		removed	bool		
		logIndex	long		
		transactionIndex	long		
		transactionHash	string		
		blockHash	string		
		blockNumber	long		
		address	string		
		data	string		
		transactionLogIndex	string		
		transactionIndexRaw	string		
		blockNumberRaw	string		
		logIndexRaw	string		
		topics	[]string		

示例


```
{
"code": 1,
"msg": "处理成功",
"data": {
"transactionHash":
"0x2072171b5233761a7eb25ecdb3e08dbcb14b7b9919ef538679a5adef1a04729f",
"transactionIndex": 0,
"blockHash":
"0xc178fc32ff61b624350c48e4c1c66099228340a7b808bfea7f17ed1d0172879c",
"blockNumber": 5443,
```

```
            "cumulativeGasUsed": null,
            "cumulativeQuotaUsed": 30639,
            "gasUsed": null,
            "quotaUsed": 30639,
            "contractAddress": null,
            "root": null,
            "status": null,
            "from": null,
            "to": null,
            "logs": [{
                "removed": false,
                "logIndex": 0,
                "transactionIndex": 0,
                "transactionHash":
"0x2072171b5233761a7eb25ecdb3e08dbcb14b7b9919ef538679a5adef1a04729f",
                "blockHash":
"0xc178fc32ff61b624350c48e4c1c66099228340a7b808bfea7f17ed1d0172879c",
                "blockNumber": 5443,
                "address": "0x728a4d500f9e2dcc01a7fa755476cf40f7cb5d1f",
                "data":
"0x00000000000000000000000000000000000000000000000000000000000000c8",
                "transactionLogIndex": "0x0",
                "topics":
["0xc6d8c0af6d21f291e7c359603aa97e0ed500f04db6e983b9fce75a91c6b8da6b"],
                "transactionIndexRaw": "0x0",
                "blockNumberRaw": "0x1543",
                "logIndexRaw": "0x0"
            }, {
                "removed": false,
                "logIndex": 1,
                "transactionIndex": 0,
                "transactionHash":
"0x2072171b5233761a7eb25ecdb3e08dbcb14b7b9919ef538679a5adef1a04729f",
                "blockHash":
"0xc178fc32ff61b624350c48e4c1c66099228340a7b808bfea7f17ed1d0172879c",
                "blockNumber": 5443,
                "address": "0x728a4d500f9e2dcc01a7fa755476cf40f7cb5d1f",
                "data":
"0x00000000000000000000000000000000000000000000000000000000000000c8",
                "transactionLogIndex": "0x1",
                "topics":
["0xfd28ec3ec2555238d8ad6f9faf3e4cd10e574ce7e7ef28b73caa53f9512f65b9",
"0x000000000000000000000000fe98e21021d9f3132103c7fd0509f8b7a32c286a"],
                "transactionIndexRaw": "0x0",
```

```
                "blockNumberRaw": "0x1543",
                "logIndexRaw": "0x1"
            }, {
                "removed": false,
                "logIndex": 2,
                "transactionIndex": 0,
                "transactionHash":
"0x2072171b5233761a7eb25ecdb3e08dbcb14b7b9919ef538679a5adef1a04729f",
                "blockHash":
"0xc178fc32ff61b624350c48e4c1c66099228340a7b808bfea7f17ed1d0172879c",
                "blockNumber": 5443,
                "address": "0x728a4d500f9e2dcc01a7fa755476cf40f7cb5d1f",
                "data": "0x",
                "transactionLogIndex": "0x2",
                "topics":
["0x59310b8be6fa298878e04229a6e8ad4e5dd6eda80b787363abf67948d720e50a",
"0x00000000000000000000000000000000000000000000000000000000000000c8",
"0x00000000000000000000000000000000000000000000000000000000000000c9",
"0x00000000000000000000000000000000000000000000000000000000000000ca"],
                "transactionIndexRaw": "0x0",
                "blockNumberRaw": "0x1543",
                "logIndexRaw": "0x2"
            }],
            "logsBloom":
"0x000000000000000000000000000000200000000000008000000000008020000000048000000
00000c000000000001000000100000000080000000000100000000000000000000000000000000
000000000000000000000000400000000000000000000000000000000000000000000000000800
040400000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000800400080000000000000000000000000000000000000020000000000
000000000000000000000000000000200000000000000000000000000080000000000002000000
0000000000000000000000000000000000000000800000",
            "errorMessage": null,
            "transactionIndexRaw": "0x0",
            "blockNumberRaw": "0x1543",
            "cumulativeGasUsedRaw": null,
            "cumulativeQuotaUsedRaw": "0x77af",
            "gasUsedRaw": null,
            "quotaUsedRaw": "0x77af"
        }
        }
```

5.4.6.7 获取交易信息接口

智能合约执行一笔交易后，可以使用该接口根据交易的哈希获取交易的详细信息。

1. 接口地址：

   https://节点网关地址/api/cita/v1/node/getTxInfoByTxHash

2. 通讯方式： POST

3. 签名算法：详见 5.4.6.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	交易哈希	txHash	string	是	
示例：					
{"header":{"appCode":"cl000620200318192657367 7572","userCode":"USER00062020031 81951281835816"},"body":{"txHash":"0x755f3e7833778f674e1b025f513f05722ba7248be43a3c 9168b880847814021a"},"mac":"MEUCIQDTFe2Gerdf7YJrG1a1Yt99M0ZQ3T1lGpsXdNmFV7 WuTgIgSkZ19abUhAJbMrJMBoD8N7f26xhpQRuR4vNAfY7EEbs="}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	交易哈希	txHash	String		
2	块哈希	blockHash	String		
3	块号	blockNumber	Int		

		data	String		
		chainId	String		
		quota	String		
		from	String		
		to	String		
		nonce	String		
		validUntilBlock	String		
		version	String		

示例


```
{
    "code": 1,
    "msg": "处理成功",
    "body": {
        "txHash":
"0xdfa70810e5706eb34fd8a75a7441850c440a162af50924f6708e4ffac929d22d",
        "blockHash":
"0x55d97ebb079d19a74e6215872e733891d0c3ad47820e3b9c7a91b0df6746af69",
        "blockNumber": 787,
        "data":
"0x60fe47b1000000000000000000000000000000000000000000000000000000000000000200",
        "chainId": "1",
        "quota": "30639",
        "from": "0xfe98e21021d9f3132103c7fd0509f8b7a32c286a",
        "to": "0x728a4d500f9e2dcc01a7fa755476cf40f7cb5d1f",
        "nonce": "5882372291862313075",
        "validUntilBlock": "865",
        "version": "2"
    }
}
```


5.4.6.8 获取块信息接口

当数据落链并生成区块链块后，可以根据块号或者块的哈希查询响应的块的信息，其中块号和块哈希不能同时为空。当同时不为空时，优先查询块号对应的块信息。

1. 接口地址：

   https://节点网关地址/api/cita/v1/node/getBlockInfo

2. 通讯方式： POST

3. 签名算法：详见 5.4.6.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	块高	blockNumber	string	否	为空时 blockHash 不能为空
2	区块 hash	blockHash	String	否	为空时 blockNumber 不能为空
示例：					
{ "header":{"appCode":"CL188103887322019090211431 4","userCode":"newuser"}, "body": { "blockNumber":22, "blockHash":"0xf27ff42d4be65329a1e7b11365e190086d92f9836168d0379e92642786db7ade" }, "mac":"MEQCIBRhaM2szckWl9N9qcqnaYXOXGQw7SfII9DlRvxcI3YVAiBt4XeNs+EUjhBNSr3IjLRPZucsuGHxfjt9RiaNIQS8cA==" }					
签名值：					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	块哈希	blockHash	String	是	
2	块号	blockNumber	Int64	是	

3	上一个块哈希	prevBlockHash	String	是	
4	落块时间	blockTime	String	是	毫秒格式时间戳
5	执行区块交易花费的费用	quotaUsed	String	是	
6	交易根哈希	transactionsRoot	String	是	
7	状态树根哈希	stateRoot	String	是	
8	回执根哈希	receiptsRoot	String	是	
9	交易信息	transactions	[]TransactionData	是	
TransactionData					
1	交易哈希	txHash	String		
		data	String		
		chainId	String		
		quota	String		
		from	String		
		to	String		
		nonce	String		
		validUntilBlock	String		
		version	String		

示例


```
{
"code": 1,

"msg": "处理成功",

"data": {
        "blockHash":
"0xecbc08817c9d8683c62831bf1d261b28682671fb423d32b6fbc7a99f9334eda6",
        "receiptsRoot":
"0x2b1ef9c90c51279ea4deae9e9aa1bfdd3a72e9aa6f0a207aea931c97ae8849e9",
        "proposer": "0x088c277a776d5073c0804d4220eb5a1018d9ad9b",
        "quotaUsed": "33391",
        "transactionsRoot":
"0xd89c96752518074a4d8582bc267f48af589a8b9bb57d4c7c6fed46053823eaa6",
        "blockNumber": 57877,
        "prevBlockHash":
"0xf85656b21e1e8ba0417cb7f4237407c81591cd2562a4c96183786373bf440ffa",
        "stateRoot":
"0x7b7b8f7081f07413f820261b00d4d1dc51276af8bbb673af9a17eb5deeade64a",
        "blockTime": "2020-09-25 16:04:51 284",
        "transactions": [
            {
```

```
                "data":
"0x4ed3885e0000000000000000000000000000000000000000000000000000000000000020000
0000000000000000000000000000000000000000000000000000000033233340000000000000
0000000000000000000000000000000000000000000000000000",
                "chainId": "1",
                "quota": "33391",
                "from": "0x3c8269e9f92de2949497813b060c399d6c40025e",
                "to": "0xc8bac50c7fc7b2d85c51e2ee9dbd779bf24c10c8",
                "nonce": "4214650517532747442",
                "txHash":
"0xd89c96752518074a4d8582bc267f48af589a8b9bb57d4c7c6fed46053823eaa6",
                "validUntilBlock": "57955",
                "version": "2"
            }
            ]
        }
    }
```


5.4.6.9 获取应用内的块高接口

根据该接口可以获取应用的块高度。

1. 接口地址：

   https://节点网关地址/api/cita/v1/node/getBlockHeight

2. 通讯方式： POST

3. 签名算法：详见 5.4.6.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
示例：					
{"header":{"appCode":"cl000620200318192657367757 2","userCode":"USER00062020031 81951281835816"},"body":{},"mac":"MEQCIBRhaM2szckWl9N9qcqnaYXOXGQw7SfII9DlR vxcI3YVAiBt4XeNs+EUjhBNSr3IjLRPZucsuGHxfjt9RiaNIQS8cA=="}					

签名值：

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	块高	data	string	否	code 不为 0 时为空

示例


```
{
  "header": {
    "code": 0,
    "msg": "处理成功"
  },
  "mac":
"MEQCICtCOdv4ZL72M3WoA9nAei2P0/PpKjlgI0Y5qeuzg61uAiA9D3TcB/+b2RMuNwVq+X
0vgiglHfM5NBhoTJPR0gCPMA==",
  "body": {
    "data": "4"
  }
}
```

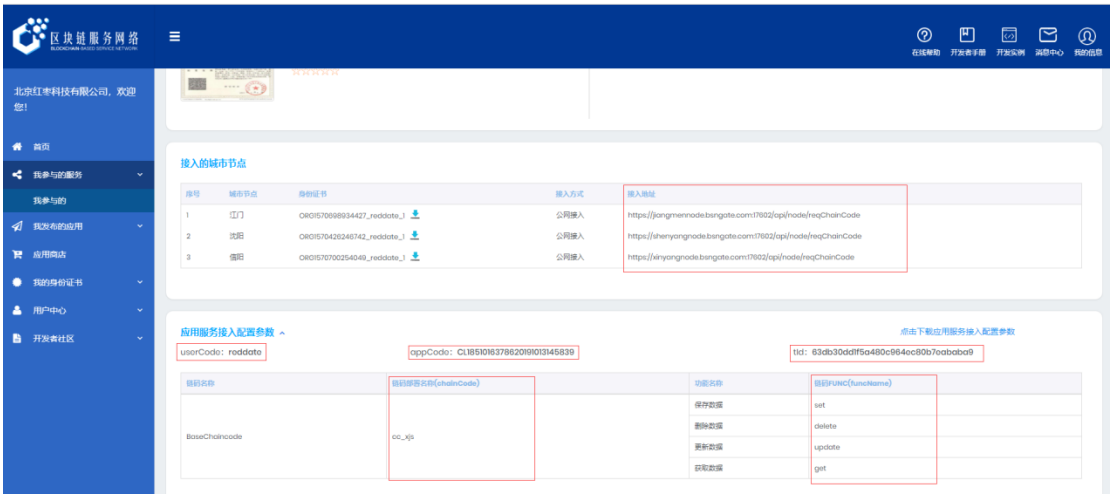

5.4.6.10 密钥上传模式下调用智能合约接口

链下业务系统在连接节点网关时，需要按照接口说明在请求中加入相应的请求参数，调用节点网关以后，节点网关会返回智能合约的执行结果。在上传公钥模式的交易中，链上交易的私钥由用户自己生成和保存，然后由客户端在本地进行上链数据的组装和签名，将签名后的数据上传至节点网关，网关将数据转发至相应的区块链节点发起交易请求。在该模式中组装数据时需要用到合约的 ABI 以及合约地址等信息，其中合约 ABI 在开发合约时编译合约得到，合约地址可以在应用的详情页面获取。在网关的 SDK 中已经实现了该上链数据的组装方法，直接调用即可。

1. 接口地址：

https://节点网关地址/api/cita/v1/node/trans

注：用户参与服务成功后可以在服务详情页面查看并下载应用链下业务系统开发所需要的应用服务配置参数、节点网关地址和应用接入密钥对，如下图：



2. 通讯方式： POST

3. 签名算法：详见 5.4.6.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
Body					
1	合约名称	contractName	String	是	
2	交易数据	transData	String	是	
示例：					
{"header":{"userCode":"USER000120200604232157969244 0","appCode":"app00012020060423 23057101002","tId":""},"mac":"MEUCIQCrjIeRVSt1uwFFGkr37bVM8pF0JgAWb40mKEBc5 HbpjgIgEzXRIgG+Q7obwuD2MY4EHo9sIsI1W71M+aQKOfAN3wU=","body":{"contractNam e":"BsnBaseContractk1","transData":"0xf9016fa008d8ebcb4b1f8205fd7883aa3ce9b9c84442407 0e55a3af6a5da5d7ee97d287385051f4d5c0083419ce07794866aefc204b8f8fdc3e45b908fd43d766 67d7f7680b8e4ebf3b24f0000000000000000000000000000000000000000000000000000000000000000					

```
000060000000000000000000000000000000000000000000000000000000000000000500000000
000000000000000000000000000000000000000000000000000000a00000000000000000000000
0000000000000000000000000000000000000000057330363034000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000026161000000000000000000000000000000000000000000000000000000000000
018187801ba0324efc9e17f1d31d95535c5103083560560f836931945d03fb69acb6fd2046b5a05fa
6f574f83b3b753c9fe40649a08c2a497af1cd804e08fed2b153af20267f23"}}
```

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	code=0 时可为 null
body					
1	交易 hash	txId	string	否	
2	交易状态	status	String	否	
3	查询数据	data	Object	否	合约查询方法时返回
示例					

5.4.6.11 链码事件注册接口

可以通过该接口注册 cita 的出块事件或者合约事件，当事件被触发时，系统将向注册的通知地址发送事件内容。

1. 接口地址：

https://节点网关地址/api/cita/v1/event/register

2. 通讯方式： POST

3. 签名算法：详见 5.4.6.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	

header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	事件类型	eventType	String	是	1：出块事件 2：合约事件
2	合约地址	contractAddress	String	否	EventType 为 1 时可以为空，为 2 时，不可与 contractName 同时为空
3	合约名	contractName	String	否	EventType 为 1 时可以为空，为 2 时，不可与 contractAddress 同时为空
4	通知地址	notifyUrl	String	是	
5	附件参数	attachArgs	String	否	
示例：					
{"header":{"userCode":"USER000120200604232157969244 0","appCode":"app0001202006042323057101002","tId":""},"mac":"MEUCIQCMP1ToZS5e8S94kYZ/8y5XfeyjRyUrPFpeIQMES3SGpQIgO8b6O8Kk/qpNTo1vbNTwyAYNaw6HBi9OkAH8Rp23j8s=","body":{"eventType":1,"contractAddress":"0x866aefc204b8f8fdc3e45b908fd43d76667d7f76","contractName":"BsnBaseContractk1","notifyUrl":"http://127.0.0.1:18080","attachArgs":"abc=123"}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	
body					
1	事件 Id	eventId	string	是	code 不为 0 时为空
data					
示例					
{ "header": { "code": 0, "msg": "处理成功" }, "mac": "MEUCIQDYSTwYhh6EDHT5Z7ukcqXW9LMjZW6WPnrv8Xt14RuH2AIgIwa5K7NK4/TThzs8z6VfkpNNJU+dzAXeypFmfjkru88=",					

```
  "body": {
    "eventId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  }
}
```

5.4.6.12 链码事件查询接口

该接口可以查询已经注册成功的链码事件的列表。

1. 接口地址：

   https://节点网关地址/api/cita/v1/event/query

2. 通讯方式： POST

3. 签名算法：详见 5.4.6.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
示例：					
{"header":{"userCode":"USER000120200604232157969244 0","appCode":"app000120200604 2323057101002","tId":""},"mac":"MEUCIQC2NTuUlsxQSWPpZwwhJK9zXEMaeYZC04Ar0P5T wyp5AQIgFvZrskasuLiYfOGxd1F9TCetWHIfENg8BCiYfNS1xGk=" }					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	
body					

1	出块事件	blockEvent	[]blockEvent	是	code 不为 0 时为空
2	合约事件	contractEvent	[]contractEvent	是	
blockEvent					
1	出块事件	eventId	string	是	code 不为 0 时为空
2	应用编号	appcode	String	是	
3	用户编号	userCode	String	是	
4	通知地址	notifyUrl	String	是	
5	附件参数	attachArgs	String	否	
6	创建时间	createTime	String	是	UTC 时间
contractEvent					
1	出块事件	eventId	string	是	
2	应用编号	appcode	String	是	
3	用户编号	userCode	String	是	
4	通知地址	notifyUrl	String	是	
5	附件参数	attachArgs	String	否	
6	创建时间	createTime	String	是	UTC 时间
7	合约地址	contractAddress	String	是	

示例


```
{
   "header": {
      "code": 0,
      "msg": "处理成功"
   },
   "mac":
"MEUCIQCQ/RjmlVkLKZw6jcLKBPh1BwK4EIQE001vUAKPVq1HTgIgXUQ7Bn+y8D8xQxYU
wtZOoh/bpteAPCUtKXZeAiN7cMU=",
   "body": {
      "blockEvent": [
         {
            "eventId": "ba537419953e4e219ceb0fe26ad5e125",
            "appCode": "app000120200604232305710100 2",
            "userCode": "USER000120200604232157969244 0",
            "notifyUrl": "http://127.0.0.1:18080",
            "attachArgs": "abc=123",
            "createTime": "0001-01-01 00:00:00.000 +0000 UTC"
         }
      ],
      "contractEvent": [
   {
            "eventId": "ba537419953e4e219ceb0fe26ad5e126",
            "appCode": "app000120200604232305710100 2",
            "userCode": "USER000120200604232157969244 0",
            "notifyUrl": "http://127.0.0.1:18080",
```

```
            "attachArgs": "abc=123",
            "createTime": "0001-01-01 00:00:00.000 +0000 UTC",
            “contractAddress”:” 0x866aefc204b8f8fdc3e45b908fd43d76667d7f76”
          }

      ]
  }
      }
```


5.4.6.13 链码事件取消接口

该接口可以取消已经注册成功的链码事件。

1. 接口地址：

   https://节点网关地址/api/cita/v1/event/remove

2. 通讯方式： POST

3. 签名算法：详见 5.4.6.1 应用接入签名算法

4. 请求参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	
3	签名值	mac	String	是	
header					
1	用户唯一标识	userCode	String	是	
2	应用唯一标识	appCode	String	是	
body					
1	事件编号	eventId	string	是	
示例：					
{"header":{"userCode":"USER000120200604232157969244 0","appCode":"app0001202006042323057101002","tId":""},"mac":"MEQCIARwCd9VEkIP6ISniKPnJRjyPJ/nqwf7kZw0rBmzSAxlAiATW7zcRM+yQ+JJTKxXV6L/S7zUT1r6Hctlb0ccAVTc7Q==","body":{"eventId":"ba537419953e4e219ceb0fe26ad5e125"}}					

5. 响应参数

序号	字段名	字段	类型	必填	备注
1	信息头	header	Map	是	
2	信息体	body	Map	是	

3	签名值	mac	String	是	
header					
1	响应标识	code	int	是	0：校验成功 -1：校验失败
2	响应信息	msg	String	否	
示例					
{ "header": { "code": 0, "msg": "处理成功" }, "mac": "MEUCIQCjkDCMi1S38CbKFmd/zN6HpacIi798S3nE/bqO8gU7ggIgCGpX3DQk1Ulf4sevbKJ6S2cDopHFJB1bAqDOvOarGFk=", "body": null }					

5.5 开发 SDK 及示例

1. BSN 网关 SDK 实例

我们为开发者提供链下业务系统调用 BSN 城市节点网关进行交易处理和查询的基础 SDK，帮助开发者可以快速完成链下业务系统与网关交互的开发。SDK 中对网关 API 进行了封装，提供交易封装、网关交易接口调用、本地生成公私钥对、上传公钥模式下用户注册和证书登记、数据签名和数据加解密等功能。

下载地址：

https://github.com/BSNDA/PCNGateway-Go-SDK

https://github.com/BSNDA/PCNGateway-Java-SDK

https://github.com/BSNDA/PCNGateway-PY-SDK

https://github.com/BSNDA/PCNGateway-CSharp-SDK

2. 示例应用

如下是我们基于预制链码包开发的链下业务系统通过网关 API 调用链码的示例源代码，包括：golang、java、C#、python 语言的示例，供您下载参考。

- Fabric 示例

下载地址：

https://github.com/BSNDA/FabricBaseChaincode

- FISCO BCOS 示例

下载地址：

https://github.com/BSNDA/FISCOBaseContract

- XuperChain 示例

下载地址：

https://github.com/BSNDA/XuperChainBaseContract

- CITA 示例

下载地址：

https://github.com/BSNDA/CITABaseContract

我们邀请对 BSN 感兴趣，并且有经验的开发者，来共同优化上述的 SDK 和示例包。如果您愿意参加，请在 GitHub 里联系我们。

5.6 测试网服务

5.6.1 概述

测试网是 BSN 为开发者免费提供的用于进行区块链应用服务测试的运行环境。开发者可以在测试网发布不限定数量的应用服务，与生产环境不同的是，发布应用服务时不需要选择智能合约部署的城市节点和配置智能合约的调用权限。测试网支持 Hyperledger Fabric、

