

BBE 文档



目录	
目录	2
功能发布记录	5
产品描述	5
介绍	5
核心概念	6
特性	7
应用场景	12
定价与购买	14
产品定价	14
选购指南	14
操作指南	16
XuperChain	16
XuperChain试用网络	16
Fabric联盟链	21
组织管理	21
联盟管理	22
节点管理	26
通道管理	28
链码管理	31
事件中心	35
SDK应用	36
证书管理	37
quorum联盟链	39
组织管理	39
联盟管理	41
节点管理	44
合约管理	47
账号管理	51
事件中心	53
兼容性说明	53
JSON-RPC防火墙	54
以太坊	57
概念介绍	57
购买创建	58
JSON-RPC防火墙	60
可信跨链平台	63
跨链服务特性介绍	63
跨链服务控制台	64
跨链应用	68
多用户访问控制	68

典型实践	71
浦发银行区块链催收	71
开发者指南	73
以太坊Dapp开发指南	73
Dapp架构及概要设计	73
智能合约开发及部署	76
Dapp应用层开发	89
Dapp部署	113
Fabric-Dapp开发指南	117
Dapp架构及概要设计	117
链码开发及部署	120
Dapp应用层开发	125
Dapp部署	148
视频指南	151
操作指导	151
API参考	151
简介	151
概述	151
接口使用规范	151
服务域名	153
公共请求头	153
公共响应头	153
错误码	153
网络创建相关接口	156
Quorum相关接口	164
联盟相关接口	164
组织相关接口	172
事件相关接口	177
钱包账户相关接口	184
合约管理相关接口	191
合约网关相关接口	197
Ethereum相关接口	209
获取以太坊网络部署列表	209
查询以太坊网络	211
删除以太坊网络	213
查询账户列表	214
创建新账户	216
查询账户私钥	217
下载keystore	218
删除账户	219
查询合约列表	220

获取合约详情	222
销毁合约	224
合约网关相关接口	225
fabric联盟链相关接口	238
组织管理	238
联盟管理	244
通道管理	258
申请、审批和通知管理	270
链码管理	279
节点管理	285
证书管理	290
超级链相关接口	294
删除超级链网络	294
获取超级链网络全量信息	295
获取超级链部署列表	297
查询预定义超级节点账户	299
获取代币总量	300
附录	302
公共部分	302
以太坊相关	302
创建接口相关	304
Quorum联盟相关	306
Fabric联盟相关	310
功能更新记录	312
常见问题	312
服务等级协议SLA	315
超级链BaaS平台服务等级协议SLA	315

功能发布记录

发布时间	功能描述
2020-06	Fabric 联盟链支持证书管理
2020-05	以太坊支持JSON-RPC防火墙 Quorum联盟支持JSON-RPC防火墙
2019-12	BDChain 联盟链上线 Fabric 支持节点日志查看 Quorum 支持节点日志查看
2019-11	合约网关支持 https 请求 Fabric支持默认背书策略 Fabric与Quorum联盟支持 免费体验 开放Fabric联盟链和Quorum联盟链 OpenApi
2019-10	Quorum联盟支持 节点资源监控 Fabric联盟支持 节点资源监控
2019-08	API新增 合约网关相关接口 ，包括Event监听、交易结果查询和调用合约等接口
2019-07	Fabric联盟链 和 Quorum联盟链

产品描述

介绍

超级链BaaS平台（Xuper BaaS）

- 超级链BaaS平台是百度区块链团队自研，为用户提供全面区块链服务的平台产品，能快速的为企业和开发者在公有云、私有云中搭建区块链网络，支持多底链框架，包括XuperChain、Fabric、Quorum以及以太坊。
- 借助于百度智能云的技术与产品积累，帮助企业和开发者快速、规范的搭建企业级去中心化业务系统，辅助企业建立企业内外的业务可信协作。
- 同时将区块链落地行业中积累的创新实践，通过产品和技术输出赋能至合作伙伴，助力合作伙伴加速落地区块链，包括多链架构、跨链数据同步、可信计算、链上链下安全、多层次激励体系等。

产品架构图



- 区块链BaaS 基于百度云容器引擎CCE，构建区块链即服务平台，用户仅需要根据企业对于区块链网络的需求进行简单的参

数配置，即可搭建出符合业务要求的区块链网络，目前提供XuperChain试用版本、企业级联盟链两种产品形态。

(1) XuperChain试用版本：定位于试用，为用户提供低版本的XuperChain试用功能，让用户能够快速检验业务系统与区块链结合的可行性。

(2) 企业级联盟链：以联盟的形式链接企业，通过数据的协作建立企业间合作的信任，适用于企业协作系统的搭建。

- 区块链中间层 解决区块链网络到业务系统构建的最后一公里，让企业开发者快速完成基于区块链网络开发搭建可信的去中心化业务系统。

1、合约网关RESTful API：

开发以太坊Dapp，与智能合约交互通常使用web3，这种方式需要开发者管理Nonce、构建交易、签名交易、解析合约返回数据等，并且在调用过程中容易出现各种错误，没有很好的提示和处理机制，对开发人员来说，并不是特别友好。基于此，我们开发的以太坊合约网关旨在为用户提供企业级的合约管理服务，使用传统的RESTful API设计让应用开发人员聚焦于自身的业务逻辑和用户体验，将复杂的合约事物提交、Nonce管理等交由合约网关来处理。

2、Oracle预言机：

由于区块链是确定性的环境，它不允许不确定的事情或因素，智能合约不管何时何地运行都必须是一致的结果，所以虚拟机（EVM）不能让智能合约有网络调用，不然结果就是不确定的。而这个特性大大限制了DApp的发展，因为很多DApp都需要与链下数据进行交互。所以区块链只能由特定的服务把外部数据传递给区块链上的智能合约，这个特定的服务就被称之为预言机Oracle。超级链BaaS平台提供封装好的基于SGX、MesaTEE的Oracle可信预言机满足用户快捷的实现链上链下数据打通。

3、通用存证 API：无需浪费精力于研究如何开发智能合约，超级链BaaS平台提供通用存证API，封装区块链与智能合约间的复杂交互，企业开发者使用传统的API方式即可将业务系统与区块链底层网络进行打通。同时，支持用户自定义存证内容的关键字段，满足用户的多样的存证诉求。在此之上，提供大文件哈希存证、音频视频指纹提取存证、基于IPFS的分布式存储等技术解决方案，适配于多种存证场景。

核心概念

区块链

区块链是分布式数据存储、点对点传输、共识机制、加密算法等计算机技术的新型应用模式。区块链（Blockchain）是比特币的一个重要概念，它本质上是一个去中心化的数据库，同时作为比特币的底层技术，是一串使用密码学方法相关联产生的数据块，每一个数据块中包含了一批比特币网络交易的信息，用于验证其信息的有效性（防伪）和生成下一个区块。

底层框架

随着区块链技术的商业应用，业务场景中需要对交易频次、身份认证、隐私保护、监管要求、响应时间等各方面进行优化或者支持，因而基于比特币的底层技术诞生了多种区块链底层框架，他们更加符合业务场景中的特定需求，目前超级链BaaS平台支持业内三种主流底层框架Ethereum、Fabric、XuperChain（百度超级链），能够基本覆盖用户的所有区块链业务应用场景。

共识机制

区块链作为一种按时间顺序存储数据的分布式数据结构，必须通过共识机制来保证所有节点维护的账本的真实和一致。往往会根据业务使用场景的情况来选择合适的共识机制，在安全性、扩展性、性能效率、资源消耗四个维度中进行权衡，最终选择最适合业务的共识机制，百度区块链团队会帮助用户选择最合适的共识机制来适配业务。

智能合约

以数字形式定义的一系列承诺（promises），包括合约参与方可以在上面执行这些承诺的协议，智能合约允许在没有第三方的情况下进行可信交易，这些交易可追踪且不可逆转。以太坊（Ethereum）项目将图灵完备的智能合约能力融入到了区块链世界，也是区块链2.0的典型标志事件。

Dapp

分布式应用/去中心化应用，Dapp是基于区块链开发的与用户进行交互的顶层系统，Dapp上承载着业务逻辑。

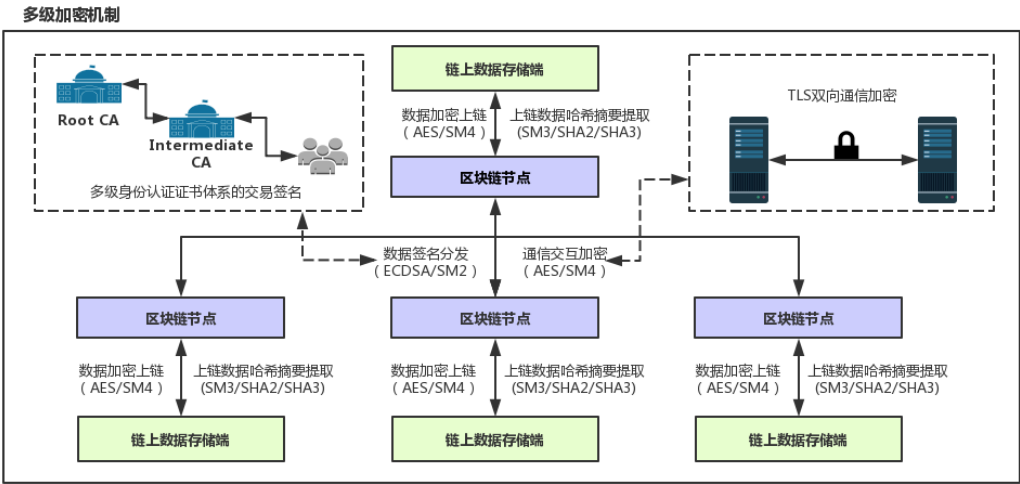
超级链BaaS平台将在这三个核心步骤中提供辅助，大幅降低开发难度，快速完成业务上链。

特性

可信计算环境

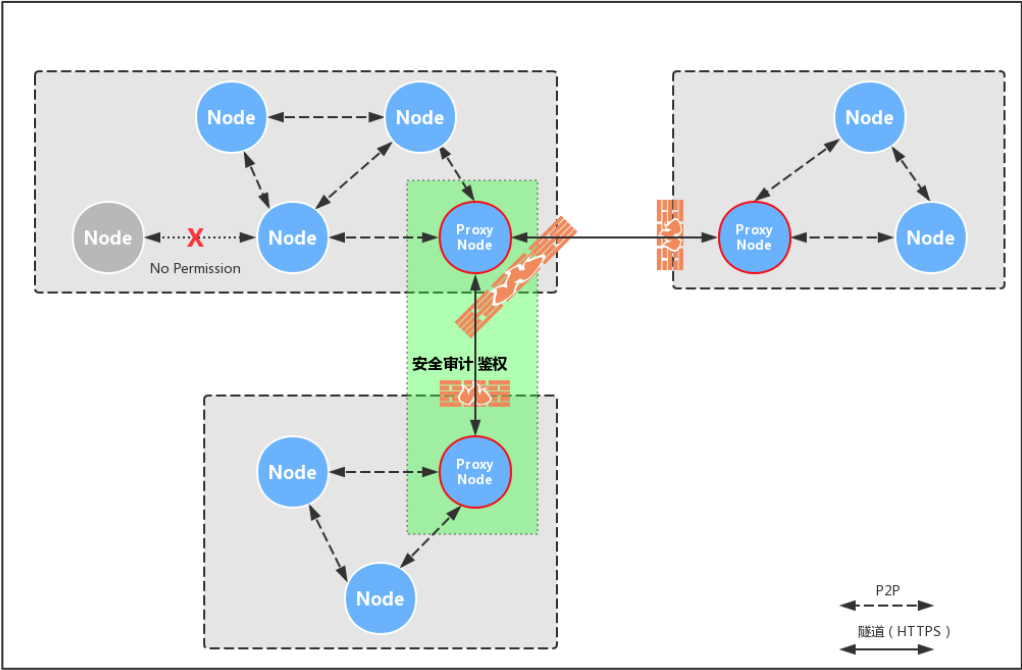
超级链BaaS平台基于以下多个维度的可信计算环境支持，实现全方位区块链网络安全保护，全时段维护业务链上应用信息、数据、执行逻辑的安全可信。

- 多级加密技术：
- 支持数据上链、数据传输、合约调用等多流程多种加密算法逐级加密及验证。

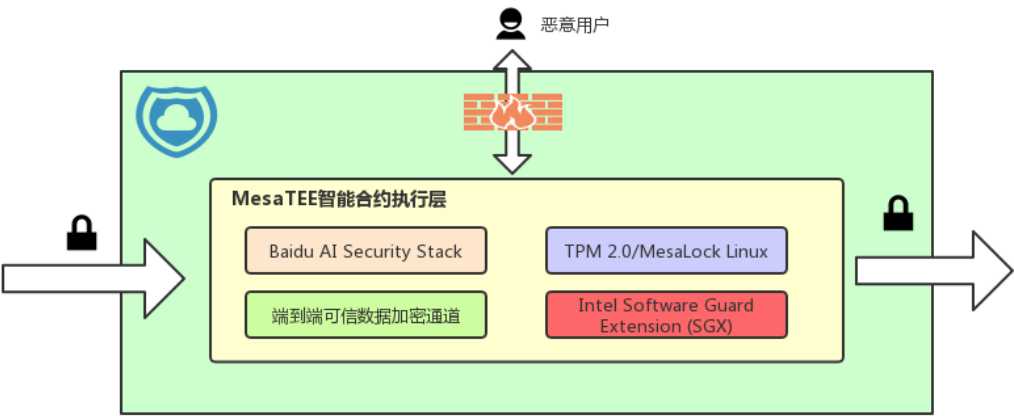


- 国际/国密标准的加密算法支持：
- 非对称加密算法：sm2、ECC
- 哈希算法：sm3、sha-2、sha-3
- 对称加密算法：sm4、aes、des
- 跨链网安全代理：多架构、多类别跨链数据交互及合约调用通过安全代理模块支持跨链数据安全加密及安全准入、审计控

制。



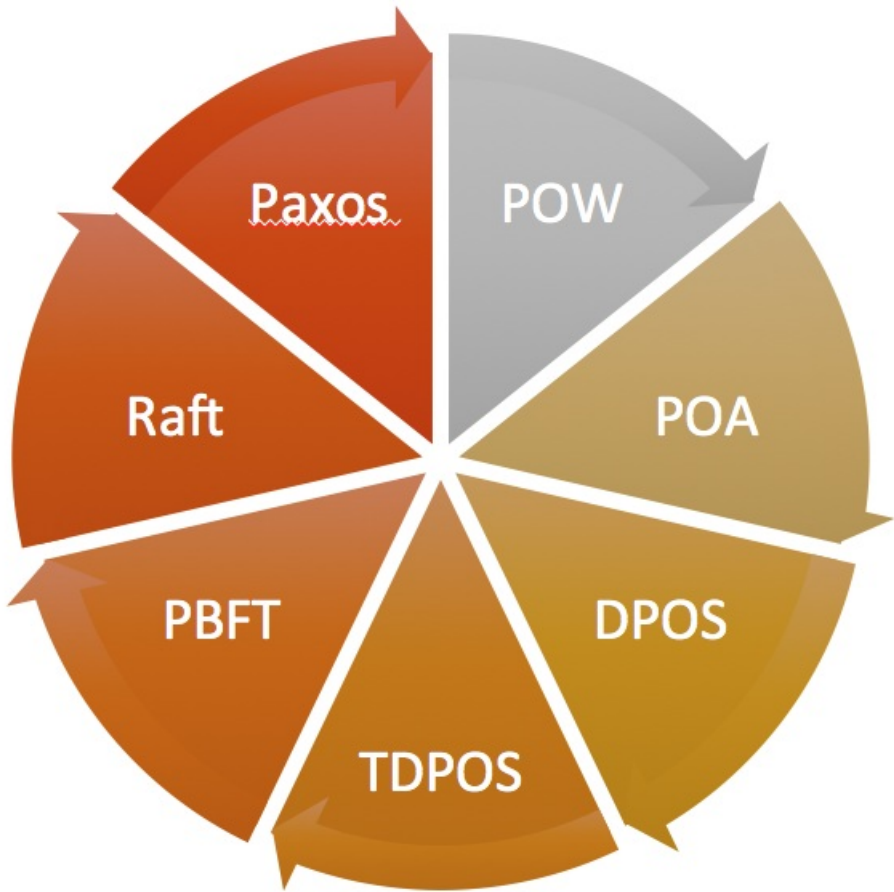
- 基于可信硬件的自研合约安全执行环境：
 - 基于SGX安全硬件及百度MesaTEE安全框架在合约层采用Baidu AI Security Stack (BASS)的Hybrid Memory Safety (HMS) 机制，免疫内存安全漏洞；同时额外加强了侧信道防御，领先业界。
 - 结合TPM 2.0与MesaLock Linux，纵深防御未知侧信道等攻击。
 - 用内存安全语言构建TLS协议栈，支持SGX Enclave-to-Enclave和Untrusted-to-Enclave两种可信验证模式。



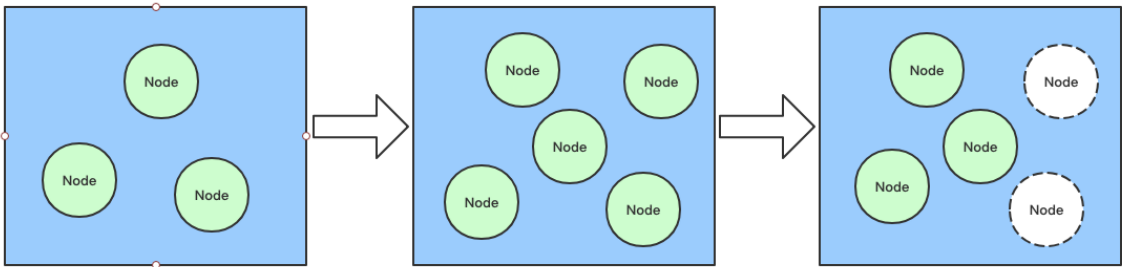
高性能高吞吐

实现基于多链架构，支持区块链网络及链上应用规模性增长。用户可根据业务场景需求选择区块链架构，进行链网参数优化及共识机制切换，突破性能与吞吐的极限。

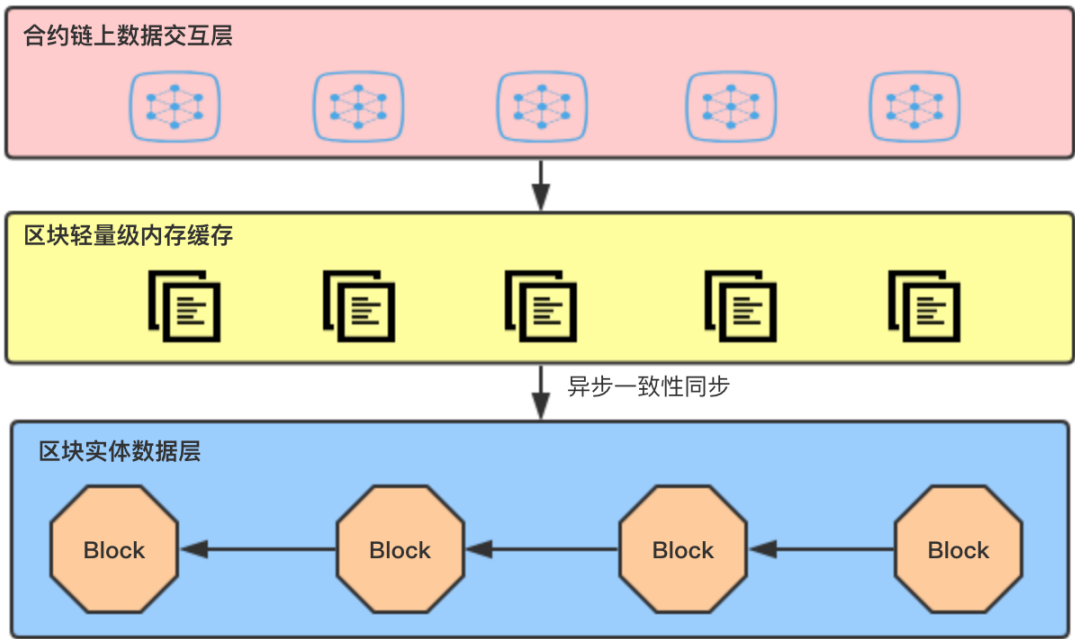
- 适配不同业务场景的多种共识机制：针对超级链BaaS平台跨链架构及具体业务场景，我们提供针对公有链、私有链以及联盟链的多种共识机制支持。
 - 公链场景下百度超级链实现基于时序的TDPOS共识算法，支持20000+TPS。
 - 联盟链场景实现基于Paxos/Raft/PBFT等多种共识机制，最大可支持10000+TPS。
 - 基于以太坊的私有链场景可选POW、POA、DPOS等多种共识机制。



- 支持多链架构水平动态扩缩容



- 基于轻量级内存缓存的架构优化



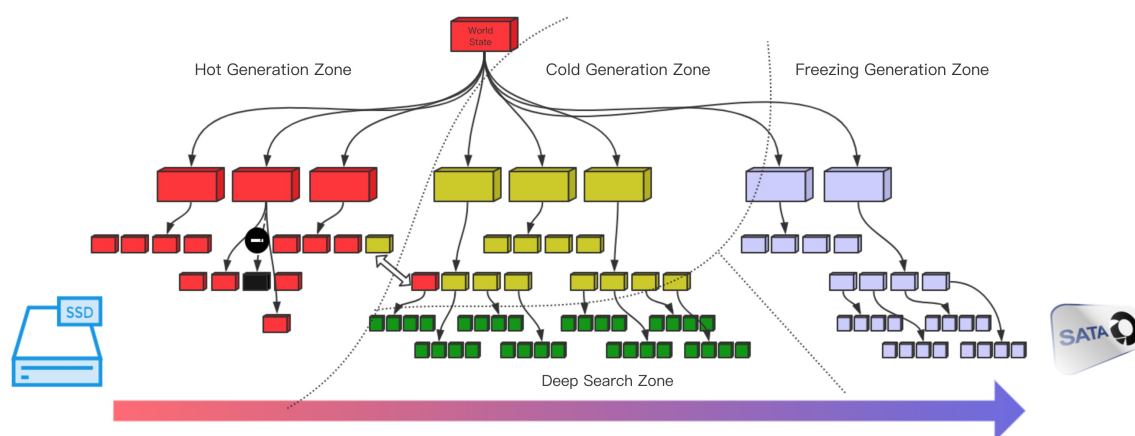
可扩展的存储

区块链通过节点间存储的高冗余来保障链上数据的高可用及安全性。这也就意味着相比于中心化的存储系统，区块链网络保存的数据副本基本随节点规模线性增长。反过来由于世界态的不可破坏性，区块链中每个节点都会尽量多地保存原始的全局数据。包括状态数据、交易数据、交易凭证甚至事件数据都会持久化到节点存储中。在实际生产环境中，区块链节点需要的存储空间大小会随着交易数量增加而持续增长。

百度智能云能提供无限量的存储空间，并且能达到数据可用性和数据安全性的业界标准。同时超级链BaaS平台结合云存储深度定制区块链节点存储机制，实现区块链专有的存储技术。这也是许多企业级用户选择超级链BaaS平台引擎来托管生产级区块链系统的原因。超级链BaaS平台引擎主要解决了如下技术难点

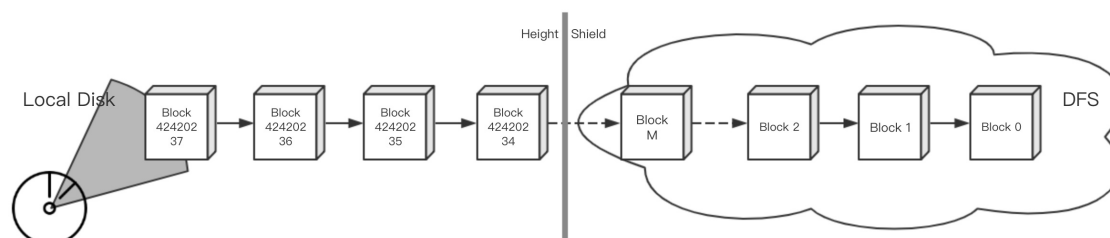
🔗 数据热度自适应存储

实现冷热数据自适应调度，将低频数据存储在SATA介质或者云存储，高频数据存储在SSD介质；实现DFS适配层，支持分布式文件系统，存储容量理论上可以扩展到PB级别。状态数据 状态数据一般位于块头，是链上各区块的索引数据。系统通过标记块头状态索引计数，将已被覆盖的状态索引踢出块头，从而保证状态数据量与高频覆写交易量解耦。其次将低频变更的状态数据分代迁移到成本更低的SATA介质或者云存储。高频变更的状态数据存放在内存和SSD介质的数据库中。同样，从数据库中读取状态节点时，本着最小IO开销的原则，仅需要读取那些需要用到的节点数据即可。根据读取频率，状态节点的索引路径也会根据热度进行打分，状态树被划分冷热区，冷区状态节点会迁移到成本更低的存储介质中。



🔗 交易数据

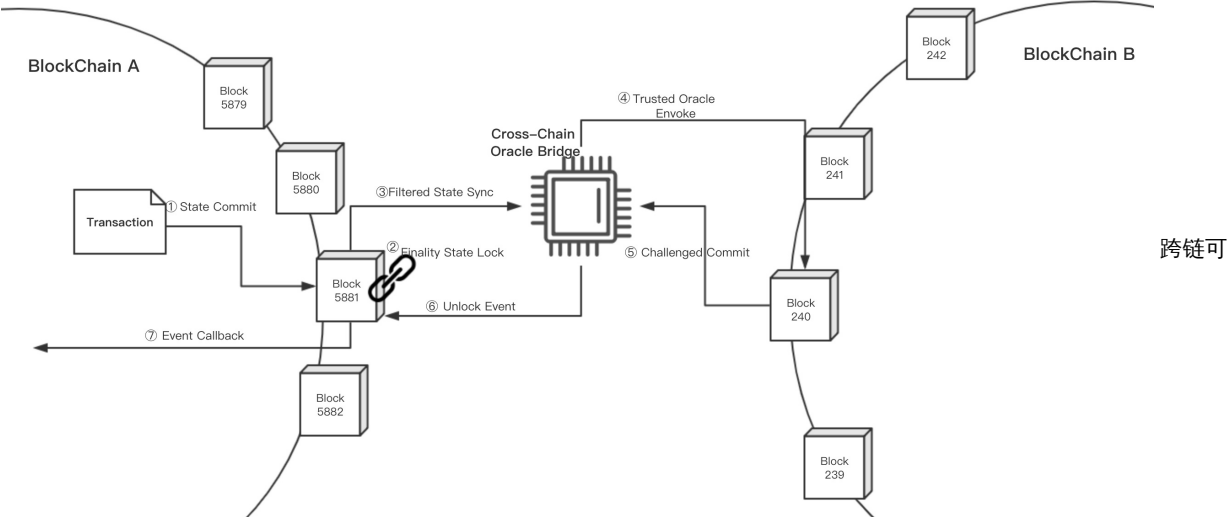
交易数据是区块中数据空间比重最大的部分。各个区块中都会包含交易信息。这些交易信息会在监督节点或新启动节点中被回溯验算。超级链BaaS平台系统会根据链长存储比，进行轻量级的计算。也就意味着当区块高度大于某阈值后，交易验算将只抽取一定比例的最新区块中。快速验算机制保证高度小于某阈值的区块将不需要参与验算，这些区块中的交易数据可以被归档保存。只需核心节点拥有这部分的数据。系统会将归档交易数据推送到DFS，当需要做全量审计或系统需要全量恢复数据时可以从归档DFS中读取。由于百度智能云DFS无容量上限，并且有成熟的可用性技术保障，归档后的交易数据可以从区块链中卸载。即使高度进一步增加，区块链节点本地存储的交易数据将被稳定在一个合理的空间范围内。



🔗 跨链可信交互

在一些复杂的应用场景中，跨链交互是一项可以扩展链网能力的功能。目前来看主要有两种用法：

1. 数据跨链打通，适用在同类或非同类链网间数据同步，比如为了做链网热切换，在不停服状态下将链网数据同步到另一条链网上；又或者为了做数据子集的有限共享，将主链上的交易数据脱敏同步到共享链网上。
2. 不同类型链网的功能互补，在当前的技术背景下，不同链网有不同功能优势。比如可以将隐私计算型链网与身份许可型联盟链（permissioned consortium）结合，前者用于在安全环境中进行加密计算，后者用于将前者的计算产出授权共享到联盟第三方节点。



信交互的技术难点包括绝对确定性（Absolute Finality）锚定与跨链预言机。超级链BaaS平台可在概率性确定共识场景中动态计算得到区块的稳固系数；同时也支持将概率性确定共识升级为绝对确定性共识。当区块足够稳定时，会被跨链预言机锚定。跨链预言机承担对接多链IO接口的作用，通过将预言机机制放大到多链上工作来实现。跨链预言机包括跨链协议转换、交易数据转义、状态数据过滤等逻辑组件。基于Plasma侧链交互模型，跨链交互分为锁定期、同步期、挑战期、解锁期。实验室状态下可在百毫秒内完成跨链数据的单周期交互。

通用智能合约

由于区块链上的数据具有不可篡改的特性，使得以太坊智能合约的开发需要异常的谨慎，智能合约程序所产生的bug无法轻易修复，并且可能产生巨大的代价。因此超级链BaaS平台针对智能合约提供了两大核心功能：

- 常用智能合约库
- 更多内置标准库

```
1 pragma solidity ^0.4.0;
2
3 import "./RBAC.sol"; // 常用智能合约库
4
5 contract User is RBAC {
6
7     function getUserAddress(string _rowData)
8     checkPermission("getUserAddress") // RBAC权限检查
9     public pure returns (string _userAddress) {
10         // 内置JSON标准库处理字符串
11         return json_get_string(json_parse(_rowData), "address");
12     }
13 }
```

常用智能合约库

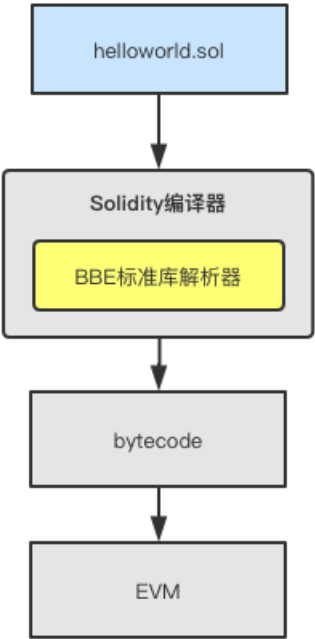
超级链BaaS平台提供常用的经过生产验证的智能合约库，包括数学计算、数组、链表、字符串操作、RBAC等。开源、模块化的设计方便用户二次开发。用户可以方便的一键部署进私有链中，如下图所示：

基础库部署

合约名称	合约简介	合约全局ID	操作
Ownable:v1.0	access包/Ownable	dce8437b-cdfd-44be-ae86-cada0e7b116e	部署
Whitelist:v1.0	access包/Whitelist	2a4b7c64-9bff-49cb-a7ad-aad76a5c5e0e	部署
Array8Lib:v1.0	array包/Array8Lib	aaa1be38-9e03-4abe-a779-3c1d9e275660	部署
Array16Lib:v1.0	array包/Array16Lib	4f1e18ea-e2c1-456f-b948-20707fcef506	部署
Array32Lib:v1.0	array包/Array32Lib	d3413ebd-2a01-4d97-8a14-1f7cc54de818	部署

更多内置标准库

超级链BaaS平台完全支持原生的Solidity合约开发，在此基础上，为了方便合约的开发，超级链BaaS平台增加了更多标准库的支持，包括JSON解析库、XML解析库、base64编解码等，使开发者把更多的精力放在业务的实现上。



合约安全审计

任何有意义的智能合约或多或少都存在错误，百度智能云区块链引擎不仅可以对智能合约进行词法分析、语法检查，还专门提供了智能合约安全检查服务，给用户提供全面的智能合约审计报告，大大减少了智能合约漏洞的出现，防患于未然，缩短了用户开发的周期并减轻了上线的压力。

超级链BaaS平台的代码分析器通过将智能合约解析成AST来进行合约的漏洞模型筛选，其中包含死循环、数值溢出等常见漏洞模式，如下图所示：

132

133 -

134

135

136

137

138

139

140

141 -

142

143

144

145

146

147

148

149 -

150

151

152

153

154

155 -

156

157

158

159

160

161

162

163 -

164

165

166

167

168

169

```
function _createThing(string _name, uint _dna) internal {
    require(msg.sender != address(0));
    uint id = things.push(Thing(_name, _dna, 1, uint32(now + cooldw
    thingToOwner[id] = msg.sender;
    ownerThingCount[msg.sender]++;
    NewThing(id, _name, _dna);
}

function _createThingForSB(string _name, uint _dna, address _owner)
    require(_owner != address(0));
    uint id = things.push(Thing(_name, _dna, 1, uint32(now + cooldw
    thingToOwner[id] = _owner;
    ownerThingCount[_owner]++;
    NewThing(id, _name, _dna);
}

function _generateRandomDna(string _str) private view returns (uint)
    uint rand = uint(keccak256(_str));
    return rand % dnaModulus;
}

// 每个玩家只能调用一次这个函数，即初始化拥有一个
function createRandomThing(string _name) public {
    require(ownerThingCount[msg.sender] == 0);
    uint randDna = _generateRandomDna(_name);
    randDna = randDna - randDna % 100;
    _createThing(_name, randDna);
}

// 临时暴露一个不安全的方法给管理员，用于初始化n个
function createRandomThingByAdmin(string _name, address _owner) publ
    uint randDna = _generateRandomDna(_name);
    randDna = randDna - randDna % 100;
    _createThingForSB(_name, randDna, _owner);
}
```

browser/ThingOwnership.sol:1:1: SyntaxError: Source file requires diffe✖
pragma solidity ^0.4.19;
^-----^

browser/ThingOwnership.sol:80:5: SyntaxError: Functions are not allowed ✖
function Ownable() public {
^ (Relevant source part starts here and spans across multiple lines)

browser/ThingOwnership.sol:80:5: Warning: This declaration shadows an ex✖
function Ownable() public {
^ (Relevant source part starts here and spans across multiple lines)
browser/ThingOwnership.sol:70:1: The shadowed declaration is here:
contract Ownable {
^ (Relevant source part starts here and spans across multiple lines).

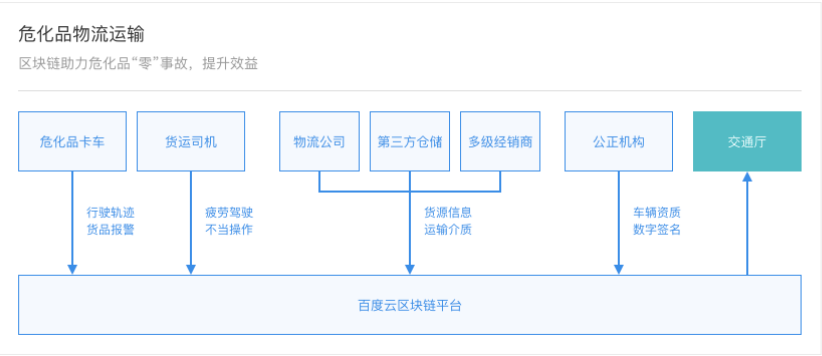
browser/ThingOwnership.sol:133:27: TypeError: Data location must be "st✖
function _createThing(string _name, uint _dna) internal {
^-----^

browser/ThingOwnership.sol:141:32: TypeError: Data location must be "st✖
function _createThingForSB(string _name, uint _dna, address _owner)
^-----^

应用场景

超级链BaaS平台服务目前已经以公有云、私有云的服务形式在多个行业进行商业落地，为客户提供可信存证、可信信息共享、可信金融、可信激励等多种服务。为金融、保险、物流、传媒、医疗、政务、零售、游戏等多种行业提供优质的区块链解决方案，如需为业务场景量身定制区块链解决方案，请于超级链BaaS平台的官网产品页首页提交服务申请，我们将根据您的业务场景，派专人与您进行沟通。部分解决方案案例如下：

物联网-危化品物流运输



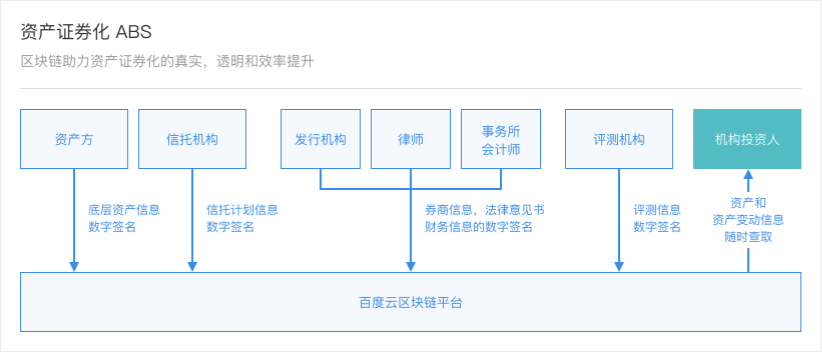
- 适合场景
 - 监管难度大，缺乏数据支撑
 - 从业人员专业度要求高
 - 资源信息不明确，易串/丢货，货物污染
 - 经销商链条不透明，高空载
- 特色
 - 安全芯片确保车载数据防伪造
 - 司机、卡车证照合规防篡改
 - 资源运输全程可溯源，易监管
 - 基于线路和数据的货运调度优化，提高运能和司机收入

金融催收



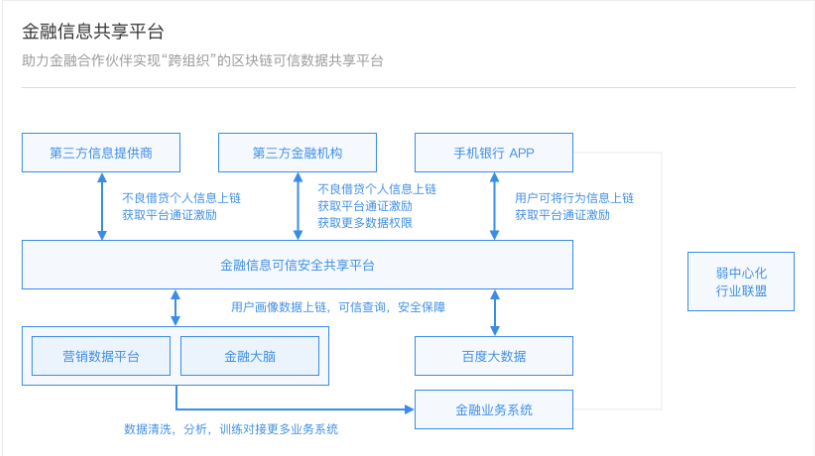
- 适合场景
 - “数据”资产化上链
 - “区块链资产”具备独特属性
 - “资产流转”多方参与
- 解决痛点
 - 差异化的资产包定义
 - 资金自动化清算结算
 - 数据真实透明安全，增强多方信任
 - 平台化运作，多分支机构接入

资产证券化ABS



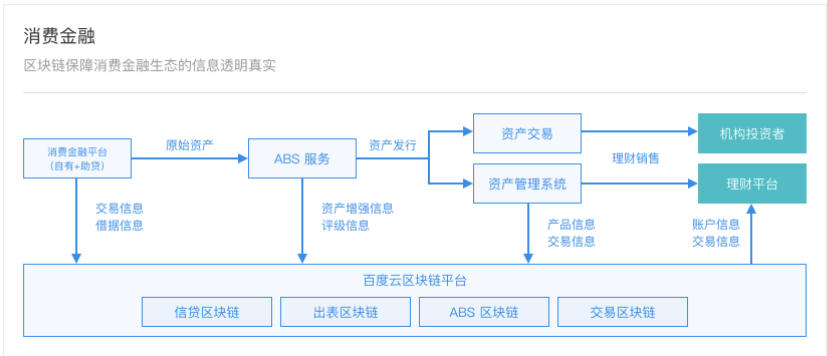
- 解决痛点
 - 资产生产，信息全透明
 - 资产记录，安全可溯源
 - 证券化效率提升
- 成果
 - 2017年百仟私募ABS
 - 2017年百度-长安新生，上交所认可的场内首单区块链ABS
 - 已支持500亿资产的真实验证

金融信息共享平台



- 解决痛点
 - 金融机构之间信息共享困难，数据利用率低
 - 数据分散在多个平台安全没有保障且数据易丢失
- 业务收益
 - 极大的提高用户数据的利用率
 - 用户画像数据上链实现可信查询、安全保障

消费金融



- 解决痛点
 - 信贷资产从产生到交易全流程真实透明
- 业务收益
 - 统一平台，支持多种区块链金融业务
 - 多链架构，实现业务定制和隔离

定价与购买

产品定价

产品定价

百度智能云超级链BaaS平台提供一键部署云上区块链网络的资源，允许用户从多种节点配置中进行选择。目前支持的主流框架包括：Hyperledger Fabric、企业以太坊Quorum、百度超级链XuperChain。本文介绍BaaS的计费规则等信息，计费项价格信息请参见 [产品价格详情](#)。

计费规则

后付费

- 计费模式：后付费
- 计费单位：元/分钟，最小单元为分钟
- 计费项：区块链网络配置资源、部署服务、存储空间

预付费

- 计费模式：预付费
- 计费单位：元/48小时
- 计费项：XuperChain试用网络使用时长

到期提醒和处理

- 到期提醒：任何区块链服务到期前2天、1天，系统均会给您发送即将到期提醒通知。
- 到期后处理：到期后立即停服，系统会发送欠费停服通知。区块链资源将为您保留7天，期间不收取费用，7天内未充值则释放，释放前1天和释放时系统都会发送释放通知。

选购指南

选购指南

百度智能云超级链BaaS平台提供区块链网络的搭建服务，支持Hyperledger Fabric、企业以太坊Quorum、百度超级链XuperChain三大主流框架。客户能够根据业务需求，灵活选择不同种类的服务类型与服务配置。

服务选择

超级链BaaS平台目前购买服务包括：

- 创建XuperChain试用网络
- 创建Fabric联盟网络
- 加入Fabric联盟网络
- 创建Quorum联盟网络
- 加入Quorum联盟网络
- 创建以太坊网络

🔗 资源配置

XuperChain试用网络

XuperChain试用网络提供创建试用网络的服务购买。试用网络默认提供资源配置实例如下：

系列	服务器规格	系统盘	CDS云盘	宽带
普通配置	4核/8GB	40G	1000G	5G

Fabric

Fabric联盟链提供两种服务购买；创建联盟，加入联盟网络。两种服务分别对应不同的资源配置实例。

服务类型	系列	服务器规格	节点数量	存储容量
创建联盟	普通配置	4核8GB	4节点	100GB
	豪华配置	16核32GB	6节点	500GB
加入联盟	普通配置	4核8GB	3节点	100GB
	豪华配置	16核32GB	3节点	500GB

Quorum

Quorum联盟链提供两种服务购买；创建联盟，加入联盟网络。两种服务分别对应不同的资源配置实例。

服务类型	系列	服务器规格	节点数量	存储容量
创建联盟	普通配置	4核8GB	4节点	100GB
	豪华配置	16核32GB	6节点	500GB
加入联盟	普通配置	4核8GB	3节点	100GB
	豪华配置	16核32GB	3节点	500GB

联盟网络资源配置系列区分

系列	说明	适用场景
普通配置	包含区块链的核心功能，提供入门级联盟链能力，支持基础级别业务需求	适合于个人开发者或中小企业用于验证系统能力或上线轻量级联盟业务
豪华配置	提供强大的联盟链能力，能够支撑大规模的业务需求，具备强大的事务处理能力与数据吞吐能力	适合于大、中型企业部署专业级别的联盟链，上线重要业务系统，同时满足企业对业务处理能力的高需求

以太坊

以太坊网络资源配置实例分为三类，详细的底层资源明细说明如下：

系列	服务器规格	系统盘	高性能磁云盘
普通配置	4核/8GB	50	50
高级配置	8核/16GB	50	300
豪华配置	12核/32GB	100	500

购买步骤

目前超级链BaaS平台开放华北区（北京）、华南区（广州）、华东区（苏州）、香港四地的服务购买。具体购买步骤如下：

- 1. 登录超级链BaaS平台控制台。
- 2. 选择服务类型，进行资源配置。
- 3. 配置资源参数，包括：

地域选区：建议选择与业务地理位置相近的选区。

节点参数：建议根据业务需求选择合理的节点资源。

购买时长：超级链BaaS平台实例资源的购买方式为包年包月的方式，可根据实际需求进行选购。

操作指南

XuperChain

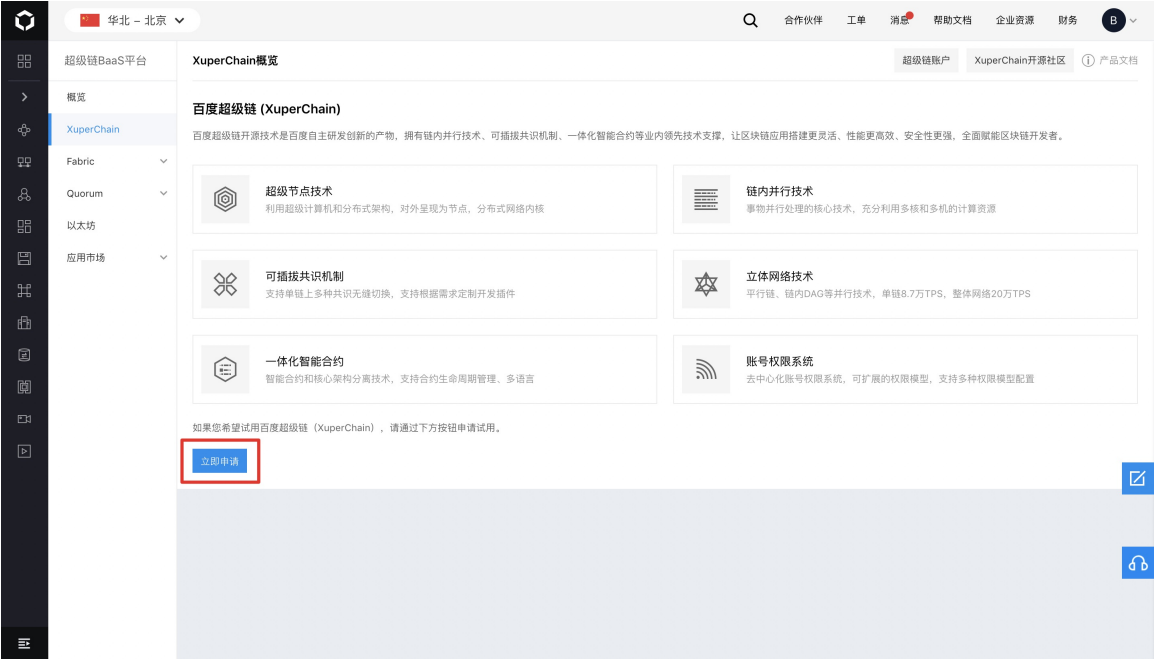
XuperChain试用网络

XuperChain试用网络

申请试用网络

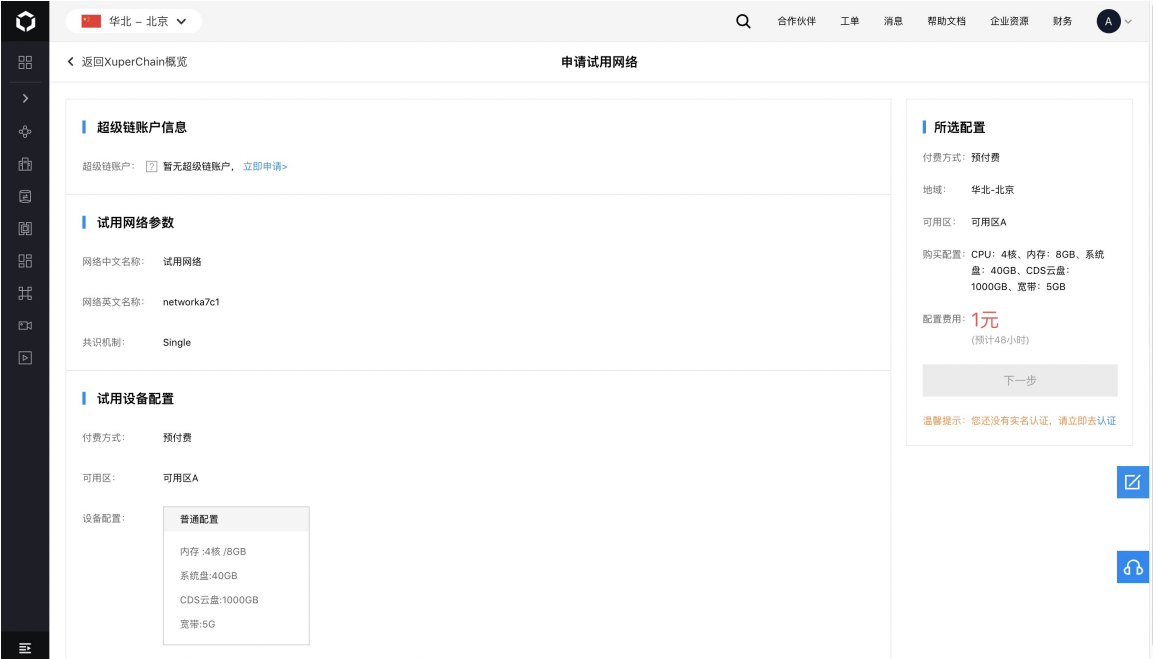
百度超级链开源技术是百度自主研发创新的产物，拥有链内并行技术、可插拔共识机制、一体化智能合约等业内领先技术支持，让区块链应用搭建更灵活、性能更高效、安全性更强，全面赋能区块链开发者。如果您希望试用百度超级链（XuperChain）框架搭建区块链网络，可在超级链BaaS平台控制台点击XuperChain，进入XuperChain概览页。

点击「立即申请」进行申请，进入申请试用网络页面。

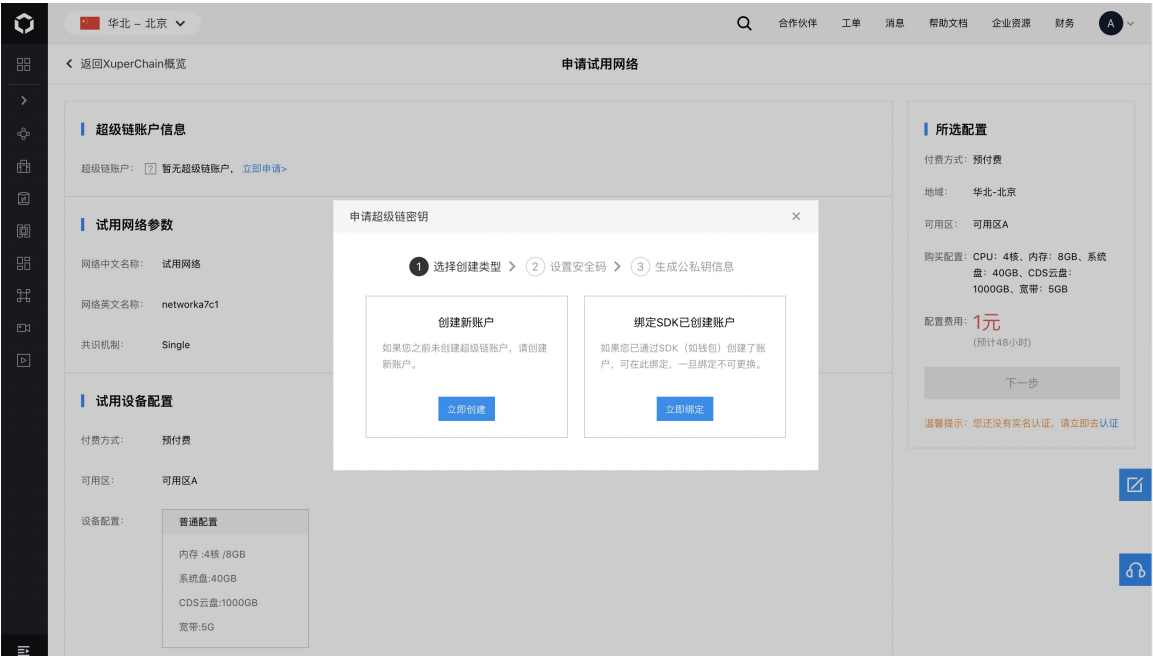


创建超级链账户

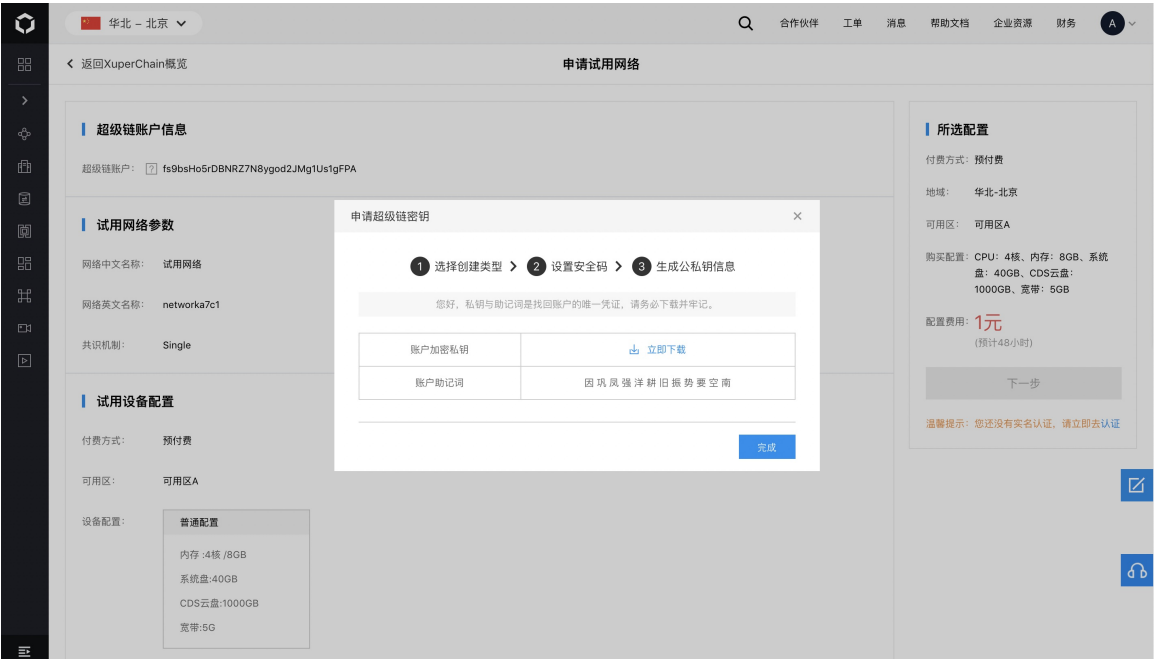
如果您尚未创建超级链账户，需要首先创建超级链账户。点击申请试用网络页「立即申请」链接：



