区块链服务网络用户手册

（Blockchain-based Service Network User Manual）

Version 1.2.0

区块链服务网络发展联盟

目录

1 BSN 介绍 1
1.1 简介 1
1.2 BSN 服务 2
1.2.1 许可链服务（Permissioned Services） 2
1.2.2 公有链服务（Permissionless Services） 3
1.2.3 跨链服务（Interchain Services） 3
1.3 名词解释 4
2 更新说明 6
3 快速入门 8
3.1 联盟链 8
3.2 教学视频 9
3.3 文档 9
4 账户注册 10
4.1 BSN 官方门户 10
4.2 BSN 国际门户 11
5 许可链服务 13
5.1 概述 13
5.2 BSN 密钥证书体系 14
5.3 应用服务发布和参与 18
5.3.1 应用服务管理 18
5.3.2 应用服务参与 30
5.3.3 证书下载和更新 34
5.3.4 应用产品管理 35
5.4 BSN 外系统接入指南 37
5.4.1 概述 37
5.4.2 智能合约打包规范 40
5.4.3 城市节点网关 Fabric API 49
5.4.4 城市节点网关 FISCO BCOS API 74
5.4.5 城市节点网关 XuperChain API 103
5.5 开发 SDK 及示例 114
6 公有链服务 116
6.1 概述 116
6.2 选择服务套餐 116
6.3 创建和管理项目 121
6.4 BSN 外系统接入指南 123
6.4.1 概述 123

6.4.2 Ethereum 124
6.4.3 EOSIO 125
6.4.4 Nervos CKB 125
6.4.5 Neo 126
6.4.6 Tezos 126
6.4.7 IRISnet 127
6.4.8 EOSIO-Dfuse 127
7 跨链服务 129
8 账户管理 130
8.1 我的账户 130
8.2 我的账单 134
8.3 我的发票 135
9 在线文档 137
10 联系我们 139

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（global.bsnbase.com）使用。但在本手册内，仍然给出了公有链服务的使用说明。BSN 的跨链服务的技术较复杂，目前还在研发中。推出后我们会在手册内补齐相关内容。所以本手册的主要内容为 BSN 的许可链服务。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。

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。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。

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(Global.bsnbase.com)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。

目前 BSN 开发团队正在建立 BSN 的“Interchain Communications Hub”（跨链通讯枢纽）。枢纽内会提供多家的跨链机制和预言机机制。预计将在 2020 年下半年推出演示版本，初步将包括基于 IRITA 的跨链服务和基于 Chainlink 的预言机服务。

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或发布方自己的业务系统直接申请参与使用。
- **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。

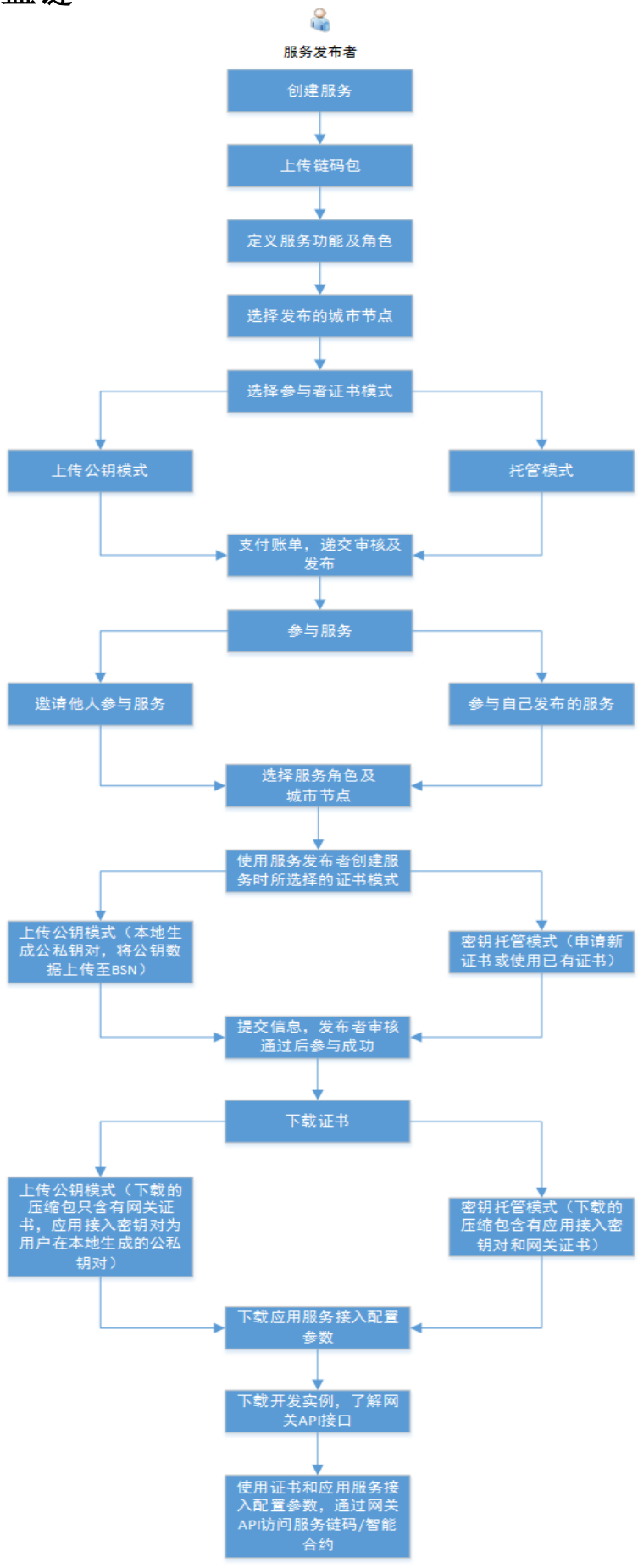
2 更新说明

发布时间	版本号	更新内容
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 global.bsnbase.com 发布。

3 快速入门

3.1 联盟链


服务发布者
创建服务
上传链码包
定义服务功能及角色
选择发布的城市节点
选择参与者证书模式
上传公钥模式
托管模式
支付账单，递交审核及发布
参与服务
邀请他人参与服务
参与自己发布的服务
选择服务角色及城市节点
使用服务发布者创建服务时所选择的证书模式
上传公钥模式（本地生成公私钥对，将公钥数据上传至BSN）
密钥托管模式（申请新证书或使用已有证书）
提交信息，发布者审核通过后参与成功
下载证书
上传公钥模式（下载的压缩包只含有网关证书，应用接入密钥对为用户在本地生成的公私钥对）
密钥托管模式（下载的压缩包含有应用接入密钥对和网关证书）
下载应用服务接入配置参数
下载开发实例，了解网关API接口
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

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。

3.2 教学视频

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

3.3 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，请访问第九章《在线文档》。

4 账户注册

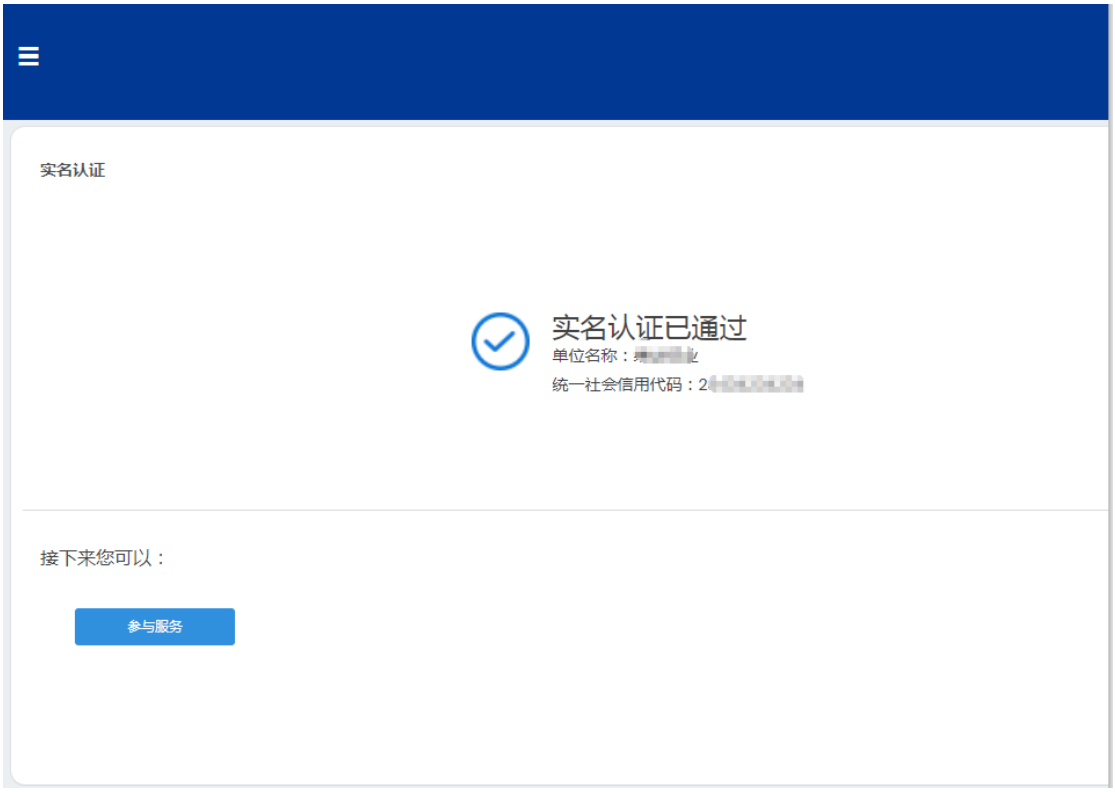
4.1 BSN 官方门户

- 打开网络地址：www.bsnbase.com；点击登录页面中的【注册申请】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。如果在 BSN 上发布应用的费用需要开具企业发票，则建议注册企业用户。
- 注册申请提交后需等待系统审核，待系统审核通过后将向申请者发送激活邮件，申请用户可根据**激活邮件**中的激活链接来激活账

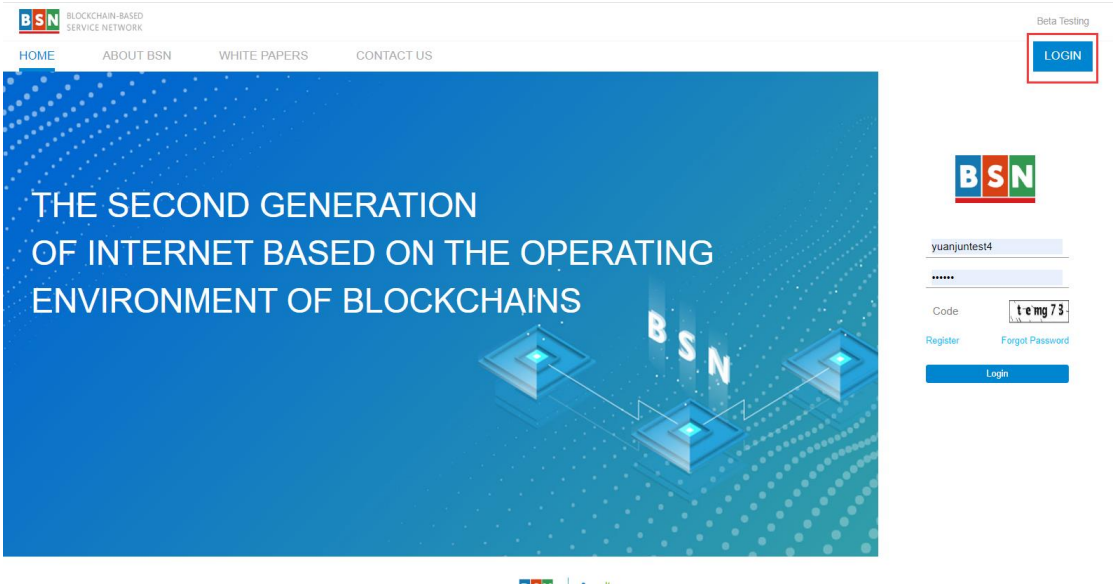
户；

- 帐户激活后即可登录系统使用。
- 注册申请审核通过的用户默认为**实名认证已通过**，用户登录区块链服务网络门户后，可在【用户中心】->【实名认证】页面查看实名认证信息。

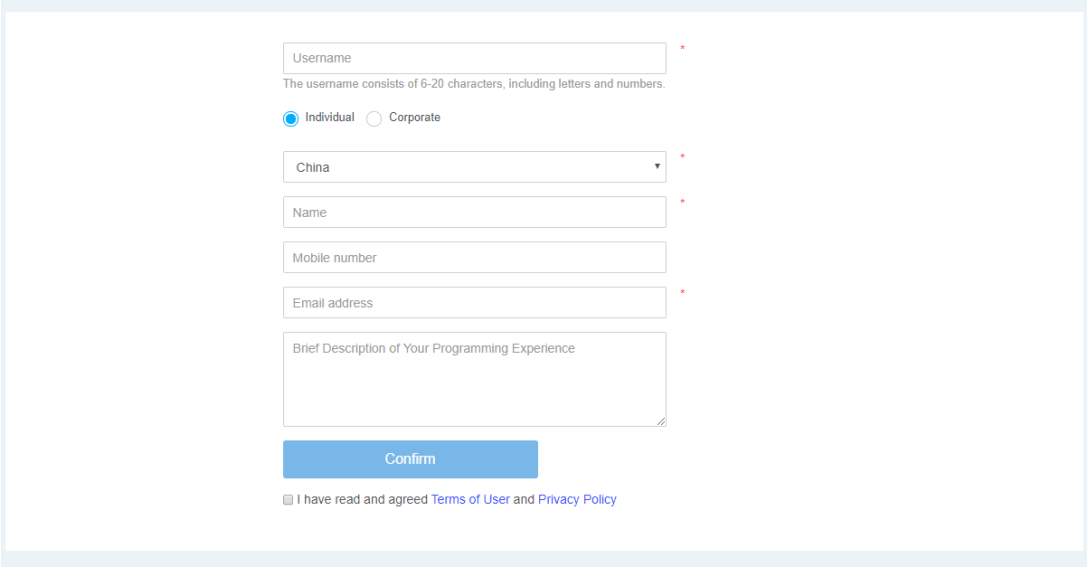


4.2 BSN 国际门户

- 打开网络地址：https://global.bsnbase.com/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。