FISCO BCOS、XuperChain 三种联盟链框架，后续将持续集成所有 BSN 适配的联盟链框架。测试网还集成了 IPFS 专网、跨链服务和预言机服务。我们将根据测试网的资源负载使用情况不定期的重置，进行应用服务智能合约、账本数据、IPFS 存储文件的清理。请不要将测试网用于生产。欢迎大家试用测试网服务并反馈问题和建议，我们将持续进行功能完善。

5.6.2 应用服务发布

应用测试服务的发布步骤如下：

- 创建新的测试服务
  - 点击【联盟链服务】->【测试网服务】页面发布服务；

  - 点击【新增】，进入创建新测试服务页面，根据提示输入相应的信息。

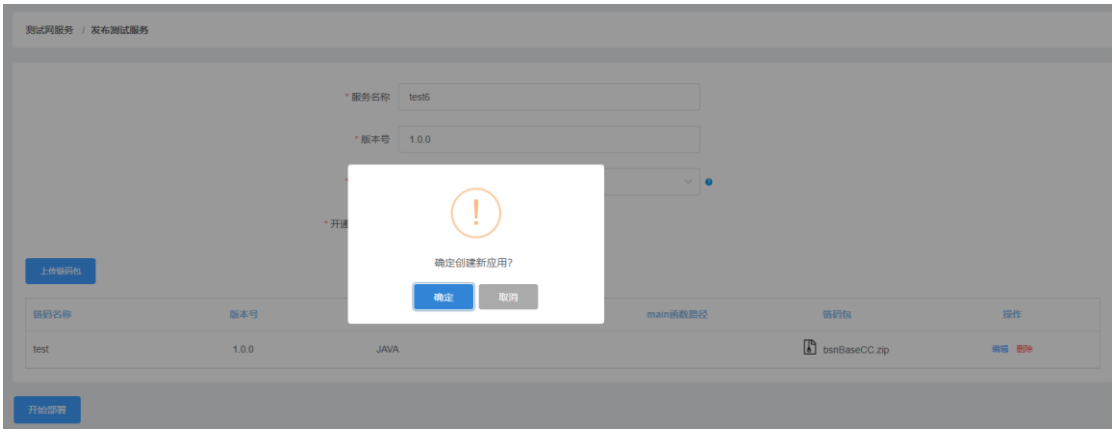
- 上传链码包

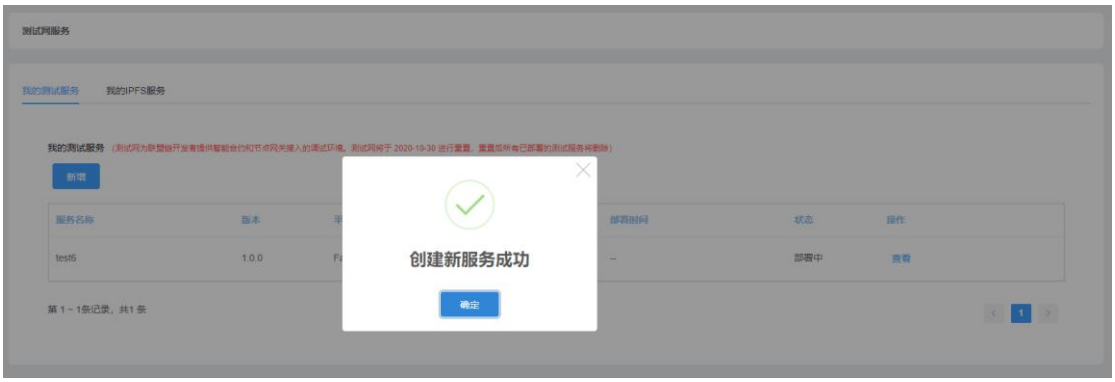
- 一个服务可以上传多个链码包，用户可在上图中点击【上传链码包】开始上传链码包，如下图：

- 根据页面进行输入，并点击【确定】来完成链码包的上传

➢ 服务部署

- 点击【开始部署】，确定创建新应用，进行服务部署





- 链码部署成功之后，开发者可以在本地进行相关的功能调试。
- **注意：为保证测试节点的资源稳定性，运维人员将定期清理测试节点上的测试链码包和测试数据。**

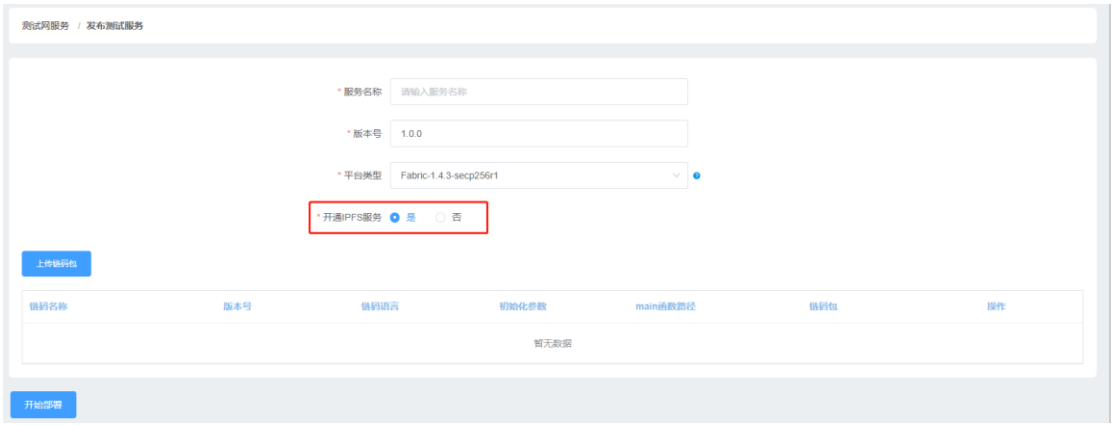
5.6.3 IPFS 服务

BSN 测试网 IPFS 专网面向应用开发提供 IPFS 常用的原生服务接口的服务网关，开发者可以无缝的与 IPFS 公网切换。为了让更多的开发者体验 IPFS 服务，我们对上传文件的大小限制为 50M，并将根据存储资源的使用情况进行不定期重置清理，请不要将测试网用于生产环境。

有两种方式可以开通 IPFS 服务：在发布测试网服务时开通； 在【我的 IPFS 服务】界面单独开通。服务开通后，系统会自动生成上载 key 和下载 key。

➢ 方式一：在发布测试服务时开通。

用户只需要在发布测试服务时在开通 IPFS 服务选项处选择“是即可开通。



➢ 方式二：在【我的 IPFS 服务】界面单独开通。

点击【测试网服务】->【我的 IPFS 服务】，选择【创建 IPFS 服务】即可开通。

✧ 注意：如果用户只需要下载文件，则可不开通 IPFS 服务，只点击上图中的【创建 IPFS 下载 key】生成下载 key 即可。

IPFS 服务文件上传，下载，更新等所有的操作都是通过接口来进行，详细的接口介绍，请访问 8.4《IPFS 服务网关接口》。

5.6.4 跨链服务

跨链通信枢纽在测试网中集成了分布科技的 Poly Enterprise 和边界智能的 IRITA 两种基于中继链机制的跨链解决方案。欢迎大家在测试网进行体验试用并反馈问题和开发建议，我们将持续完善功能。

详细的跨链介绍及示例，请访问第十章《跨链服务》。

5.6.5 预言机服务

目前 BSN 预言机已在测试网上线，集成了微众基于联盟链的 Truora 解决方案，支持获取随机数和获取汇率两种数据的 API，用户也可自行开发链下 API 作为数据源。欢迎大家在测试网进行体验试用并反馈问题和开发建议，我们也将持续完善功能并集成更多的预言机解决方案。

详细的预言机服务介绍及示例，请访问第十一章《预言机服务》。

6 BSN 专有节点服务

6.1 概述

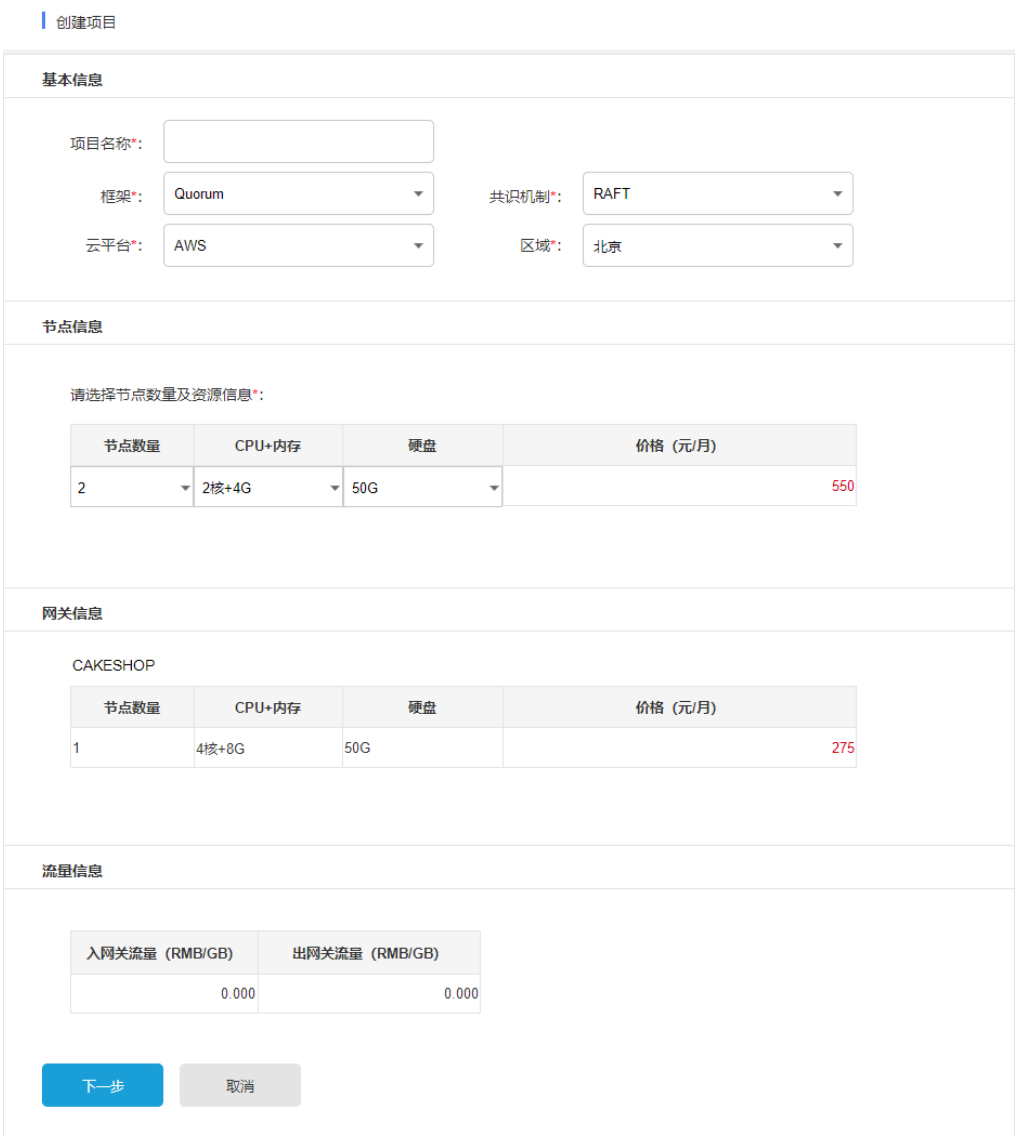
BSN 专有节点服务应用了 BSN 的多底层框架适配、虚拟化容器、自动化部署和节点网关等技术，为用户提供“开箱即用”的区块链专享云服务。用户可以在 BSN 门户快速创建自己专享的联盟链运行环境，配置节点 CPU、内存、磁盘容量等参数；可以自主的进行节点管理、发布智能合约、访问节点数据、监控区块链运行状态等。专有节点不对底层框架的 API 做限制，开发者通过网关接入专有节点后，所有的接口开发者都可调用。

当前版本的专有节点服务支持用户基于 AWS 云平台上的 BSN 城市节点搭建 ConsenSys Quorum（是一套开源、免费且聚焦企业级应用的区块链平台框架）联盟链环境，版本号: v20.10.0。共识机制支持 Raft 和 IBFT 机制。

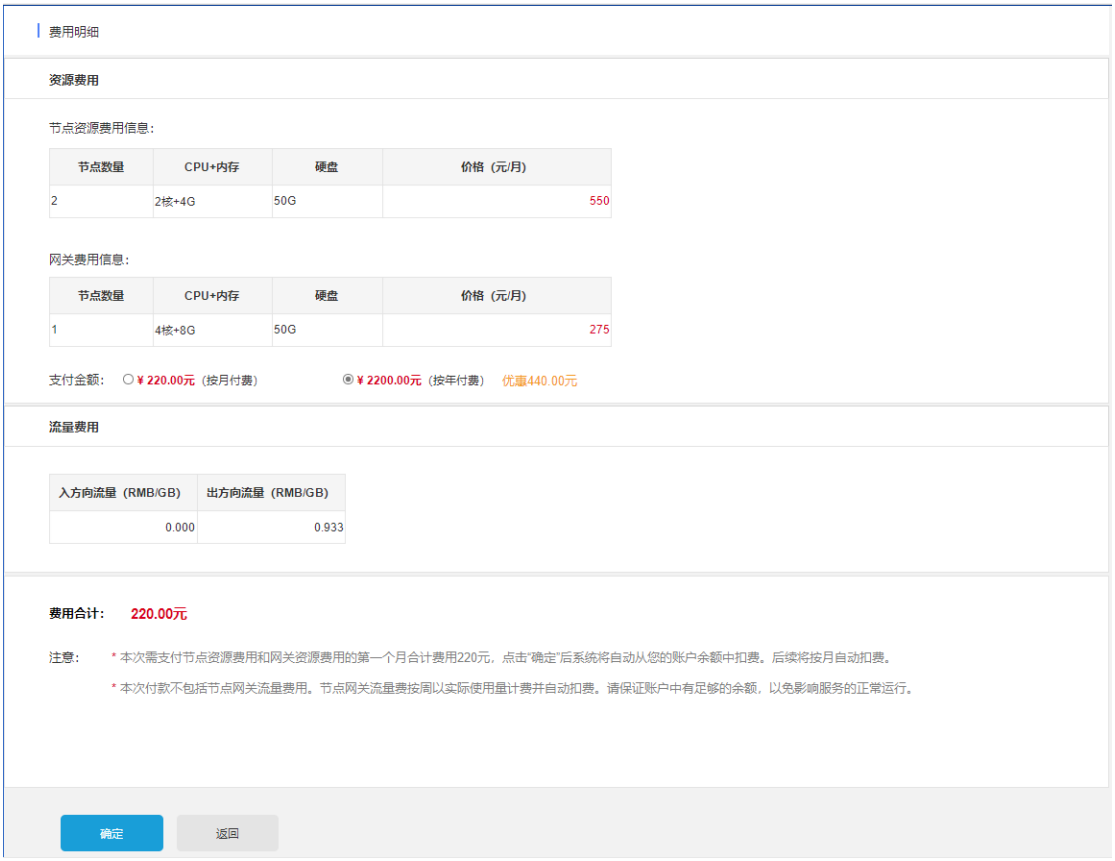
6.2 项目创建

- 点击【联盟链服务】->【专有节点服务】页面创建项目；

- 点击“创建项目”，进入到【创建项目】页面，根据提示输入或选择相应的信息；网关和流量信息在此处展示了使用的资源配置及计费价格，不需要再选择。



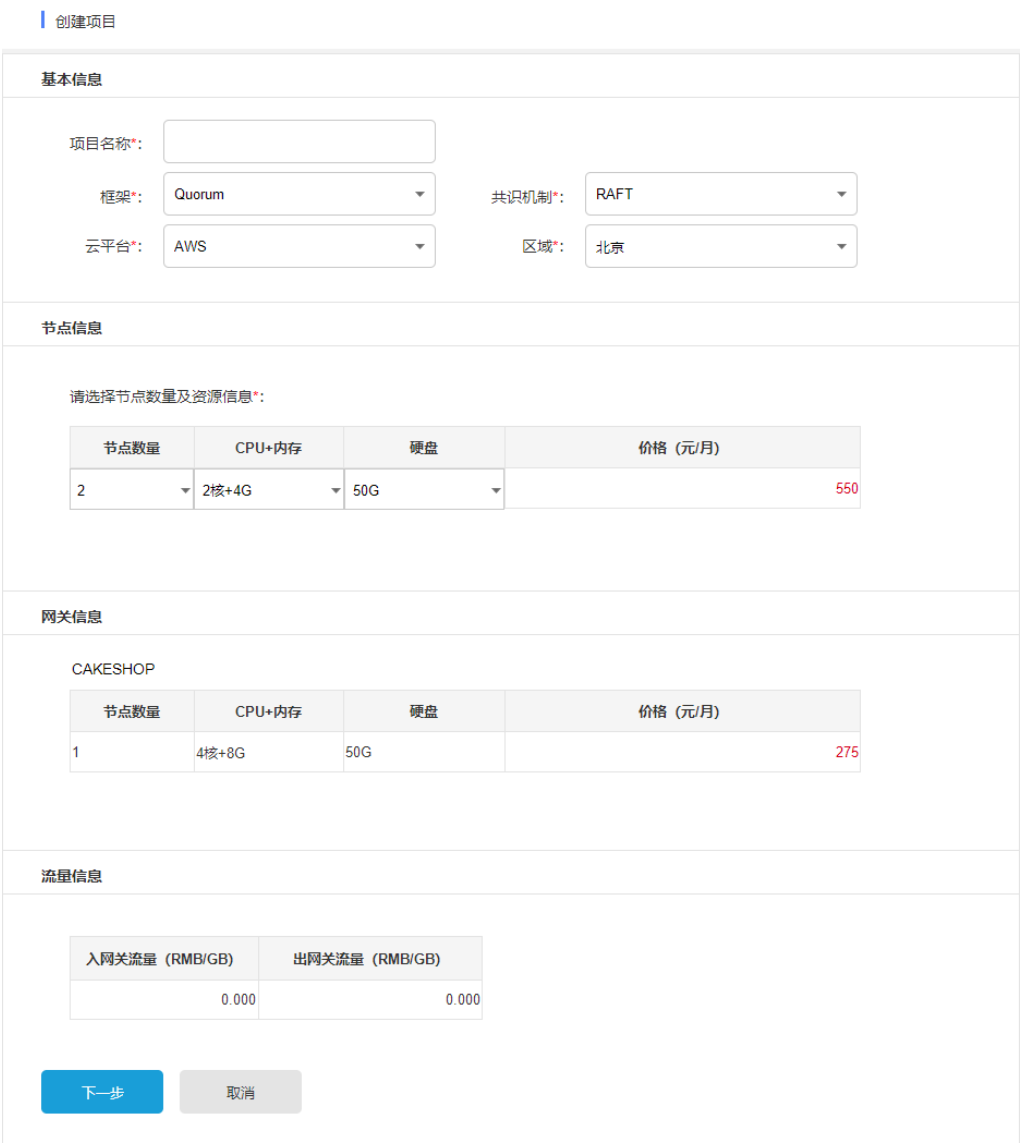
➢ 点击“下一步”进入到【费用明细】页面：



- 创建者确认费用明细后，点击“确定”按钮进行支付，支付时将从用户个人（或企业）账户中扣款，扣款成功后，等系统自动进行专有节点的部署。若扣款失败，则账单将保留 72 小时，过期失效，若仍想开通专有节点服务，可通过编辑项目重新提交或重新创建项目。