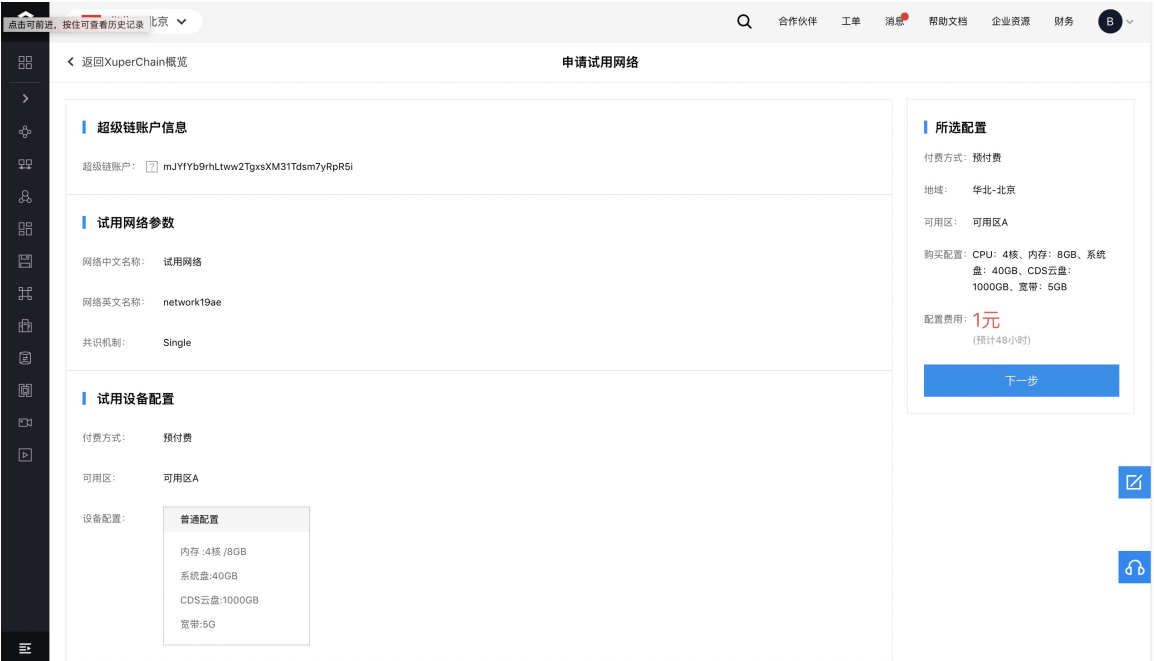
选择创建类型：



下载私钥与助记词：



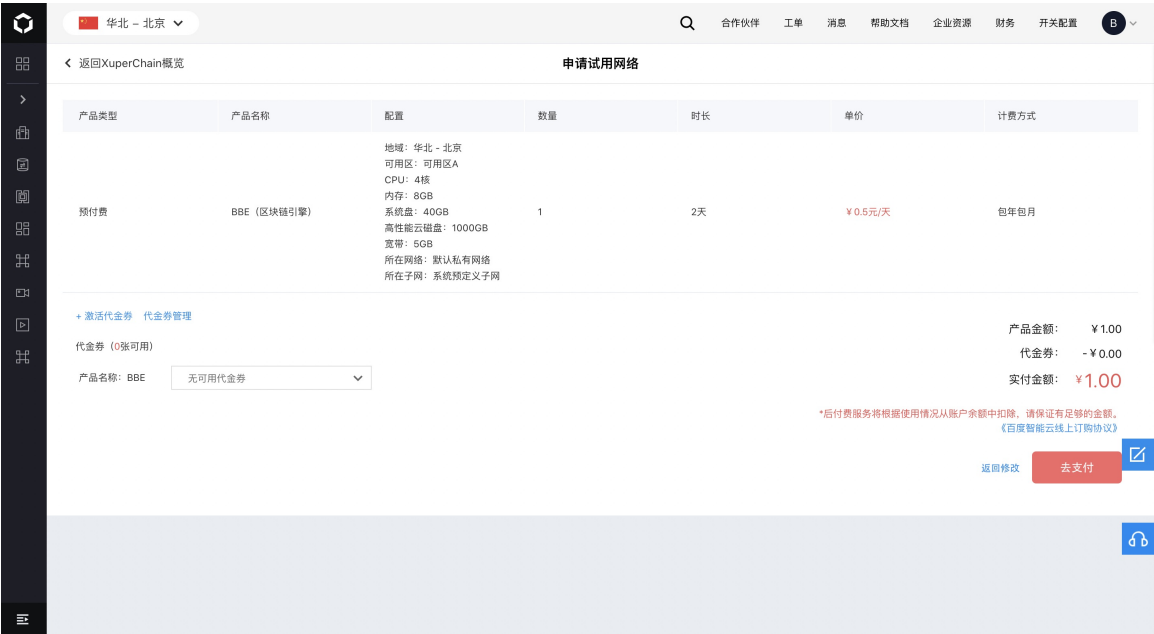
创建成功后，申请试用网络页「超级链账户信息」模块，显示超级链账户地址：



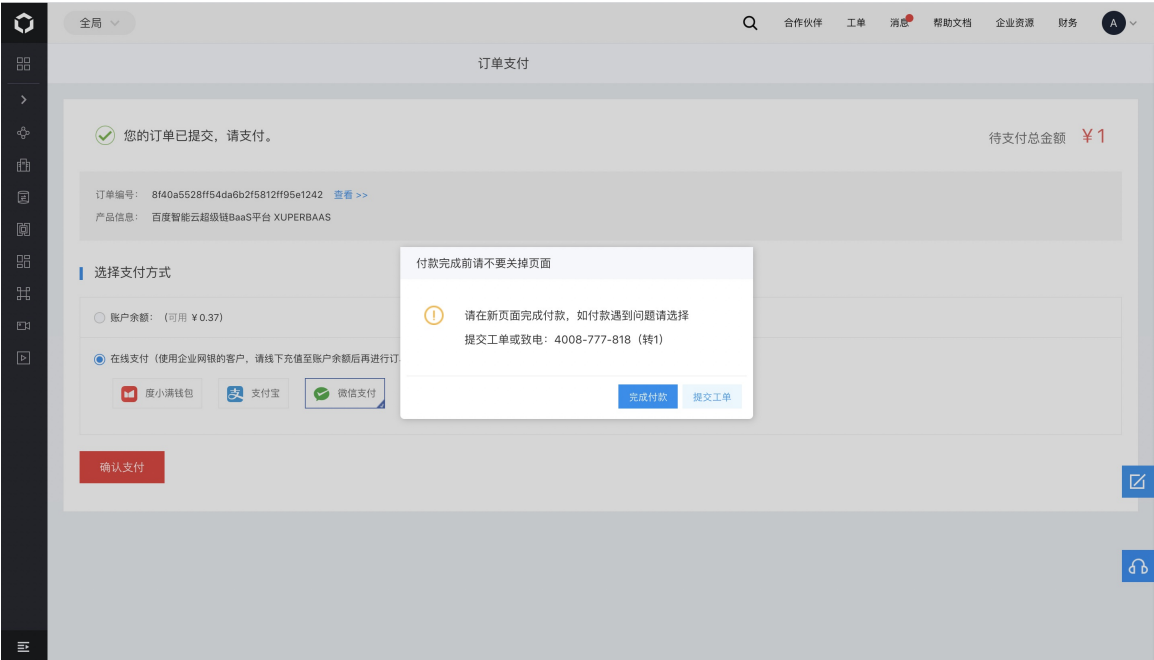
购买试用网络

XuperChain提供1元试用48小时的试用功能，您可以在申请试用网络页面，通过预付形式进行购买48小时的试用网络时间。

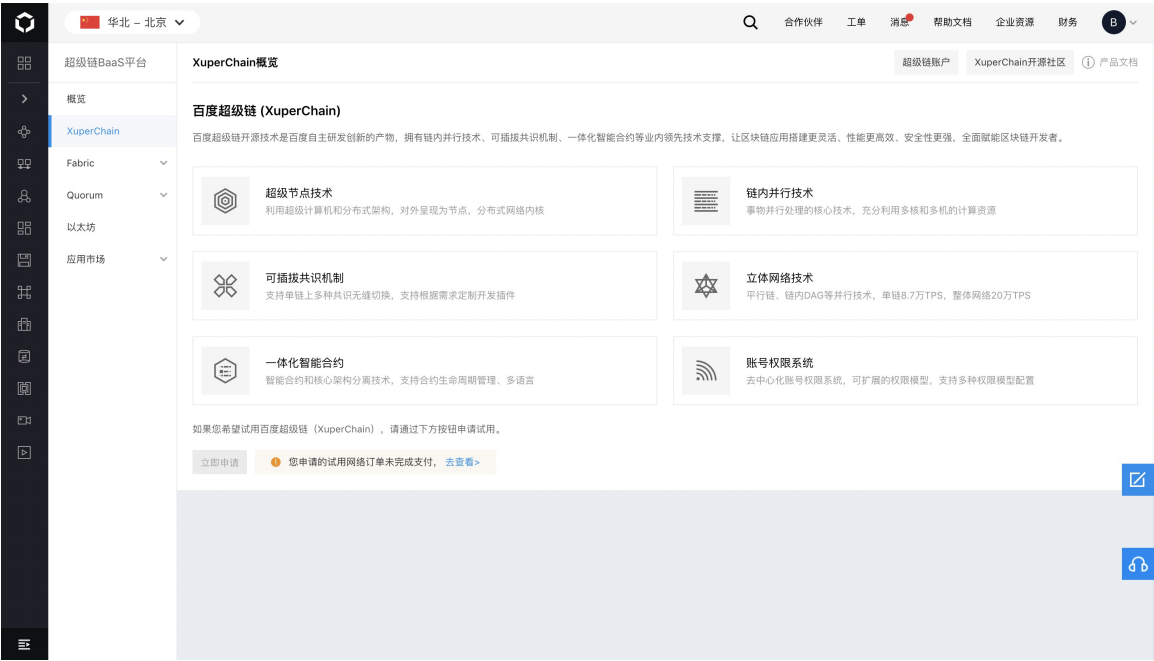
确认订单页面：



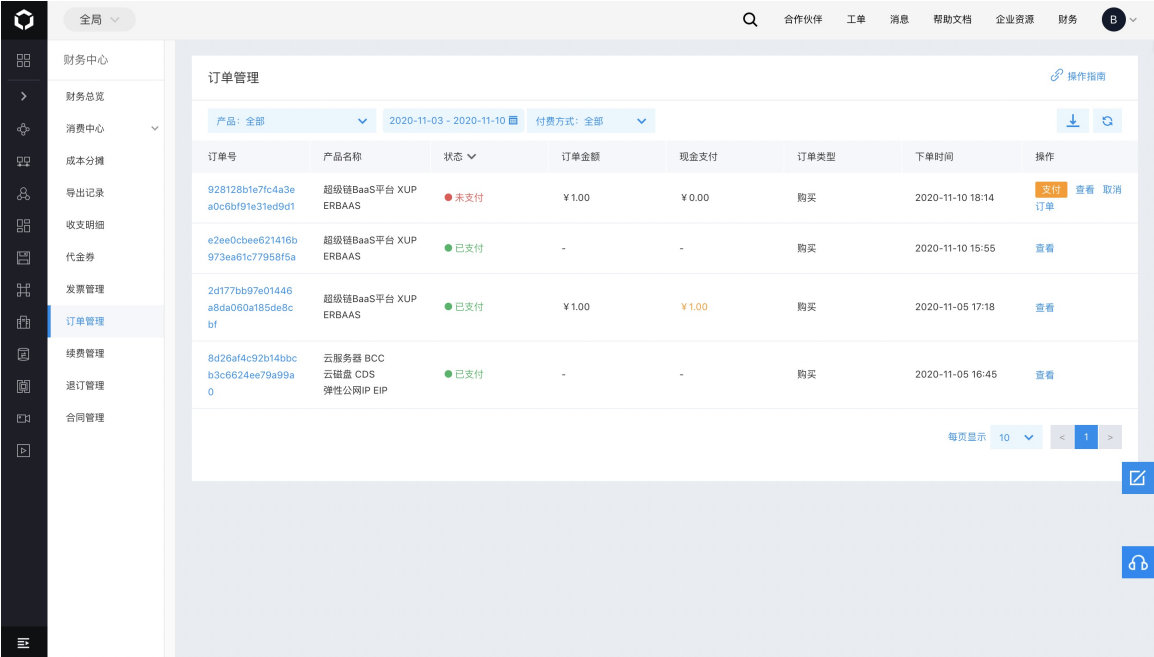
支付订单页面：



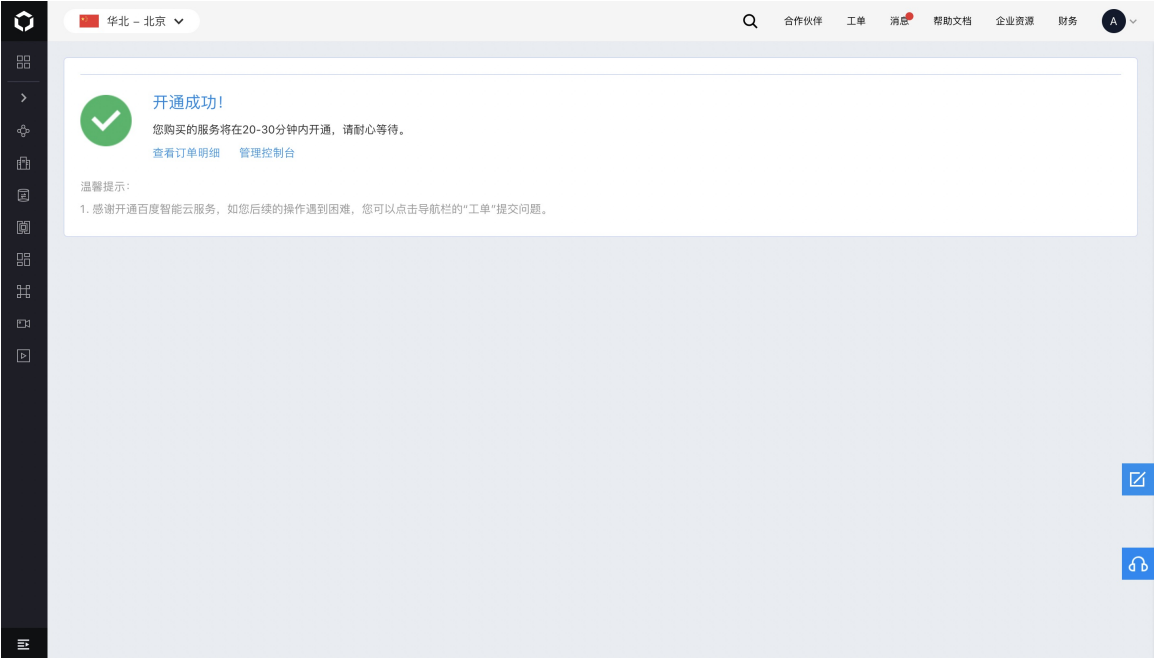
如果订单未支付成功, 则回到XuperChain概览页, 会提示待支付订单：



您可以在订单中心重新支付：

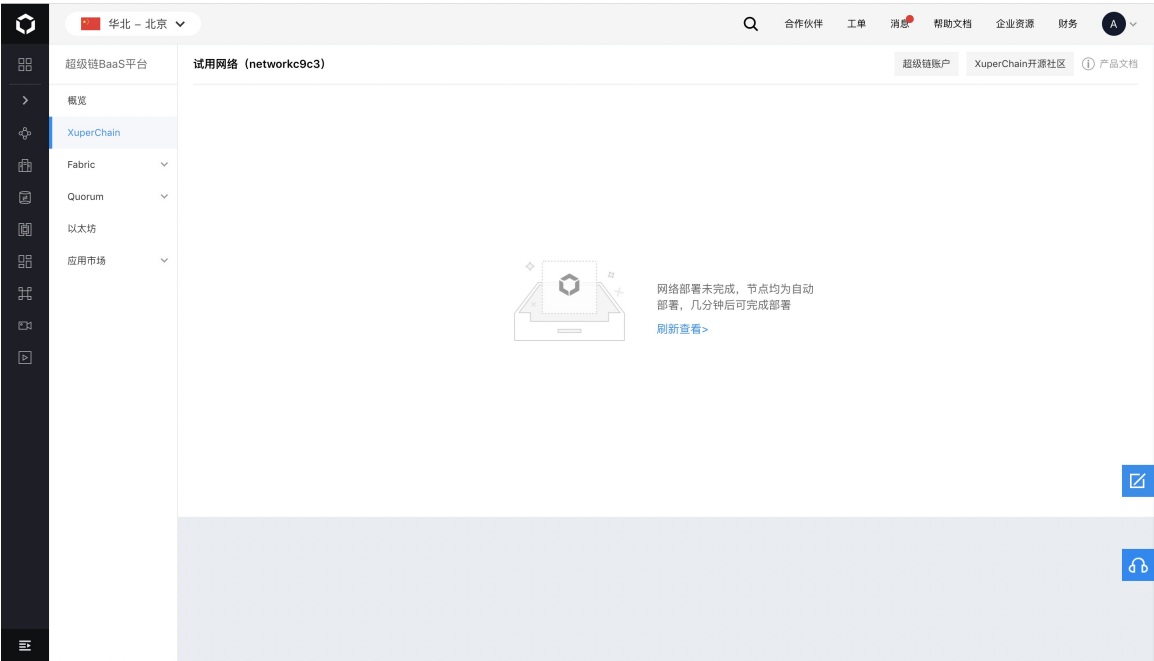


如果订单支付成功，则跳转到支付成功页面：

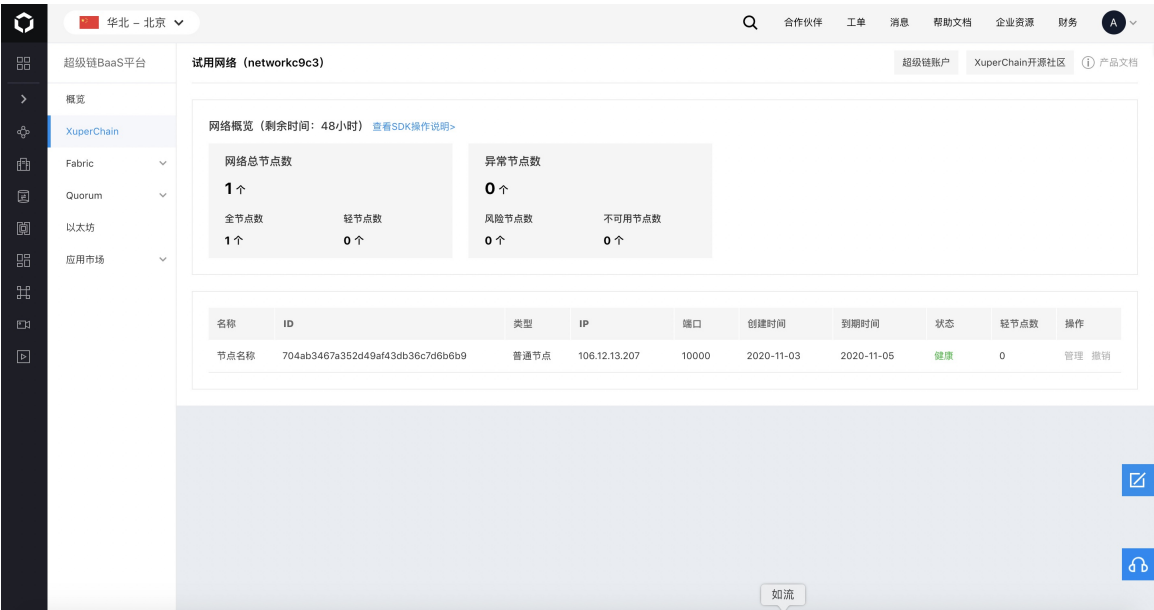


部署区块链网络

购买成功后，点击「管理控制台」，可回到XuperChain概览页，请您耐心等待，XuperChain需要一段时间进行网络部署：



部署成功后页面：



您可以试用48小时。

调用SDK

您可以通过点击「查看SDK操作说明」，可通过SDK进行试用。

查看节点信息

您可以通过查看节点列表，查看需要用到的节点信息。

Fabric联盟链

组织管理

组织管理

Fabric联盟由一个或多个组织组成。用户需要先创建组织，以组织的身份创建或加入联盟。一个组织可以加入多个联盟，但对于某个联盟只允许用户下一个组织加入

创建组织

- 1. 进入【组织】菜单，点击【创建组织】，用户即可快速创建组织。组织在未加入联盟阶段由百度云免费创建。

2. 创建组织时需要设置的参数：

参数	说明	举例
组织名称	用于联盟中组织间相互识别，在某个区域中唯一； 支持大小写字母、数字、中文以及_ / .特殊字符，长度1-20	金融监管方
英文名	用于生成fabric Mspld； 支持小写字母、数字以及-，必须以字母开头，长度1-16；	org1（其mspld默认为org1MSP）
组织描述	对该组织的描述	用于加入联盟“供应链联盟”所用组织



删除组织 只有当前组织未加入任何联盟时才可删除。

1. 进入【组织】菜单，将选中组织删除。

组织名称	英文名	所在联盟	组织内权限	创建时间	操作
测试组织	test-org		admin	2019-11-22 10:54:39	删除

联盟管理

联盟介绍

联盟介绍

联盟是由多个组织而成，联盟创建者即为**盟主**，加入联盟者为**成员组织**

fabric 架构

类别	名称
共识机制	Kafka
状态数据库	CouchDB
BatchTimeout	2s
BatchSize	{AbsoluteMaxBytes: 103809024, MaxMessageCount: 10, PreferredMaxBytes: 524288}
Orderer数量	4
每个组织peer数量	2

盟主和成员组织区别

1. 盟主为联盟创建者，拥有所有orderer节点

- 2. 成员组织为联盟加入者，必须由盟主邀请加入
- 3. 盟主在通道创建中具有最终审批权，必须由盟主审批才可创建
- 4. 成员组织可自愿退出联盟
- 5. 在所有成员组织退出联盟后，盟主才可删除联盟

联盟列表

- 1. 进入【联盟】菜单，可查看联盟列表
- 2. 根据组织筛选列表，可支持ALL查询：

+ 创建联盟

联盟名称	英文名	联盟成员	联盟状态	创建时间	盟主	
first	first	zh-test	运行中	2019-11-20 16:33:00	zh-test	组织 ALL zh-test 00 节点管理 通道管理 删除联盟
esb-test	esb-test	金融监管方, zh-test	运行中	2019-11-11 11:47:22	zh-test	联盟管理 成员管理 节点管理 通道管理 删除联盟

每页展示 10 < 1 >

功能介绍

- 联盟管理
 - 创建联盟
 - 加入联盟
- 节点管理
 - 节点监控
 - 区块链浏览器
- 通道管理
- 链码管理
 - 上传链码
 - 操作链码
 - 编辑链码
- sdk 应用

创建联盟

创建联盟

联盟是由多个组织而成，联盟创建者即为盟主，创建联盟前必须创建组织；联盟创建成功后开始计费

创建联盟主要包含以下操作：

- 创建联盟
- 购买集群
- 开始创建

创建联盟

1. 进入【联盟】菜单，点击【创建联盟】，用户即可快速创建联盟网络。
2. 创建联盟时需要设置的参数：

参数	说明	是否必须
初始组织	盟主的参与组织	是
联盟名称	支持大小写字母、数字、中文以及_ /特殊字符，不能以特殊字符开头，长度1-20	是
英文名	支持小写字母、数字以及-，必须以字母开头，长度1-16	是
联盟描述	对该联盟的描述	否
通道创建策略	联盟成员在创建新通道时遵循的创建规则。 Any(任一组织同意)：创建新通道时，只需任意参与组织的同意 All(全部组织同意)：创建新通道时，需要所有参与组织的同意。 联盟内所有通道的创立都需要经过盟主的审批批准，盟主审批未通过或不满足通道创建策略都无法创建新通道。	是
共识机制	kafka	是

创建联盟

初始组织

zfh-test

联盟名称

请输入联盟名称

支持大小写字母、数字、中文以及_ /特殊字符，不能以特殊字符开头，长度1-20

英文名

请输入联盟英文名

支持小写字母、数字以及-，必须以字母开头，长度1-16

联盟描述

请输入联盟描述

0/200

通道创建策略

ANY

共识机制

Kafka共识

购买集群

支付方式

后付费

可用区

可用区B

T

容器集群节点

普通配置

普通型III

内存：4核 / 8GB

系统盘：40GB

高性能云磁盘：100GB

豪华配置

普通型III

内存：16核 / 32GB

系统盘：40GB

高性能云磁盘：500GB

所选配置

清空配置

支付方式：后付费

地域：华东 - 苏州

可用区：可用区B

购买配置：CPU：4核、内存：8GB、系统盘：40GB

购买配额：可用区B：4台

配置费用：**¥ 0.1363/分钟** (按秒计费)

计 ¥ 196.27/天 ¥ 5888.10/月

下一步

购买集群 部署联盟链的机器配置

1. 购买集群参数

参数	说明	是否必须
付费方式	暂时只支持“后付费”	是
可用区	是指在同一区域下，电力和网络相互独立的区域 只可选一个区	是
容器集群节点	目前包括两种配置（见集群配置）	是

2. 集群配置及选购意见可参考 [选购指南](#)

开始创建 联盟创建一般需要10分钟左右，请您耐心等待。联盟创建成功后，联盟状态变为运行中。

加入联盟

加入联盟 联盟创建成功之后，联盟内成员能够邀请百度云上用户加入联盟；加入联盟后成为联盟的成员组织；**盟主和成员组织必须在同一区，当成功加入联盟后开始计费。**

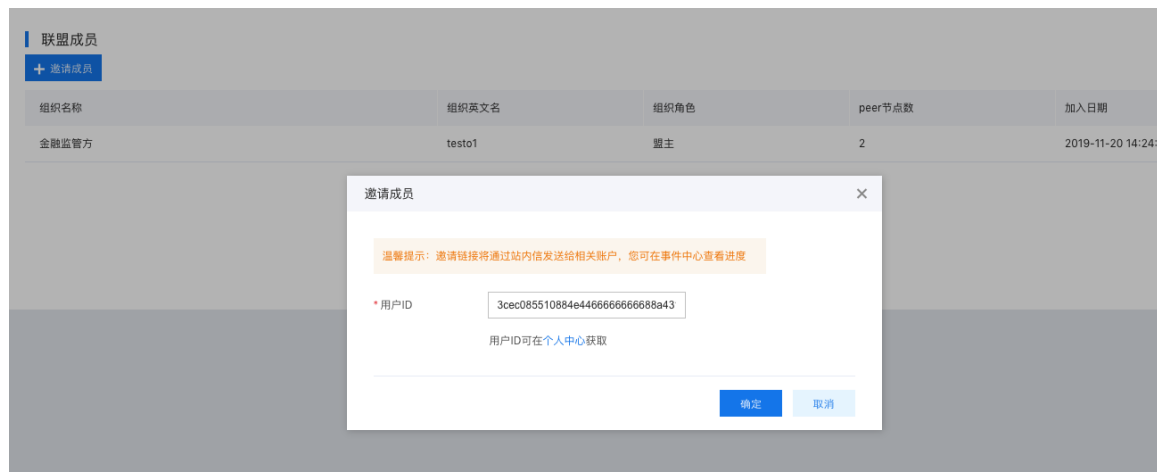
加入联盟主要包含以下操作：

24

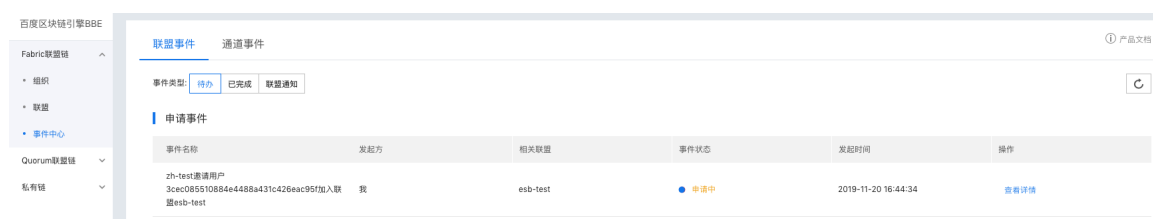
- 邀请加入
- 用户审批
- 购买集群
- 开始加入

邀请加入

1. 进入【联盟】菜单，点击【成员管理】进入【联盟成员页】，点击【邀请成员】，在弹出窗口中填写被邀请用户的百度云账号id，邀请以及邀请链接将以站内信的形式发送给相应用户。



2. 邀请方可在【事件中心】中【联盟事件】-【申请事件】查看当前邀请进程（具体操作可查看[事件中心](#)）：



用户审批

1. 被邀请用户会在【消息】中收到邀请通知：



2. 根据通知内容选择响应区域，在【事件中心】中【联盟事件】-【审批事件】中可查看加入联盟的邀请：

zh-test邀请用户 3cec085510884e4488a431c426eac95f加入联盟 盟esb-test	zh-test	esb-test	●●● 审批中	2019-11-20 16:44:34	接受 拒绝
--	---------	----------	---------	---------------------	-------

3. 点击【接受】后可再次点击【确认加入联盟】开始加入联盟操作：

zh-test邀请用户 3cec085510884e4488a431c426eac95f加入联盟 盟esb-test	zh-test	esb-test	● 审批通过	2019-11-20 16:44:34	确认加入联盟
--	---------	----------	--------	---------------------	--------

4. 加入联盟时需要设置的参数：

参数	说明	是否必须
初始组织	加入联盟所需组织	是

加入联盟

选择组织：

zh-test

组织配置：2个节点

CA部署方式：ECDSA

证书加密方式：平台托管

购买集群

支付方式：

后付费

可用区：

可用区B

容器集群节点：

普通配置

普通型III
内存：4核 / 8GB
系统盘：40GB
高性能云磁盘：100GB

豪华配置

普通型III
内存：16核 / 32GB
系统盘：40GB
高性能云磁盘：500GB

所选配置

清空配置

支付方式：后付费

地域：华东 - 苏州

可用区：可用区B

购买配置：CPU：4核、内存：8GB、系统盘：40GB

购买配额：可用区B：3台

配置费用：**¥0.08945/分钟**
(预计 ¥128.81/天 ¥3864.30/月)

有优惠

下一步

购买集群 部署联盟链的机器配置

1. 购买集群参数

参数	说明	是否必须
付费方式	暂时只支持“后付费”	是
可用区	是指在同一区域下，电力和网络相互独立的区域 只可选一个区	是
容器集群节点	目前包括两种配置（见集群配置）	是

2. 集群配置及选购意见可参考 [选购指南](#)

开始加入 加入联盟一般需要10分钟左右，请您耐心等待。加入联盟成功后，联盟状态变为运行中。

节点管理

节点监控

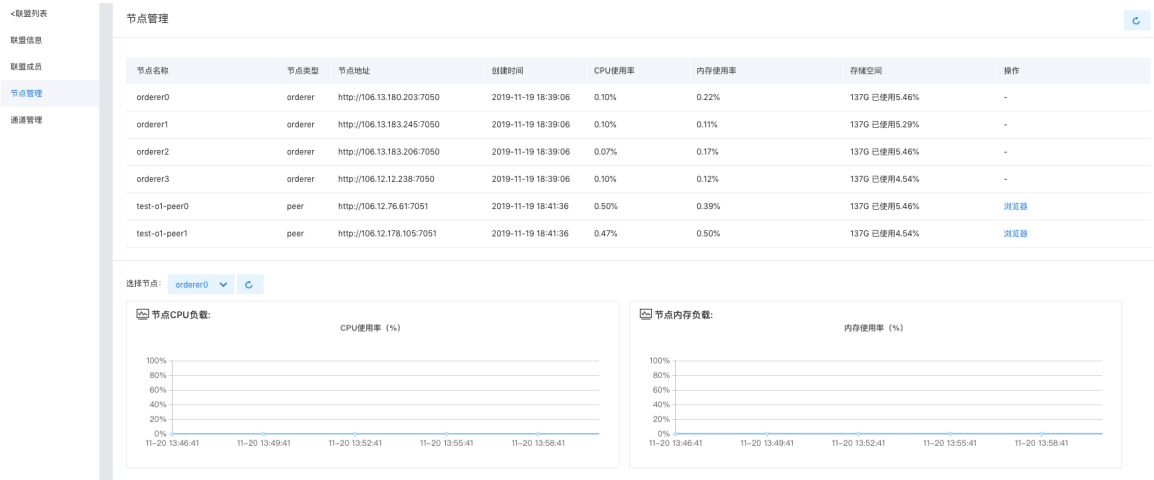
节点监控

本文档主要引导用户对节点磁盘、内存和cpu等使用信息进行监控

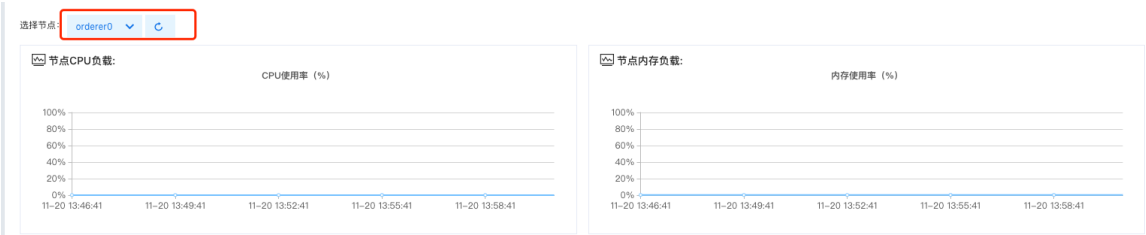
节点查看

1. 进入【联盟】菜单，点击【节点管理】进入【节点管理】页面，可看到在该联盟下所拥有节点的基本信息，包括节点地址、cpu/内存/磁盘使用率等等。

26



cpu和内存使用率 在【节点管理】页面中，可以通过切换来查看不同节点cpu和内存实时的使用率；对于单个节点，左上角提供刷新按钮



区块链浏览器

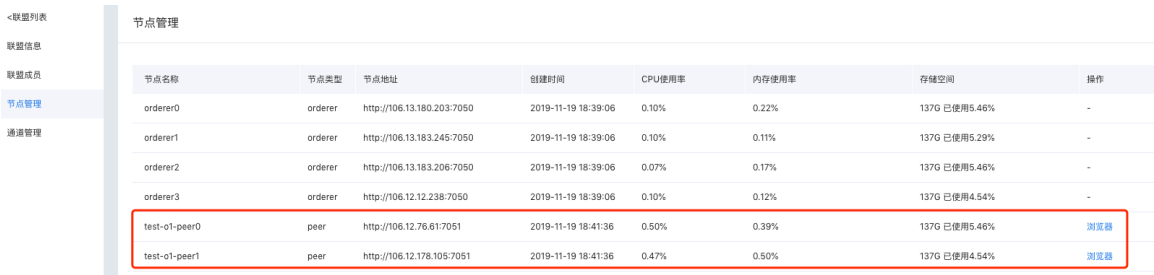
区块链浏览器 区块链浏览器不仅提供区块链网络、区块、交易和链码等统计信息，也提供区块和交易的详细信息。

区块链浏览器主要包含以下信息：

- 浏览器入口
- 浏览器状态
- 浏览器介绍

浏览器入口

1. 进入【联盟】菜单，点击【节点管理】进入【节点管理】页面
2. 仅peer类型的节点才有浏览器入口



浏览器状态 浏览器有三种状态

状态	原因	显示
未创建	当用户未加入任何一个通道时	图一
正在创建	当用户成功加入一个通道后，开始创建浏览器	图二
创建成功	当用户成功加入一个通道后，创建浏览器成功	图三

testo1-peer0	peer	http://106.13.180.200:7051	2019-11-20 14:40:43	1.05%	0.40%	137G 已使用3.51%	您还未加入任何通道，请加入通道后查看
testo1-peer1	peer	http://106.12.60.66:7051	2019-11-20 14:40:43	0.83%	0.40%	137G 已使用3.91%	浏览器

图一

testo1-peer0	peer	http://106.13.180.200:7051	2019-11-20 14:40:43	1.05%	0.40%	137G 已使用3.51%	浏览器正在创建中，请稍后查看
testo1-peer1	peer	http://106.12.60.66:7051	2019-11-20 14:40:43	0.83%	0.40%	137G 已使用3.91%	浏览器

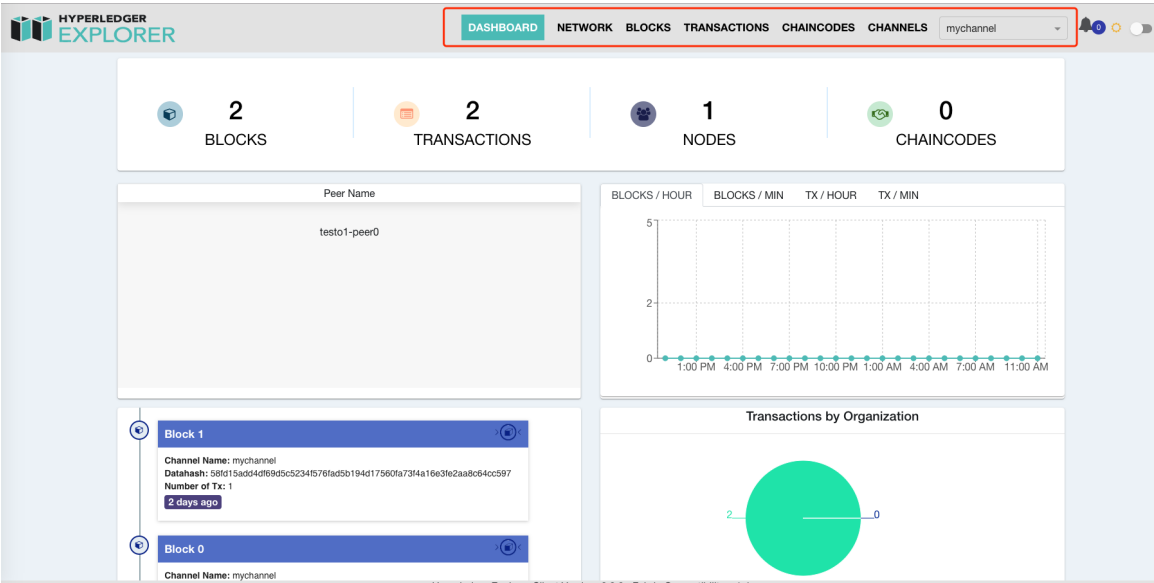
图二

testo1-peer0	peer	http://106.13.180.200:7051	2019-11-20 14:40:43	0.78%	0.57%	137G 已使用4.46%	浏览器
testo1-peer1	peer	http://106.12.60.66:7051	2019-11-20 14:40:43	0.80%	0.54%	137G 已使用3.91%	浏览器

图三

浏览器介绍 浏览器支持多通道切换，主要分为六个模块：

名称	说明
DASHBOARD	区块链网络、区块、交易和链码等统计信息 根据时间，打包区块和交易的情况
NETWORK	当前浏览器所对应节点
BLOCKS	检索区块
TRANSACTIONS	检索交易
CHAINCODES	检索当前运行链码
CHANNELS	检索已加入的通道



通道管理

通道管理

同一联盟中的成员能够加入到联盟中的任意通道。联盟成员通过两种形式加入到通道中：

- 1. 创建通道并邀请其他联盟成员加入
- 2. 联盟成员申请加入某通道

通道管理主要包含以下操作：

- 通道管理
- 创建通道

- 加入通道

管理页入口

- 进入【联盟】菜单，选中联盟点击【通道管理】进入【通道管理】页面
- 在【通道管理】页中可进行创建通道、查看已加入通道列表、查看未加入通道列表、加入通道和进入链码管理页操作

通道管理

+ 创建通道

已加入

未加入

通道名称	通道描述	发起组织	通道成员	通道状态	操作
channel1	channel1	zh-test	zh-test	● 运行中	申请加入

通道管理

+ 创建通道

已加入

未加入

通道名称	通道描述	发起组织	通道成员	通道状态	加入时间	操作
mychannel	mychannel	金融监管方	金融监管方	● 运行中	2019-11-20 15:17:23	链码管理

创建通道 创通创建需要三个流程：

- 创建通道申请
- 审批
- 创建通道

创建通道申请

- 在盟主创建联盟时，指定了创建通道的规则（只有一个）；

规则名称	规则说明
Any	创建新通道时，只需任意参与组织的同意
All	创建新通道时，需要所有参与组织的同意

- 不论何种规则，联盟内所有通道的创立都需要经过盟主的审批批准

3. 创建通道参数

参数	是否必须	说明
通道名称	是	支持大小写字母、数字、中文以及_/.特殊字符，不能以特殊字符开头，长度1-20
通道英文名称	是	支持小写字母、数字以及-，必须以字母开头，长度1-16
加入组织	是	在通道创建时，能够同时邀请同联盟中的其他组织成员加入通道
通道描述	是	对通道业务、通道特征的描述



4. 发出申请后，可在【事件中心】中【通道事件】-【申请事件】查看进度：

组织金融监管方申请创建通道mychannel1	我	mychannel1	●● 申请中	2019-11-21 16:26:53	查看详情
-------------------------	---	------------	--------	---------------------	----------------------

5. 受邀请组织将收到站内信通知

审批

1. 受邀请组织可在【事件中心】中【通道事件】-【审批事件】进行审批：

事件名称	发起方	相关通道	事件状态	发起时间	操作
组织金融监管方申请创建通道mychannel1	金融监管方	mychannel1	●● 审批中	2019-11-21 16:26:53	通过 驳回

创建通道

1. 审批完成后，通道处于“待启动”状态，点击“创建通道”即可

mychannel1	mychannel1	金融监管方	金融监管方,zh-test	●● 待启动	2019-11-21 16:26:53	创建通道
------------	------------	-------	---------------	--------	---------------------	----------------------

加入通道 加入通道有两种方式：

- 创建通道时，加入组织选择邀请组织（参见上）
- 加入“未加入”通道，过程如下：

- 点击“未加入”选框
- 选中通道，点击“申请加入”，这时通道内成员会收到您申请加入的申请

已加入		未加入			
通道名称	通道描述	发起组织	通道成员	通道状态	操作
channel1	channel1	zh-test	zh-test	运行中	申请加入
dev	dev	zh-test	zh-test	运行中	申请加入
mychannel	mychannel	zh-test	zh-test	运行中	申请加入

- 您可在【事件中心】中【通道事件】-【申请事件】查看进度
- 等待通道内成员审批通过后
- 您可在【事件中心】中【通道事件】-【申请事件】操作加入通道

组织金融监管方申请加入通道channel1	我	channel1	● 申请通过	2019-11-21 16:39:33	确认加入通道 查看详情
-----------------------	---	----------	--------	---------------------	---

通道详情 基本介绍 您可通过点击通道名称进入通道详情页：

已加入 未加入

通道名称	英文名	通道描述	发起组织	通道成员	通道状态	加入时间	操作
channel1	channel1	channel1	org1	org1	● 运行中	2020-06-17 10:32:23	链码管理

通道详情页分为“基本信息”和“通道成员”两部分：

< channel1 ● 运行中

基本信息

UUID: 5e6aa006-8bf6-a210-a94c-70bd39a2da20
发起时间: 2020-06-17 10:32:23
通道描述: channel1

所属联盟: -q
出块大小: 20
发起组织: org1
出块超时时间: 2000ms

通道成员

组织名称	英文名	组织描述	加入时间
org1	org1	张虹的测试组织1	2020-06-17 10:32:23

修改通道配置 您可以在【通道详情】-【基本信息】中修改通道的出块配置

- 出块大小：每个区块包含最多交易笔数，当并发要求比较高时可以调大该值；
- 出块超时时间：打包区块的超时时间，及时性要求较高时可以调小该值；

链码管理

上传链码

上传链码 在部署链码前需要提交链码元数据，包括文件、名称等等。上传链码有两种途径：

- 链码管理页中上传
- 链码编辑器中上传

本篇主要介绍“链码管理页中上传”方式，也是分为两个步骤：

- 压缩链码文件
- 上传链码

链码管理页 进入【联盟】菜单，选择联盟点击【通道管理】，选择对应通道点击【链码管理】进入链码管理页

压缩链码文件 暂时只支持go语言版本链码，链码引用的第三方库均放入vendor中，然后将链码外层文件夹打包成tar.gz或zip 压缩命令

- zip -r **.zip chaincode
- tar -czvf **.tar.gz chaincode

上传链码

- 进入【链码管理】页，点击【新增链码】进行链码上传

新增链码

* 通道名称

dev

* 链码名称

请输入链码名称

支持小写字母、数字以及-，不可以-开头和结尾，长度1-20

* 版本

请输入版本

要求两位版本号，格式如1.3

* 实例化参数

请输入实例化参数

init + 参数，如：init,a,100,b,200

背书策略

链码所关联的背书策略，例如OR("Org1MSP.member";"Org2MSP.member"),
其中Org1,Org2为组织英文名
如果为空则采用默认背书策略，即通道内任一成员背书

* 代码文件

请选择文件

只支持go语言链码，链码引用的第三方库放入vendor中。将链码外层文件夹打包成tar.gz或zip
操作命令如(cd /src && tar czf chaincode.tar.gz *)

确认

2. 上传链码参数

参数	是否必须	说明
链码名称	是	支持小写字母、数字以及-，不可以-开头和结尾，长度1-20
链码版本	是	要求两位版本号，格式如1.3
初始化参数	是	链码的具体执行参数信息
背书策略	否	链码所关联的背书策略，填写方式见下文“背书策略” 如果为空则为默认背书策略，即通道内任何成员背书即可
代码文件	是	即第一步压的压缩包

3. 上传成功后，多出一条链码条目：

链码名称	版本	初始化参数	背书策略	提交时间	部署时间	链码状态	操作
mycc	1.0	init,a,100,b,200	OR("Org1MSP.member";"Org2MSP.member")	2019-11-21 18:32:07	2019-11-21 18:32:07	● 未发布	发布到通道 编辑链码

背书策略 支持 and or outof 三种表达式（大小写均可），示例：

- and('Org1MSP.member','Org2MSP.member') 表示需要Org1和Org2的成员共同背书
- or('Org1MSP.member','Org2MSP.member') 表示Org1和Org2任何一个组织成员背书即可
- outof(2,'Org1MSP.member','Org2MSP.member','Org3MSP.member') 表示需要Org1、org2和Org3中任意两个组织成员共同背书

我们提供了简便的背书书写方式，输入框会自动弹出组织信息，用户选择即可(组织间用英文逗号隔离)，如下：

背书策略

OR(ding-org.member x)

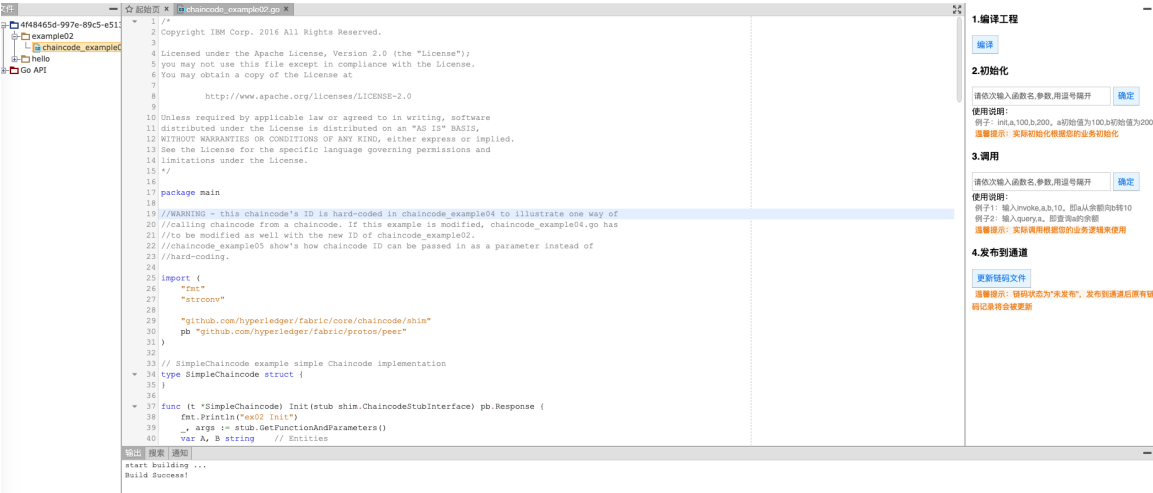
链码所关联的背书策略，例如C ding-org > member \$P.member"),
其中Org1,Org2为组织英文名
如果为空则采用默认背书策略，即通道内任一成员背书

编辑链码

编辑链码 超级链BaaS平台提供前端编辑器，主要用于用户编辑、测试链码。更重要的是在安装其他用户上传的链码前审查链码。

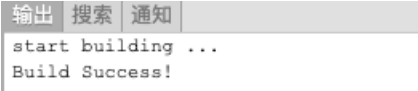
编辑完链码必须进行编译，只有通过编译的链码才可进行后续测试、上传链码或修改链码文件等操作 创建链码

- 1. 进入【链码管理】页面，点击【链码在线编辑器】
- 2. 跳转到在线编辑器页面，需要输入用户AK、SK
- 3. 进入后添加目录和文件，编写链码：

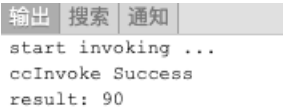
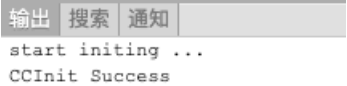


- 4. 编写完的链码可在线进行编译和测试：

- 编译链码，再底部会显示编译信息：



- 测试链码，包括初始化和调用操作，也会再底部显示初始化和调用结果：



- 5. 编辑完的链码可以下载到本地或者直接上传链码（与[上传链码](#)相似，最终新增一条链码条目）

编辑已有链码

- 1. 进入【链码管理】页面，选中链码点击【编辑链码】（链码上传者显示“编辑链码”，非上传者显示“查看链码”）：

链码名称	版本	初始化参数	背书策略	提交时间	部署时间	链码状态	操作
cccc	1.0	init_a,100,b,200	OR("zh-testMSP.member")	2019-11-12 15:16:04	2019-11-12 15:17:15	运行中	编辑链码

- 2. 跳转到在线编辑器页面，需要输入用户AK、SK
- 3. 可在线进行编译和测试，校验链码是否符合预期（参考上）
- 4. 对于“未发布”（链码状态可查看[操作链码](#)）的链码可进行编辑，然后替换旧链码文件

1.编译工程

编译

2.初始化

init,a,100,b,200

确定

使用说明：

例子：init,a,100,b,200。a初始值为100,b初始值为200

温馨提示：实际初始化根据您的业务初始化

3.调用

query,a

确定

使用说明：

例子1：输入invoke,a,b,10。即a从余额向b转10

例子2：输入query,a。即查询a的余额

温馨提示：实际调用根据您的业务逻辑来使用

4.发布到通道

更新链码文件

温馨提示：链码状态为“未发布”，发布到通道后原有链码记录将会被更新

操作链码

操作链码 一个用户上传链码并将链码发布到通道后，通道内的组织均可安装部署该链码

链码状态

状态名称	说明	可进行的操作
未发布	上传链码后的状态 此时链码只能由上传者看到	编辑链码并替换链码文件 发布链码
初始状态	上传者将链码发布到通道后的状态 此时通道内的组织均可看到该链码 此时上传者再次编辑链码不可更新链码文件，只能重新上传链码	安装链码
已经安装	将链码安装到当前组织所有peer节点上	实例化链码/升级链码
运行中	实例化/升级链码成功后状态	
停止运行	被升级后的链码状态	

发布链码 用户上传完链码，此时链码只有用户自己看到，通道内其他组织不可见；此时用户可通过【编辑链码】对链码进行修改；发布链码后，链码对通道内所有组织成员可见，用户不可再对链码进行编辑，发布操作如下：

mycc1	1.0	init,a,100,b,200	OR("Org1MSP.member";"Org2MSP.member")	2019-11-21 18:32:44	2019-11-21 18:32:44	● 未发布	发布到通道 编辑链码	安装链码
-------	-----	------------------	---------------------------------------	---------------------	---------------------	-------	------------	------

1. 当链码被发布到通道后，可进一步执行“安装”操作：

mycc1	1.0	init,a,100,b,200	OR("Org1MSP.member";"Org2MSP.member")	2019-11-21 18:32:44	2019-11-21 18:32:44	● 初始状态	安装 编辑链码
-------	-----	------------------	---------------------------------------	---------------------	---------------------	--------	---------

2. 至于安装后链码的状态有两种情况：

- 如果链码未被通道内其他组织实例化，链码状态显示“已经安装”
- 如果链码已经被通道内其他组织实例化，链码状态显示“运行中”，因为通道内链码只可被实例化一次 **实例化链码** 安装完链

码，如果链码状态为“已经安装”则可进一步进行实例化操作，如下：

mycc1	1.0	init,a,100,b,200	OR("Org1MSP.member";"Org2MSP.member")	2019-11-21 18:32:44	2019-11-21 18:32:44	● 已经安装	实例化 编辑链码
-------	-----	------------------	---------------------------------------	---------------------	---------------------	--------	----------

升级链码 升级是针对已经在运行的同名链码而言，也就是当前必须有链码是在运行中状态。链码升级前需要上传同名高版本的链码（[上传链码](#)）；之后进行发布到通道和安装操作，安装后即可显示升级按钮，点击“升级”完成链码升级：

mycc	2.0	init,a,100,b,200	OR("testo1MSP.member")	2019-11-21 20:25:23	2019-11-21 20:26:35	● 运行中	编辑链码
mycc	3.0	init,a,100,b,200	OR("testo1MSP.member")	2019-11-21 20:27:10	2019-11-21 20:27:10	● 已经安装	升级 编辑链码

被升级的链码状态更改为“停止运行”，同一通道只能有一个同名链码在运行，如下：

mycc	2.0	init,a,100,b,200	OR("testo1MSP.member")	2019-11-21 20:25:23	2019-11-21 20:26:35	● 停止运行	
mycc	3.0	init,a,100,b,200	OR("testo1MSP.member")	2019-11-21 20:27:10	2019-11-21 20:29:35	● 运行中	编辑链码

事件中心

联盟事件

联盟事件

fabric 联盟链中有很多申请、审批的事件流，关于联盟的事件流在联盟事件中

申请事件

1. 当盟主邀请成员加入联盟时，会添加一条申请事件，具体可看 [加入联盟](#)

申请事件					
事件名称	发起方	相关联盟	事件状态	发起时间	操作
test2邀请用户q加入联盟sasatest2	我	sasatest2	●● 申请中	2019-07-22 07:10:57	查看详情

- 2.当“申请事件”处理完，会归档到“已完成”

审批事件

1. 当成员组织被邀请加入联盟时，会添加一条审批事件

审批事件					
事件名称	发起方	相关联盟	事件状态	发起时间	操作
test3邀请用户3cec085510884e4488a431c426eac95f加入联盟test3	test3	test3	●● 审批中	2019-07-26 00:14:21	接受 拒绝

2. 当用户选择“接受”可进一步执行【加入联盟操作】，加入成功后会归档到“已完成”
3. 当用户选择“拒绝”则直接归档到“已完成”

申请事件					
事件名称	发起方	相关联盟	事件状态	发起时间	操作
test2邀请用户q加入联盟sasatest2	我	sasatest2	●● 申请中	2019-07-22 07:10:57	查看详情

联盟通知

通知联盟中事件，比如org1已加入联盟

通道事件

通道事件

fabric 联盟链中有很多申请、审批的事件流，关于通道的事件流在通道事件中

申请事件

1. 当某个用户申请加入某个通道或申请创建通道时，会添加一条申请事件，具体可看 [通道管理](#)

申请事件					
事件名称	发起方	相关通道	事件状态	发起时间	操作
组织我的组织申请创建通道channel1	我	channel1	● 申请通过	2019-11-19 21:58:48	查看详情

2.当申请事件处理完，会归档到“已完成”

审批事件

1. 当某个用户被邀请加入通道时，会添加一条审批事件

审批事件

事件名称	发起方	相关通道	事件状态	发起时间	结束时间
组织oo申请创建通道mychannel	oo	mychannel	● 审批通过	2019-11-11 10:59:38	2019-11-11 10:59:38

2. 当用户选择“接受”可进一步执行“加入通道”，加入成功后会归档到“已完成”

3. 当用户选择“拒绝”则直接归档到“已完成”

通道通知

通知通道中事件，比如mychannel安装了链码mycc

事件名称	发起方	发起时间
组织ef57a762-01c6-427f-9d86-94426b582c2a加入通道	esb-test	2019-11-21 16:40:52
组织金融监管方申请加入通道channel1	zh-test	2019-11-21 16:39:33
亲爱的审批者，通道创建审批成功	esb-test	2019-11-21 16:33:34
组织金融监管方申请创建通道mychannel1	金融监管方	2019-11-21 16:26:53
亲爱的审批者，通道创建审批成功	esb-test	2019-11-18 21:19:35

SDK应用

SDK应用

本文档指导用户如何利用SDK与链码进行交互

前提 需要用户对官方 [fabric-sdk-go](#) 有一定了解

下载SDK

1. 进入【联盟】菜单，点击【联盟管理】进入【联盟信息】页面，点击【下载SDK文件】；如下图：

联盟信息

联盟成员

节点管理

通道管理

26a0f780-c244-e42f-f19d-4bd41d07d484 ● 运行中

基本信息

联盟名称：first

共识机制：Kafka

联盟描述：first

英文名：first

数据库类型：CouchDB

证书管理：[下载sdk文件](#)

SDK应用 结构概述 | 名称 | 类型| 应用必须 | 说明 | | ----- |----- |----- |:----| | crypto-config |文件夹 |是|包含调用链码所需私钥和证书 (User1) ；

包含连接orderer所用tls证书 | | sdk_config.yaml| 文件 |是| sdk的配置文件，包含联盟组织节点和orderer节点信息| | demo | 文件夹 |否| 包含示例链码；

包含基于fabric-sdk-go、 fabric-sdk-java和fabric-nodejs-sdk的调用链码的示例程序 | | README.md | 文件 |否| 使用说明|

应用步骤 1、Golang sdk

- 根据实际应用，修改sdk_config.yaml和crypto-config路径

- 下载fabric-sdk-go所需依赖，百度提供依赖下载：[vendor](#)
- 编写链码调用程序，可参考（demo/go-sdk/main.go）

2、Java sdk

- 根据实际应用，修改sdk_config.yaml和crypto-config的路径
- 下载fabric-sdk-java所需依赖，使用maven可以自动下载相关依赖
- 编写链码调用程序，可参考（demo/java-sdk/src/main/java/com/fabric/example/main.go）

3、NodeJs sdk

- 根据实际应用，修改sdk_config.yaml和crypto-config路径
- 下载fabric-sdk-nodejs所需依赖，需要自行安装好nodejs环境和npm工具，使用npm install工具安装依赖
- 编写链码调用程序，可参考（demo/node-sdk/main.js）

证书管理

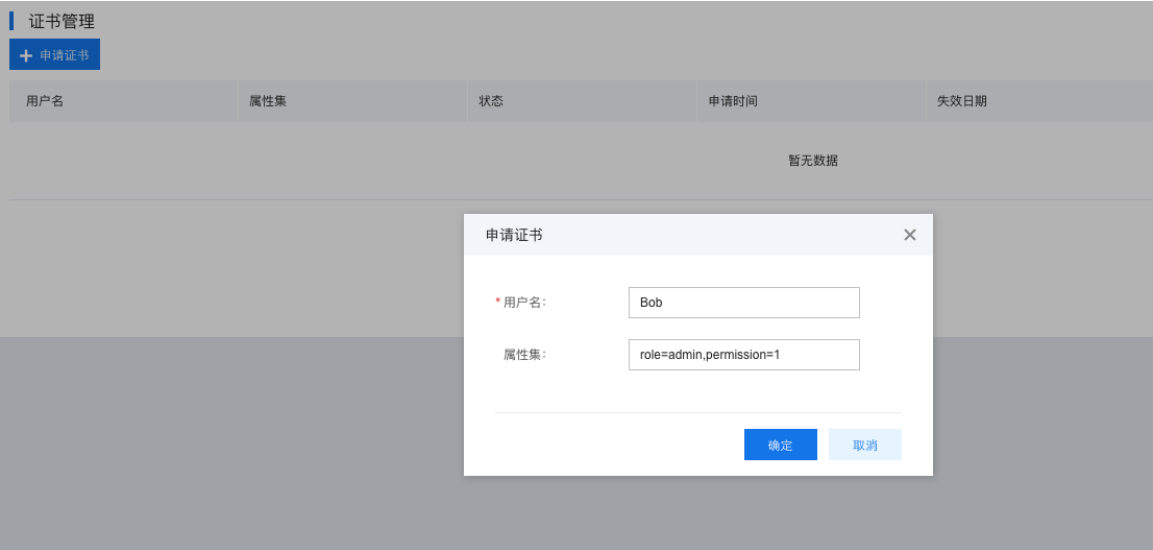
🔗 证书管理

证书管理支持client证书的申请和下载，用户可使用下载的证书调用链码并进一步使用证书属性集实现丰富的权限校验。

申请证书

1. 进入【联盟】菜单，点击联盟名称进入联盟详情页，点击【证书管理】进入证书管理页。
2. 申请证书时需要设置的参数：

参数	说明	举例
用户名	证书的唯一标识	Bob
属性集	key=value格式，多个键值对以英文逗号分割	role=admin,permission=1



下载并使用证书

1. 申请完证书后可在证书操作栏进行证书下载：

证书管理						
+ 申请证书						
用户名	属性集	状态	申请时间	失效日期	公钥密码算法	操作
Bob	role=admin,permission=1	● 正常	2020-06-17 11:54:44	2021-06-17 11:54:44	ECC	下载

2. 下载证书与SDK的配合使用，将下载的证书解压到 sdk 的crypto-config/组织名称/users 并将demo中UserName改为证书用户名即可（比如：Bob）

属性集实现权限认证 Hyperledger Fabric 支持使用基于属性的访问控制（ABAC，Attribute-Based Access Control）来控制用户的访问权限。在进行申请证书时，可以包含一个或多个属性名称和值，链码运行时可以提取属性值来进行访问控制决策。

例如，您正在开发一个“黑名单”的应用，需要控制上传黑名单的权限。您可以在链码中指定permission权限为1的用户才可以上传黑名单，然后申请证书时为有上传黑名单权限的client指定permission=1

关于 Hyperledger Fabric ABAC 功能，更多请参见：[Attribute-Based Access Control](#)

黑名单 demo,其中对上传操作进行权限校验：

```
package main

import (
    "fmt"

    "github.com/hyperledger/fabric/core/chaincode/shim"
    "github.com/hyperledger/fabric/core/chaincode/shim/ext/cid"
    pb "github.com/hyperledger/fabric/protos/peer"
)

type BlacklistChaincode struct {
}

func (t *BlacklistChaincode) Init(stub shim.ChaincodeStubInterface) pb.Response {
    return shim.Success(nil)
}

func (t *BlacklistChaincode) Invoke(stub shim.ChaincodeStubInterface) pb.Response {
    function, args := stub.GetFunctionAndParameters()

    if function == "upload" {
        return t.upload(stub, args)
    } else if function == "query" {
        return t.query(stub, args)
    }

    return shim.Error("Invalid invoke function name. Expecting \"upload\" \"query\"")
}

func (t *BlacklistChaincode) upload(stub shim.ChaincodeStubInterface, args []string) pb.Response {
    if len(args) != 2 {
        return shim.Error("Incorrect number of arguments. Expecting 2")
    }

    // 校验调用用户是否有上传黑名单权限
    err := cid.AssertAttributeValue(stub, "permission", "1")
    if err != nil {
        return shim.Error("The client identity does not have the permission to upload blacklist")
    }

    // Write the state back to the ledger
    err = stub.PutState(args[0], []byte(args[1]))
}
```

```

if err != nil {
    return shim.Error(err.Error())
}

return shim.Success(nil)
}

func (t *BlacklistChaincode) query(stub shim.ChaincodeStubInterface, args []string) pb.Response {
    if len(args) != 1 {
        return shim.Error("Incorrect number of arguments. Expecting name of the person to query")
    }

    // Get the state from the ledger
    Avalbytes, err := stub.GetState(args[0])
    if err != nil {
        jsonResp := "{\"Error\":\"Failed to get state for " + args[0] + "\"}"
        return shim.Error(jsonResp)
    }

    if Avalbytes == nil {
        jsonResp := "{\"Error\":\"Nil content for " + args[0] + "\"}"
        return shim.Error(jsonResp)
    }

    return shim.Success(Avalbytes)
}

func main() {
    err := shim.Start(new(BlacklistChaincode))
    if err != nil {
        fmt.Printf("Error starting Blacklist chaincode: %s", err)
    }
}

```

quorum联盟链

组织管理

创建组织

Quorum联盟由一个或多个组织组成。用户需要先创建组织，以组织的身份加入或者创建联盟。

1. 登录[百度智能云超级链BaaS平台控制台](#)，点击进入Quorum联盟链中。
2. 选择“组织”，进入“组织”页面。

The screenshot shows the 'Organizations' page in the Quorum BaaS platform. The left sidebar contains navigation options: Fabric联盟链, Quorum联盟链 (selected), 组织 (selected), 联盟, 事件中心, and 私有链. The main content area displays a workflow for creating an organization, joining a consortium, managing contracts, and configuring DApp parameters. Below the workflow is a table listing existing organizations.