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。

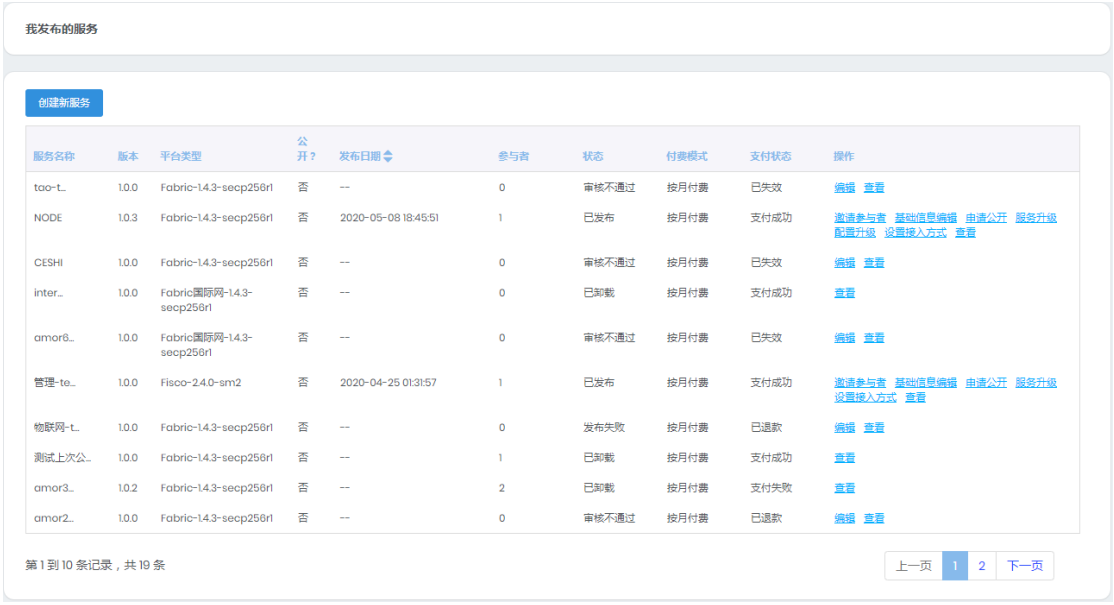
5.3 应用服务发布和参与

5.3.1 应用服务管理

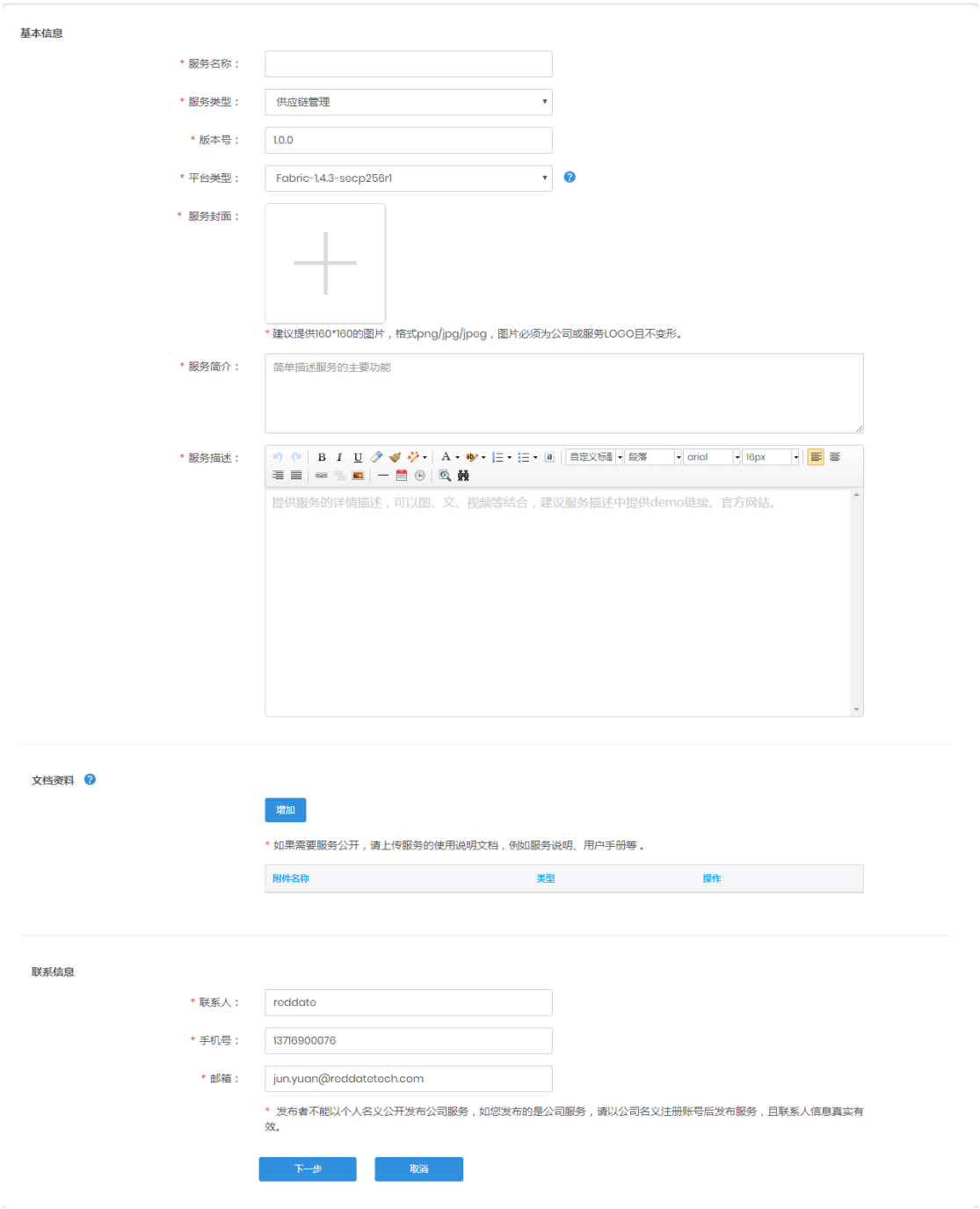
5.3.1.1 应用服务发布

服务发布的操作步骤如下：

- 创建新服务
  - 点击【我发布的应用】->【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。



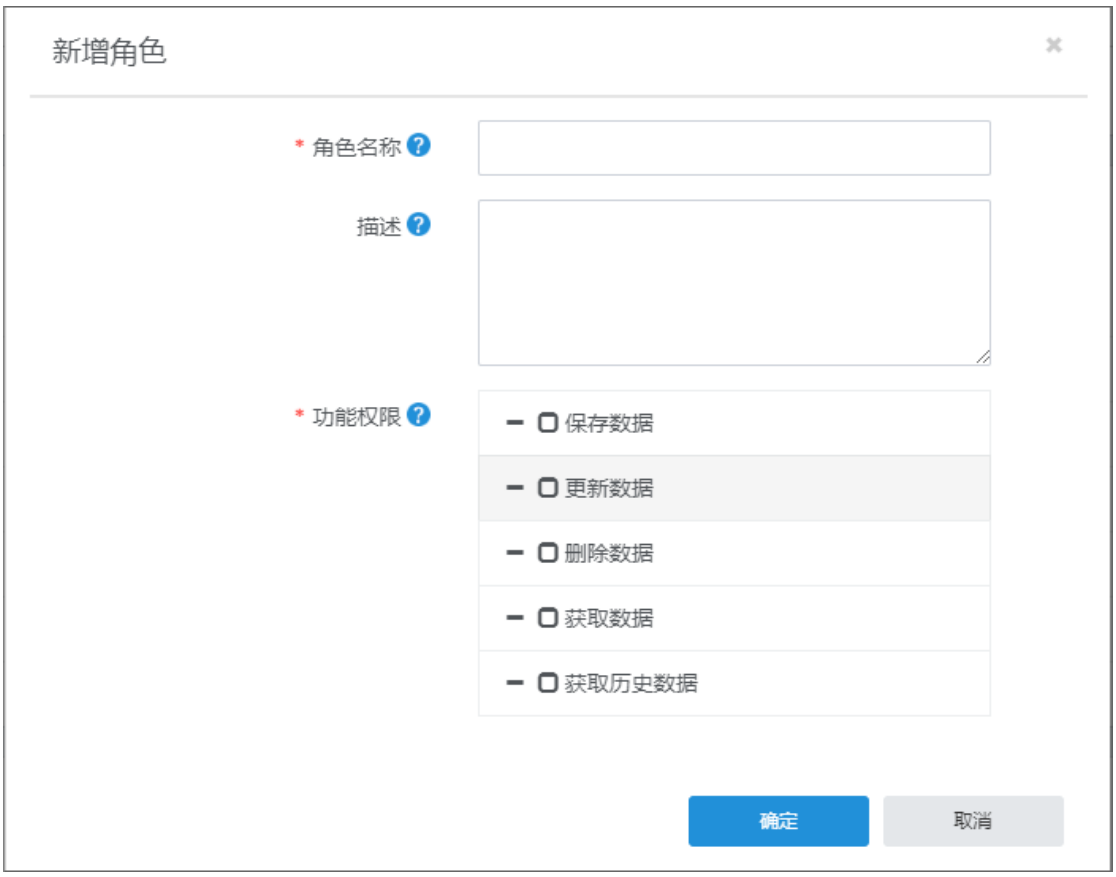
- 上传链码包
  - 一个服务可以上传多个链码包，用户可在上图中点击【下一步】开始上传链码包，如下图：

- 根据页面进行输入，并点击【确定】来完成链码包的上传。

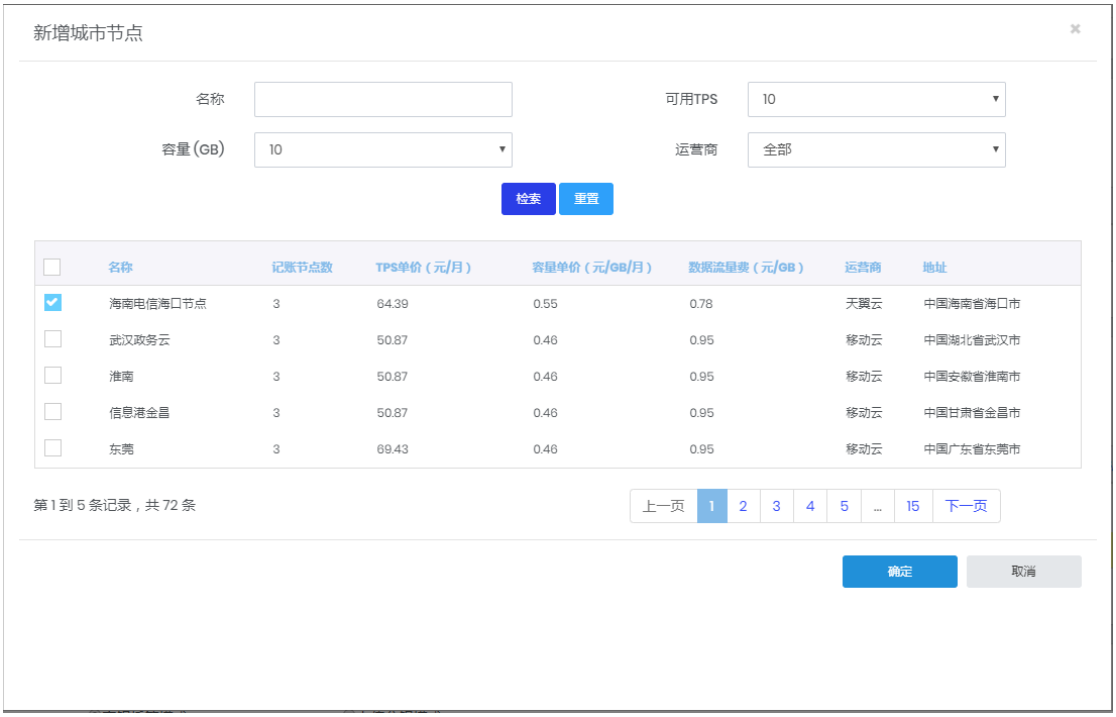
➢ 定义服务功能及角色

- 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：

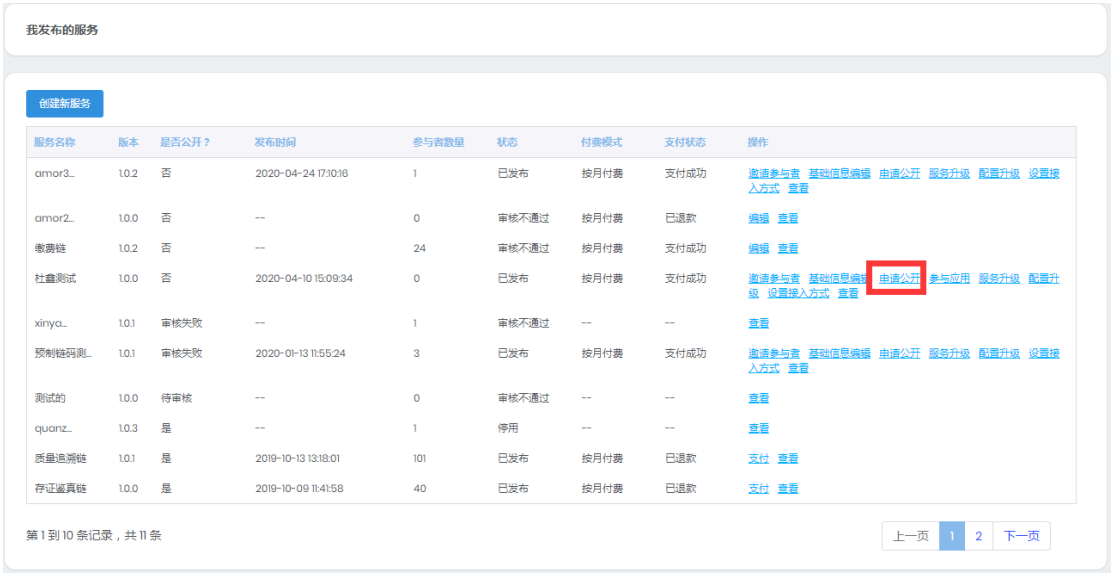
- 进入【我发布的应用->我发布的服务】页面，选择状态为【已发布】的服务，点击【邀请参与者】，显示如下：

- 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【首页】->【我发布的应用】->【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【我发布的应用】->【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

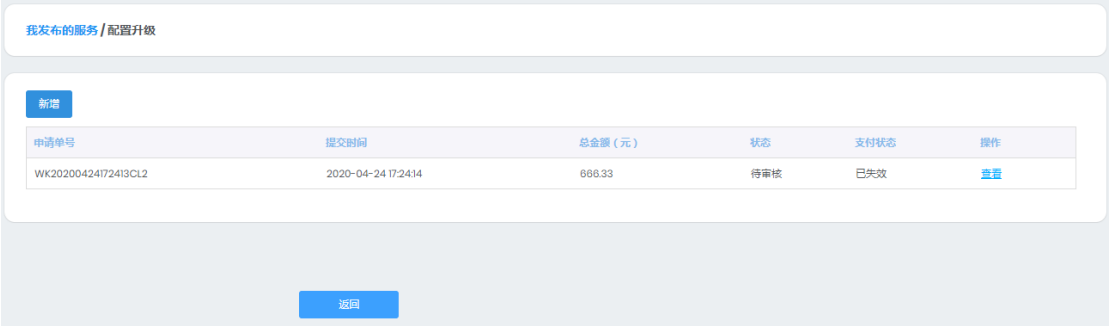
发布者可以通过使用【配置升级】功能来升级服务的城市节点及城市节点中的 TPS、容量和记账节点。配置升级时需要支付相应的资源升级费用。

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，

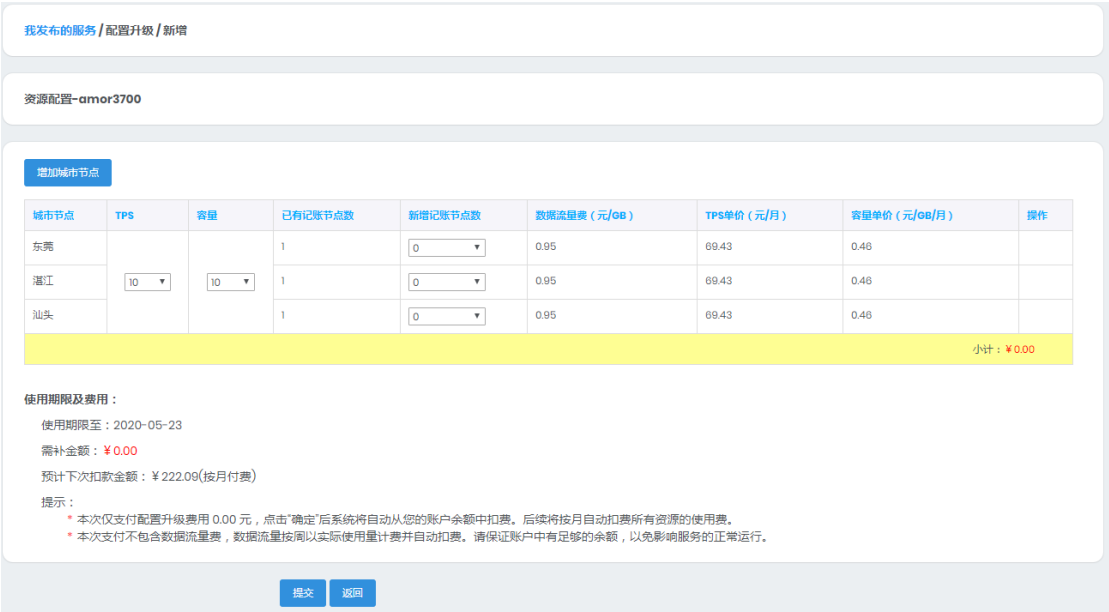
只能在原配置的基础上升级。

配置升级的操作步骤如下：

- 在【我发布的应用->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；

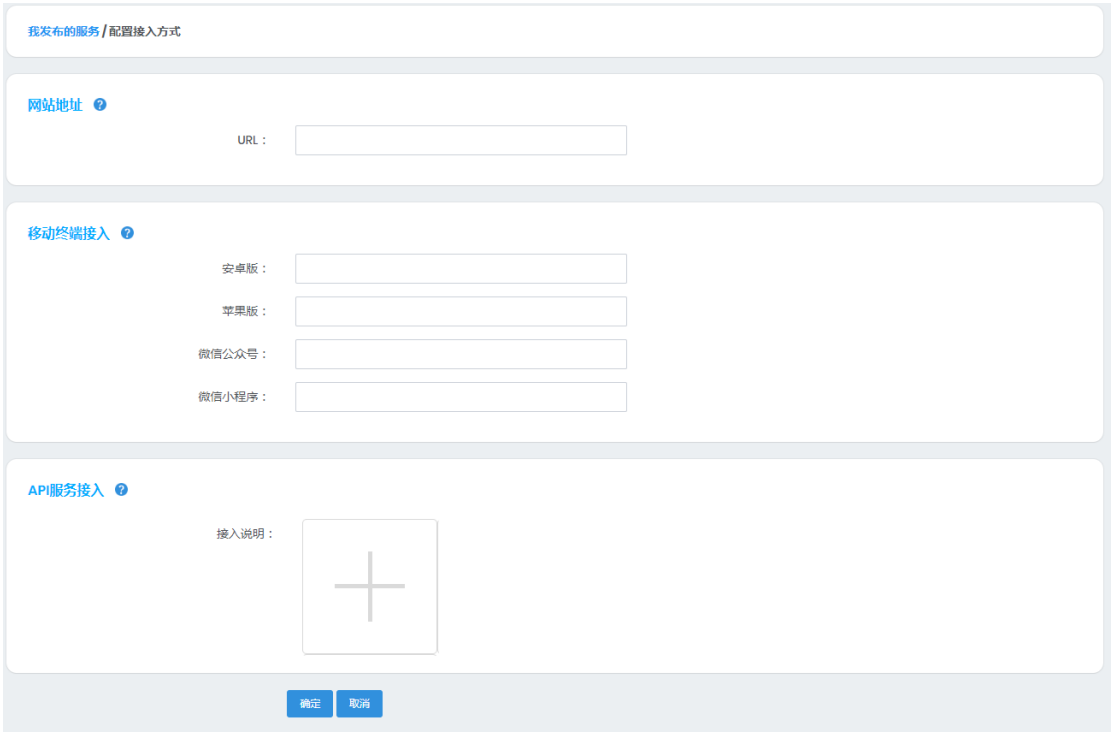
注意：配置升级时支付的费用为升级费用，即补足升级前配置和升级后配置在账单周期内剩余时间的差额。升级成功以后，下个周期起将按新配置进行周期扣款。

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【我发布的应用->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