6.3 项目编辑

- 对于状态为“未部署”且账单失效 和“部署失败”且已全额退款的专有节点服务可进行编辑，在编辑界面，可以重新修改基本信息和节点信息。



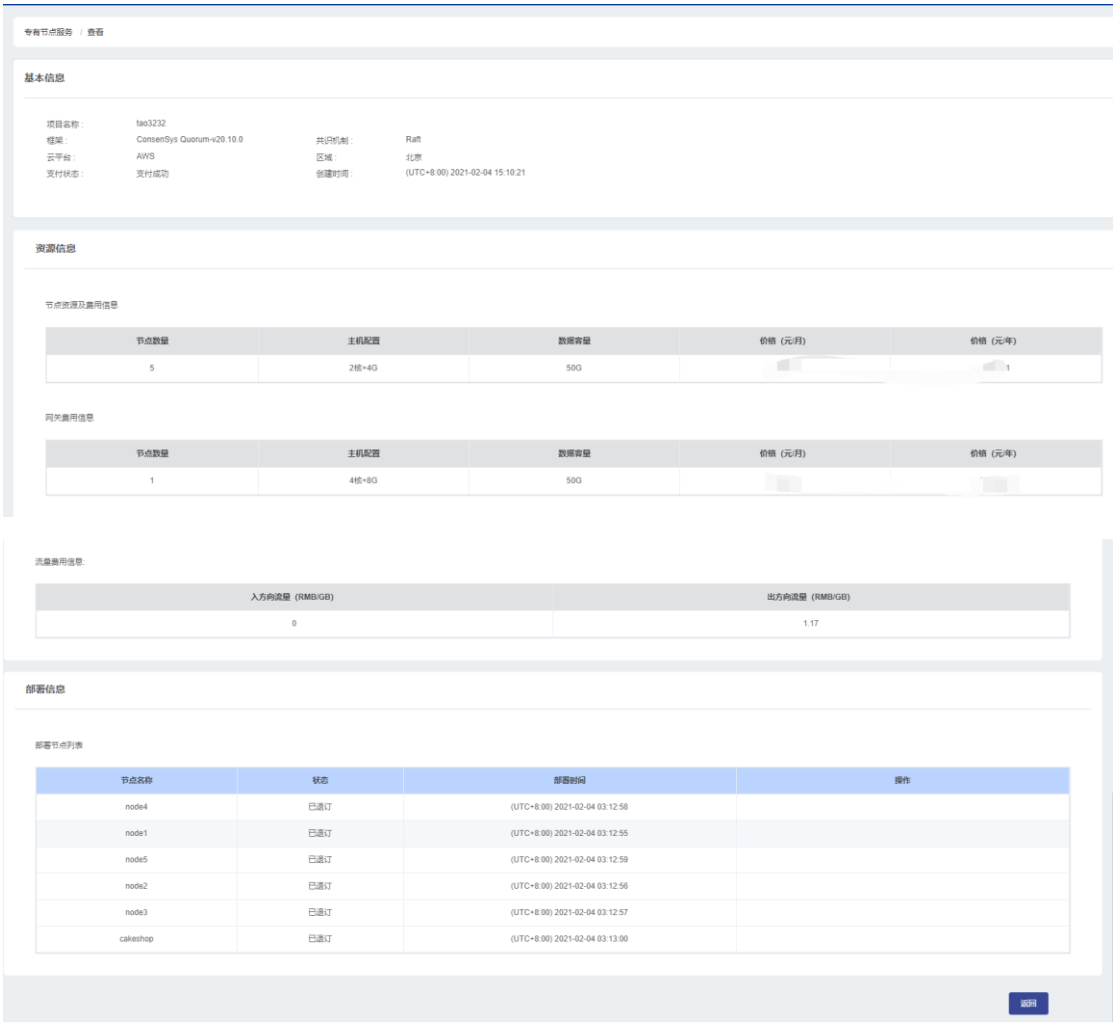
- 修改后进入确认【费用明细】页面，支付成功后等待专有节点的部署。

6.4 项目删除

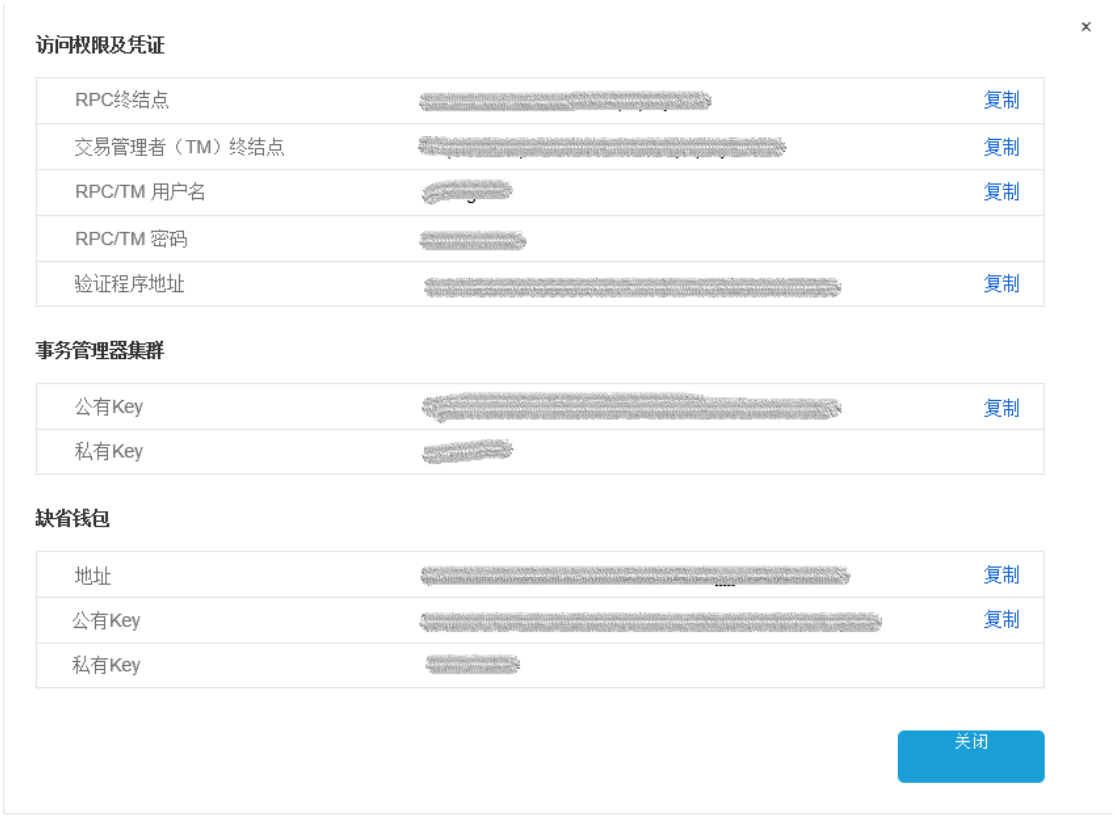
- “未部署”且账单已失效和“部署失败”已全额退款的专有节点服务可删除。

6.5 项目查看

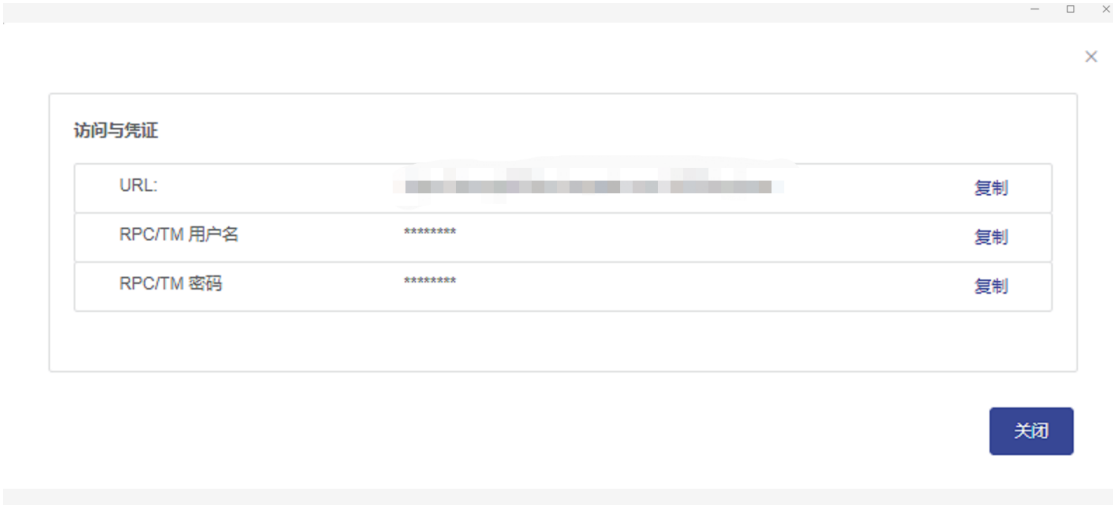
- 在专有节点服务列表中，点击“查看”，可查看专有节点的【基本信息】和【部署信息】。



- 在【部署信息】部分，可以查看节点信息和浏览器信息。点击节点信息对应的“查看”，显示专有节点的访问信息、权限凭证、事务管理器集群等信息：



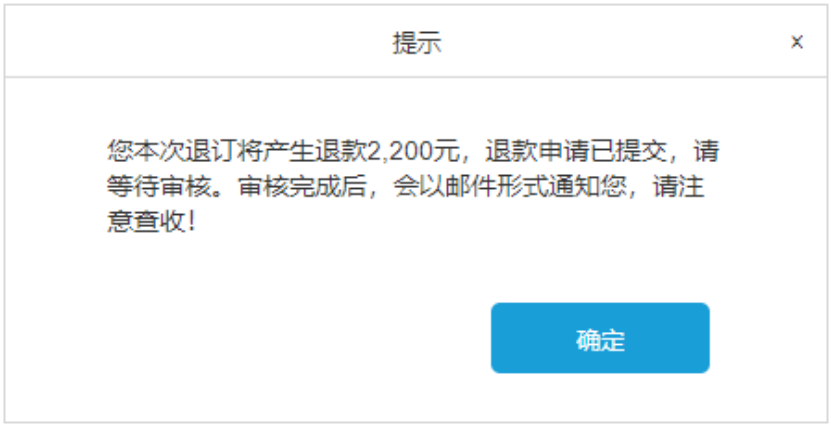
- 点击浏览器的“查看”，显示区块链浏览器的访问地址：



6.6 项目退订

- 对于运行中的专有节点服务，用户在专有节点服务列表中可以选择退订：

- 对于节点和网关资源按月支付的用户，退订时不产生退款；对于节点和网关资源按年支付的用户，退订时按下一个月到计费周期截止的时间点退款，退款时可退月份的费用将取消按年的优惠政策，按实际剩余月份统计退款。



7 开放联盟链服务

7.1 概述

开放联盟链是用于部署和运行各类区块链应用的一站式区块链服务运行环境。与传统联盟链服务相比，开放联盟链的应用共享记账节点资源，不同应用的智能合约可以相互可见及调用，共享使用区块链数据账本；链外业务系统可以通过节点网关简单、快速接入区块链网络进行交易处理。

目前 BSN 开放联盟链已推出文昌链（基于 COSMOS/IRISnet SDK）和泰安链（基于 FISCO BCOS），BSN 后续将持续集成更多的底层框架，并在 2021 年全年提供给开发者免费使用，欢迎大家体验并提出宝贵的意见和建议。

7.2 开放联盟链操作指南

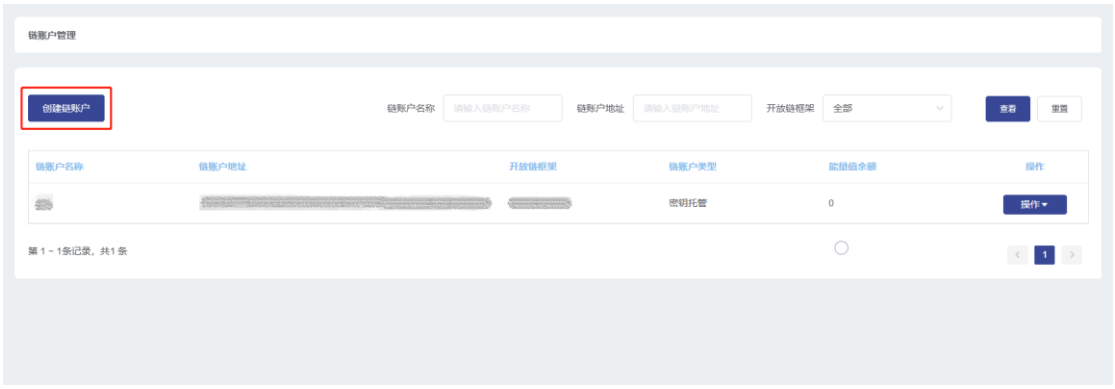
7.2.1 链账户管理

在访问链上前，用户需进行链账户的创建。在链上交易处理过程中，会根据交易处理消耗的算力和存储资源从链账户中扣除相应的能量值。开放联盟链在 2021 年免费开放使用，我们会为每个链账户赠送一定数量的能量值，用户可免费进行使用。

7.2.1.1 创建链账户

用户在此模块中进行链账户的创建，具体步骤如下：

- 进入【链账户管理】模块，点击【创建链账户】按钮；

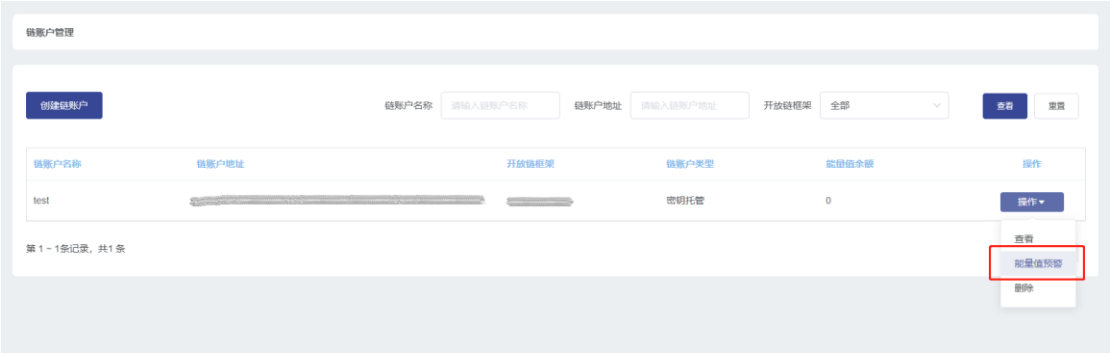


- 在链账户模块中点击创建链账户进入新增页面。页面中输入链账户名称并选择开放链框架，请注意：完成创建后，该链账户只可用于访问所选开放链框架且不可修改。最后，用户需选择该链账户的类型，平台提供包括密钥托管模式、发布公钥模式以及上传链账户地址模式三种类型：
  - **密钥托管模式**：由系统负责生成链账户的密钥，用户可以进行下载使用。
  - **发布公钥模式**：由用户线下自行生成密钥，并将公钥上传至系统。
  - **上传链账户地址**：如果用户已有链账户，则系统支持用户自行将链账户地址上传。请注意，系统不允许重复上传链账户地址。

7.2.1.2 管理链账户

用户可以在此模块中对已完成创建的链账户进行管理，包括查询、删除等。

- 查询链账户
  - 输入【链账户名称】、【链账户地址】并选择【开放链框架】后，点击【查看】按钮，系统在当前列表中筛选符合输入条件的链账户。
  - 点击【重置】按钮后，输入内容清空。
- 能量值预警



  - 在链账户列表中的操作列点击【能量值预警】后，系统打开能量值预警窗口。

在【能量值小于等于】的输入框中输入预警值，点击确定后，该链账户在后续的调用中，如果能量值小于预警值，则会给当前用户的邮箱、手机号中发送提示信息。

- 删除链账户
  - 在链账户列表中的操作列点击【删除】后，弹窗确定是否删除，点击【确定】则系统删除该链账户，点击【取消】则不进行删除操作。

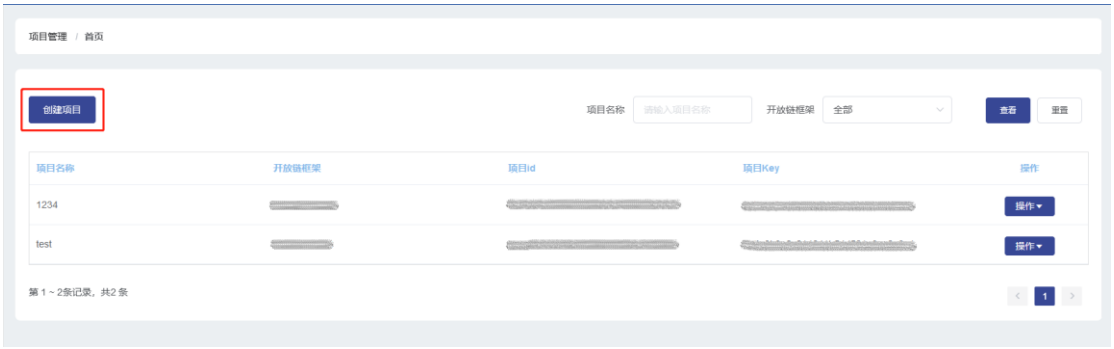
7.2.2 项目管理

开放联盟链通过项目来管理智能合约，用户将合约通过项目进行上传并部署，完成部署的合约即可进行链上调用。

7.2.2.1 创建项目

用户在此模块中进行项目的创建，具体步骤如下：

- 进入【项目管理】模块，点击【创建项目】按钮；

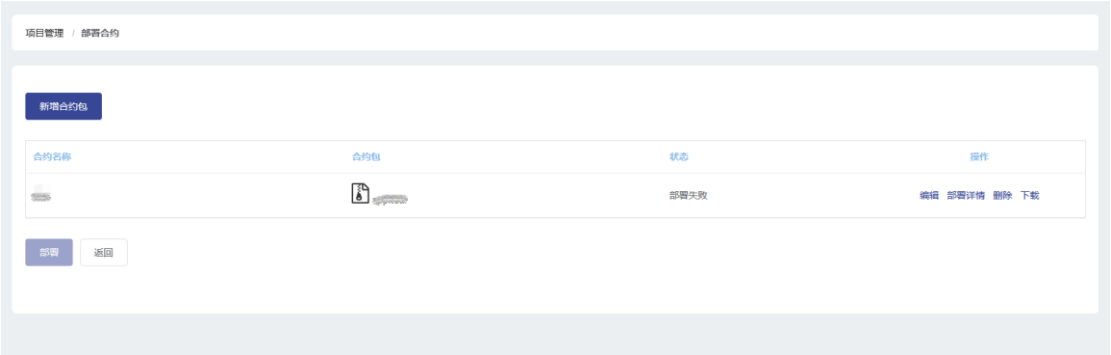


- 在创建项目页面中输入【项目名称】，选择【开放链框架】即可完成创建。请注意，项目创建完成后不允许修改开放链框架。

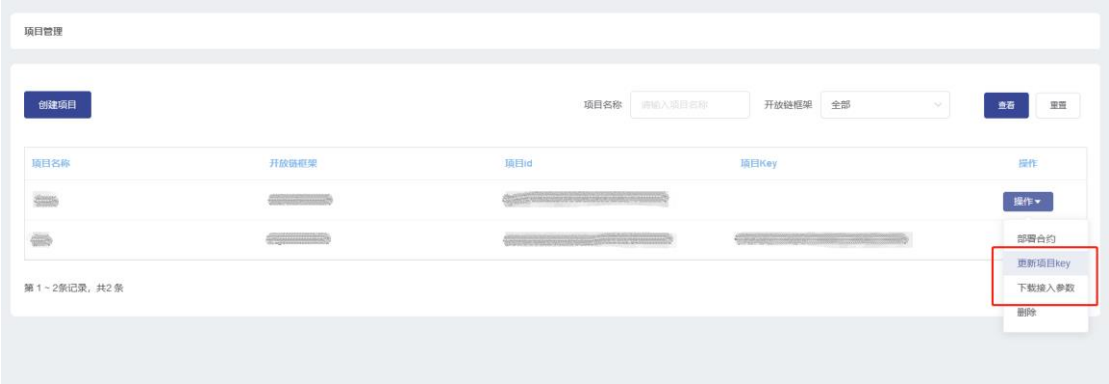
7.2.2.2 管理项目

- 查询项目
  - 输入【项目名称】并选择【开放链框架】后，点击【查看】按钮，系统在当前列表中筛选符合输入条件的项目。
  - 点击【重置】按钮后，输入内容清空。

- 部署合约



  - 在项目列表的操作列点击【部署合约】按钮进入合约管理，用户可针对该项目进行合约的新增、编辑、删除、下载等操作
  - 新增合约包：点击【新增合约包】按钮后打开弹窗进入新增页面，页面中需输入【合约名称】并上传【合约包】。
  - 编辑合约包：对部署失败的智能合约进行重新上传。重新上传后，该智能合约将重新执行部署。
  - 查看部署详情：点击后弹窗展示该合约包的部署结果。
  - 删除合约包：部署失败的智能合约，可点击删除按钮执行合约删除操作，弹窗确定是否删除，点击【确定】则系统删除该智能合约，点击【取消】则不进行删除操作。
  - 下载合约包：点击后将该合约包下载至本地。
- 项目 key 操作
  - 在项目列表中点击【启用项目 key】，可生成项目 key；
  - 项目 key 启用后，点击【更新项目 key】按钮后，该项目 key 将重置，旧的项目 key 将无法使用；
  - 点击【禁用项目 key】，则该项目 key 不可用，如需恢复则点击【启用项目 key】按钮，系统将会为该项目生成新的项目 key。



- 下载接入参数
  - 在项目列表中点击【下载接入参数】按钮后，系统会将该项目的相关接入信息以 excel 表格的形式下载至本地。
  - 合约部署成功后，也可以在【下载接入参数】里面查看合约地址。
- 删除项目
  - 在项目列表中点击【删除项目】按钮后，弹窗确定是否删除，点击【确定】则系统删除该项目，点击【取消】则不进行删除操作。

7.3 网关接入说明

在完成项目的创建后，系统会为每个项目创建项目 ID、项目 KEY，用户在访问网关时需要通过以上字段进行接入授权验证，通过验证后即可访问链上。为保证访问安全，系统支持对项目 KEY 进行更新或禁用。

7.3.1 文昌链（基于 COSMOS/IRISnet SDK）网关接入说明

- 密钥算法

文昌链（基于 COSMOS/IRISnet SDK）通过 sm2 算法生成链账户地址以及公私钥和助记词，具体可以参考 SDK 中提供的方法。

- 接入方式

文昌链（基于 COSMOS/IRISnet SDK）支持 rpc 与 rest 接口。

- 密钥模式

支持密钥托管模式以及上传链账户模式，不支持非托管的 SDK 请求。在密钥托管模式下，用户无法直接通过 SDK 调用合约，但可以通过 REST-API 进行合约的部署与调用；

- 网关地址规则

只要传了 BsnAccount 认为是通过网关托管加签（密钥托管模式），若非托管（上传链账户地址模式）不需要传。

rest 访问地址为：

网关地址/api/[项目 id]/[BsnAccount]/[Protocol]/ {链上接口地址}；

如果使用项目 key 接入的话，需按照网关地址/api/[项目 id]/[BsnAccount]/[Protocol]/ {链上接口地址}格式拼接请求地址，同时请求报文头 header 中增加 x-api-key:{项目 key 值}。

示例：

密钥托管模式：https://opbningxia.bsngate.com:18602/api/722f0f239ac7422a8d8ae72406ea010f/OPB0722dac33ddd42038c4d9ed932da6b9a/rest/swagger/#/

上传链账户地址模式：https://opbningxia.bsngate.com:18602/api/722f0f949ac11aba822ae72406ea010f/rest/swagger/#/

rpc 访问地址为：

网关地址/api/[项目 id]/[BsnAccount]/[Protocol]/rpc；

如果使用项目 key 接入的话，需按照网关地址/api/[项目

id]/[BsnAccount]/[Protocol]/rpc 格式拼接请求地址，同时请求报文头 header 中增加 x-api-key:{项目 key 值}。

示例：

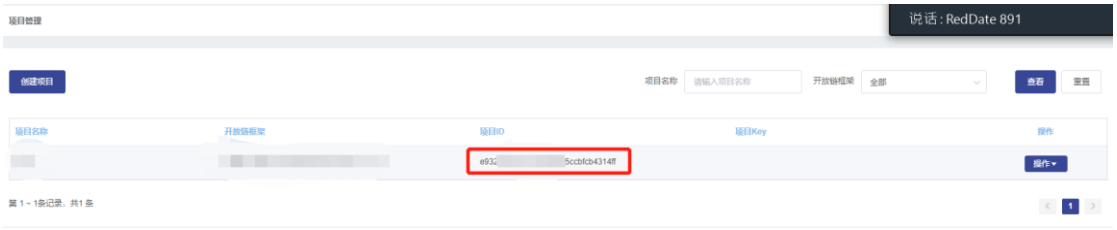
密钥托管模式：

https://opbningxia.bsngate.com:18602/api/722f0f949ac74aba8d8ae724eeea010f/OPB0728dacccddd42038c4d9ed95ada6b9a/rpc