组织名称	英文名	组织描述	所在联盟	创建时间	操作
org0	org0	test	bbe-experience	2019-11-18 16:57:44	删除
org1	org1	test	-	2019-11-20 16:02:51	删除

3. 点击“创建”或“创建组织”，弹出创建组织页面，填写配置信息，包括：

- 组织名称
- 英文名称
- 组织描述

创建组织

温馨提示：CA部署方式默认有平台自动生成，组织信息创建成功后不可修改

* 组织名称：

请输入组织名称

支持大小写字母、数字、中文以及_/.特殊字符，必须以字母开头，长度1-20

* 英文名：

请输入组织英文名

支持小写字母、数字以及-，必须以字母开头，长度1-16

* 组织描述：

请输入组织描述

0/200

确定

取消

4. 完成配置后点击“确认”，完成组织的创建。
5. 返回“组织”页面，即可看到已创建的组织。如果该组织已创建或加入联盟，则在“所在联盟”中将显示联盟名称。

查看组织

在“组织”页面中，点击“组织名称”即可进入到组织详情页面。

<

org0

删除

英文名：org0 组织描述：test

所在联盟

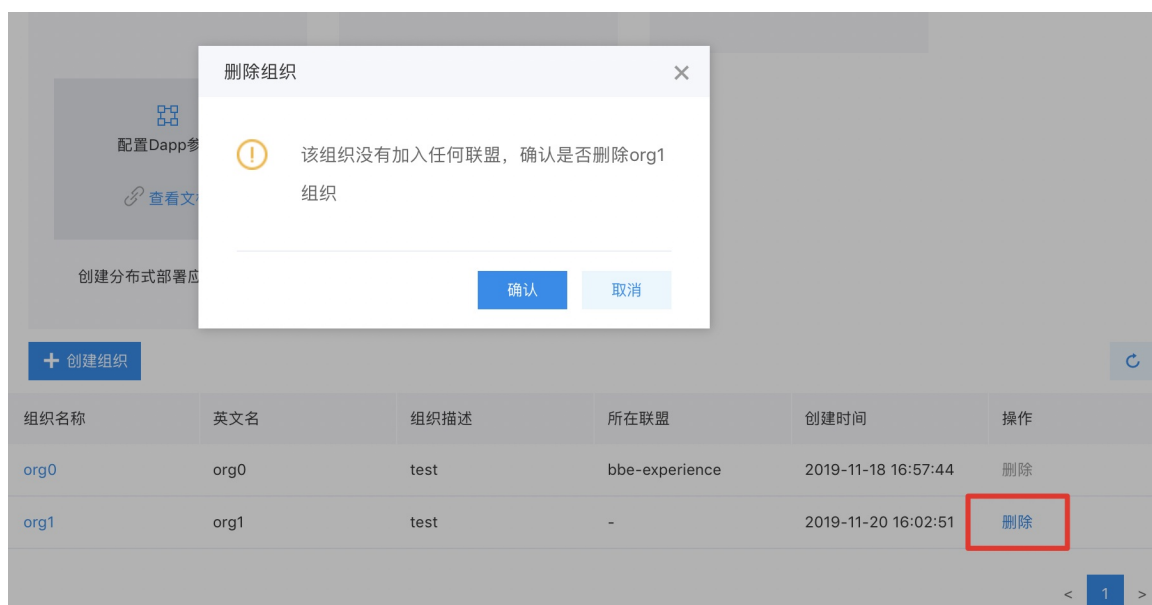
温馨提示：仅能被邀请加入联盟

联盟名称	英文名	联盟状态
bbe-experience	bbe-experience	● 运行中

如果该组织已创建或加入联盟，则在“所在联盟”中将显示联盟信息。

删除组织

在“组织”页面中，点击组织列表中的“删除”按钮，弹出删除确认框，点击“确认”，将删除该组织。 如果该组织已加入联盟，则禁止删除，需要先在“联盟”中“退出联盟”后才可删除。



联盟管理

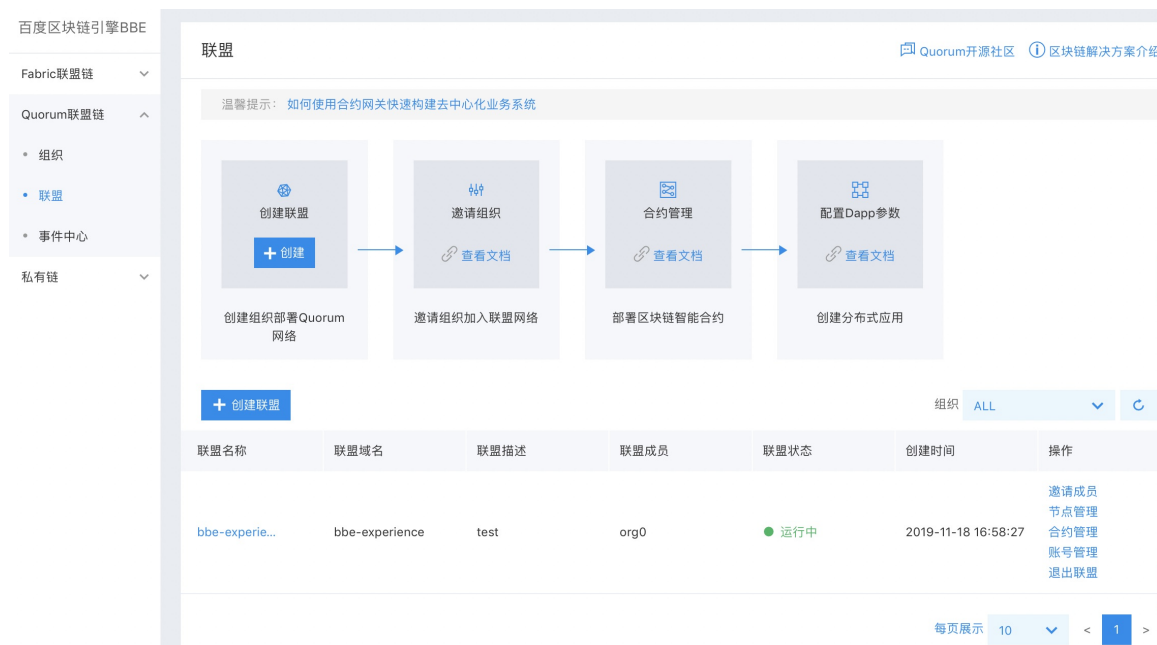
创建联盟

联盟代表一个区块链业务团体，可以由一个或多个组织组成。

目前，组织能够通过两种形式参与到联盟网络中：

- 创建联盟
- 接受邀请加入联盟

1. 进入[超级链BaaS平台](#)控制台中的Quorum联盟链，点击“联盟”菜单，进入“联盟”页面。已创建的联盟将显示在联盟列表中。



2. 点击“创建”或“创建联盟”按钮进入到联盟创建页面，填写配置信息，包括：

- 联盟名称：建议与联盟业务相关联，便于识别。
- 联盟英文名：联盟全局的唯一标示。
- 联盟描述：对联盟业务、联盟特征的描述。
- 初始组织：用户可以从已有的组织中选择或者创建新的组织来参与联盟。
- 本组织交易节点数：只同步块而不参与出块的节点个数（eg：普通配置建议1个，豪华配置建议2个）。

- 共识机制：默认IBFT共识。
- 出块周期(s)：默认每5s一个区块（eg：该配置影响出块时间，配置过小会增加分叉风险，建议采用默认配置）。
- GasLimit：一次交易可使用的Gas的上限（eg：一笔转账一般为21000，如配置过小，可能导致复杂合约因gas耗尽而执行失败）。
- 目标GasLimit：GasLimit动态调整的下限，要求大于等于3000000并且小于等于GasLimit（eg：该配置会影响到一个区块中可包含的交易个数，如无特殊要求，建议采用默认配置）。
- 可用区：可用区是指在同一区域下，电力和网络互相独立的区域，故障会被隔离在一个可用区内。
- 容器集群节点：可选择普通配置或豪华配置，普通配置默认启动4个验证节点，豪华配置默认启动7个验证节点。

创建联盟

联盟名称:

请输入联盟名称

支持大小写字母、数字、中文以及-/_/特殊字符，不能以特殊字符开头，长度1-20

联盟英文名:

请输入联盟英文名

支持小写字母、数字以及-，必须以字母开头，长度1-16

联盟描述:

请输入联盟描述

0/200

初始组织:

org1

本组织交易节点数:

-

1

+

?

共识机制:

IBFT

出块周期(s):

-

5

+

GasLimit:

4294967295

?

目标GasLimit:

4294967295

?

所选配置

清空配置

付费方式: 后付费

地域: 华北 - 北京

可用区: 可用区C

购买配置: CPU: 4核、内存: 8GB、系统盘: 40GB

购买配额: 可用区C: 4台

配置费用:

¥0.1327/分钟

(预计 ¥191.09/天 ¥5732.70/月)

下一步

购买集群

可用区:

☐ 可用区A

☐ 可用区B

☒ 可用区C

多个可用区的资源完全一致；创建服务时，会将容器自动化地分配至多个可用区的节点上，构建高可用架构。

容器集群节点:

普通配置

普通型III

内存: 4核 / 8GB

系统盘: 40GB

高性能云磁盘: 100GB

豪华配置

普通型III

内存: 16核 / 32GB

系统盘: 40GB

高性能云磁盘: 500GB

3. 填写完相关配置信息后，右侧会提示配置费用，确认无误后点击“下一步”。

4. 进入创建联盟的订单确认页面，确认无误后点击“去支付”，如需要修改则可点击“返回修改”。

创建联盟						
< 返回联盟列表						
产品类型	产品名称	配置	数量	时长	单价	计费方式
后付费	BBE（百度区块链引擎）	地域: 华北 - 北京 可用区: 可用区C CPU: 4核 内存: 8GB 系统盘: 40GB 高性能云磁盘: 100GB 所在网络: 默认私有网络 所在子网: 系统预定义子网	4	-	¥0.033175/分钟/台	按时长
*后付费服务将根据使用情况从账户余额中扣除，请保证有足够的金额。 《百度智能云线上订购协议》 返回修改 去支付						

5. 支付成功后将跳转回“联盟”页面，在联盟列表中将看到正在创建的联盟。大约等待15分钟左右，联盟将创建完毕。

42

查看联盟

在“联盟”页面中，点击“联盟名称”进入到联盟详情页面。

联盟详情页面包括：

- 1. 联盟基本信息，包含共识机制、同步模式、出块周期、联盟描述。
- 2. 联盟成员，该联盟下所包含的组织。

<联盟列表

联盟信息

节点管理

合约管理

账号管理

bbe-experience

基本信息

共识机制：IBFT

同步模式：full

出块周期：5

联盟描述：test

联盟成员

+ 邀请成员

组织名称	组织英文名	组织状态	交易节点数	验证节点数	加入日期	节点类型
org0	org0	运行中	1	4	2019-11-18 16:58:27	云上节点

邀请加入联盟

联盟创建成功之后，联盟内成员能够邀请百度云上用户加入联盟。

- 1. 在“联盟”页面中联盟列表“操作”列中点击“邀请成员”或进入到“联盟详情”页面中点击“邀请成员”，弹出“邀请成员”对话框，填写被邀请人的用户ID（目前只支持邀请百度云上用户加入联盟）。

邀请成员

温馨提示：邀请链接将通过站内信发送给相关账户，您可在事件中心查看进度

* 用户ID

请输入用户ID

用户ID可在个人中心获取

确定

取消

- 2. 被邀请用户进入自己的百度云超级链BaaS平台控制台的Quorum联盟链中的"事件中心"即可看到审批邀请事件（需要和邀请人的联盟网络在同一地域），接受或拒绝邀请。

百度区块链引擎BBE

Fabric联盟链

Quorum联盟链

组织

联盟

事件中心

私有链

事件中心

事件类型

待办

已完成

通知

申请事件

事件名称	发起方	相关联盟	事件状态	发起时间	结束时间	操作
暂无数据						

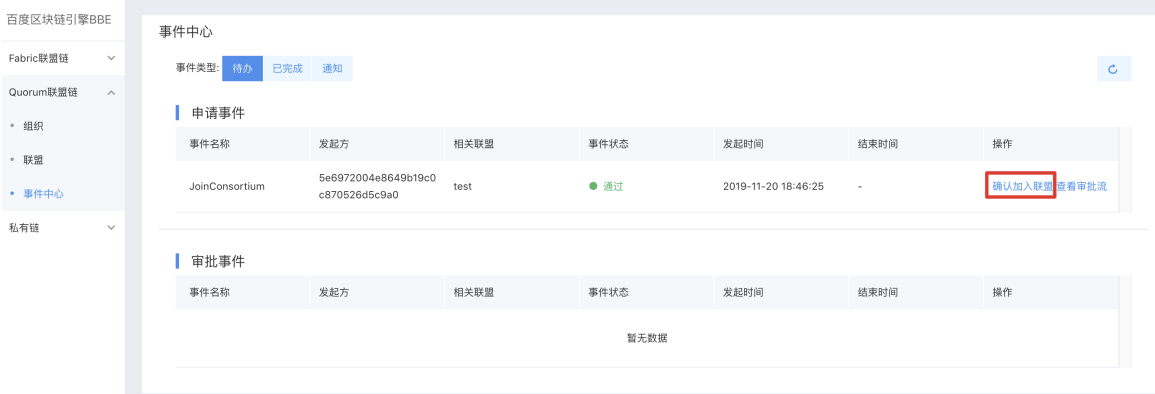
审批事件

事件名称	发起方	相关联盟	事件状态	发起时间	结束时间	操作
InviteJoinConsortium	org0	Consortium: test	审批中	2019-11-20 18:44:20	2019-11-20 18:44:20	接受邀请 拒绝邀请 查看审批流

3. 被邀请用户接收邀请后，需要联盟中其他成员审批用户的加入，选择“通过申请”或“驳回申请”。



4. 当全员审批通过后，被邀请人在其Quorum联盟的“事件中心”中即可看到“申请事件”的状态。



5. 点击“确认加入联盟”，进入加入联盟创建页面，填写配置信息。



6. 配置信息确认后点击“下一步”进入到订单确认页面，确认无误后点击“支付”后，将创建网络节点并加入联盟。

退出、删除联盟

进入“联盟”页面，在联盟列表中的“操作”列，点击“退出联盟”，联盟中的组织成员能够退出联盟。如联盟中最后一个组织退出联盟后，联盟网络将同时删除。

节点管理

节点管理

Quorum联盟由多个节点组成，节点分为：

- 验证节点，验证交易并出块的节点。普通配置套餐默认4个验证节点，豪华套餐默认7个验证节点。
- 交易节点，只同步块而不参与出块的节点，可用来接收交易请求。

进入“Quorum联盟链”中的“联盟”菜单，点击“节点管理”即可进入节点管理页面。

联盟列表

联盟信息

节点管理

合约管理

账号管理

节点管理

节点ID/节点类型	创建时间	隐私合约交易地址	JSON RPC	CPU使用率	内存使用率	存储空间	操作
213c998b 验证节点	2019-11-18 16:58:27	QuiHQeT1yaiaQp LuxiqwggcHi3\...	http://180.76.137. 13:8081/quorur...	31.525%	12.086%	137G 已使用4.617%	智能合约编辑器 区块链浏览器 节点控制台
22d38206 验证节点	2019-11-18 16:58:27	BiUP5x1v+FfV9c QYd2Qyy71K6Z...	http://180.76.137. 13:8081/quorur...	13.025%	11.088%	137G 已使用5.538%	智能合约编辑器 区块链浏览器 节点控制台
3c75e8da 验证节点	2019-11-18 16:58:27	37xJg4nK0JARxb hehcuQt9S1h3\...	http://180.76.137. 13:8081/quorur...	13.3%	10.937%	137G 已使用5.335%	智能合约编辑器 区块链浏览器 节点控制台
7ebb8ef4 交易节点	2019-11-18 16:58:27	KOpssilhzm5Ok4 duRgongqREsU...	http://180.76.137. 13:8081/quorur...	11.9%	10.95%	137G 已使用5.538%	智能合约编辑器 区块链浏览器 节点控制台
ba55f548 验证节点	2019-11-18 16:58:27	JgGgiQp4iwjUWq vXfPRDPvZRPC...	http://180.76.137. 13:8081/quorur...	9.75%	10.955%	137G 已使用4.617%	智能合约编辑器 区块链浏览器 节点控制台

选择节点: 213c998b

Node CPU负载: CPU使用率 (%)

Node 内存负载: 内存使用率 (%)

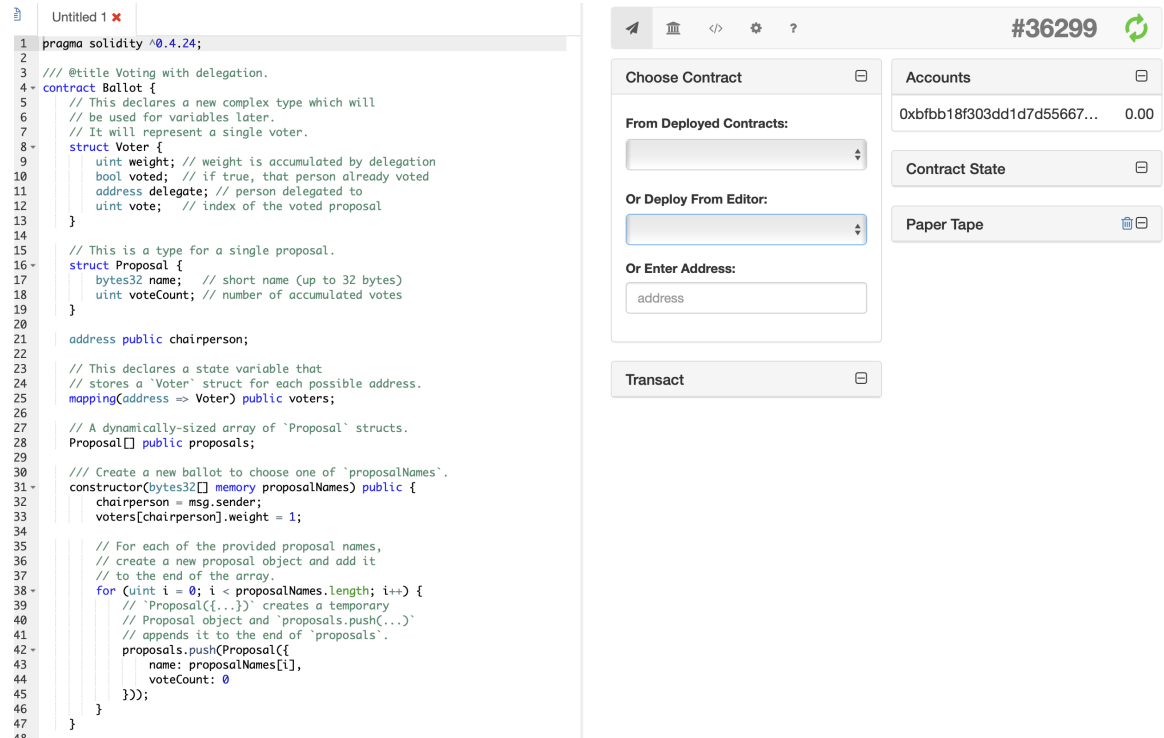
在节点管理页面包含的内容有：

- 节点基本信息，包括节点ID、节点类型。
- 网络相关信息，包括隐私合约交易地址、节点对应的JSON-RPC地址。
- 节点监控信息，包括CPU使用率、内存使用率、存储空间。

智能合约编辑器

每一个节点提供了一个智能合约编辑器，在合约编辑器中，可以开发调试、部署与调用合约。

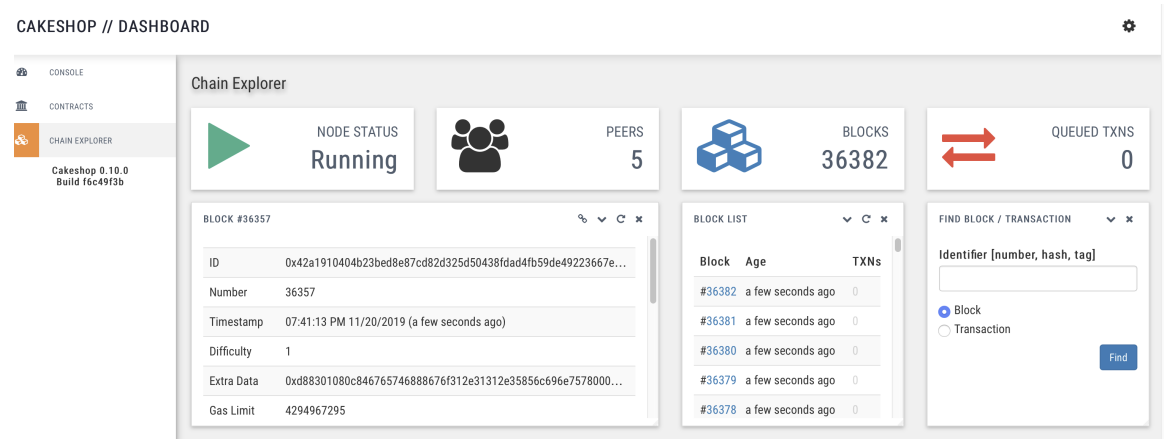
点击进入“智能合约编辑器”后，编辑器默认自带了名为“Ballot”的投票合约。合约编辑器的具体使用详见：[合约管理](#)



区块链浏览器

区块链浏览器提供了Quorum联盟网络区块相关的信息浏览与查询。

在“节点管理”页面中，点击某一个节点的“区块链浏览器”进入，如下图所示。



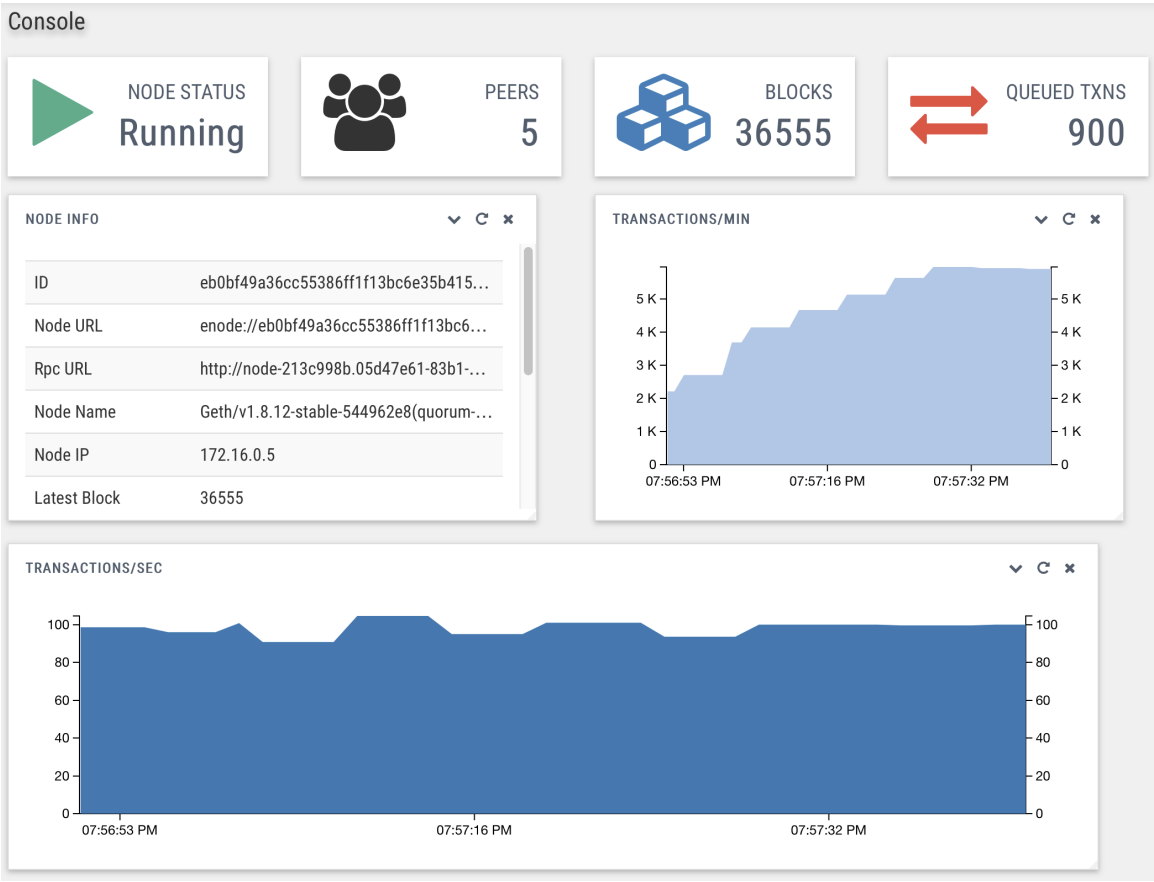
区块链浏览器中包含的信息有：

- BLOCK（区块信息），包含区块高度、区块Hash、区块时间、难度、额外数据、Gas Limit。
- BLOCK LIST（区块列表），动态更新的区块列表，以及区块中包括的交易个数。
- FIND BLOCK/TRANSACTION，根据区块高度或交易hash进行检索。

节点控制台

节点控制台与区块链浏览器均采用cakeshop提供服务，节点控制台主要提供节点相关信息。

在“节点管理”页面中，点击某一个节点的“节点控制台”进入，如下图所示。



在节点控制台所包含的信息有：

- NODE INFO，节点基本信息。
- TRANSACTIONS/MIN，每分钟交易个数。
- TRANSACTIONS/SEC，每秒交易个数。
- BLOCKS/MIN，每分钟区块个数。
- NODE LOG，节点日志。

合约管理

简介

用户可以在合约管理页中查看和管理通过合约编辑器部署的用户合约和平台提供的基础库合约。合约分为公有合约和私有合约，其中平台基础库合约部署后为公有合约。

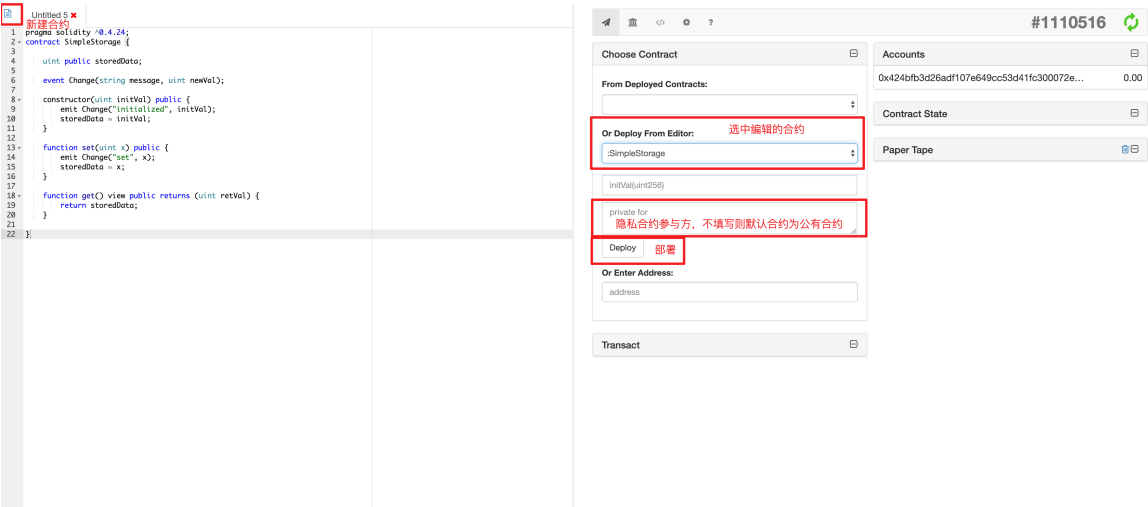


基础库部署

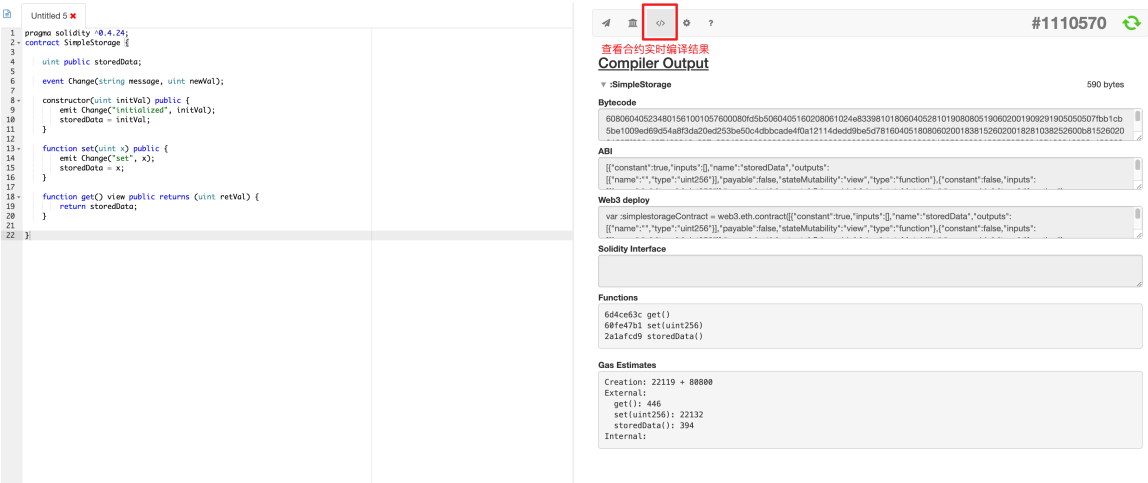
进入合约管理页，点击【基础库部署】，会展示出平台提供的基础合约列表，包括基于角色权限访问的RBAC合约、生成随机数的Randomness合约、Array数组合约等。选择需要部署的基础合约，点击【部署】。部署成功后返回合约管理页，在公有合约中可以看到刚部署的基础合约。

自定义合约部署

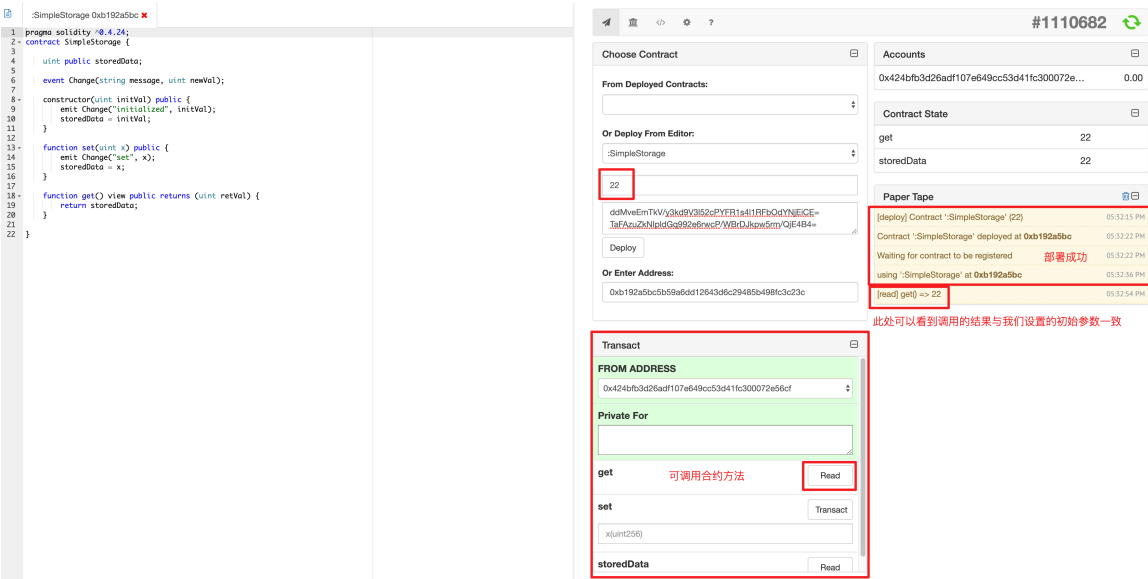
用户可通过Quorum合约编辑器cakeshop部署自定义合约，点击节点管理-智能合约编辑器，即可进入cakeshop合约编辑界面，其中左边为合约编辑区，右边为操作区，如果想部署隐私合约，图中private for字段需要填写隐私合约参与方的隐私合约交易地址，每个地址占一行，隐私合约交易地址可在节点管理界面获取



下图可查看合约实时编译结果



下图演示部署成功之后调用合约方法



🔗 合约详情查看

在合约管理页中，点击需要查看的合约名称，会跳转到该合约的详情页面。详情页面包含的合约信息有：

- 基本信息

- 合约简介
- ABI
- 合约源码

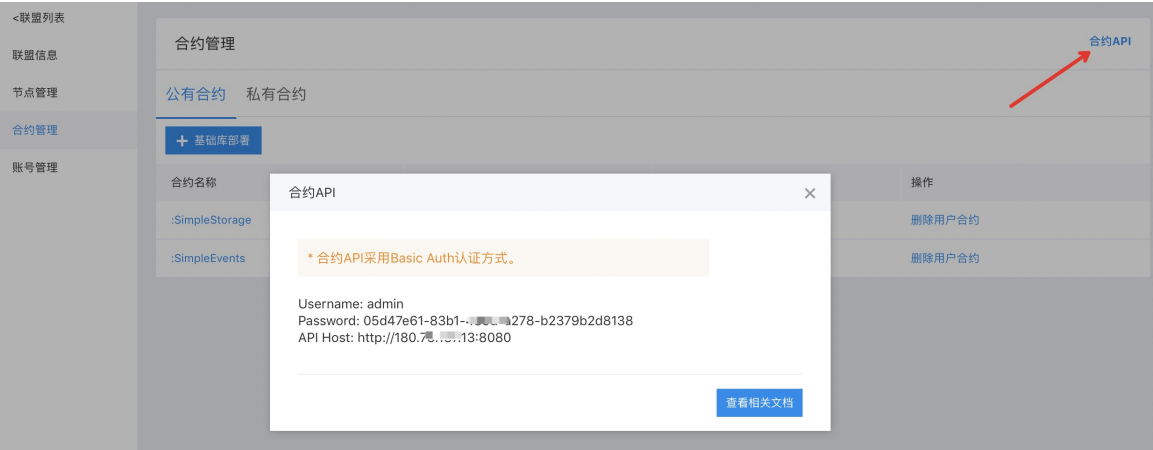


合约网关

合约网关是构建在Quorum联盟网络之上智能合约调用中间层，提供强一致的Nonce管理；合约方法编码、签名与解码；链上日志检索；事件订阅等服务。使用RESTful API设计方便开发者将区块链服务集成到自身业务中。

合约API 合约API为合约网关API调用方式，在“合约管理”界面右上角有“合约API”按钮，点击后弹出对话框，提示合约API相关信息，包括：

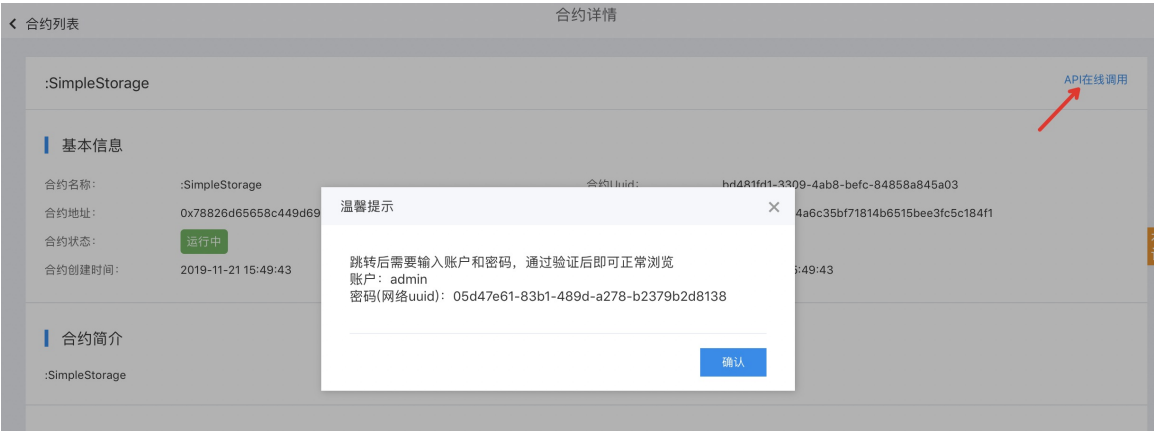
- Username，合约网关API调用采用Basic Auth认证方式，Username为认证用户名。
- Password，认证密码。
- API Host，合约网关API调用域名地址。



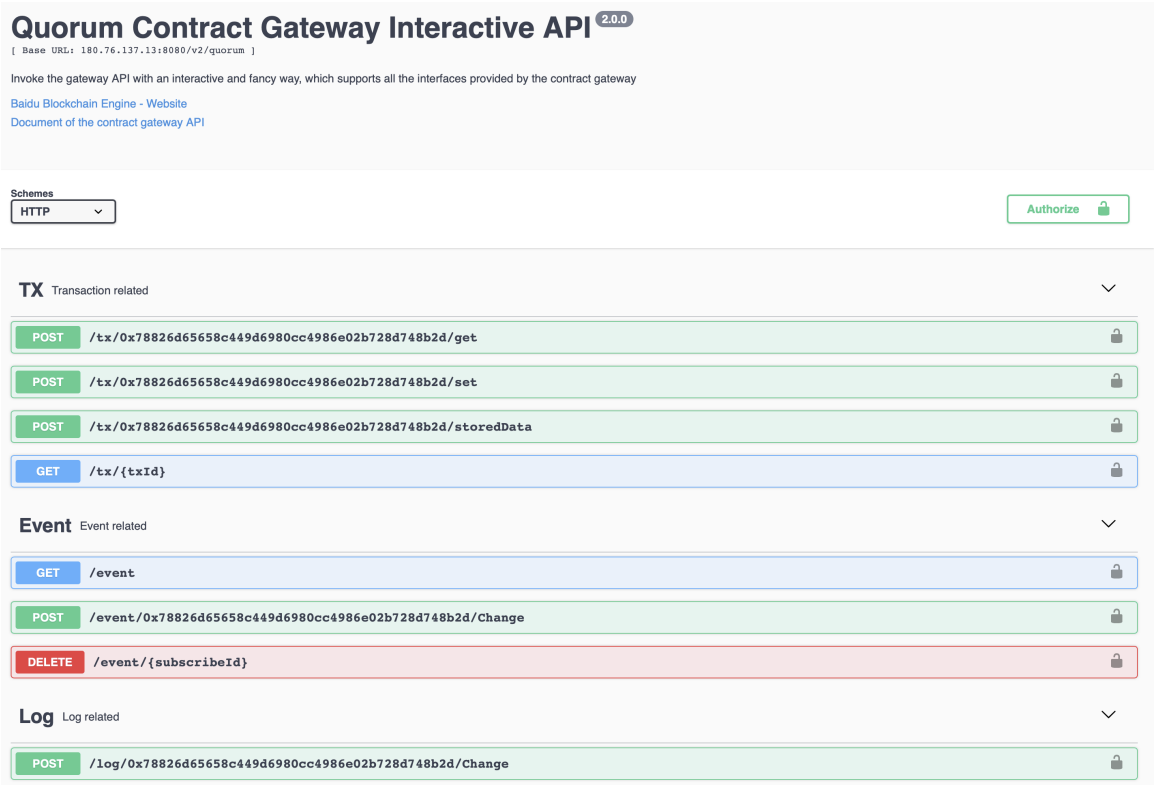
点击“查看相关文档”，会跳转到合约网关的API文档地址中。[文档地址](#)

API在线调用 API在线调用是合约网关提供的swagger OpenAPI服务，可以对合约进行在线调用。

- 1.在“合约详情”页面中，点击右上角的“API在线调用”，弹出提示框。



2. 点击“确认”后跳转到输入弹出框提示的用户名和密码，进入到在线调用页面。如下图为合约网关可调用和执行的“SimpleStorage”合约相关方法。



其中：

- TX，表示交易相关接口调用，如调用“SimpleStorage”合约中的get方法、set方法和storedData变量以及根据txId（交易Hash）获取交易结果。
- Event，表示事件相关接口调用，如GET请求类型获取已订阅的事件、POST请求类型发起事件订阅、DELETE请求类型删除已订阅事件。
- LOG，根据合约中的Event方法检索历史日志。

3.调用SimpleStorage合约中的“set”方法，可根据请求“body”中的“Model”查看请求参数说明，填写必填参数，如下图示例中填写了请求的账户地址from、请求的账户私钥privateKey、合约方法参数invokeParameters中的x值。点击“Execute”执行调用，调用结果在“Responses”中可见。

🔗 合约删除

账号管理

Quorum网络中的账户与以太坊网络中的账户是一个概念，账户由私钥控制，可以拥有、转账以太币，也可以部署和调用合约等。

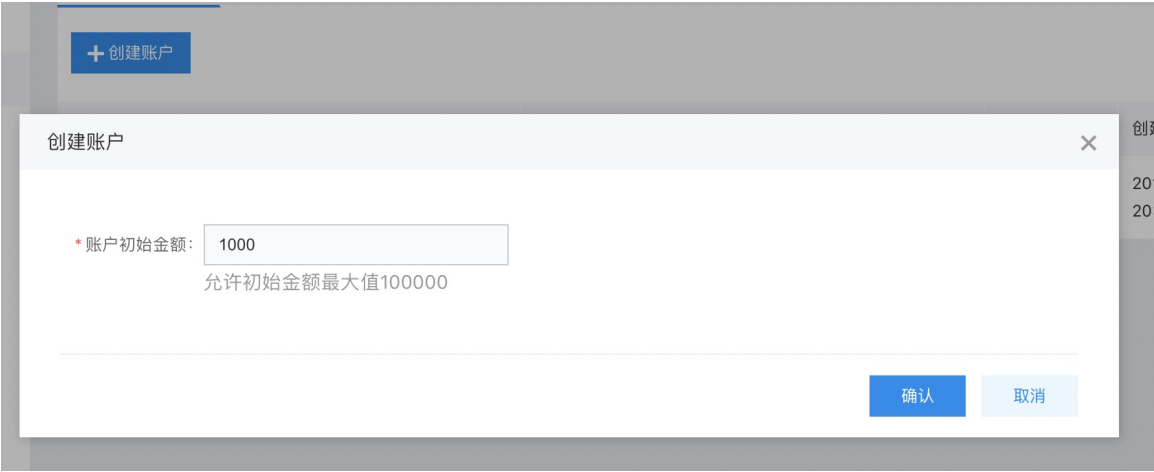
- 组织钱包账户，包含用户创建的账户。
- 系统预定义钱包账户，包含网络创建时预定义的账户。

51



创建账户

在账号管理页面中点击“创建账户”，弹出创建账户对话框，填写账户初始金额，并点击“确认”，创建成功的账户在“账号管理”页面中“组织钱包账户”中即可看到。

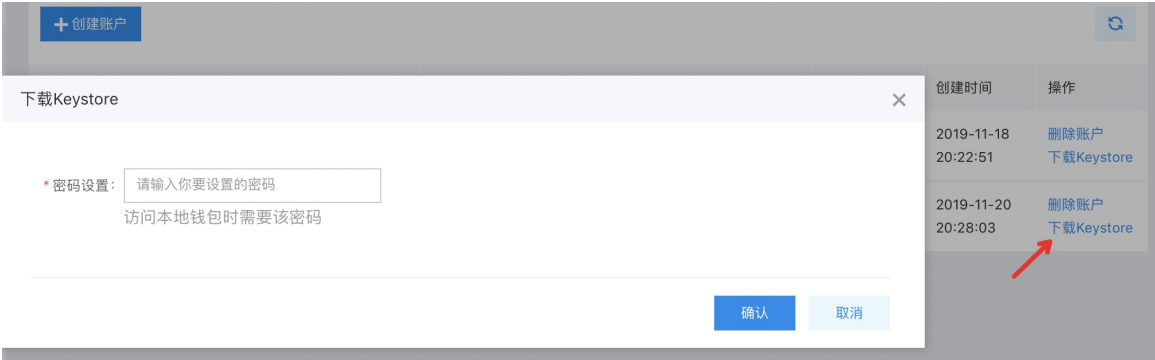


刚创建账户余额是0，等待十几秒，交易执行成功后，创建账户填写的余额将到账。

下载Keystore

以太坊账户Keystore是私钥的一种加密文件，使用Keystore文件加密码即可控制以太坊账户。

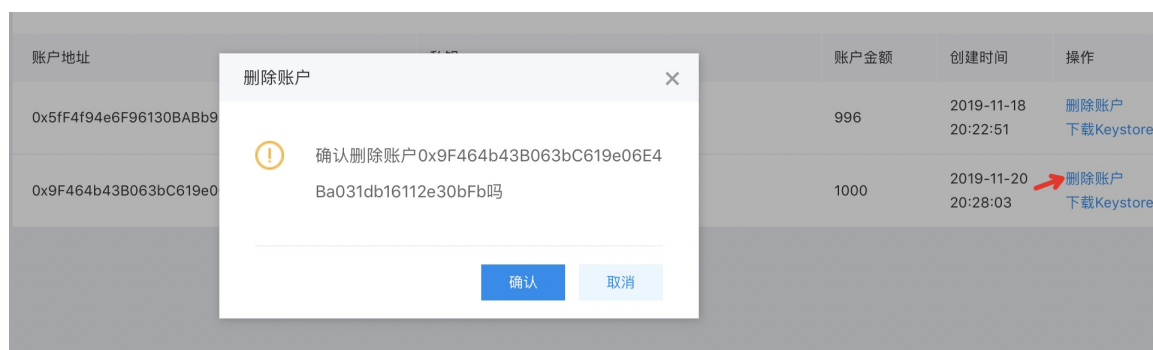
1. 在账号管理页面中，选择要下载Keystore的账户，点击“下载Keystore”，弹出对话框。



2. 在对话框中填写需要设置的密码，点击“确定”，即可下载该账户的Keystore文件。

删除账户

1. 在账号管理页面中，选择要删除的账户，点击“删除账户”，弹出确认框。



2. 点击“确认”，即可从“账号管理”中删除该账户。

删除后的账户在创建的网络里依然存在，只是在“账号管理”页面中删除。

事件中心

事件中心

事件中心能够帮助用户对联盟事件进行管理。事件分为待办、已完成、通知三个维度。待办、已完成为与用户直接相关的相应类型事件，通知为用户所在联盟的动态通知。

联盟事件

邀请加入联盟事件

1. 盟主邀请百度云用户加入联盟。
2. 被邀请用户审批加入联盟邀请。
3. 若用户接受邀请将触发新的加入联盟申请，联盟内所有成员审批申请。
4. 全员审批通过后用户加入联盟成功。

联盟通知 同一联盟中的成员能接收到联盟中动态通知，包括：

1. 联盟中新成员的加入。
2. 联盟中成员的退出。

兼容性说明

Quorum与Ethereum兼容性说明

Quorum是J.P.Morgan基于Ethereum开发的企业级区块链平台，相对于Ethereum，Quorum替换了共识算法，因此拥有更高的性能。除此之外，Quorum还提供Transaction和Contract的私有化功能，网络与节点的权限管理。

兼容说明：

- 兼容Solidity合约
- 兼容Remix合约编辑器（XUPERBAAS Quorum默认支持的编辑器为cakeshop，因为remix不支持部署隐私合约）

Quorum API接口说明

Quorum兼容Ethereum大部分web3接口，除了Ethereum提供的JSON-RPC之外，Quorum还做了如下改进：

- 修改web3.eth.sendTransaction方法，支持隐私交易
- 新增web3.eth.sendRawPrivateTransaction方法，支持发送raw transactions
- 新增eth_storageRoot方法，返回给定地址（合约/账户等）的存储根

- 新增eth_getQuorumPayload方法，返回Tessera/constellation的未加密payload
- 新增eth_sendTransactionAsync方法，异步发送交易

Quorum API 参考文档

🔗 附：Quorum工具链

- [Quorum Golang SDK](#)
- [Quorum Javascript SDK](#)
- [Quorum .NET SDK](#)
- [Quorum Java SDK](#)
- [Quorum客户端](#)

JSON-RPC防火墙

🔗 JSON-RPC防火墙

介绍

JSON-RPC防火墙是一种可配置的JSON-RPC请求鉴权服务。 管理员可为JSON-RPC调用者创建账号并根据JSON-RPC方法为只读或读写为调用者账号授权。

防火墙使用Basic Auth方式认证鉴权，服务启用后，网络JSON-RPC地址不变，调用者在使用JSON-RPC时需要携带认证信息。

开通服务

点击”Quorum联盟名称“后进入Quorum网络详情页面，在页面左侧点击”JSON-RPC防火墙“进入服务页面。

首次进入JSON-RPC防火墙服务时，会弹出服务开启弹出框，点击”确认“后可开通服务，如下图所示。



说明：服务开启后暂无法关闭，请谨慎选择。

创建JSON-RPC权限账号

点击”创建JSON-RPC权限账号“按钮，弹出账号创建弹出框，填写：

- 用户名，由6-12位字母数字构成
- 密码，点击”生成并复制密码“自动创建，服务不会存储密码，请妥善保管

填写完成后点击”确定“创建。如下图所示。



使用账号

创建的账号通过JSON-RPC可以请求区块链网络。其中JSON-RPC的地址可以在“以太坊详情”页面中查看，如JSON-RPC地址为<http://XXX:8081/quorum-jsonrpc/6c2eefbe-2671-4bf7-8f54-75b400b83b65/3cdd0399>。

JSON-RPC防火墙采用Basic Auth认证方式，开启服务后有以下两种方式通过JSON-RPC调用网络。

方式一：

将创建的用户名和密码，携带在JSON-RPC地址中进行请求，如:

<http://userName:passWord@XXX:8081/quorum-jsonrpc/6c2eefbe-2671-4bf7-8f54-75b400b83b65/3cdd0399>

其中“userName”是账号名，“passWord”是账号密码。

方式二：

将认证信息放到JSON-RPC的请求头中，请求头中增加“Authorization”字段，如：

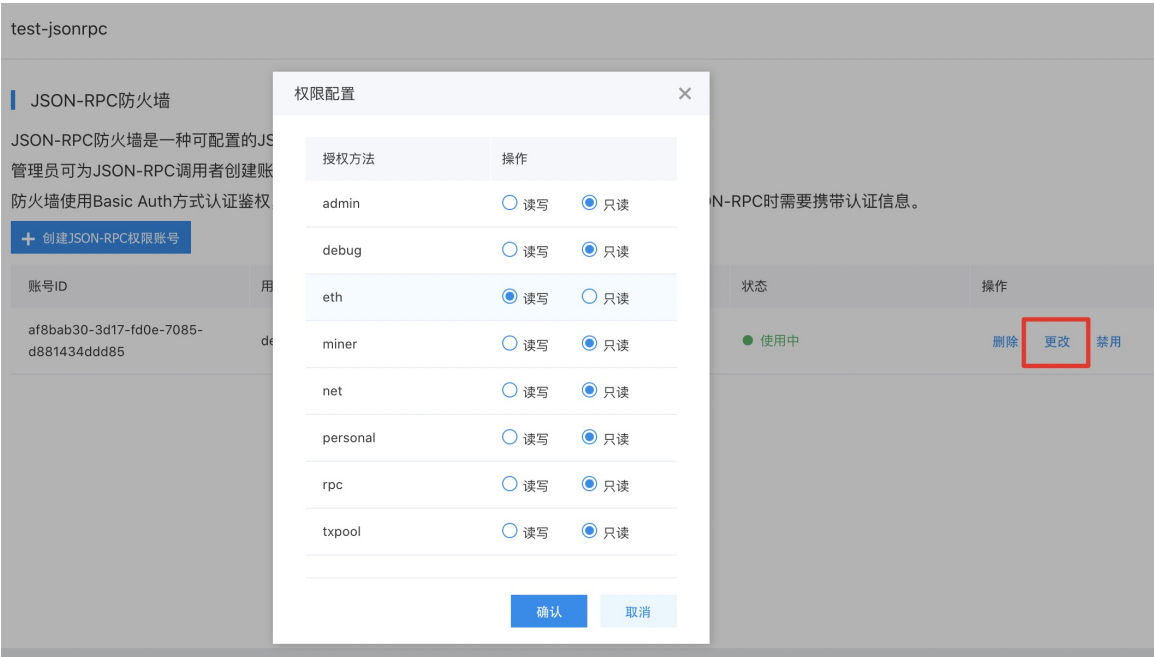
```
curl -X POST
http://XXX:8081/quorum-jsonrpc/6c2eefbe-2671-4bf7-8f54-75b400b83b65/3cdd0399 \
-H 'Authorization: Basic ZGV2ZWxvcDpYTkZZOHlhZ2JFMFI=' \
-H 'Content-Type: application/json' \
-d '{"jsonrpc": "2.0", "id": 1, "method": "eth_blockNumber", "params": []}'
```

其中Authorization生成方式为:

"Basic " + base64encode(userName:passWord)

更改账号权限

在已创建的账号列表中点击“更改”，弹出权限配置框，可以授权用户所调用的JSON-RPC方法为只读或读写进行授权。如下图所示。



授权方法包括：

- admin，提供节点管理相关方法，如admin.removePeer可以删除网络节点
- debug，提供节点debug相关方法，如debug.TraceBlockByHash可以通过交易hash跟踪交易
- miner，提供挖矿相关方法，如miner.stop()可以停止节点挖矿
- personal，提供账户管理相关方法，如personal.unlockAccount()可以解锁节点账户
- txpool，提供交易池相关方法，如txpool.Status()查看交易池状态
- eth，jsonrpc基础方法，提供eth相关方法，如eth.sendTransaction发送交易
- net，提供网络相关方法，如net.peerCount查看节点个数
- rpc，提供rpc相关方法，如rpc.modules查看rpc提供的方法种类

具体信息可阅读[官方文档](#)

禁用

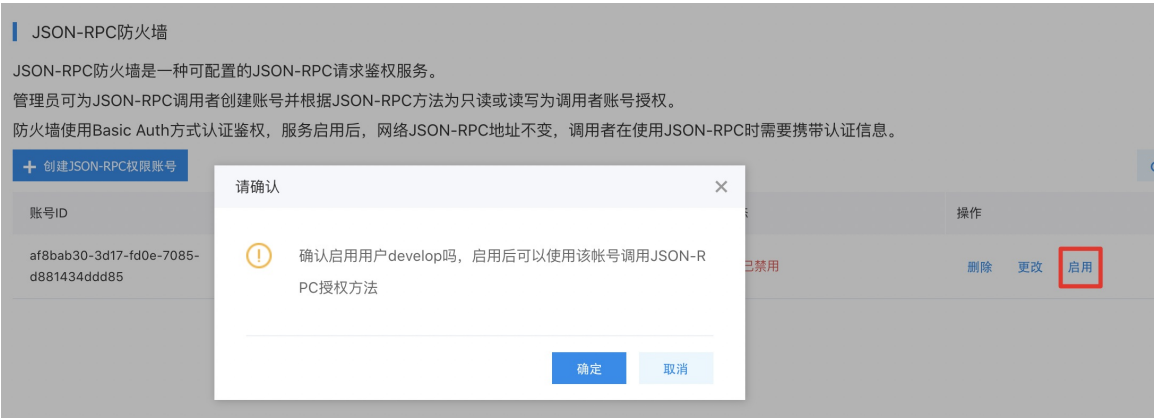
在已创建的账号列表中点击“禁用”，弹出确认框，点击“确定”禁用该账号。如下图所示。



禁用后，该账号将不能在通过JSON-RPC请求网络。

启用

账号禁用后可点击”启用“，在弹出的确认框中点击”确定“重新开启该账号，如下图所示。



删除

如需删除账号，点击账号列表中的”删除“按钮，在弹出的确认框中点击”确认“，即可删除。

删除后该账号将不能在通过JSON-RPC请求网络。 操作如下图所示。



以太坊
概念介绍

概念介绍

以太坊

以太坊（Ethereum）是一个开源的有智能合约功能的公共区块链平台，通过其专用加密货币以太币（Ether）提供去中心化的以太虚拟机（Ethereum Virtual Machine）来处理点对点合约。

以太坊虚拟机（EVM）

以太坊虚拟机（EVM）是以太坊中智能合约的运行环境。它不仅被沙箱封装起来，事实上它被完全隔离，也就是说运行在EVM内部的代码不能接触到网络、文件系统或者其它进程。甚至智能合约之间也只有有限的调用。