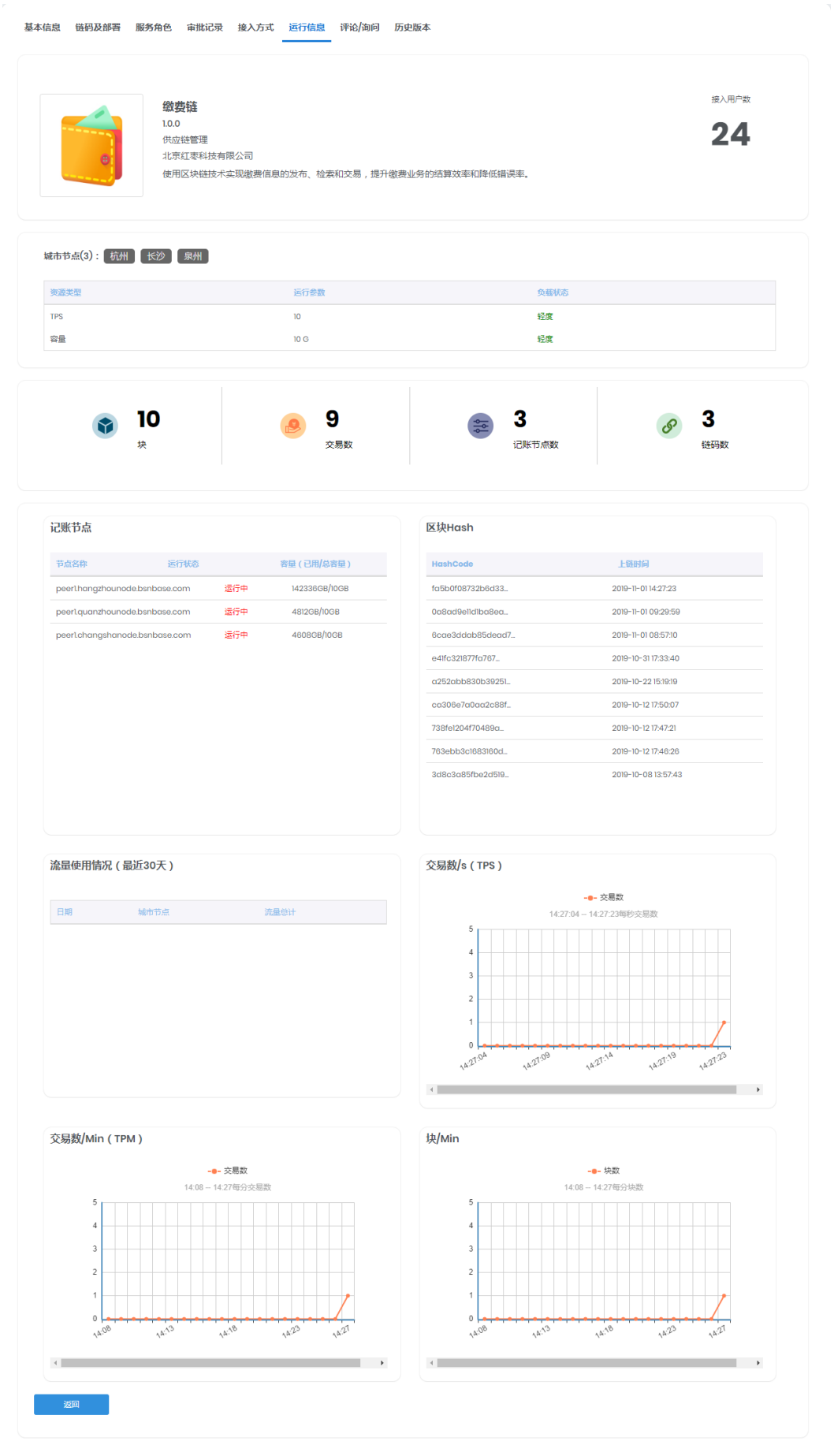
服务发布成功之后，系统将根据支付周期为服务生成资源使用账单，并根据服务实际使用的流量情况按周生成流量使用账单。当账单生成时，系统将从用户的账户中自动扣款，若扣款失败且账单超过 72 小时未支付则系统将自动停用服务（在服务的停用期间，由于其仍然占用服务资源，系统仍会按周期生成资源使用账单）。服务停用以后，链上数据无法访问。

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【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
14:27:04 – 14:27:23每秒交易数
交易数/Min（TPM）
交易数
14:08 – 14:27每分交易数
块/Min
块数
14:08 – 14:27每分块数
返回

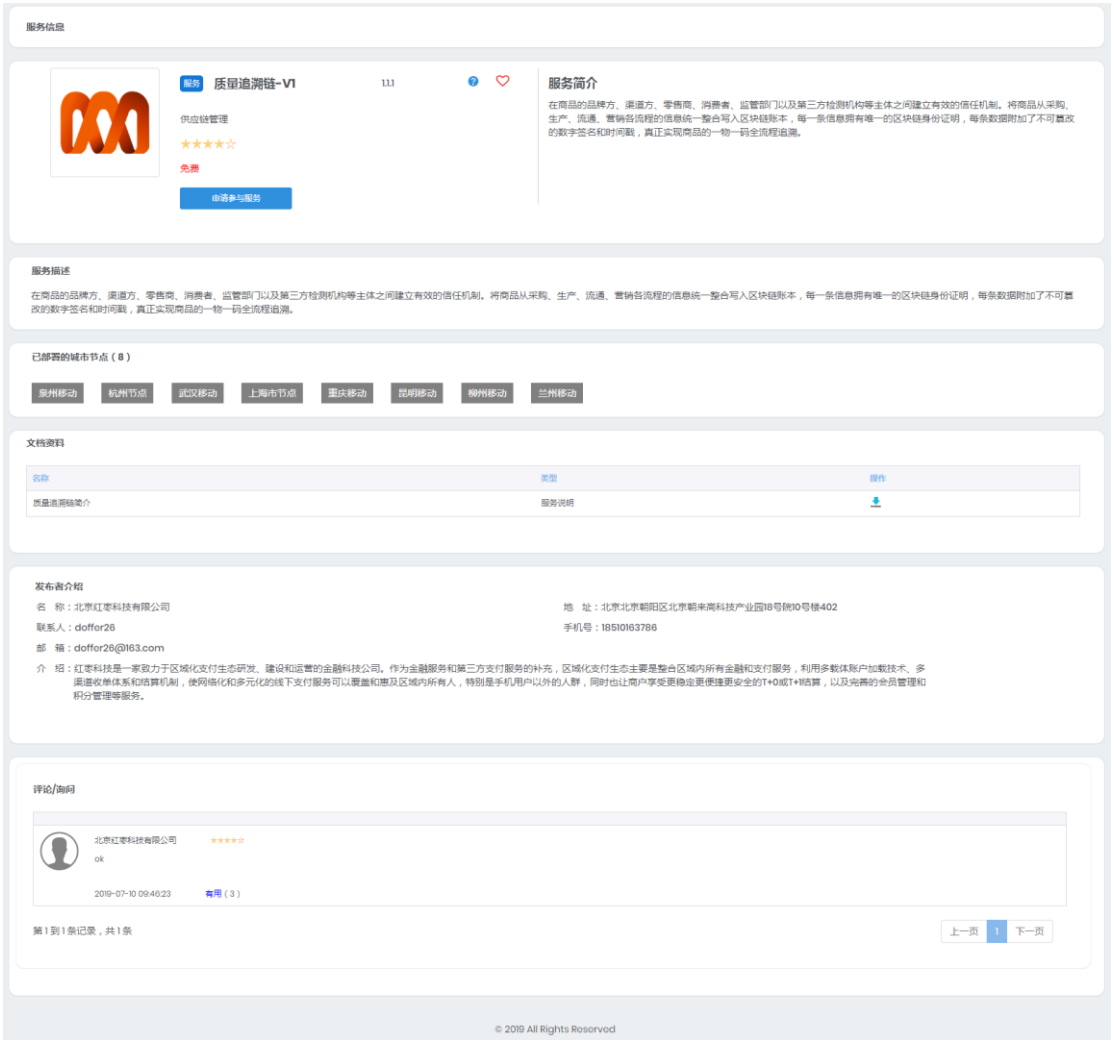
5.3.2 应用服务参与

5.3.2.1 应用服务参与

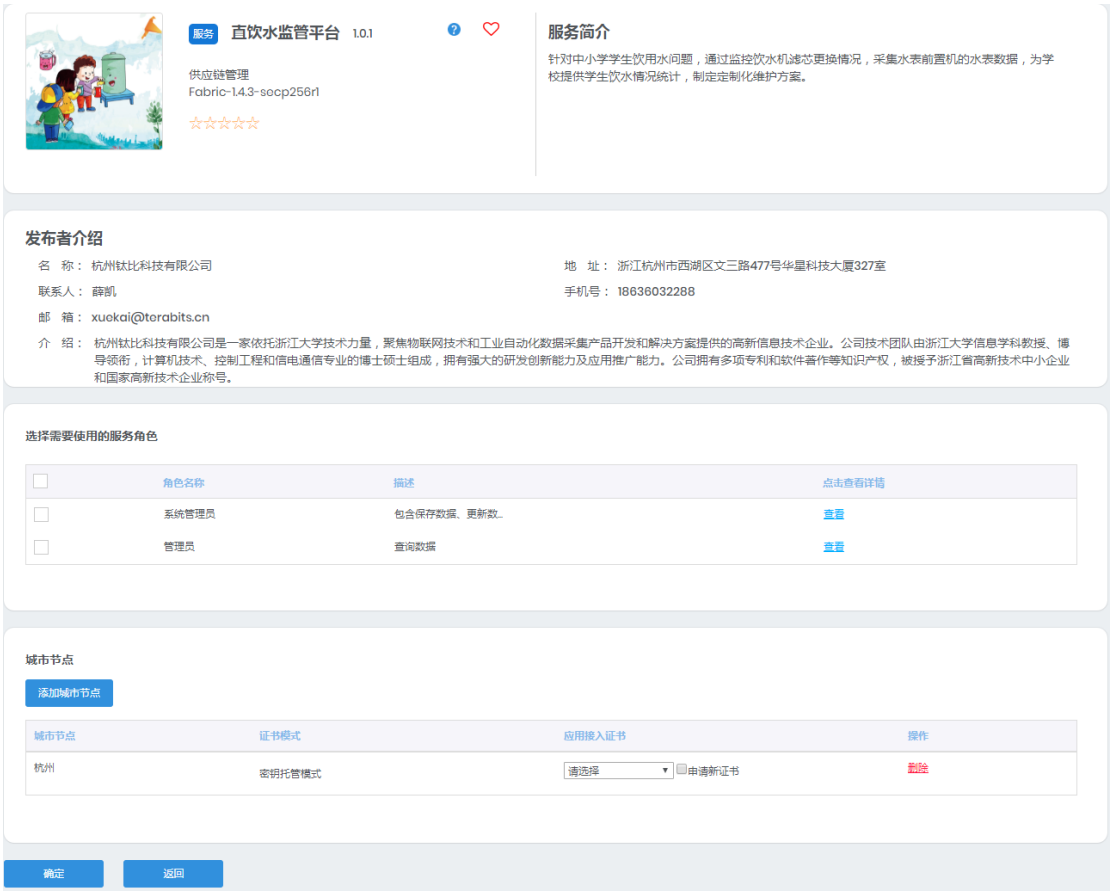
1、 应用服务参与

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【首页】->【我参与的应用】->【我参与的】页面或【应用商店】，选择一个未参与的应用去参与：

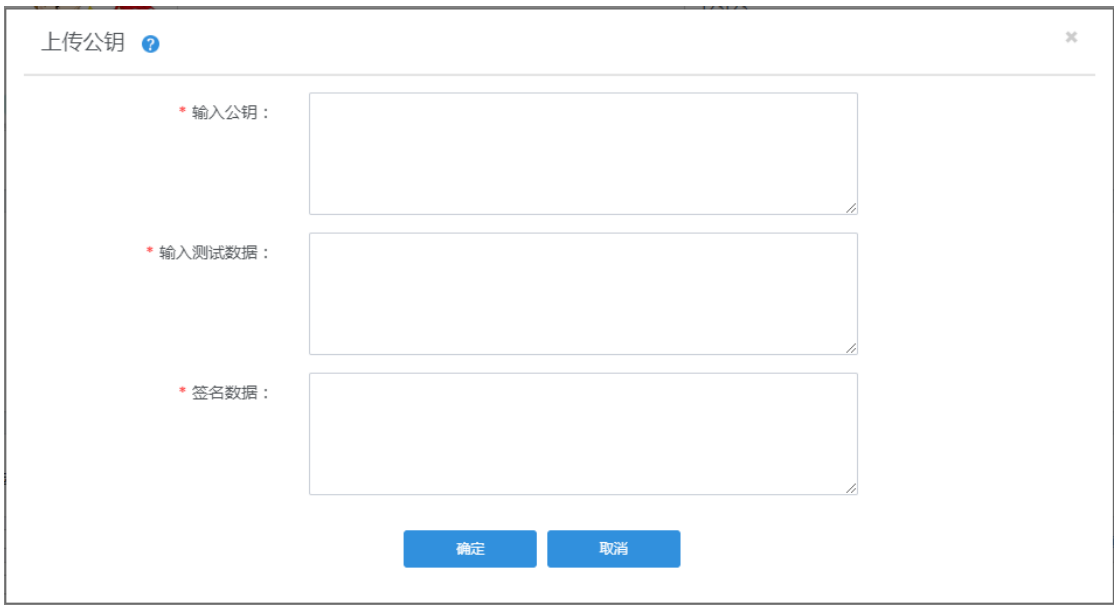


- 点击【申请参与服务】，进入选择应用角色及城市节点页面：



- 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；



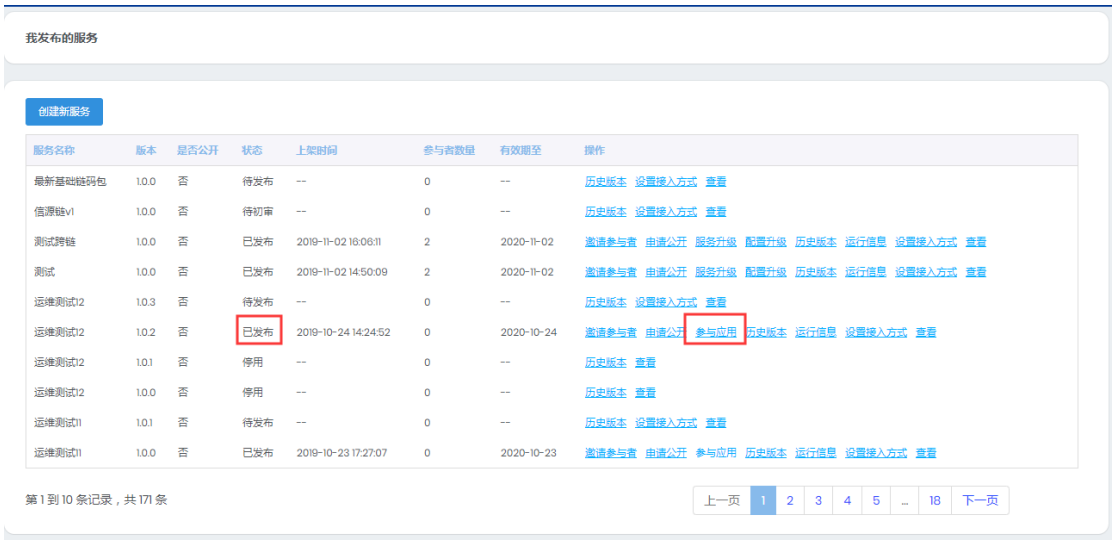
密钥托管模式：可以选择该城市节点上已有的证书或者申请一个新的证书。

- 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

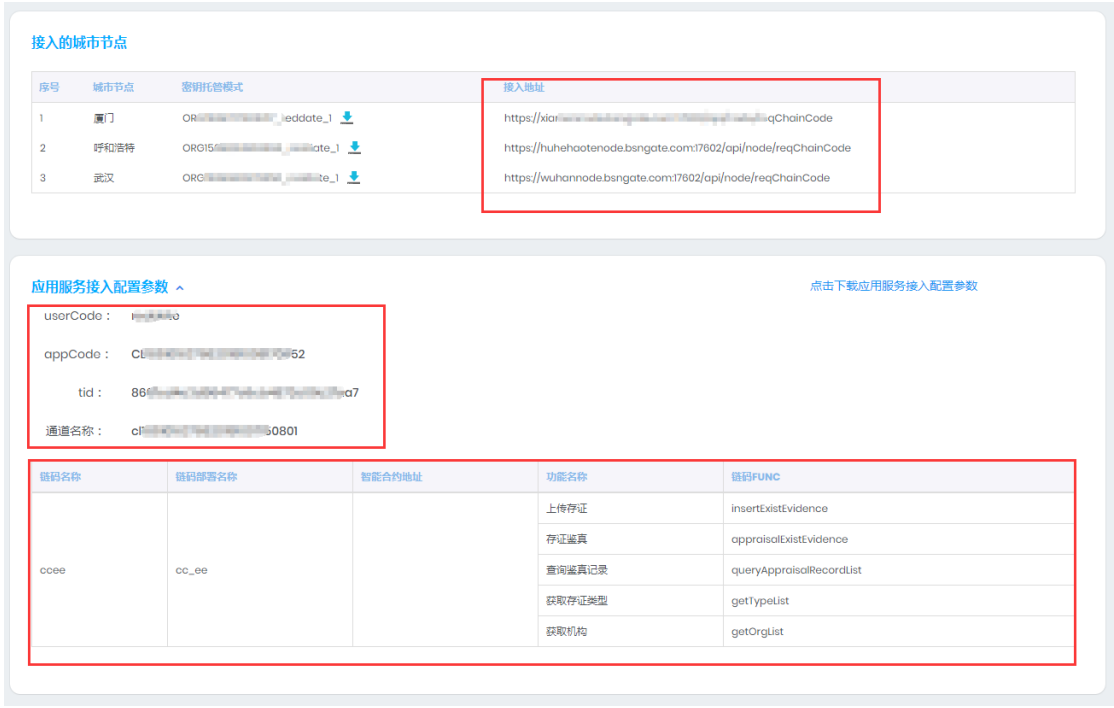
服务发布后，发布者默认不参与自己发布的服务，如需参与，则可以在【我发布的服务】列表中点击【参与应用】来参与自己发布的服务（具体步骤同上述应用服务参与步骤），如下图：



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

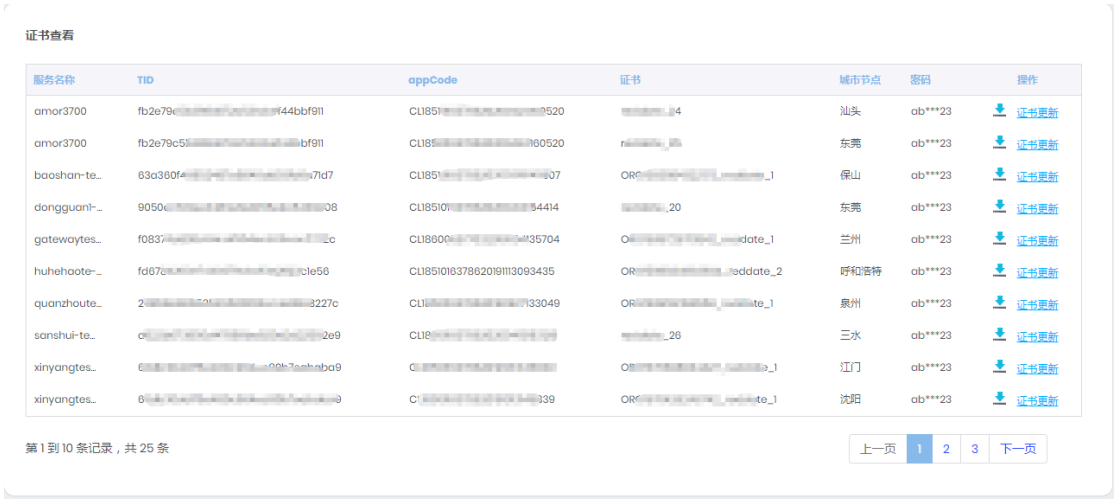


5.3.3 证书下载和更新

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【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：