上传链账户地址模式：

https://opbningxia.bsngate.com:18602/api/722f0f911ac74aba8d8ae72411ea010f/rpc

注释：项目 id：创建项目后生成的项目 ID，见下图；



BsnAccount：BSN 链账户标识，链账户管理->链账户详情页面可看到，如图：

Protocol：开放链-协议(rpc,rest)。

- 合约语言

文昌链（基于 COSMOS/IRISnet SDK）官方使用 Rust 语言进行合约的开发。

- 开发文档

https://github.com/bianjieai/bsn-docs/tree/main/irita-opb

- 开发实例

https://github.com/CosmWasm/cosmwasm-examples

7.3.2 泰安链（基于 FISCO BCOS）网关接入说明

- 密钥算法

泰安链（基于 FISCO BCOS）通过 ECDSA(secp256k1)算法生成链账户地址以及公私钥，具体可以参考 SDK 中提供的方法。或者使用 openssl 生成密钥，具体参考：

https://www.bsnbase.com/static/tmpFile/bzsc/helper/5-2.html

- 接入方式

泰安链（基于 FISCO BCOS）目前仅支持 JSON-RPC API 方式接入，API 文档参考：

https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/api.html

- 密钥模式

支持密钥托管模式、发布公钥模式以及上传链账户地址模式。

在发布公钥模式与上传链账户地址模式下，用户自己管理密钥，完全按照 API 文档发起交易。

在密钥托管模式下，由于 BSN 需要代替用户对上链数据进行签名，所以无法按照 API 文档报文格式提交交易，

需要按照以下格式提交数据：

```
{
    "method": "sendRawTransaction",
    "id": 1,
    "jsonrpc": "2.0",
    "params": {
        "blockNumber": 95, //当前块高
        "contractAbi": "合约 ABI",
        "contractBin": "合约 BIN", //非必填，若为合约部署该字段必填
        "contractAddress": "合约地址", //非必填，若为合约调用该字段必填
        "funcName": "合约方法", //非必填，若为合约调用该字段必填
        "funcParam" : [
            "参数"
        ],
        "groupId": 1
    }
}
```

影响接口为：

sendRawTransaction、sendRawTransactionAndGetProof

- 网关地址规则：

只要传了 BsnAccount 认为是通过网关托管加签（密钥托管模式）），若非托管（发布公钥模式和上传链账户地址模式）不需要传。

rpc 访问地址为：

网关地址/api/[项目 id]/[BsnAccount]/[Protocol]；

如果使用项目 key 接入的话，需按照网关地址/api/[项目 id]/[BsnAccount]/[Protocol]格式拼接请求地址，同时请求报文头 header 中增加 x-api-key:{项目 key 值}。

示例：

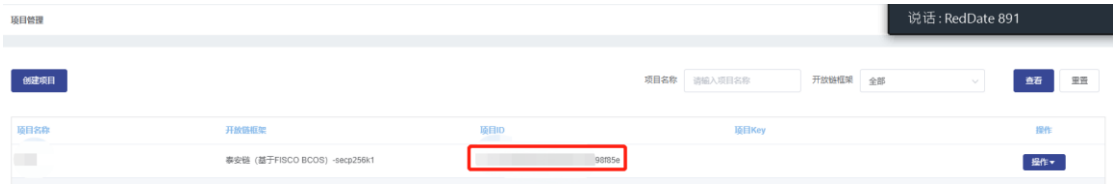
密钥托管模式：

https://opbningxia.bsngate.com:18602/api/88017b9a333ff484882f48f16c663453e/OPBb641225457ee447096357b7e77c4e0cc/rpc/

非托管模式：

https://opbningxia.bsngate.com:18602/api/88017b3333bdff484882f48f16c633453e/rpc/

注释：项目 id：创建项目后生成的项目 ID，见下图；



BsnAccount：BSN 链账户标识，链账户管理->链账户详情页面可看到，如图：

Protocol：开放链-协议(rpc)。

- 合约语言

泰安链（基于 FISCO BCOS）官方使用 Solidity 语言进行合约的开发。

- 开发文档：

https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/app_dev/index.html

- 开发实例：

https://github.com/WeBankBlockchain/SmartDev-Contract/tree/dev-bsn

8 IPFS 专网服务

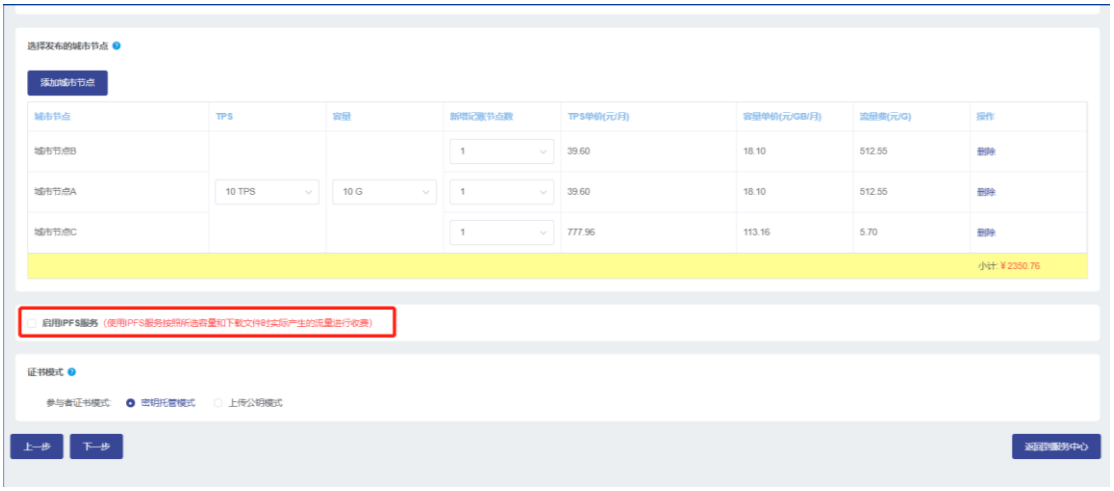
8.1 IPFS 专网概述

IPFS（Inter Planetary File System 星际文件系统）是去中心化存储、点对点传输的分布式文件系统。越来越多的区块链应用为了缓解其链上数据存储压力选择集成 IPFS 作为文件存储管理服务来提升应用处理的性能和效率。BSN 搭建了基于 IPFS 的分部署文件服务专网，面向应用开发提供 IPFS 常用的原生服务接口的服务网关，应用可无缝的与 IPFS 公网切换。

8.2 IPFS 服务开通

有两种方式可以开通 IPFS 服务：在发布服务时开通； 在【IPFS 服务】界面单独开通。服务开通后，系统会自动生成上传 key 和下载 key。

- 方式一：在发布服务时开通。
  - 用户只需要在发布服务时在启用 IPFS 服务选项处选择“启用 IPFS 服务”，账单完成支付后即可开通。



- 方式二：在【IPFS 服务】界面单独开通。

- 进入主页，点击【IPFS 服务】，选择【创建 IPFS 服务】进入服务创建页面。

- 选择容量，点击【下一步】，选择按月付费或按年付费

- 勾选【我已阅读并同意《BSN 服务发布协议》】，点击【确定】后，系统生成账单，并提示用户支付服务费用，用户确定以后，系统从用户的账户中进行扣款，扣款成功后服务实时开通。若

服务扣款失败，服务对应的账单将保留 72 小时，过期失效。若仍想使用服务，需重新开通服务并支付；

- 如果用户账户内没有余额或余额不足，需要先进行充值。然后到【我的账单】进行支付；
- 服务的费用主要是容量费用，流量使用费根据服务的实际使用量按周扣款，服务开通时无须支付，按年支付时具有一定的折扣优惠；
- 服务开通后，系统会返回上传 Key、下载 Key 及网关地址信息。上传 key 和下载 key 可根据用户需要进行更新。

- 点击【查看 IPFS 接入说明】，可查询到 IPFS 服务的接口地址。

IPFS接入说明

城市节点A上传网关地址：

功能名称	功能接口地址
2.1.1.上传文件或目录	
2.2.1写入IPFS块	
2.3.1.上传IPFS对象	
2.4.1.添加DAG节点	

注意：IPFS具体功能接入的接口地址需使用城市节点的网关地址+功能接口地址；如在"城市节点A"上传文件或目录 的接口地址为"

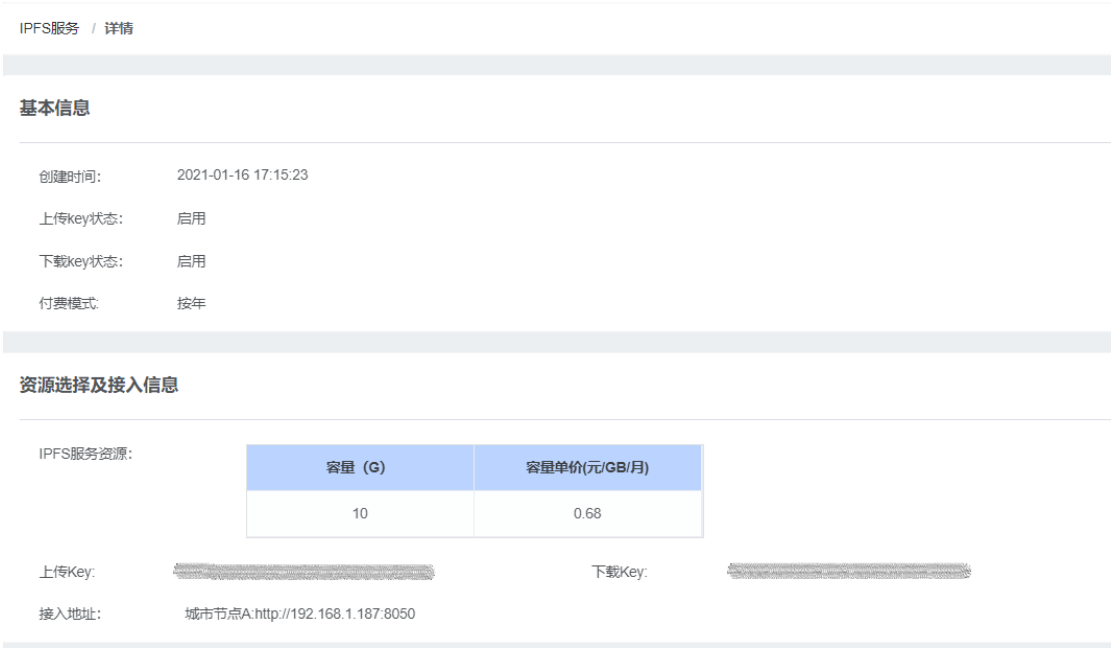
城市节点A下载网关地址：

功能名称	功能接口地址
获取所有数据块列表	/api/v0/ls
获取数据块子块信息	/api/v0/refs
显示IPFS对象	/api/v0/cat
下载IPFS对象	/api/v0/get
读取IPFS块	/api/v0/block/get
显示IPFS块	/api/v0/block/stat
格式化显示IPFS对象	/api/v0/object/get
读取IPFS对象原始数据	/api/v0/object/data
显示DAG节点统计信息	/api/v0/object/stat
获取DAG节点	/api/v0/dag/get
解析IPID块	/api/v0/dag/resolve
固定IPFS对象	/api/v0/pin/add
显示IPFS固定对象	/api/v0/pin/ls
解除IPFS对象的固定	/api/v0/pin/rm
获取IPFS版本	api/v0/version

确定

✧ 注意：如果用户只需要下载文件，则可不开通 IPFS 服务，只点击【创建下载 key】生成下载 key 即可。

■ 点击【查看】，可查询到 IPFS 服务的基本信息、资源及接入信息、资源使用信息以及文件上传信息。



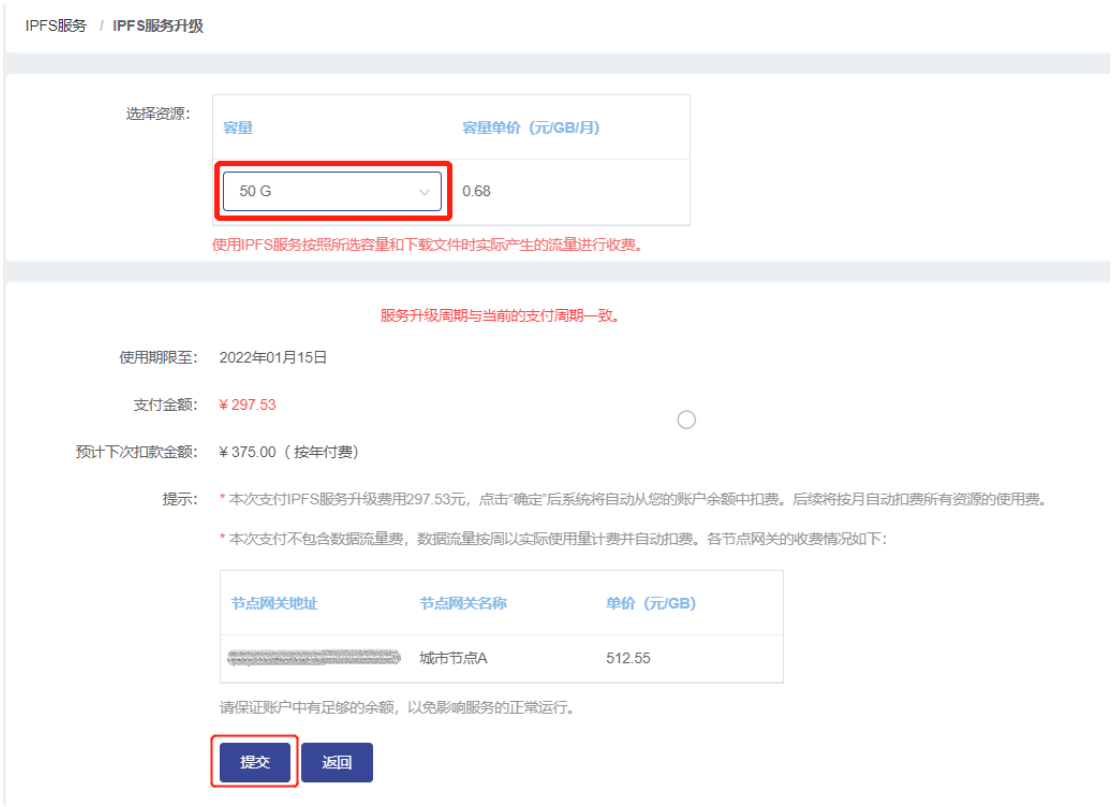
8.3 IPFS 服务升级与退订

IPFS 服务开通后，如果需要更多的容量来进行文件存储，用户可选择服务升级。如果不再需要 IPFS 服务，可选择退订，退订后，文件会被删除。

- IPFS 服务升级：
  - 点击【IPFS 服务】进入详情页面。

- 点击【升级】，进入升级页面。

- 点击【新增】，选择容量，提交，支付成功即可完成升级，升级后的付款周期与当前周期保持一致。



- IPFS 服务退订：
  - 点击【IPFS 服务】进入详情页面。

- 点击【退订】，会有弹窗提醒，点击【确定】，即可完成退订。

- 服务退订后，文件会被删除，上传 key 失效，不影响下载 key 的使用。系统会根据 IPFS 服务时间进行退款，其中按月支付的账单和流量使用账单不允许退款，按年支付的账单退款时会退未使用月份的费用，并且不再享受年度折扣优惠。

8.4 IPFS 服务网关接口

8.4.1 文件上传接口

通过此接口可以将文件或目录上传至 IPFS 网络。

1. 接口地址： https://节点网关地址/ipfs/[peer 名称]/上传 key/api/v0/add
2. 通讯方式：POST
3. 请求参数

序号	字段名	字段	类型	必填	备注
1	待添加文件	arg	file	Y	待添加到 ipfs 的文件路径 注：该参数应用于 body 中，其他参数应用于 URL
2	是否生成 DAG	trickle	bool	N	是否使用 trickle-dag 格式算法生成 DAG(有向无环图),默认为 false，当此节段为 false 时，生成 DAG 节点所对应的统计信息中链接数量大于 0 true 时链接数量等于 0
3	叶节节点是否使用原始块	raw-leaves	bool	N	是否为叶子节点使用原始块(实验参数) 当 trickle 和此参数为 true 时，DAG 节点统计信息中区块大小以及链接大小数据为 0 其它情况下 DAG 节点统计信息中区块大小以及链接大小数据大于 0
4	是否持久化	pin	bool	N	是否在添加的时候持久化 (不被垃圾回收所回收)
5	是否不输出	silent	bool	N	当 silent 为 true 时不返回，false 时则返回上传结果
6	流式进度数据	progress	bool	N	流式进度数据
7	是否递归	recursive	bool	N	是否递归添加目录路径
8	是否隐藏	hidden	bool	N	是否添加隐藏文件，只有在递归的时候才有用
9	是否提供符号链接	dereference-args	bool	N	提供的符号链接
10	stdin 名称	stdin-name	string	N	如果文件名称为 sou，则分配一个名称 rce 是 stdin
11	是否最小输出	quiet	bool	N	是否最小输出

12	是否只写入哈希值	quieter	bool	N	是否只写入哈希值
13	仅出块和 HASH	only-hash	bool	N	是否仅仅出块和 hash，不写入磁盘
14	是否使用目录来包装文件	wrap-with-directory	bool	N	是否使用目录对象来包装文件
15	成块算法	chunker	string	N	成块算法，默认值：size-262144

示例

```
// POST
curl "https://网关地址/ipfs/上传 key/api/v0/add? pin=true&trickle=false&raw-leaves=true&silent=false"\
    -X POST \
    -H "Content-Type: multipart/form-data" \
    -F file=@"/sample-result.json"
```

4. 响应参数

序号	字段名	字段	类型	备注
1	文件名	Name	string	
2	CID	Hash	string	
3	文件大小	Size	string	

示例

```
{
    "Name": "sample-result.json",
    "Hash": "QmSTkR1kkqMuGEeBS49dxVJjgHRMH6cUYa7D3tcHDQ3ea3",
    "Size": "2120"
}
```

8.4.2 获取所有数据块列表接口

通过调用此接口可以获取所有的数据块列表。

1. 接口地址： https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/ls
2. 通讯方式：POST
3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	IPFS 路径	arg	string	Y	要输出的对象的路径，此值为文件 CID 或文件哈希值

2	是否输出表头	headers	bool	N	
3	是否解析链接对象	resolve-type	bool	N	是否解析链接的对象来检测其数据类型，当 resolve-type 参数设置为 true 时，则表示强制显示分片大小和类型。
4	显示所有信息	size	bool	N	是否显示所有信息，当 resolve-type 参数设置为 true 时，此参数设置将会标记为无效，当 stream 参数设置为 false 时，设置 size 为 true 时，则会显示大小和类型，反之则不会显示
5	是否启用流式传输	stream	bool	N	在遍历目录项时启用实验性的流式传输，此字段必须为 true
示例					
// POST curl "https://网关地址/ipfs/下载 key/api/v0 /ls?arg=QmY2Pp7vcCgD5vw6oXn3HSxeAeDa2eujEaRSKGTA4BZUXB&headers=false&resolve-type=false&size=true "					

4. 响应参数

序号	字段名	字段	类型	备注
1	对象列表	Objects	[]Object	
Object				
1	CID	Hash	string	
2	链接列表	Links	[]Link	
Link				
1	数据块名称	Name	string	
2	数据块哈希	Hash	string	
3	数据块大小	Size	int	此响应参数会根据请求参数的 resolve-type 和 size 共同决定是否返回
4	数据块类型	Type	int	此响应参数会根据请求参数的 resolve-type 和 size 共同决定是否返回
5	目标信息	Target	string	
示例				


```
{
 "Objects": [{
  "Hash": "QmY2Pp7vcCgD5vw6oXn3HSxeAeDa2eujEaRSKGTA4BZUXB",
  "Links": [{
   "Name": "",
   "Hash": "QmaviNBByTz2sSM5eqEtK5nhsqazo7XYiJkK9BqTCxHUgh",
   "Size": 45613056,
```

```
    "Type": 2,
    "Target": ""
   }]
  }]
}
```

8.4.3 获取数据块子块信息

通过调用此接口可以获取数据子块信息。

1. 接口地址： https://节点网关地址/ipfs/[peer 名称]/下载key/api/v0/refs

2. 通讯方式：POST

3. 请求参数及响应数据

请求数据					
序号	**字段名**	**字段**	**类型**	**必填**	**备　　注**
1	对象路径	arg	string	Y	要列出引用的对象的路径
2	是否使用边缘格式	edges	bool	N	发出边缘格式。&lt；from&gt； -&gt； &lt；to>
3	是否唯一	unique	bool	N	从输出中省略重复的引用
4	最大深度	max-depth	int	N	仅对于递归引用，将 fetch 和 list 限制为给定深度
请求示例					

响应数据

序号	字段名	字段	类型	备注
1	CID 或数据块哈希	Ref	string	
2	异常信息	Err	string	

响应示例

Edges 参数为 true

```
{
    "Ref":          "QmSnB3aDCtumdjgMY3dWXhD2bq7yJfzs31AcgeRbJErnZJ          ->
QmPMMmfQXjFDHXG9Mm6miQSK78XeWgKJv5riL6Zotq6xuC",
    "Err": ""
}
```

Edges 参数为 false

```
{
    "Ref": "QmPMMmfQXjFDHXG9Mm6miQSK78XeWgKJv5riL6Zotq6xuC",
    "Err": ""
}
```

8.4.4 显示 IPFS 对象接口

显示或下载 IPFS 对象数据（此接口是将所有数据块拼成以后组合返回，针对命令进行了如下操作：ipfs cat hash1 hash2 hash2 ……，hashn > 文件）。

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/cat

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	IPFS 路径	arg	string	Y	要输出的对象的路径，此值为文件 CID 或数据块哈希

					注：该参数应用于 body 中其他参数应用于 URL
2	偏移量	offset	int64	N	开始读取的字节偏移量
3	长度	length	int64	N	要读取的最大字节数
示例					
// POST curl "https:// 网关地址/ipfs/下载 key /api/v0/cat?arg=QmY2Pp7vcCgD5vw6oXn3HSxeAeDa2eujEaRSKGTA4BZUXB&offset=0&length=5000" 响应返回 This endpoint returns a `text/plain` response body.					