账户（Accounts）

以太坊中有两类账户，它们共用同一个地址空间。外部账户，该类账户被公钥-私钥对控制。合约账户，该类账户被存储在账户中的代码控制。外部账户的地址是由公钥决定的，合约账户的地址是在创建合约时确定的（这个地址由合约创建者的地址和该地址发出过的交易数量计算得到，地址发出过的交易数量也被称作"nonce"）。

交易（Transactions）

一笔交易是一条消息，从一个账户发送到另一个账户。交易可以包含二进制数据（payload）和以太币。

Gas

以太坊上的每笔交易都会被收取一定数量的gas，gas的目的是限制执行交易所需的工作量，同时为执行支付费用。当EVM执行

交易时，gas将按照特定规则被逐渐消耗。

Xuper BaaS for Ethereum



购买创建

购买创建

集群套餐选择

1. 登录百度智能云管理控制台，选择“产品服务 > 超级链BaaS平台”，点击“以太坊 > 创建以太坊网络”。



2. 根据业务应用场景，选择合适的计算资源进行购买，目前提供三种标准规格的套餐：普通配置、高级配置、豪华配置



普通配置	高级配置	豪华配置
普通III	普通III	普通III
内存：4核/8G	内存：8核/16G	内存：12核/32G
系统盘：50GB	系统盘：50GB	系统盘：50GB
高性能云磁盘：40GB	高性能云磁盘：300GB	高性能云磁盘：500GB

网络初始参数配置



置)。

- 目标GasLimit：GasLimit动态调整的下限，要求大于等于3000000并且小于等于GasLimit（eg：该配置会影响到一个区块中可包含的交易个数，如无特殊要求，建议采用默认配置）。

日志：

- 日志级别：设置日志展示的级别有 0=silent, 1=error, 2=warn, 3=info, 4=debug, 5=detail

JSON-RPC防火墙

JSON-RPC防火墙

介绍

JSON-RPC防火墙是一种可配置的JSON-RPC请求鉴权服务。管理员可为JSON-RPC调用者创建账号并根据JSON-RPC方法为只读或读写为调用者账号授权。

防火墙使用Basic Auth方式认证鉴权，服务启用后，网络JSON-RPC地址不变，调用者在使用JSON-RPC时需要携带认证信息。

开通服务

点击“以太坊网络名称”后进入以太坊网络详情页面，在页面左侧点击“JSON-RPC防火墙”进入服务页面。

首次进入JSON-RPC防火墙服务时，会弹出服务开启弹出框，点击“确认”后可开通服务，如下图所示。



说明：服务开启后暂无法关闭，请谨慎选择。

创建JSON-RPC权限账号

点击“创建JSON-RPC权限账号”按钮，弹出账号创建弹出框，填写：

- 用户名，由6-12位字母数字构成
- 密码，点击“生成并复制密码”自动创建，服务不会存储密码，请妥善保管

填写完成后点击“确定”创建。如下图所示。



使用账号

创建的账号通过JSON-RPC可以请求区块链网络。其中JSON-RPC的地址可以在“以太坊详情”页面中查看，如JSON-RPC地址为<http://XXX:8081/eth-jsonrpc/cbd72890-22c4-fb2d-262e-101759996922>。

JSON-RPC防火墙采用Basic Auth认证方式，开启服务后有以下两种方式通过JSON-RPC调用网络。

方式一：

将创建的用户名和密码，携带在JSON-RPC地址中进行请求，如：

<http://userName:passWord@XXX:8081/eth-jsonrpc/cbd72890-22c4-fb2d-262e-101759996922>

其中“userName”是账号名，“passWord”是账号密码。

方式二：

将认证信息放到JSON-RPC的请求头中，请求头中增加“Authorization”字段，如：

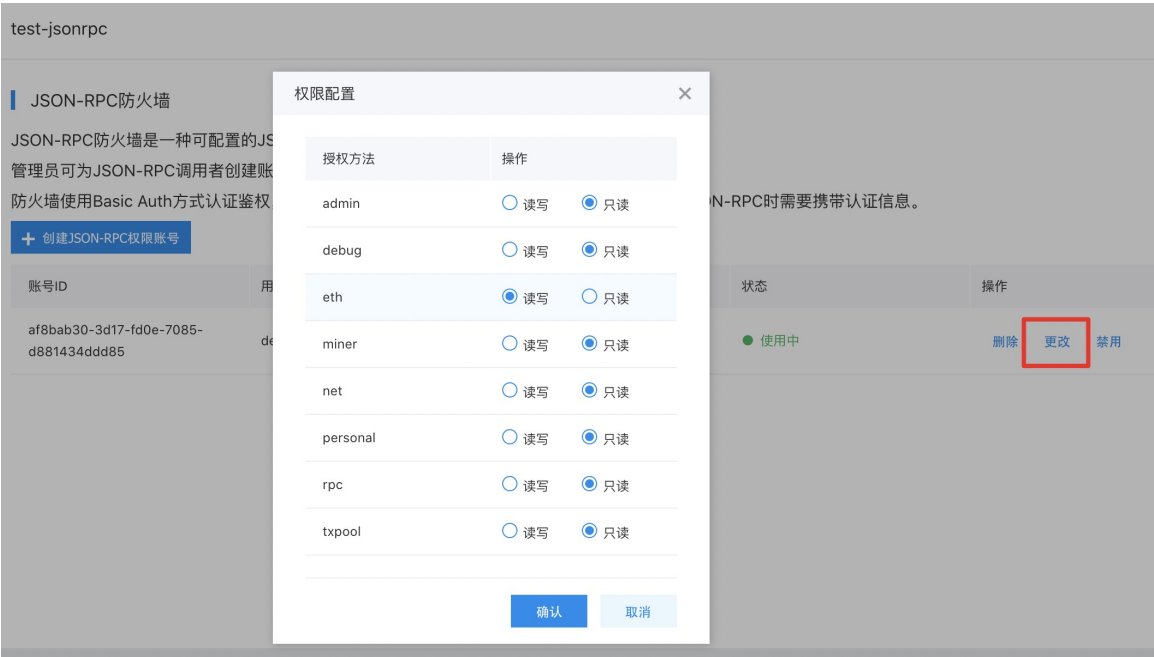
```
curl -X POST
http://XXX:8081/eth-jsonrpc/cbd72890-22c4-fb2d-262e-101759996922 \
-H 'Authorization: Basic ZGV2ZWxvcDpYTkZZOHlhZ2JFMFI=' \
-H 'Content-Type: application/json' \
-d '{"jsonrpc": "2.0", "id": 1, "method": "eth_blockNumber", "params": []}'
```

其中Authorization生成方式为：

"Basic " + base64encode(userName:passWord)

更改账号权限

在已创建的账号列表中点击“更改”，弹出权限配置框，可以授权用户所调用的JSON-RPC方法为只读或读写进行授权。如下图所示。



授权方法包括：

- admin，提供节点管理相关方法，如admin.removePeer可以删除网络节点
- debug，提供节点debug相关方法，如debug.TraceBlockByHash可以通过交易hash跟踪交易
- miner，提供挖矿相关方法，如miner.stop()可以停止节点挖矿
- personal，提供账户管理相关方法，如personal.unlockAccount()可以解锁节点账户
- txpool，提供交易池相关方法，如txpool.Status()查看交易池状态
- eth，jsonrpc基础方法，提供eth相关方法，如eth.sendTransaction发送交易
- net，提供网络相关方法，如net.peerCount查看节点个数
- rpc，提供rpc相关方法，如rpc.modules查看rpc提供的方法种类

[具体信息可阅读官方文档](#)

禁用

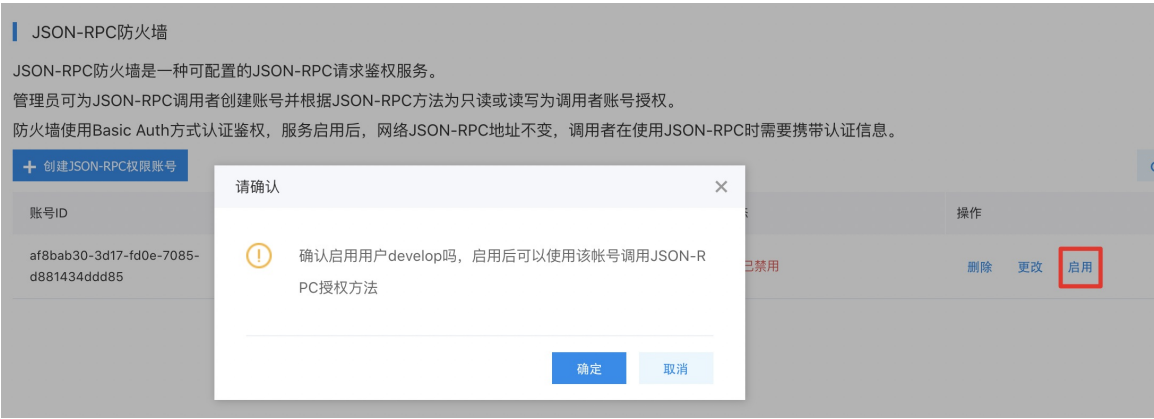
在已创建的账号列表中点击“禁用”，弹出确认框，点击“确定”禁用该账号。如下图所示。



禁用后，该账号将不能在通过JSON-RPC请求网络。

启用

账号禁用后可点击”启用“，在弹出的确认框中点击”确定“重新开启该账号，如下图所示。



删除

如需删除账号，点击账号列表中的”删除“按钮，在弹出的确认框中点击”确认“，即可删除。

删除后该账号将不能在通过JSON-RPC请求网络。 操作如下图所示。



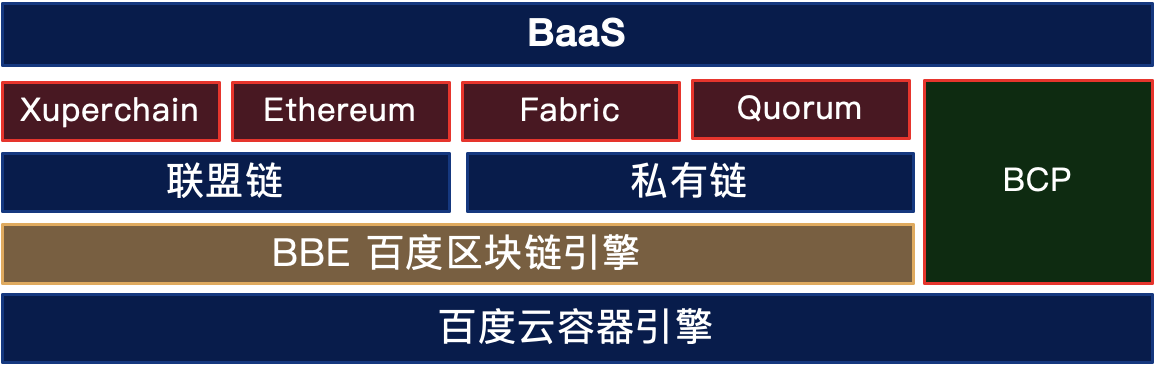
可信跨链平台

跨链服务特性介绍

简介

超级链跨链技术XuperCross是通用的、标准化的区块链互操作解决方案。百度超级链是跨链协议和标准制定的领导者，牵头制定IEEE跨链标准和国家信通院跨链标准，推出XuperCross则致力于构建各个行业领域协同发展的区块链生态。

百度可信跨链平台BCP是依托于XuperCross的跨链平台，为BaaS平台上的不同区块链提供跨链服务，帮助企业快速构建跨链应用。



架构介绍

XuperCross通过XIP底层协议解决了异构区块链（包括公链、私链和联盟链等）系统之间的互操作问题，以抽象的方式描述了跨链要解决的问题并设计了通用的解决方案；XIP包含一系列子协议，包括Naming Protocol 跨链网络的命名系统、 Cross Chain Transaction Consistency Protocol 跨链事务的一致性协议、Data Authentication and Communication Protocol 数据认证和通讯

协议。

业务场景	通用版权	分布式身份	通用积分	电子票据	通用存证
异构链支持	Xuperchain	Fabric	Quorum	Ethereum	EOS
解决方案	中继方案		TEE 中继		中继链
	XIP 跨链协议				
应用层	跨链查询		单向跨链事务		双向跨链事务
事务层	事务管理	二阶提交	存在性验证	跨链合约	证明转化协议
传输层	数据认证和通讯协议		寻址协议		加密传输
数据层	资源抽象和定位		跨链网络命名		数据序列化

这些基础的协议和标准保证了跨链数据交互的真实性验证和跨链事务的ACID特性（原子性、一致性、隔离性、持久性），实现了不同服务模型、不同区块链网络的跨链互通。



跨链服务控制台

创建联盟链

创建联盟链是进行跨链连接的第一步，目前跨链服务支持的区块链类型为Fabric，且只有北京地域的联盟链支持跨链，后续会支持更多区块链类型和更多地域。

创建Fabric联盟链参考：[创建联盟](#)

创建Fabric通道参考：[通道管理](#)

企业管理

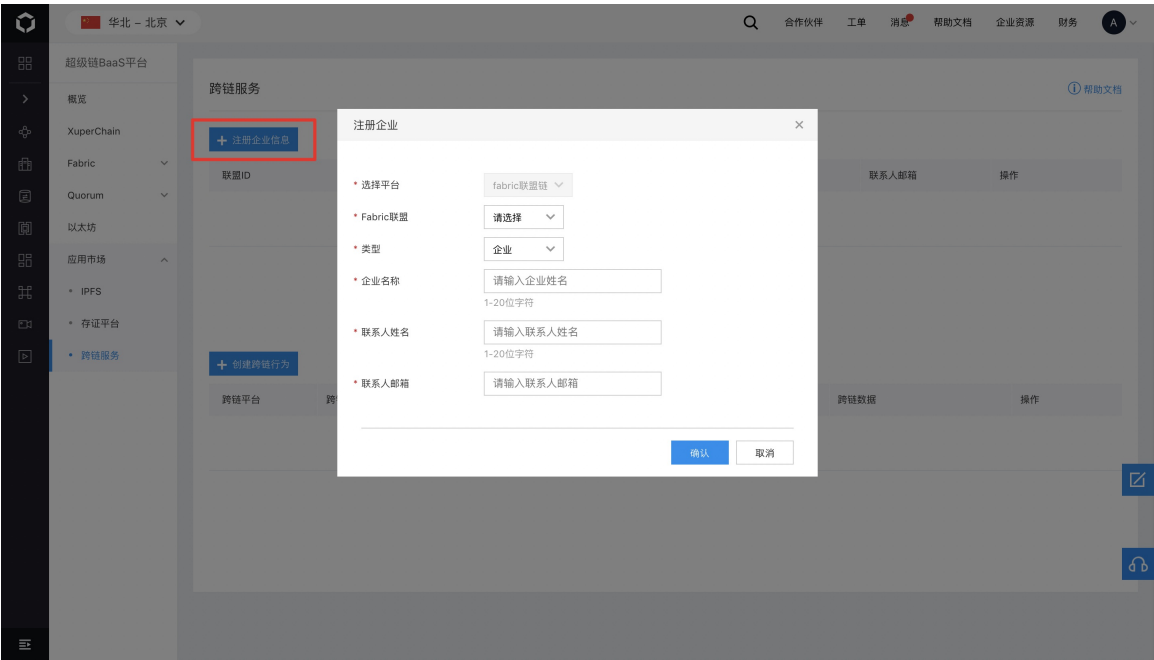
企业是参与跨链连接的一个实体，企业列表展示了在可信跨链平台注册的所有企业。企业管理主要包含以下操作：

- 创建企业
- 编辑企业
- 删除企业

注册企业

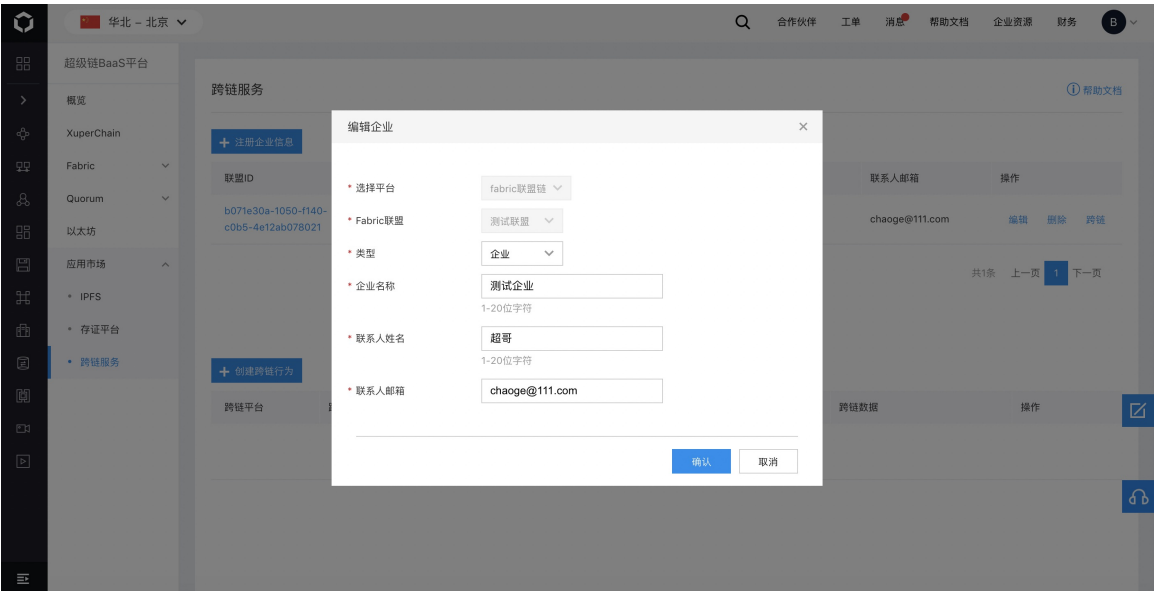
1. 进入【应用市场】菜单，点击【跨链服务】进入跨链服务控制台，点击【注册企业信息】。
2. 注册企业时需要设置的参数：

参数	说明	举例
选择平台	区块链类型，目前仅支持fabric	fabric
Fabric联盟	已经创建的Fabric联盟链名称	cc-1
类型	企业或个人	企业
企业名称	用于企业列表中企业间相互识别，在跨链平台中唯一；支持大小写字母、数字、中文以及_ / .特殊字符，长度1-20	百度公司
联系人姓名	用于在企业列表中展示信息；支持大小写字母、数字、中文以及_ / .特殊字符，长度1-20	李明
联系人邮箱	用于在企业列表中展示信息,便于他人联系，以进行跨链连接	liming@gmail.com



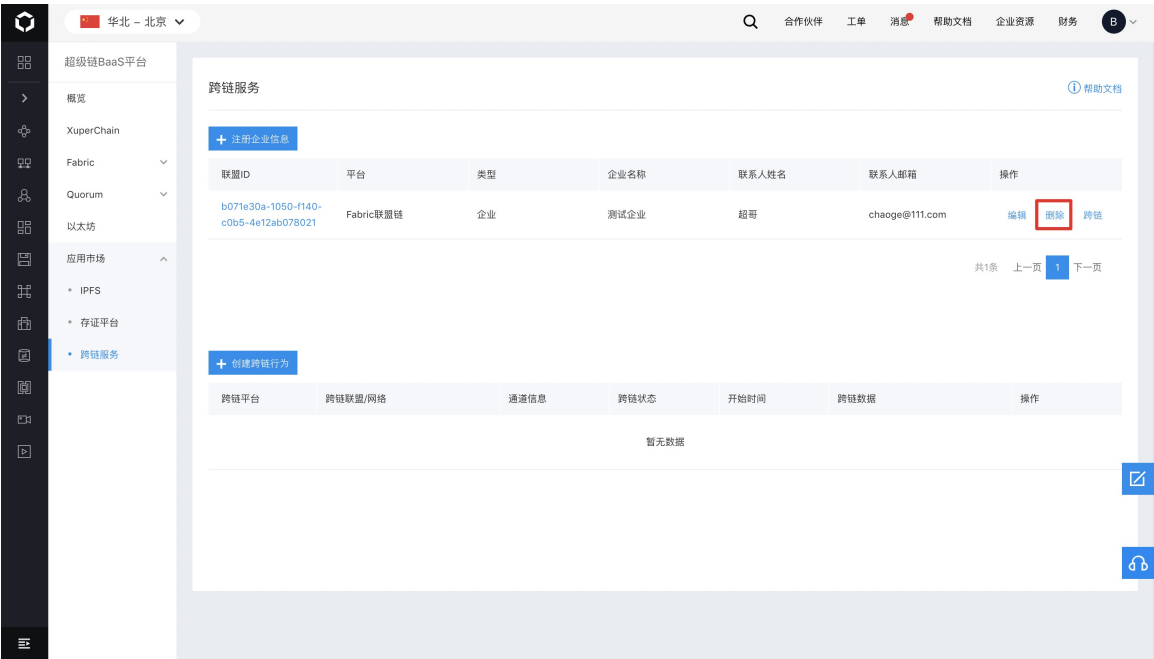
编辑企业 只有当前用户自己创建的企业可以编辑。

- 1. 进入【应用市场】菜单，点击【跨链服务】进入跨链服务控制台，选中要编辑的企业进行编辑。
- 2. 支持更新的字段为企业名称、联系人姓名和联系人邮箱。



删除企业 只有当前用户自己创建的企业，且当前企业的联盟链被删除时才可删除。

进入【应用市场】菜单，点击【跨链服务】进入跨链服务控制台，选中要删除的企业进行删除。



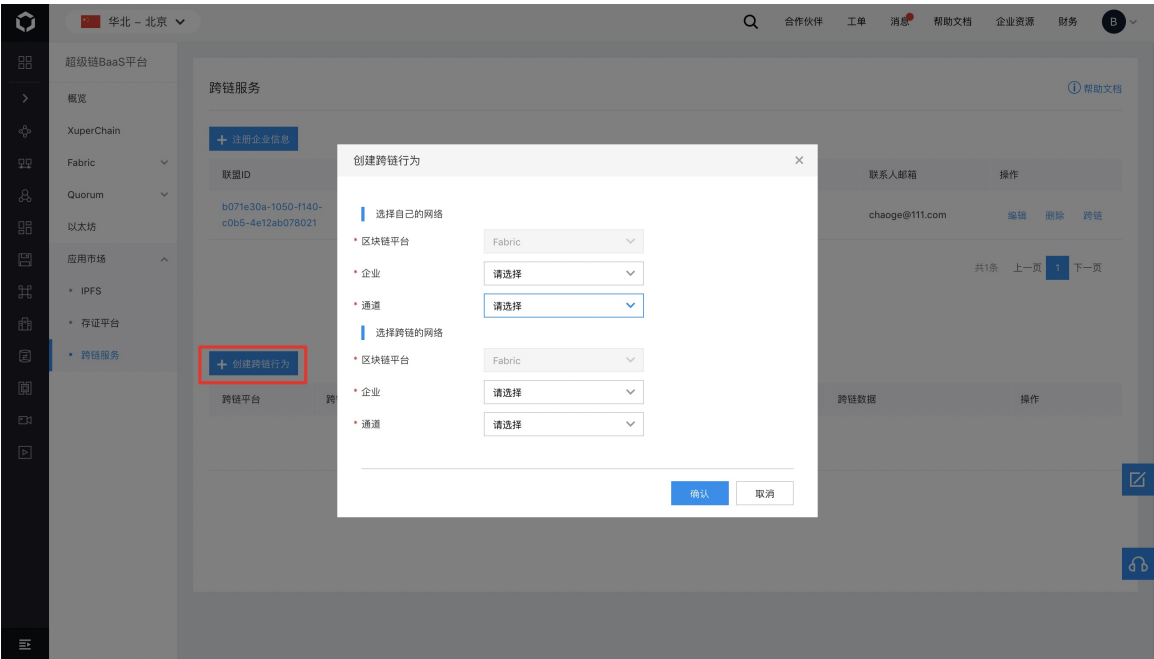
跨链连接管理

跨链连接管理主要包含以下操作：

- 创建连接
- 接受邀请
- 断开连接
- 查看详情

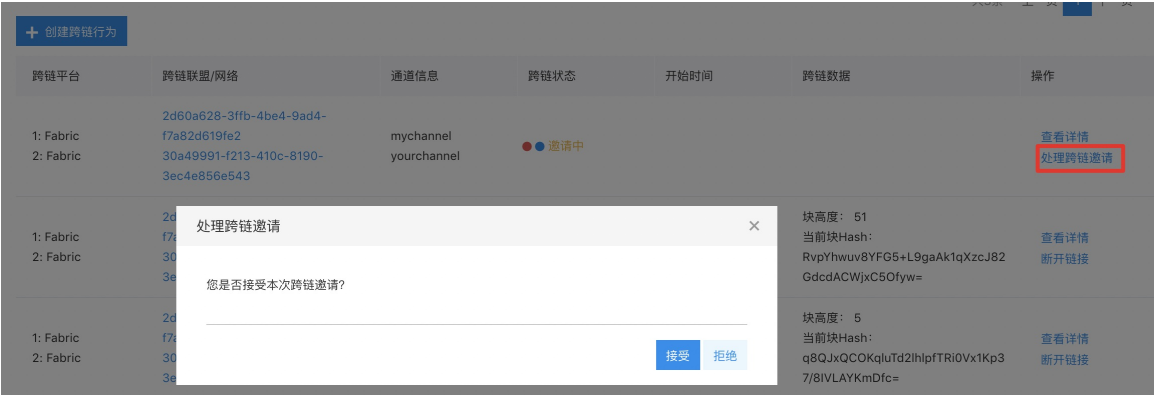
创建连接

1. 进入【应用市场】菜单，点击【跨链服务】进入跨链服务控制台，在企业列表中选择要与之跨链的企业，点击跨链；或者直接点击下方的创建跨链行为。
2. 选择自己和对方的企业和通道，点击确认，操作完成之后会给对方企业发起跨链邀请。



接受邀请

1. 受邀请方在跨链列表中选择对应连接，点击处理跨链邀请：



2. 点击拒绝会中止跨链连接的建立，点击接受后会进入跨链服务部署状态（包括部署跨链合约、积分合约和可信中继的建立）



3. 等待片刻后，跨链服务状态会变更为运行中，并且显示对方区块高度和hash值，说明跨链连接建立成功。



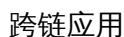
断开连接 跨链连接建立后，任意一方可以中止跨链。

1. 在跨链连接列表中选择对应连接，点击断开连接。
2. 点击确认，跨链将被中止。



查看详情

1. 在跨链连接列表中选择对应连接，点击查看详情。
2. 将会跳转到联盟详情的跨链服务页面，将会展示该联盟参与进行的所有跨链连接。



④ 前提

- 🔗 下载SDK

🔗 跨链应用

应用步骤

- ## 多用户访问控制

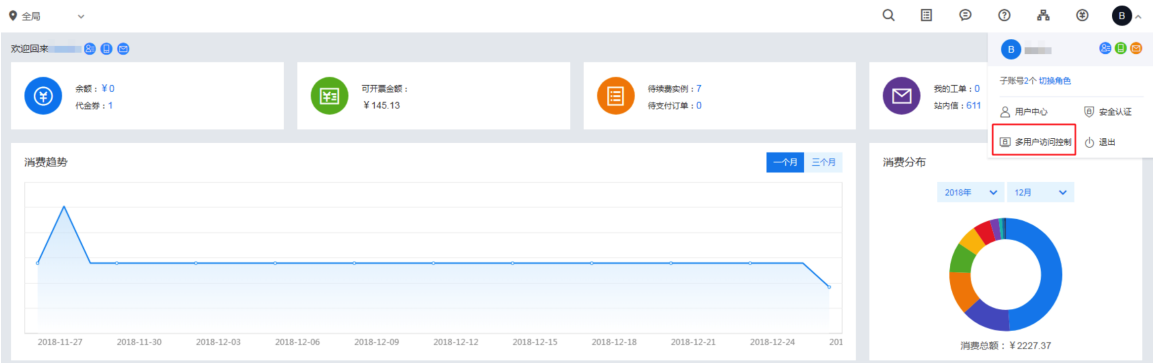
介绍

- 中大型企业客户：对公司内多个员工授权管理；
- 偏技术型vendor或SAAS的平台商：对代理客户进行资源和权限管理；

- 中小开发者或小企业：添加项目成员或协作者，进行资源管理。

创建用户

- 主账号用户登录后在控制台选择“多用户访问控制”进入用户管理页面。



- 在左侧导航栏点击“用户管理”，在“子用户管理列表”页，点击“新建用户”。
- 在弹出的“新建用户”对话框中，完成填写“用户名”和确认，返回“子用户管理列表”区可以查看到刚刚创建的子用户。

策略概述

超级链BaaS平台仅支持系统策略，实现超级链BaaS平台的产品级权限控制。

- 系统策略：百度智能云系统为管理资源而预定义的权限集，这类策略可直接为子用户授权，用户只能使用而不能修改。

系统策略

系统策略包含管理权限、运维权限和只读权限3种策略，权限范围详细如下：

策略名称	权限说明	权限范围
BBEFullControlPolicy	完全控制管理超级链BaaS平台的权限：子用户具备超级链BaaS平台完全管理权限。 可以创建/删除组织、创建/删除联盟、事件中心的审批。 此权限包含运维管理和只读权限，适用于全局管理员。	Fabric： 组织管理：创建组织、删除组织 联盟管理：创建联盟、邀请加入联盟、加入联盟、退出联盟、删除联盟 通道管理：申请创建通道、申请加入通道、创建通道、加入通道、修改通道描述、通过审批、驳回审批 事件中心：联盟审批、通道审批 以太坊： 网络相关：创建以太坊网络、删除以太坊网络 Quorum： 组织管理：创建组织、删除组织 联盟管理：创建联盟、确认加入联盟、退出联盟 事件中心：提交邀请、通过／驳回审批
BBEOperatePolicy	运维管理超级链BaaS平台的权限:子用户只具备超级链BaaS平台开发权限。 可以部署Fabric链码、部署以太坊智能合约和下载相关SDK 此权限包含只读权限，适用于进行	Fabric： 链码管理：提交链码文件、上传Chaincode、发布Chaincode、安装Chaincode、实例化Chaincode、升级Chaincode、下载链码、获取链码授权信息、替换链码文件、下载SDK压缩包 以太坊： 账户管理：创建新账户、查询账户私钥、下载keystore、删除账户 智能合约管理：部署合约、销毁合约

	此权限包含开发权限，适用于开发人员。	Quorum : 账户管理：创建新账户、查询账户私钥、下载keystore、删除账户 智能合约管理：部署合约、销毁合约
BBEReadPolicy	只读超级链BaaS权限：只具备相关列表、详情的查询权限。 适用于非开发者的业务人员。	Fabric : 组织管理：组织列表、详细信息 联盟管理：联盟列表、联盟详细信息、联盟组织列表、联盟通道列表 通道管理：通道列表、通道详情 链码管理：链码列表、链码详情 事件中心：审批列表、申请列表、审批流、联盟通知、通道通知 节点管理：节点列表查询、节点资源列表、cpu和内存时序图 部署相关：任务状态查询 以太坊： 网络相关：查询以太坊网络 账户管理：查询账户列表 智能合约管理：查看合约列表、获取未部署的合约基础库列表、查看合约基础库详情 部署相关：获取用户网络部署详情 Quorum : 组织管理：查看组织列表、查看组织详情 联盟管理：查询联盟列表、查询联盟详细信息、查询联盟节点 事件中心：查看申请列表、查看审批列表、查看审批流、查看通知列表 节点管理：节点资源监控、节点CPU与内存资源监控 账户管理：查询账户列表 智能合约管理：查看合约、查看合约列表、获取未部署的合约基础库列表、查看合约基础库详情

🔗 用户授权

在“用户管理->子用户管理列表页”的对应子用户的“操作”列选择“添加权限”，并为用户选择系统权限或自定义策略进行授权。

说明：如果在不修改已有策略规则的情况下修改某子用户的权限，只能通过删除已有的策略并添加新的策略来实现，不能取消勾选已经添加过的策略权限。

🔗 子用户登录

主账号完成对子用户的授权后，可以将链接发送给子用户；子用户可以通过IAM用户登录链接登录主账号的管理控制台，根据被授权的策略对主账户资源进行操作和查看。



其他详细操作参考：[多用户访问控制](#)。

典型实践

浦发银行区块链催收

本实践基于百度智能云区块链服务的商业项目编写，帮助您体验百度智能云区块链服务。

客户案例

百度智能云与浦发银行打造“立体链网”，推出智能区块链金融催收解决方案。

场景说明

业务场景

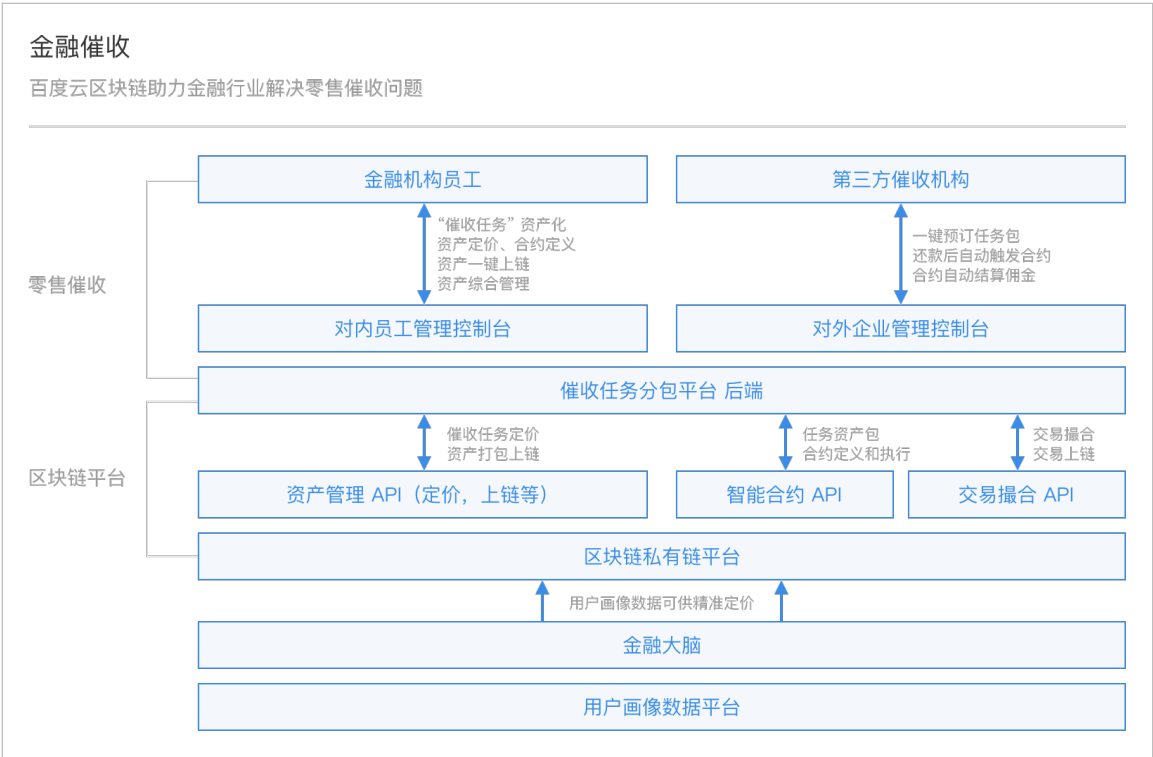
在金融领域，由于催收规范不清晰、平台地域分散和信息不透明等因素，催收行业往往执行效率低下，甚至出现暴力催收等行业乱象。但同时，零售信贷催收弱中心化、多组织的分工形式恰恰为区块链提供了很好的应用场景。百度智能云与浦发银行的合作正是基于分布式、去（多）中心化的核心逻辑，通过技术手段推进信贷催收透明化、规范化与智能化的标杆级业务案例。

业务难点

在零售信贷催收场景中存在信息透明度不高、平台地域分散、无有效激励体系、结算周期长等问题。

方案描述

本方案以云计算、区块链等为核心技术基础，与浦发银行金融业务系统实现对接，平台后续规划通过接入百度智能云金融大脑、百度开放大数据等能力，打通金融数据的清洗、分析与数据训练等数据加工各环节，为上层的智能定价、用户画像等应用场景创新提供技术支撑。



资产管理方面

双方共建区块链催收任务分包平台，通过差异化资产包与催收任务的统一上链与管理，实现全流程的信息共享、真实和透明，同时规避了以往线下信息交互与结款等繁琐流程，大大提升了催收效率、信息安全性和平台信任度。此外，通过接入智能合约，根据合约定义执行催收任务，实现资金自动化的结算清算，有效规避了规范混乱和结算周期长等行业痛点。

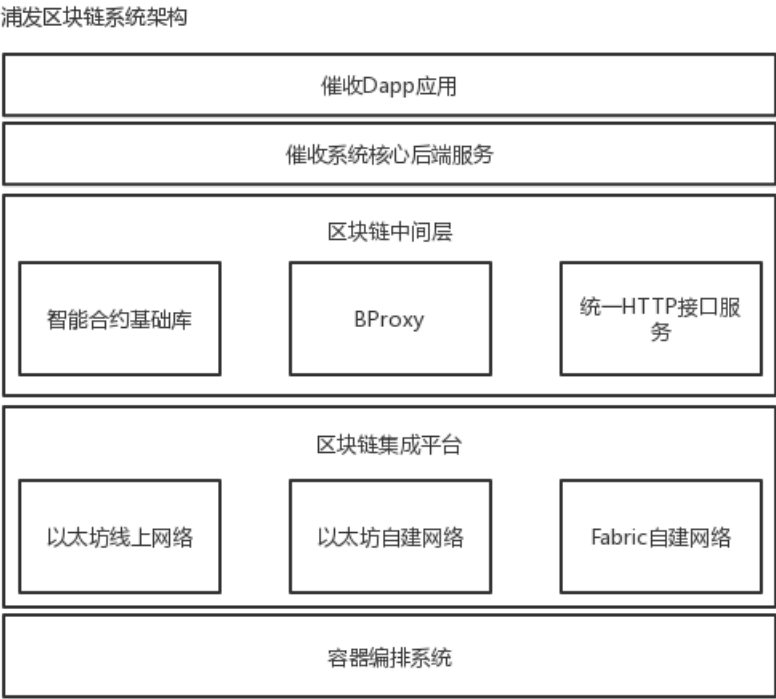
联盟建设方面

双方以联盟链技术平台为依托，重构催收业务流程，通过信任机制的优化推动弱中心化行业联盟的建立，为包括第三方金融机

构、第三方信息提供商与用户等在内的多方组织提供可信、高效与安全的金融服务。在上层业务创新方面，双方则通过对接百度金融大脑，依托用户画像打造任务定价模型，寻找最合理的定价和分包商激励标准体系，达成算法与用户、第三方能力的优化匹配，为金融机构大大提升了效益产出。

🔗 系统架构

本节介绍架构中子系统间逻辑关系和子系统功能。



🔗 容器编排系统

一套私有部署的容器集群管理系统。区块链网络的各节点以及区块链平台的管理服务组件都由容器编排系统托管。基于Kubernetes二次开发，在区块链项目中，容器调度功能只服务上层区块链组件，不支持用户其他的如自定义容器的托管。

🔗 区块链集成平台

一个可以部署、管理、监控区块链网络的平台。支持以太坊、Hyperledger Fabric两种区块链网络方案。

• 以太坊线上网络

独立的生产环境网络，与用户创建的其他以太坊网络有隔离，不允许普通用户删除。线上以太坊集群将部署支持特定业务的服务层。有明确的最小资源限制。

• 以太坊自建网络

由用户通过区块链集成平台创建的以太坊网络，支持对网络的创建删除和节点缩扩容。

• Fabric自建网络

由用户通过区块链集成平台创建的Fabric网络，支持对网络的创建删除和节点缩扩容，CA证书签发以及channel创建。

🔗 区块链中间层

• 智能合约基础库

核心工作：区块链Dapp开发的大量工作集中在合约开发，以及将合约调用接口转变成web形式的接口。代码复用：合约开发类似于传统应用开发，solidity支持库引用。为了复用代码，合约内部逻辑应该再抽象并进行对象化封装。

• BProxy

区块链BProxy解决了如何将公链区块链方案引入私有化场景，实现多方的身份互信管理，同时也在跨网环境中解决了数据上链的问题。

• 统一HTTP接口

统一HTTP接口提供一个RESTful的接口服务。对用户屏蔽区块链接口的复杂性，提供一套通用的合约调用接口。并且将不同类型业务的数据模型统一到区块链的逻辑控制合约中，完成数据模型与智能合约逻辑控制的解耦。

催收系统核心后端服务

催收系统后端服务，提供催收APP和Web端的所有定制功能的后端接口需求。负责将催收业务逻辑映射到HTTP接口提供的合约逻辑调用中。同时提供必要的链外数据存储。

催收系统Dapp应用

面向用户交互，是一个手机应用或者web前端。

场景总结

本实践适合于：

- “数据”资产化上链
- “区块链资产”具备独特属性
- “资产流转”多方参与

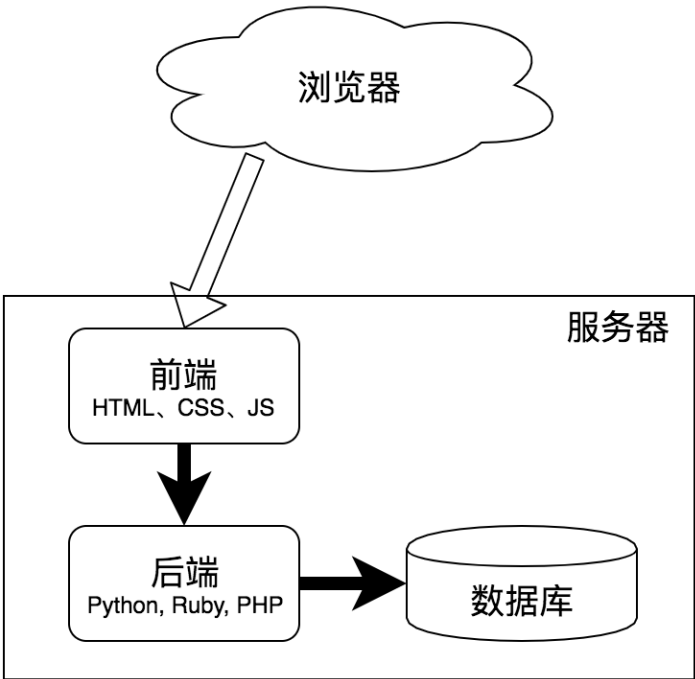
开发者指南

以太坊Dapp开发指南

Dapp架构及概要设计

什么是Dapp

web应用开发的一个经典web应用架构B/S结构为例：



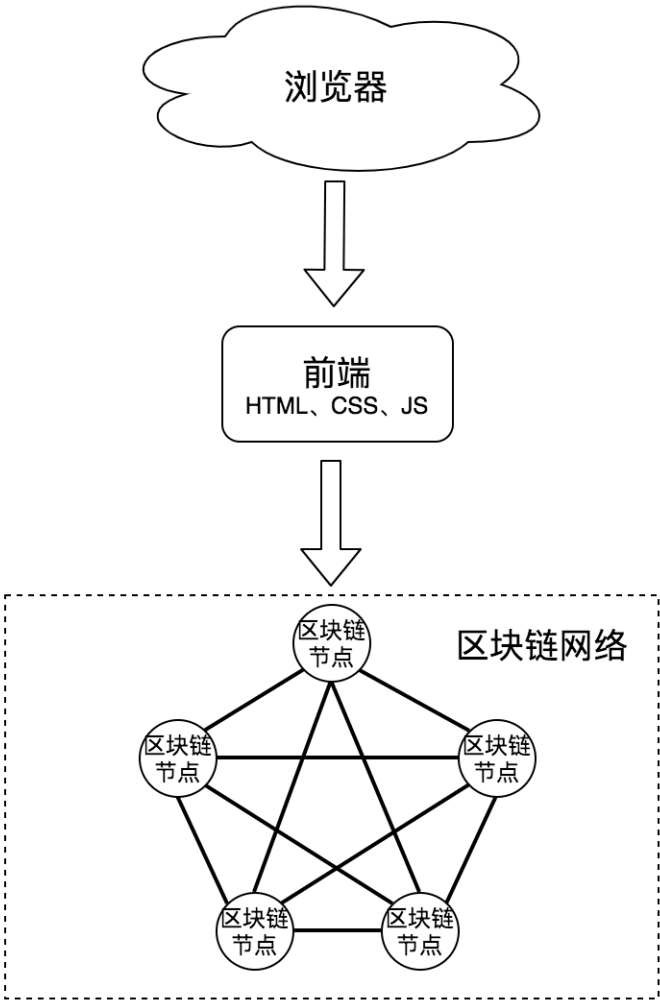
一般来说，我们开发完web应用后会把web服务部署到公网上，现在比较流行的方式是部署到云端VPS。无论在服务端我们用了什么负载均衡、容器化等技术，又无论客户端用户用的是浏览器还是一个API构造器。实质上所有的客户都将通过请求我们部署的服务器暴露的HTTP接口来访问应用。在服务器上，我们写的web应用程序会再做一些数据操作，比如对数据库或缓存做CRUD操作，最终将结果反馈给客户端。

这个B/S架构如此流行，是因为它适用于绝大多数应用场景。但是也有例外，当用户或者服务提供者认为数据比应用更有价值，又或者服务提供者想提供的是数据服务，并想有效控制用户对数据写的权利。此时基于B/S结构的应用会显得力不从心。

和中心化的web应用不同，区块链使用一套开源的网络规范，提供了数据P2P实时共享同步的特性。终端客户可以随时将一个客户端设备接入区块链P2P网络，基于标准的协议，区块链终端将实时同步到最新的数据。在这个数据服务中，不再有中心化的服务端存在，这就是所谓的“对等无中心”特性。

Dapp是基于区块链构建的web应用，区别于B/S结构的传统web服务，Dapp充分吸取了区块链“对等无中心”的特点，采用了Browser/Blockchain的结构。对于服务商而言，只需将数据同步到区块链上，并定义一套详细的数据写规则。而数据的调用读取展示逻辑完全由数据的用户定义。这样既通过区块链保证了数据的可信，同时不同的数据用户也可以按需实现不同的Dapp来挖掘数据bb据的价值。和中心化的web应用不同，区块链使用一套开源的网络规范，提供了数据P2P实时共享同步的特性。终端客户可以随时将一个客户端设备接入区块链P2P网络，基于标准的协议，区块链终端将实时同步到最新的数据。在这个数据服务中，不再有中心化的服务端存在，这就是所谓的“对等无中心”特性。

下图是一个典型的Dapp架构：



🔗 我们要实现什么Dapp

我们要实现的是一个基于区块链的便签板应用。页面效果图如下：



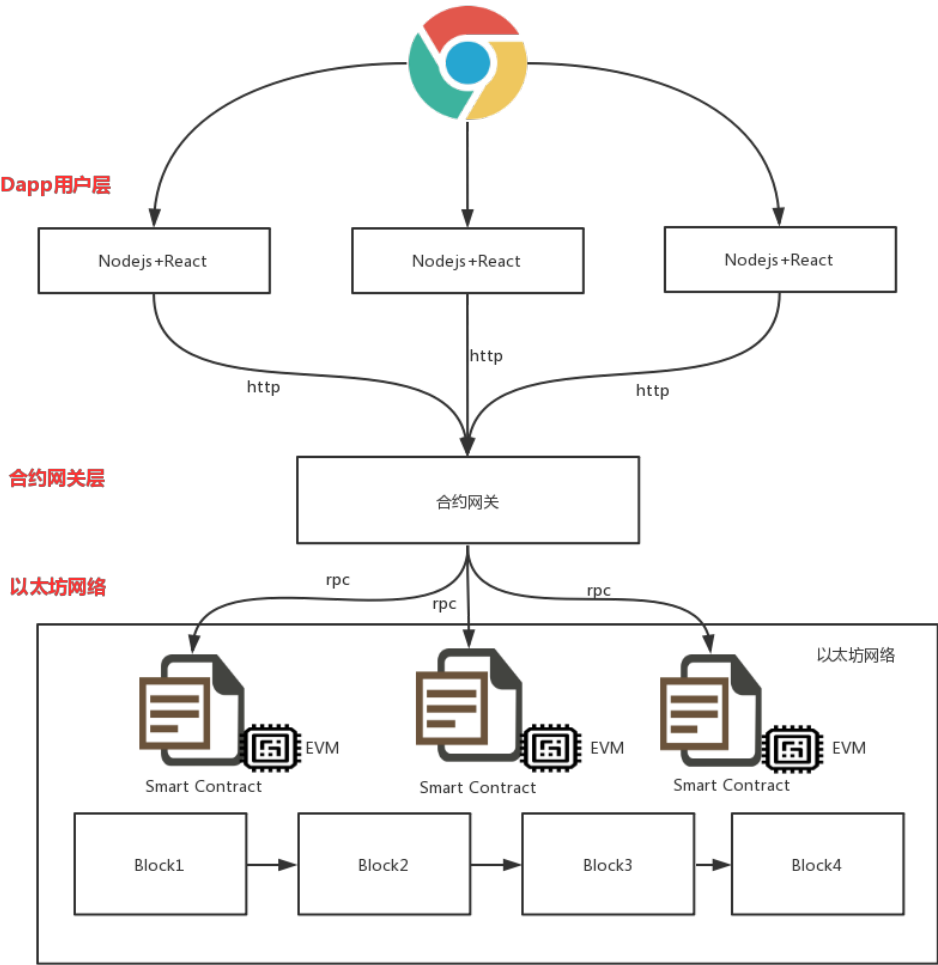
用户可以在便签板上添加新的便签或者修改已有的便签，所有在浏览器上打开了这个Dapp的用户都可以同步看到便签的更新，便签被更新的通知。

这是一个实现起来相对简单的Dapp，本文档会详细介绍它的实现过程。

🔗 标签板Dapp简介

这个Dapp主要由两部分组成，包括部署在以太坊网络中的智能合约以及依赖智能合约负责业务功能交互的web应用。智能合约采用solidity开发，而web应用没有后端服务器，而是直接调用以太坊节点接口，所以直接采用javascript开发，使用react框架。

Dapp组件间的调用关系如图：



合约网关是超级链BaaS平台为以太坊网络打造的应用中间层。在以往的开发中，如果要和以太坊节点通信，需要使用web3等库进行一系列繁琐复杂的流程，包括nonce管理、交易打包和封装、交易结果解析等，将开发人员的时间耗费在无谓的数据处理上。合约网关采用RESTful API设计，开发人员只需关注自己的业务逻辑，无需考虑底层的复杂交易管理，大大提升开发的效率，降低开发人员的负担。

我们所要开发便签板应用较为简单，核心功能是将用户创建的文本便签存储到区块链上。需要实现的关键用例只有三个：

1. 用户新建便签

用户点击新增，在编辑窗口中写入便签题目和便签内容，点击“上链”。应用将便签（便签题目、便签内容）存入，再将编辑窗口关闭。

2. 用户更新便签

用户点击需要更新的便签，在编辑窗口中修改便签题目或便签内容，点击“上链”。应用将便签（便签题目、便签正文）更新，再将编辑窗口关闭。

3. 用户查看所有便签

用户打开应用即可看到所有链上的便签，即便签板。便签分为便签题目和便签内容，在便签板上由于空间有限，每个便签题目和正文只能显示最多20个字符。用户可以点击某一个便签打开编辑更新面板，即可查看全部内容。

智能合约开发及部署

IDE环境准备

本文采用Solidity语言开发以太坊智能合约。Solidity的开发环境我们直接使用超级链BaaS平台系统中提供的Remix web开发平

台。因为超级链BaaS平台中提供的Remix IDE默认直接与目标以太坊网络相连，所以Remix IDE同时也可以作为部署、测试工具使用。

按照以下方法进入Remix IDE:

1. 进入以太坊列表页，找到想要部署智能合约的目标以太坊网络，点击右侧的智能合约编辑器，系统会弹出新浏览器tab

以太坊网络名称	状态	创建时间	操作
 test	 运行中	2019-11-04 20:41:40	以太坊控制台 以太坊浏览器 智能合约编辑器 智能合约管理 钱包账户管理

2. 在跳转前，会提示我们的账户密码，并在登录页中输入用户名和密码

登录

http://[redacted]:8084

您与此网站的连接不是私密连接

用户名

密码

取消

登录

3. 等待Remix IDE页面加载完成，智能合约IDE主面板已经可以使用

CompileRunSettingsAnalysisDebuggerSupportTest

Start to compile

☒ Auto compile

☐ Hide warnings

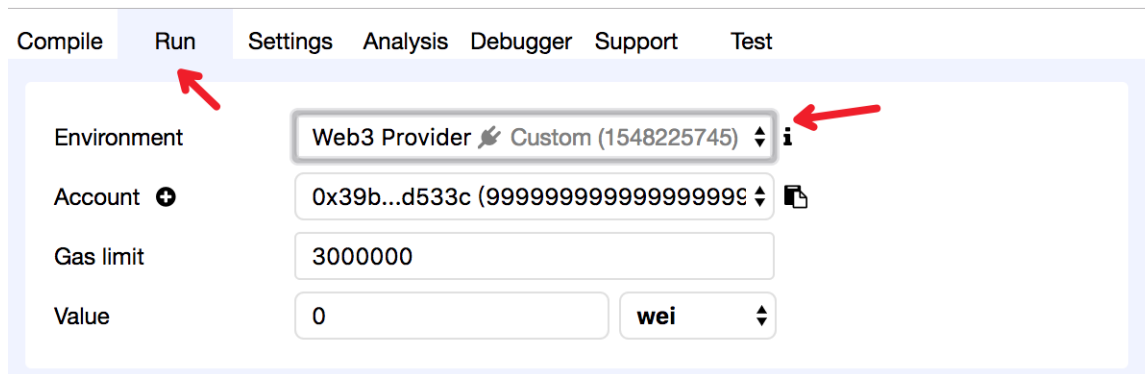
Details

Publish on Swarm

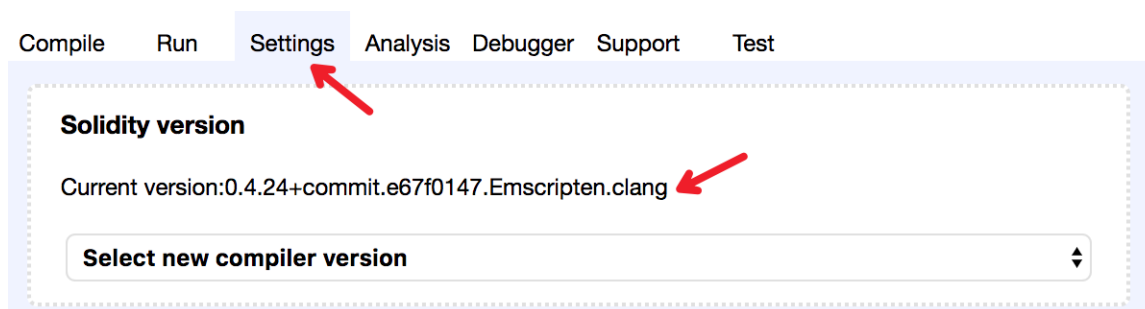
 ABI

 Bytecode

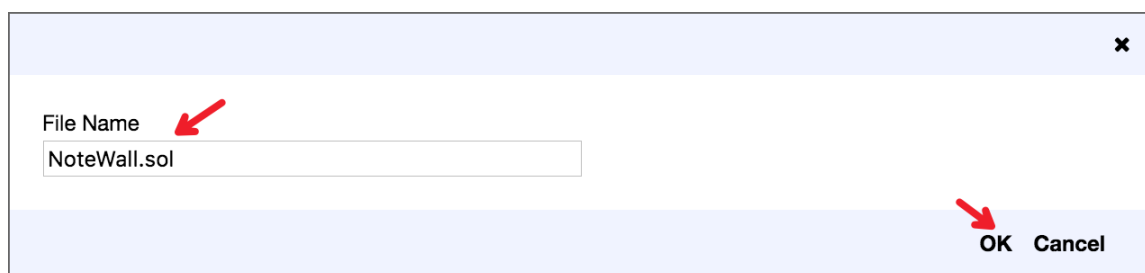
4. 在Run面板中确认已经连接目标以太坊的Web3 Provider



5. 在Settings面板选择0.4.24版本的Solidity



6. 在最左侧点击“+”，新建一个名为NoteWall.sol的智能合约



智能合约实现

编写智能合约的Solidity基础语法请读者自行阅读或参考完整源码。这里主要介绍便签板合约的主要实现。

便签对象的实现

便签板合约的实现关键就是定义一个便签在区块链上存储的数据格式。我们首先定义一个便签的结构体

```
struct Note {
string title;
string content;
uint256 id;
}
```

便签由三部分构成，string类型的标题，string类型的便签正文以及一个256位的无符号整数类型表示的便签ID。

便签组的实现

接下来就是便签“对象”们如何组织化地保存下来，这里我们需要再定义一个mapping。

```
mapping(uint256 => Note) notes;
```

这个mapping管理着从ID到Note对象的索引。从用例来分析，便签板需要用到查找的场景主要有两处，一处是用户点击便签，编辑便签更新，另一处是用户查看所有的便签。由于Solidity中mapping不具有遍历性，所以我们需要再维护一个记录便签ID的数组，数组本身是可遍历的。

```
uint256[] notelds;
```

新增便签的实现

新增一个便签除了需要新建一个Note对象，赋值title和content外，最主要的就是要为每一个Note对象分配一个不重复的ID。这里我们利用合约方法原子性的特点，通过维护一个累加的全局ID计数器来实现Note ID的唯一性。

具体原理就是当新建一个Note对象时，读取全局的ID计数器，将当前计数器值作为Note对象的ID，并将ID值++。

所以我们需要声明一个nextId，全局变量

```
uint256 nextId
```

接下来是新增便签的函数

```
/// Insert one new note object
function InsertNote(string _title, string _content) public {
    Note storage note = notes[nextId];
    notelds.push(nextId);
    note.id = nextId;
    note.title = _title;
    note.content = _content;
    nextId = nextId + 1;
}
```

新增一个需要存储的Note对象，然后将nextId赋给note，并填入对应的标题和正文；此外将note索引到notes mapping中，插入新ID到notelds数组中。

更新便签的实现

更新便签与新增便签稍有差别，需要从给定的Note ID中找到note对象再进行赋值。

```
function UpdateNote(uint256 _id, string _title, string _content) public {
    Note storage note = notes[_id];
    note.title = _title;
    note.content = _content;
}
```

获取所有的便签

按照用例3，用户应该会希望一次性获取所有的便签，但这在Solidity中无法通过一个方法就满足需求，原因是当前的solidity版本不支持方法返回一个struct或者struct的数组。所以我们拆分成两个函数来实现这个功能。第一个函数可以返回所有的便签ID，第二个函数则可以根据便签ID返回便签的标题和正文。