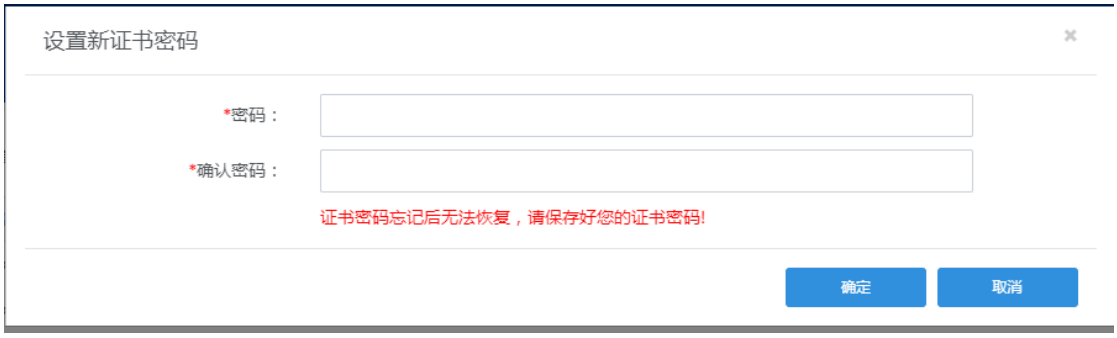


- 点击证书下载，并输入证书生成时设置的密码即可下载证书。

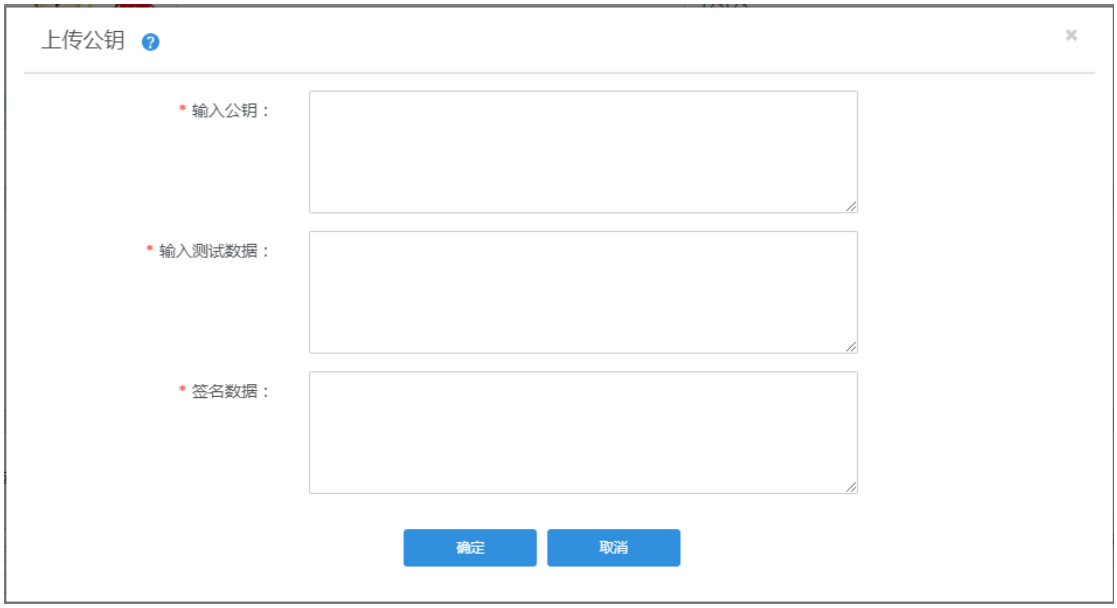
注意：由于使用上传公钥模式参与服务时，用户只需上传公钥，私钥由用户自行保管，所以无须下载证书。

2、 证书更新步骤如下：

- 点击【我的身份证书->我托管的证书】或【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

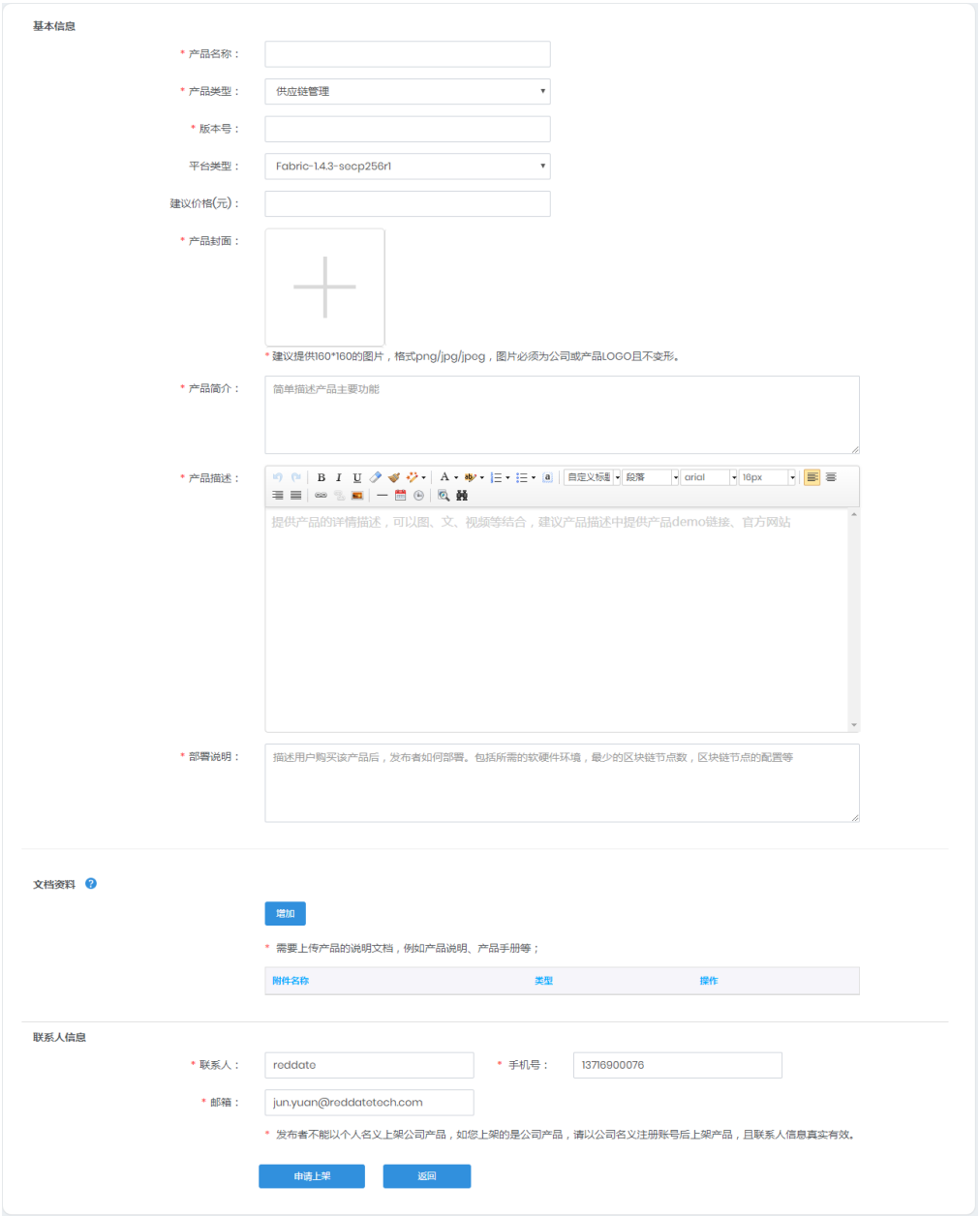
应用产品指没有在区块链服务网络中发布和运行，仅在服务网络门户的应用商店内上架的区块链应用。产品销售后，由购买方在服务网络上发布后方可以投入使用。产品发布完成之后即成为服务，此时购买方成为应用发布者，并可以邀请其他用户参与。产品的设置是让某些开发者仅以开发和销售产品为商业模式，而不需要投入成本去运营自己的产品（对比互联网，产品开发者类似提供网站搭建的开发者，而服务发布者类似经营网站的运营者）。

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【我发布的应用】->【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。

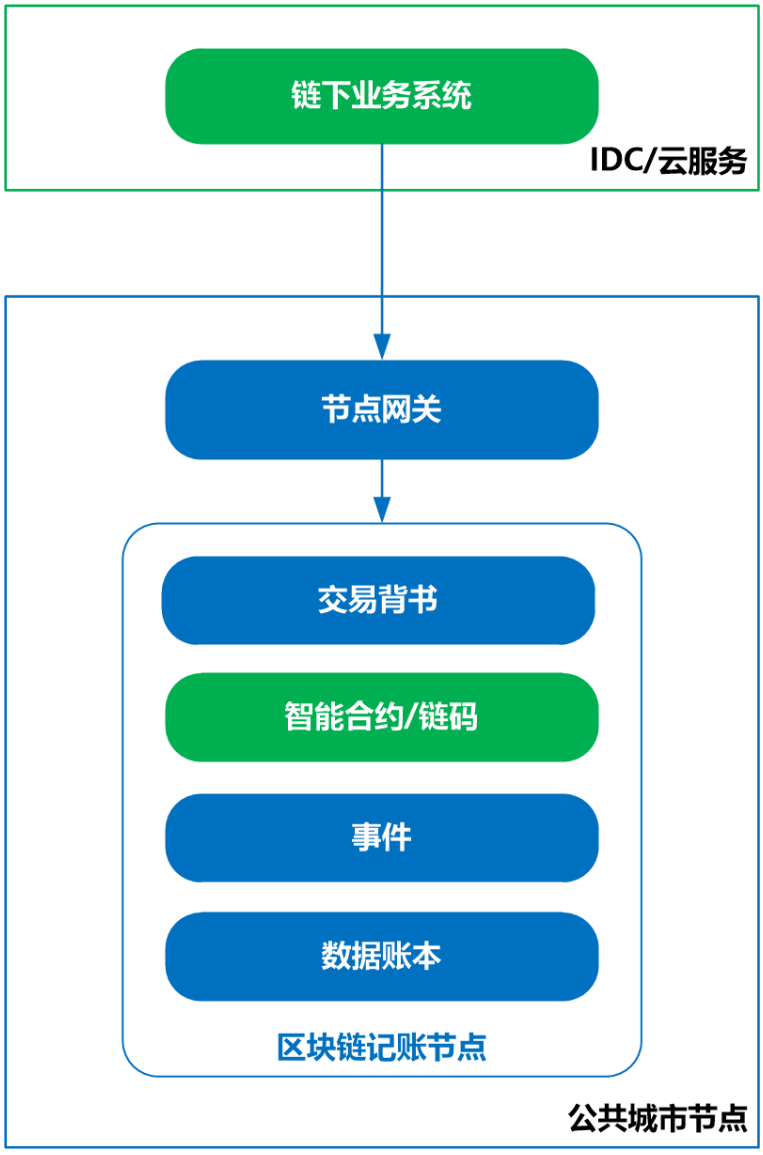
5.4 BSN 外系统接入指南

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）。

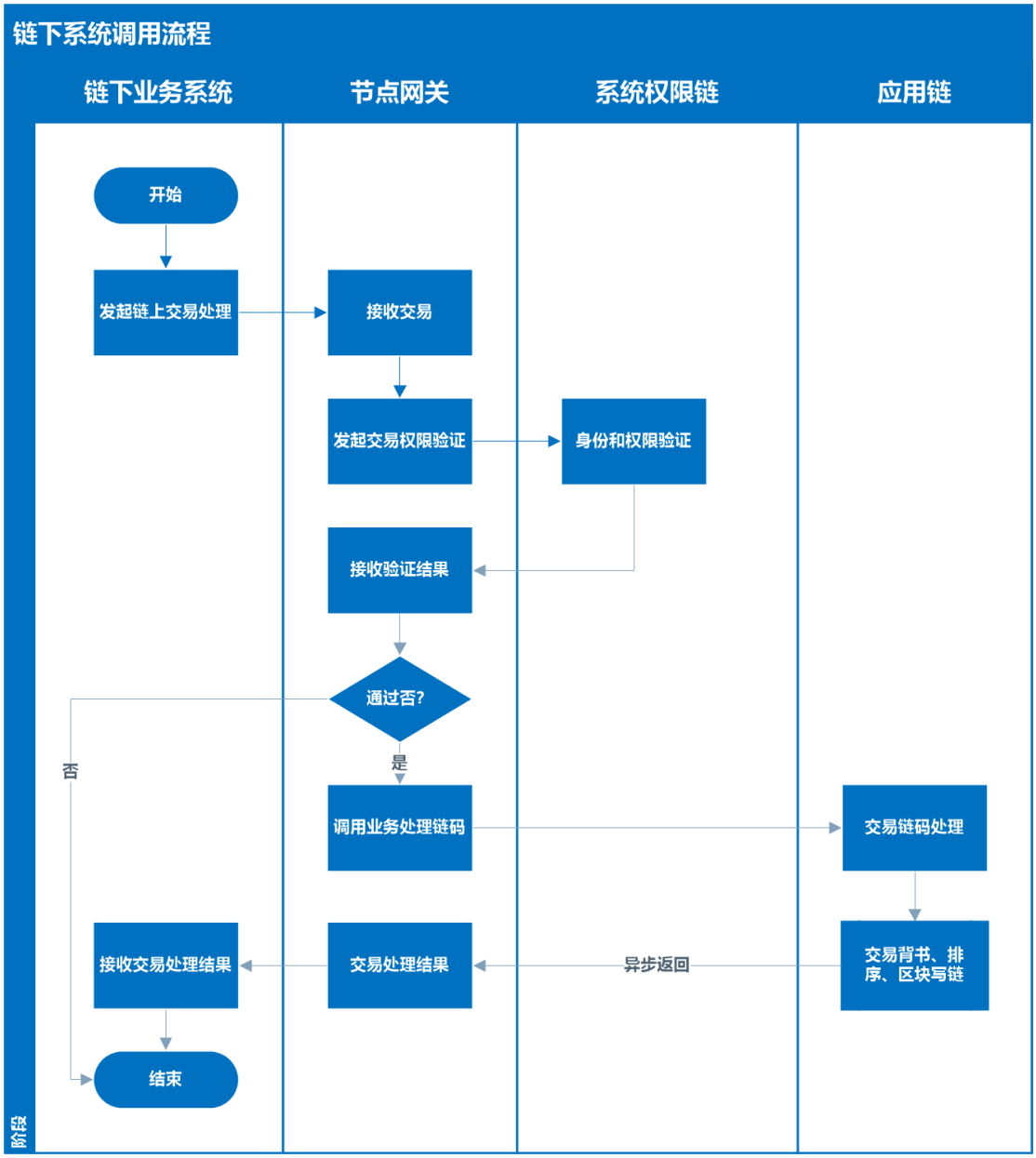
完整的区块链应用一般包含两部分：链上智能合约（应用服务链码（DApp Service Chaincode））和链下业务系统/BSN 外业务系统

（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。

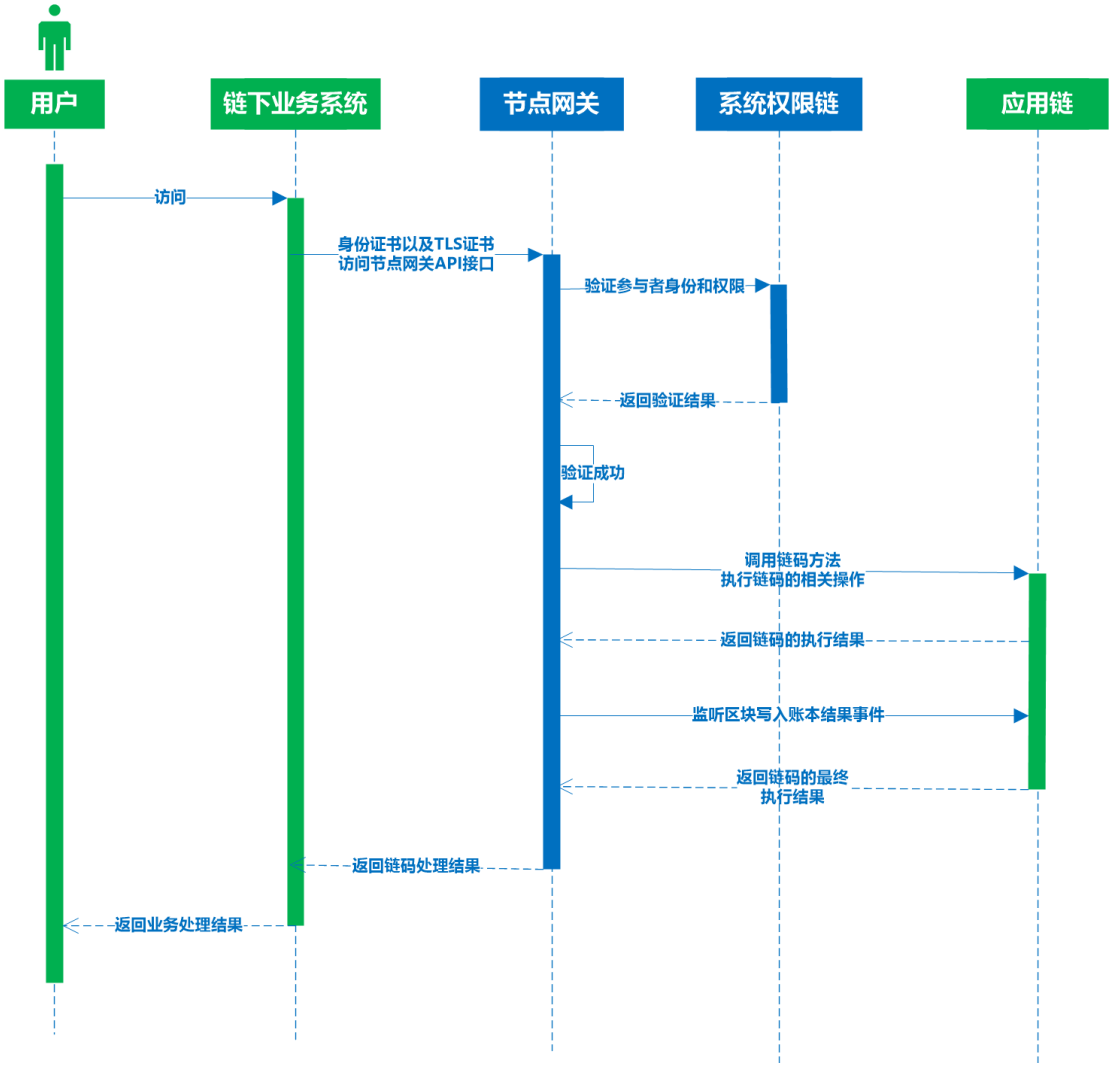


开发者需要自行开发应用的链下业务系统的业务逻辑，开发过程中可通过城市节点网关 API 来调用应用链码/智能合约执行交易事务处理、完成数据写链和数据查询等操作。

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

例：BsnBaseChainCode 链码包（预制链码包）

```
BsnBaseChainCode
├──main.go
├──ChainCode/
├──models/
└──utils/
```