8.4.5 下载 IPFS 对象接口

通过此接口可以对文件上传所对应的 IPFS 对象数据进行下载。

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/get

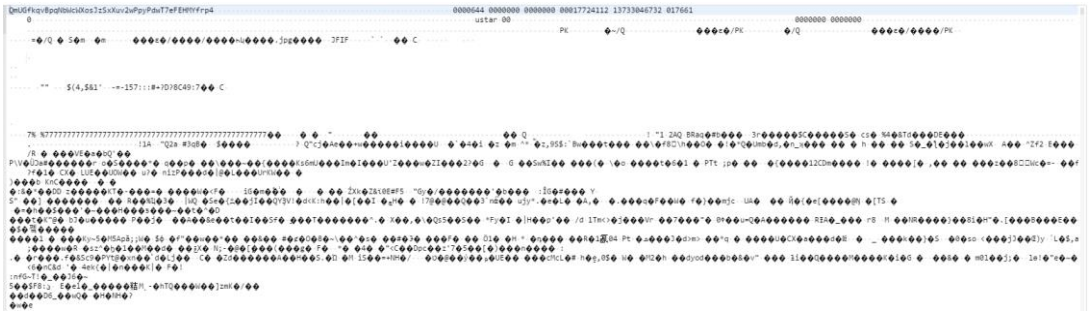
2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	IPFS 路径	arg	string	Y	要输出的对象的路径，此值为文件 CID 或文件哈希值
2	下载 key	downKey	String	Y	下载 key
3	输出结果路径	output	string	N	
4	是否输出 TAR 包	archive	bool	N	默认为否
5	是否启用压缩	compress	bool	N	GZIP 压缩格式，默认为否
6	压缩等级	compression-level	int64	N	等级范围：1~9
示例					
// POST curl "https:// 网关地址/ipfs/下载 key /api/v0/get?arg=QmY2Pp7vcCgD5vw6oXn3HSxeAeDa2eujEaRSKGTA4BZUXB&output=true&archive=true&compress=true&compression-level=9"					

4. 响应参数

下载成功。示例：



8.4.6 写入 IPFS 块接口

通过此接口可以将自定义文件数据结构上传至 IPFS 网络。

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/上传 key/api/v0/block/put

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	待添加文件	arg	file	Y	需要被存储为 IPFS 块的数据 注：该参数应用于 body 中，其他参数应用于 URL
2	格式	format	string	N	所创建块的 cid 格式,默认为 v0
3	multihash 类型	mhtype	string	N	默认为 sha-256
4	multihash 长度	mhlen	int	N	默认为-1
5	是否固定	pin	bool	N	以递归方式固定添加的块，默认为 false
示例					
// POST curl "https:// 网关地址/ipfs/上传 key /api/v0/block/put?format=v0&mhtype=sha2-256&mhlen=-1&pin=true " -X POST \ -H "Content-Type: multipart/form-data" \ -F file=@"/sample-result.json"					

4. 响应参数

序号	字段名	字段	类型	备注

1	键值	Key	string	
2	大小	Size	int	
示例				

{
"Key": "QmTStHSziV4cfuYaHKbRYnmBb5XCzdj67kV7jNvbKKr2qv",
"Size": 162
}				

8.4.7 读取 IPFS 块接口

通过此接口可以获取 IPFS 原始块数据。

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/block/ get

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	区块 KEY	arg	string	Y	添加文件所对应的文件哈希或写入 IPFS 块时所对应的 KEY 值，针对获取添加文件所对应的文件哈希，建议使用 cat 接口
示例					
// POST Curl "https:// 网关地址/ipfs/下载 key /api/v0/block/get?arg=QmPtW8pj7k5pwLDSdpGqaZNJ7RGEkdvveqek2YpmdKeRg1 "					

4. 响应参数

   显示成功

   示例：

8.4.8 显示 IPFS 块接口

通过此接口可以获取 IPFS 原始块摘要信息。

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/block/stat

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	区块 KEY	arg	string	Y	添加文件所对应的文件哈希或写入 IPFS 块时所对应的 KEY 值
示例					
// POST Curl "https:// 网关地址/ipfs/下载 key /api/v0/block/stat?arg=QmPtW8pj7k5pwLDSdpGqaZNJ7RGEkdvveqek2YpmdKeRg1"					

4. 响应参数

序号	字段名	字段	类型	备注
1	键值	Key	string	
2	大小	Size	int	此区块数据大小也可以通过显示 DAG 节点统计信息中进行获取，对应统计信息中的区块数据大小
示例				

{
Key: "QmfQ5QAjvg4GtA3wg3adpnDJug8ktA1BxurVqBD8rtgVjM",
Size: 18
}				

8.4.9 上传 IPFS 对象接口

通过此接口可以将自定义的 IPFS 对象格式文件上传至 IPFS 网

络，并输入存储为 DAG 对象和输出显示生成的密钥。

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/上传 key/api/v0/object/put

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	待添加文件	arg	file	Y	待添加到 ipfs 的文件路径 注：该参数应用于 body 中，其他参数应用于 URL。
2	是否递归添加目录内容	inputenc	string	N	输入数据的编码类型，protobuf 或 json 类型中的一个，默认值为 json
3	是否最小出输	datafieldenc	string	N	数据字段的编码类型，text 和 base64，默认值为 text
4	是否固定	pin	bool	N	添加时是否固定 IPFS 对象，默认为 false
5	是否最小输出	quiet	bool	N	是否最小输出
示例					
// POST Curl "https:// 网关地址/ipfs/上传 key /api/v0/object/put?inputenc=json&datafieldenc=text&pin=false " -X POST \ -H "Content-Type: multipart/form-data" \ -F file=@"/sample-result.json"					

4. 响应参数

序号	字段名	字段	类型	备注
1	CID	Hash	string	
示例				

{
Hash": "QmXg9Pp2ytZ14xgmQjYEiHjVjMFXzCVVEcRTWJBmLgR39V"
}				

8.4.10 格式化显示 IPFS 对象接口

通过此接口获取 IPFS 网络中序列化后 DAG 节点信息列表集合。

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/object/get

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	对象 KEY	arg	string	Y	要查询的对象的键值
2	数据编码类型	data-encoding	string	N	表示响应结果中的数据字段是以什么编码类型所对应的数据进行显示，text 或 base64，默认为 txt
示例					
// POST Curl "https:// 网关地址/ipfs/下载 key /api/v0/object/get?arg=QmaviNBByTz2sSM5eqEtK5nhsqazo7XYiJkK9BqTCxHUgh&data-encoding=base64"					

4. 响应参数

序号	字段名	字段	类型	备注
1	数据哈希	Data	string	数据哈希显示格式是根据请求参数中的 data-encoding 进行指定
2	Link 列表	Links	[]Link	
Link				
1	数据块名称	Name	string	
2	数据块哈希	Hash	string	
3	数据块大小	Size	int	此数据块大小对应 DAG 节点统计信息中的树的大小
示例				

```
{
"Hash": "<string>",
   "Links": [
      {
         "Hash": "<string>",
```

```
        "Name": "<string>",
        "Size": "<uint64>"
      }
  ]
}
```

8.4.11 读取 IPFS 对象原始数据接口

通过此接口可以获取 IPFS 对象的裸数据信息

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/object/data

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	对象 KEY	arg	string	Y	要查询的对象的键值，此值可以是 CID 也可以是 DAG 树节点所对应的哈希
示例					
// POST Curl "https:// 网关地址/ipfs/下载 key /api/v0/object/data?arg=QmPtW8pj7k5pwLDSdpGqaZNJ7RGEkdvveqek2YpmdKeRg1"					

4. 响应参数

序号	字段名	字段	类型	备注
1	原始字节	原始字节	String	
示例				

{
显示原始字节
}				

8.4.12 显示 DAG 节点统计信息接口

通过调用此接口可以获取 DAG 节点的统计信息。

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/object/stat

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	对象 KEY	arg	string	Y	要查询的对象的键值
2	是否可读显示	human	bool	N	以可读格式获取大小
示例					
// POST Curl "https:// 网关地址/ipfs/下载 key /api/v0/object/stat?arg=QmaviNBByTz2sSM5eqEtK5nhsqazo7XYiJkK9BqTCxHUgh "					

4. 响应参数

序号	字段名	字段	类型	备注
1	CID	Hash	string	
2	链接数量	NumLinks	number	节点包含的链接数
3	区块大小	BlockSize	number	本区块大小（链接大小+数据块大小）
4	链接大小	LinksSize	number	链接块部分的大小，此大小用于链接所有子对象所对应的大小
5	数据大小	DataSize	number	区块内数据所对应的大小
6	树的大小	CumulativeSize	number	树的大小（块大小+链接大小）
示例				

```
{
    "Hash": "QmaviNBByTz2sSM5eqEtK5nhsqazo7XYiJkK9BqTCxHUgh",
    "NumLinks": 174,
    "BlockSize": 8362,
    "LinksSize": 7659,
    "DataSize": 703,
    "CumulativeSize": 45623854
}
```

8.4.13 添加 DAG 节点接口

通过此接口可以向 IPFS 网络中添加一个 DAG 节点（DAG: directed acyclic graph）

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/上传 key/api/v0/dag/put

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	待添加文件	arg	file	Y	要添加的节点对象 注：该参数应用于 body 中，其他参数应用于 URL
2	转换成对象的格式	format	string	N	将需要添加的对象转化为指定格式。默认 cbor
3	格式	input-enc	string	N	将输入对象转化为指定格式。默认 json
4	是否固定	pin	bool	N	是否在添加的时候持久化，默认为 false
5	使用的哈希函数	hash	string	N	默认为 sha-256
示例					
// POST Curl "https:// 网关地址/ipfs/上传 key /api/v0/object/patch/rm-link?arg=QmbEs4v19wQXzYAKwnMAm6FF2Adwv72QSdEjizmwYiay1Y&arg=xmail_.pdf -X POST \ -H "Content-Type: multipart/form-data" \ -F file=@"/sample-result.json”					

4. 响应参数

序号	字段名	字段	类型	备注
1	hash	Cid	string	
示例				
{ "Cid": {				

```
      "/": "<cid-string>"
   }
}
```

8.4.14 获取 DAG 节点接口

通过此接口可以获取 IPFS 网络中的一个 DAG 节点信息。

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/dag/get

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	要获取的对象	arg	string	Y	
示例					
// POST Curl "https:// 网关地址/ipfs/下载 key /api/v0/dag/get?arg=QmaviNBByTz2sSM5eqEtK5nhsqazo7XYiJkK9BqTCxHUgh”					

4. 响应参数

序号	字段名	字段	类型	备注
1	数据内容	data	string	
2	link 数组	links	[]Link	
Link				
1	数据块名称	Name	string	
2	数据块哈希	CID	string	
3	数据块大小	Size	int	
示例				
{ { "data": "CAIYgIDgFSCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCA				

gBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAE
CCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQII
CAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggI
AQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgB
AggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECC
AgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICA
ECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQ
IICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAgg
IAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAg
BAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAEC
CAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIICAECCAgBAggIAQIIC
AECCAgBAggIAQIICAEA==",
    "links": [
        {
            "Name": "",
            "Size": 262158,
            "Cid": {
                "/": "QmPtW8pj7k5pwLDSdpGqaZNJ7RGEkdvveqek2YpmdKeRg1"
            }
        },
        {
            "Name": "",
            "Size": 262158,
            "Cid": {
                "/": "QmSsih1M4sac1p2rGFoknH9m4GJv3JEtcxR2wu3SohiKCm"
            }
        },
        {
            "Name": "",
            "Size": 262158,
            "Cid": {
                "/": "QmVvjSyBfa8csz9V28n7Dt65MCN8s3u3Mq5cVR8SkMj3q6"
            }
        }
    }

8.4.15 解析 IPID 块

通过此接口可以解析 IPID 块。

1. 接口地址： https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/dag/resolve

2. 通讯方式：POST

3. 请求参数及响应数据

请求数据

序号	字段名	字段	类型	必填	备　注
1	解析路径	arg	string	Y	解析路径

请求示例

```
http://192.168.1.172:10153/api/v0/dag/resolve?arg=QmaviNBByTz2sSM5eqEtK5nhsqazo7XYiJkK9BqTCxHUgh
```

响应数据

序号	字段名	字段	类型	备注
1	CID	CID	string	

响应示例

```
{
    "Cid": {
        "/": "QmaviNBByTz2sSM5eqEtK5nhsqazo7XYiJkK9BqTCxHUgh"
    },
    "RemPath": ""
}
```

8.4.16　固定 IPFS 对象接口

通过此接口可以将 IPFS 网络中对象固定到本地存储。

1. 接口地址：

   https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/pin/add

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	IPFS 路径	arg	string	Y	要锁定的对象路径
2	是否递归固定	recursive	bool	N	是否递归固定
3	进度是否显示	progress	bool	N	是否显示进度
示例					
// POST Curl "https:// 网关地址/ipfs/下载 key /api/v0/pin/add?arg=/ipfs/QmZZmY4KCu9r3e7M2Pcn46Fc5qbn6NpzaAGaYb22kbfTqm”					

4. 响应参数

序号	字段名	字段	类型	备注
1	固定对象信息列表	Pins	[]string	
示例				
{ "Pins": ["QmZZmY4KCu9r3e7M2Pcn46Fc5qbn6NpzaAGaYb22kbfTqm"]}				

8.4.17　显示 IPFS 固定对象接口

通过此接口可以列出 IPFS 网络中已经被固定到本地存储的对象列表。

1. 接口地址：

https://节点网关地址/ipfs/[peer 名称]/下载 key/api/v0/pin/ls

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	IPFS 路径	arg	string	N	要锁定的对象路径

2	键锁定类型	type	string	N	要列出的锁定键的类型 direct：直接的 indirect：间接的 recursive：递归的 all：所有类型，此字段如果根据类型进行查询，必须传已经存在锁定键的类型。
3	是否仅输出对象哈希	stream	bool	N	是否仅输出对象的哈希值
示例					
// POST Curl "https:// 网关地址/ipfs/下载 key /api/v0/pin/ls?arg=QmTStHSziV4cfuYaHKbRYnmBb5XCzdj67kV7jNvbKKr2qv&type=all&quiet=true&stream=true -X POST \ -H "Content-Type: multipart/form-data" \ -F file=@"/sample-result.json”					

4. 响应参数

序号	字段名	字段	类型	备注
stream 参数为 true				
1	CID	CID	string	
2	键锁定类型	Type	string	
stream 参数为 false				
1	键值数据列表	Keys	[]Key	
Key				
1	CID	CID	string	
2	键锁定类型	Type	string	
示例				

stream 参数为 true：

```
{
    "Cid": "QmTStHSziV4cfuYaHKbRYnmBb5XCzdj67kV7jNvbKKr2qv",
    "Type": "recursive"
}
```

stream 参数为 false：

```
{
    "Keys": {
        "QmTStHSziV4cfuYaHKbRYnmBb5XCzdj67kV7jNvbKKr2qv": {
            "Type": "recursive"
        }
    }
}
```

8.4.18 解除 IPFS 对象的固定接口

通过此接口可以将 IPFS 网络中已经固定的对象进行解除，解除后的对象进行删除操作。

1. 接口地址：

https://节点网关地址/ipfs/[peer 名称]/下载

key/api/v0/pin/rm

2. 通讯方式：POST

3. 请求参数

序号	字段名	字段	类型	必填	备　　注
1	IPFS 路径	arg	string	Y	要锁定的对象路径
2	是否递归解除固定	recursive	bool	N	是否递归解除固定
示例					
// POST Curl "https:// 网关地址/ipfs/下载 key/api/v0/pin/rm?arg=QmTStHSziV4cfuYaHKbRYnmBb5XCzdj67kV7jNvbKKr2qv”					

4. 响应参数

序号	字段名	字段	类型	备注
1	固定对象信息列表	Pins	[]string	
示例				

{
"Pins": [
"QmTStHSziV4cfuYaHKbRYnmBb5XCzdj67kV7jNvbKKr2qv"
]
}				

8.4.19 获取 IPFS 版本

通过此接口可以获取 IPFS 版本。

1.接口地址：

https://节点网关地址/ipfs/[peer 名称]/下载

key/api/v0/version

2 通讯方式：POST

3 请求参数

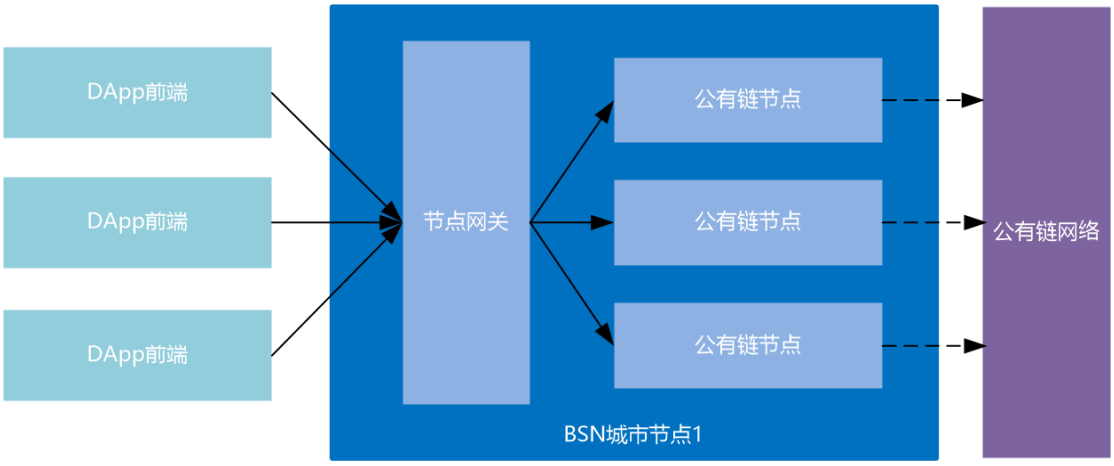
序号	字段名	字段	类型	必填	备　注
1	显示版本号	number	bool	N	是否只显示版本号
2	显示提交哈希	commit	bool	N	是否只显示提交哈希
3	显示仓储版本	repo	bool	N	是否显示仓储版本
4	显示所有信息	all	bool	N	是否显示所有信息
请求示例					
`http://192.168.6.243:5001/api/v0/version?number=false&commit=false&repo=false&all=false`					

9 公有链服务

公有链服务为 DApp 开发者提供了一个接入公链主网或测试网的节点途径。目前区块链服务网络（BSN）国际门户已经适配了多个公链框架，其它公链框架将按批次每月陆续进行适配。公有链开发者无需再自己搭建公有链网络节点，通过 BSN 便可直接接入公有链网络，而且每月仅需要支付极低的一次性月费，即可同时调用所有 BSN 已适配的公链节点。由于国内政策目前暂不开放公有链，公有链服务只在 BSN 国际门户使用，具体请访问：

https:// www.bsnbase.io/static/tmpFile/bzsc/6permissionlessservices/6-1.html

区块链服务网络（BSN）为公有链 DApp 开发者提供共享或专有的公有链节点（Public Nodes），开发者通过访问公共城市节点(Public City Nodes 或 PCN)的网关（gateway），便可快速的接入各类公有链网络。



开发者在 BSN 门户内选择公链框架（netcode）创建公链项目后，会得到节点网关的域名地址（url）、项目编号（id）、项目密钥（key），公链支持协议{protocol}，公链网关接口地址。

开发者通过 http 的方式访问节点网关，需按照 https://{url}/api/{id}/{netcode}/{protocol}/{subUrl}格式拼接请

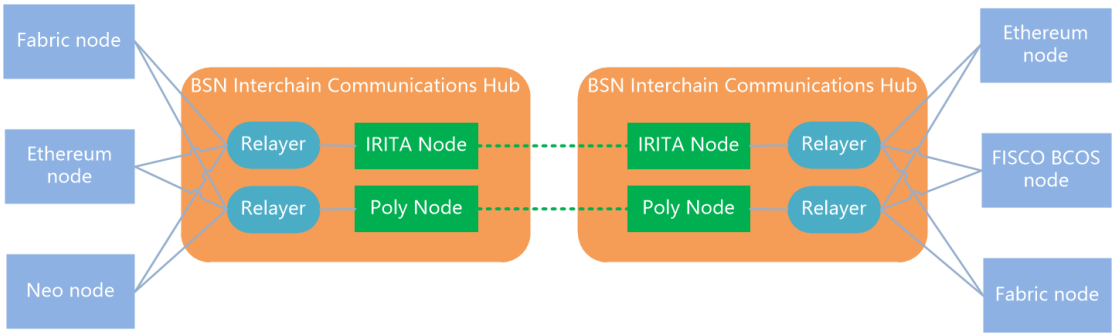
求地址，如启用了项目密钥需要在请求报文头 header 中增加 x-api-key:{key}。 如果公链节点提供了多个组件，则需增加{subUrl}，如 Nervos 的 CKB 除了 RPC 服务外，还有个 indexer 组件服务，{subUrl}应填写 indexer，{subUrl}为可选项。

开发者通过 websocket 的方式访问节点网关，需将 key 和 subUrl 拼接到目标机的路径地址内，按照{url}/api/{id}/{key}/{netcode}/{subUrl}格式进行拼接，如果没有启用项目密钥，则不需要填写{key}。如果没有 subUrl，则不需填写，即开发者可以将/api 后面的内容，认为是一个目标机的方法名。

BSN 国际门户现已适配了各种公链，更多公链信息及开发者资源请访问：
https://www.bsnbase.io/static/tmpFile/bzsc/6permissionlessservices/6-1.html

10 跨链服务

区块链跨链是两个或多个区块链网络间进行账本数据互操作的过程，实现数据、资产和信息的传递、交换。在区块链服务网络中每一个应用链维护自己的交易、共识和账本处理体系，承载着不同应用业务的数据信息。跨链机制实现应用链之间的数据共享和业务协同，破除链间“孤岛”局面，保障数据在多个应用链中安全可信流转，实现互通互联互操作。跨链的主要功能包括：跨链注册管理机制、跨链合约功能、跨链交易验证、跨链消息路由协议、跨链事务原子性保证等。



BSN 跨链通信枢纽采用异构链的跨链协议和双层结构设计，使用中继链作为跨链协调器，多条异构链作为跨链事务执行器，Relayer 作为跨链信息的搬运工，通过解决跨链信息的有效性、安全性和事务性等问题，实现了一套安全、易用、高效的跨链体系，支持如下特性：

- 支持同构链和异构链；
- 支持任意信息跨链；
- 接入简单方便，应用链不需要做定制开发适配，仅需要部署两个的智能合约；
- 事务性支持，不仅支持具有事务最终一致性需求的应用场景，而且同时还支持具有事务强一致性的需求应用场景，可支持任意事务且可扩展到任意数量的链；

- 跨链协议安全可靠，以密码学、共识算法等为基础，各应用链可自行验证跨链交易的合法性，从而保证跨链交互的安全性；

BSN 的“Interchain Communications Hub”（跨链通信枢纽）已进入商用，集成了分布科技的 Poly Enterprise 跨链解决方案，支持联盟链与联盟链、联盟链与 ETH 测试网及 NEO 测试网之间的相互跨链。基于 IRITA 的跨链解决方案也在适配中，预计将在下个迭代版本中推出商用版本。