获取所有便签ID的函数：

```
/// Get all note ids
function GetNotelds() public view returns(uint256[] _notelds) {
    return notelds;
}
```

读取便签标题和正文

```
/// Get one note object by its id
function GetNote(uint256 id) public view returns(string, string) {
    return (notes[id].title, notes[id].content);
}
```

这里我们使用了多值返回的特性。

所以最终获取所有的便签需要上层通过两步完成，首先调用GetNotelds获取所有便签ID，之后再遍历便签ID，获取每个便签的标题和正文。虽然从传统软件设计看并不是最优的实现方式，但囿于EVM Solidity本身语法的局限性，智能合约的逻辑有时候需要做一些规避。

智能合约完整代码

```
pragma solidity ^0.4.0;
contract NoteWall {
    struct Note {
        string title;
        string content;
        uint256 id;
    }
    mapping(uint256 => Note) notes;
    uint256[] notelds;
    uint256 nextId;

    /// Create a new ballot with $_numProposals) different proposals.
    constructor() public {
        nextId = 0;
    }

    /// Get all note ids
    function GetNotelds() public view returns(uint256[] _notelds) {
        return notelds;
    }

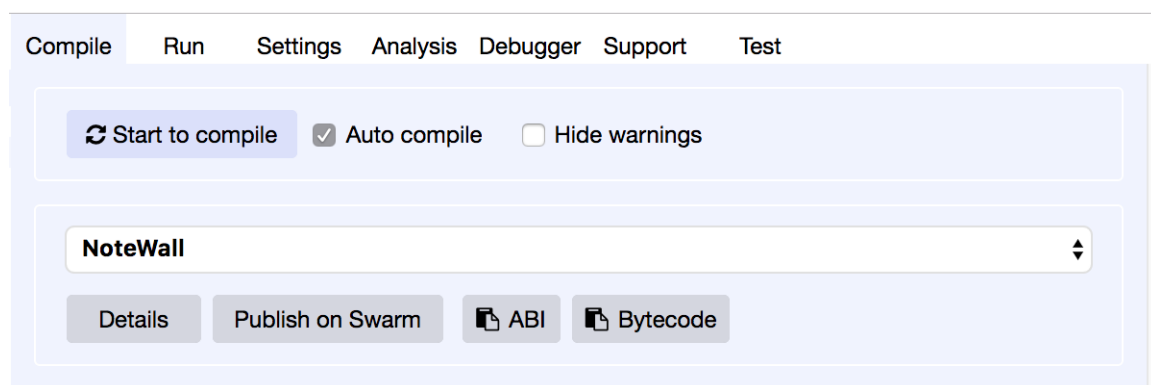
    /// Get one note object by its id
    function GetNote(uint256 id) public view returns(string, string) {
        return (notes[id].title, notes[id].content);
    }

    /// Insert one new note object
    function InsertNote(string _title, string _content) public {
        Note storage note = notes[nextId];
        notelds.push(nextId);
        note.id = nextId;
        note.title = _title;
        note.content = _content;
        nextId = nextId + 1;
    }

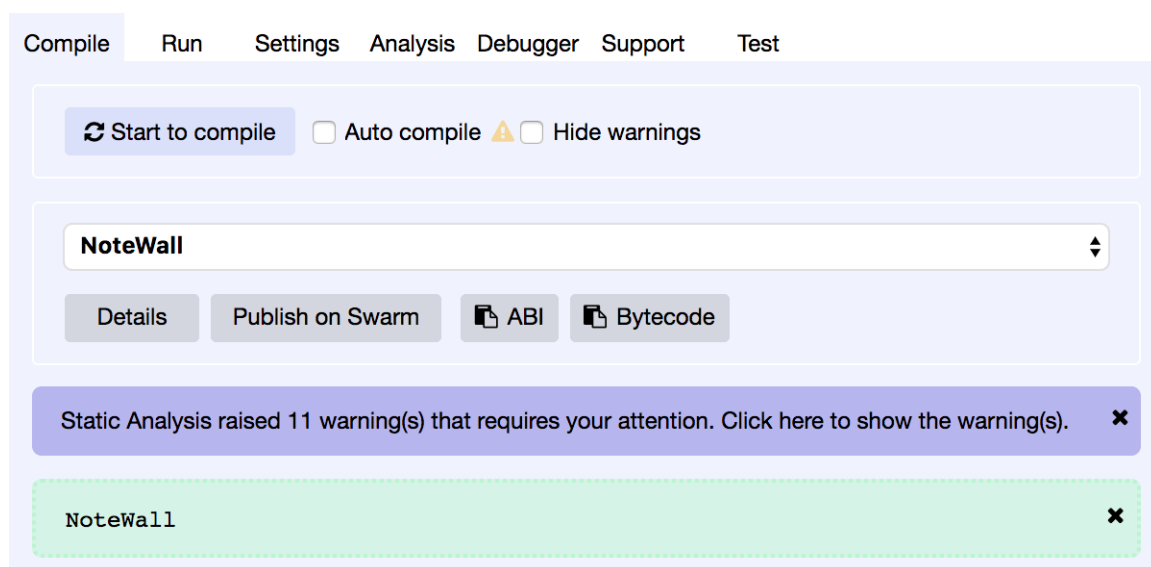
    /// Update one note by its id
    function UpdateNote(uint256 _id, string _title, string _content) public {
        Note storage note = notes[_id];
        note.title = _title;
        note.content = _content;
    }
}
```

Remix IDE会自动保存合约文件

在Remix中点击右侧的Compile面板



点击Start to compile进行编译，编译完成后如下图：



🔗 智能合约测试

Remix开发团队提供了Solidity的测试框架，可以开发完成简单的断言式单测。新建NoteWall_test.sol合约，此合约用来对NoteWall.sol做单测覆盖。NoteWall_test.sol的内容如下：

```
pragma solidity ^0.4.7;
import "remix_tests.sol"; // this import is automatically injected by Remix.
import "./NoteWall.sol";

contract testNoteWall {
    NoteWall noteWallToTest;
    function beforeAll () {
        noteWallToTest = new NoteWall();
    }

    /// test InsertNote function
    function checkInsertNote () public {
        noteWallToTest.InsertNote("note title", "note content");
        var notelds = noteWallToTest.GetNotelds();
        var lastNoteld = notelds[notelds.length - 1];
        var (title, content) = noteWallToTest.GetNote(lastNoteld);
        Assert.equal(title, "note title", "note title should be the note compositions");
        Assert.equal(content, "note content", "note content should be the note compositions");
    }

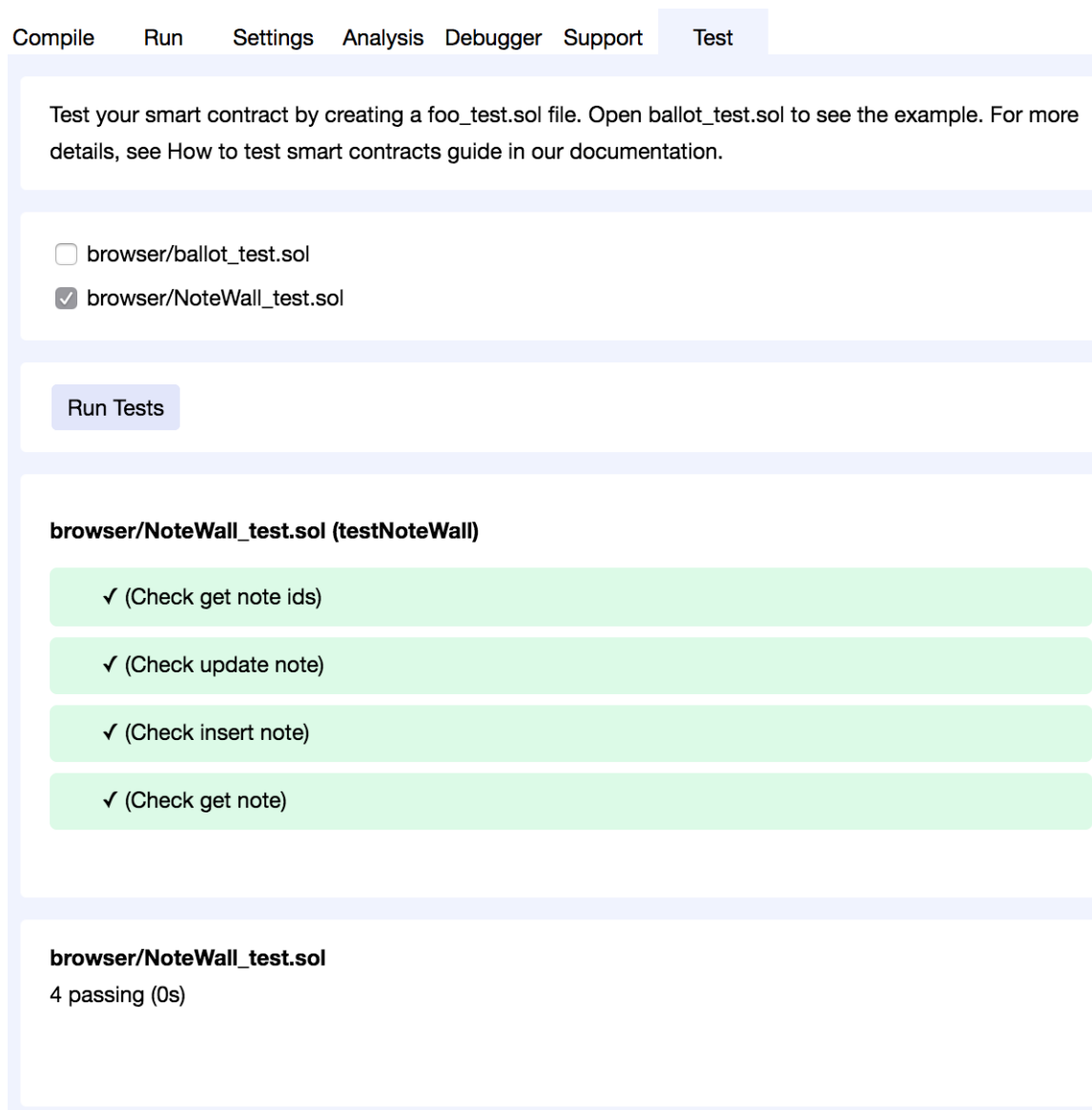
    /// test UpdateNote function
    function checkUpdateNote () public {
        noteWallToTest.InsertNote("note title", "note content");
        var notelds = noteWallToTest.GetNotelds();
        var lastNoteld = notelds[notelds.length - 1];
        noteWallToTest.UpdateNote(lastNoteld, "new title", "new content");
        var (title, content) = noteWallToTest.GetNote(lastNoteld);
        Assert.equal(title, "new title", "new title should be the updated note compositions");
        Assert.equal(content, "new content", "new content should be the updated note compositions");
    }

    /// test GetNotelds function
    function checkGetNotelds () public {
        noteWallToTest.InsertNote("note title 0", "note content 0");
        noteWallToTest.InsertNote("note title 1", "note content 1");
        noteWallToTest.InsertNote("note title 2", "note content 2");
        var notelds = noteWallToTest.GetNotelds();
        Assert.equal(notelds.length, 3, "notelds' length should equal 3");
    }

    /// test GetNote function
    function checkGetNote () public {
        noteWallToTest.InsertNote("note title", "note content");
        var notelds = noteWallToTest.GetNotelds();
        var lastNoteld = notelds[notelds.length - 1];
        var (title, content) = noteWallToTest.GetNote(lastNoteld);
        Assert.equal(title, "note title", "note title should be the note compositions");
        Assert.equal(content, "note content", "note content should be the note compositions");
    }
}
```

由于只是运用了remix_tests.sol中的断言语法，实现并不复杂，此处不做赘述。

点击右侧Test标签，进入测试面板。勾选browser/NoteWall_test.sol，点击“Run Tests”按钮开始测试。运行完单测后，测试结论应如下图：



智能合约部署

单测运行完成后，我们开始进行智能合约的部署。如果开发者对合约代码没有信心，可以先将合约部署到Remix自带的EVM虚拟机（Javascript VM）。由于过程是类似的，这里我们将说明直接把合约部署到以太坊网络中的过程。

点击右侧面板“Run”标签，将GasLimit填为3000000000，点击下方“Deploy”按钮进行合约部署。为了保护用户的安全，需要输入百度云的access key和secret key

部署信息 ✕

百度云AccessKeyID:

[去获取](#) [查看百度云AccessKeyID说明](#)

百度云SecretAccessKey:

[去获取](#) [查看百度云SecretAccessKey说明](#)

OK

Cancel

如果没有的话，可以按照提示获取或创建Access Key即可。

部署时长取决于您目标区块链网络的出块时长，当部署完成后，在console面板中可以看到交易的信息。如下图：

✔ **[block:115421 txIndex:0]** from:0x0de...4343a
 to:NoteWall.(constructor) value:0 wei data:0x608...d0029 logs:0
 hash:0x79f...65259 Debug ^

status	0x1 Transaction mined and execution succeed
transaction hash	0x79fd35b8fc119e70c5fcb5906c8096f15b541c6a8db219e478e95359fda65259
from	0x0de428247c564b8944a17f3c1d44dfe979c4343a
to	NoteWall.(constructor)
gas	605863 gas
transaction cost	605863 gas
hash	0x79fd35b8fc119e70c5fcb5906c8096f15b541c6a8db219e478e95359fda65259
input	0x608...d0029
decoded input	{ }
decoded output	-
logs	[]
value	0 wei

>

这里是详细的合约部署交易信息。

我们保留当前的窗口不要关闭，有几处重要的数据需要摘出来。

🔗 获取已部署合约的重要信息

获取合约地址

在右侧面板的“Deployed Contracts”中复制

Deployed Contracts

▼
 NoteWall at 0x37a...51fb3 (blockchain)

×

InsertNote

string_title, string_content

▼

UpdateNote

uint256 _id, string_title, string_content

▼

GetNote

uint256 id

▼

GetNotelds

此外，我们也可以从合约管理中心获取合约的相关信息

NoteWall

基本信息

合约名称:

NoteWall

合约地址:

0xc1882e900f9070c62f5f323bf238635a6417c632

合约状态:

运行中

合约创建时间:

2019-07-30 15:42:34

网络Id:

合约Uuid:

owner地址:

0x87264310fa4b3c130c09827ee4ec144804e7fdd0

合约类型:

自定义合约

合约更新时间:

2019-07-30 15:42:34

84

熟悉以太坊Dapp开发的用户可能会想到要提取ABI文件进行开发。但是通过合约网关调用智能合约，我们无需配置ABI文件，无需处理复杂的交易生成和解析，把相关交易参数发送给合约网关就可以实现合约方法的调用。

获取合约网关地址 我们的Dapp需要和合约网关通信，因此我们需要前往管理中心获取合约网关的通信地址

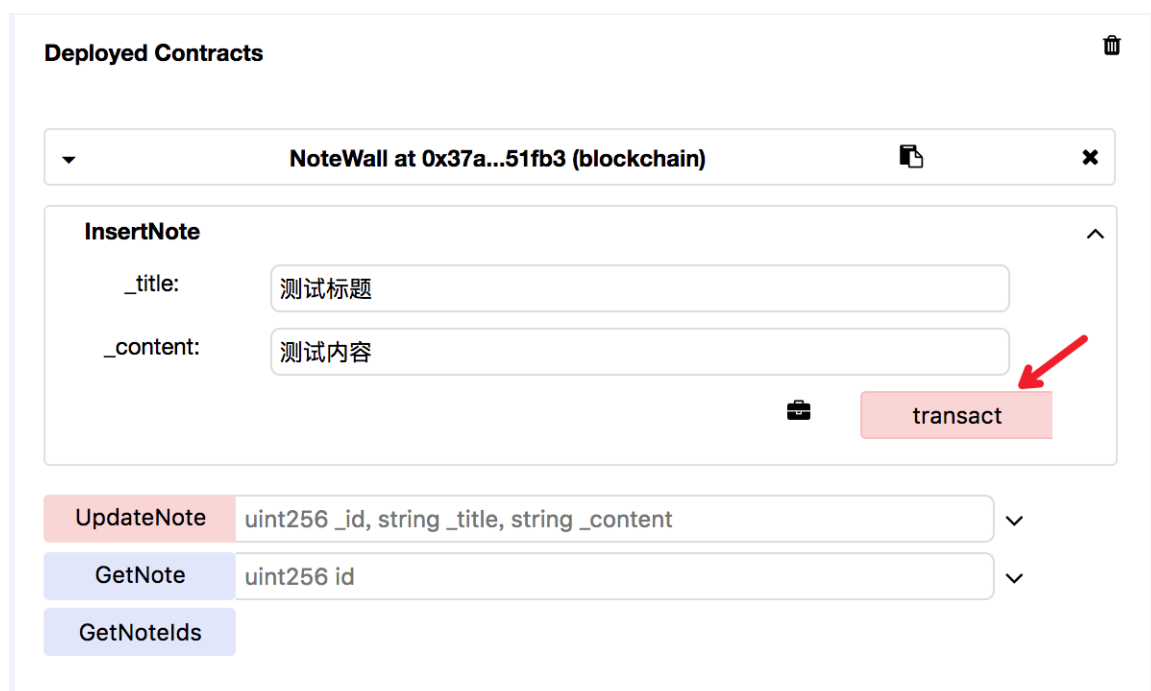


在Remix中调用已部署合约

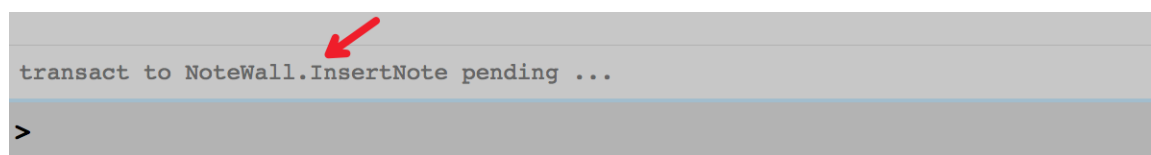
这一步主要是为了验证已部署合约是否正常工作。我们可以在Remix中直接调用刚才部署的合约。

验证InsertNote方法

在Run面板中打开InsertNote框，在_title和_content中填入相应的参数。下图为例：



点击“transact”发起调用交易，同上文一样，passphrase输入123。点击OK后交易发起。我们可以在Console中看到交易正在等待入块。



等待片刻，当交易入块后，我们在Console面板中可以看到交易详情

✔
[block:115512 txIndex:0]
from:0x0de...4343a
to:NoteWall.InsertNote(string,string) 0x37a...51fb3 value:0 wei
data:0x947...00000 logs:1 hash:0x659...7879d

Debug
⤴

status	0x1 Transaction mined and execution succeed
transaction hash	0x659783bb153fdc8c280e6befa5061557597899fba0ec286f4cd20cbcc6c7879d
from	0x0de428247c564b8944a17f3c1d44dfe979c4343a
to	NoteWall.InsertNote(string,string) 0x37a4d7c3800a81f6534a1761530d14828ec51fb3
gas	114228 gas
transaction cost	114228 gas
hash	0x659783bb153fdc8c280e6befa5061557597899fba0ec286f4cd20cbcc6c7879d
input	0x947...00000
decoded input	<pre>{ "string_title": "测试标题", "string_content": "测试内容" }</pre>
decoded output	-
logs	<pre>[{ "from": "0x37a4d7c3800a81f6534a1761530d14828ec51fb3", "topic": "0xce11699641de10a7a17a1a2b2991db19349dfd897c44699d5b59b8b5d0539c1f", "event": "NoteEvent", "args": { "0": "测试标题", "1": "测试内容", "title": "测试标题", "content": "测试内容", "length": 2 } }]</pre>
value	0 wei

验证GetNotelds方法

在Run面板中点击最下方的GetNotelds，由于是只读方法，不需要发起交易，面板上应该会直接返回结果，应是一个元素的数组。

Deployed Contracts

NoteWall at 0x37a...51fb3 (blockchain)

InsertNote

_title:

string

_content:

string

transact

UpdateNote

uint256 _id, string _title, string _content

GetNote

uint256 id

GetNotelds

0: uint256[]: _notelds 0

验证GetNote方法

在Run面板中打开最下方的GetNote，在id字段中填入0，点击call。由于也是只读方法，不需要发起交易，面板上应该会直接返回结果，应是刚才新建的Note信息。

Deployed Contracts

NoteWall at 0x37a...51fb3 (blockchain)

InsertNote

string _title, string _content

UpdateNote

uint256 _id, string _title, string _content

GetNote

id:

uint256

call

0: string: 测试标题

1: string: 测试内容

GetNotelds

0: uint256[]: _notelds 0

验证UpdateNote方法

我们再验证UpdateNote方法，已知已有NoteId为0，填入新的标题和内容。由于需要发起新的交易，所以需要稍作等待。

Deployed Contracts



NoteWall at 0x37a...51fb3 (blockchain)

InsertNote string_title, string_content

UpdateNote

_id:

0

_title:

更新后的标题

_content:

更新后的正文

transact

交易完成：

[block:115553 txIndex:0] from:0x0de...4343a

to:NoteWall.UpdateNote(uint256,string,string) 0x37a...51fb3

value:0 wei data:0xc69...00000 logs:1 hash:0xa3a...18208

Debug

status	0x1 Transaction mined and execution succeed
transaction hash	0xa3aba438b1aac2baafddb32486ab9b78721ad075b9bddbef8cac59c256618208
from	0x0de428247c564b8944a17f3c1d44dfe979c4343a
to	NoteWall.UpdateNote(uint256,string,string) 0x37a4d7c3800a81f6534a1761530d14828ec51fb3
gas	49593 gas
transaction cost	49593 gas
hash	0xa3aba438b1aac2baafddb32486ab9b78721ad075b9bddbef8cac59c256618208
input	0xc69...00000
decoded input	{ "uint256 _id": "0", "string _title": "更新后的标题", "string _content": "更新后的正文" }
decoded output	-
logs	[{ "from": "0x37a4d7c3800a81f6534a1761530d14828ec51fb3", "topic": "0xcel1699641de10a7a17a1a2b2991db19349dfd897c44699d5b59b8b5d0539c1f", "event": "NoteEvent", "args": { "0": "更新后的标题", "1": "更新后的正文", "title": "更新后的标题", "content": "更新后的正文", "length": 2 } }]
value	0 wei

重新GetNote：

Deployed Contracts



▼

NoteWall at 0x37a...51fb3 (blockchain)

×

InsertNote

string_title, string_content

▼

UpdateNote

uint256_id, string_title, string_content

▼

GetNote

id: uint256

call

0: string: 更新后的标题

1: string: 更新后的正文

重新GetNotelds，确认没有新增加Note：

Deployed Contracts



▼

NoteWall at 0x37a...51fb3 (blockchain)

×

InsertNote

string_title, string_content

▼

UpdateNote

uint256_id, string_title, string_content

▼

GetNote

uint256 id

▼

0: string: 更新后的标题

1: string: 更新后的正文

GetNotelds

0: uint256[]: _notelds 0

到此，智能合约的开发、测试、部署、验证步骤已经完成了。接下来我们开始进行上层应用层的开发。

Dapp应用层开发

Dapp应用层要做什么

Dapp应用层最主要的是实现用户侧的交互逻辑，包括web页面和页面事件响应。同时不同的操作会带来不同的事件，我们还需要针对页面事件去调用后端智能合约方法，存入便签、更新便签、读取便签等。由于是基于React框架做的实现，我们首先要准备一套Dapp应用层的开发环境。

Dapp应用层开发环境准备

- 首先需要安装nodejs环境，本文采用的是v9.11.2版本，npm版本为v5.6.0。我们可以使用nvm安装，也可以通过系统自带的包管理安装。具体的下载安装方法这里不再赘述，可参考官方文档：[Node.js下载安装指导](#)
- 准备好nodejs和npm环境后，我们可以一键创建一个React项目。根据不同的环境，我们有两种命令。

```
$ npx create-react-app notewall-dapp
or
$ create-react-app notewall-dapp
```

3. 执行完之后，我们可以进入notewall-dapp目录，目录里是一个标准的未装任何依赖的项目目录。这个项目里我们需要安装如下依赖：

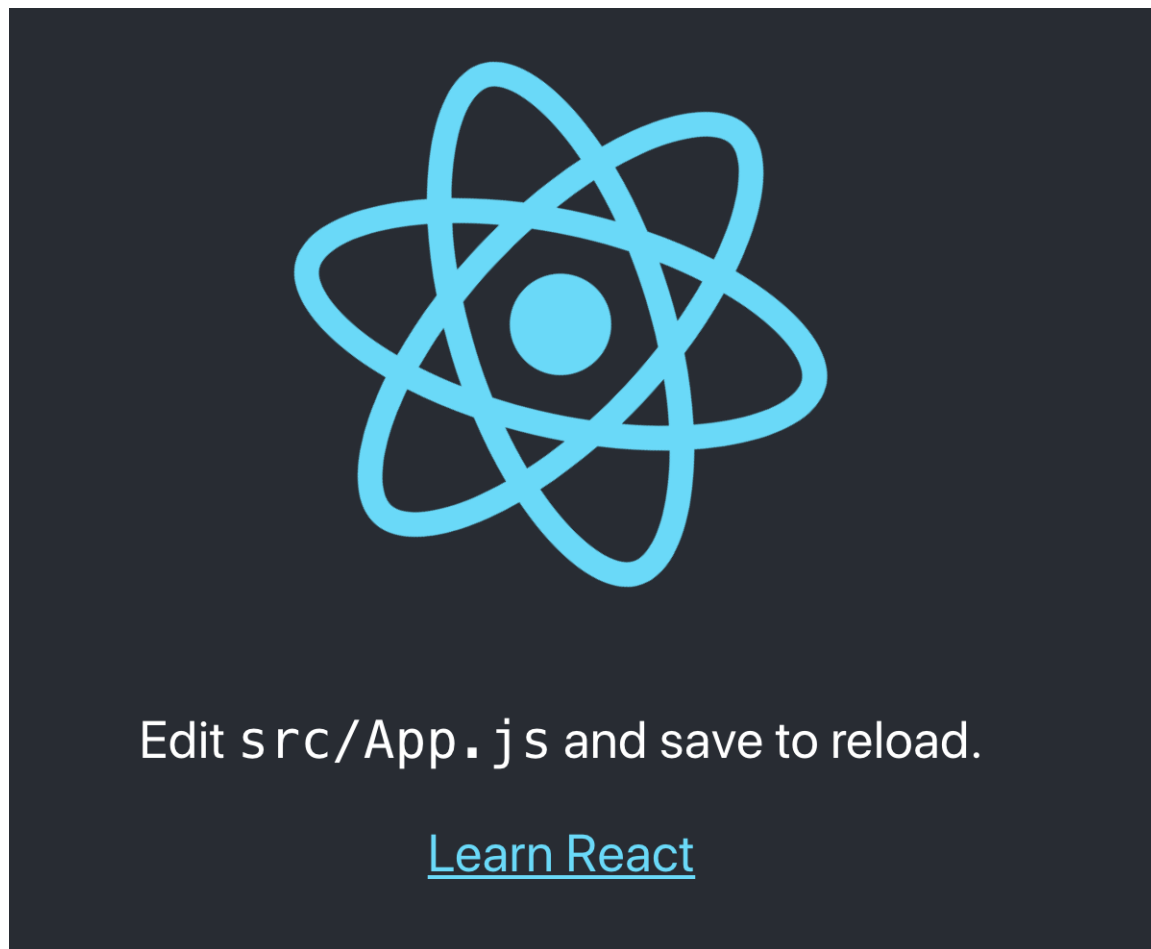
```
$ cd notewall-dapp  
$ npm install --save react-toastify react-bootstrap axios
```

其中react-toastify是一个第三方的弹窗通知库；axios是一个http请求库，完成我们对合约网关的请求。

4. 试着启动一下

```
$ npm start
```

npm会启动这个react项目，并且将http server监听在本地3000端口。浏览器访问<http://localhost:3000>可以查看结果。



如果我们看到了如下页面显示，则代表项目环境初始化完成。

为了能让项目运行我们的便签，我们需要更改根目录的App.js如下。

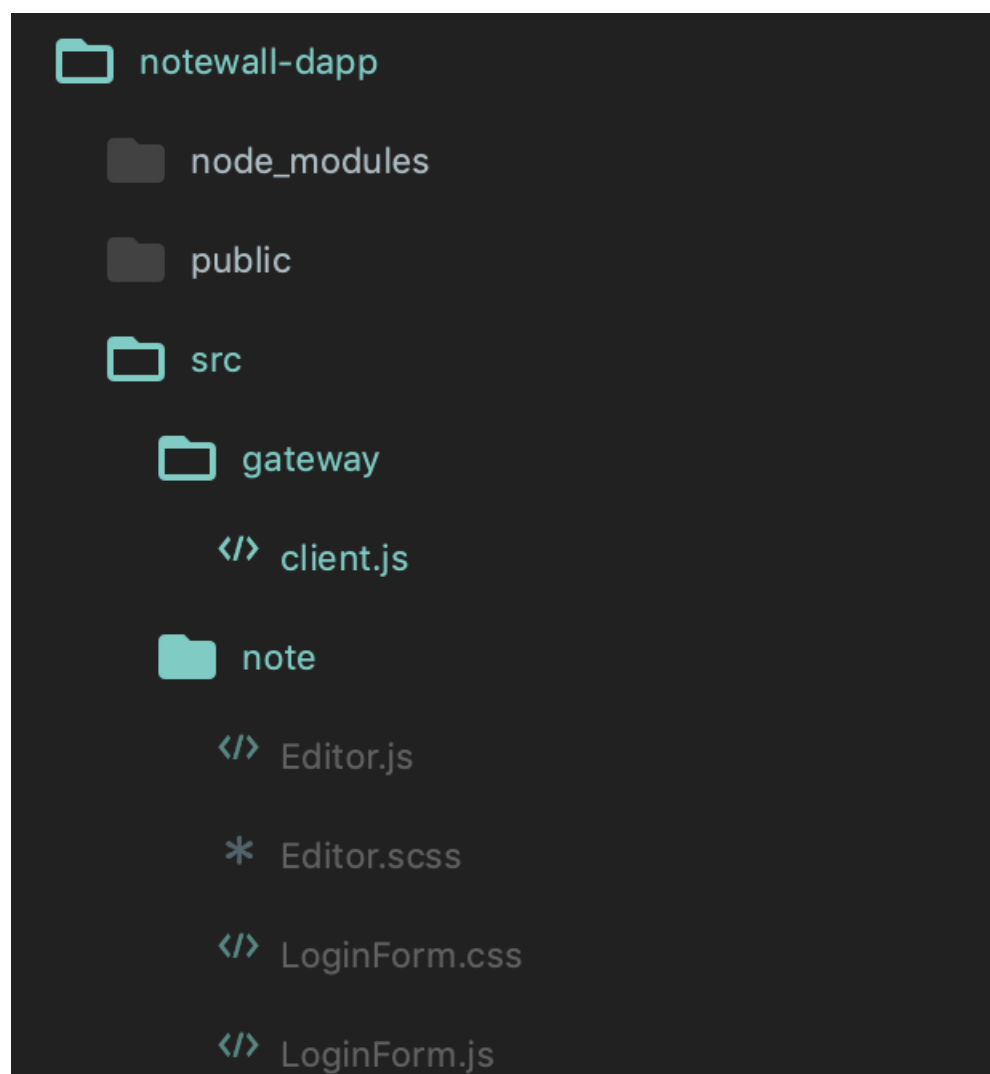
```
/**
 * @file notewall app entry
 * @author XUPERBAAS Team
 */
import React, {Component} from 'react';
import './App.css';
import NoteWall from './note/NoteWall.js';

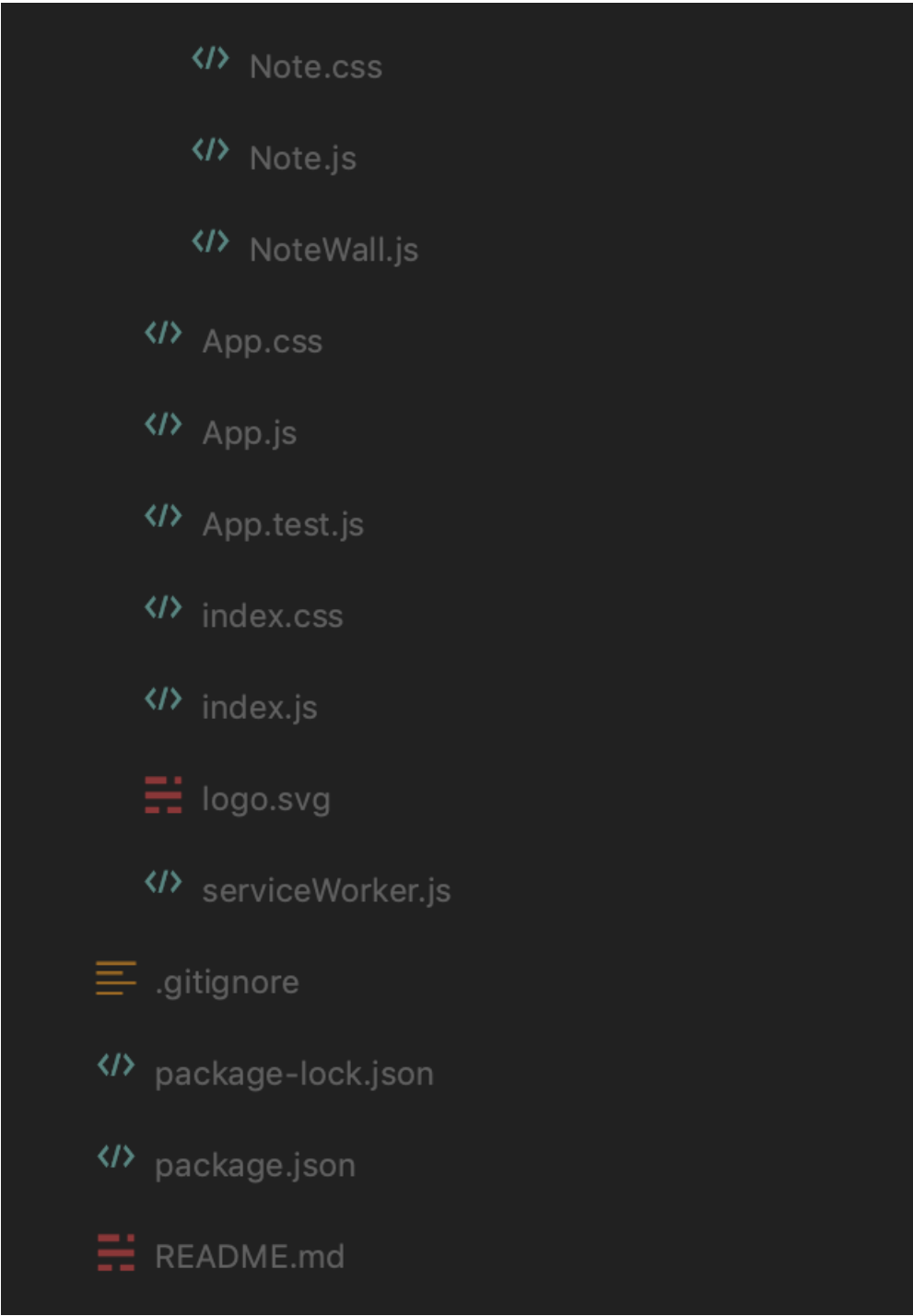
export default class App extends Component {
  render() {
    return (
      <div className='App'>
        <header className='App-header'>
          <NoteWall />
        </header>
      </div>
    );
  }
}
```

🔗 Dapp应用层开发

如上所述，Dapp应用层主要分为两大部分，一个是页面展示与交互逻辑，另一个是与智能合约的交互。与传统web应用需要后端服务器不同的是，Dapp应用层只需要前端部分，Dapp的后端是区块链节点。我们通过调用以太坊节点上的接口向智能合约发起调用。而这些调用全部需要在前端Javascript中完成。

我们可以规划项目目录，新建两个子目录分别为gateway和note。gateway中主要实现与合约网关的交互逻辑，note目录中主要实现用于前端展示的组件。最终的项目目录如下图所示。





```
</> Note.css
</> Note.js
</> NoteWall.js
</> App.css
</> App.js
</> App.test.js
</> index.css
</> index.js
logo.svg
</> serviceWorker.js
.gitignore
</> package-lock.json
</> package.json
README.md
```

合约网关交互

由于前端页面展示风格千变万化，我们可以先从与合约网关交互这一个相对靠后的逻辑层次实现起来。与智能合约网关的交互用的是http请求，我们直接实用一个http库axios进行通信。我们从实际情况考虑，Dapp是将合约的相关操作和结果展示在前端页面，所以我们完成和合约网关的交互，开发对应的前端页面，就可以完成开发Dapp的需求。

结合最初的用例分析，我们需要在三个事件中与智能合约进行交互。

1. 当用户打开应用时，需要展示所有的便签
2. 当用户新增便签点击上链后，需要将便签内容保存。相对于我们已经实现的智能合约是InsertNote操作。
3. 当用户更新便签点击上链后，需要将便签内容的更新保存下来。在合约中是UpdateNote操作。

1、Client构造

我们将这些交互抽象到一个NotewallGatewayClient中，在gateway中新建一个client.js。新建一个NoteWallGatewayClient类


```

import Axios from 'axios';
export default class NoteWallGatewayClient {
  /**
   * @param gatewayAddress {string} 合约网关地址
   * @param address {string} 合约部署地址
   * @param senderAddress {string} 发起交易的账户地址
   * @param senderPrivateKey {string} 发起交易的账户私钥
   */
  constructor(gatewayAddress, address, senderAddress, senderPrivateKey) {
    this.baseUrl = gatewayAddress + "/v2/ethereum/tx/";
    this.address = address;
    this.senderPrivateKey = senderPrivateKey;
    this.sender = senderAddress;
    this.notes = {};
    // 合约网关采用basic auth认证, 对合约网关的http请求设置一个拦截器, 配置相应的header内容。其中account和password是我们之前获取的用户名和密码。
    Axios.interceptors.request.use((config) => {
      config.headers={
        "Authorization":'Basic '+btoa('admin:9581564b-93d9-d5bd-ea97-d639d83ca32c'),
        "Content-Type":'application/json',
      }
      return config;
    },(error) =>{
      return Promise.reject(error);
    });
  }
}

```

通过合约网关, 分为交易类和非交易类型, 这两种类型请求的参数有所不同, 我们将这两种类型的请求封装一下方便后续调用。

```

// 交易类型请求
transaction(invoke, method){
  let url = this.baseUrl+this.address+"/"+method
  return Axios.post(url,{
    from:this.sender,
    privateKey:this.senderPrivateKey,
    invokeParameters:invoke
  })
}
// 非交易类型请求
call(invoke, method){
  let url = this.baseUrl+this.address+"/"+method
  return Axios.post(url,{
    from:this.sender,
    invokeParameters:invoke
  })
}

```

创建交易类请求后会得到交易Hash, 合约网关提供了根据交易Hash查询上链结果查询接口, 我们对该接口进行封装。

```

replyTx(id){
  let url = this.baseUrl+id
  return Axios.get(url)
}

```

2、交易类方法调用

这里我们先实现创建note和更新note两个交易类的方法。

```

/**
 * @param title {string} 便签标题
 * @param content {string} 便签正文
 * @param callback {function} 回调函数
 */
insertNote(title, content, callback) {
  let insert={
    "_title": title,
    "_content": content
  }
  this.transaction(insert, "InsertNote").then(resp => {
    let timerId=setInterval(()=>{
      this.replyTx(resp.data.result.txId).then(resp =>{
        if (Object.keys(resp.data.result).length !== 0) {
          clearInterval(timerId);
          callback(null, resp.data.result)
        }
      })
    },3000);
  });
}

/**
 * @param id {int} 便签id
 * @param title {string} 便签标题
 * @param content {string} 便签正文
 * @param callback {function} 回调函数
 */
updateNote(id, title, content, callback) {
  let update={
    "_id": id,
    "_title": title,
    "_content": content
  }
  this.transaction(update, "UpdateNote").then(resp => {
    console.log(resp);
    let timerId=setInterval(()=>{
      this.replyTx(resp.data.result.txId).then(resp =>{
        if (Object.keys(resp.data.result).length !== 0) {
          clearInterval(timerId);
          callback(null,resp.data.result);
        }
      })
    },3000);
  });
}

```

可以看到，我们进行创建和更新note的代码很简单，只需要将请求的合约函数相关参数组装为object对象，然后向合约网关发起post请求接受交易Id结果即可；此外，为了方便检测交易结果是否上链，我们设置了一个定时器，每隔3s检查下交易的状态。

3、非交易类方法调用

非交易类方法主要为view类型的合约方法，我们可以使用合约网关提供的接口直接调用相关的合约方法，会得到合约返回结果。

在具体用例中，我们需要粘连合约中GetNoteIds和GetNote方法。从而满足页面渲染时直接展示所有合约的需求。

这里我们定义一个getAllNotes方法：

```

/**
 * callback的参数为notes
 * notes格式为map[noteld] {
 *   id {string}
 *   title {string}
 *   content {string}
 * }
 */
getAllNotes(callback){
  let getNotelds = {
  }
  this.call(getNotelds, "GetNotelds").then(async resp => {
    let notelds = resp.data.result._notelds;
    for (const noteld of notelds) {
      let id = {
        "id": noteld
      }
      this.call(id, "GetNote").then(resp => {
        let results=resp.data.result;
        let note={id:noteld};
        note["title"]=results.Result0;
        note["content"]=results.Result1;
        this.notes[noteld] = note;
      });
    }
    return this.notes;
  }).then(notes => {
    callback(notes);
  });
}

```

到此，与合约交互部分的逻辑都已经开发完成了。NoteWallWeb3Client向外暴露三个方法分别是getAllNotes、insertNote、updateNote。

页面开发

开发页面前，我们首先将页面交互元素做一个拆分。主要的交互模块有：

1. 登录表单，登录表单需要用户填写合约地址、发起交易的账户信息等。用户只有提交了登录表单才能查看合约。
2. 便签，每一个便签样式类似，但内容不同，一个便签对应合约存储中的一个便签实例。
3. 便签板，上面挂载所有的便签元素，并控制其他模块是否渲染。
4. 编辑器，编辑器用来给用户创建和更新便签时写入内容用。应该允许用户输入标题、内容。

1、导入样式依赖

本项目会依赖到bootstrap样式表，所以在public/index.html中我们插入bootstrap.css

```

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.0/css/bootstrap.min.css"
integrity="sha384-PDle/QlglONtM1aqA2Qemk5gPOE7wFq8+Em+G/hmo5lqOCCmYZLv3fVRDJ4MMwEA"
crossorigin="anonymous">

```

2、登录表单

登录表单对应到web3Client的参数，需要用户输入以太坊JSONRPC地址、合约地址、交易账号、交易秘钥四个参数。如图：



The image shows a login form with a light blue background and a dark blue border. It contains four input fields with labels in Chinese: '合约网关地址' (Contract Gateway Address), '合约地址' (Contract Address), '交易账号' (Transaction Account), and '交易密钥' (Transaction Private Key). The first field is highlighted with a blue border. Below the fields is a blue button labeled '登录' (Login).

我们直接使用react-bootstrap类库构建表单，当用户点击登录时，会将变量反传到父组件。

```
/**
 * @file login form component
 * @author BaaS Team
 */
import React, {Component} from 'react';
import {
  Button,
  Form,
  Jumbotron
} from 'react-bootstrap';
import './LoginForm.css';

export default class LoginForm extends Component {
  constructor(props) {
    super(props);

    this.state = {
      gatewayAddress: '',
      contractAddress: '',
      senderAddress: '',
      senderPrivateKey: ''
    };
  }

  // 这里我们只做最简单的字符串长度校验，有兴趣的读者可以引用web3库的地址Formatter做更标准的地址检查
  validateForm() {
    return this.state.gatewayAddress.length > 0
      && this.state.contractAddress.length > 0
      && this.state.senderAddress.length > 0
      && this.state.senderPrivateKey.length > 0;
  }
}
```

```

// 输入元素内容有变更时即赋值
handleChange = event => {
  this.setState({
    [event.target.id]: event.target.value
  });
}

// 这里会调用父组件传入的saveContractInfo方法，通知上层表单提交
handleSubmit = event => {
  this.props.saveContractInfo(this.state.gatewayAddress, this.state.contractAddress,
    this.state.senderAddress, this.state.senderPrivateKey);
}

render() {
  return (
    <div className='LoginForm'>
      <Jumbotron>
        <form onSubmit={this.handleSubmit}>
          <Form.Group controlId='gatewayAddress'>
            <Form.Label>合约网关地址</Form.Label>
            <Form.Control
              autoFocus
              type='text'
              value={this.state.gatewayAddress}
              onChange={this.handleChange}
              size='sm'
            />
          </Form.Group>
          <Form.Group controlId='contractAddress'>
            <Form.Label>合约地址</Form.Label>
            <Form.Control
              value={this.state.contractAddress}
              onChange={this.handleChange}
              type='text'
              size='sm'
            />
          </Form.Group>
          <Form.Group controlId='senderAddress'>
            <Form.Label>交易账号</Form.Label>
            <Form.Control
              value={this.state.senderAddress}
              onChange={this.handleChange}
              type='text'
              size='sm'
            />
          </Form.Group>
          <Form.Group controlId='senderPrivateKey'>
            <Form.Label>交易密钥</Form.Label>
            <Form.Control
              value={this.state.senderPrivateKey}
              onChange={this.handleChange}
              type='password'
              size='sm'
            />
          </Form.Group>
          <Form.Group>
            <Button block
              size="large"
              type="submit"
              variant="primary"
              disabled={!this.validateForm()}
              className="mt-5"
            />
          </Form.Group>
        </form>
      </Jumbotron>
    </div>
  );
}

```

```
        >登录</Button>
      </Form.Group>
    </form>
  </Jumbotron>
</div>
);
}
}
```

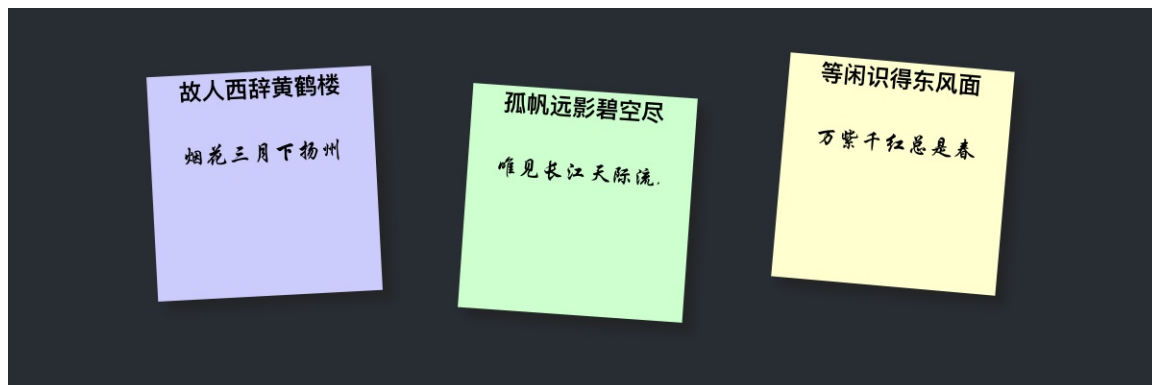
登录表单样式LoginForm.css如下。

```
@media all and (min-width: 480px) {
  .LoginForm {
    padding: 60px 0;
  }

  .LoginForm form {
    max-width: 540px;
    width: 540px;
    margin: 0 auto;
    font-size: 15px;
    text-align: left;
    color: #111;
  }
}
```

便签

便签模块主要是在样式控制上，这里我们将便签设计成不同颜色区分，会有不同倾斜角的正方形卡片。标题文字加粗。如效果图：



Note元素组件定义：

```

/**
 * @file note component
 * @author BaaS Team
 */
import React, {Component} from 'react';
import './Note.css';

/**
 * @props
 * id {int}
 * title {string}
 * content {string}
 * onClick {function} 被点击时需要调用，上层组件做响应
 */
export default class Note extends Component {
  render() {
    const {id, title, content} = this.props;
    return (
      <div className="note" onClick={() =>
        this.props.onClick(id, title, content)}>
        <div className="note-panel">
          <h2>{this.props.title}</h2>
          <p>{this.props.content.substring(0, 20)}</p>
        </div>
      </div>
    );
  }
}

```

样式控制代码：

```

body {
  margin: 1em;
}

.note-wall,
.note {
  list-style: none;
}

.note-wall {
  overflow: hidden;
  padding: 3em;
}

.note-panel {
  display: block;
  width: 5em;
  height: 5em;
  padding: 2px;
  text-decoration: none;
  color: #000;
  background: #ffc;
  -webkit-box-shadow: 5px 5px 7px rgba(33, 33, 33, .7);
  -moz-box-shadow: 5px 5px 7px rgba(33, 33, 33, 1);
  box-shadow: 5px 5px 7px rgba(33, 33, 33, .7);
  -webkit-transition: -webkit-transform .15s linear;
  -moz-transition: -moz-transform .15s linear;
  -o-transition: -o-transform .15s linear;
}

```

```
.note {
  float: left;
  margin: 1em;
}

.note-panel h2 {
  padding-bottom: 10px;
  font-size: 20%;
  font-weight: 700;
}

.note-panel p {
  font-family: "STXingkai", "Reenie Beanie", "Microsoft Yahei", arial, sans-serif;
  font-size: 20%;
}

.note-panel {
  -webkit-transform: rotate(-6deg);
  -moz-transform: rotate(-6deg);
  -o-transform: rotate(-6deg);
}

.note-wall div:nth-child(even) .note-panel {
  position: relative;
  top: 5px;
  background: #cfc;
  -webkit-transform: rotate(4deg);
  -moz-transform: rotate(4deg);
  -o-transform: rotate(4deg);
}

.note-wall div:nth-child(3n) .note-panel {
  position: relative;
  top: -5px;
  background: #ccf;
  -webkit-transform: rotate(-3deg);
  -moz-transform: rotate(-3deg);
  -o-transform: rotate(-3deg);
}

.note-wall div:nth-child(5n) .note-panel {
  position: relative;
  top: -10px;
  -webkit-transform: rotate(5deg);
  -moz-transform: rotate(5deg);
  -o-transform: rotate(5deg);
}

.note-panel:focus {
  position: relative;
  z-index: 5;
  -webkit-box-shadow: 10px 10px 7px rgba(0, 0, 0, .7);
  -moz-box-shadow: 10px 10px 7px rgba(0, 0, 0, .7);
  box-shadow: 10px 10px 7px rgba(0, 0, 0, .7);
  -webkit-transform: scale(1.25);
  -moz-transform: scale(1.25);
  -o-transform: scale(1.25);
}
```

1、便签板

便签板组件负责挂载其余组件，并负责通过子组件的事件回调完成子组件的通信。主要有以下核心控制逻辑：1.当用户未登录时，不显示便签板，只显示登录表单。登录表单提交后，初始化gatewayClient，为后续渲染便签做准备。便签板NoteWall.js代码如下。

```
/**
 * @file notewall class. main component in app.
 * @author BaaS Team
 */
import React, {Component} from 'react';
import LoginForm from './LoginForm.js';
import NoteWallGatewayClient from '../gateway/client';

export default class NoteWall extends Component {
  constructor(props) {
    super(props);
    this.state = {
      login: false
    };
    this.saveContractInfo = this.saveContractInfo.bind(this);
    this.gatewayClient = null;
    // 控制是否登录
    this.login = false;
  }

  // 定义了LoginForm回调方法，初始化gatewayClient
  // gatewayClient将被用于获取便签和创建、更新便签
  saveContractInfo(gatewayAddress, contractAddress, senderAddress, senderPrivateKey) {
    this.gatewayClient = new NoteWallGatewayClient(gatewayAddress, contractAddress, senderAddress,
senderPrivateKey);
    this.setState({login: true});
  }

  render() {
    if (!this.state.login) {
      return (
        <div className='note-wall'>
          <LoginForm saveContractInfo={this.saveContractInfo} />
        </div>
      );
    }
  }
}
```

2.当用户登录后，增加渲染便签。全量便签数据通过gatewayClient周期性获取。

```
/**
 * @file notewall class. main component in app.
 * @author BaaS Team
 */
import React, {Component} from 'react';
import LoginForm from './LoginForm.js';
import Note from './Note.js';
import NoteWallGatewayClient from '../gateway/client.js';

export default class NoteWall extends Component {
  constructor(props) {
    super(props);
    this.state = {
      editorShow: false,
      login: false,
```

```

    noteList: []
  };

  this.periodicGetList = this.periodicGetList.bind(this);
  this.saveContractInfo = this.saveContractInfo.bind(this);
  this.gatewayClient = null;
  this.login = false;
}

saveContractInfo(gatewayAddress, contractAddress, senderAddress, senderPrivateKey) {
  this.gatewayClient = new NoteWallGatewayClient(gatewayAddress, contractAddress, senderAddress,
senderPrivateKey);
  this.setState({login: true});
}

// Fetch the list on first mount
componentWillMount() {
  this.getListFromGateway();
  this.periodicGetList();
}

// 每三秒获取一次全量note列表
periodicGetList() {
  setTimeout(function () {
    this.getListFromGateway();
    this.periodicGetList();
  }
    .bind(this),
    3000
  );
}

// 获取全量note列表
getListFromGateway() {
  // 未登录时由于没有client，跳过查询
  if (!this.state.login) {
    return;
  }
  this.gatewayClient.getAllNotes(notes => {
    this.setState({
      noteList: notes
    });
  });
}

render() {
  let list = this.state.noteList;
  // 如果已登录，则渲染便签
  if (this.state.login) {
    return (
      <div className='note-wall'>
        <ToastContainer className='toast-notification' />
        {
          Object.keys(list).map(noteId => {
            let note = list[noteId];
            return (
              <Note key={note.id} id={note.id} title={note.title}
                content={note.content} onClick={this.showEditor} />
            );
          })
        }
      </div>
    );
  }
}

```

```

    }
    return (
      <div className='note-wall'>
        <LoginForm saveContractInfo={this.saveContractInfo} />
      </div>
    );
  }
}

```

3. 增加editor便签编辑器的控制。

便签编辑器在新建便签和更新便签时需要显示。当更新便签时需要知道便签的ID、标题和正文，所以需要当做参数传入。另外只有当用户需要创建和更新时才需要显示编辑器，所以编辑器需要能够响应这些事件，控制是否显示。同时，编辑器负责便签的存入，需要调用合约接口，需要用到gatewayClient。另外，编辑器存入便签后的相关receipt或者错误数据应该能够通知给用户，所以需要传入回调用的通知函数。

目前为止，因为编辑器还没有实现，我们先假定编辑器组件名字为Editor。这样NoteWall.js文件将更新成。

```

/**
 * @file notewall class. main component in app.
 * @author BaaS Team
 */
import React, {Component} from 'react';
import LoginForm from './LoginForm.js';
import Note from './Note.js';
import Editor from './Editor.js';
import NoteWallGatewayClient from '../gateway/client.js';

export default class NoteWall extends Component {
  constructor(props) {
    super(props);
    this.state = {
      editorShow: false,
      login: false,
      noteList: []
    };
    this.showEditor = this.showEditor.bind(this);
    this.closeEditor = this.closeEditor.bind(this);
    this.periodicGetList = this.periodicGetList.bind(this);
    this.saveContractInfo = this.saveContractInfo.bind(this);
    this.notify = this.notify.bind(this);
    this.errNotify = this.notify.bind(this);
    this.gatewayClient = null;
    this.login = false;
  }

  // 开关编辑器用
  showEditor(id, title, content) {
    this.setState({
      editorShow: !this.state.editorShow,
      id: id,
      title: title,
      content: content
    });
  }

  // 关闭编辑器用，关闭编辑器后将立即获取便签一次
  closeEditor() {
    this.setState({
      editorShow: false
    });
  }
}

```

```

    }, () => {
      this.getListFromGateway();
    });
  }

  componentWillMount() {
    this.getListFromGateway();
    this.periodicGetList();
  }

  periodicGetList() {
    setTimeout(function () {
      this.getListFromGateway();
      this.periodicGetList();
    }
      .bind(this),
      3000
    );
  }

  getListFromGateway() {
    if (!this.state.login) {
      return;
    }
    this.gatewayClient.getAllNotes(notes => {
      this.setState({
        noteList: notes
      });
    });
  }

  saveContractInfo(gatewayAddress, contractAddress, senderAddress, senderPrivateKey) {
    this.gatewayClient = new NoteWallGatewayClient(gatewayAddress, contractAddress, senderAddress,
senderPrivateKey);
    this.setState({login: true});
  }

  // 通知函数，调用toast库，通知用户入链信息
  notify(msg) {
    toast(msg);
  }

  // 异常通知函数，调用toast.error方法，通知用户入链异常信息
  errNotify(msg) {
    toast.error(msg, {
      autoClose: 18000
    });
  }

  render() {
    let list = this.state.noteList;
    if (this.state.login) {
      return (
        <div className='note-wall'>
          <ToastContainer className='toast-notification' />
          {
            Object.keys(list).map(noteId => {
              let note = list[noteId];
              return (
                <Note key={note.id} id={note.id} title={note.title}
                  content={note.content} onClick={this.showEditor} />
              );
            })
          }
        </div>
      );
    }
  }
}

```

```

    }
    { /* 参数作用将在Editor组件时看到 */ }
    <Editor key={this.state.id}
      show={this.state.editorShow}
      id={this.state.id}
      title={this.state.title}
      content={this.state.content}
      onClick={this.showEditor}
      closeEditor={this.closeEditor}
      gatewayClient={this.gatewayClient}
      notify={this.notify}
      errNotify={this.errNotify}
    />
  </div>
);
}
return (
  <div className='note-wall'>
    <LoginForm saveContractInfo={this.saveContractInfo} />
  </div>
);
}
}
}

```

4.最后我们在便签板上还需要增加一个“新增便签”的按钮，用户点击后弹出编辑器。所以完整的便签板代码如下：

```

/**
 * @file notewall class. main component in app.
 * @author BaaS Team
 */
import React, {Component} from 'react';
import LoginForm from './LoginForm.js';
import Note from './Note.js';
import Editor from './Editor.js';
import NoteWallGatewayClient from '../gateway/client.js';
import {ToastContainer, toast} from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';

export default class NoteWall extends Component {
  constructor(props) {
    super(props);
    this.state = {
      editorShow: false,
      login: false,
      noteList: []
    };
    this.showEditor = this.showEditor.bind(this);
    this.closeEditor = this.closeEditor.bind(this);
    this.periodicGetList = this.periodicGetList.bind(this);
    this.createNote = this.createNote.bind(this);
    this.saveContractInfo = this.saveContractInfo.bind(this);
    this.notify = this.notify.bind(this);
    this.errNotify = this.notify.bind(this);
    this.gatewayClient = null;
    this.login = false;
  }

  createNote() {
    if (!this.state.editorShow) {
      this.showEditor(null, '', '');
    }
  }
}