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

例：BsnBaseChainCode 链码包

```
BsnBaseChainCode
|   build.gradle
└──src
    └──main
        └──java
            └──com.example.javacc
                └──javacc.java
```

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

例：BsnBaseChainCode 链码包

```
BsnBaseChainCode
    |   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。

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

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。

5.4.3 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）秘钥算法的 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

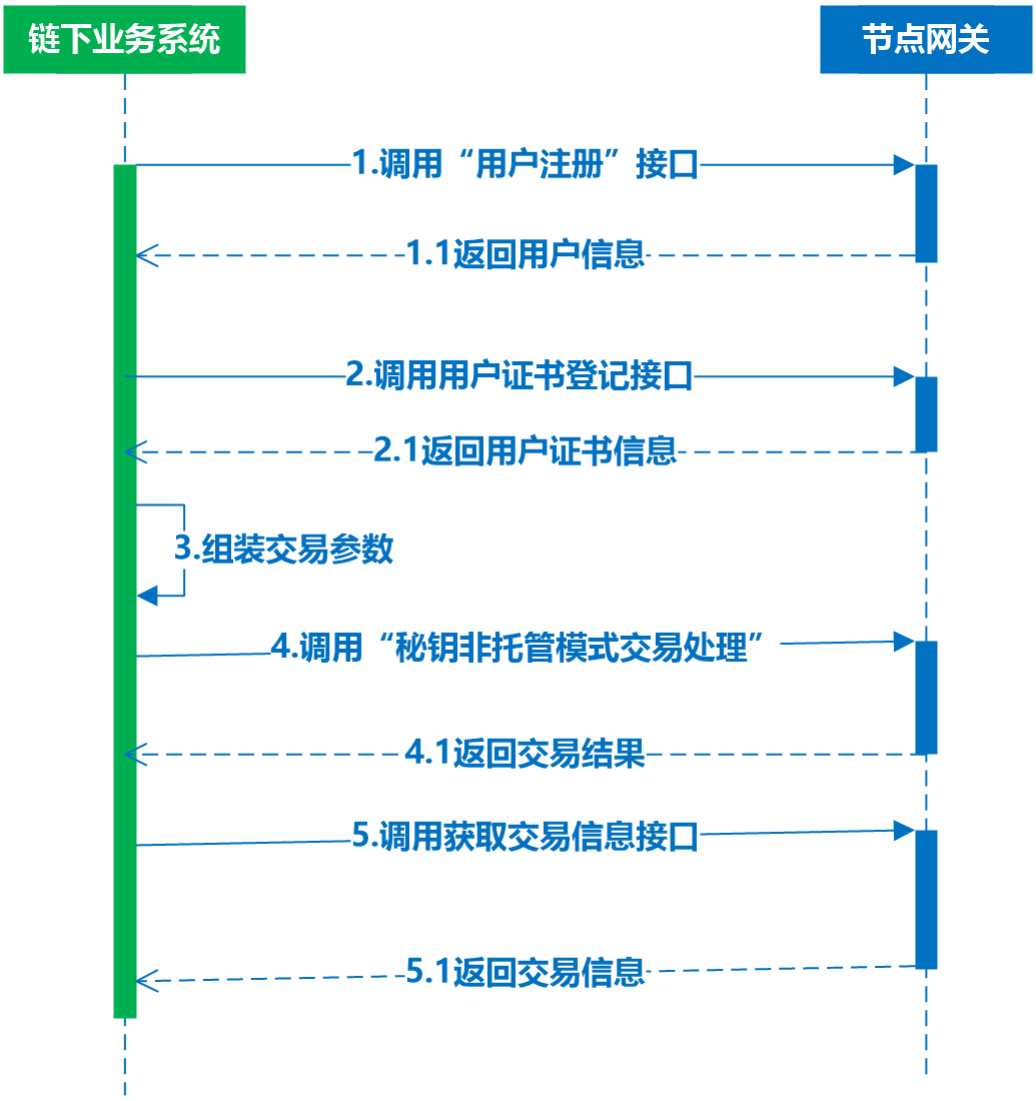
用户在参与应用时需要两个密钥对：应用接入密钥对和用户交易密钥对。在该模式下用户需要自己在本地生成公私钥对，私钥本地保存，只需要将公钥或公钥证书申请文件上传 BSN。

- **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":"USER00012020041519580108712920","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":"USER000120200415195801087129 2","appCode":"app0001202004161020152918451","tId":""},"mac":"MEUCIQDCa3T1c8Fim3LFVfgvelllC/wKWtFnyOl5FK7FXgddFwIgGHXApypixu9RpkHl13z80ZYdVeyRObX7icU3XWk2+VI=","body":{"name":"user01","secret":"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/KdVP6bJjHhV2pZPTe9Cn4xcOrPV4E7ZsA=",
  "body": {
    "name": "user01",
    "secret": "123456"
  }
}
```

5.4.3.5 密钥托管模式下调用链码接口

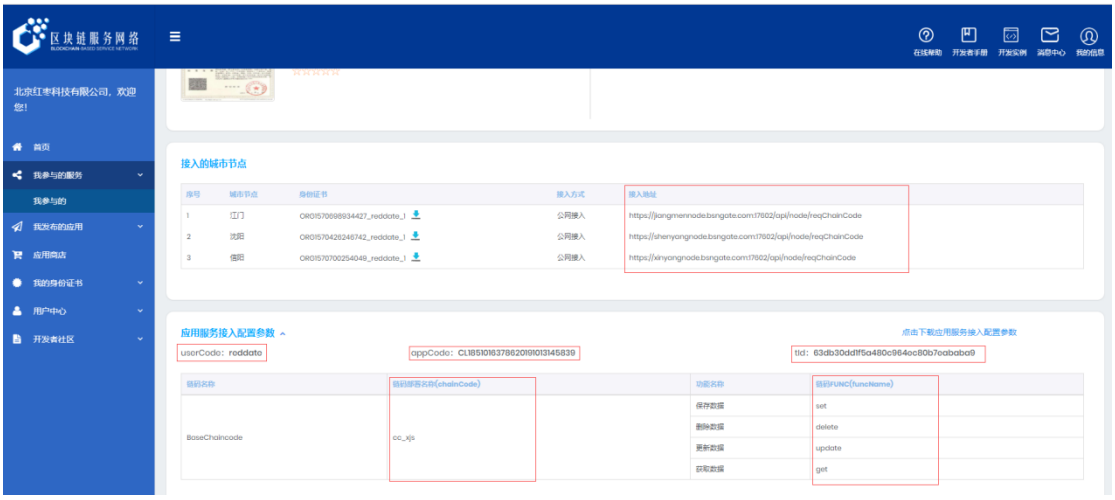
链下系统通过城市节点网关调用链码的交易处理函数时，需要按照接口说明在请求中加入相应的请求参数，调用节点网关后，节点网关会返回链码的执行结果。

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","appCode":"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":"USER000120200415195801087129 2","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/gIgfu93jqzTzk0DU2lfM
KnExcwVbgelWMLvLmwKplCXNBA=",
  "body": {
    "cert": "-----BEGIN CERTIFICATE-----
\nMIICvTCCAmSgAwIBAgIUcqn2HmCYmq/V2yKbnxuvc49KU00wCgYIKoZIzj0EAwIw\nTj
ELMAkGA1UEBhMCQ04xEDAOBgNVBAgTB0JlaWppbmcxDDAKBgNVBAoTA0JTTjEP\n
MA0GA1UECxMGY2xpZW50MQ4wDAYDVQQDEwVic25jYTAgFw0yMDA0MjEwNTAzM
```

DBa\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":"app0001202004161020152918451","tId":""},"mac":"MEUCIQCv8EZ2OqbSbI9xGGKX06Mquh+g+NhhbUoAJBbnemXdagIgNMF7W7ecu5uej9BpVx04qwJuVijbgcp3VYIcjDK0Z38=","body":{"transData":"Cq0KCrsJCpcBCAMaCwi9gPr0BRD0o+Z2IhxhcHAwMDAxMjAyMDA0MTYxMDIwMTUyOTE4N					

DUxKkBjM2M2NTIzOTU4YzM4MTExOTJiOGQzNThkZDI2MTdmMWIxNGNiNjYxZGU2Y
jAyMmMxYTgyMjI2OWU4YThjNDhkOiYSJBIiY2NfYXBwMDAwMTIwMjAwNDE2MTAy
MDE1MjkxODQ1MV8wMBKeCAqBCAoLT3JnYk5vZGVNU1AS8QctLS0tLUJFR0lOIENFUl
RJRklDQVRFLS0tLS0KTUlJQ3ZUQ0NBbVNnQXdJQkFnSVVWanBGZTJFaERFaHJlOHBB
VTh4bkd3dXhPbU13Q2dZSUtvWkl6ajBFQXdJdwpUakVMTUFrR0ExVUVCaE1DUTA0eEVE
QU9CZ05WQkFnVEIwSmxhV3BwYm1jeEREQUtCZ05WQkFvVEEwSlRUakVQCk1BMEdB
MVVFQ3hNR1kyeHBaVzUwTVE0d0RBWURWUVFERXdWaWMyNWpZVEFnRncweU1EQ
TBNVGt3TkRNek1EQmEKR0E4eU1UQXdNRE15TVRFeE1EUXdNRm93YkRFOE1BMEdB
MVVFQ3hNR1kyeHBaVzUwTUE4R0ExVUVDeE1JYjNKbgpZbTV2WkdVd0RnWURWUVF
MRXdkaWMyNWlZWE5sTUFvR0ExVUVDeE1EWTI5dE1Td3dLZ1lEVlFRRERTjBaWE4w
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":"USER000120200415195801087129 2","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@app00012020041610201529184 51",
        "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":"USER000120200415195801087 1292","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@app00012020041610201529184 51",
            "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":"USER00012020041519580108712 92","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 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/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	是	
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	是	
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=="}

5.4.3.13 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/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="}					

5.4.3.14 链码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	是	
2	城市编码	orgCode	String	是	
3	注册的链码事件 key	eventKey	String	是	

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	是	链码中设置的事件名称
7	当前链码的交易 ID	txId	String	是	
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
示例：					
{"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=="}				

5.4.3.15 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）秘钥算法的 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。

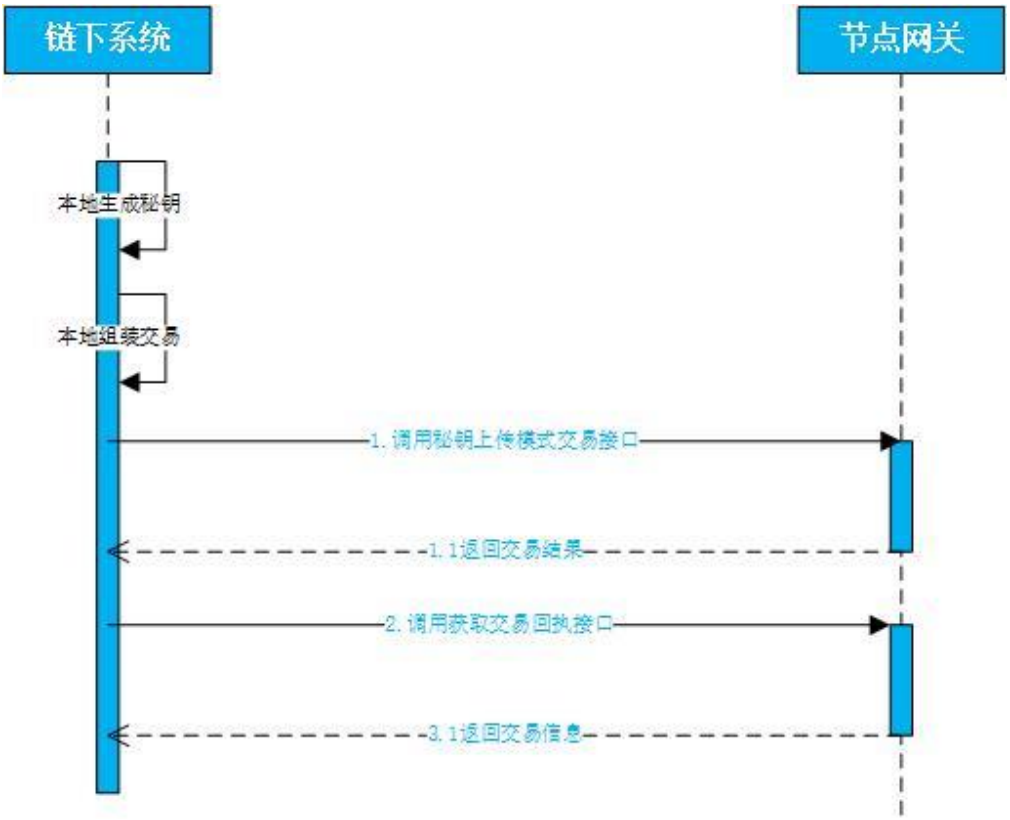
交易流程：

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":"USER00012020041519580108712920","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": "app000120200416102015291845 1"
  }
}
```


5.4.4.4 注册用户接口

用户参与一个 FISCO 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/L2wyXoddgTnZdqRsAiBxEBMeCOZ8M97OCRUAMZNMcL974vhzjOS/tk8/wbgbsA==",
  "body": {
    "userId": "100003",
    "userAddress": "0x14647a48303b5e1c77934583883ebc327ba3b297"
```

```
    }
}
```

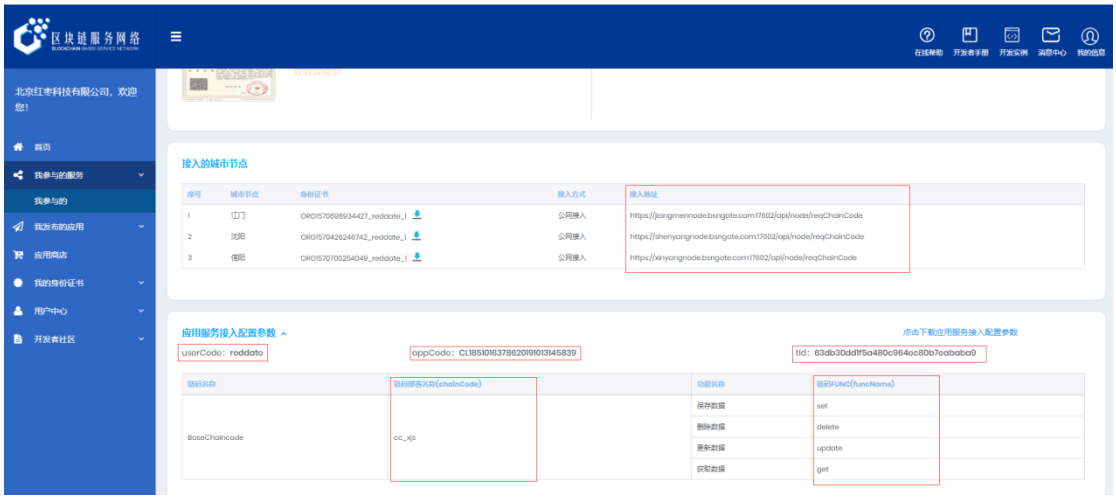
5.4.4.5 密钥托管模式下调用智能合约接口

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":"cl000620200318192657367 7572","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					

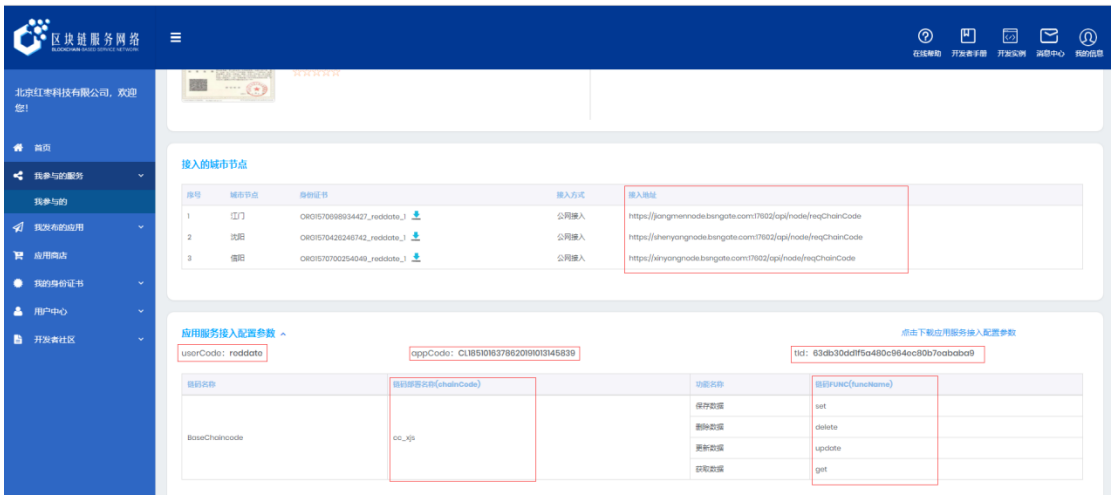
5.4.4.6 密钥上传模式下调用智能合约接口

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 67d7f7680b8e4ebf3b24f0000000000000000000000000000000000000000000000000000000000000 0000600000000000000000000000000000000000000000000000000000000000000000000000000500000000 0000000000000000000000000000000000000000000000000a0000000000000000000000 0000000000000000000000000000000000000000573036303400000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000000000000000000 0000000000002616100000000000000000000000000000000000000000000000000000000000000 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	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":"cl000620200318192657367757 2","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 ffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff160217905550600080905549 06101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffff1663c9 2a7801604080519081016040528060068152602001 7f745f626173650000000000000000000000 0000000000000000000000000000000815250604051826 3ffffffff167c0100000000000000000000000 0000000000000000000000000000000028152600401620001019190620002 4a565b60206040