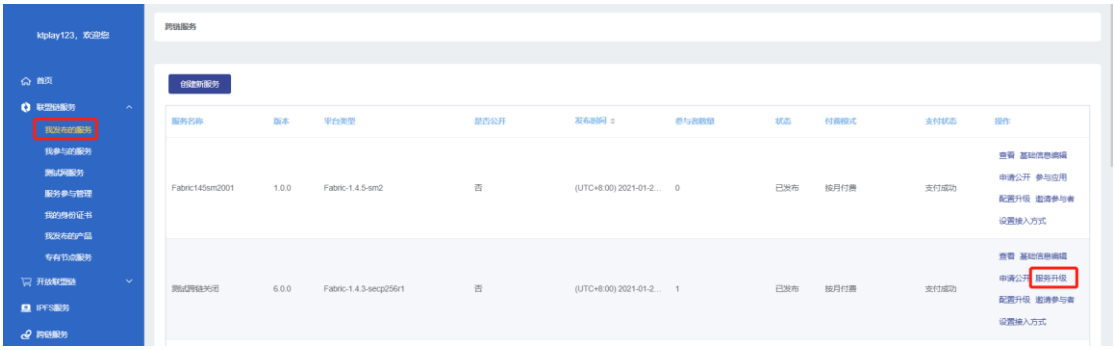
跨链服务也已在测试网上线，集成了分布科技的 Poly Enterprise 和边界智能的 IRITA 两种基于中继链机制的跨链解决方案。欢迎大家在测试网进行体验试用并反馈问题和开发建议，我们将持续完善功能。

10.1 跨链服务管理

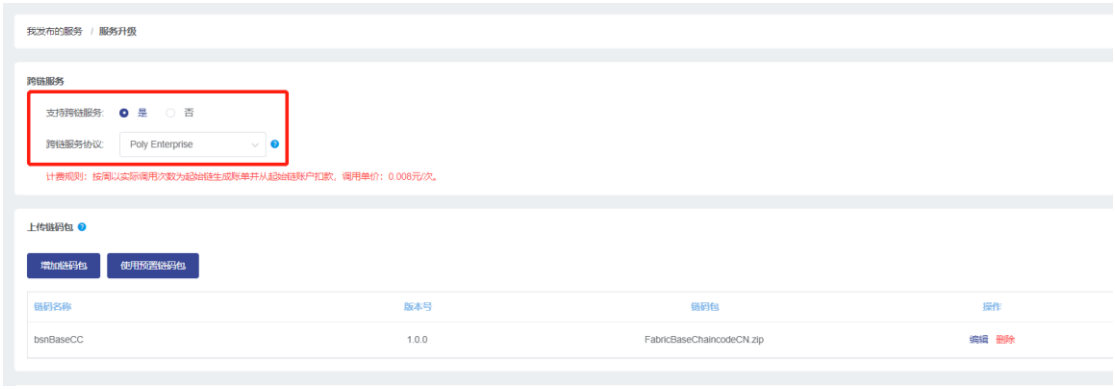
10.1.1 跨链服务开通

有两种方式可以开通跨链服务：发布者可以在服务升级时开通，也可以选择单独开通跨链服务。

- **方式一**：在服务升级时开通。对于已经发布完成的服务，发布者可以通过服务升级功能来开通跨链服务。
  - 在【联盟链服务】->【我发布的服务】页面，点击“操作”列的【服务升级】，进入服务升级页面；



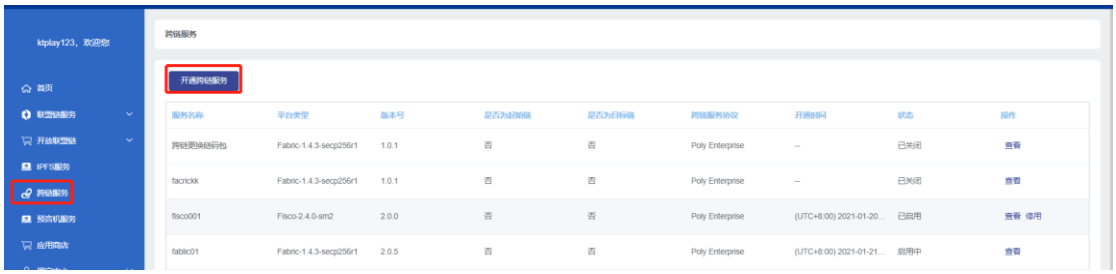
- 选择是否【支持跨链服务】并选择“跨链服务协议”，完成后点击【确定】提交即可，后台运营审核通过后，跨链服务开通成功。



☆ 注意：如果只开通跨链服务，不需上传新的链码包；开通跨链后，跨链调用时，需要起始链和目标链双方线下沟通跨链参数、方法及规范。

➢ **方式二：**在跨链服务模块单独开通。

- 进入主页，点击【跨链服务】



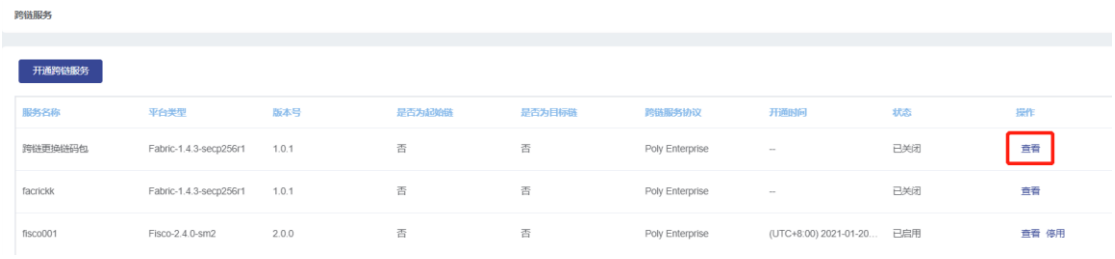
- 点击【开通跨链服务】进入【已发布的服务】页面，点击开通【开通跨链服务】；

- 后面开通流程与【方式一服务升级时开通】一致，可参考上文进行后续的开通操作。

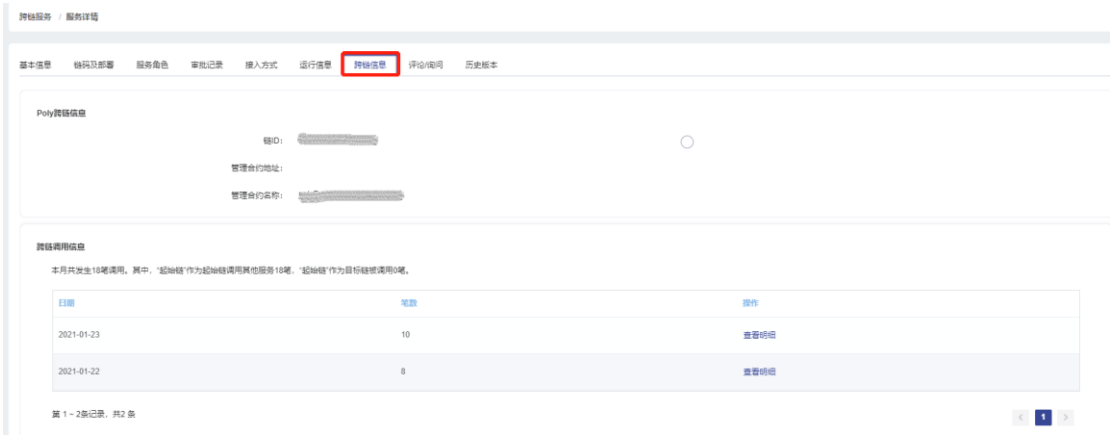
☆ 注意：已开通跨链的服务，跨链协议不可更改，只有在重新开通跨链服务时可重新选择。

10.1.2 跨链服务查看

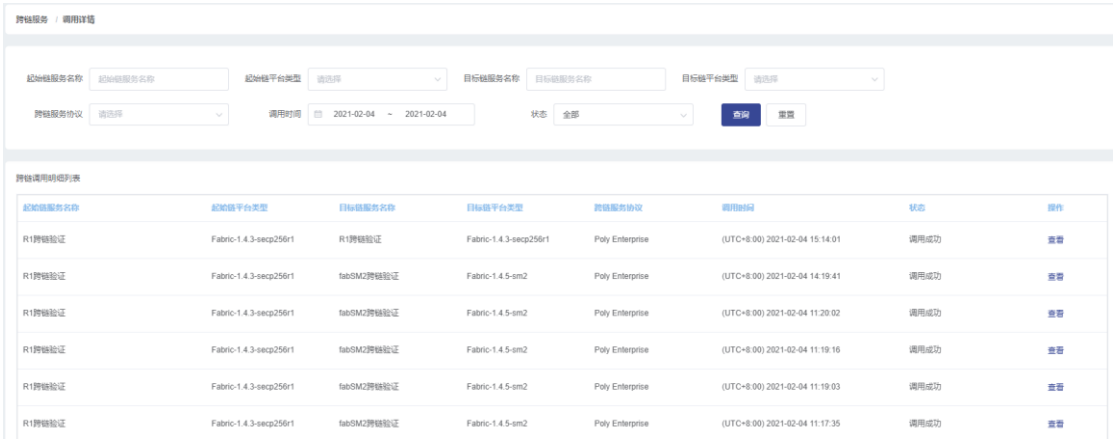
- 进入主页，点击在【跨链服务】，可看到我开通的跨链服务列表；



- 选择需要查看的服务，点击操作列【查看】按钮，选择【跨链信息】，可查看链 ID、管理合约地址、管理合约名称及跨链调用信息；



- 在跨链调用信息页面，点击【查看明细】，跳转至【调用详情】页面，填写跨链的相关查询条件，对跨链调用的明细信息进行查询，如图。

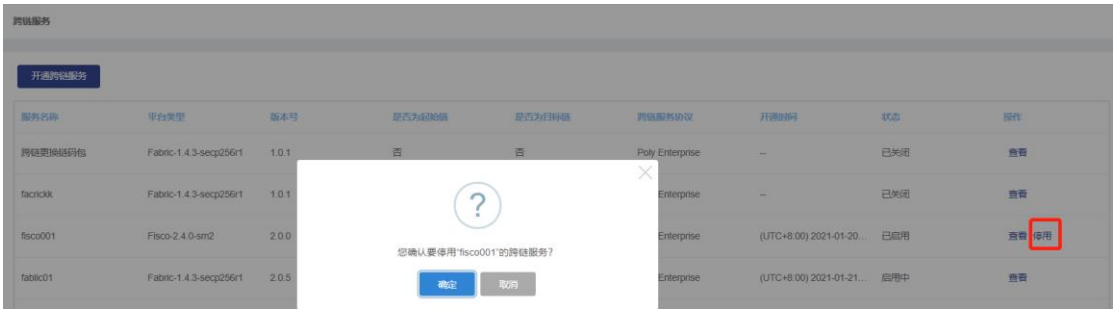


- 点击【跨链调用明细列表】中操作列的【查看】按钮，进入【跨链调用详情查看】页面，可对跨链调用详情的基本信息进行查看，如图。

10.1.3 跨链服务停用与启用

➢ 跨链服务的停用

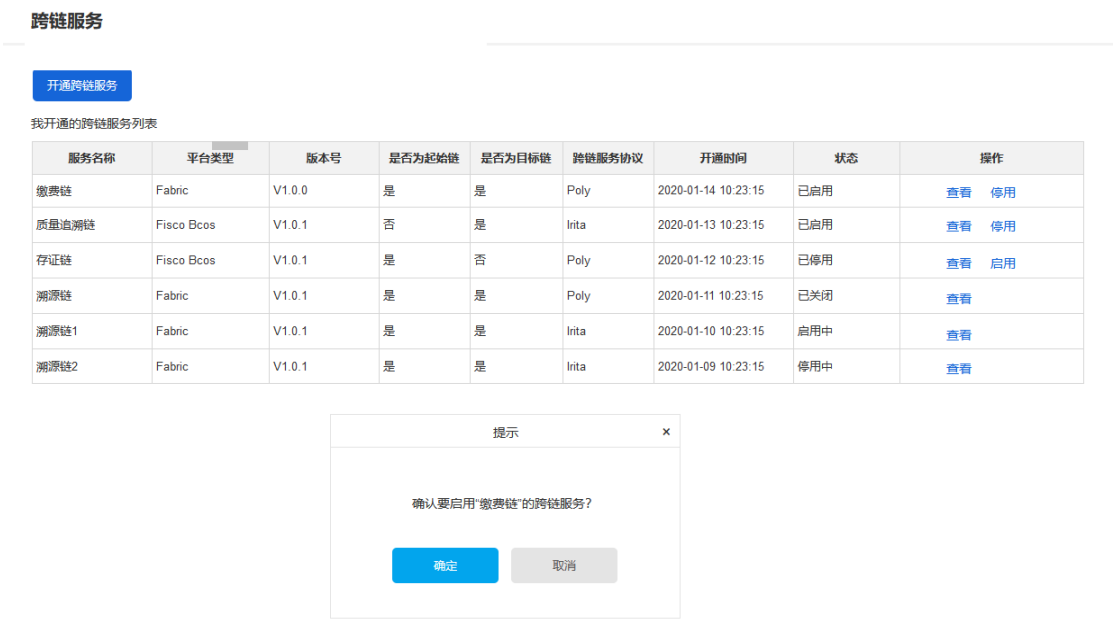
- 进入主页，点击在【跨链服务】，可看到我开通的跨链服务列表，选择需要停用的服务，点击操作列【停用】按钮。



- 弹出提示信息中的提示“确认要停用“缴费链”的跨链服务？”再点击【确定】按钮后，停用对应的跨链应用；

➢ 跨链服务的启用

- 进入主页，点击在【跨链服务】，可看到我开通的跨链服务列表，选择需要启用的服务，点击操作列【启用】按钮。



- 弹出提示信息中的提示“确认要启用“缴费链”的跨链服务？”再点击【确定】按钮后，启用对应的跨链服务。

10.2 基于 Poly Enterprise 的跨链

10.2.1 概述

一个完整的跨链交易交互需要编写多条链的应用合约，比如在以

太坊网络上有一个应用合约，在 BSN 服务网络上有一个 FISCO BCOS 应用合约，二者通过跨链协议保证互通信息的正确性进行跨链交互。跨链合约包含管理合约和应用合约：管理合约是跨链核心协议的实现内容，由 BSN 开发团队开发并部署在每条链上；应用合约需要区块链应用开发者依据跨链协议实现并部署在区块链网络中。

管理合约包括如下合约实现：

1. ETH 和 FISCO BCOS
   - EthCrossChainManager：包含管理的逻辑
   - EthCrossChainData：用于保存并操作数据
   - EthCrossChainManagerProxy：用于实现逻辑合约的升级
2. NEO
   - CCMC：包含管理的逻辑
3. Fabric
   - CCM：包含管理的逻辑
4. 测试网跨链管理合约地址

下面是基于 Poly Enterprise 的跨链服务所部署的框架名称，链 ID 以及跨链合约名称或地址。

测试网	框架名称	链 ID	跨链管理合约	应用演示合约
中国	Fabric	8	ccm	myhellopoly
	FISCO BCOS	6	0x315bF6da1f040355d283CCc41f04e8933Ff98f01	0x69d0ba0866ee3d9abd19b06ad8ac6f49023e19b8
国际	Fabric	9	ccm	myhellopoly
	FISCO BCOS	7	0x327A05F0f7F575206BE10AcB80D98845C1c0033e	0xbc393345ab22eea8f16ac98dd430657903e19e73
Ropsten	Ethereum	2	0xF6993b7d73B2827420689Dbc0b3068D24E6e467F	0x0b89e4f2103c4700de5ae96f370f3708c5572211
Testnet	Neo	4	0x10b6edbb6e44188d0ff39065442081b13bbd109b	0x0ea9e760ca350d950d01b32c35127b3f7c0c18b5

应用跨链包含如下功能：

1. Outbound 是由发起跨链交易请求的源链应用合约实现，将交易请求信息从源链应用合约发送到目标链应用合约。用户可以调用源链应用合约中自行定义的某个方法，该方法需要实现调用跨链管理合约的 crossChain 方法，跨链管理合约会通过事件把跨链信息发送到目标链。
2. Inbound 是由接收跨链交易请求的目标链应用合约实现，将从源链发来的交易请求信息传进目标链应用合约。跨链管理合约会接收并验证接收到的跨链信息，跨链协议要求在跨链信息里面包含目标链应用合约和函数名，然后管理合约会调用指定合约地址的指定方法，并把指定的信息传递给目标链应用合约。

10.2.2 Hyperledger Fabric 的跨链

10.2.2.1 BSN 生产环境应用合约开发指南

Fabric 应用合约开发根据自身的业务场景而定，主要实现包含两部分功能：如果是发起跨链交易的源链，其应用合约通过跨链去访问目标链需要实现 outbound，如果是接收跨链交易的目标链，其应用合约需要实现 inbound。Fabric 的链 ID 和跨链管理合约的名称在用户开通跨链服务时会通过 BSN 运维管理系统自动分配和生成，可在 BSN 门户进行查看。

以下是 BSN 生产环境中源链发起跨链交易调用的示例：

```
  /**
 * @Author AndyCao
 * @Date 2020-11-4 18:27
 * @Description 此方法用于对其它目标链进行跨链调用（此方法可自行定义）
 * @Param
```

```
 * @Return
 **/
func (t *HelloPoly) say(stub shim.ChaincodeStubInterface, args []string)
peer.Response {
   setLogger("The say method is called......")
   defer setLogger("End called say method......")
   // 参数检查
   if len(args) != 4 {

      return shim.Error("Parameter error ! ! ! ")

   }
   // 保存数据
   if err:=stub.PutState(args[1],[]byte(args[2]));err!=nil{
      return shim.Error(fmt.Sprintf("Failed to save data: %v", err))
   }
   // 构建跨链管理合约调用参数
   // args[0] 为目标链 id
   // args[1] 为目标链应用合约名称（或合约地址）
   // args[2] 为目标链调用方法名
   // args[3] 为调用参数
   ccArgs := []string{"crossChain",
   args[0],
   hex.EncodeToString([]byte(args[1])),
   args[2],
   hex.EncodeToString([]byte(args[3]))}
   // 调用跨链管理合约
   var resp = stub.InvokeChaincode(string("ccm"), packArgs(ccArgs), "")
if resp.Status != shim.OK {
      return shim.Error(fmt.Sprintf("Failed to call the cross-chain
management contract %s: %s", "ccm", resp.Message))
   }
   // 设置事件通知
   if err := stub.SetEvent("from_ccm", resp.Payload); err != nil {
      return shim.Error(fmt.Sprintf("Event setting failed: %v", err))
   }
   setLogger("Successfully call the cross-chain management contract for
cross-chain: (target chain ID: %d, target chain contract address: %x,
cross-chain message: %s)", args[1], args[1], args[2])
   return shim.Success(nil)
}
func packArgs(args []string) [][]byte {
   r := [][]byte{}
   for _, s := range args {
      temp := []byte(s)
```

```
        r = append(r, temp)
    }
    return r
}
```

BSN 生产环境中目标链接收跨链交易调用的示例：

```
/**
 * @Author AndyCao
 * @Date 2020-11-4 18:28
 * @Description 此方法用于对其它目标链进行跨链调用（此方法可自行定义）
 * @Param
 * @Return
 **/
func (t *HelloPoly) hear(stub shim.ChaincodeStubInterface, args []string)
peer.Response {
    setLogger("The hear method is called......")
    defer setLogger("End called hear method......")
    // 参数检查
    if len(args) != 1 {

        return shim.Error("Parameter error！！！")

    }

    // 保存源链所提交的跨链信息
    if err:=stub.PutState("FABRIC_CROSS_CHAIN",[]byte(args[0]));err!=nil{
        return shim.Error(fmt.Sprintf("Failed to save data: %v", err))
    }
    return shim.Success([]byte("SUCCESS"))
}
```

10.2.2.2 BSN 测试网应用合约开发指南

Fabric 在中国测试网链 ID 是 8，在国际测试网的链 ID 是 9，这个链 ID 是注册在 Poly 网络中的链 ID，而非 Fabric 所对应的通道。Fabric 跨链合约的名称为 ccm。

以下是 BSN 测试网源链发起跨链交易调用的示例：

```
/**
 * @Author AndyCao
 * @Date 2020-11-4 18:27
 * @Description 此方法用于对其它目标链进行跨链调用（此方法可自行定义）
 * @Param
```

```
 * @Return
 **/
func (t *HelloPoly) say(stub shim.ChaincodeStubInterface, args []string)
peer.Response {
   setLogger("The say method is called......")
   defer setLogger("End called say method......")
   // 参数检查
   if len(args) != 4 {

      return shim.Error("Parameter error！！！")

   }
   // 保存数据
   if err:=stub.PutState(args[1],[]byte(args[2]));err!=nil{
      return shim.Error(fmt.Sprintf("Failed to save data: %v", err))
   }
   // 构建跨链管理合约调用参数
   invokeArgs := make([][]byte, 6)
   // 设置跨链管理合约被调用的方法
   invokeArgs[0] = []byte("crossChain")
   // 设置目标链在 Poly 网络中所对应的链 ID（国内网所对应的链 ID 是 8，国际网所对应的
链是 9）
   invokeArgs[1] = []byte(args[0])
// 设置目标链应用合约地址，注：
   // 1、目标链为 fabric，则为应用合约的名称，如：mycc，目标链为
fisco/eth/neo，则为应用合约地址，如：11...，前面不要加 0x
   // 2、传给跨链管理合约参数必须用 hex.EncodeToString 将 bytes 转换成 16 进制
字符串，再转换成 byte 数组
   if args[0]=="8"||args[0]=="9"{
       invokeArgs[2] = []byte(hex.EncodeToString([]byte(args[1])))
    }else{
       invokeArgs[2] = []byte(args[1])
    }
   // 目标链应用合约方法
   invokeArgs[3] = []byte("hear")
   // 目标链应用合约所需要传递的跨链信息（注：传给跨链管理合约参数必须用
hex.EncodeToString 将 bytes 转换成 16 进制字符串，再转换成 byte 数组）
   invokeArgs[4] = []byte(hex.EncodeToString([]byte(args[2])))
   // 应用合约的名字
   invokeArgs[5] = []byte(args[3])
   // 调用跨链管理合约
   var resp = stub.InvokeChaincode(string("ccm"), invokeArgs, "")
   if resp.Status != shim.OK {
      return shim.Error(fmt.Sprintf("Failed to call the cross-chain
management contract %s: %s", "ccm", resp.Message))
```

```
    }
    // 设置事件通知
    if err := stub.SetEvent("from_ccm", resp.Payload); err != nil {
        return shim.Error(fmt.Sprintf("Event setting failed: %v", err))
    }
    setLogger("Successfully call the cross-chain management contract for
cross-chain: (target chain ID: %d, target chain contract address: %x,
cross-chain message: %s)", args[1], args[1], args[2])
    return shim.Success(nil)
}
```