```

```

    }

    showEditor(id, title, content) {
      this.setState({
        editorShow: !this.state.editorShow,
        id: id,
        title: title,
        content: content
      });
    }
  }

  closeEditor() {
    this.setState({
      editorShow: false
    }, () => {
      this.getListFromGateway();
    });
  }

  componentWillMount() {
    this.getListFromGateway();
    this.periodicGetList();
  }

  periodicGetList() {
    setTimeout(function () {
      this.getListFromGateway();
      this.periodicGetList();
    }
    .bind(this),
    3000
  );
}

  getListFromGateway() {
    if (!this.state.login) {
      return;
    }
    this.gatewayClient.getAllNotes(notes => {
      this.setState({
        noteList: notes
      });
    });
  }

  saveContractInfo(gatewayAddress, contractAddress, senderAddress, senderPrivateKey) {
    this.gatewayClient = new NoteWallGatewayClient(gatewayAddress, contractAddress, senderAddress,
senderPrivateKey);
    this.setState({login: true});
  }

  notify(msg) {
    toast(msg);
  }

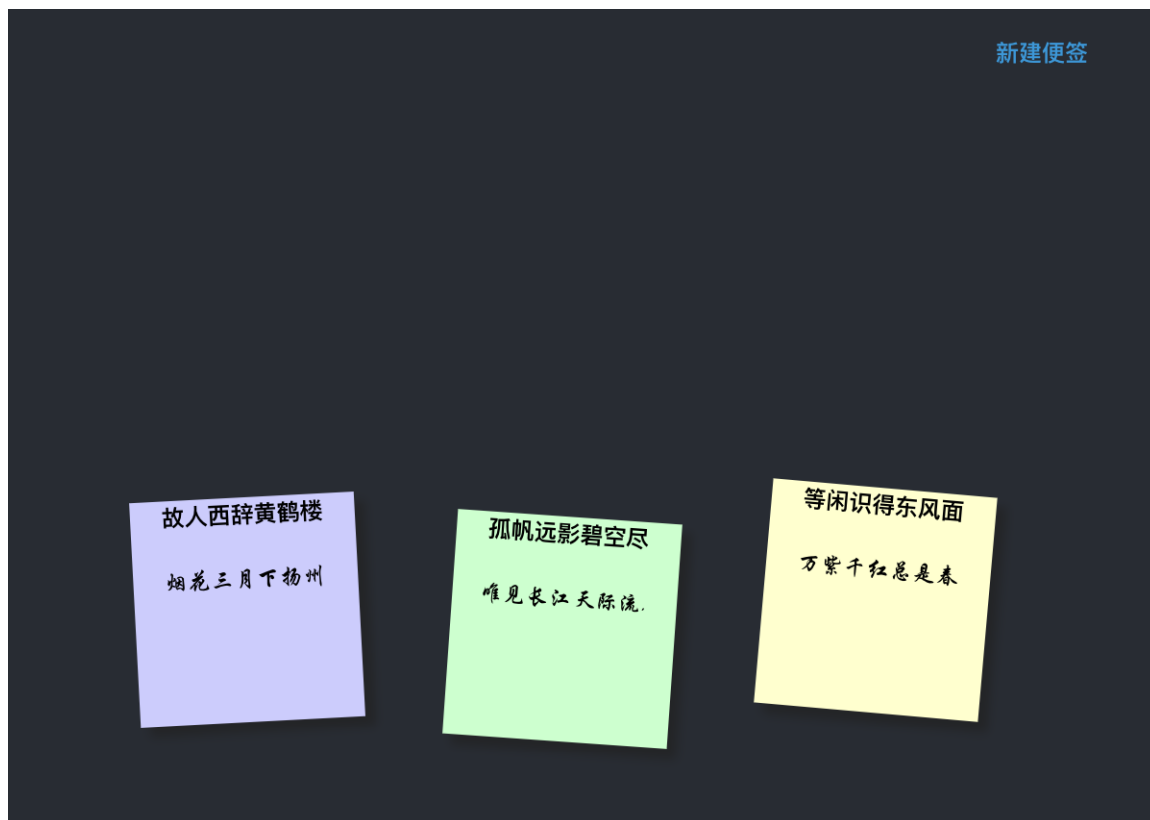
  errNotify(msg) {
    toast.error(msg, {
      autoClose: 18000
    });
  }

  render() {
    let list = this.state.noteList;

```

```
let list = this.state.noteList;
if (this.state.login) {
  return (
    <div className='note-wall'>
      <ToastContainer className='toast-notification' />
      <button className='editor-btn header-btn' onClick={this.createNote}>新建便签</button>
      {
        Object.keys(list).map(noteId => {
          let note = list[noteId];
          return (
            <Note key={note.id} id={note.id} title={note.title}
              content={note.content} onClick={this.showEditor} />
          );
        })
      }
      <Editor key={this.state.id}
        show={this.state.editorShow}
        id={this.state.id}
        title={this.state.title}
        content={this.state.content}
        onClick={this.showEditor}
        closeEditor={this.closeEditor}
        gatewayClient={this.gatewayClient}
        notify={this.notify}
        errNotify={this.errNotify}
      />
    </div>
  );
}
return (
  <div className='note-wall'>
    <LoginForm saveContractInfo={this.saveContractInfo} />
  </div>
);
}
```

便签板完成后，效果图如下：



2、编辑器

编辑器本身需要完成四个功能：

1.如果是已有便签更新，那么需要显示便签标题、便签正文； 2.右上角需要有一个“上链”的按钮，用户更新完成后可以点击这个按钮完成便签存入； 3.当便签上链交易完成后，编辑器退出； 4.当用户放弃编辑时，点击编辑器外即可退出编辑。

编辑Editor.js，代码如下：

```
/**
 * @file editor component
 * @author BaaS Team
 */
import React, {Component} from 'react';
import './Editor.scss';

const titleRef = React.createRef();
const contentRef = React.createRef();

export default class Editor extends Component {
  constructor(props) {
    super(props);
    this.buttonClicked = this.buttonClicked.bind(this);
    this.saveNoteWithGateway = this.saveNoteWithGateway.bind(this);
    this.clickMargin = this.clickMargin.bind(this);
    // 上链中，编辑器不应退出，直到上链结束
    this.marginLock = false;
    this.state = {
      btnMessage: '上链'
    };
    this.gatewayClient = this.props.gatewayClient;
  }

  componentDidMount() {
    titleRef.current.focus();
  }
}
```



```
buttonClicked() {
  if (this.marginLock) {
    return;
  }
  // 修改按钮显示
  this.setState({
    btnMessage: '上链中'
  });
  this.marginLock = true;
  this.saveNoteWithGateway();
}

saveNoteWithGateway() {
  if (this.props.id !== null) {
    this.gatewayClient.updateNote(this.props.id,
      titleRef.current.innerText,
      contentRef.current.innerText,
      (err, receipt) => {
        if (!err) {
          this.props.notify(`已上链. \n 区块高度${receipt.blockNumber}`);
          this.props.closeEditor();
        }
        else {
          this.props.errNotify(`区块链交易遇到问题: ${err}`);
        }
        this.setState({
          btnMessage: '上链'
        });
        this.marginLock = false;
      });
  }
  // 新增便签
  else {
    this.gatewayClient.insertNote(titleRef.current.innerText,
      contentRef.current.innerText,
      (err, receipt) => {
        if (!err) {
          this.props.notify(`已上链. \n 区块高度${receipt.blockNumber}`);
          this.props.closeEditor();
        }
        else {
          this.props.errNotify(`区块链交易遇到问题: ${err}`);
        }
        this.setState({
          btnMessage: '上链'
        });
        this.marginLock = false;
      });
  }
}

// 如果点击编辑器外部，编辑器退出
clickMargin() {
  if (!this.marginLock) {
    this.props.onClick(this.props.title, this.props.content);
  }
}

render() {
  return (
    <div className={this.props.show ? 'editor display-block' : 'editor display-none'}
      onClick={() => this.clickMargin()}>
```

```

    <div id='notebook-paper' onClick={e => {
      e.stopPropagation();
    }}
  >
    <header>
      <h1 id='title-text' contentEditable='true' suppressContentEditableWarning={true}
        ref={titleRef} autoFocus>{this.props.title}</h1>
      <button className='editor-btn header-btn' onClick={this.buttonClicked}>
        {this.state.btnMessage}
      </button>
    </header>
    <div id='content'>
      <div id='content-text' contentEditable='true'
        suppressContentEditableWarning={true} ref={contentRef}>
        {this.props.content}
      </div>
    </div>
  </div>
</div>
);
}
}

```

Editor的样式如下。

```

.editor {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0, 0, 0, 0.6);
}

.editor-main {
  position: fixed;
  background: white;
  width: 80%;
  height: auto;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}

.display-block {
  display: block;
}

.display-none {
  display: none;
}

* {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  -ms-box-sizing: border-box;
  -o-box-sizing: border-box;
  box-sizing: border-box;
}

body {
  background: #f1f1f1;
}

```

```

background: #111111,
font-family:helvetica neue, helvetica, arial, sans-serif;
font-weight:200;
}

#title-text, #content-text {
  &:focus {
    outline: 0;
  }
}

#notebook-paper {
  text-align: left;
  color: #050000;
  width:960px;
  height:500px;
  background: linear-gradient(to bottom,white 29px,#00b0d7 1px);
  margin:50px auto;
  background-size: 100% 30px;
  position:relative;
  padding-top:150px;
  padding-left:160px;
  padding-right:20px;
  overflow:auto;
  border-radius:5px;
  -webkit-box-shadow:3px 3px 3px rgba(0,0,0,.2),0px 0px 6px rgba(0,0,0,.2);
  -moz-box-shadow:3px 3px 3px rgba(0,0,0,.2),0px 0px 6px rgba(0,0,0,.2);
  -ms-box-shadow:3px 3px 3px rgba(0,0,0,.2),0px 0px 6px rgba(0,0,0,.2);
  -o-box-shadow:3px 3px 3px rgba(0,0,0,.2),0px 0px 6px rgba(0,0,0,.2);
  box-shadow:3px 3px 3px rgba(0,0,0,.2),0px 0px 6px rgba(0,0,0,.2);
  &:before {
    content:'';
    display:block;
    position:absolute;
    z-index:1;
    top:0;
    left:140px;
    height:100%;
    width:1px;
    background:#db4034;
  }
  header {
    height:150px;
    width:100%;
    background:white;
    position:absolute;
    top:0;
    left:0;
    h1 {
      font-size:60px;
      line-height:60px;
      padding:127px 20px 0 160px;
    }
  }
  #content {
    margin-top:67px;
    font-size:20px;
    line-height:30px;
  }

  #hipsum {
    margin:0 0 30px 0;
  }

```

```
}

//Colours
$green: #2ecc71;
$red: #e74c3c;
$blue: #3498db;
$yellow: #f1c40f;
$purple: #8e44ad;
$turquoise: #1abc9c;

// Basic Button Style
.editor-btn {
  box-sizing: border-box;
  appearance: none;
  background-color: transparent;
  border: 2px solid $red;
  border-radius: 0.6em;
  color: $red;
  cursor: pointer;
  display: block;
  align-self: center;
  font-size: 3px;
  font-weight: 500;
  line-height: 1;
  margin: 20px;
  padding: 5px 2px;
  text-decoration: none;
  text-align: center;
  text-transform: uppercase;
  font-family: 'Montserrat', sans-serif;
  font-weight: 700;

  &:hover {
    color: #fff;
    outline: 0;
  }
  &:focus {
    outline: 0;
  }
}

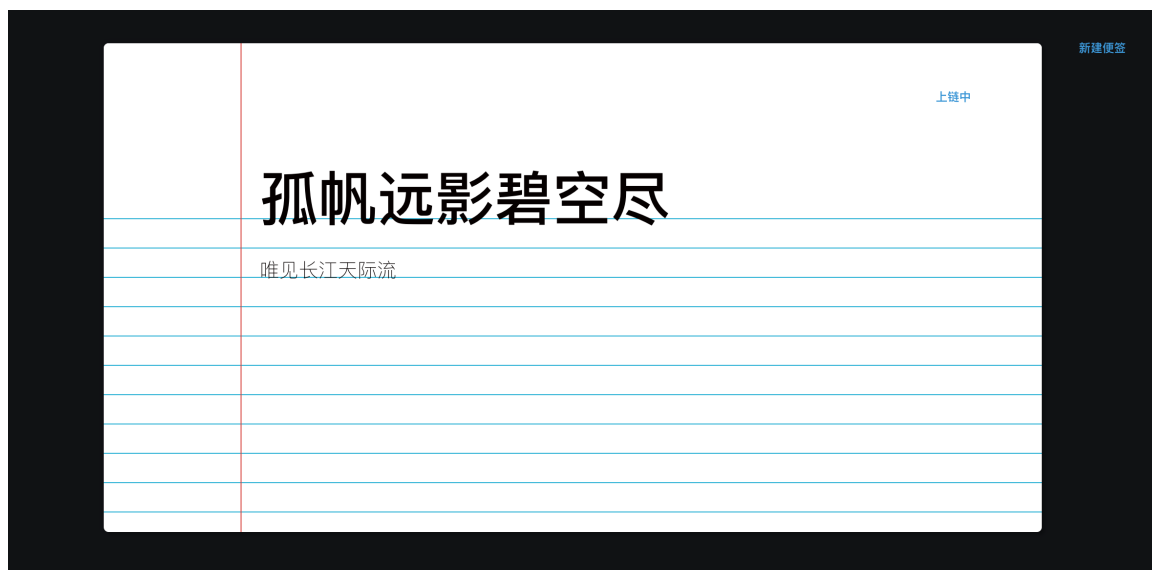
.header-btn {
  border-color: $blue;
  border: 0;
  border-radius: 0;
  color: $blue;
  position: absolute;
  top: 20px;
  right: 20px;
  width: 100px;
  height: 30px;
  overflow: hidden;
  z-index: 1;
  transition: color 150ms ease-in-out;

  &:after {
    content: '';
    position: absolute;
    display: block;
    top: 0;
    left: 50%;
```

```
transform: translateX(-50%);
width: 0;
height: 100%;
background: $blue;
z-index: -1;
transition: width 150ms ease-in-out;
}

&:hover {
  color: #fff;
  &:after {
    width: 110%;
  }
}
}
```

实现完成后效果如下 编辑便签



上链完成，编辑器退出，弹出通知条



到此Dapp应用层开发完毕，我们可以在本地手动测试一下创建便签、更新便签、退出编辑等功能。接下来我们准备将Dapp进行部署。

Dapp部署

Dapp部署是指将我们开发完成的Dapp应用层进行部署。因为Dapp应用层只是一个前端React App，理论上我们只是寻找一个Dapp js的托管服务器。当用户访问Dapp的线上地址时，浏览器获得HTML+CSS+JS，之后所有的交互响应都不再与托管服务器

有关。

这里我们依托百度智能云CCE容器服务来部署我们的托管服务器。

🔗 什么是CCE容器服务

CCE容器服务是基于Kubernetes构建的轻量级分布式容器编排引擎。事实上，超级链BaaS平台服务提供的区块链节点也是托管在CCE集群上的。Kubernetes能够帮助用户快速部署定义好的应用服务，并且能够完全自动化地对应用实例进行健康检查、故障重启自恢复。另外还有应用实例自动水平扩缩容、持续集成构建流、配置管理、服务发现等高级功能。Kubernetes是业界服务托管的趋势。

使用CCE做应用托管，首先需要将应用打包成Docker镜像。因为Kubernetes对应用实例（Pod）的所有定义都是围绕容器实例（Container）来完成的。

🔗 Dapp镜像化

Docker是一种有助于加速开发和部署过程的技术，Docker可以很轻松地将小型独立服务运行环境封装起来。它还有助于消除特定于环境的错误，因为我们可以本地复制生产环境。

Docker环境安装

Docker安装请参考[官方文档](#)，一般的，可以使用系统包管理器一键安装。这里不再赘述。本文使用的Docker版本为1.9.1，是较老版本，请读者确保Docker版本大于1.9.1。

Dockerfile定义

我们在项目根目录下创建一个Dockerfile文件，内容为

```
FROM node:10.15.1
RUN mkdir /usr/src/app
WORKDIR /usr/src/app
ENV PATH /usr/src/app/node_modules/.bin:$PATH
COPY package.json /usr/src/app/package.json
RUN npm install --silent
COPY . /usr/src/app
RUN npm run build
RUN apt-get update && apt-get install -y nginx
RUN cp -a /usr/src/app/build/* /var/www/html/
RUN rm -rf /usr/src
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

再创建一个.dockerignore文件，过滤掉不需要载入到镜像内的文件，内容为

```
node_modules
.git
.gitignore
README.md
```

Docker镜像制作

运行docker build命令即可

```
$ docker build -t notewall-dapp:latest .
```

构建好本地镜像后，我们可以直接在本地运行这个Docker镜像，确认镜像没有问题。

```
$ docker run -p 8080:80 -d notewall-dapp:latest
```

用浏览器访问本地的8080，就能够看到我们的便签板应用啦。

Docker镜像上传

接下来我们将构建好的Dapp镜像上传推送到百度智能云镜像仓库，用来在百度智能云CCE服务上直接部署Dapp。

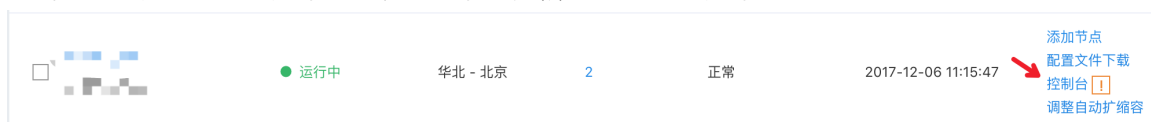
上传前我们需要开通镜像，请参考[百度智能云镜像仓库使用说明](#)

配合本文档，我们提供了一个已经上传好的镜像地址 `hub.baidubce.com/jpaas-public/notewall-dapp:2.1.0`

🔗 如何用百度CCE服务发布Dapp

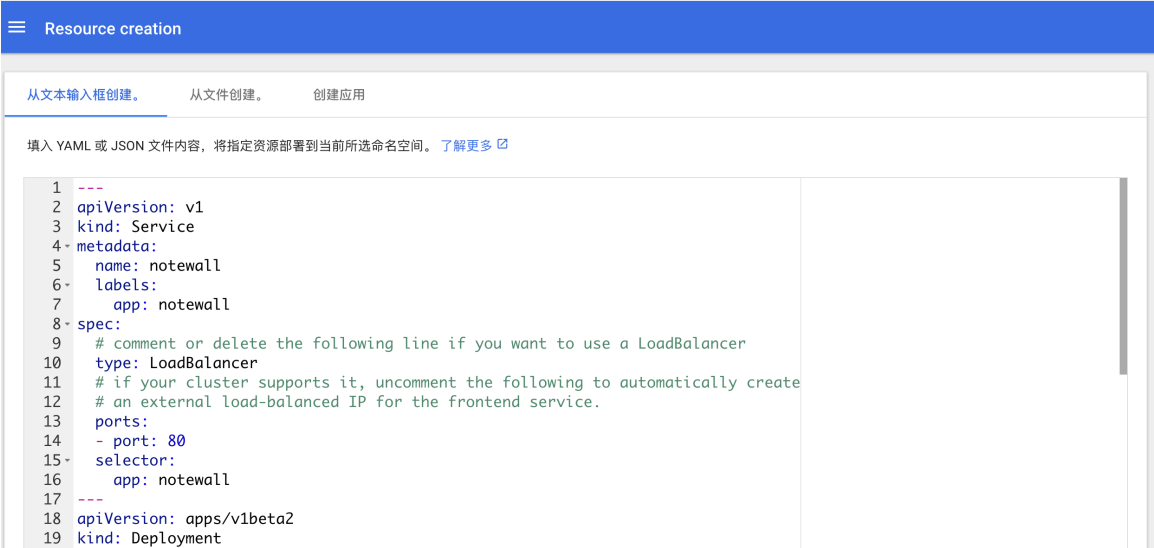
在上传完镜像后，我们开始试用CCE服务。

1. 首先我们需要按照[使用说明](#)创建集群。
2. 我们打开创建好的CCE集群控制台，进入集群列表页，点击集群右侧的“控制台”



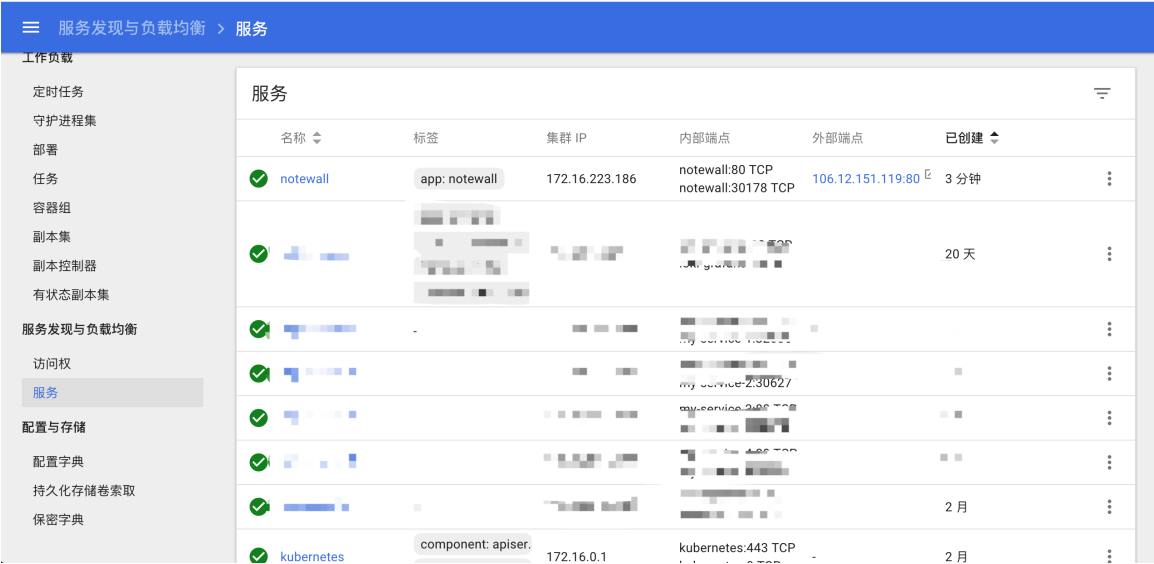
3. 在新打开的控制台中，我们点击右上角“+创建”按钮，在“从文本输入框创建”标签页中，我们输入如下yaml内容：

```
---
apiVersion: v1
kind: Service
metadata:
  name: notewall
  labels:
    app: notewall
spec:
  # comment or delete the following line if you want to use a LoadBalancer
  type: LoadBalancer
  # if your cluster supports it, uncomment the following to automatically create
  # an external load-balanced IP for the frontend service.
  ports:
  - port: 80
  selector:
    app: notewall
---
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: notewall
spec:
  selector:
    matchLabels:
      app: notewall
  replicas: 3
  template:
    metadata:
      labels:
        app: notewall
    spec:
      containers:
      - name: notewall-dapp
        # 这里替换成自行推送的镜像地址，这里填的是默认已有的镜像地址
        image: hub.baidubce.com/jpaas-public/notewall-dapp:2.1.0
        ports:
        - containerPort: 80
```



点击上传后，就会按照上述yaml描述的资源规范在CCE集群创建一个外网可以访问的服务。

4.稍等片刻，我们可打开左侧“服务发现与负载均衡”，“服务”列表内找到notewall服务，点击外部端点提供的IP:Port形式的URL就能访问到便签板Dapp



合约网关地址

合约地址

交易账号

交易密钥

登录

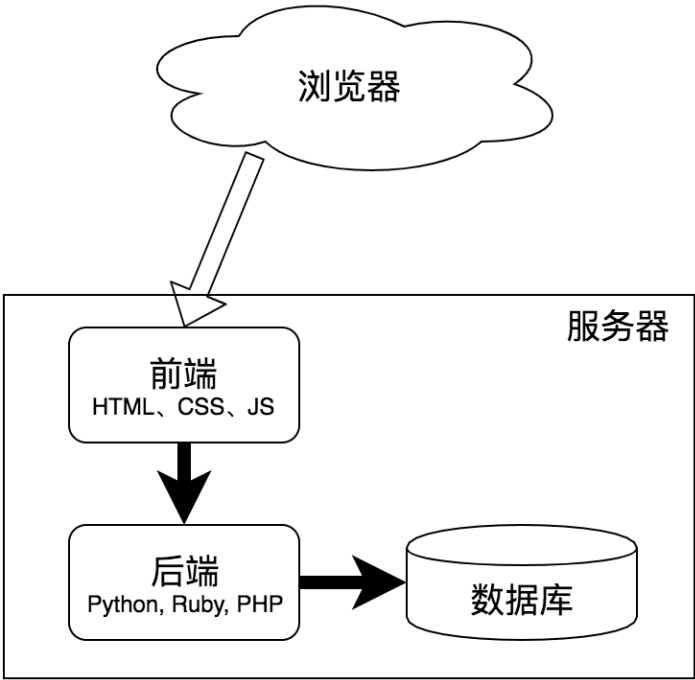
到此便签板Dapp部署完成，外网已经能够访问。

Fabric-Dapp开发指南

Dapp架构及概要设计

什么是Dapp

如果我们熟悉web应用开发，一个经典的web应用架构是B/S结构的



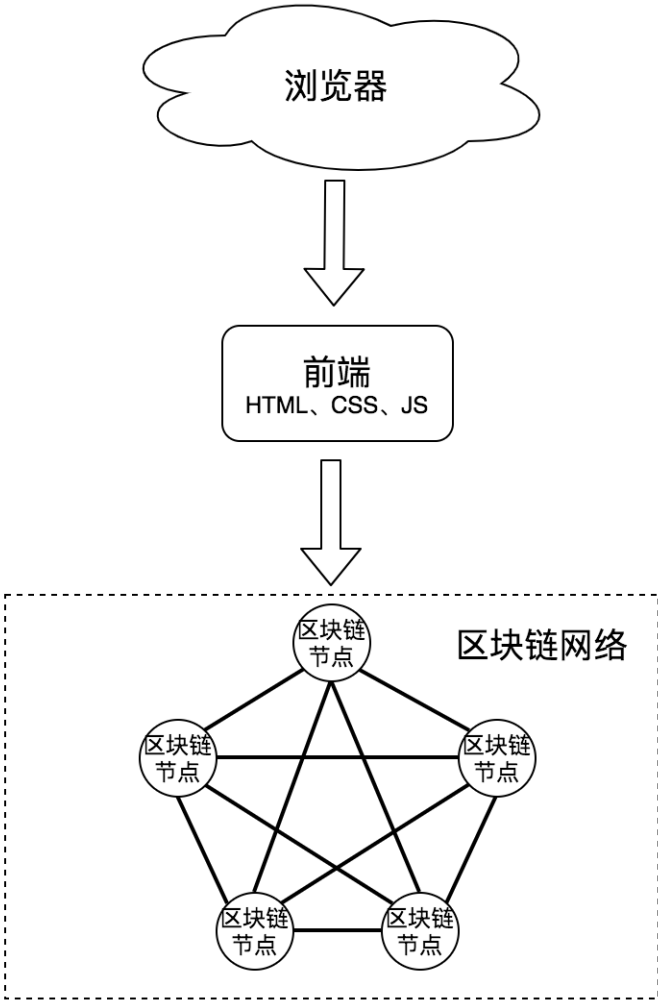
一般来说，我们开发完web应用后会把web服务部署到公网上，现在比较流行的方式是部署到云端VPS。无论在服务端我们用了什么负载均衡、容器化等技术，又无论客户端用户用的是浏览器还是一个API构造器。实质上所有的客户都将通过请求我们部署的服务器暴露的HTTP接口来访问应用。在服务器上，我们写的web应用程序会再做一些数据操作，比如对数据库或缓存做CRUD

操作，最终将结果反馈给客户端。

这个B/S架构如此流行，是因为它适用于绝大多数应用场景。但是也有例外，当用户或者服务提供者认为数据比应用更有价值，又或者服务提供者想提供的是数据服务，并想有效控制用户对数据写的权利。此时基于B/S结构的应用会显得力不从心。和中心化的web应用不同，区块链使用一套开源的网络规范，提供了数据P2P实时共享同步的特性。终端客户可以随时将一个客户端设备接入区块链P2P网络，基于标准的协议，区块链终端将实时同步到最新的数据。在这个数据服务中，不再有中心化的服务端存在，这就是所谓的“对等无中心”特性。

Dapp是基于区块链构建的web应用，区别于B/S结构的传统web服务，Dapp充分吸取了区块链“对等无中心”的特点，采用了Browser/Blockchain的结构。对于服务商而言，只需将数据同步到区块链上，并定义一套详细的数据写规则。而数据的调用读取展示逻辑完全由数据的用户定义。这样既通过区块链保证了数据的可信，同时不同的数据用户也可以按需实现不同的Dapp来挖掘数据的价值。和中心化的web应用不同，区块链使用一套开源的网络规范，提供了数据P2P实时共享同步的特性。终端客户可以随时将一个客户端设备接入区块链P2P网络，基于标准的协议，区块链终端将实时同步到最新的数据。

下图是一个典型的Dapp架构。



🔗 我们要实现什么Dapp

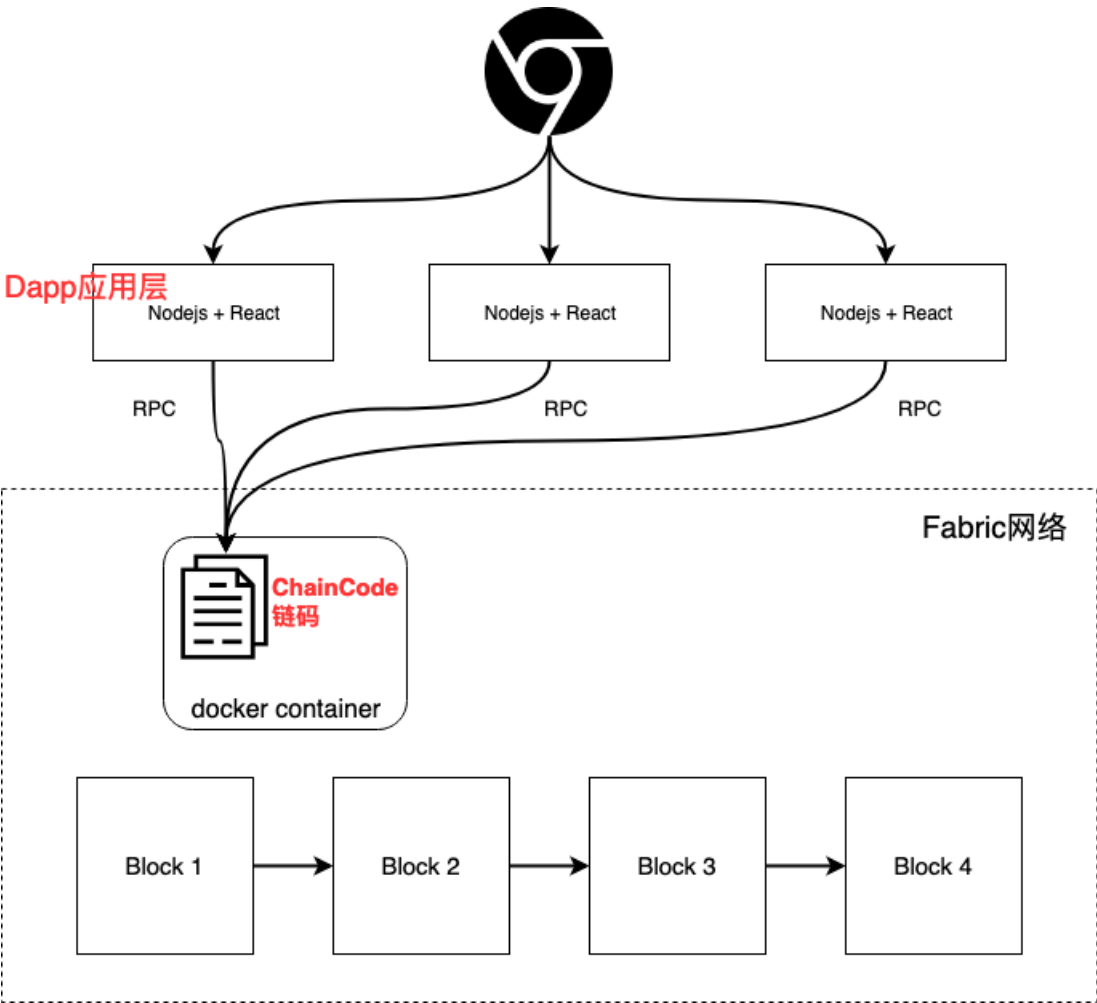
我们要实现的是一个基于区块链的便签板应用。页面效果图如下



用户可以在便签板上添加新的便签或者修改已有的便签，所有在浏览器上打开了这个Dapp的用户都可以同步看到便签的更新，便签被更新的通知。这是一个实现起来相对简单的Dapp，本文档会详细介绍它的实现过程。

🔗 便签板Dapp简介

这个Dapp主要由两部分组成，包括部署在Fabric网络中的智能合约以及依赖智能合约负责业务功能交互的web应用。Fabric中的智能合约被称作链码，本Dapp中链码采用golang开发，而web应用使用react框架+JavaScript+Node.js实现。Dapp组件间的调用关系如图。



- 我们所要开发便签板应用较为简单，核心功能是将用户创建的文本便签存储到区块链上。需要实现的关键用例只有三个
1. 用户新建便签 用户点击新增，在编辑窗口中写入便签题目和便签内容，点击“上链”。应用将便签（便签题目、便签内容）存入，再将编辑窗口关闭。

2. 用户更新便签 用户点击需要更新的便签，在编辑窗口中修改便签题目或便签内容，点击“上链”。应用将便签（便签题目、便签正文）更新，再将编辑窗口关闭。
3. 用户查看所有便签 用户打开应用即可看到所有链上的便签，即便签板。便签分为便签题目和便签内容，在便签板上由于空间有限，每个便签题目和正文只能显示最多20个字符。用户可以点击某一个便签打开编辑更新面板，即可查看全部内容。

链码开发及部署

IDE环境准备

本文采用Go语言开发Fabric链码，Golang IDE请读者自行准备。

链码实现

编写链码的Golang基础语法请读者自行阅读或参考完整源码，这里主要介绍便签板合约的主要实现。

链码对象

Golang实现的链码必须定义合约对象，对象定义为空即可，代码如下

```
type SmartContract struct {  
}
```

便签对象的实现 便签板合约的实现关键就是定义一个便签在区块链上存储的数据格式。我们首先定义一个便签的结构体

```
type Note struct {  
    Id int `json:"id"`  
    Title string `json:"title"`  
    Content string `json:"content"`  
}
```

便签由三部分构成，string类型的标题，string类型的便签正文以及一个64位（也有可能为32位，与CPU位数有关）的有符号整数类型表示的便签ID。

Init接口的实现 在本合约中，并不需要初始化部分便签数据，所以Init接口直接返回成功，具体实现方法如下

```
func (s *SmartContract) Init(APIStub shim.ChaincodeStubInterface) sc.Response {  
    return shim.Success(nil)  
}
```

Invoke接口的实现 Fabric链码的核心接口是invoke，这也是用户实现业务逻辑的地方。在本示例中，主要实现的业务逻辑包括新增便签、更新便签、获取所有的便签。以下就这三个业务进行详细说明。

新增便签的实现 新增便签需要构建一个Note对象，对象中的title和content由用户定义传入，因为本Dapp只是用于示例，所以Id由提交时的时间戳生成，实际生产环境中可以采用其他算法，保证Id的唯一性。Fabric的链上的数据必须为key-value格式，而且key必须为string类型，value必须为byte数组。在本合约中，我们使用Id作为key，所以在调用该方法时，传入的三个参数，Id、title、content，都为字符串类型。在构建Note对象时需要先将string类型的Id转换为int类型，并将Note对象转换为byte数组格式。新增便签的函数如下

```

func (s *SmartContract) insert(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {
if len(args) != 3 {
return shim.Error("Incorrect number of arguments. Expecting 3")
}

noteId, transErr := strconv.Atoi(args[0])
if transErr != nil {
fmt.Printf("Error trans args to get note id: %s", transErr)
return shim.Error("Incorrect type of arguments. Id expecting int")
}

var note = Note {
Id: noteId,
Title: args[1],
Content: args[2],
}

carAsBytes, _ := json.Marshal(note)
err := APIstub.PutState(args[0], carAsBytes)
if err != nil {
return shim.Error(err.Error())
}

return shim.Success(nil)
}

```

更新便签的实现 更新便签与新增便签稍有差别，需先从给定的Note ID中找到note对象，如果对象不存在进行报错，如果对象存在再进行更新。具体代码如下

```

func (s *SmartContract) update(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {
if len(args) != 3 {
return shim.Error("Incorrect number of arguments. Expecting 3")
}

noteAsBytes, _ := APIstub.GetState(args[0])
if noteAsBytes == nil {
return shim.Error("Note not found.")
}
note := Note{}

json.Unmarshal(noteAsBytes, &note)
note.Title = args[1]
note.Content = args[2]

noteAsBytes, _ = json.Marshal(note)
err := APIstub.PutState(args[0], noteAsBytes)
if err != nil {
return shim.Error(err.Error())
}
return shim.Success(nil)
}

```

获取所有的便签 Fabric链码提供的接口中支持遍历功能，通过指定start和endkey，以字典排序查询到相应记录，如果start和endkey都设置为空字符串，查询的是所有记录。与新增便签一样，查询出来的note对象也是以byte数组方式存储的，需要进行解码。之后以Id为key，以note对象为value构建Map，并将Map编码为byte数组返回。具体的代码如下

```

func (s *SmartContract) queryAll(APIstub shim.ChaincodeStubInterface) sc.Response {
resultsIterator, err := APIstub.GetStateByRange("", "")
if err != nil {
return shim.Error(err.Error())
}
defer resultsIterator.Close()

// buffer is a JSON array containing QueryResults
notes := map[int]Note{}

for resultsIterator.HasNext() {
queryResponse, err := resultsIterator.Next()
if err != nil {
return shim.Error(err.Error())
}

var tmpNote Note
noteId, keyTransErr := strconv.Atoi(queryResponse.Key)
if keyTransErr != nil {
fmt.Printf("Error trans note id: %s", keyTransErr)
continue
}
if transErr := json.Unmarshal(queryResponse.Value, &tmpNote); err != nil {
fmt.Printf("Error trans note: %s", transErr)
notes[noteId] = Note{}
} else {
notes[noteId] = tmpNote
}
}

result, _ := json.Marshal(notes)
return shim.Success(result)
}

```

链码完整代码

```

package main

import (
"encoding/json"
"fmt"
"strconv"

"github.com/hyperledger/fabric/core/chaincode/shim"
sc "github.com/hyperledger/fabric/protos/peer"
)

// Define the Smart Contract structure
type SmartContract struct {
}

type Note struct {
Id int `json:"id"`
Title string `json:"title"`
Content string `json:"content"`
}

func (s *SmartContract) Init(APIstub shim.ChaincodeStubInterface) sc.Response {
return shim.Success(nil)
}

```

```

func (s *SmartContract) Invoke(APIstub shim.ChaincodeStubInterface) sc.Response {
    function, args := APIstub.GetFunctionAndParameters()
    if function == "queryAll" {
        return s.queryAll(APIstub)
    } else if function == "insert" {
        return s.insert(APIstub, args)
    } else if function == "update" {
        return s.update(APIstub, args)
    }

    return shim.Error("Invalid Smart Contract function name.")
}

func (s *SmartContract) queryAll(APIstub shim.ChaincodeStubInterface) sc.Response {
    resultsIterator, err := APIstub.GetStateByRange("", "")
    if err != nil {
        return shim.Error(err.Error())
    }
    defer resultsIterator.Close()

    // buffer is a JSON array containing QueryResults
    notes := map[int]Note{}

    for resultsIterator.HasNext() {
        queryResponse, err := resultsIterator.Next()
        if err != nil {
            return shim.Error(err.Error())
        }

        var tmpNote Note
        noteld, keyTransErr := strconv.Atoi(queryResponse.Key)
        if keyTransErr != nil {
            fmt.Printf("Error trans note id: %s", keyTransErr)
            continue
        }
        if transErr := json.Unmarshal(queryResponse.Value, &tmpNote); err != nil {
            fmt.Printf("Error trans note: %s", transErr)
            notes[noteld] = Note{}
        } else {
            notes[noteld] = tmpNote
        }
    }

    result, _ := json.Marshal(notes)
    return shim.Success(result)
}

func (s *SmartContract) insert(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

    if len(args) != 3 {
        return shim.Error("Incorrect number of arguments. Expecting 3")
    }

    noteld, transErr := strconv.Atoi(args[0])
    if transErr != nil {
        fmt.Printf("Error trans args to get note id: %s", transErr)
        return shim.Error("Incorrect type of arguments. Id expecting int")
    }

    var note = Note {
        Id: noteld,

```

```

    Title: args[1],
    Content: args[2],
}

carAsBytes, _ := json.Marshal(note)
err := APIStub.PutState(args[0], carAsBytes)
if err != nil {
    return shim.Error(err.Error())
}

return shim.Success(nil)
}

func (s *SmartContract) update(APIStub shim.ChaincodeStubInterface, args []string) sc.Response {

    if len(args) != 3 {
        return shim.Error("Incorrect number of arguments. Expecting 3")
    }

    noteAsBytes, _ := APIStub.GetState(args[0])
    if noteAsBytes == nil {
        return shim.Error("Note not found.")
    }
    note := Note{}

    json.Unmarshal(noteAsBytes, &note)
    note.Title = args[1]
    note.Content = args[2]

    noteAsBytes, _ = json.Marshal(note)
    err := APIStub.PutState(args[0], noteAsBytes)
    if err != nil {
        return shim.Error(err.Error())
    }
    return shim.Success(nil)
}

// The main function is only relevant in unit test mode. Only included here for completeness.
func main() {
    // Create a new Smart Contract
    err := shim.Start(new(SmartContract))
    if err != nil {
        fmt.Printf("Error creating new Smart Contract: %s", err)
    }
}

```

链码部署

进入Fabric列表页，找到希望安装链码的网络，点击右侧的“通道管理”链接

+ 创建Fabric网络		删除			
Fabric网络名称	Fabric通道名称	状态	创建时间	操作	
 ai-face-test	face-channel	 运行中	2019-06-12 10:13:01	通道管理	
 nml-dapp-test	nml-channel	 运行中	2019-07-05 19:17:19	 通道管理	

进入通道管理页面后，找到希望安装链码的通道，点击右侧的“链码管理”链接

通道名称	参与机构	通道状态	操作
nml-channel	org1	 运行中	 链码管理

进入链码管理页面后点击左侧的“新增链码”按钮

链码列表-nml-channel

+ 新增链码

链码名称	版本	创建时间	部署时间	链码状态	操作
------	----	------	------	------	----

在弹出的链码配置框中填写链码配置信息（单个链码文件请置于文件夹下，将文件夹整体打包上传）

新增链码

* 通道名称

nml-channel

* 通道id

f8c85eeb-d25c-33f1-4381-7d50b8

* 链码名称

map

支持小写字母、数字以及-特殊字符。必须以字母开头，长度1-65

* 版本

1.0

要求两位版本号，格式如1.3

* 链码安装路径

map

相对于/src的包路径

* 初始化参数

init

链码的具体执行参数信息，直接填Args内的参数，如init, a, 100, b, 200

* 背书策略

OR("org1MSP")

链码所关联的背书策略，例如OR("Org1MSP.member","Org2MSP.member")

* 代码文件

请选择文件

文件上传成功

请将代码和完整依赖库打成tar.gz或zip包，
操作命令如(cd /src && tar czf chaincode.tar.gz *)

确认

点击确认后会显示已上传的链码列表

链码列表-nml-channel

<返回通道管理

+ 新增链码

链码名称	版本	创建时间	部署时间	链码状态	操作
map	1.0	2019-07-09 03:51:41	2019-07-09 03:51:41	● 链码已上传	安装

链码安装

已上传的链码可以进行安装，点击链码列表页中对应链码右侧的”安装“按钮，进行链码的安装，安装完成后链码状态会变为安装完成

map	1.1	2019-07-08 19:54:51	2019-07-08 19:54:51	● 安装完成	实例化
-----	-----	---------------------	---------------------	--------	-----

链码实例化

安装成功的链码可以进行实例化，点击对应链码右侧的”实例化“按钮，进行链码的实例化，实例化成功后链码状态变为运行中，至此链码安装完成

链码列表-nml-channel

<返回通道管理

+ 新增链码

链码名称	版本	创建时间	部署时间	链码状态	操作
map	1.0	2019-07-04 19:32:29	2019-07-05 11:35:39	● 运行中	

Dapp应用层开发

🔗 Dapp应用层要做什么

Dapp应用层最主要的是实现用户侧的交互逻辑，包括web页面和页面事件响应。同时不同的操作会带来不同的事件，我们还需要针对页面事件去调用后端链码方法，存入便签、更新便签、读取便签等。

由于是基于React框架做的实现，我们首先要准备一套Dapp应用层的开发环境。

🔗 Dapp应用层开发环境准备

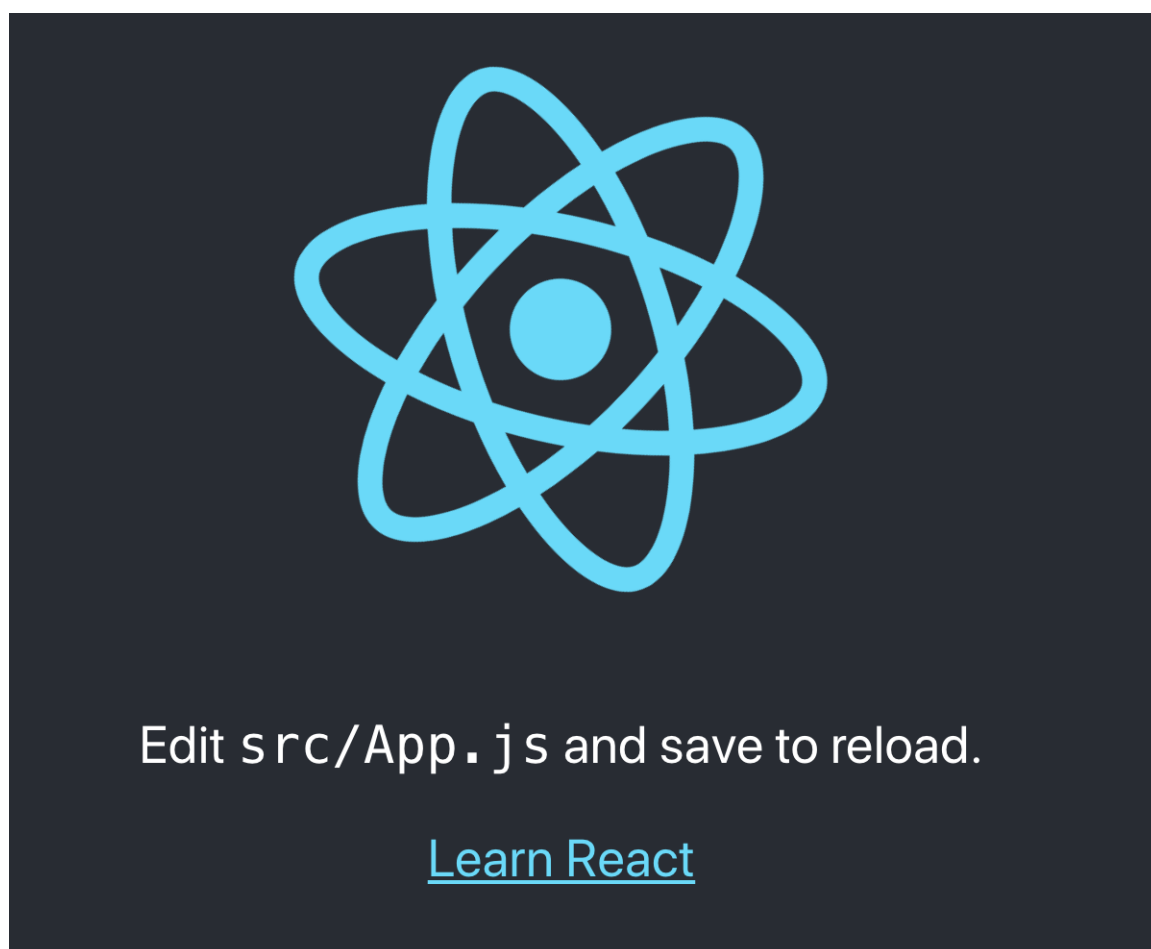
1. 首先需要安装nodejs环境，本文采用的是v10.16.0版本，npm版本为v6.9.0。我们可以使用nvm安装，也可以通过系统自带的包管理安装。具体的下载安装方法这里不再赘述，可参考官方文档：[Node.js下载安装指导](#)
2. 准备好nodejs和npm环境后，我们可以一键创建一个React项目。根据不同的环境，我们有两种命令。

```
$ npx create-react-app notewall-dapp
or
$ create-react-app notewall-dapp
```

3. 执行完之后，我们可以进入notewall-dapp目录，目录里是一个标准的未装任何依赖的项目目录。
4. 试着启动一下

```
$ npm start
```

npm会启动这个react项目，并且将http server监听在本地3000端口。浏览器访问<http://localhost:3000>可以查看结果。



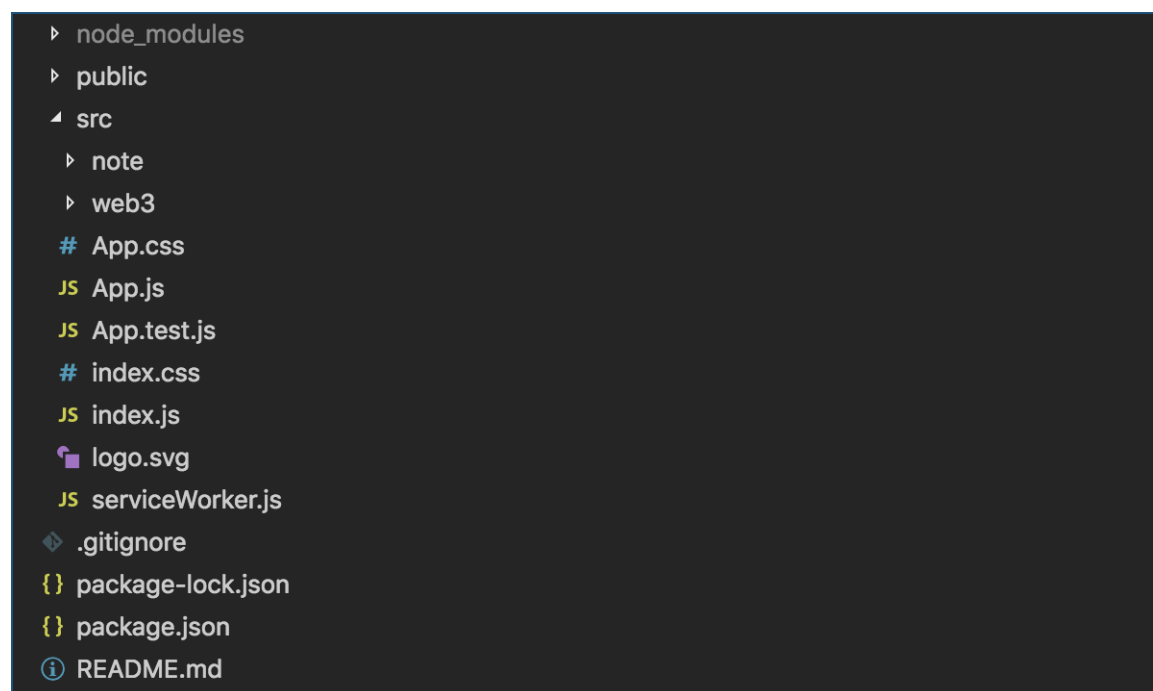
如果我们看到了如下页面显示，则代表项目环境初始化完成，我们可以开始编写应用代码了。

🔗 Dapp应用层开发

如上所述，Dapp应用层主要分为两大部分，一个是页面展示与交互逻辑，另一个是与链码的交互。页面展示与交互逻辑由前端和后端部分完成，与链码的交互由后端部分完成。与以太坊不需要后端服务器不同，Fabric与链码交互时，需要msp、证书等信

息，需要由后端保存，所以Fabric Dapp多了后端服务器的实现。

我们可以规划项目目录，新建两个子目录分别为client和server。server中主要实现与合约的交互逻辑，client目录中主要实现用于前端展示的组件，为了保持实现语言上的一致性，前后端统一用nodejs实现。



合约交互 由于前端页面展示风格千变万化，我们可以先从后端逻辑实现起来。与链码的交互，我们使用Fabric官方提供的Node SDK。结合最初的用例分析，我们需要在三个事件中与链码进行交互。

1. 当用户打开应用时，需要展示所有的便签
2. 当用户新增便签点击上链后，需要将便签内容保存。相对于我们已经实现的链码是InsertNote操作。
3. 当用户更新便签点击上链后，需要将便签内容的更新保存下来。在合约中是UpdateNote操作。 **准备网络配置** 为了实现在外网环境下与Dapp的交互我们在超级链BaaS平台中创建联盟链类型的fabric网络，在网络详情页面可以看到网络的详细配置，在页面右上角可以下载生成的网络证书文件。

nml-dapp-test

删除

下载网络证书文件

↻

fabric网络信息如下：

channel相关信息

名称	OrdererOrg	PeerOrgs	状态
nml-channel	org-orderer	org1	运行中

机构相关信息

名称	类型
org1	peer
org-orderer	orderer

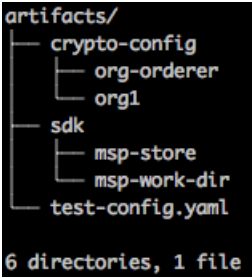
基本信息

orderer数目:	1	peer数目:	1
Explorer地址:	http://106.12.61.2:8080	网络ID:	f8c85eeb-d25c-33f1-4381-7d50b85c4dc2

GRPC地址信息

机构名称	地址列表
org1	106.12.60.243:7051
org-orderer	106.12.60.245:7050

因为生成Fabric Client需要根据网络信息进行配置，需要提前准备好并使服务启动时能够读取。我们在项目中创建artifacts目录用于存放这些信息，Client配置中主要是网络信息及证书。目录如下：



网络配置test_config.yaml详细内容如下：

```
name: "config"

version: 1.0.0

client:
  organization: org1
  logging:
    level: debug
  credentialStore:
    path: artifacts/sdk/msp-store
  cryptoStore:
    path: artifacts/sdk/msp-work-dir
BCCSP:
  security:
    enabled: true
  default:
    provider: "SW"
  hashAlgorithm: "SHA2"
  softVerify: true
  level: 256
tlsCerts:
  systemCertPool: false
client:
  logFile:
```

```

keyfile:
certfile:

organizations:
  org-orderer:
    mspid: org-ordererMSP
    certificateAuthorities:
      - ca.org-orderer
    adminPrivateKey:
      path: artifacts/crypto-config/org-orderer/users/Admin@org-
orderer/msp/keystore/0210362ecc00e5511bd924799dccadc3b015d804c0f0a7633041dfb2471ecc0e_sk
    signedCert:
      path: artifacts/crypto-config/org-orderer/users/Admin@org-orderer/msp/admincerts/Admin@org-orderer-cert.pem
    peers:

  org1:
    mspid: org1MSP
    certificateAuthorities:
      - ca.org1
    adminPrivateKey:
      path: artifacts/crypto-
config/org1/users/Admin@org1/msp/keystore/f4be32c6b1332d9b334fc10805e46f0074dacb1ac4123b23ff1daf9bf2041c
k
    signedCert:
      path: artifacts/crypto-config/org1/users/Admin@org1/msp/admincerts/Admin@org1-cert.pem
    peers:
      - org1-peer0

certificateAuthorities:

  ca.org-orderer:
    url: http://org-orderer-rootca:7053
    httpOptions:
      verify: false
    tlsCACerts:
      path: artifacts/crypto-config/org-orderer/tlsca/tlsca.org-orderer-cert.pem

  ca.org1:
    url: http://org1-rootca:7054
    httpOptions:
      verify: false
    tlsCACerts:
      path: artifacts/crypto-config/org-orderer/tlsca/tlsca.org-orderer-cert.pem

orderers:

  orderer0:
    url: grpc://106.12.60.245:7050
    grpcOptions:
      ssl-target-name-override: orderer0
      allow-insecure: 0
    tlsCACerts:
      path: artifacts/crypto-config/org-orderer/tlsca/tlsca.org-orderer-cert.pem

peers:

  org1-peer0:
    url: grpc://106.12.60.243:7051

```

```
    grpcOptions:
      ssl-target-name-override: org1-peer0
    tlsCACerts:
      path: artifacts/crypto-config/org1/tlsca/tlsca.org1-cert.pem

channels:
  nml-channel:
    orderers:
      - orderer0

  peers:
    org1-peer0:
      endorsingPeer: true
      chaincodeQuery: true
      ledgerQuery: true
      eventSource: true

  policies:
    queryChannelConfig:
      minResponses: 1
      maxTargets: 1
    retryOpts:
      attempts: 5
      initialBackoff: 500ms
      maxBackoff: 5s
      backoffFactor: 2.0
```

后端服务启动 服务端文件的目录结构为：

```
server
├── app.js
├── bin
│   └── www
├── routes
│   ├── api.js
│   ├── ethereum
│   │   └── web3
│   │       └── client.js
│   ├── fabric
│   │   ├── chaincode
│   │   └── client.js
│   └── index.js
├── solidity
│   └── NoteWall.sol
└── views
    ├── error.jade
    ├── index.jade
    └── layout.jade
```

服务启动时会默认展示主页，具体代码如下：

```
/**
 * @file index.js file
 * @author BaaS Team
 */
var express = require('express');
var router = express.Router();

// GET home page
router.get('/', function (req, res, next) {
  res.render('index', {
    title: 'Express'
  });
});

module.exports = router;
```

除此之外，后端服务启动时会构建NoteWallFabricClient对象，并定义路由，具体代码如下：