```
518083038160008780 3b1580156200011c57600080fd5b505af115801562000131573d6000803e3
d6000fd5b505050506040513d601f19601f820116820180604052506200015791908101906200017
4565b50620002f4565b60006200016c8251620002a3565b905092915050565b600060208284031
215620001875760008 0fd5b60006200019784828501620001 5e565b91505092915050565b600062
0001ad8262000298565b808452620001c381602086016020860162000 2ad565b620001ce8162000
2e3565b602085010191505092915050565b6000601382527f626173655f6b65792c626173655f76
616c756500000000000000000000000000000060208301526040820190509190505 65b600060078252
7f626173655f696400000000000000000000000000000000000000000000000000000000000060208301526040
820190509190505 65b60006060820190508181036000830152620002668184620001a0565b905
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":"0xf27ff42d4be65329a1e7b11365e190086d92f9836168d0379e92642786db7ade" }, "mac":"MEUCIC5/z5u+4sO7SyBi//nk/QJ8WzBQU4y6WWzZViHZ2XbrAiEAuK0+QKIgkdDMxeGwh18FJuPJm6TdmTzc0SaSMXfxOAI="}					
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
ffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff160217905550600080905490
6101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff1663c9
2a7801604080519081016040528060068152602001 7f745f626173650000000000000000000000
0000000000000000000000000000000081525060405182 63ffffffff167c0100000000000000000000000
0000000000000000000000000000000000000028152600401620001019190620002 4a565b60206040
5180830381600087803b1580156200011c57600080fd5b505af11580156200013157 3d6000803e3
d6000fd5b505050506040513d601f19601f820116820180604052506200015791908101906200 01
74565b50620002f4565b60006200016c8251620002a3565b905092915050565b600060208284031
215620001875760008fd5b600062000197848285016200015e565b91505092915050565b600062
```

0001ad8262000298565b808452620001c381602086016020860162000 2ad565b620001ce81620002e3565b602085010191505092915050565b6000601382527f626173655f6b65792c626173655f76616c756500000000000000000000000000602083015260408201905091905056 5b600060078252 7f626173655f6964000000000000000000000000000000000000000000000000000000000000602083015260408201905091905056 5b600060608201905081810360008301526200026681846200001a0565b905081810360208301526200027b8162000213565b905081810360408301520000 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 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 fiscoo 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/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": "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":"USER00012020060423215796924 40","appCode":"app00012020060423230571010 02"},"body":{"eventId":"5b5b865f8dc94ae59d215cf26aa81d69","orgCode":"ORG20200411141716923 60","appCode":"app00012020060423230571010 02","attachArgs":"abc=123","nonceStr":"52f080f27ff045eb87e21812d12cee40","eventType":1,"eventData":"{\"appId\":\"app00012020060423230571010 02\",\"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

目前，XUPERCHAIN 的应用服务不支持上传公钥模式。

5.4.5.3 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：					
{ "header": { "appCode": "CL188103887322019090211431 4", "userCode": "newuser" }, "body": { "userId": "abc" }, "mac": "MEYCIQDUCidxeWRLhPjrYhfJIhkiJS+67lwnV/K4j+fkvpsRZgIhALtJvPwORTB9T6HwqdyDMvIgFDMd5p5CX61+pd6KFfqZ" }					
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": "MEUCIBiWAmyojuNPUHeZAPV006TFq1HSbzDUXZQBNLITcOtMAiEAuQ1mfxEkSJvqRlqqx6fRE8sO03woVxYSiJiuSDjEBx4=", "body": {					

```
    "userId": " abc",
    " userAddr": "2CcmGvEFpC16MAn3MzheMy5eR4FqzaPcHV"
  }
}
```

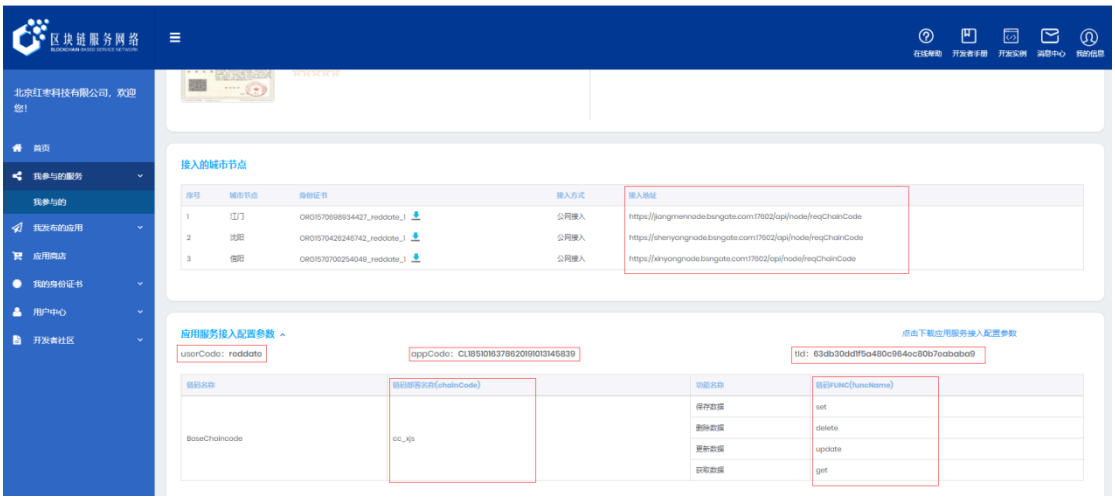
5.4.5.4 密钥托管模式下调用智能合约接口

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+oZgLAIhAOXpXJBcSOl3d4hcgcs6AvA0F6qnRtU2WwIqNyC1LqCh"
}
```

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"
    }
}
```


5.4.5.5 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
        }
}
```