BSN 测试网目标链接收跨链交易调用的示例：

```
/**
 * @Author AndyCao
 * @Date 2020-11-4 18:28
 * @Description 此方法用于对其它目标链进行跨链调用（此方法可自行定义）
 * @Param
 * @Return
 **/
func (t *HelloPoly) hear(stub shim.ChaincodeStubInterface, args []string)
peer.Response {
    setLogger("The hear method is called......")
    defer setLogger("End called hear method......")
    // 参数检查
    if len(args) != 1 {

        return shim.Error("Parameter error！！！")

    }

    // 保存源链所提交的跨链信息
    if err:=stub.PutState("FABRIC_CROSS_CHAIN",[]byte(args[0]));err!=nil{
        return shim.Error(fmt.Sprintf("Failed to save data: %v", err))
    }
    return shim.Success([]byte("SUCCESS"))
}
```

10.2.2.3 演示合约实例

- BSN 生产环境 Github:

  https://github.com/BSNDA/ICH/tree/main/sample/polychain/fabric-contract/online/hellopoly

- BSN 测试网 Github:

https://github.com/BSNDA/ICH/tree/main/sample/polychain/fabric-contract/testnet/hellopoly

10.2.3 FISCO BCOS 的跨链

10.2.3.1 BSN 生产环境应用合约开发指南

FISCO BCOS 应用合约开发根据自身的业务场景而定，主要实现包含两部分功能：如果是发起跨链交易的源链，其应用合约通过跨链去访问目标链需要实现 outbound，如果是接收跨链交易的目标链，其应用合约需要实现 inbound。FISCO BCOS 的链 ID 和跨链管理合约的地址及名称在用户开通跨链服务时会通过 BSN 运维管理系统自动分配和生成，可在 BSN 门户进行查看。

以下是 BSN 生产环境中

```
/**
 * @dev 通过调用 say 方法实现跨链调用
 * @param _toChainId 被调用的目标链在 Poly 网络中所对应的链 ID
 * @param _somethingWoW 跨链传递的参数
 * @return bool
 **/
function say(uint64 _toChainId, bytes _somethingWoW) public returns (bool){
    //获取跨链管理合约接口
    IEthCrossChainManagerProxy eccmp = IEthCrossChainManagerProxy(managerProxyContract);
    //获取跨链管理合约地址
    address eccmAddr = eccmp.getEthCrossChainManager();
```

```
    //获取跨链管理合约对象

    IEthCrossChainManager eccm = IEthCrossChainManager(eccmAddr);

    //获取目标链应用合约地址

    bytes memory toProxyHash = proxyHashMap[_toChainId];

    //调用跨链

    require(eccm.crossChain(_toChainId, toProxyHash, "hear", _somethingWoW), "CrossChainManager
crossChain executed error!");

    emit Say(_toChainId, toProxyHash, _somethingWoW);

    return true;

}
```

BSN 生产环境目标链接收跨链交易调用的示例：

```
/**

 * @param _somethingWoW 跨链传递的参数

 * @param _fromContractAddr 被调用的应用合约地址

 * @param _toChainId 被调用的合约框架 chainId

 * @return bool

 **/

function hear(bytes _somethingWoW, bytes _fromContractAddr, uint64 _toChainId) public returns (bool){

    hearSomeThing = _somethingWoW;

    emit Hear(_somethingWoW, _fromContractAddr);

    return true;

}
```

10.2.3.2 BSN 测试网应用合约开发指南

FISCO BCOS 在中国测试网链 ID 是 6，在国际测试网的链 ID 是 7。FISCO BCOS 链 Enterprise 链 ID，非 FISCO BCOS 群组 ID。BSN 测试网中应用合约示例与生产环境一致，详情请访问 10.2.3.1BSN 生产环境应用合约开发指南。

10.2.3.3 演示合约实例

- BSN 生产环境与 BSN 测试网 Github: https://github.com/BSNDA/ICH/tree/main/sample/polychain/fisco_contracts/contracts

10.2.4 Ethereum Ropsten 的跨链

10.2.4.1 应用合约开发指南

ETH 应用合约开发根据自身的业务场景而定，主要实现包含两部分功能：如果是发起跨链交易的源链，其应用合约通过跨链去访问目标链需要实现 outbound，如果是接收跨链交易的目标链，其应用合约需要实现 inbound。ETH 测试网的链 ID 是 2。链 ID 是注册在 Poly Enterprise 中的链 ID。此配置适用于 BSN 生产环境与 BSN 测试网。

源链发起跨链交易调用的示例：

```
/**
 * @dev 通过调用 say 方法实现跨链调用
 * @param _toChainId 被调用的目标链在 Poly 网络中所对应的链 ID
 * @param _somethingWoW 跨链传递的参数
```

```
 * @return bool

 **/

function say(uint64 _toChainId, bytes _somethingWoW) public returns (bool){

    //获取跨链管理合约接口

    IEthCrossChainManagerProxy eccmp = IEthCrossChainManagerProxy(managerProxyContract);

    //获取跨链管理合约地址

    address eccmAddr = eccmp.getEthCrossChainManager();

    //获取跨链管理合约对象

    IEthCrossChainManager eccm = IEthCrossChainManager(eccmAddr);

    //获取目标链应用合约地址

    bytes memory toProxyHash = proxyHashMap[_toChainId];

    //调用跨链

    require(eccm.crossChain(_toChainId, toProxyHash, "hear", _somethingWoW), "CrossChainManager
crossChain executed error!");

    emit Say(_toChainId, toProxyHash, _somethingWoW);

    return true;

}
```

目标链接收跨链交易调用的示例：

```
/**

 * @param _somethingWoW 跨链传递的参数

 * @param _fromContractAddr 被调用的应用合约地址

 * @param _toChainId 被调用的合约框架 chainId
```

```
 * @return bool

 **/

function hear(bytes _somethingWoW, bytes _fromContractAddr, uint64 _toChainId) public returns (bool){

    hearSomeThing = _somethingWoW;

    emit Hear(_somethingWoW, _fromContractAddr);

    return true;

}
```

10.2.4.2 演示合约实例

- GitHub：https://github.com/BSNDA/ICH/tree/main/sample/polychain/eth_contracts/contracts

10.2.5 Neo 测试网的跨链

10.2.5.1 应用合约开发指南

Neo 应用合约开发根据自身的业务场景而定，主要实现包含两部分功能：如果是发起跨链交易的源链，其应用合约通过跨链去访问目标链需要实现 outbound，如果是接收跨链交易的目标链，其应用合约需要实现 inbound。Neo 测试网的链 ID 是 4。链 ID 是注册在 Poly Enterprise 中的链 ID。此配置适用于 BSN 生产环境与 BSN 测试网。

源链发起跨链交易调用的示例：

```
/// <summary>
        /// 此方法用于对其它目标链进行跨链调用（此方法可自行定义）
        /// </summary>
        /// <param name="toChainId">目标链在 Poly 网络中所对应的链 ID</param>
```

```
        /// <param name="msg">目标链应用合约所需要传递的跨链信息</param>
        /// <returns></returns>
        [DisplayName("say")]
        public static bool Say(BigInteger toChainId, byte[] msg)
        {
            // 获取目标链上的应用合约
            var toProxyHash = HelloPoly.GetProxyHash(toChainId);

            // 获取 CCMC 合约地址
            var ccmcScriptHash = HelloPoly.GetProxyHash(neoChainID);

            // 跨链调用
            bool success = (bool)((DynCall)ccmcScriptHash.ToDelegate())("CrossChain", new object[]
{ toChainId, toProxyHash, "hear", msg });

            HelloPoly.Notify(success, "[HelloPoly]-Say: Failed to call CCMC.");

            // 事件通知
            HelloPoly.SayEvent(toChainId, toProxyHash);
            return true;
        }
```

目标链接收跨链交易调用的示例：

```
/// <summary>
        /// 此方法用于对其它目标链进行跨链调用（此方法可自行定义）
        /// </summary>
        /// <param name="fromChainId">源链在 Poly 网络中所对应的链 ID</param>
        /// <param name="toChainId">目标链在 Poly 网络中所对应的链 ID</param>
        /// <param name="msg">接收到源链发送的跨链信息</param>
        /// <param name="callingScriptHash">回调脚本哈希</param>
        /// <returns></returns>
        [DisplayName("hear")]
        public static bool Hear(byte[] inputBytes, byte[] fromProxyContract, BigInteger fromChainId, byte[]
callingScriptHash)
        {
            // 写入账本
            Storage.Put(fromProxyContract, inputBytes);

            // 事件通知
            HearEvent(fromChainId, fromProxyContract, inputBytes);

            return true;
        }
```

10.2.5.2 演示合约实例

- GitHub: https://github.com/BSNDA/ICH/tree/main/sample/polychain/neo_contracts

10.3 基于 IRITA 的跨链

10.3.1 概述

BSN IRITA 跨链服务是基于 BSN 跨链通信枢纽组成的跨链网络，在整个跨链通信中，跨链服务调用者通过自己的应用合约发起跨链调用，部署应用合约的框架的 Relayer 服务在获得跨链请求后向 HUB 发起跨链交易，对应的跨链服务的 Service Provider 在获得请求之后，向目标链发起交易，获取交易结果后向 HUB 返回交易结果，Relayer 在收到 HUB 的返回结果后调用跨链合约向应用链返回交易结果。

在整个跨链交易流程中，由应用合约、跨链合约、Relayer、HUB、Service Provider、服务等完成整个调用流程。

测试网	框架名称	跨链管理合约名称/地址
中国	Fabric	cc_cross
	FISCO BCOS	0xdaa3b22adeef09c9416f01db654035ba8c729522
国际	Fabric	cc_cross
	FISCO BCOS	0xcdad7e31edb24fc5f7c1aaf9edb8b8640d2fe3ca

10.3.2 基于 IRITA 的跨链架构

10.3.2.1 跨链合约

10.3.2.1.1 Service Market

Service Market 是负责管理跨链服务信息的模块，包含着以下

的功能：

- 添加服务绑定（Add Service Binding）：向应用链增加可用的跨链服务信息，包含：服务名、服务描述、服务提供者等信息。
- 更新服务绑定（update Service Binding）：当跨链服务信息需要变更时，可以调用该功能，修改跨链服务绑定信息。
- 获取服务信息列表（get Service Bindings）：查询当前可用的跨链服务列表。

10.3.2.1.2 Service Core

Service Core 是整个跨链合约的核心，负责接收应用合约的跨链请求、relayer 返回的请求结果、并且回调应用合约返回跨链结果，包含以下功能：

- Call Service：发起跨链调用的功能，由应用合约调用，传入调用的跨链服务名、输入参数、回调的合约信息等，调用成功将返回唯一的请求 ID。
- Set Response：返回跨链调用结果，由 Relayer 调用该功能，向应用链写入跨链调用的结果，如果在服务调用时需要回调某个合约方法，将在该功能中回调对应的合约。
- Get Response：查询跨链调用结果，可以由应用合约调用，根据跨链交易请求 ID 查询本次请求的结果信息。

10.3.2.2 应用合约

应用合约是由用户开发的调用跨链服务的合约，在应用合约中可以调用跨链合约的 Call Service 功能，发起一个跨链调用，并且提供一个 Callback 的功能供跨链合约返回跨链交易的结果，如不提供，需要用户在合约中调用 Get Response 主动查询服务调用结果。

10.3.2.3 Relayer

Relayer 是负责监听跨链合约向 HUB 提交跨链请求的服务，在 Relayer 中，将会负责监听来自跨链合约的 Call Service 功能的事件，收到跨链请求后将会向 HUB 提交对应的跨链请求。

10.3.2.4 Service Provider

Service provider 是监听 HUB 的跨链请求，向目标服务发起调用并且返回交易结果的服务，它连接着 HUB 和目标跨链服务。

10.3.2.5 跨链服务

由服务提供者开发的服务，可以是某一个区块链上的合约，供其他区块链上的应用跨链调用。一个服务成为跨链服务需要 Service Provider 的支持。

10.3.2.6 Interchain Communications Hub

跨链枢纽，是 BSN 跨链服务的核心组件，负责接收 Relayer 提交的跨链请求，验证交易，向 Service Provider 发起跨链交易，获取交易结果，验证并返回 Relayer。

10.3.3 BSN 测试网中的跨链服务

在 BSN 测试网中，我们提供了基于 ETH Ropsten 测试网部署的 MintBase 合约以及基于 FISCO BCOS 部署的存证合约做为跨链服务，用户可以在 BSN 测试网的 Fabric 、FISCO BCOS 应用链中调用这两个服务来体验跨链服务。

调用服务之前需要用户开发应用合约或者使用 BSN 提供的示例合约来调用。

10.3.3.1 基于 Fabric 框架的跨链应用合约

10.3.3.1.1 应用合约开发说明

1. 开发准备：

需要首先获取 BSN 跨链消费合约的帮助包，目前只有 go 语言版本，后续会增加其他版本。

```
cd $GOPATH
        mkdir -p src/github.com/BSNDA && cd src/github.com/BSNDA
        git clone https://github.com/BSNDA/ICH.git
```

2. 跨链调用：

创建 Fabric 链码对象和 Invoke 方法后，引入下面的包

```
import (
    "github.com/BSNDA/ICH/sample/irita/consumers/fabric/crosschaincode"
)
```

在 invoke 方法中直接调用 crosschaincode.CallService 方法，该方法的参数如下:

- stub : shim.ChaincodeStubInterface
- serviceName：调用的跨链服务名称，ETH 的为 nft ，FISCO BCOS 为 bcos-store
- input：跨链服务的输入对象，
- callbackCC ：回调的合约名称
- callbackFcn ：回调的合约方法名称
- timeout：超时时间

其中 input 参数根据跨链服务不同，传入的类型不同，在 ETH 的服务中 input 结构如下:

```
    type Input struct {
    ABIEncoded      string `json:"abi_encoded,omitempty"`
    To              string `json:"to"`
    AmountToMint string `json:"amount_to_mint"`
    MetaID          string `json:"meta_id"`
    SetPrice        string `json:"set_price"`
    IsForSale       bool   `json:"is_for_sale"`
}
```

在 FISCO BCOS 服务中 input 结构如下:

```
    type BcosInput struct {
    Value string `json:"value"`
}
```

调用成功，将返回唯一的请求 ID，请注意保存该值，在回调方法中可以根据该值判断跨链结果。

3. 结果回调:

跨链合约在收到 relayer 返回的跨链交易结果后，将会调用跨链交易时传入的回调合约名以及方法名返回跨链结果信息， 其中调用的第一个参数为返回信息，返回值为 JSON 的字符串，格式为:

```
    type ServiceResponse struct {
    RequestId     string `json:"requestID,omitempty"`
    ErrMsg        string `json:"errMsg,omitempty"`
    Output        string `json:"output,omitempty"`
    IcRequestId string `json:"icRequestID,omitempty"`
}
```

也可以使用 crosschaincode.GetCallBackInfo() 序列化该值，其中 RequestId 为调用跨链合约成功时返回的唯一请求 ID， 可以根据该字段进行相关业务处理。

Output 为跨链结果返回值，该字段为 JSON 格式的字符串，格式如下：

```
    type InputData struct {
  Header interface{} `json:"header"`
  Body interface{} `json:"body"`
}
```

其中 Body 为对应服务的输出对象，

在 ETH 的服务中 output 结构如下:

```
    type Output struct {
  NftID string `json:"nft_id"`
}
```

在 FISCO BCOS 服务中 output 结构如下:

```
    type BcosOutput struct {
  Key string `json:"key"`
}
```

4. 链码打包

由于该链码引用了外部的包，需要在打包时将外部包同时打包，可以使用 govendor 打包

安装 govendor

```
    go get -u -v github.com/kardianos/govendor
```

在项目的 main 方法目录中执行

```
    govendor init
govendor add -tree github.com/BSNDA/ICH/sample/irita/consumers/fabric/crosschaincode
```

最后将项目以及 vendor 目录压缩，在 BSN 门户上传合约包，进行部署。

10.3.3.1.2 示例合约

- GitHub:

  https://github.com/BSNDA/ICH/tree/main/sample/irita/consumers/fabric/chaincode

10.3.3.2 基于 FISCO BCOS 框架的跨链应用合约

使用 Solidity 开发基于 iService 的跨链应用合约。适用于 EVM 兼容的应用链平台，如 Ethereum、FISCO BCOS 等。

10.3.3.2.1 应用合约开发说明

IService Client：为方便开发者，我们提供了 iService Client 合约。iService Client 封装了与 iService 核心代理合约的交互，并实现了事件触发、请求校验、状态维护等功能，可以帮助开发人员快速进行开发。

IService Client 合约在示例合约代码中可以找到。

1. 导入 iService Client

导入本地的 iService Client 合约，例如：

```
import ServiceClient.sol
```

✧*提示：* 也可以直接将 iService Client 合约代码作为消费合约的一部分。

2. 继承 iService Client

```
contract <consuming-contract-name> is iServiceClient {

}
```

3. 实现 iService 调用

- 设置 iService Consumer Proxy 即 iService 核心扩展合约地

址。测试网该合约地址为：

0xdaa3b22adeef09c9416f01db654035ba8c729522

通过调用继承于 iService Client 的方法 setIServiceConsumerProxy(address _iServiceConsumerProxy) 即可。或者在合约构造函数中传入。如下所示：

```
        constructor(
    address _iServiceConsumerProxy
)
    public
{
setIServiceConsumerProxy(_iServiceConsumer);
}
```

- 实现回调方法

当 iService 核心扩展合约接收到跨链调用响应时，将调用消费合约的回调函数回传结果。

回调接口为：

```
            function callback(
    bytes32 _requestID,
    string calldata _output
)
```

- 发起 iService 调用

通过调用继承于 iServiceClient 的 sendIServiceRequest 即可发起跨链服务调用：

```
        bytes32 memory requestID = sendIServiceRequest(
    serviceName,
    requestInput,
```

```
        timeout,
        address(this),
        this.callback.selector
);
```

✧*提示：*开发者进行跨链服务调用时，需要从 iService Market Ex 即部署在应用链上的 iService 市场中获取有关服务的信息，诸如服务名称、服务输入与输出规范等。

4. NFT 服务消费合约示例

NFT 服务由部署在 Ethereum Ropsten 测试网上的 NFT 合约 提供。此合约用于创建 NFT 资产。

NFT 服务的定义如下：

- 服务名称：nft
- 服务输入规范：

```
{
  "type": "object",
  "properties": {
    "to": {
      "description": "address to which the NFT will be minted",
      "type": "string"
    },
    "amount_to_mint": {
      "description": "amount of the NFT to be minted",
      "type": "string"
    },
    "meta_id": {
      "description": "meta id",
      "type": "string"
```

```
        },
        "set_price": {
            "description": "price in Ethereum Wei",
            "type": "string"
        },
        "is_for_sale": {
            "description": "whether or not the minted NFT is for sale",
            "type": "boolean"
        }
    }
}
```


- 服务输出规范：


```
        {
    "type": "object",
    "properties": {
        "nft_id": {
            "description": "id of the minted NFT",
            "type": "string"
        }
    }
}
```


开发者可以在应用链上开发相应的消费合约实现跨链创建 NFT 资产。

10.3.3.2.2 示例合约

- GitHub:

  https://github.com/BSNDA/ICH/tree/main/sample/irita/consumers/fiscobcos/NFTServiceConsumer

11 预言机服务

区块链是一个确定性的、封闭的系统环境，区块链中智能合约产生的交易也需要确定的结果，因此智能合约的宿主虚拟机（VM）禁止在构建交易时加入不确定的外部调用，这个处理机制导致区块链内无法主动获取外部数据，为了解决这个问题，我们引入了预言机。预言机是区块链和现实世界之间的桥梁，通过将现实世界的数据输入到区块链上，将区块链和现实世界连接起来，为智能合约提供与外部世界的连接性。

BSN 的预言机服务的主要功能包括：预言机管理合约管理机制、预言机应用合约功能、预言机链下服务。

BSN 预言机服务将会集成中心化和去中心化预言机机制供链上获取链外数据，开发者可根据自身需求选取不同形式的预言机来实现自己的获取链外数据的需求。

目前 BSN 预言机已在测试网上线，集成了微众基于联盟链的 Truora 解决方案，支持获取随机数和获取汇率两种数据的 API，用户也可自行开发链下 API 作为数据源。欢迎大家在测试网进行体验试用并反馈问题和开发建议，我们也将持续完善功能并集成更多的预言机解决方案。

11.1 基于 Truora 的预言机

11.1.1 概述

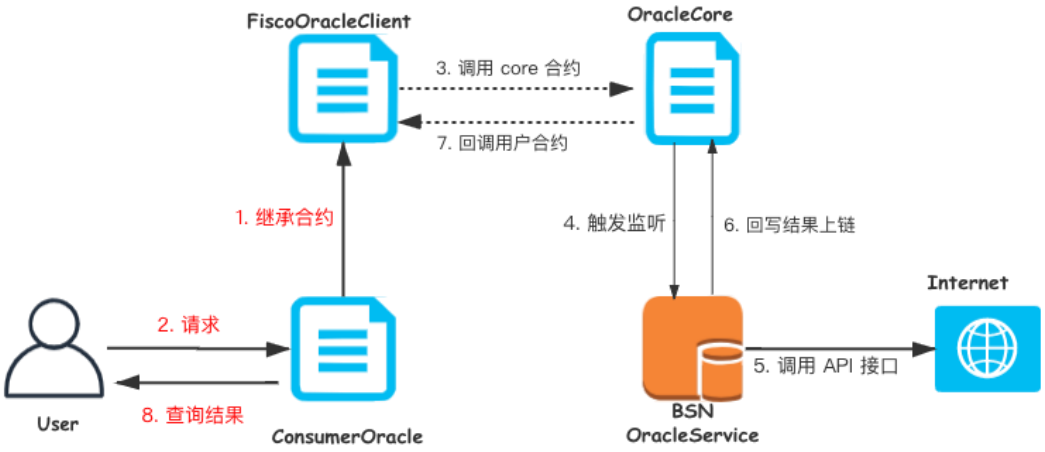
Truora 是 FISCO -BCOS 区块链平台的预言机服务解决方案。是在广泛调研的基础上针对联盟链场景设计的预言机服务。

测试网	框架名称	预言机管理合约名称/地址
中国	FISCO BCOS	0x992ca93808581e88e656831724a813db0d11f059

11.1.2 基于 Truora 的预言机架构及特性

11.1.2.1 基于 Truora 的预言机架构

Truora 预言机主要分为链上部分和链下部分。 链上部分主要是 oracle 相关合约，链下部分主要是 java 服务，负责监听合约的事件，采集结果并回写到 oracle 合约。如下图所示：



11.1.2.2 基于 Truora 的预言机特性

Truora 目前已支持汇率，随机数，天气信息等的外部数据获取，主要特性如下：

- 支持 API 访问链下数据源
- 支持国密
- 支持集群部署
- 支持多链多群组
- 支持多数据格式访问
- 支持请求状态查询

11.1.3 BSN 测试网中的预言机服务开发指南

11.1.3.1 应用合约开发说明

用户可以参考 APISampleOracle.sol 合约实现自己的 oracle 业务合约。

合约解析如下：

- 用户合约需继承 FiscoOracleClient 合约。

```
contract APISampleOracle is FiscoOracleClient
```