```
/**
 * @file /api handler
 * @author BaaS Team
 */
const express = require('express');
const NoteWallWeb3Client = require('./ethereum/web3/client.js');
const NoteWallFabricClient = require('./fabric/chaincode/client');
const router = express.Router();

let mode = process.env.BLOCKCHAIN_TYPE.toUpperCase();
let blockchainClient;
if (mode === 'ETHEREUM') {
  let httpProvider = process.env.HTTP_PROVIDER;
  if (!httpProvider) {
    console.error('HTTP_PROVIDER not defined');
    process.exit(1);
  }
  let contractAddress = process.env.CONTRACT_ADDRESS;
  if (!contractAddress) {
    console.error('CONTRACT_ADDRESS not defined');
    process.exit(1);
  }
  let senderAddress = process.env.SENDER_ADDRESS;
  if (!senderAddress) {
    console.error('SENDER_ADDRESS not defined');
    process.exit(1);
  }
  let senderPrivateKey = process.env.SENDER_PRIVATE_KEY;
  if (!senderPrivateKey) {
    console.error('SENDER_PRIVATE_KEY not defined');
    process.exit(1);
  }
  blockchainClient = new NoteWallWeb3Client(httpProvider, contractAddress, senderAddress, senderPrivateKey);
}
else if (mode === 'FABRIC') {
  let networkConfigPath = process.env.FABRIC_NETWORK_CONFIG_PATH;
  if (!networkConfigPath) {
    console.error('FABRIC_NETWORK_CONFIG_PATH not defined');
    process.exit(1);
  }
  let channelName = process.env.FABRIC_CHANNEL_NAME;
  if (!channelName) {
    console.error('FABRIC_CHANNEL_NAME not defined');
```

```

    process.exit(1);
  }
  let chaincodeName = process.env.FABRIC_CHAINCODE_NAME;
  if (!chaincodeName) {
    console.error('FABRIC_CHAINCODE_NAME not defined');
    process.exit(1);
  }
  blockchainClient = new NoteWallFabricClient(networkConfigPath, channelName, chaincodeName);
}
else {
  console.error('BLOCKCHAIN_TYPE not valid. ETHEREUM or FABRIC');
}

router.route('/note').post((req, res, next) => {
  let id = req.body.id;
  let title = req.body.title;
  let content = req.body.content;
  console.log(`${id} ${title} ${content}`);
  // update
  if (id) {
    blockchainClient.updateNote(id, title, content, (err, receipt) => {
      if (!err) {
        res.json({status: 0, message: 'ok'});
      }
      else {
        res.json({status: 1, message: err});
      }
    });
  }
  // insert
  else {
    blockchainClient.insertNote(title, content, (err, receipt) => {
      if (!err) {
        res.json({status: 0, message: 'ok'});
      }
      else {
        res.json({status: 1, message: err});
      }
    });
  }
});

router.get('/notes', (req, res, next) => {
  blockchainClient.getAllNotes(notes => {
    res.json({status: 0, message: notes});
  });
});

module.exports = router;

```

Client构造 我们将这些交互抽象到一个NoteWallFabricClient中，在routes目录中创建fabric/chaincode子目录，并在chaincode目录下创建client.js文件。新建一个NoteWallFabricClient类


```

class NoteWallFabricClient {
  constructor(networkConfigPath, channelName, chaincodeName) {
    let client = fabricClient.loadFromConfig(networkConfigPath);
    client.initCredentialStores();
    let channel = client.getChannel(channelName);
    this.client = client;
    this.channel = channel;
    this.chaincodeName = chaincodeName;
  }
}

module.exports = NoteWallFabricClient;

```

构造函数通过Fabric网络配置文件构建了一个Client，实际使用时需要根据用户自身网络信息配置该文件。然后从client对象中获取channel对象，并将chaincode名称保存。**交易类方法调用** 这里我们先实现创建note交易的方法。

```

/**
 * @param title {string} 便签标题
 * @param content {string} 便签正文
 * @param callback {function} 回调函数
 */
async insertNote(title, content, callback) {
  // note id由当前时间戳生成，用户可以强化此处id生成方法
  let id = + new Date();
  // 用于标识是否发向所有peer的交易提案成功
  let allGood = true;
  let errorMessage = null;

  try {
    let txId = this.client.newTransactionID(true);

    // 构建交易请求
    let request = {
      chaincodeId: this.chaincodeName,
      fcn: 'insert',
      args: [id.toString(), title, content],
      txId: txId
    };

    // 发送交易提案，也即背书
    let results = await this.channel.sendTransactionProposal(request, true);
    let proposalResponses = results[0];
    let proposal = results[1];

    // 处理所有peer的返回
    for (const i in proposalResponses) {
      if (proposalResponses[i] instanceof Error) {
        allGood = false;
        errorMessage = util.format('invoke chaincode proposal resulted in an error :: %s',
          proposalResponses[i].toString());
        console.log(errorMessage);
      } else if (proposalResponses[i].response && proposalResponses[i].response.status === 200) {
        console.info('invoke chaincode proposal was good');
      } else {
        allGood = false;
        errorMessage = util.format('invoke chaincode proposal failed for an unknown reason %j',
          proposalResponses[i]);
        console.log(errorMessage);
      }
    }
  }
}

```

```

// 如果返回都成功，将交易提交给orderer
if (allGood) {
  const promises = [];
  let eventHubs = await this.channel.getChannelEventHubsForOrg();
  eventHubs.forEach(eh => {
    console.log('invokeEventPromise - setting up event:' + eh.getName());
    let invokeEventPromise = new Promise((resolve, reject) => {
      let eventTimeout = setTimeout(() => {
        let message = 'REQUEST_TIMEOUT:' + eh.getPeerAddr();
        console.log(message);
        eh.disconnect();
      }, 3000);
      eh.registerTxEvent(txId.getTransactionID(), (tx, code, blockNum) => {
        console.log('The chaincode invoke chaincode transaction has been committed on peer %s',
          eh.getPeerAddr());
        console.log('Transaction %s has status of %s in block %s', tx, code, blockNum);
        clearTimeout(eventTimeout);

        if (code !== 'VALID') {
          let message = util.format('The invoke chaincode transaction was invalid, code:%s',
            code);
          console.log(message);
          reject(new Error(message));
        } else {
          let message = 'The invoke chaincode transaction was valid.';
          console.log(message);
          resolve(message);
        }
      }, err => {
        clearTimeout(eventTimeout);
        console.log('err:' + err);
        reject(err);
      },
      // the default for 'unregister' is true for transaction listeners
      // so no real need to set here, however for 'disconnect'
      // the default is false as most event hubs are long running
      // in this use case we are using it only once
      {unregister: true, disconnect: true}
    );
    eh.connect();
  });
  promises.push(invokeEventPromise);
});
const ordererRequest = {
  txId: txId,
  proposalResponses: proposalResponses,
  proposal: proposal
};
const sendPromise = this.channel.sendTransaction(ordererRequest);

promises.push(sendPromise);
let results = await Promise.all(promises);
let response = results.pop();
if (response.status === 'SUCCESS') {
  console.log('Successfully sent transaction to the orderer.');
```

// 监控event hubs结果

```

for (let i in results) {

```

```

        let eventHubResult = results[i];
        let eventHub = eventHubs[i];
        console.log('Event results for event hub :%s', eventHub.getPeerAddr());
        if (typeof eventHubResult === 'string') {
            console.log(eventHubResult);
        } else {
            if (!errorMessage) {
                errorMessage = eventHubResult.toString();
            }
            console.log(eventHubResult.toString());
        }
    }
}

} catch (error) {
    console.error('Failed to invoke due to error: ' + error.stack ? error.stack : error);
    errorMessage = error.toString();
}
return callback(errorMessage, null);
}

```

更新note的方法和新建note的方法基本一致，唯一的区别就是将链码函数从insert改为update。非交易类方法调用 非交易类方法主要为view类型的合约方法，可以直接查询而不用生成区块。具体代码如下：

```

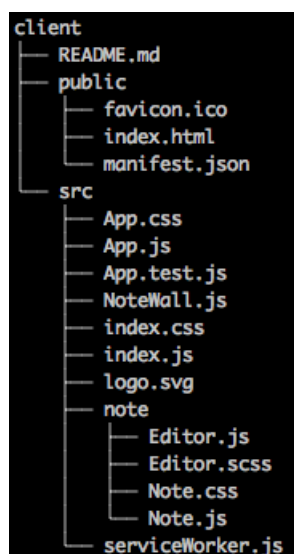
/**
 * callback的参数为notes
 * notes格式为map[notelId] {
 *   id {int}
 *   title {string}
 *   content {string}
 * }
 */
async getAllNotes(callback) {
    let request = {
        chaincodeId: this.chaincodeName,
        fcn: 'queryAll',
        args: ['']
    };
    let queryResponses = await this.channel.queryByChaincode(request, true);
    console.log('Query has completed, checking results:' + (queryResponses));
    return callback(JSON.parse(queryResponses.toString('utf8')));
}

```

到此，与合约交互部分的逻辑都已经开发完成了。NoteWallFabricClient向外暴露三个方法分别是getAllNotes、insertNote、updateNote。页面开发 开发页面前，我们首先将页面交互元素做一个拆分。主要的交互模块有

1. 便签，每一个便签样式类似，但内容不同，一个便签对应合约存储中的一个便签实例；
2. 便签板，上面挂载所有的便签元素，并控制其他模块是否渲染；
3. 编辑器，编辑器用来给用户创建和更新便签时写入内容用。应该允许用户输入标题、内容；

客户端文件目录结构为：



主页 主页比较简单，主要内容是定义了主渲染区“root”，代码如下

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>React App</title>
  <link rel="shortcut icon" href="public/favicon.ico">
  <meta name="viewport" content="width=device-width, initial-scale=1,
  shrink-to-fit=no">
  <meta name="theme-color" content="#000000">
  <!--
    manifest.json provides metadata used when your web app is added to the
    homescreen on Android. See https://developers.google.com/web/fundamentals/web-app-manifest/
  -->
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json">
  <!--
    Notice the use of %PUBLIC_URL% in the tags above.
    It will be replaced with the URL of the `public` folder during the build.
    Only files inside the `public` folder can be referenced from the HTML.

    Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
    work correctly both with client-side routing and a non-root public URL.
    Learn how to configure a non-root public URL by running `npm run build`.
  -->
  <meta http-equiv="X-UA-Compatible" content="IE=Edge">
</head>
<body>
  <noscript>
    You need to enable JavaScript to run this app.
  </noscript>
  <div id="root"></div>
  <!--
    This HTML file is a template.
    If you open it directly in the browser, you will see an empty page.

    You can add webfonts, meta tags, or analytics to this file.
    The build step will place the bundled scripts into the <body> tag.

    To begin the development, run `npm start` or `yarn start`.
    To create a production bundle, use `npm run build` or `yarn build`.
  -->
</body>
</html>
  
```

主页默认渲染便签板，代码如下：

```
/**
 * @file client index
 * @author BaaS Team
 */
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: http://bit.ly/CRA-PWA
serviceWorker.unregister();
```

index样式控制表：

```
body {
  margin: 0;
  padding: 0;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
  "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
  sans-serif;

  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
  monospace;
}
```

index中渲染了APP对象，渲染代码如下：

```
/**
 * @file app component
 * @author BCE BaaS Team
 */

import React, {Component} from 'react';
import './App.css';
import NoteWall from './NoteWall';

export default class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <NoteWall />
        </header>
      </div>
    );
  }
}
```

APP样式控制表：

```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  animation: App-logo-spin infinite 20s linear;
}

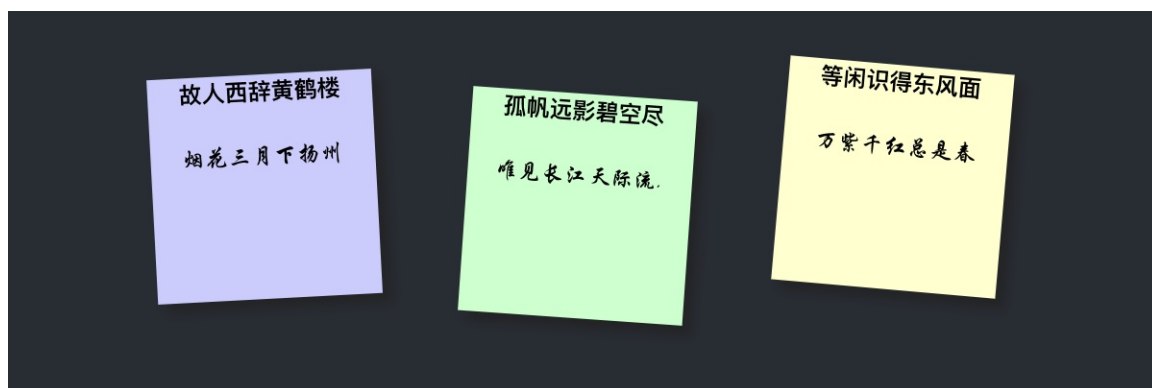
.App-header {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
  font-size: calc(10px + 2vmin);
  color: #fff;
  background-color: #282c34;

  align-items: center;
  justify-content: center;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

便签 便签模块主要是在样式控制上，这里我们将便签设计成不同颜色区分，会有不同倾斜角的正方形卡片。标题文字加粗。如效果图：



Note元素组件定义

```
/**
 * @file note component
 * @author BaaS Team
 */
import React, {Component} from 'react';
import './Note.css';

export default class Note extends Component {
  render() {
    const {id, title, content} = this.props;
    return (
      <div className="note" onClick={() =>
        this.props.onClick(id, title, content)}>
        <div className="note-panel">
          <h2>{this.props.title}</h2>
          <p>{this.props.content}</p>
        </div>
      </div>
    );
  }
}
```

样式控制代码：

```
* {
  margin: 0;
  padding: 0;
}

body {
  margin: 1em;
  font-family: arial, sans-serif;
  font-size: 100%;
  color: #fff;
  background: #666;
}

h2,
p {
  font-size: 100%;
  font-weight: 400;
}

.note-wall,
.note {
  list-style: none;
}

.note-wall {
  overflow: hidden;
  padding: 3em;
}

.note-panel {
  display: block;
  width: 5em;
  height: 5em;
  padding: 2px;
  text-decoration: none;
  color: #000;
  background: #ffc;
```

```
-webkit-box-shadow: 5px 5px 7px rgba(33, 33, 33, .7);
-moz-box-shadow: 5px 5px 7px rgba(33, 33, 33, 1);
    box-shadow: 5px 5px 7px rgba(33, 33, 33, .7);
-webkit-transition: -webkit-transform .15s linear;
-moz-transition:    -moz-transform .15s linear;
-o-transition:      -o-transform .15s linear;
}

.note {
    float: left;
    margin: 1em;
}

.note-panel h2 {
    padding-bottom: 10px;
    font-size: 20%;
    font-weight: 700;
}

.note-panel p {
    font-family: "Reenie Beanie", arial, sans-serif;
    font-size: 20%;
}

.note-panel {
    -webkit-transform: rotate(-6deg);
    -moz-transform: rotate(-6deg);
    -o-transform: rotate(-6deg);
}

.note-wall div:nth-child(even) .note-panel {
    position: relative;
    top: 5px;
    background: #cfc;
    -webkit-transform: rotate(4deg);
    -moz-transform: rotate(4deg);
    -o-transform: rotate(4deg);
}

.note-wall div:nth-child(3n) .note-panel {
    position: relative;
    top: -5px;
    background: #ccf;
    -webkit-transform: rotate(-3deg);
    -moz-transform: rotate(-3deg);
    -o-transform: rotate(-3deg);
}

.note-wall div:nth-child(5n) .note-panel {
    position: relative;
    top: -10px;
    -webkit-transform: rotate(5deg);
    -moz-transform: rotate(5deg);
    -o-transform: rotate(5deg);
}

.note-panel:focus {
    position: relative;
    z-index: 5;
    -webkit-box-shadow: 10px 10px 7px rgba(0, 0, 0, .7);
    -moz-box-shadow: 10px 10px 7px rgba(0, 0, 0, .7);
    box-shadow: 10px 10px 7px rgba(0, 0, 0, .7);
}
```



```

    -webkit-transform: scale(1.25);
    -moz-transform: scale(1.25);
    -o-transform: scale(1.25);
  }

```

便签板 便签板组件负责挂载其余组件，并负责通过子组件的事件回调完成子组件的通信。主要有以下核心控制逻辑：

```

/**
 * @file notewall class. main component in app.
 * @author BaaS Team
 */
import React, {Component} from 'react';
import Note from './note/Note.js';
import Editor from './note/Editor.js';

export default class NoteWall extends Component {
  constructor(props) {
    super(props);
    this.state = {
      editorShow: false,
      noteList: []
    };
    this.showEditor = this.showEditor.bind(this);
    this.closeEditor = this.closeEditor.bind(this);
    this.periodicGetList = this.periodicGetList.bind(this);
  }

  showEditor(id, title, content) {
    this.setState({
      editorShow: !this.state.editorShow,
      id: id,
      title: title,
      content: content
    });
  }

  closeEditor() {
    this.setState({
      editorShow: false
    }, () => {
      this.getList();
    });
  }

  // Fetch the list on first mount
  componentWillMount() {
    this.getList();
    this.periodicGetList();
  }

  periodicGetList() {
    setTimeout(
      function () {
        this.getList();
        this.periodicGetList();
      }
      .bind(this),
      3000
    );
  }
}

```

// Retrieves the list of items from the Firebase app

```
// Retrieves the list of items from the Express app
getList() {
  fetch('/api/notes')
    .then(res => res.json())
    .then(res => this.setState({noteList: res.message}));
}

render() {
  let list = this.state.noteList;
  return (
    <div className="note-wall">
      <button className="btn header-btn" onClick={e => {
        this.showEditor(null, '', '');
      }}>新建</button>
      {
        Object.keys(list).map(notelId => {
          let note = list[notelId];
          return (
            <Note key={note.id} id={note.id} title={note.title}
              content={note.content} onClick={this.showEditor} />
          );
        })
      }

      <Editor show={this.state.editorShow} id={this.state.id} title={this.state.title}
        content={this.state.content} onClick={this.showEditor} closeEditor={this.closeEditor} />
    </div>
  );
}
```

便签板完成后，效果图如下：



编辑器 编辑器本身需要完成四个功能：

1. 如果是已有便签更新，那么需要显示便签标题、便签正文；

2. 右上角需要有一个“上链”的按钮，用户更新完成后可以点击这个按钮完成便签存入；
3. 当便签上链交易完成后，编辑器退出；
4. 当用户放弃编辑时，点击编辑器外即可退出编辑。

编辑Editor.js，代码如下

```
/**
 * @file editor component
 * @author BaaS Team
 */
import React, {Component} from 'react';
import './Editor.scss';

const titleRef = React.createRef();
const contentRef = React.createRef();

export default class Editor extends Component {
  constructor(props) {
    super(props);
    this.saveNote = this.saveNote.bind(this);
  }

  saveNote() {
    fetch('/api/note', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({
        id: this.props.id,
        title: titleRef.current.innerText,
        content: contentRef.current.innerText
      })
    })
    .then(res => {
      res.json();
    })
    .then(list => this.setState({
      noteList: list
    }), () => this.props.closeEditor());
  }

  render() {
    return (
      <div className={this.props.show ? 'editor display-block' : 'editor display-none'}
        onClick={() => this.props.onClick(this.props.title, this.props.content)}>
        <div id="notebook-paper" onClick={e => {
          e.stopPropagation();
        }}>
          <header>
            <h1 id="title-text" contentEditable="true" suppressContentEditableWarning={true}
              ref={titleRef}>{this.props.title}</h1>
            <button className="btn header-btn" onClick={this.saveNote}>保存</button>
          </header>
          <div id="content">
            <div id="content-text" contentEditable="true"
              suppressContentEditableWarning={true} ref={contentRef}>
              {this.props.content}
            </div>
          </div>
        </div>
      </div>
    );
  }
}
```

Editor的样式表:

```
.editor {
  position: fixed;
  top: 0;
  left: 0;
  width:100%;
  height: 100%;
  background: rgba(0, 0, 0, 0.6);
}

.editor-main {
  position:fixed;
  background: white;
  width: 80%;
  height: auto;
  top:50%;
  left:50%;
  transform: translate(-50%,-50%);
}

.display-block {
  display: block;
}

.display-none {
  display: none;
}

* {
  -webkit-box-sizing:border-box;
  -moz-box-sizing:border-box;
  -ms-box-sizing:border-box;
  -o-box-sizing:border-box;
  box-sizing:border-box;
}

body {
  background: #f1f1f1;
  font-family:helvetica neue, helvetica, arial, sans-serif;
  font-weight:200;
}

#### notebook-paper {
  text-align: left;
  color: #050000;
  width:960px;
  height:500px;
  background: linear-gradient(to bottom,white 29px,#00b0d7 1px);
  margin:50px auto;
  background-size: 100% 30px;
  position:relative;
  padding-top:150px;
  padding-left:160px;
  padding-right:20px;
  overflow:hidden;
  border-radius:5px;
  -webkit-box-shadow:3px 3px 3px rgba(0,0,0,.2),0px 0px 6px rgba(0,0,0,.2);
  -moz-box-shadow:3px 3px 3px rgba(0,0,0,.2),0px 0px 6px rgba(0,0,0,.2);
  -ms-box-shadow:3px 3px 3px rgba(0,0,0,.2),0px 0px 6px rgba(0,0,0,.2);
  -o-box-shadow:3px 3px 3px rgba(0,0,0,.2),0px 0px 6px rgba(0,0,0,.2);
  box-shadow:3px 3px 3px rgba(0,0,0,.2),0px 0px 6px rgba(0,0,0,.2);
}
```

```
&:before {
  content:'';
  display:block;
  position:absolute;
  z-index:1;
  top:0;
  left:140px;
  height:100%;
  width:1px;
  background:#db4034;
}
header {
  height:150px;
  width:100%;
  background:white;
  position:absolute;
  top:0;
  left:0;
  h1 {
    font-size:60px;
    line-height:60px;
    padding:127px 20px 0 160px;
  }
}
#content {
  margin-top:67px;
  font-size:20px;
  line-height:30px;
}

#hipsum {
  margin:0 0 30px 0;
}
}

//Colours
$green: #2ecc71;
$red: #e74c3c;
$blue: #3498db;
$yellow: #f1c40f;
$purple: #8e44ad;
$turquoise: #1abc9c;

// Basic Button Style
.btn {
  box-sizing: border-box;
  appearance: none;
  background-color: transparent;
  border: 2px solid $red;
  border-radius: 0.6em;
  color: $red;
  cursor: pointer;
  display: block;
  align-self: center;
  font-size: 3px;
  font-weight: 500;
  line-height: 1;
  margin: 20px;
  padding: 5px 2px;
  text-decoration: none;
  text-align: center;
  text-transform: uppercase;
}
```

```
font-family: 'Montserrat', sans-serif;
font-weight: 700;

&:hover,
&:focus {
  color: #fff;
  outline: 0;
}
}

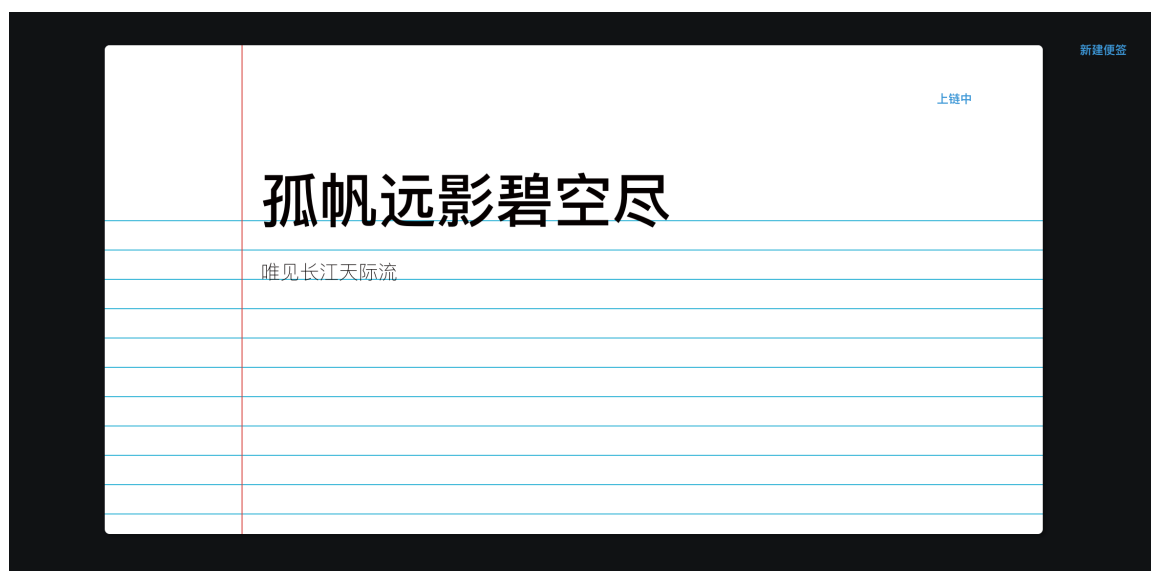
.header-btn {
  border-color: $purple;
  // border: 0;
  border-radius: 0;
  color: $purple;
  position: absolute;
  top: 20px;
  right: 20px;
  width: 50px;
  height: 30px;
  overflow: hidden;
  z-index: 1;
  transition: color 150ms ease-in-out;

  &:after {
    content: "";
    position: absolute;
    display: block;
    top: 0;
    left: 50%;
    transform: translateX(-50%);
    width: 0;
    height: 100%;
    background: $purple;
    z-index: -1;
    transition: width 150ms ease-in-out;
  }

  &:hover {
    color: #fff;
    &:after {
      width: 110%;
    }
  }
}
```

实现完成后效果如下

编辑便签



到此Dapp应用层开发完毕，我们可以在本地手动测试一下创建便签、更新便签、退出编辑等功能。接下来我们准备将Dapp进行部署。

Dapp部署

Dapp部署是指将我们开发完成的Dapp应用层进行部署。因为Dapp应用层只是一个前端React App，理论上我们只是寻找一个Dapp js的托管服务器。当用户访问Dapp的线上地址时，浏览器获得HTML+CSS+JS，之后所有的交互响应都不再与托管服务器有关。

这里我们依托百度云CCE容器服务来部署我们的托管服务器。

什么是CCE容器服务

CCE容器服务是基于Kubernetes构建的轻量级分布式容器编排引擎。事实上，超级链BaaS平台服务提供的区块链节点也是托管在CCE集群上的。Kubernetes能够帮助用户快速部署定义好的应用服务，并且能够完全自动化地对应用实例进行健康检查、故障重启自恢复。另外还有应用实例自动水平扩缩容、持续集成构建流、配置管理、服务发现等高级功能。Kubernetes是业界服务托管的趋势。

使用CCE做应用托管，首先需要将应用打包成Docker镜像。因为Kubernetes对应用实例（Pod）的所有定义都是围绕容器实例（Container）来完成的。

Dapp镜像化

Docker是一种有助于加速开发和部署过程的技术，Docker可以很轻松地将小型独立服务运行环境封装起来。它还有助于消除特定于环境的错误，因为我们可以在本地复制生产环境。**Docker环境安装** Docker安装请参考[官方文档](#)，一般的，可以使用系统包管理器一键安装。这里不再赘述。本文使用的Docker版本为18.09.02，请读者确保Docker版本至少大于1.9.1。**Dockerfile定义** 我们在项目根目录下创建一个Dockerfile文件，内容为


```

FROM node:10.16.0
RUN mkdir /usr/src/appdock
WORKDIR /usr/src/app
ENV PATH /usr/src/app/node_modules/.bin:$PATH
COPY package.json /usr/src/app/package.json
RUN npm install --silent
COPY . /usr/src/app
ARG BLOCKCHAIN_TYPE
ENV BLOCKCHAIN_TYPE='fabric'
ARG FABRIC_NETWORK_CONFIG_PATH
ENV FABRIC_NETWORK_CONFIG_PATH='artifacts/test-config.yaml'
ARG FABRIC_CHANNEL_NAME
ENV FABRIC_CHANNEL_NAME='nml-channel'
ARG FABRIC_CHAINCODE_NAME
ENV FABRIC_CHAINCODE_NAME='map'
RUN npm run build
RUN apt-get update && apt-get install -y nginx
RUN cp -a /usr/src/app/dist/* /var/www/html/
RUN echo "daemon off;">>/etc/nginx/nginx.conf
RUN chmod a+x ./entrypoint.sh
RUN sed -i '53i location ^~ /api/ { proxy_pass http://127.0.0.1:4000; }' /etc/nginx/sites-enabled/default
EXPOSE 80
ENTRYPOINT ["../entrypoint.sh"]

//enrtpoint.sh
##### !/bin/bash
nohup npm run server &
nginx

```

再创建一个.dockerignore文件，过滤掉不需要载入到镜像内的文件，内容为

```

node_modules
.git
.gitignore
README.md

```

Docker镜像制作 运行docker build命令即可

```
$ docker build -t notewall-dapp:latest .
```

构建好本地镜像后，我们可以直接在本地运行这个Docker镜像，确认镜像没有问题。

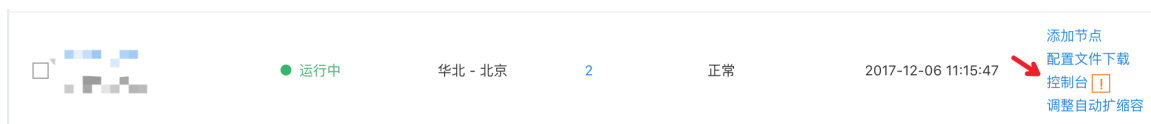
```
$ docker run -p 8080:80 -d notewall-dapp:latest
```

用浏览器访问本地的8080，就能够看到我们的便签板应用啦。 **Docker镜像上传** 接下来我们将构建好的Dapp镜像上传推送到百度云镜像仓库，用来在百度云CCE服务上直接部署Dapp。上传前我们需要开通镜像，请参考[百度云镜像仓库使用说明](#) 配合本文档，我们提供了一个已经上传好的镜像地址 `hub.baidubce.com/jpaas-public/notewall-dapp:latest`

🔗 如何用百度CCE服务发布Dapp

在上传完镜像后，我们开始试用CCE服务。

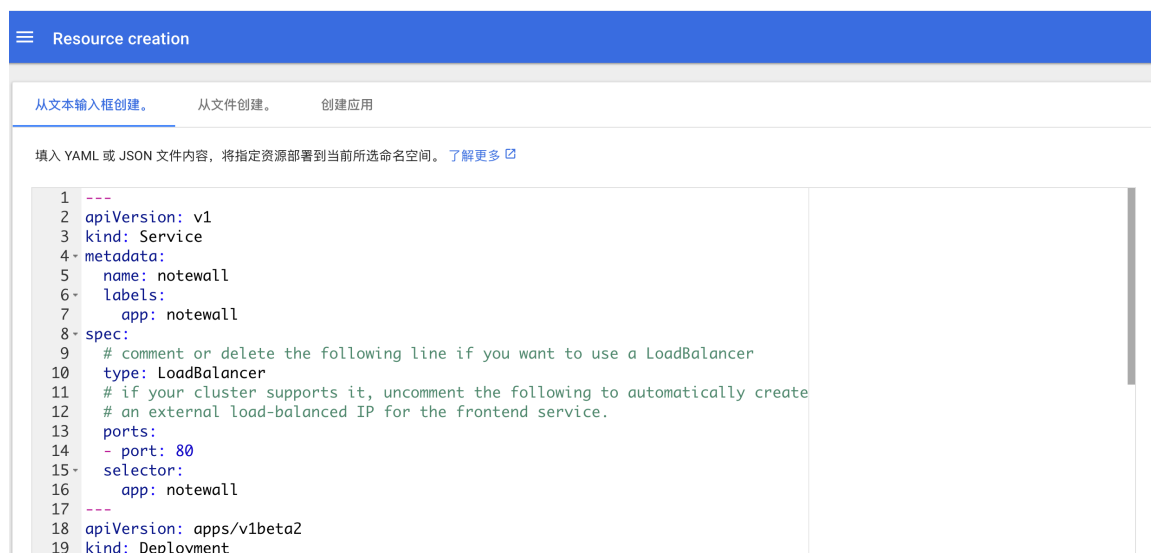
1. 首先我们需要按照[使用说明](#)创建集群。
2. 我们打开创建好的CCE集群控制台，进入集群列表页，点击集群右侧的“控制台”



3. 在新打开的控制台中，我们点击右上角“+创建”按钮，在“从文本输入框创建”标签页中，我们输入如下yaml内容

```
---
apiVersion: v1
kind: Service
metadata:
  name: notewall
  labels:
    app: notewall
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: notewall
---
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: notewall
spec:
  selector:
    matchLabels:
      app: notewall
  replicas: 3
  template:
    metadata:
      labels:
        app: notewall
    spec:
      containers:
      - name: notewall-dapp
        image: hub.baidubce.com/jpaas-public/notewall-dapp:latest
        ports:
        - containerPort: 80
```

如图所示



点击上传后，就会按照上述yaml描述的资源规范在CCE集群创建一个外网可以访问的服务。

- 稍等片刻，我们可打开左侧“服务发现与负载均衡”，“服务”列表内找到notewall服务，点击外部端点提供的IP:Port形式的URL就能访问到便签板Dapp

到此我们的便签板Dapp部署完成，外网已经能够访问。

视频指南

操作指导

🔗 Quorum联盟链操作演示

- 介绍百度智能云Quorum联盟链和使用联盟链配置便签的演示操作。



API参考

简介

概述

🔗 概述

超级链BaaS平台（Xuper BaaS）是企业或开发者快速搭建区块链网络的BaaS（Blockchain as a Service）平台，目前完美兼容Ethereum、Fabric、百度自研超级链三种主流框架。

如果您是初次调用百度智能云产品的API，可以观看[API入门视频指南](#)，快速掌握调用API的方法。

接口使用规范

🔗 API认证机制

所有API的安全认证一律采用Access Key与请求签名机制。Access Key由Access Key ID和Secret Access Key组成，均为字符串。对于每个HTTP请求，使用下面所描述的算法生成一个认证字符串。提交认证字符串放在Authorization头域里。服务端根据生成算法验证认证字符串的正确性。认证字符串的格式为**bce-auth-v{version}/{accessKeyId}/{timestamp}/{expirationPeriodInSeconds}/{signedHeaders}/{signature}**。

- version是正整数。
- timestamp是生成签名时的UTC时间。
- expirationPeriodInSeconds表示签名有效期限。
- signedHeaders是签名算法中涉及到的头域列表。头域名之间用分号（;）分隔，如host;x-bce-date。列表按照字典序排列。（本API签名仅使用host和x-bce-date两个header）
- signature是256位签名的十六进制表示，由64个小写字母组成。

当百度智能云接收到用户的请求后，系统将使用相同的SK和同样的认证机制生成认证字符串，并与用户请求中包含的认证字符串进行比对。如果认证字符串相同，系统认为用户拥有指定的操作权限，并执行相关操作；如果认证字符串不同，系统将忽略

该操作并返回错误码。

鉴权认证机制的详细内容请参见[鉴权认证](#)。

🔗 接口使用规范

规范化字符串

通常一个字符串中可以包含任何Unicode字符。在编程中这种灵活性会带来不少困扰。因此引入“规范字符串”的概念。一个规范字符串只包含百分号编码字符以及URI（Uniform Resource Identifier）非保留字符（Unreserved Characters）。RFC 3986规定URI非保留字符包括以下字符：字母（A-Z，a-z）、数字（0-9）、连字号（-）、点号（.）、下划线（_）、波浪线（~）。将任意一个字符串转换为规范字符串的方式是：

- 将字符串转换成UTF-8编码的字节流。
- 保留所有URI非保留字符原样不变。
- 对其余字节做一次RFC 3986中规定的百分号编码（Percent-Encoding），即一个%后面跟着两个表示该字节值的十六进制字母。字母一律采用大写形式。示例：原字符串：this is an example for 测试, 对应的规范字符串：this%20is%20an%20example%20for%20%E6%B5%8B%E8%AF%95。

编码规范

- 可解析内容，所有request/response body内容目前均使用UTF-8编码，后续会支持更多encoding类型。
- 在请求时，需要对以下做UrlEncode：
 - Objectname，其中，Resource做UrlEncode的时候需要忽略“/”。
 - Querystring的Value。
 - x-bce-copy-source（忽略“/”）。
 - 自定义Meta:Meta Value只支持可见的ASCII字符，如果需要其它的字符，推荐使用UrlEncode处理。

日期与时间

日期与时间的表示有多种方式。为统一起见，除非是约定俗成或者有相应规范的，凡是HTTP标准中规定的表示日期和时间字段用GMT，其他日期时间表示的地方一律采用UTC时间，遵循ISO 8601，并做以下约束：

- 表示日期一律采用YYYY-MM-DD方式，例如2014-06-01表示2014年6月1日。
- 表示时间一律采用hh:mm:ss方式，并在最后加一个大写字母Z表示UTC时间。例如23:00:10Z表示UTC时间23点0分10秒。
- 凡涉及日期和时间合并表示时，在两者中间加大写字母T，例如2014-06-01T23:00:10Z表示UTC时间2014年6月1日23点0分10秒。

请求响应格式标准

- HTTP请求，Querystring中参数的Key，为首字母小写的驼峰方式。如uploadId，partNumber等。
- 所有用户自定义Meta，以x-bce-meta-*的形式放Header中，自定义Meta总大小不得超过2K。x-bce-meta-*的Key会被Server端统一按照小写进行处理。

例如：用户使用PutObject接口上传了x-bce-meta-DeMo:value，Server端会统一按照小写x-bce-meta-demo:value进行处理，用户在使用GetObject接口时，Server端的返回值为x-bce-meta-demo:value。

- 除RFC2616规定的标准Header外，其他Header以x-bce-*的形式定义。
- 所有JSON中，Key均为首字母小写的驼峰方式。
- 每个请求响应中均带有x-bce-request-id和x-bce-debug-id这两个Header。

- Header中Date、Content-MD5、Content-Type、Content-Length等相关字段遵守RFC 2616约束。
- 依据HTTP协议的规定，Content-MD5既要MD5也要进行Base64编码，其计算方法如下：

Content-MD5 = "Content-MD5"
":" md5-digest
md5-digest = <base64 of 128 bit MD5 digest as per RFC 1864>

服务域名

🔗 服务域名

XUPERBAAS API的服务域名为：

区域	服务端点Endpoint	协议
北京	bbe.bj.baidubce.com	HTTP and HTTPS
广州	bbe.gz.baidubce.com	HTTP and HTTPS
苏州	bbe.su.baidubce.com	HTTP and HTTPS
保定	bbe.bd.baidubce.com	HTTP and HTTPS

说明：XUPERBAAS API支持HTTP和HTTPS两种调用方式。为了提升数据的安全性，建议通过HTTPS调用。

公共请求头

🔗 公共请求头

头域	说明	是否必须
Authorization	包含Access Key与请求签名。	必须
x-bce-date	该请求创建的时间，表示日期一律采用YYYY-MM-DD方式，例如2014-06-01表示2014年6月1日。如果用户使用了标准的Date域，该头域可以不填。当两者同时存在时，以x-bce-date为准。	可选
x-bce-content-sha256	表示内容部分的SHA256签名的十六进制字符串，其中内容指HTTP Request Payload Body，即Content部分在被HTTP encode之前的原始数据。	可选
x-bce-if-match	同If-Match语义。	可选
x-bce-if-none-match	同If-None-Match语义。	可选

公共响应头

🔗 公共响应头

头域	说明
Content-Length	RFC2616中定义的HTTP请求内容的类型。
x-bce-request-id	对应请求的requestId。

错误码

🔗 错误码

错误信息除了HTTP状态码以外，应同时在HTTP body中包含下表的参数，内容如下：

参数名	类型	说明
requestId	String	导致该错误的requestId
code	String	表示具体错误类型
message	String	有关该错误的详细说明

例如：

```
{
  "code": "IllegalRequestUrl",
  "message": "The requested url belongs to domain which is not under acceleration",
  "requestId": " 81d0b05f-5ad4-1f22-8068-d5c9de60a1d7"
}
```

公共错误码

错误码	消息	HTTP状态码	语义
AccessDenied	Access denied.	403 Forbidden	无权限访问对应的资源。
InappropriateJSON	The JSON you provided was well-formed and valid, but not appropriate for this operation.	400 Bad Request	请求中的JSON格式正确，但语义上不符合要求。如缺少某个必需项，或者值类型不匹配等。出于兼容性考虑，对于所有无法识别的项应直接忽略，不应该返回这个错误。
InternalError	We encountered an internal error. Please try again.	500 Internal Server Error	所有未定义的其他错误。在有明确对应的其他类型的错误时（包括通用的和服务自定义的）不应该使用。
InvalidAccessKeyId	The Access Key ID you provided does not exist in our records.	403 Forbidden	Access Key ID不存在。
InvalidHTTPAuthHeader	The HTTP authorization header is invalid. Consult the service documentation for details.	400 Bad Request	Authorization头域格式错误。
InvalidHTTPRequest	There was an error in the body of your HTTP request.	400 Bad Request	HTTP body格式错误。例如不符合指定的Encoding等。
InvalidURI	Could not parse the specified URI.	400 Bad Request	URI形式不正确。例如一些服务定义的关键词不匹配等。对于ID不匹配等问题，应定义更加具体的错误码，例如NoSuchKey。
MalformedJSON	The JSON you provided was not well-formed.	400 Bad Request	JSON格式不合法。
InvalidVersion	The API version is not supported.	404 Not Found	API的版本号不合法。

rsion	The API version specified was invalid.	Not Found	URI的版本号不合法。
OptInRequired	A subscription for the service is required.	403 Forbidden	没有开通对应的服务。
PreconditionFailed	The specified If-Match header doesn't match the ETag header.	412 Precondition Failed	详见ETag。
Request Expired	Request has expired. Timestamp date is XXX.	400 Bad Request	请求超时。XXX要改成x-bce-date的值。如果请求中只有Date，则需要将Date转换为datetime。
IdempotentParameterMismatch	The request uses the same client token as a previous, but non-identical request.	403 Forbidden	clientToken对应的API参数不一样。
SignatureDoesNotMatch	The request signature we calculated does not match the signature you provided. Check your Secret Access Key and signing method. Consult the service documentation for details.	400 Bad Request	Authorization头域中附带的签名和服务端验证不一致。

XUPERBAAS错误码

错误码	消息	HTTP状态码	语义
WrongNetworkType	wrong network type	400 Bad Request	错误的网络类型
IllegalArguments	illegal arguments, check your input	400 Bad Request	参数错误
NetworkNotExist	network not exist	404 Not Found	网络不存在
OperationFailed	network is in status: {}, operation failed	403 Forbidden	网络处于禁止删除的状态
IllegalSubServiceType	illegal subServiceType	400 Bad Request	不合法的配置类型
IllegalNodeNum	nodeNum must be a multiple of logical zone size	400 Bad Request	不合法的节点数
LogicalZoneNotExist	logical zone is not exist	400 Bad Request	可用区不存在
JoinFailed	consortium member is not running	400 Bad Request	联盟其他成员操作中，当前不可加入
ChargeStopped	request forbidden	400 Bad Request	用户欠费，禁止操作
ErrEndorsementPolicy	ErrEndorsementPolicy	400 Bad Request	背书策略错误
ErrCCConflict	ErrCCConflict	400 Bad Request	链码版本冲突
ErrCCContent	ErrCCContent	400 Bad Request	链码内容错误
ErrCCFileTimeOut	ErrCCFileTimeOut	400 Bad Request	链码文件超时
ErrCCName	ErrCCName	400 Bad Request	链码文件名称不合法

网络创建相关接口

创建网络接口

接口描述

创建一个区块链网络，当前可选的网络类别有超级链试用网络、Fabric联盟链创建与加入、Quorum联盟创建与加入和以太坊四类，其中以以太坊可选普通、高级和豪华三种配置，Fabric联盟链创建与加入、Quorum联盟创建与加入网络可选普通和豪华两种配置；必须填写网络所对应的参数，超级链试用网络对应参数为xChainExperienceForm，创建Fabric联盟链对应的参数为fabricConsortiumForm，加入Fabric联盟链参数为joinFabricConsortiumForm，创建Quorum联盟链对应的参数quorumConsortiumForm，加入Quorum联盟链对应的参数为joinQuorumConsortiumForm，以太坊对应参数为ethereumForm。

权限说明

请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项

如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法


```
POST /v1/blockchain HTTP/1.1
Host: bbe.bj.baidubce.com
Date: <Date>
Authorization: <Authorization_String>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
networkType	String	body	指定创建区块链网络的类型，xChainExperience代超级链试用网络，fabricConsortiumOwner代表创建Fabric联盟，fabricConsortiumParticipant代表加入fabric联盟,quorumConsortiumOwner代表创建Quorum联盟，quorumConsortiumParticipant代表加入Quorum联盟,ether代表以太坊网络	必须
bbeOrderContent	BaseCreateOrderRequestVo	body	bcc创建的订单内容,ITEM表示BBE订单模型，参考 BbeConfig	必须
logicalZone	[]String	body	节点可用区，可填zoneA, zoneB, zoneC	必须
xChainExperienceForm	XChainExperienceForm	body	超级链网络参数	选填，如networkType 为 xChainExperience，则此项必填
fabricConsortiumForm	FabricConsortiumForm	body	创建Fabric联盟	选填，如networkType 为 fabricConsortiumOwner，则此项必填
joinFabricConsortiumForm	JoinFabricConsortiumForm	body	加入Fabric联盟	选填，如networkType 为 fabricConsortiumParticipant，则此项必填
quorumConsortiumForm	QuorumConsortiumForm	body	创建Quorum联盟	选填，如networkType 为 quorumConsortiumOwner，则此项必填
joinQuorumConsortiumForm	JoinQuorumConsortiumForm	body	加入Quorum联盟	选填，如networkType 为 quorumConsortiumParticipant，则此项必填
ethereumForm	EthereumForm	body	以太坊网络参数	选填，如networkType 为 ether，则此项必填

响应（Response）

- 响应头域
除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
networkUuid	String	网络Id

失败响应参考错误码部分。

示例

- 创建超级链试用网络请求示例

```

POST /v1/blockchain HTTP/1.1
content-type: application/json
host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/0b9f07c38e8f3815424002ef2326b2d66d73439f591d561b7d5bfa3fad8c558d
{
  "networkType": "xChainExperience",
  "bbeOrderContent": {
    "region": "bj",
    "orderType": "NEW",
    "items": [
      {
        "config": {
          "serviceType": "BBE",
          "productType": "prepay",
          "key": "bbe",
          "count": 1,
          "time": 2,
          "timeUnit": "DAY",
          "flavor": [
            {
              "name": "subServiceType",
              "value": "XuperChainTrail"
            },
            {
              "name": "traildays",
              "value": 1
            }
          ]
        }
      }
    ]
  },
  "xChainExperienceForm": {
    "tpl": "xplatform",
    "address": "jvPVe8vkJL5xFtAqN42xADroLGwkEzGhz",
    "sign": "",
    "netType": "4",
    "deployType": "00000010",
    "nodeObj": {
      "level": 1,
      "nodes": [
        {
          "id": 5000,
          "displayName": "节点名称2",
          "count": 1
        }
      ]
    },
    "techBased": 5,
    "netName": "baby_hello",
    "netNameCn": "中国好孩阿达",
    "consensus": "single",
    "isExplorer": "1"
  }
}

```

- 创建以太坊网络请求示例

```
POST /v1/blockchain HTTP/1.1
content-type: application/json
host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/0b9f07c38e8f3815424002ef2326b2d66d73439f591d561b7d5bfa3fad8c558d
{
  "bbeOrderContent": {
    "items": [{
      "config": {
        "flavor": [{
          "name": "subServiceType",
          "value": "EthereumDefaultConf"
        },
        {
          "name": "nodeNum",
          "value": "1"
        }
      ]
    }
  ]},
  "logicalZone": [
    "zoneC"
  ],
  "networkType": "ether",
  "ethereumForm": {
    "networkName": "test-ether",
    "nodeNumber": 2,
    "targetGasLimit": 4294967295,
    "maxPeers": 2,
    "verbosity": 1,
    "syncMode": "full",
    "gasPrice": 18000000000,
    "difficulty": 1024,
    "gasLimit": 4294967295
  }
}
```

- 创建Fabric联盟链

```
POST /v1/blockchain HTTP/1.1
content-type: application/json
host: bbe.bj.baidubce.com
x-bce-date: 2019-10-22T13:32:30Z
x-bce-request-id: 028bd2fe-8c80-4d2d-add5-090aeeeba47b
authorization: bce-auth-v1/3bfb8ac1a424d8a9c611caa5e750e60/2019-10-22T13:32:30Z/1800/host/f14313bd47eb9feebe7e8c5769e5ab1aa7b4fdd216cf4e8159ac2ecbfb5c7980
{
  "bbeOrderContent": {
    "items": [{
      "config": {
        "flavor": [{
          "name": "subServiceType",
          "value": "ConsortiumCreateDefaultConf"
        }]
      }
    ]
  },
  "logicalZone": [
    "zoneC"
  ],
  "networkType": "fabricConsortiumOwner",
  "fabricConsortiumForm": {
    "name": "first",
    "englishName": "first",
    "channelPolicy": "any",
    "consensus": "kafka",
    "ordererOrg": "ca38e31c-17b1-2d71-1269-a180972d82a3",
    "description": "fabric-cons-qa-auto-test"
  }
}
```

- 加入Fabric联盟链

```
POST /v1/blockchain HTTP/1.1
content-type: application/json
host: bbe.bj.baidubce.com
x-bce-date: 2019-10-22T13:32:30Z
x-bce-request-id: 028bd2fe-8c80-4d2d-add5-090aeeeba47b
authorization: bce-auth-v1/3bfbc8ac1a424d8a9c611caa5e750e60/2019-10-22T13:32:30Z/1800/host/f14313bd47eb9feebe7e8c5769e5ab1aa7b4fdd216cf4e8159ac2ecbf5c7980
{
  "bbeOrderContent": {
    "items": [{
      "config": {
        "flavor": [{
          "name": "subServiceType",
          "value": "ConsortiumJoinDefaultConf"
        }]
      }
    ]
  },
  "logicalZone": [
    "zoneC"
  ],
  "networkType": "fabricConsortiumParticipant",
  "joinFabricConsortiumForm": {
    "consortiumUuid": "dcc7fc68-b281-e98d-37aa-73cd15c6eef0",
    "orgUuid": "75339ab5-4e5c-fbdd-ac94-66a457d199c6",
    "approvalUuid": "7180faa0-28a9-a7f4-940f-24d0995ccd90-0"
  }
}
```

- 创建Quorum联盟请求示例

```
POST /v1/blockchain HTTP/1.1
content-type: application/json
host: bbe.bj.baidubce.com
x-bce-date: 2019-10-22T12:02:48Z
x-bce-request-id: abaae432-7f8b-4d07-bc6b-245a36dc305d
authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-22T12:02:48Z/1800/host/df7101be69d2cef005d61a6f4c549127f120971d0c7dc69b7b42c33feaed6d
{
  "bbeOrderContent": {
    "items": [{
      "config": {
        "flavor": [{
          "name": "subServiceType",
          "value": "QuorumDefaultConf"
        }]
      }
    }]
  },
  "logicalZone": [
    "zoneC"
  ],
  "networkType": "quorumConsortiumOwner",
  "quorumConsortiumForm": {
    "orgUuid": "bf22b53a-793f-4e81-8a48-b16cfff64999",
    "name": "test-quorum",
    "domain": "test-quorum",
    "description": "test-quorum",
    "nodeNumber": 1,
    "gasLimit": 4294967295,
    "targetGasLimit": 4294967295,
    "consensus": "IBFT",
    "blockPeriod": 5
  }
}
```

- 加入Quorum联盟请求示例

```
POST /v1/blockchain HTTP/1.1
content-type: application/json
host: bbe.bj.baidubce.com
x-bce-date: 2019-10-22T12:02:48Z
x-bce-request-id: 51947834-abd5-4158-b703-e6f77c03c041
authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-22T12:02:48Z/1800/host/df7101be69d2cef005d61a6f4c549127f120971d0c7dc69b7b42c33feaed6d3
{
  "bbeOrderContent": {
    "items": [{
      "config": {
        "flavor": [{
          "name": "subServiceType",
          "value": "QuorumDefaultConf"
        }]
      }
    }]
  },
  "logicalZone": [
    "zoneC"
  ],
  "networkType": "quorumConsortiumParticipant",
  "joinQuorumConsortiumForm": {
    "orgUuid": "bf22b53a-793f-4e81-8a48-b16cff64999",
    "applicationUuid": "55a723e9-c0ef-41e4-979d-706751a9531b",
    "nodeNumber": 1
  }
}
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 10:25:02 GMT
{
  "data": {
    "networkUuid": "b80f8971-c21f-46bc-967d-9a35e09b60d4"
  }
}
```

Quorum相关接口

联盟相关接口

获取Quorum联盟列表

获取Quorum联盟列表

接口描述 获取用户部署的Quorum联盟列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法


```
GET /v1/quorum/consortium/consortiums?orgUuid={OrgUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query参数	组织uuid，为空表示不指定组织，返回所有加入的联盟	选填

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
orderBy	String	查询结果排序依据字段
order	String	排序
pageNo	Integer	查询结果页码
pageSize	Integer	每页包含的最大数量
totalCount	Integer	查询结果条数
result	[]QuorumNetworkInfo	网络详情列表， 结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/consortium/consortiums?orgUuid=bd1f706b-965e-ea99-1071-57b5904b3d73 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T02:49:15Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T02:49:15Z/1800/host/51d09d99b383621ddd4052c55dcd4a3b334b0a103c782fd4ccb58d9e1c1b26cd
```

- 响应示例

```
200
Content-Length: 2414
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 02:49:15 GMT
{
  "orderBy": "",
  "order": "",
  "pageNo": 1,
  "pageSize": 1000,
  "totalCount": 1,
  "result": [
    {
      "name": "test-quorum",
      "domain": "test-quorum",
      "uuid": "35b33e7e-62f5-a875-7b39-43a5b7de3d0f",
      "orgUuid": "bd1f706b-965e-ea99-1071-57b5903b3d73",
```

```

    "description": "test-quorum",
    "status": "RUNNING",
    "createTime": "2019-09-03T10:51:51+08:00",
    "consensus": "IBFT",
    "genesisConfig": {
      "config": {
        "chainId": 1567479110,
        "homesteadBlock": 0,
        "eip150Block": 0,
        "eip150Hash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
        "eip155Block": 0,
        "eip158Block": 0,
        "byzantiumBlock": 0,
        "istanbul": {
          "epoch": 30000,
          "policy": 0
        },
        "txnSizeLimit": 0
      },
      "nonce": "0x0",
      "timestamp": "0x0",
      "extraData":
"0x0000000000000000000000000000000000000000000000000000000000000000f89af85494424bfb3d26adf10
",
      "gasLimit": "0xffffffff",
      "difficulty": "0x1",
      "mixHash": "0x63746963616c2062797a616e74696e652066661756c7420746f6c6572616e6365",
      "coinbase": "0x000000000000000000000000000000000000000000000000",
      "alloc": {
        "20a12bfa8a1bdd676da3311a9ea1cf6bdd415426": {
          "balance": "0xc097ce7bc90715b34b9f1000000000"
        },
        "253035da1b1d7cbdf6f7a4d79149a9fd03824053": {
          "balance": "0xc097ce7bc90715b34b9f1000000000"
        },
        "6559675fef6a2107b9a8176c2fa6d18a37084780": {
          "balance": "0xc097ce7bc90715b34b9f1000000000"
        },
        "9b1fc94ba654375b375091248543772e6f46e8ad": {
          "balance": "0xc097ce7bc90715b34b9f1000000000"
        },
        "ec4859413cb7f06fdd5cc2a225a90ba12645df63": {
          "balance": "0xc097ce7bc90715b34b9f1000000000"
        }
      },
      "number": "0x0",
      "gasUsed": "0x0",
      "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000"
    },
    "nodeConfig": {
      "targetGasLimit": 4294967295,
      "maxPeer": 25,
      "syncMode": "full",
      "gasPrice": 0,
      "blockPeriod": 5,
      "istanbulRequestTimeout": 10000,
      "cpuLimit": 0,
      "memoryLimit": 0
    },
    "errorMessage": "",
    "members": [

```

```
{
  "uuid": "e0e7cf90-5255-43e9-9615-7aff3b6fdd59",
  "orgUuid": "bd1f706b-965e-ea99-1071-57b5903b3d73",
  "orgName": "org0",
  "orgDomain": "org0",
  "consortiumUuid": "35b33e7e-62f5-a875-7b39-42a5b7de3d0f",
  "consortiumName": "test-quorum",
  "consortiumDomain": "test-quorum",
  "userId": "eca97e148cb74e9683d7b7240829d1ff",
  "cluster_id": "c-46htTlaQ",
  "validatorNumber": 4,
  "transactionNodeNumber": 1,
  "joinedTime": "2019-09-03T10:51:51+08:00",
  "status": "RUNNING",
  "errMessage": ""
}
```

🔗 获取Quorum联盟详情

获取Quorum联盟详情

接口描述 获取用户部署的Quorum联盟详情。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/quorum/consortium/consortium?consortiumUuid={ConsortiumUuid}&orgUuid={OrgUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query参数	组织Uuid	必须
consortiumUuid	String	Query参数	联盟Uuid	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
success	Bool	请求结果，true表示请求成功
result	[]QuorumNetworkInfo	网络详情列表， 结构具体定义见附录说明

失败响应参考错误码部分。

示例

● 请求示例

```
GET /v1/quorum/consortium/consortium?orgUuid=bd1f706b-965e-ea99-1071-57b5903b3d73&consortiumUuid=35b33e7e-62f5-a875-7b39-42a5b7de3d0f HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

● 响应示例

```
200
Content-Length: 3080
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": {
    "name": "test-quorum",
    "domain": "test-quorum",
    "uuid": "35b33e7e-62f5-a875-7b39-42a5b7de3d0f",
    "orgUuid": "bd1f706b-965e-ea99-1071-57b5903b3d73",
    "description": "test-quorum",
    "status": "RUNNING",
    "createTime": "2019-09-03T10:51:51+08:00",
    "consensus": "IBFT",
    "genesisConfig": {
      "config": {
        "chainId": 1567479110,
        "homesteadBlock": 0,
        "eip150Block": 0,
        "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
        "eip155Block": 0,
        "eip158Block": 0,
        "byzantiumBlock": 0,
        "istanbul": {
          "epoch": 30000,
          "policy": 0
        },
        "txnSizeLimit": 0
      },
      "nonce": "0x0",
      "timestamp": "0x0",
      "extraData":
"0x0000000000000000000000000000000000000000000000000000000000000000f89af85494424bfb3d26adf10",
      "gasLimit": "0xffffffff",
      "difficulty": "0x1",
      "mixHash": "0x63746963616c2062797a616e74696e65206661756c7420746f6c6572616e6365",
      "coinbase": "0x0000000000000000000000000000000000000000000000000000000000000000",
      "alloc": {
        "20a12bfa8a1bdd676da3311a9ea1cf6bdd415426": {
          "balance": "0xc097ce7bc90715b34b9f1000000000"
        },
        "253035da1b1d7cbdf6f7a4d79149a9fd03824053": {
          "balance": "0xc097ce7bc90715b34b9f1000000000"
        },
        "6559675fef6a2107b9a8176c2fa6d18a37084780": {
```