5.4.5.6 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":"CL1881038873220190902114314",
        "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":
"MEUCIA1qT8akZGUMG8leQYNhGhJSZULpUkP4ZPJYq1aRWagbAiEAiLES1A0NYsWa5N
wuiIAOY3I/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"
    }
}
```


5.4.5.7 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

5.5 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

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。

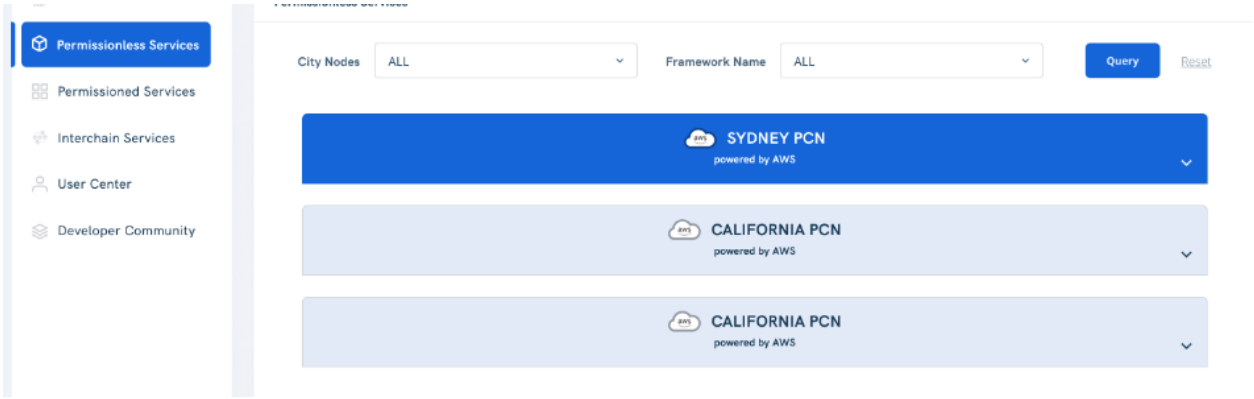
6 公有链服务

6.1 概述

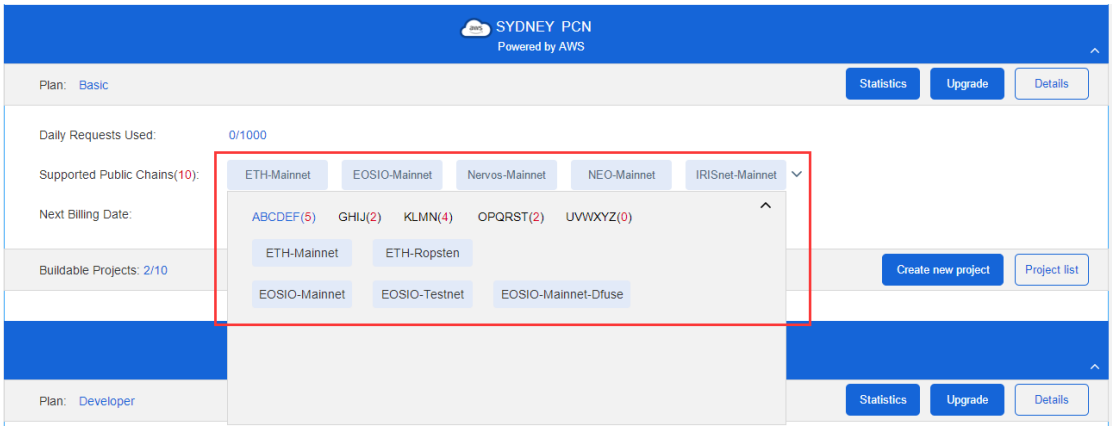
公有链服务为 DApp 开发者提供了一个接入公链主网或测试网的节点途径。目前区块链服务网络（BSN）已经适配了多个公链框架，其它公链框架将按批次每月陆续进行适配。公有链开发者无需再自己搭建公有链网络节点，通过 BSN 便可直接接入公有链网络，而且每月仅需要支付极低的一次性月费，即可同时调用所有 BSN 已适配的公链节点。

6.2 选择服务套餐

在公有链服务的列表页，用户可以选择不同的城市节点来参加公有链服务。排在列表上方的蓝色节点代表在该城市节点上已经购买付费套餐或正在使用免费套餐，列表下方灰色的节点代表在该城市节点还没有购买或使用过套餐：



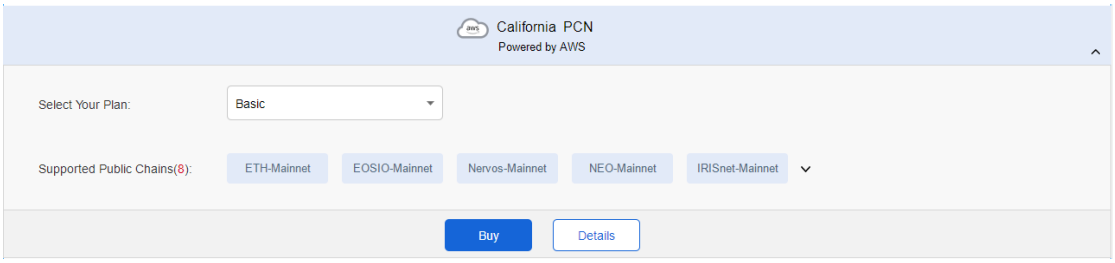
点击城市节点展开后，可以看到该城市节点所支持的所有公有链框架，用户可根据需要决定是否选择该城市节点作为接入入口，不同的城市节点所支持的公有链框架可能会不同。一般情况下，我们建议开发者选择离自己较近的城市节点，接入速度会相对较快：



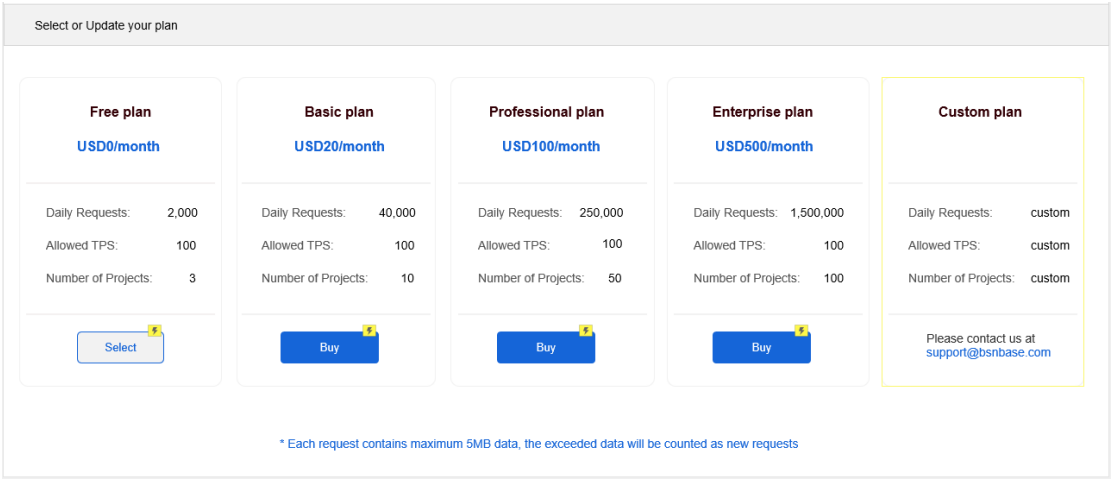
公有链套餐包括免费套餐，每天可以有 2000 个请求(Requests)。收费套餐有每月 20 美金、100 美金和 500 美金三种，分别包括每日 4 万，25 万和 150 万请求。所有请求可以自由分配到任何公有链上。

选择套餐步骤如下：

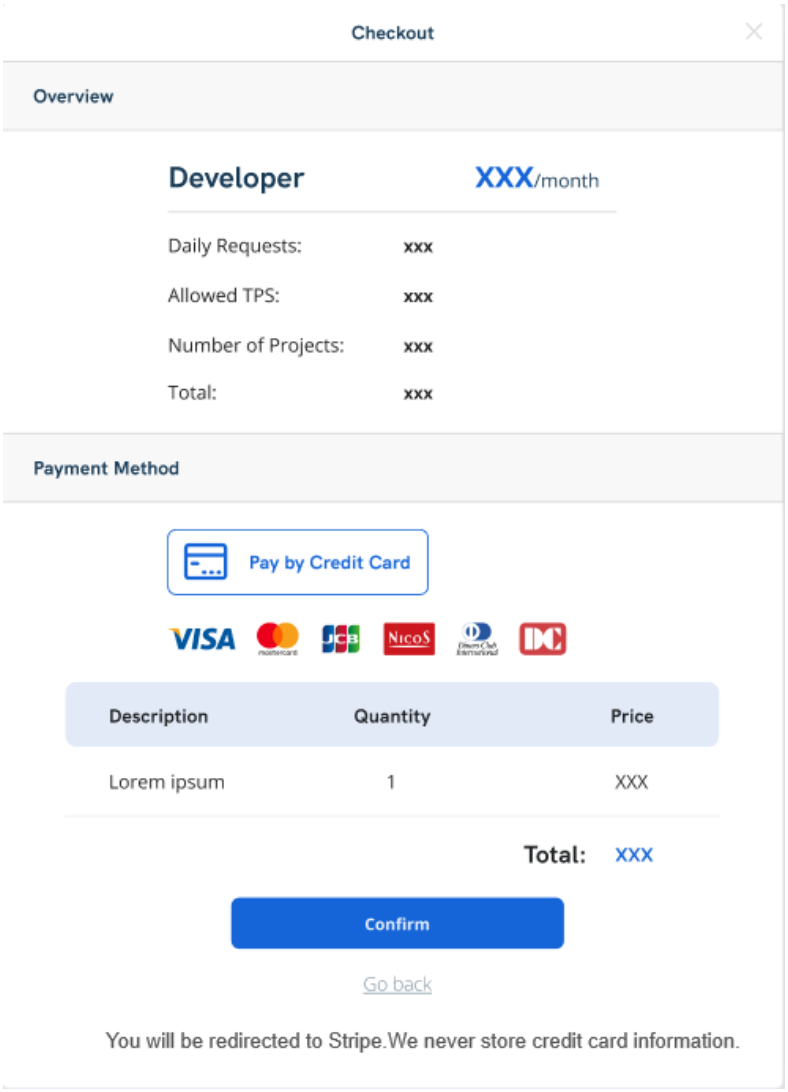
- 在公有链服务的列表页，用户根据需求选择城市节点后，在选择套餐处选择一个套餐（目前一个城市节点只支持使用一种套餐，可以进行套餐升级。但用户可以在不同城市节点加入不同或相同的套餐。免费套餐只能选择一次），点击购买按钮，进入套餐选择页面：



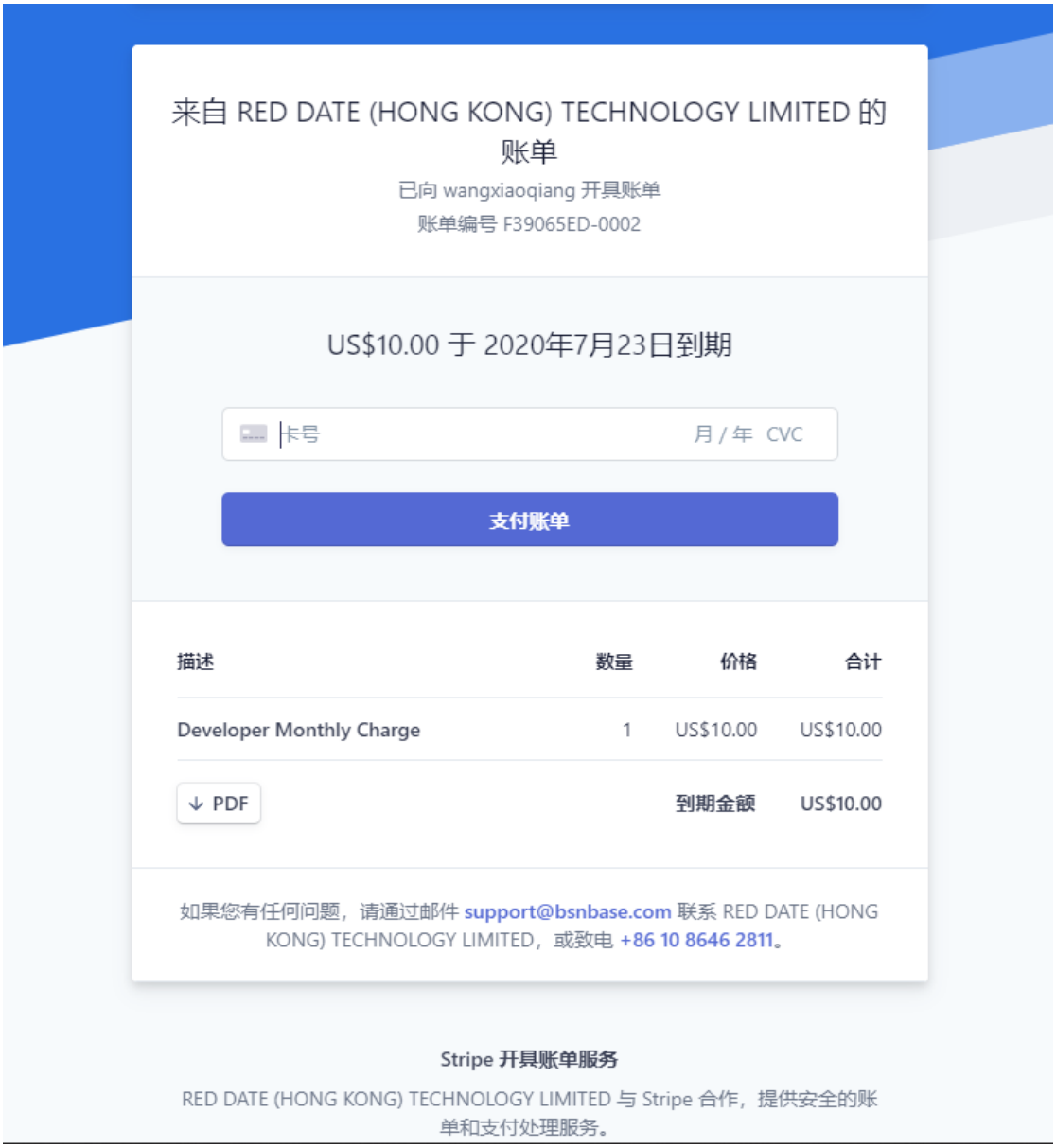
- 在选择或升级套餐页面，选择想要购买的套餐并点击购买按钮（注：每一个请求最大可容纳 5MB 的数据，如果数据超过最大可容纳量，则超过的部分会按照新的请求处理）：



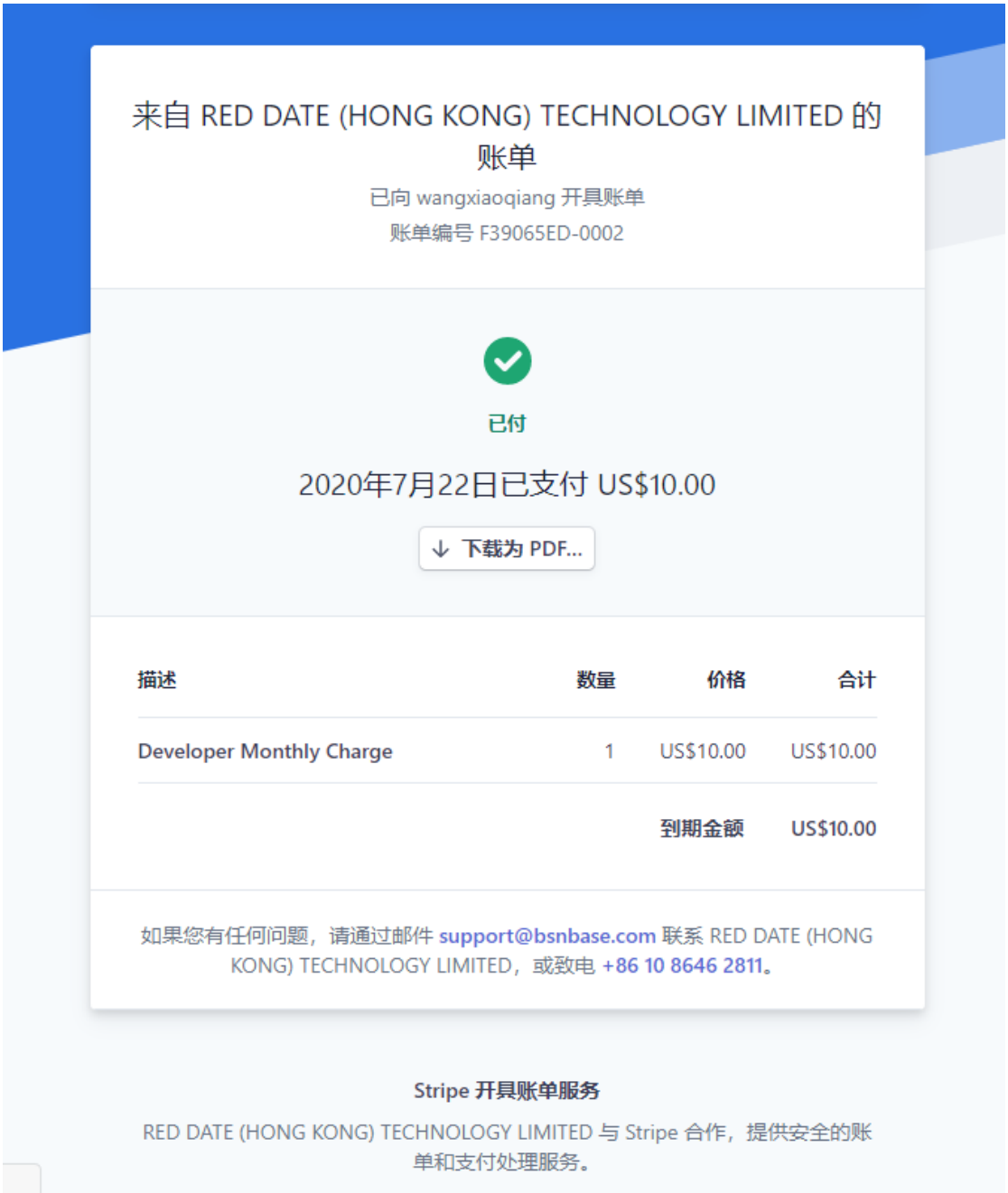
- 页面跳转到【确认支付】页面，用户确认套餐信息、支付方式以及需要支付的总金额，确认无误后点击“确认”按钮；



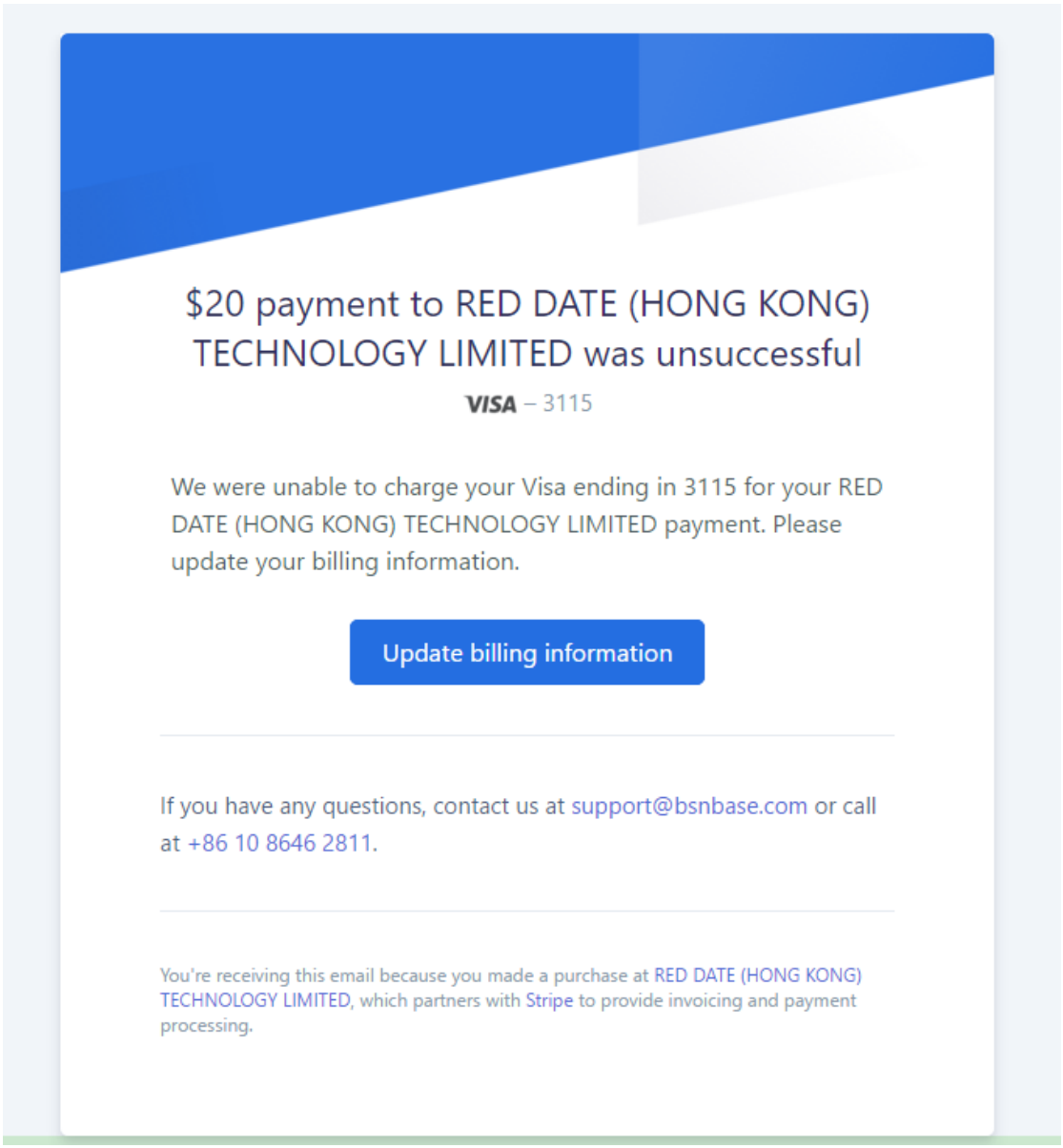
- 页面跳转到 Stripe 自带的支付页面，用户在该页面输入卡号信息进行支付。所有信用卡信息输入都在 Stripe 网站。BSN 官网永远不会接触到用户的信息卡信息：



- 支付成功后客户会收到通知邮件，并且客户也可下载 PDF 版收据和账单。



- 若支付失败，则系统会发送邮件通知客户，并建议客户到 BSN 国际门户网站【我的账号】页面查看详细信息：

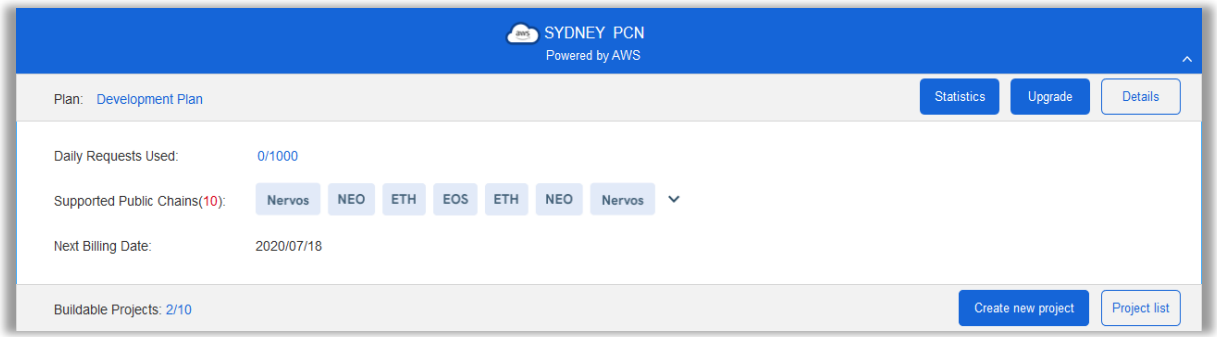


6.3 创建和管理项目

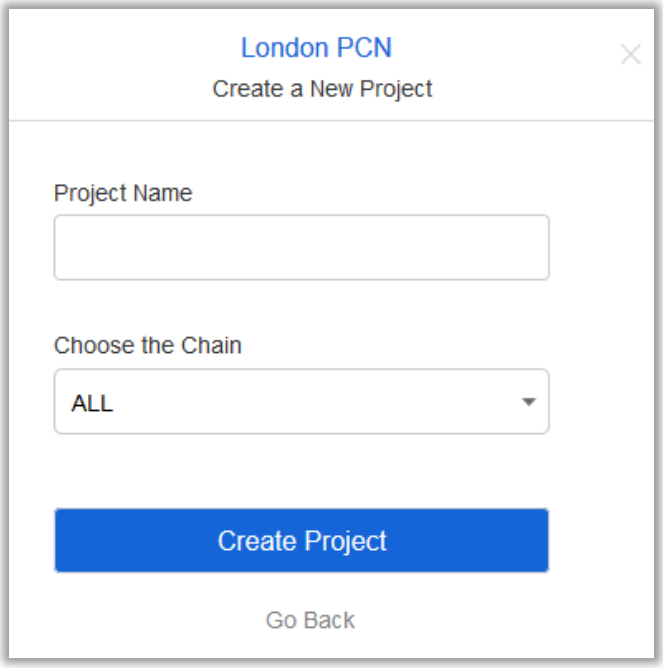
用户在一个城市节点的一个套餐下可以通过创建多个不同的项目来更方便的管理 DApp。

创建和管理项目的操作步骤如下：

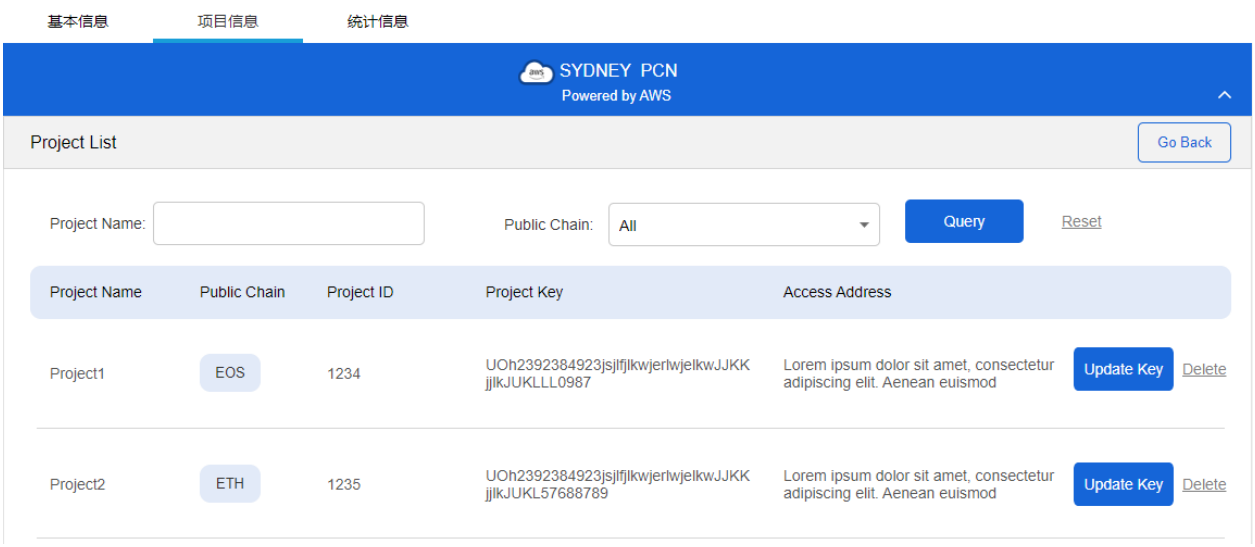
➢ 在公有链服务的列表页，在已使用免费或付费套餐的城市节点上，点击创建项目按钮创建项目。



➢ 在弹出的新窗口中，输入项目名称并选择部署应用的公有链框架，创建项目。每个项目只能选择一种公有链：



➢ 项目创建完成后，生成项目 ID、项目 Key 以及接入地址，页面自动跳转到【项目信息】页面：

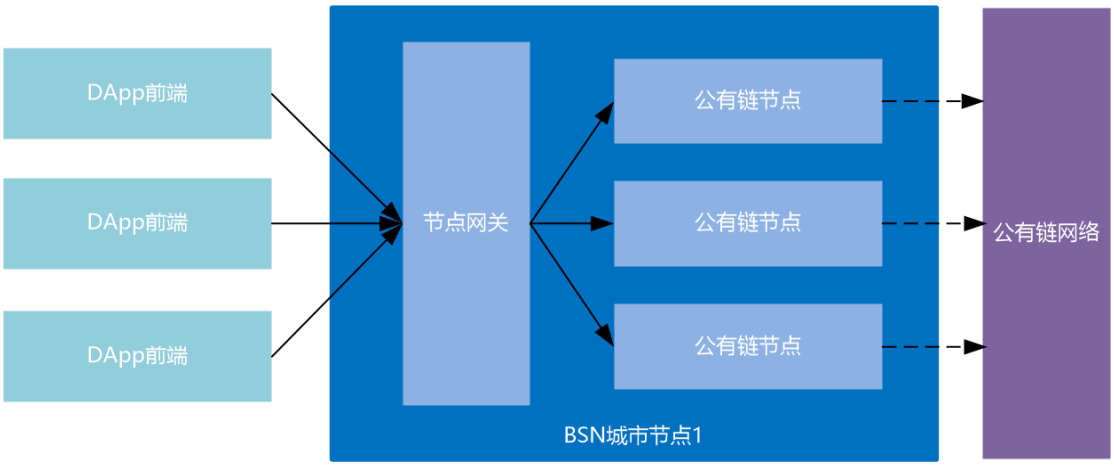


- 在【项目信息】页面，可以查询、查看项目信息；可针对某一项目更新项目 Key；也可删除某一项目信息。
- 用户使用项目 ID、项目 Key 以及接入地址通过节点网关接入公有链应用。

6.4 BSN 外系统接入指南

6.4.1 概述

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}格式拼接请求地址，同时请求报文头 header 中增加 x-api-key:{key}，如公链节点提供了多个组件，则需增加 x-api-sub-path:{subUrl}，如 Nervos 的 CKB 除了 RPC 服务外,还有个 indexer 组件服务,{subUrl}应填写 indexer。

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 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。

6.4.2 Ethereum

Ethereum is a global, open-source platform for decentralized applications. On Ethereum, you can write code that controls digital value, runs exactly as programmed, and is accessible anywhere in the world.更多开发者资源请访问：https://ethereum.org/en/developers/

BSN 城市节点网关适配了以太坊 JSON RPC API，所以开发者可以通过 JSON-RPC 的方式向节点网关发起交易请求。详细对接说明请访问：http://cw.hubwiz.com/card/c/ethereum-json-rpc-api/

以下为城市节点网关的其它错误码定义：

错误码	交易错误码	错误码描述
500	-32099	服务内部异常
503		
429	-32098	TPS,TPD 限流
401	-32097	验证权限失败

6.4.3 EOSIO

EOSIO is a blockchain platform designed for the real world. Built for both public and private use cases, EOSIO is customizable to suit a wide range of business needs across industries with rich role-based security permissions, industry-leading speeds and secure application processing. 更多开发者资源请访问：https://developers.eos.io

BSN 城市节点网关适配了 EOSIO 的 JSON RPC API，所以开发者可以通过 JSON-RPC 的方式向节点网关发起交易请求。详细对接说明请访问：
https://developers.eos.io/manuals/eos/latest/nodeos/plugins/chain_api_plugin/api-reference/index#operation/get_block

以下为城市节点网关的其它错误码定义：

错误码	交易错误码	错误码描述
401	3090000	验证权限失败
429	3210000	TPS,TPD 限流
500	3100000	服务内部异常
503		Service Unavailable

6.4.4 Nervos CKB

The Nervos Network is an open source public blockchain ecosystem and collection of protocols solving the biggest challenges facing blockchains like Bitcoin and Ethereum today.更多开发者资源请访问：https://docs.nervos.org/