- 构造函数需要传入指定的 Truora 服务方地址。

```
constructor(address oracleAddress) public {
        oracleCoreAddress = oracleAddress;
}
```

- 设定自己要访问的 url。修改 url 变量赋值即可。

```
function request() public returns (bytes32 requestId){
        // Set your URL
        // url =
"plain(https://www.random.org/integers/?num=100&min=1&max=100&col=1&base=10&format=plain&rn
d=new)";
        url = "json(https://api.exchangerate-api.com/v4/latest/CNY).rates.JPY";
        bytes32    requestId = oracleQuery(oracleCoreAddress, url, timesAmount);
        validIds[requestId] = true;
        return requestId;
}
```

- 必须实现 callback(bytes32 _requestId, int256 _result) 方法，用于 Truora 预言机回调获取的结果。
- get()方法获取本次请求结果，可自行修改此函数，获取结果后进行自己业务逻辑的计算。

11.1.3.2 链下 API 接口开发说明

目前 Truora 提供了两种链下 API（获取链下随机数、获取人民币对其他币种汇率）供用户调用：

```
//获取链下随机数 API
plain(https://www.random.org/integers/?num=100&min=1&max=100&col=1&base=10&format=plain&rnd=new)
//获取人民币对日元汇率 API
json(https://api.exchangerate-api.com/v4/latest/CNY).rates.JPY
```

如果用户需要自行开发链下 API 接口，需遵循以下规则：

- 链下 API 接口目前支持 json 和 text/plain 两种访问格式。并且链下 API 的 url 必须支持 HTTPS 访问。
- json 格式遵循 jsonpath 格式，子元素 用 "." 表示。
- text/plain 格式默认取第一行结果值。
- 目前预言机返回类型值仅支持 int256 型数据,由于 solidity 不支持浮点数。 会将结果放大 timesAmount 倍， 合约里可以设置 timesAmount 参数， 默认值是 10*18。

11.1.3.3 示例合约

➢ GitHub:

https://github.com/WeBankBlockchain/Truora-Service/tree/bsn/contracts/1.0/sol-0.4/oracle

12 账户管理

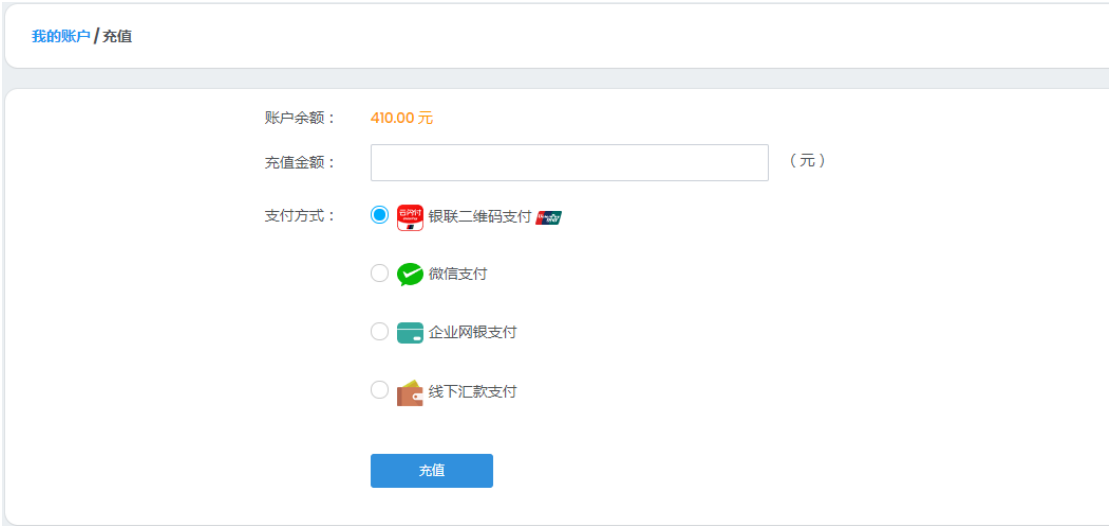
12.1 我的账户

1. 充值

服务发布、产品上架及其他各类操作引发的扣款行为都将从我的账户中支出，用户须通过充值来保证账户中有充足的余额。

充值操作步骤如下：

- 在【用户中心->我的账户】中点击【充值】进入充值页面，如下：



- 输入充值金额，并选择支付方式，点击【充值】进行付款。付款完成之后即可完成充值。

注意：

- 若用户选择线下汇款，则需要将资金转入指定的账户中。汇款时需要在所有可以输入备注的地方输入汇款识别码。线下汇款到账日期为 1-3 个工作日，若超过 3 个工作日未到账，请联系客服（客服电话：400-071-8215，工作日 08:00 - 17:30）。
- 个人用户不支持用线下汇款的方式进行充值。

- 如果是企业用户，请务必使用企业银行账户进行充值。如果使用个人银行账户，则在开对公发票时比较麻烦，需要个人提供证明文件。

2. 提现

用户可以将账户内的余额提现。操作步骤如下：

- 在【个人中心->我的账户】中点击【提现】，显示如下窗口：

- 输入本次提现金额，并点击【申请提现】进行提现。申请时将扣除账户中的余额，并需要等待运营人员审核；
- 审核通过后，运营人员将资金转入用户的账户中完成提现；
- 提现的资金按照充值时的路径原路返回。账户内因为优惠券等活动产生的金额不允许提现。

3. 退款

按月支付的账单和流量使用账单不允许退款，按年支付的账单且剩余账期超过 1 个月时，如确需退款请联系客服（客服电话：400-071-8215，工作日 08:00 - 17:30）。

4. 开发票

详见 12.3 我的发票

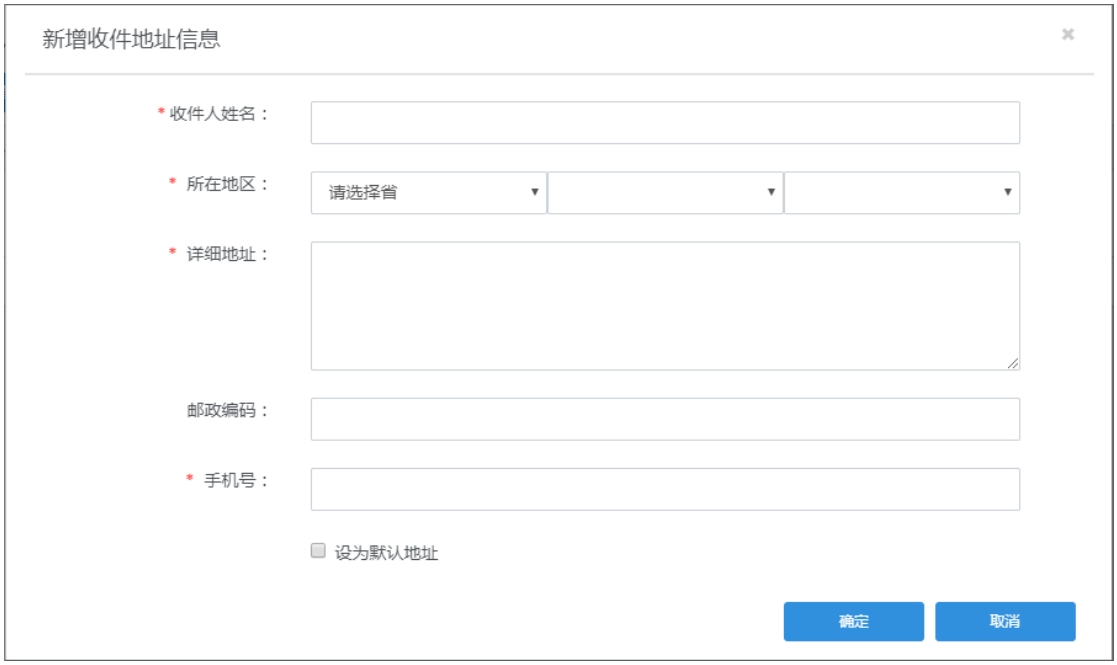
5. 管理收件地址

区块链服务网络目前仅支持开纸质发票，本模块用于管理发票的收件地址。

操作步骤如下：

- 在【用户中心->我的账户】中点击【管理收件地址】进入收件地址管理页面，如下：

- 点击【新增收件地址】显示收件地址新增窗口，如下：



- 根据要求输入收件地址信息，并点击【确定】完成新增。

6. 管理发票抬头

用于管理发票的抬头信息，操作步骤如下：

- 在【用户中心->我的账户】中点击【发票抬头管理】进入发票抬

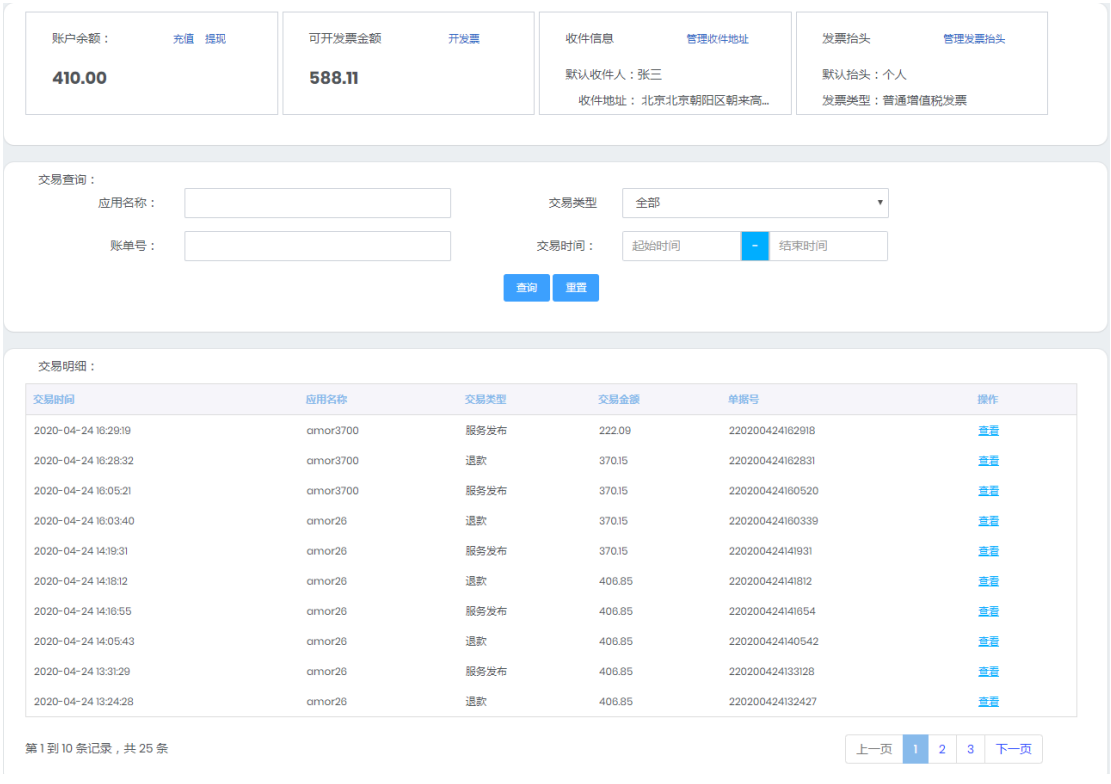
头管理页面，如下：

- 点击新增【发票抬头】显示发票抬头新增窗口，如下：

- 选择抬头类型，并输入相应的发票抬头信息，点击【确定】完成发票抬头新增。

7. 查看交易明细

用户可以在【用户中心->我的账户】中查看用户账户的资金变动明细，包括：充值、提现、退款、服务发布、服务配置升级、服务周期扣款、产品上架、产品周期扣款、流量使用费等。如下图：



12.2 我的账单

用于显示应用产品、应用服务、IPFS 服务、跨链服务以及专有节点服务产生的账单信息。

应用服务产生的费用主要包括资源使用费和数据流量使用费。其中资源使用费为先付费模式，计费要素包括 TPS、存储量、记账节点数；数据流量使用费为后付费模式，计费要素为流量的实际使用量。

应用产品产生的费用主要为产品上架费，为先付费模式。

IPFS 服务产生的费用主要包括资源使用费和数据流量使用费。资源使用费为先付费模式，按照容量计费；数据流量使用费为后付费模式，按流量的实际使用量计费。

跨链服务产生的费用按跨链请求数计费，为后付费模式。

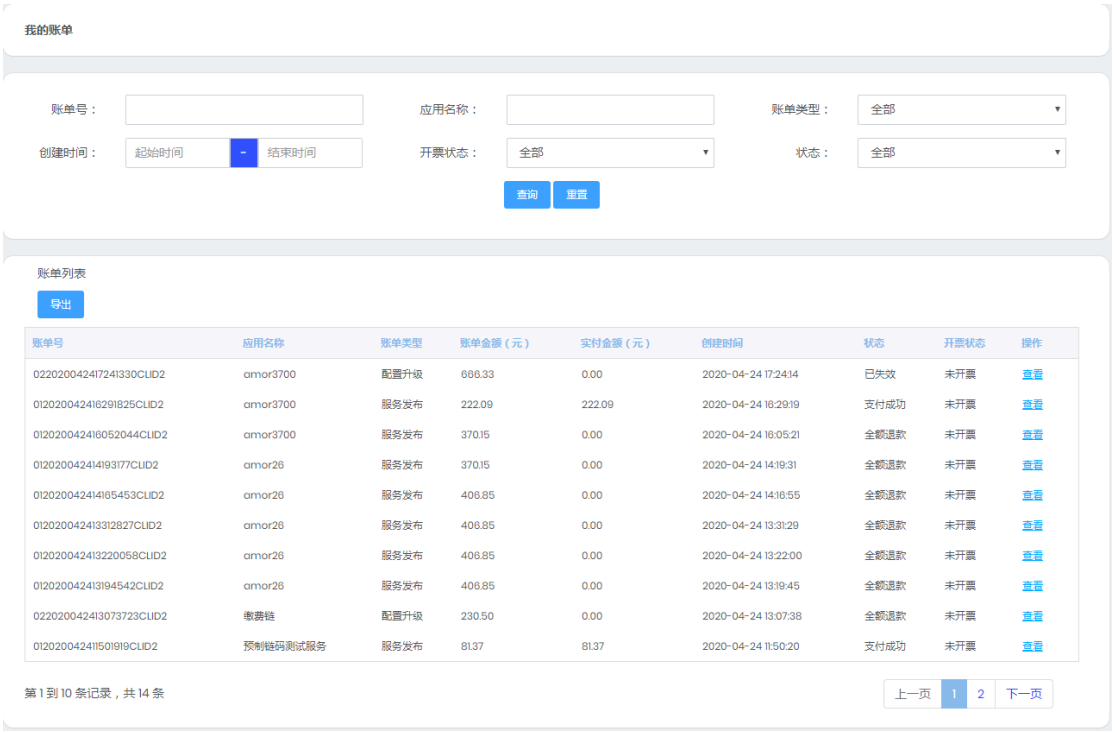
专有节点服务产生的费用主要包括资源费用费和数据流量费。

资源使用费为先付费模式，计费要素包括云平台、底层框架、区域、节点数量、主机配置以及磁盘容量；数据流量费使用费为后付费模式，计费要素为流量的实际使用量。

上述费用产生时都将生成账单，包括：

- **服务发布：**服务申请发布时支付；金额根据 TPS、存储量、记账节点计算；
- **服务配置升级：**配置升级时支付，为新配置和旧配置在账单周期剩余时间内的差额；
- **服务周期扣款：**服务发布之后，系统按年或按月进行周期性扣款；
- **数据流量周期扣款：**服务发布之后，系统根据服务的实际流量使用情况按周进行扣款；
- **产品上架：**产品申请上架时进行支付；

- **产品周期扣款：**产品上架之后，系统按月进行周期性扣款。
- **IPFS 服务开通：**服务开通时支付；金额根据容量计算；
- **IPFS 服务升级：**服务升级时支付，为新配置和旧配置在账单周期剩余时间内的差额；
- **IPFS 服务周期：**服务开通之后，系统按年或按月进行周期性扣款；
- **IPFS 流量周期：**系统根据文件下载的实际流量使用情况按周进行扣款；
- **跨链服务周期：**系统根据实际的跨链次数按周进行扣款；
- **专有节点服务开通：**服务开通时支付，金额根据云平台、底层框架、区域、节点数量、主机配置以及磁盘容量计算；
- **专有节点服务周期：**服务开通后，系统按年或按月进行周期性扣款；
- **专有节点流量周期：**系统根据实际流量使用情况按周进行扣款；

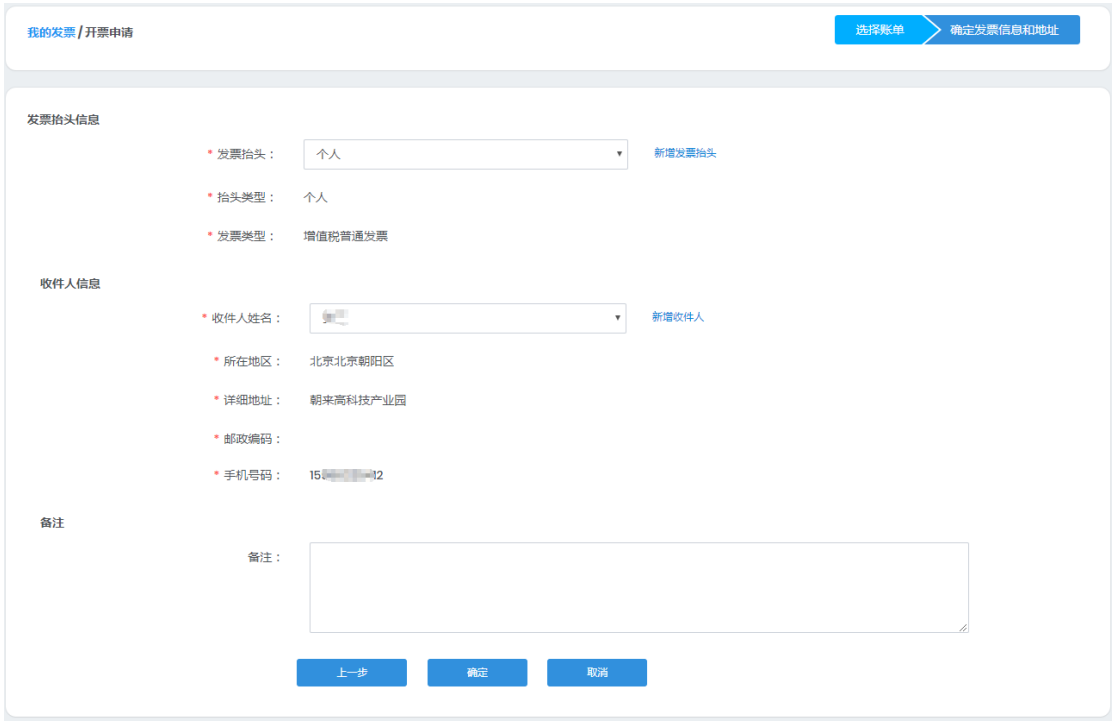


12.3 我的发票

用户可以向区块链服务网络运营方申请开发票，操作步骤如下：

- 在【用户中心->我的发票】页面点击【新增发票】进入账单选择页面，如下：

- 用户选择需要开票的账单，系统根据所选的账单的实付金额计算开票金额。点击【下一步】进入发票信息页面，如下：



- 选择发票抬头和发票收件人，并点击【确定】提交开票申请；
- 开票申请提交之后需要等待运营人员审核，待审核通过之后，运营人员将开具纸质、电子发票，并邮寄、发送给开票申请人。

13 在线文档

BSN 中国官网：

白皮书			
名称	版本号	更新时间	操作
区块链服务网络基础白皮书	V1.0.5	2020-02-05	PDF
区块链服务网络技术白皮书	V1.0.0	2020-04-25	PDF

文档资料			
名称	版本	更新日期	操作
BSN 官网在线帮助	1.4.0	2021-01-31	Online PDF
Hyperledger Fabric 示例	1.0.0	2020-07-31	Github
FISCO BCOS 示例	1.0.0	2020-07-31	Github
XuperChain 示例	1.0.0	2020-07-31	Github
城市节点网关 SDK 实例	1.0.0	2020-07-31	Github

我们邀请对 BSN 感兴趣，并且有经验的开发者，来共同优化上述的 SDK 和示例包。如果您愿意参加，请在 GitHub 里联系我们。

联盟链框架		
名称	官网地址	操作
Hyperledger Fabric	https://www.hyperledger.org/	Github Documentation
FISCO BCOS	http://fisco-bcos.org/	Github Documentation
XuperChain	https://xchain.baidu.com/	Github Documentation
CITA	https://www.citahub.com	Github Documentation
ConsenSys Quorum	https://consensys.net/quorum/	Github Documentation

BSN 国际官网：

White Papers			
Name	Version	Update	Details
BSN Introduction White paper	V1.05	February 5th,2020	PDF
BSN Technical White Paper	V1.0.0	April 25th,2020	PDF

Site Documents			
Name	Version	Update	Details
Online Help	1.4.0	January 31st,2021	Online PDF
Fabric Examples	1.0.1	April 24th,2020	Github

FISCO BCOS Examples	1.0.1	April 24th,2020	Github
SDK Examples	1.0.1	April 24th,2020	Github

Permissioned Frameworks		
Name	**Official Website**	**Details**
Hyperledger Fabric	https://www.hyperledger.org/	Github Documentation
FISCO BCOS	http://fisco-bcos.org/	Github Documentation

Public Chains		
Name	**Official Website**	**Details**
Nervos	https://www.nervos.org/	Github Documentation
NEO	https://neo.org/	Github Documentation
ETH	https://ethereum.org/	Github Documentation
Tezos	https://tezoscommons.org/	Github Documentation
EOS	https://eos.io/	Github Documentation
IRISNET	https://www.irisnet.org/	Github Documentation
dfuse-eos	https://dfuse.io/en/home/?utm_source=BSN	Github Documentation
Algorand	https://algorand.foundation/	Github Documentation
Solana	https://solana.com/	Github Documentation
ShareRing	https://sharering.network/	Github Documentation
BTY	https://www.bityuan.com/index	Github Documentation
Oasis Network	http://www.oasisprotocol.org	Github Documentation
Polkadot	https://polkadot.network/	Github Documentation

14 联系我们

如果您有任何问题，或发现本手册有任何错误，请联系我们：

客服电话： 400-071-8215（工作日：08:00 - 17:30）

邮箱：support@bsnbase.com

客服二维码：

BSN 公众号：

国际社交媒体号：