BSN 城市节点网关适配了 Nervos 的 JSON RPC API，所以开发者可以通过 JSON-RPC 的方式向节点网关发起交易请求。详细对接说明请访问：https://docs.ckb.dev/docs/docs/HTTPRPC/rpc-specification/

以下为城市节点网关的其它错误码定义：

错误码	交易错误码	错误码描述
500	-32099	服务内部异常
503		
429	-32098	TPS,TPD 限流
401	-32097	验证权限失败

6.4.5 Neo

Neo is an open-source, community driven platform that is leveraging the intrinsic advantages of blockchain technology to realize the optimized digital world of the future.更多开发者资源请访问：https://neo.org/dev

BSN 城市节点网关适配了 Neo 的 JSON RPC API，所以开发者可以通过 JSON-RPC 的方式向节点网关发起交易请求。详细对接说明请访问：https://docs.neo.org/docs/zh-cn/reference/rpc/latest-version/api.html

以下为城市节点网关的其它错误码定义：

错误码	交易错误码	错误码描述
500	-32099	服务内部异常
503		
429	-32098	TPS,TPD 限流
401	-32097	验证权限失败

6.4.6 Tezos

Tezos is an open-source platform for assets and applications backed by a global community of validators, researchers, and builders. Tezos is designed to provide the safety and code correctness required for assets and other high value use cases. Its native smart contract language, Michelson, facilitates formal verification, a methodology commonly used in mission-critical environments such as the aerospace, nuclear, and semiconductor industries.更多开发者资源请访问：https://developers.tezos.com

BSN 城市节点网关适配了 Tezos 的 JSON RPC API，所以开发者可以通过 JSON-RPC 的方式向节点网关发起交易请求。详细对接说明请访问：https://tezos.gitlab.io/api/rpc.html

以下为城市节点网关的其它错误码定义：

错误码	交易错误码	错误码描述
401	3090000	验证权限失败
429	3210000	TPS,TPD 限流
500	3100000	服务内部异常
503		Service Unavailable

6.4.7 IRISnet

IRISnet (a.k.a IRIS Hub) is designed to be the foundation for the next generation distributed applications. Built with Cosmos-SDK, IRIS Hub enables cross-chain interoperability through a unified service model, while providing a variety of modules to support DeFi applications.更多开发者资源请访问：https://www.irisnet.org/docs

BSN 城市节点网关适配了 IRISnet 的 JSON RPC API，所以开发者可以通过 JSON-RPC 的方式向节点网关发起交易请求。详细对接说明请访问：https://www.irisnet.org/docs/light-client/intro.html#rest-apis

以下为城市节点网关的其它错误码定义：

错误码	交易错误码	错误码描述
500	-32099	服务内部异常
503		
429	-32098	TPS,TPD 限流
401	-32097	验证权限失败

6.4.8 EOSIO-Dfuse

Dfuse is a massively scalable open-source platform for searching and processing blockchain data. Dfuse empowers developers with capabilities to build modern blockchain applications with fast, fluid interfaces that deliver exceptional user experiences.

更多开发者资源请访问：https://docs.dfuse.io/

BSN 城市节点网关适配了 dfuse 的 JSON RPC API 和 GraphQL，所以开发者可以通过 JSON-RPC 或 GranphQL 的方式向节点网关发起 EOSIO 的交易请求。详细对接说明请访问：

https://docs.dfuse.io/reference/eosio/

以下为城市节点网关的其它错误码定义：

错误码	交易错误码	错误码描述
401	3090000	验证权限失败
429	3210000	TPS,TPD 限流
500	3100000	服务内部异常
503		Service Unavailable

7 跨链服务

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。

目前 BSN 开发团队正在建立 BSN 的“Interchain Communications Hub”（跨链通讯枢纽）。枢纽内会提供多家的跨链机制和预言机机制。预计将在 2020 年下半年推出演示版本，将包括基于 IRITA 的跨链服务和基于 Chainlink 的预言机服务。

演示版本推出后，会在目前的 BSN 官网内推出“跨链服务”项目。敬请大家期待。

8 账户管理

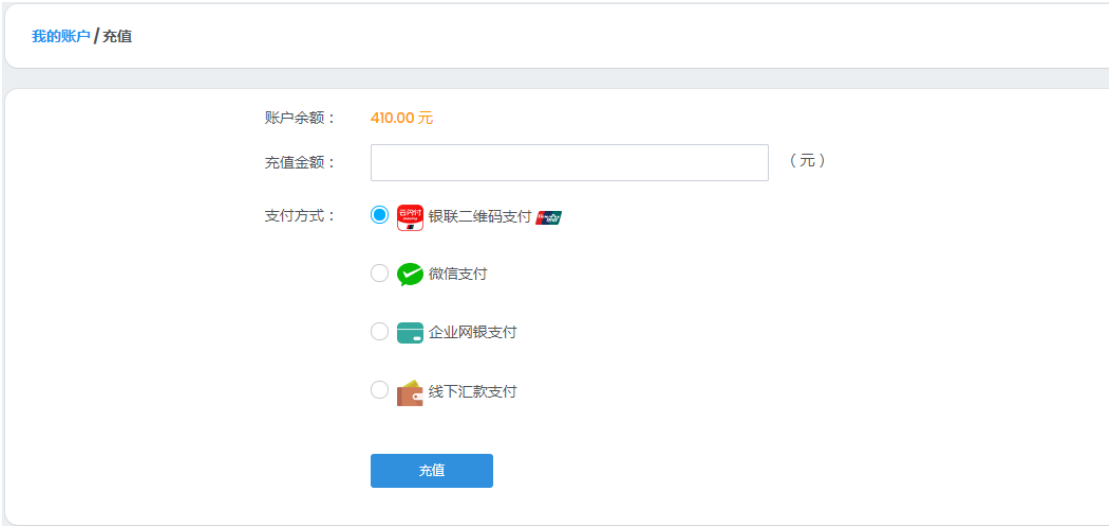
8.1 我的账户

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 3.10 我的发票

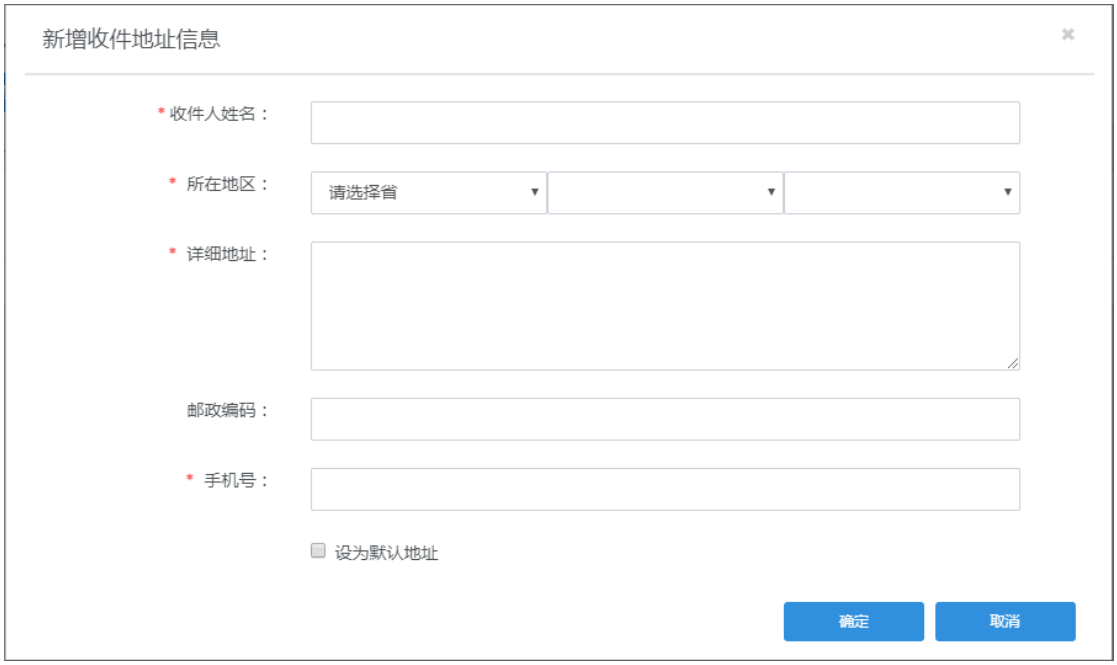
5. 管理收件地址

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

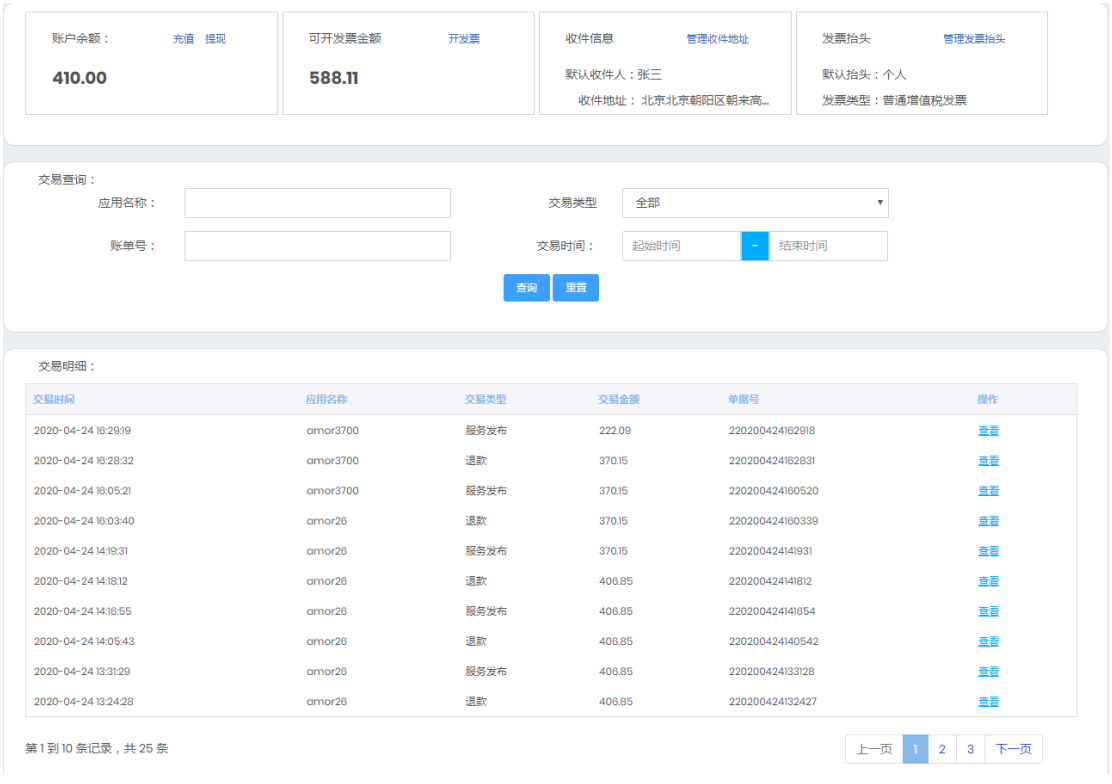
头管理页面，如下：

➢ 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：



8.2 我的账单

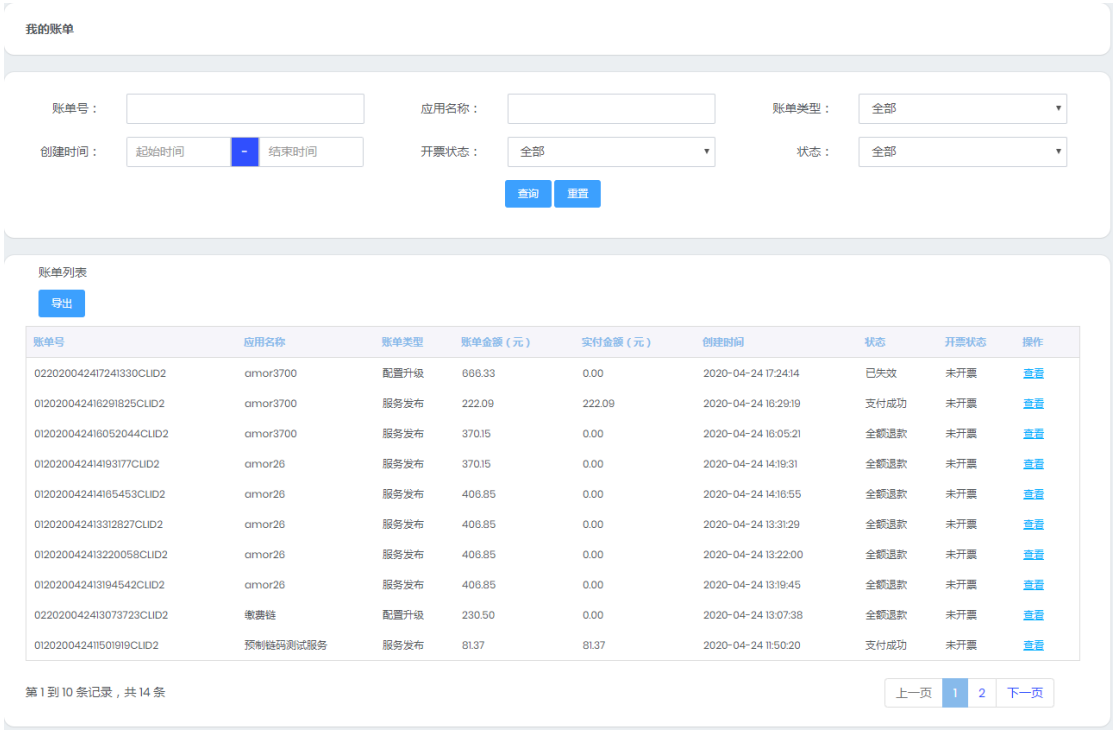
用于显示应用产品或应用服务产生的账单信息。

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。

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。

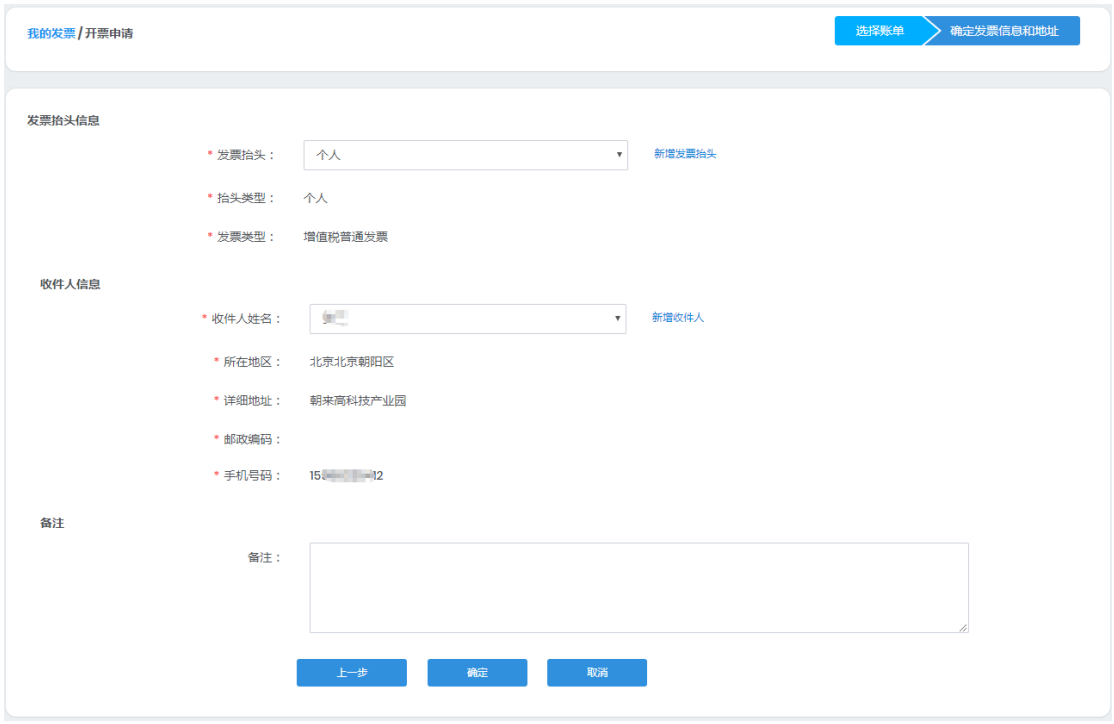


8.3 我的发票

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发票，并邮寄给开票申请人。

9 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	1.2.0	2020-07-31	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

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	--	--	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://www.dfuse.io/	Github Documentation

10 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：