```

    "balance": "0xc097ce7bc90715b34b9f1000000000"
  },
  "9b1fc94ba654375b375091248543772e6f46e8ad": {
    "balance": "0xc097ce7bc90715b34b9f1000000000"
  },
  "ec4859413cb7f06fdd5cc2a225a90ba12645df63": {
    "balance": "0xc097ce7bc90715b34b9f1000000000"
  }
},
"number": "0x0",
"gasUsed": "0x0",
"parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
},
"nodeConfig": {
  "targetGasLimit": 4294967295,
  "maxPeer": 25,
  "syncMode": "full",
  "gasPrice": 0,
  "blockPeriod": 5,
  "istanbulRequestTimeout": 10000,
  "cpuLimit": 0,
  "memoryLimit": 0
},
"errorMessage": "",
"members": [{
  "uuid": "e0e7cf90-5255-43e9-9615-7aff3b6fdd59",
  "orgUuid": "bd1f706b-965e-ea99-1071-57b5903b3d73",
  "orgName": "org0",
  "orgDomain": "org0",
  "consortiumUuid": "35b33e7e-62f5-a875-7b39-42a5b7de3d0f",
  "consortiumName": "test-quorum",
  "consortiumDomain": "test-quorum",
  "userId": "eca97e148cb74e9683d7b7240829d1ff",
  "cluster_id": "c-46htTlfQ",
  "validatorNumber": 4,
  "transactionNodeNumber": 1,
  "joinedTime": "2019-09-03T10:51:51+08:00",
  "status": "RUNNING",
  "errMessage": ""
}]
},
"code": ""
}

```

🔗 获取Quorum联盟节点

获取Quorum联盟节点

接口描述 获取用户部署的Quorum联盟节点。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/quorum/consortium/nodes?consortiumUuid={ConsortiumUuid}&orgUuid={OrgUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query参数	组织Uuid	必须
consortiumUuid	String	Query参数	联盟Uuid	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
pageNo	Integer	查询结果页码
pageSize	Integer	每页包含的最大数量
result	[]QuorumNodeInfo	节点详情列表， 结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/consortium/nodes?orgUuid=bd1f706b-965e-ea99-1071-57b5903b3d73&consortiumUuid=35b33e7e-62f5-a875-7b39-42a5b7de3d0f HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 3080
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "pageNo": 1,
  "pageSize": 1000,
  "result": [{
    "uuid": "8b543a22-9724-47da-9546-7905321900c8",
    "shortId": "8b543a22",
    "orgUuid": "bd1f706b-965e-ea99-1071-57b5903b3d73",
    "consortiumUuid": "35b33e7e-62f5-a875-7b39-42a5b7de3d0f",
    "consortiumMemberUuid": "e0e7cf90-5255-43e9-9615-7aff3b6fdd59",
    "userId": "eca97e148cb74e9683d7b7240829d1ff",
    "cluster_id": "c-46htIfQ",
    "type": "validatorNode",
    "nodeConfig": {
      "targetGasLimit": 4294967295,
```

```

    "maxPeer": 25,
    "syncMode": "full",
    "gasPrice": 0,
    "blockPeriod": 5,
    "istanbulRequestTimeout": 10000,
    "cpuLimit": 0,
    "memoryLimit": 0
  },
  "coinbase": "0x424Bfb3d26AdF107E649Cc53D41fc300072e56CF",
  "keystore": "...",
  "contractKey": "IRypnANDc3fZpxh53u4rXvmJpZnE6ii37ZaWPvALdxs=",
  "jsonRPC": "http://xxx:8081/quorum-jsonrpc/e0e7cf90-5255-43e9-9615-7aff3b6fdd59/8b543a22",
  "contractEditorUrl": "http://xxx:8081/quorum-cakeshop/e0e7cf90-5255-43e9-9615-7aff3b6fdd59/8b543a22/cakeshop/sandbox.html",
  "blockChainBrowserUrl": "http://xxx:8081/quorum-cakeshop/e0e7cf90-5255-43e9-9615-7aff3b6fdd59/8b543a22/cakeshop/#section=explorer",
  "nodeConsoleUrl": "http://xxx:8081/quorum-cakeshop/e0e7cf90-5255-43e9-9615-7aff3b6fdd59/8b543a22/cakeshop/#section=console",
  "status": "RUNNING",
  "createTime": "2019-09-03T10:51:50+08:00"
},
{
  "uuid": "a56d49f8-61d1-41ad-9515-93743214ff10",
  "shortId": "a56d49f8",
  "orgUuid": "bd1f706b-965e-ea99-1071-57b5903b3d73",
  "consortiumUuid": "35b33e7e-62f5-a875-7b39-42a5b7de3d0f",
  "consortiumMemberUuid": "e0e7cf90-5255-43e9-9615-7aff3b6fdd59",
  "userId": "eca97e148cb74e9683d7b7240829d1ff",
  "cluster_id": "c-46htTlfQ",
  "type": "transactionNode",
  "nodeConfig": {
    "targetGasLimit": 4294967295,
    "maxPeer": 25,
    "syncMode": "full",
    "gasPrice": 0,
    "blockPeriod": 5,
    "istanbulRequestTimeout": 10000,
    "cpuLimit": 0,
    "memoryLimit": 0
  },
  "coinbase": "0xe539E83F8a6E402dea476fF8A61A202f0A1c8E83",
  "keystore": "...",
  "contractKey": "LfVoXHNhPID+MYwQa5e60KILThyTM2GXNcuEx6RoViY=",
  "jsonRPC": "http://xxx:8081/quorum-jsonrpc/e0e7cf90-5255-43e9-9615-7aff3b6fdd59/a56d49f8",
  "contractEditorUrl": "http://xxx:8081/quorum-cakeshop/e0e7cf90-5255-43e9-9615-7aff3b6fdd59/a56d49f8/cakeshop/sandbox.html",
  "blockChainBrowserUrl": "http://xxx:8081/quorum-cakeshop/e0e7cf90-5255-43e9-9615-7aff3b6fdd59/a56d49f8/cakeshop/#section=explorer",
  "nodeConsoleUrl": "http://xxx:8081/quorum-cakeshop/e0e7cf90-5255-43e9-9615-7aff3b6fdd59/a56d49f8/cakeshop/#section=console",
  "status": "RUNNING",
  "createTime": "2019-09-03T10:51:51+08:00"
}
]
}

```

退出Quorum联盟

退出Quorum联盟

接口描述 退出Quorum联盟。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
DELETE /v1/quorum/consortium/consortium?orgUuid={OrgUuid}&consortiumUuid={ConsortiumUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query参数	组织Uuid	必须
consortiumUuid	String	Query参数	联盟Uuid	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。

- 响应参数

无
失败响应参考错误码部分。

示例

- 请求示例

```
DELETE /v1/quorum/consortium/consortium?orgUuid=bd1f706b-965e-ea79-1071-57b5903b3d73&consortiumUuid=35b33e7e-62f5-a885-7b39-42a5b7de3d0f HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
```

组织相关接口

创建组织

创建组织

接口描述 创建组织。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回 403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
POST /v1/quorum/consortium/org HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
name	String	Body参数	组织名称	必须
domain	String	Body参数	组织英文名称	必须
description	String	Body参数	组织描述	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
success	Bool	请求结果，true表示请求成功
code	String	错误码
result	String	组织Uuid

失败响应参考错误码部分。

示例

- 请求示例

```
POST /v1/quorum/consortium/org HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
{
  "name": "test-org",
  "domain": "test-org",
  "description": "test-org"
}
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": "60548153-1930-434f-8c36-0ba827ae9368",
  "code": ""
}
```

🔗 获取组织列表

获取组织列表

接口描述 获取组织列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/quorum/consortium/orgs HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数
无

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
pageNo	Integer	查询结果页码
pageSize	Integer	当前页面包含结果条数
result	[]OrgInfo	组织结果列表， 结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/consortium/orgs HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

● 响应示例

```
200
Content-Length: 532
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "pageNo": 1,
  "pageSize": 2,
  "result": [{
    "uuid": "bd1f706b-965e-ea99-1071-57b5903b3d73",
    "name": "org0",
    "domain": "org0",
    "description": "test",
    "userId": "eca97e148cb74e9683d7b7240829d1ff",
    "createTime": "2019-07-29T16:16:11+08:00"
  },
  {
    "uuid": "60548153-1930-434f-8c36-0ba827ae9368",
    "name": "test-org",
    "domain": "test-org",
    "description": "test-org",
    "userId": "eca97e148cb74e9683d7b7240829d1ff",
    "createTime": "2019-10-23T15:10:56+08:00"
  }
]
```

🔗 获取组织详情

获取组织详情

接口描述 获取组织详情。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

● 请求语法

```
GET /v1/quorum/consortium/org?orgUuid={OrgUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

● 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query参数	组织Uuid	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。

● 响应参数

名称	类型	描述
success	Bool	查询结果，true表示查询成功
code	String	结果码
result	OrgInfo	组织信息， 结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/consortium/org?orgUid=60548153-1930-434f-8c36-0ba827ae9368 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 1499
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": {
    "uuid": "60548153-1930-434f-8c36-0ba827ae9368",
    "name": "test-org",
    "domain": "test-org",
    "description": "test-org",
    "userId": "eca97e148cb74e9683d7b7240829d1ff",
    "cert": "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----\n",
    "createTime": "2019-10-23T15:10:56+08:00"
  },
  "code": ""
}
```

🔗 删除组织

删除组织

接口描述 删除组织。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
DELETE /v1/quorum/consortium/org/{orgUid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	URL参数	组织Uuid	必须

响应 (Response)

- 响应头域 除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
success	Bool	查询结果，true表示查询成功
code	String	结果码
result	String	删除结果

失败响应参考错误码部分。

示例

请求示例

```
DELETE /v1/quorum/consortium/org/60548153-1930-434f-8c36-0ba827ae9368 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

响应示例

```
200
Content-Length: 1499
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": "ok",
  "code": ""
}
```

事件相关接口

发起加入联盟邀请

发起加入联盟邀请

接口描述 向百度云用户发起加入Quorum联盟邀请。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求 (Request)

请求语法

```
POST /v1/quorum/consortium/invitation HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
fromOrgUuid	String	Body参数	邀请方组织Uuid	必须
toUserId	String	Body参数	被邀请方百度云账户id	必须
consortiumUuid	String	Body参数	联盟Uuid	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
success	Bool	查询结果，true表示查询成功
code	String	结果码
result	String	发起邀请结果

失败响应参考错误码部分。

示例

- 请求示例

```
POST /v1/quorum/consortium/invitation HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8

{
  "fromOrgUuid": "384badd7-e12f-e3e1-dacf-54dfcea6bec1",
  "toUserId": "5e6972004e8649b19c2c870526d4c9a0",
  "consortiumUuid": "a438505a-cfdc-87c7-16f4-b16d8f5ed2ef"
}
```

- 响应示例

```
200
Content-Length: 1499
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": "ok",
  "code": ""
}
```

🔗 获取申请列表

获取申请列表

接口描述 获取联盟相关申请列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/quorum/consortium/applications?orgUuid={OrgUuid}&done={Done} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query参数	组织Uuid	非必须
done	Bool	Query参数	是否已完成	非必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。

- 响应参数

名称	类型	描述
pageNo	Integer	查询结果页数
pageSize	Integer	查询结果条数
result	[]ApplicationsInfo	申请结果列表， 结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/consortium/applications?orgUuid=384badd7-e02f-e3e1-dacf-54dfcea6bec1&done=false HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 620
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "pageNo": 1,
  "pageSize": 1,
  "result": [{
    "uuid": "00b4769c-4134-4813-95f5-6476f6e7ff44",
    "consortiumUuid": "a438505a-cfdc-87c6-16f4-b16d8f5ed2ef",
    "target": "Consortium: test-quorum",
    "targetUuid": "a438505a-cfdc-87c6-16f4-b16d8f5ed2ef",
    "initiator": "test-org",
    "initiatorUuid": "384badd7-e02f-e3e1-dacf-54dfcea6bec1",
    "userId": "eca97e148cb74e9683d7b7240829d1ff",
    "orgUuid": "384badd7-e02f-e3e1-dacf-54dfcea6bec1",
    "type": "InviteJoinConsortium",
    "state": "DOING",
    "createTime": "2019-10-23T16:33:59+08:00",
    "done": false
  }]
}
```

🔗 获取审批列表

获取审批列表

接口描述 获取联盟相关审批列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/quorum/consortium/approvals?orgUuid={OrgUuid}&done={Done} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query参数	组织Uuid	非必须
done	Bool	Query参数	是否已完成	非必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
pageNo	Integer	查询结果页数
pageSize	Integer	查询结果条数
result	[]ApprovalsInfo	审批结果列表， 结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/consortium/approvals?orgUuid=384badd7-e02f-e3e1-dacf-54dfcea6bec1&done=false HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 620
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "pageNo": 1,
  "pageSize": 1,
  "result": [{
    "uuid": "d52fb004-3483-f7f5-aa4d-4e371e559460-0",
    "applicationUuid": "d52fb004-3483-f7f5-aa4d-4e371e559460",
    "target": "Consortium: test-quorum",
    "initiator": "5e6972004e8649b19c0c870526d5c9a0",
    "initiatorUuid": "",
    "orgUuid": "384badd7-e02f-e3e1-dacf-54dfcea6bec1",
    "orgName": "test-org",
    "userId": "eca97e148cb74e9683d7b7240829d1ff",
    "type": "JoinConsortium",
    "state": "DOING",
    "message": "",
    "createTime": "2019-10-23T16:53:45+08:00",
    "finishTime": "2019-10-23T16:53:45+08:00",
    "done": false
  }]
}
```

🔗 获取审批流

获取审批流

接口描述 获取审批流。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求 (Request)

- 请求语法

```
GET /v1/quorum/consortium/approval/{applicationUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
applicationUuid	String	URL参数	申请Uuid	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
pageNo	Integer	查询结果页数
pageSize	Integer	查询结果条数
result	[]ApprovalsInfo	审批流结果， 结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/consortium/approval/d52fb004-3483-f7f5-aa4d-4e371e559460 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 644
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "pageNo": 1,
  "pageSize": 1,
  "result": [{
    "uuid": "d52fb004-3483-f7f5-aa4d-4e371e559460-0",
    "applicationUuid": "d52fb004-3483-f7f5-aa4d-4e371e559460",
    "target": "Consortium: test-quorum",
    "initiator": "5e6972004e8649b19c0c870526d5c9a0",
    "initiatorUuid": "",
    "orgUuid": "384badd7-e02f-e3e1-dacf-54dfcea6bec1",
    "orgName": "test-org",
    "userId": "eca97e148cb74e9683d7b7240829d1ff",
    "type": "JoinConsortium",
    "state": "DOING",
    "message": "",
    "createTime": "2019-10-23T16:53:45+08:00",
    "finishTime": "2019-10-23T16:53:45+08:00",
    "done": false
  }]
}
```

通过／驳回审批

通过／驳回审批

接口描述 通过或驳回审批。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
PATCH /v1/quorum/consortium/approval/{approvalUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
approvalUuid	String	URL参数	审批Uuid	必须
approved	Bool	Body参数	true通过，false驳回	必须
reasonMsg	String	Body参数	通过或驳回原因	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。

- 响应参数

名称	类型	描述
success	Bool	请求结果，true表示请求成功
result	String	处理结果
code	String	结果码

失败响应参考错误码部分。

示例

- 请求示例

```
PATCH /v1/quorum/consortium/approval/d52fb004-3483-f7f5-aa4d-4e371e559460 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
{
  "approved":true,
  "reasonMsg":"通过"
}
```

- 响应示例

```
200
Content-Length: 53
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": "ok",
  "code": ""
}
```

钱包账户相关接口

创建新账户

创建新账户

接口描述 创建新账户。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求 (Request)

- 请求语法

```
POST /v1/quorum/account HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	String	Body参数	联盟Uuid	必须
orgUuid	String	Body参数	组织Uuid	必须
balance	Integer	Body参数	账户余额	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
success	Bool	请求结果，true表示请求成功
address	String	账户地址
txId	String	交易Hash

失败响应参考错误码部分。

示例

- 请求示例

```
POST /v1/quorum/account HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
{
  "balance":10000,
  "consortiumUuid":"a438505a-cfdc-87c6-16f4-b16d8f5ed2ef",
  "orgUuid":"384badd7-e02f-e3e1-dacf-54dfcea6bec1"
}
```

- 响应示例

```
200
Content-Length: 196
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": {
    "address": "0xA221D5815feC308EF7B30a9685aF8B682572Ee53",
    "txId": "0xe380b6e9f1b6a681ec91dcc526f47cd333bf65bade83a61aa55b282969f30a"
  },
  "code": ""
}
```

🔗 获取钱包账户列表

获取钱包账户列表

接口描述 获取钱包账户列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/quorum/accounts?consortiumUuid={ConsortiumUuid}&orgUuid={OrgUuid}&accountType={AccountType}
HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	String	Query 参数	联盟Uuid	必须
orgUuid	String	Query 参数	组织Uuid	必须
accountType	String	Query 参数	账户类型，可选类型包括：genesis：查询预定义账户；currentOrg：查询当前组织创建的账户；otherOrg：查询其他组织创建的账户	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
pageNo	Integer	查询结果页码
pageSize	Integer	当前页面包含结果条数
address	String	账户地址
privateKey	String	私钥，需要通过私钥接口查询
balance	String	余额
consortiumUuid	String	联盟Uuid
orgUuid	String	组织Uuid
createTime	String	创建时间

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/accounts?consortiumUuid=a438505a-cfdc-87c6-16f4-b16d8f5ed2ef&orgUuid=384badd7-e02f-e3e1-dacf-54dfcea6bec1&accountType=genesis HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 1442
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "pageNo": 1,
  "pageSize": 5,
  "result": [{
    "address": "0x41Eb707473c53D20cCDF2e1a1f69dc0a395A9baE",
    "privateKey": "",
    "balance": 1000000000000000000,
    "consortiumUuid": "a438505a-cfdc-87c6-16f4-b16d8f5ed2ef",
    "orgUuid": "genesis",
    "createTime": "2019-10-23T15:01:15+08:00"
  },
  {
    "address": "0x329A6dC6F020Fe0a8a7aCAb1B3B7E32573101beF",
    "privateKey": "",
    "balance": 1000000000000000000,
    "consortiumUuid": "a438505a-cfdc-87c6-16f4-b16d8f5ed2ef",
    "orgUuid": "genesis",
    "createTime": "2019-10-23T15:01:15+08:00"
  }
  ]
}
```

🔗 查询账户私钥

查询账户私钥

接口描述 查询账户私钥。

权限说明 请求发起人需要具有合法的AccessKeyId和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求 (Request)

- 请求语法

```
GET /v1/quorum/privatekey?consortiumUuid={ConsortiumUuid}&orgUuid={OrgUuid}&address={Address} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	String	Query参数	联盟Uuid	必须
orgUuid	String	Query参数	组织Uuid	必须
address	String	Query参数	账户地址	必须

响应 (Response)

- 响应头域 除公共头域外，无其它特殊头域。

- 响应参数

名称	类型	描述
success	Bool	请求结果，true表示请求成功
result	String	账户私钥
txId	String	交易Hash

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/privatekey?consortiumUid=a438505a-cfdc-87c6-16f4-b16d8f5ed2ef&orgUid=384badd7-e02f-e3e1-dacf-54dfcea6bec1&address=0xA221D5815feC308EF7B30a9685aF8B682572Ee53 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 115
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": "0b6d5ee1809c7b0a83588dd6a809b26f4169e7e5930a45d13582ff3de30783703",
  "code": ""
}
```

🔗 下载账户keystore

下载账户keystore

接口描述 下载账户keystore。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/quorum/keystore?consortiumUid={ConsortiumUid}&orgUid={OrgUid}&address={Address}&passphrase={Passphrase} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	String	Query参数	联盟Uuid	必须
orgUuid	String	Query参数	组织Uuid	必须
address	String	Query参数	账户地址	必须
passphrase	String	Query参数	账户加密密码	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
success	Bool	请求结果，true表示请求成功
address	String	账户地址
keystore	String	账户keystore

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/keystore?consortiumUuid=a438505a-cfdc-87c6-16f4-b16d8f5ed2ef&orgUuid=384badd7-e02f-e3e1-dacf-54dfcea6bec1&address=0xA221D5815feC308EF7B30a9685aF8B682572Ee53&passphrase=123456
HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 673
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": {
    "address": "0xA221D5815feC308EF7B30a9685aF8B682572Ee53",
    "keystore": "{\"address\":\"a221d5815fec308ef7b30a9685af8b682572ee53\", \"crypto\":{\"cipher\":\"aes-128-ctr\", \"ciphertext\":\"3067e66d7836fc9b4aa23a17cd7350160656004b4ef9651356102c3103b49e5\", \"cipherparar\":{\"iv\":\"0ccc523c8c9293c312dd21e08aee7158\"}, \"kdf\":\"scrypt\", \"kdfparams\":{\"dklen\":\"32\", \"n\":\"262144\", \"p\":\"1\", \"r\":\"8\", \"salt\":\"fb095dfc069969cc8f4e3fe3d07b00dbe425dd9a5595529be7-5551-4031-9793-6112526bab78\", \"version\":\"3\"}}\", \"code\": \"\"
  }
}
```

删除账户

删除账户

接口描述 删除账户。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
DELETE /v1/quorum/account?consortiumUuid={ConsortiumUuid}&orgUuid={OrgUuid}&address={Address} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	String	Query参数	联盟Uuid	必须
orgUuid	String	Query参数	组织Uuid	必须
address	String	Query参数	账户地址，系统预定义账户不能删除	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。

- 响应参数

名称	类型	描述
success	Bool	请求结果，true表示请求成功
result	String	账户地址
code	String	结果码

失败响应参考错误码部分。

示例

- 请求示例

```
DELETE v1/quorum/account?consortiumUuid=a438505a-cfdc-87c6-16f4-b16d8f5ed2ef&orgUuid=384badd7-e02f-e3e1-dacf-54dfcea6bec1&address=0xA221D5815feC308EF7B30a9685aF8B682572Ee53 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 93
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": "0xA221D5815feC308EF7B30a9685aF8B682572Ee53",
  "code": ""
}
```

合约管理相关接口

🔗 获取合约列表

获取合约列表

接口描述 获取已部署合约列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/quorum/contracts?consortiumUuid={ConsortiumUuid}&orgUuid={OrgUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	String	Query参数	联盟Uuid	必须
orgUuid	String	Query参数	组织Uuid	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
contractUuid	String	合约Uuid
consortiumUuid	String	联盟Uuid
orgUuid	String	组织Uuid
name	String	合约名称
description	String	合约描述
type	String	合约类型，platform为平台基础库合约，user为用户自定义合约
status	String	合约状态，包含None Deployed、Deployed、Deploying和Deployment Failed四种，分别为未部署、运行中，部署中，部署失败
ownerAddress	String	合约发布者地址
isPrivate	Bool	是否为私有合约
privateFor	Array	私钥合约交易地址
createTime	String	创建时间
updateTime	String	更新时间
address	String	合约地址

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/contracts?orgUuid=988ba131-e04b-8111-b44f-fb35b8dc80e6&consortiumUuid=cb73ac79-62bc-6070-aadc-24a05eaf55ae HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```

200
Content-Length: 1335
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "pageNo": 1,
  "pageSize": 2,
  "result": [{
    "contractUuid": "3f90151d-a416-4ca5-99df-dcecf5bfb09e",
    "consortiumUuid": "cb73ac79-62bc-6070-aadc-24a05eaf55ae",
    "orgUuid": "988ba131-e04b-8111-b44f-fb35b8dc80e6",
    "name": ":Simple",
    "description": ":Simple",
    "type": "user",
    "status": "Deployed",
    "ownerAddress": "0x4345d073d9e947fb148802b25c7b6d996bac11a1",
    "isPrivate": true,
    "privateFor": [
      "1wXBgBiyXxaqw/lhlt2om/ldRPNmmiWJvrVDi38ojw0=",
      "gQTX2Tam6aM9/7o+b3ttE+YaTkudJpBVR9T8+/PkBC4="
    ],
    "createdTime": "2019-10-29T16:05:53+08:00",
    "updatedTime": "2019-10-29T16:05:53+08:00",
    "address": "0x46331570c943501915c88ced7a33d9799bcb36572"
  },
  {
    "contractUuid": "840ff186-950e-4169-90f0-69f0a073cac4",
    "consortiumUuid": "cb73ac79-62bc-6070-aadc-24a05eaf55ae",
    "orgUuid": "988ba131-e04b-8111-b44f-fb35b8dc80e6",
    "name": ":SimplePrivate",
    "description": ":Simple",
    "type": "user",
    "status": "Deployed",
    "ownerAddress": "0x4345d073d9e947fb148802b25c7b6d996bac11a1",
    "isPrivate": false,
    "privateFor": [],
    "createdTime": "2019-10-29T16:05:09+08:00",
    "updatedTime": "2019-10-29T16:05:09+08:00",
    "address": "0xea5475c749c93df7c30fd777f2114e3dfbf35bfd"
  }
]
}

```

🔗 获取合约详情

获取合约详情

接口描述 获取已部署合约详情。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求 (Request)

- 请求语法

```
GET /v1/quorum/contract?consortiumUuid={ConsortiumUuid}&orgUuid={OrgUuid}&contractUuid={ContractUuid}
HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	String	Query参数	联盟Uuid	必须
orgUuid	String	Query参数	组织Uuid	必须
contractUuid	String	Query参数	合约Uuid	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
contractUuid	String	合约Uuid
consortiumUuid	String	联盟Uuid
orgUuid	String	组织Uuid
name	String	合约名称
description	String	合约描述
type	String	合约类型，platform为平台基础库合约，user为用户自定义合约
status	String	合约状态，包含None Deployed、Deployed、Deploying和Deployment Failed四种，分别为未部署、运行中，部署中，部署失败
ownerAddress	String	合约发布者地址
isPrivate	Bool	是否为私有合约
privateFor	Array	私钥合约交易地址
createTime	String	创建时间
updateTime	String	更新时间
address	String	合约地址
abi	String	合约abi
sourceCode	Object	合约源代码

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/quorum/contract?orgUuid=988ba131-e04b-8111-b44f-fb35b8dc80e6&contractUuid=3f90151d-a416-4ca5-99df-dcecf5bfb09e&consortiumUuid=cb73ac79-62bc-6070-aadc-24a05eaf55ae HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 2652
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": {
    "contractUuid": "3f90151d-a416-4ca5-99df-dcecf5bfb09e",
    "consortiumUuid": "cb73ac79-62bc-6070-aadc-24a05eaf55ae",
    "orgUuid": "988ba131-e04b-8111-b44f-fb35b8dc80e6",
    "name": ":Simple",
    "description": ":Simple",
    "type": "user",
    "status": "Deployed",
    "ownerAddress": "0x4345d073d9e947fb148802b25c7b6d996bac11a1",
    "isPrivate": true,
    "privateFor": [
      "1wXBgBiyXxaqw/lhlt2om/ldRPNmmiWJvrVDi38ojw0=",
      "gQTX2Tam6aM9/7o+b3ttE+YaTkudJpBVR9T8+/PkBC4="
    ],
    "createdTime": "2019-10-29T16:05:53+08:00",
    "updatedTime": "2019-10-29T16:05:53+08:00",
    "address": "0x46331570c943501915c88ced7a33d979bcb36572",
    "abi": "[{"constant":false,"inputs":[{"name":"i","type":"int64"},{"name":"s","type":"string"}],"name":"set","outputs":[]},"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":[]},"name":"storedS","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":[]},"name":"get","outputs":[{"name":"i","type":"int64"},{"name":"s","type":"string"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs":[]},"name":"storedI","outputs":[{"name":"","type":"int64"}],"payable":false,"stateMutability":"view","type":"function"},{"anonymous":false,"inputs":[{"indexed":true,"name":"from","type":"address"},{"indexed":true,"name":"i","type":"int64"},{"indexed":true,"name":"s","type":"string"},{"indexed":false,"name":"h","type":"bytes32"},{"indexed":false,"name":"m","type":"string"}],"name":"ChangeEvent","type":"event"}],
    "sourceCode": {
      ":Simple": "pragma solidity >=0.4.24;\n\n * @title Simple Storage with events\n * @dev Read and write values to the chain\n */\ncontract Simple {\n  int64 public storedI;\n  string public storedS;\n\n  event ChangeEvent(\n    address indexed from,\n    int64 indexed i,\n    string indexed s,\n    bytes32 h,\n    string m\n  );\n\n  /**\n   * @dev Set the value\n   * @param i The new value integer\n   * @param s The new value string\n   */\n  function set(int64 i, string memory s) public {\n    storedI = i;\n    storedS = s;\n    emit ChangeEvent(msg.sender, i, s, keccak256(bytes(s)), s);\n  }\n\n  /**\n   * @dev Get the value\n   */\n  function get() public view returns (int64 i, string memory s) {\n    return (storedI, storedS);\n  }\n}\n\n",
      "code": ""
    }
  }
}
```

🔗 销毁合约

销毁合约

接口描述 销毁已部署合约。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
DELETE /v1/quorum/contract?consortiumUuid={ConsortiumUuid}&orgUuid={OrgUuid}&contractUuid={ContractUuid}&ownerKey={OwnerKey} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	String	Query参数	联盟Uuid	必须
orgUuid	String	Query参数	组织Uuid	必须
contractUuid	String	Query参数	合约Uuid	必须
ownerKey	String	Query参数	合约发布者私钥	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
success	Bool	删除结果，true表示删除成功
result	String	合约Uuid
code	String	结果码

失败响应参考错误码部分。

示例

- 请求示例

```
DELETE /v1/quorum/contract?orgUuid=988ba131-e04b-8111-b44f-fb35b8dc80e6&contractUuid=3f90151d-a416-4ca5-99df-dcecf5bfb09e&consortiumUuid=cb73ac79-62bc-6070-aadc-24a05eaf55ae&ownerKey=1e2214d8ae56a48485e17b2c10af581c7dee7d3a9ef74b2182ccaef967cfdd3d HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例


```
200
Content-Length: 87
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "success": true,
  "result": "3f90151d-a416-4ca5-99df-dcecf5bfb09e",
  "code": ""
}
```

合约网关相关接口

🔗 接口说明

简介 Quorum是在以太坊(Ethereum)的基础上开发的，提供私有智能合约执行方案，并满足企业级的性能要求。基于Quorum开发Dapp，与智能合约交互通常使用web3，需要开发者管理Nonce、构建交易、签名交易、解析合约返回数据等，并且在调用过程中容易出现各种错误，没有很好的提示和处理机制，对开发人员来说，并不是特别友好。基于此，我们提供的Quorum合约网关旨在为用户提供企业级的合约管理服务，使用传统的RESTful API设计让应用开发人员聚焦于自身的业务逻辑和用户体验，将复杂的合约事物提交、Nonce管理等交由合约网关来处理。Quorum合约网关支持发起私有交易和公开交易、链上结果查询以及Event日志检索功能。

使用流程 Quorum合约网关使用流程如下图所示。



服务域名 Quorum合约网关接口服务域名地址详见BBE中所创建或加入的Quorum联盟网络详情。[查看Quorum联盟](#)

认证机制 Quorum合约网关接口采用Basic auth认证方式。其中用户名和密码详见所创建或加入的Quorum联盟。[查看Quorum联盟](#)

幂等性

当调用创建接口时如果遇到了请求超时或服务器内部错误，用户可能会尝试重发请求，这时用户通过clientToken参数避免创建出比预期要多的资源，即保证请求的幂等性。

幂等性基于clientToken，clientToken是一个长度不超过64位的ASCII字符串，通常放在query string里。

如果用户使用同一个clientToken值调用创建接口，则服务端会返回相同的请求结果。因此用户在遇到错误进行重试的时候，可以通过提供相同的clientToken值，来确保只创建一个资源。

clientToken的有效期为24小时，以服务端最后一次收到该clientToken为准。

🔗 调用合约方法接口

调用合约方法接口

□ □ □ □ □ □

接口描述

调用Quorum智能合约方法，包括call和transaction类型方法。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建的以太坊网络。[查看Quorum联盟](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回401 Unauthorized，错误信息：401 Authorization Required。

服务域名

Quorum合约网关接口服务域名地址<ContractGateWayHost>详见BBE中所创建或加入的Quorum联盟详情。[查看Quorum联盟](#)

请求结构

```
POST /v2/quorum/tx/{contract}/{funcName}?clientToken={clientToken} HTTP/1.1
Host: <ContractGateWayHost>
Content-Type: application/json
```

请求头域

头域	类型	说明	是否必须
Content-Type	String	值为application/json	必须

请求参数

名称	类型	位置	描述	是否必须
contract	String	URL参数	合约地址	必须
funcName	String	URL参数	合约方法名称	必须
clientToken	String	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串	非必须
from	String	RequestBody参数	合约调用者地址	必须
jsonrpcUrl	String	RequestBody参数	节点JSON-RPC地址	选填，私有交易必填，公开交易可为空
callbackUrl	String	RequestBody参数	transaction类型交易，上链成功后回调地址	选填
privateKey	String	RequestBody参数	合约调用者地址私钥	选填，transaction类型必填，call类型可为空
privateFor	Array	RequestBody参数	私有合约相关节点公钥	必须，公开合约值为空
invokeParameters	JsonObject	RequestBody参数	合约方法参数	选填，根据合约方法所需参数填写
optConfigs	OptConfigs	RequestBody参数	合约通用参数，可选包括gasLimit、nonce、ethValue	选填

使用限制

- 支持合约方法输入参数: intX/uintX, string, address, array, slice, bytes, bytes32
- 支持合约方法输出参数: intX/uintX, string, address, slice, bytes, bytes32

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
requestId	String	用户请求Id
result	JsonObject	合约方法的返回数据
txId	String	transaction类型请求时返回的交易Hash

Transaction类型交易请求示例

- 说明：
- optConfigs中的参数为选填，如不指定或值为0，合约网关将根据合约方法计算nonce、gas。
 - callbackUrl回调参数为transaction类型交易上链成功后，合约网关会将上链信息以POST请求形式推送给回调地址，推送信息详见[交易结果查询接口](#)

```
POST /v2/quorum/tx/0x5c695f5b6c14d4d59b1b2f65a7fb66f02fe15469/setValue?clientToken=562dcec4-61c7-4bd5-932e-673861022f8b HTTP/1.1
Host: 106.12.189.XXX:8080
Content-Type: application/json

{
  "from": "0x04c512DEA0943e27D036EA5550987d0fF8b48a60",
  "privateKey": "3BAB234C5FC756C317BB1CF12F3F5C*****114B0F2B2033BDBA41AA22F6734E4",
  "jsonrpcUrl": "http://106.12.189.XXX:8081/quorum-jsonrpc/ba9f7aad-2f49-4aab-8bd2-12d7ab009937/6d398463",
  "callbackUrl": "http://xxx",
  "privateFor":
  ["DMAxFR47Q8oli4Qdb+IM6iepOkxFqz5XMNsZgiFhbEY=", "pRpHnO/G+9tRlCfiy/OuSXuNRWy9J+d22f7I05W6LHQ="],
  "invokeParameters": {
    "i": 111,
    "s": "baidu"
  },
  "optConfigs": {
    "ethValue": 0,
    "gasLimit": 0,
    "nonce": 0
  }
}
```

Call类型交易请求示例

```
POST /v2/quorum/tx/0x5c695f5b6c14d4d59b1b2f65a7fb66f02fe15469/getValue HTTP/1.1
Host: 106.12.189.XXX:8080
Content-Type: application/json

{
  "from": "0x04c512DEA0943e27D036EA5550987d0fF8b48a60",
  "jsonrpcUrl": "http://106.12.189.XXX:8081/quorum-jsonrpc/ee7aa8d8-ac62-4824-9e2d-585c5ac54b17/249f2b8a",
  "privateFor":
  ["DMAxFR47Q8oli4Qdb+IM6iepOkxFqz5XMNsZgiFhbEY=", "pRpHnO/G+9tRlCfiy/OuSXuNRWy9J+d22f7I05W6LHQ="],
  "invokeParameters": {}
}
```

Transaction类型请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 09:22:16 GMT
Content Type: application/json; charset=utf-8
Content-Length: 157

{
  "requestId": "7b4faa25-d74f-49aa-9299-8de6981dec59",
  "result": {
    "txId": "0x9efd1bae01055189f067d7f0a0095f9626672894f5132bc4573abb3728dde026"
  }
}
```

说明：根据txId获取交易上链结果详见：[交易结果查询接口](#)

Call类型请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 09:46:00 GMT
Content Type: application/json; charset=utf-8
Content-Length: 107

{
  "requestId": "91c0ed8b-69c8-42c2-8daf-6f8c23e29000",
  "result": {
    "i": 111,
    "s": "baidu"
  }
}
```

交易结果查询接口

交易结果查询接口

接口描述

根据交易Hash，查询交易的上链结果。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建或加入的Quorum联盟。[查看Quorum联盟](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回401 Unauthorized，错误信息：401 Authorization Required。

服务域名

以太坊合约网关接口服务域名地址 <ContractGateWayHost> 详见BBE中所创建或加入的Quorum联盟。[查看Quorum联盟](#)

请求结构

```
GET /v2/quorum/tx/{txId}
Host: <ContractGateWayHost>
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

名称	类型	位置	描述	是否必须
txId	String	URL参数	交易Hash	必须

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	位置	描述
requestId	String	用户请求携带的X-Request-Id
result	JsonObject	交易结果数据
status	Integer	交易状态，1表示成功，0表示失败
from	String	合约调用者地址
contract	String	合约地址
ethValue	Integer	向合约发送的以太币数量
gas	Integer	交易消耗的gas
transactionHash	String	交易hash
nonce	Integer	nonce值
blockNumber	Integer	交易所在区块高度
currBlockNumber	Integer	当前区块高度
blockHash	String	区块Hash
logs	Array	交易输出日志
v	Integer	交易类型标识，37、38表示私有交易

请求示例

```
GET /v2/quorum/tx/0xd7752833e7da2e56dbb52075bbe931e38a59967f43bb83db3a49320feec4e109 HTTP/1.1
Host: 106.12.189.XXX:8080
```

交易已打包到区块中响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 09:55:34 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 518

{
  "requestId": "aaaac3f4-1931-4670-9c7f-d56a898483b1",
  "result": {
    "status": 1,
    "from": "0x04c512DEA0943e27D036EA5550987d0fF8b48a60",
    "contract": "0x5C695f5b6c14D4D59b1B2F65A7fB66f02fe15469",
    "ethValue": 0,
    "gas": 0,
    "transactionHash": "0x9a91045a3841e315229c9e36d10d86b882f709fd1bf889875a179cf43b218eb7",
    "nonce": 80,
    "blockNumber": 34645,
    "blockHash": "0x55d3cc65f7d0eaf4444e5bbe1083fb31c6a1aff7129742703a18f58f7f856f22",
    "currBlockNumber": 34738,
    "logs": [],
    "v": 37
  }
}
```

交易在区块中不存在响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 09:56:24 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 73

{
  "requestId": "27161565-71ff-470f-9010-6808bb2cdf17",
  "result": {}
}
```

调用合约方法详见：[调用合约方法接口](#)

🔗 Event日志检索接口

Event日志检索接口

□ □ □ □ □ □

接口描述

Event（事件）触发后形成日志记录在区块里，本接口提供根据区块范围和Event主题对日志进行检索功能。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建或加入的Quorum联盟。[查看Quorum联盟](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回401 Unauthorized，错误信息：401 Authorization Required。

服务域名

Quorum合约网关接口服务域名地址<ContractGateWayHost>详见BBE中所创建或加入的Quorum联盟。[查看Quorum联盟](#)

请求结构

```
POST /v2/quorum/log/{contract}/{eventName} HTTP/1.1
Host: <ContractGateWayHost>
Content-Type: application/json
```

请求头域

头域	类型	说明	是否必须
Content-Type	String	值为application/json	必须

请求参数

名称	类型	位置	描述	是否必须
contract	String	URL参数	合约地址	必须
eventName	String	URL参数	合约事件名称	必须
fromBlock	Integer	RequestBody参数	起始区块高度，默认为0	非必须
toBlock	Integer	RequestBody参数	结束区块高度，默认为最新区块高度	非必须
jsonrpcUrl	String	RequestBody参数	节点JSON-RPC地址	选填，私有交易必填，公开交易选填
invokeParameters	JsonObject	RequestBody参数	根据指定合约事件topic过滤，如未指定，则默认返回全部	选填

使用限制

- 事件检索支持主题参数类型: intX/uintX, string, address, bytes, bytes32
- 使用indexed修饰的事件主题输出不支持string和动态类型数组

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
requestId	String	用户请求Id
result	Array	日志检索返回结果
contract	String	合约地址
data	JsonObject	检索的日志信息，包括indexed修饰的topics数据和未修饰的data数据
blockNumber	Integer	发生本事件的区块高度
blockHash	String	发生本事件的区块哈希
txHash	String	发生本事件的交易哈希
txIndex	Integer	发生本事件的交易所在区块交易的位置
index	Integer	本日志在收据中的位置
removed	Bool	日志是否被回退

请求示例

说明：Quorum支持对indexed修饰的topic进行多条件过滤，即指定多个topic关键字则会按照多个关键字进行过滤。如本请求示例中使用address类型的from和int64类型的i进行日志检索。

```
POST /v2/quorum/log/0x5c695f5b6c14d4d59b1b2f65a7fb66f02fe15469/ChangeEvent HTTP/1.1
Host: 106.12.189.XXX:8080
Content-Type: application/json

{
  "fromBlock":0,
  "toBlock":123439,
  "jsonrpcUrl":"http://106.12.189.XXX:8081/quorum-jsonrpc/ba9f7aad-2f49-4aab-8bd2-12d7ab009937/6d398463",
  "invokeParameters":{
    "from":["0x04c512DEA0943e27D036EA5550987d0ff8b48a60"]
  }
}
```

请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 10:07:15 GMT
Content Type: application/json; charset=utf-8
Content-Length: 1412
{
  "requestId": "9e7002d3-247d-4686-9555-4e07f81f3a67",
  "result": [
    {
      "contract": "0x5c695f5b6c14d4d59b1b2f65a7fb66f02fe15469",
      "data": {
        "from": "0x04c512dea0943e27d036ea5550987d0ff8b48a60",
        "i": 111,
        "s": "baidu"
      },
      "blockNumber": 34341,
      "txHash": "0x9efd1bae01055189f067d7f0a0095f9626672894f5132bc4573abb3728dde026",
      "txIndex": 0,
      "blockHash": "0xedff2a83e915169f115bf3bc90d37276411c20df1aa9c2a8dd218d74983646c1",
      "index": 0,
      "removed": false
    },
    {
      "contract": "0x5c695f5b6c14d4d59b1b2f65a7fb66f02fe15469",
      "data": {
        "from": "0x04c512dea0943e27d036ea5550987d0ff8b48a60",
        "i": 222,
        "s": "bbe"
      },
      "blockNumber": 34783,
      "txHash": "0xf8fbac3d191d4f3e9748c398886a51d06aadab9c84220180c09a8118031e96df",
      "txIndex": 0,
      "blockHash": "0xe76c51789b84ed9b9a8ff7707957dc34980da0b695ea042d930dad7c2aab86b2",
      "index": 0,
      "removed": false
    }
  ]
}
```


Event监听注册接口

□□□□□□

接口描述

Event（事件）触发后形成日志记录在区块里，通过对Event的监听，可以掌握交易的上链确认情况。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建的Quorum网络。[查看Quorum网络](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回 401 Unauthorized，错误信息：401 Authorization Required。

服务域名 Quorum合约网关接口服务域名地址<ContractGateWayHost>详见BBE中所创建的Quorum网络详情。[查看Quorum网络](#)

请求结构

```
POST /v2/quorum/event/{contract}/{eventName}?clientToken={clientToken} HTTP/1.1
Host: <ContractGateWayHost>
Content-Type: application/json
```

请求头域

头域	说明	是否必须
Content-Type	值为application/json	必须

请求参数

名称	类型	位置	描述	是否必须
contract	String	URL参数	合约地址	必须
eventName	String	URL参数	需要监听的事件名称	必须
clientToken	String	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串	非必须
jsonrpcUrl	String	RequestBody参数	节点JSON-RPC地址，隐私交易必选，公开交易非必选	选填
subscribeUrl	String	RequestBody参数	接收事件消息地址	必须
invokeParameters	JsonObject	RequestBody参数	根据指定合约事件topic过滤，如未指定，则默认返回全部	非必须

使用限制

- 事件监听支持主题参数类型: intX/uintX, string, address, bytes, bytes32
- 使用indexed修饰的事件主题输出不支持string和动态类型数组
- 同一联盟组织最多支持注册20个Event监听

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
requestId	String	用户请求携带的X-Request-Id
result	JsonObject	请求返回数据
subscribelId	String	事件监听ID

请求示例

说明：

- Quorum Event支持对indexed修饰的topic进行多条件过滤，即指定多个topic关键字则会按照多个关键字进行过滤。如本请求示例中，对隐私交易使用address类型的from和int64类型的i进行事件日志过滤，只有触发的事件同时满足过滤条件时，事件数据才会被推送给用户的接收地址。
- 当监听的事件被触发时，推送事件内容详见：[交易结果查询接口](#)

```
POST /v2/quorum/event/0x5c695f5b6c14d4d59b1b2f65a7fb66f02fe15469/ChangeEvent?clientToken=25773871-af56-4878-8b3b-7f59de41f22c HTTP/1.1
Host: 106.12.189.XXX:8080
Content-Type: application/json

{
  "jsonrpcUrl": "http://106.13.189.xxx:8081/quorum-jsonrpc/a11e5052-a7ce-41dd-ae55-769bdbca5342/17440778",
  "subscribeUrl": "http://XXX",
  "invokeParameters": {
    "i": [111, 222]
  }
}
```

请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 10:46:35 GMT
Content Type: application/json; charset=utf-8
Content-Length: 134
{
  "requestId": "7a739a8e-dd26-48df-9119-1fa9ca19f71d",
  "result": {
    "subscribelId": "7a739a8e-dd26-48df-9119-1fa9ca19f71d"
  }
}
```

🔗 Event监听查询接口

Event监听查询接口

接口描述

本接口提供对已注册Event监听的查询。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建的Quorum网络。[查看Quorum网络](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回 401 Unauthorized，错误信息：401 Authorization Required。

服务域名 Quorum合约网关接口服务域名地址<ContractGateWayHost>详见BBE中所创建的Quorum网络详情。[查看Quorum网络](#)

请求结构

```
GET /v2/quorum/event?contract={contract}&subscribelid={subscribelid} HTTP/1.1
Host: <ContractGateWayHost>
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

名称	类型	位置	描述	是否必须
subscribelid	String	Query参数	事件监听ID	选填
contract	String	Query参数	事件所在合约地址	选填

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
requestId	String	用户请求携带的X-Request-Id
result	JsonObject	事件监听注册查询结果
count	Integer	事件监听注册个数
subscribeList	Array	事件监听注册列表
<i>subscribelid</i>	String	事件监听注册ID
<i>contract</i>	String	事件所在合约地址
<i>subscribeUrl</i>	String	用户接收事件数据URL
<i>invokeDescription</i>	InvokeDescription	事件监听描述
<i>name</i>	String	事件名称
<i>params</i>	Array	事件监听过滤条件
<i>subscribeTime</i>	String	事件监听注册时间

请求示例

说明：可根据事件订阅ID和事件所在合约地址进行检索过滤，如下方请求示例中，使用合约地址所以查询条件。

GET /v2/quorum/event?contract=0x5c695f5b6c14d4d59b1b2f65a7fb66f02fe15469 HTTP/1.1
Host: 106.12.189.XXX:8080

请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 10:47:41 GMT
Content Type: application/json; charset=utf-8
Content-Length: 562
{
  "requestId": "2825fcac-55c5-4eb1-8be6-758185ba523a",
  "result": {
    "count": 1,
    "subscribeList": [
      {
        "subscribeId": "7a739a8e-dd26-48df-9119-1fa9ca19f71d",
        "contract": "0x5C695f5b6c14D4D59b1B2F65A7fB66f02fe15469",
        "subscribeUrl": "http://XXX",
        "invokeDescription": {
          "name": "ChangeEvent",
          "params": {
            "i": [
              111,
              222
            ]
          }
        },
        "subscribeTime": "2019-09-19T10:46:35.177469455Z"
      }
    ]
  }
}
```

🔗 Event监听注销接口

Event监听注销接口

□ □ □ □ □ □

接口描述

本接口提供对已注册Event监听的注销。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建的Quorum网络。[查看Quorum网络](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回401 Unauthorized，错误信息：401 Authorization Required。

服务域名 Quorum合约网关接口服务域名地址<ContractGateWayHost>详见BBE中所创建的Quorum网络详情。[查看Quorum网络](#)

请求结构

```
DELETE /v2/quorum/event/{subscribeId} HTTP/1.1
Host: <ContractGateWayHost>
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

名称	类型	位置	描述	是否必须
subscribeId	String	URL参数	事件监听ID	必须

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
requestId	String	用户请求Id
result	String	事件监听注销结果

请求示例

```
DELETE /v2/quorum/event/7a739a8e-dd26-48df-9119-1fa9ca19f71d HTTP/1.1
Host: 106.12.189.XXX:8080
```

请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 10:52:29 GMT
Content Type: application/json; charset=utf-8
Content-Length: 77
{
  "requestId": "b35bcd26-5021-49f8-ac54-046f565f5b20",
  "result": "Done"
}
```

Ethereum相关接口

获取以太坊网络部署列表

 获取以太坊网络部署列表

接口描述 获取用户部署的所有以太坊网络列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/ethereum/list
Host: bbe.bj.baidubce.com
Date: <Date>
Authorization: <Authorization_String>
```

- 请求头域 除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	参数位置	描述	是否必须
marker	String	Query参数	批量获取列表的查询的起始位置，是一个由系统生成的字符串	否
maxKeys	Integer	Query参数	每页包含的最大数量，最大数量通常不超过1000，缺省值为1000	否
status	String	Query参数	bbe状态,状态间"," 分割,可以不填写,则返回前端显示状态	否

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	Integer	每页包含的最大数量
result	[]NetworkInfo	网络详情列表，结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/ethereum/list HTTP/1.1
content-type: application/json
host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 6171
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 10:24:02 GMT

{
  "nextMarker": "-1",
  "marker": "-1",
  "maxKeys": 1000,
  "isTruncated": false,
  "result": [
    {
      "blockchainUuid": "1f7833ed-e378-4108-9f71-9d110386b298",
      "clusterUuid": "c-QOUyL2yh",
      "clusterName": "lql",
      "accountId": "00dc1b52d8354d9193536e4dd2c41ae6",
      "networkUuid": "2b031568-5073-4967-bc86-bff6a242cd11",
      "networkName": "test-lql",
      "contractUuid": "",
      "chaincodeUuid": "",
      "networkType": null,
      "dappUuid": "",
      "taskUuid": "1a204bf5-f0c7-4064-85bb-3b199311fd4b",
      "createTime": 1546500733000,
      "deleteTime": 31536001000,
      "updateTime": null,
      "clusterStatus": "RUNNING",
      "taskStatus": "DONE",
      "channels": "",
      "remixUrl": null,
      "netstatsAddress": "http://10.107.245.134:8000",
      "explorerAddress": null
    }
  ]
}
```

查询以太坊网络

查询以太坊网络

接口描述 根据网络ID查询以太坊网络详情。

权限说明 请求发起人需要具有合法的AccessKeyId和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/ethereum/network/{networkUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	参数位置	描述	是否必须
networkUuid	String	URL参数	网络ID	必须

响应（Response）

- 响应头域

除公共头域外，无其它特殊头域。

- 响应参数

名称	类型	描述
networks	[]EthereumNetworkInfo	以太坊网络详情列表，结构具体定义见 附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/ethereum/network/6c8054dd-f2e0-b6df-3c1c-12e886759888 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 2572
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "name": "eth-test",
    "uuid": "6c8054dd-f2e0-b6df-3c1c-12e886759888",
    "networkSize": 2,
    "chainId": 1574307081,
    "targetGasLimit": 3000000,
    "maxPeers": 2,
    "verbosity": 0,
    "syncMode": "full",
    "gasPrice": 0,
    "difficulty": 1024,
    "gasLimit": 4294967295,
    "genesisAccounts": [
      {
        "address": "0xf8f3682DeFfE2E106dd700d78Fb883Dd3423fb84",
        "privateKey":
"xhEN1IYkqDXPXWtUPvhl3q4sWi80vTc3cYbkCf9kgJiDtYkTMjoOnoH25vKUd0426EuglQer+612axjvJYnL9uk7MAks2k9H6St",
        "balance": 1000000000000000000
      },
      {
        "address": "0xd8D06bCCC44F4D15B4dEf80Ca5af554670F982bf",
        "privateKey":
"SN8p1iy2q2BFnXNQPPJbKuEN8VH4b02eKFHKfME2vUfPQqdYTMOLHaR4VKKdkQJ1XOHL9GAuPmqfnYFatENz31EyZaeol9A",
        "balance": 1000000000000000000
      }
    ]
  }
}
```



```

    }
  ],
  "minerAccount": [
    {
      "address": "0x16ddaaD43bCe97212e6E4e2E8Af75307D9dAE4D6",
      "privateKey":
"jbF117h/7W8byjp7pnDQQQUEnvIG9hsMOB+ODEHggqB9gk52FYgpYOK7goZXlqBT+dlgT7Yri4orvDGM1vw/ADGWksZW5Sstj
",
      "balance": 10000000000000000000
    }
  ],
  "accounts": [
    {
      "address": "0xf2aD836E1d63778E14DF6a7A3d022Bd102a3B370",
      "privateKey":
"QuhLS4KBJp7mi766s1cW6L7Zcl2/wO6zMwkft4J9uF47lhg2L2QYozvtwSeS/GFj8+WHX/YquvBkwVODC5vazEI9diXHvORoC
",
      "balance": -1
    }
  ],
  "nodeList": {
    "gethNodes": [
      {
        "name": "node-1",
        "ip": "172.18.235.90",
        "jsonrpcPort": 8545,
        "wsrpcPort": 8546,
        "ipcListenPort": 30303,
        "ipcDiscoveryPort": 30303,
        "role": "miner",
        "desp": ""
      },
      {
        "name": "node-2",
        "ip": "172.18.106.208",
        "jsonrpcPort": 8545,
        "wsrpcPort": 8546,
        "ipcListenPort": 30303,
        "ipcDiscoveryPort": 30303,
        "role": "miner",
        "desp": ""
      }
    ]
  },
  "state": "RUNNING",
  "startTime": "2019-11-21T11:31:22+08:00",
  "netstatsAddress": "http://106.13.107.203:8084",
  "remixAddress": "http://106.13.107.203:8083/#endpoint=http://106.13.107.203:8081/eth-jsonrpc/6c8054dd-f2e0-b6df-3c1c-12e886759888&region=bd",
  "explorerAddress": "http://106.13.107.203:8082?uuid=6c8054dd-f2e0-b6df-3c1c-12e886759888",
  "jsonrpcAddress": "http://106.13.107.203:8081/eth-jsonrpc/6c8054dd-f2e0-b6df-3c1c-12e886759888",
  "deleted": false,
  "errorMessage": "",
  "UserId": "eca97e148cb74e9683d7b7240829d1ff"
}
}

```

删除以太坊网络

接口描述 根据网络ID删除以太坊网络。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
DELETE /v1/ethereum/network/{networkUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	参数位置	描述	是否必须
networkUuid	String	URL参数	网络ID	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。

- 响应参数 失败响应参考错误码部分。

示例

- 请求示例

```
DELETE /v1/ethereum/network/f5696c47-002d-40fe-89b7-8cc202181a7f HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872bfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 10:24:02 GMT
```

查询账户列表

查询账户列表

接口描述 查询指定网络下的账户列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/ethereum/account?networkUuid={NetworkUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	参数位置	描述	是否必须
networkUuid	String	Query参数	网络ID	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
name	String	钱包名称
address	String	以太坊账户地址
password	String	Keystore密码
passwordConfirm	String	Keystore确认密码
privateKey	String	以太坊账户私钥，需要调用 查询账户私钥接口 获取
balance	String	以太坊账户余额

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/ethereum/account?networkUuid=f5696c47-002d-40fe-89b7-8cc202181a7f HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 163
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "accounts": [{
      "name": null,
      "address": "0x42328C24028Bb92366039b9Cc78dcC320461Cd69",
      "password": null,
      "passwordConfirm": null,
      "privateKey": null,
      "balance": "1000000000000000000"
    }]
  }
}
```

创建新账户

创建新账户

接口描述 创建一个以太坊账户。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
POST /v1/ethereum/account HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	参数位置	描述	是否必须
networkUuid	String	Requestbody参数	网络ID	必须
balance	Long	Requestbody参数	初始余额	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
address	String	以太坊账户地址
txId	String	交易ID

失败响应参考错误码部分。

示例

- 请求示例

```
POST /v1/ethereum/account HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
{
  "balance": 1000,
  "networkUid": "c3414b84-80cc-4c11-9933-78631724aab5"
}
```

- 响应示例

```
200
Content-Length: 153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "address": "0x42328C24028Bb92366039b9Cc78dcC320461Cd69",
    "txId": "xxx"
  }
}
```

查询账户私钥

🔗 查询账户私钥

接口描述 查询以太坊账户对应的私钥。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/ethereum/account/privateKey?address={Address}&networkUid={NetworkUid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求参数

名称	类型	参数位置	描述	是否必须
address	String	Query参数	以太坊账户地址	必须
networkUid	String	Query参数	网络ID	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。

- 响应参数

名称	类型	描述
privateKey	String	以太坊账户私钥

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/ethereum/account/privateKey?
address=0x42328C24028Bb92366039b9Cc78dcC320461Cd69&networkUuid=f5696c47-002d-40fe-89b7-
8cc202181a7f HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-
06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "privateKey": "7746566e612dace4de5e143347bb33620d299ba7fd9159cd625d808b79a2c4ed"
  }
}
```

下载keystore



接口描述 下载一个账户的keystore

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求 (Request)

- 请求语法

```
GET /v1/ethereum/keystore?passphrase={Passphrase}&address={Address}&networkUuid={NetworkUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	参数位置	描述	是否必须
networkUuid	String	Query参数	网络ID	必须
address	String	Query参数	以太坊账户地址	必须
passphrase	String	Query参数	密码	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
address	String	以太坊账户地址
keystore	String	keystore

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/ethereum/keystore?
passphrase=123&address=0x42328C24028Bb92366039b9Cc78dcC320461Cd69&networkUuid=f5696c47-002d-
40fe-89b7-8cc202181a7f HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-
06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "address": "0x42328C24028Bb92366039b9Cc78dcC320461Cd69",
    "keystore": "xxx"
  }
}
```

删除账户



接口描述 删除一个以太坊账户。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
DELETE /v1/ethereum/account/{address}?networkUuid={NetworkUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	参数位置	描述	是否必须
address	String	URL参数	以太坊账户地址	必须
networkUuid	String	Query参数	网络ID	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数 失败响应参考错误码部分。

示例

- 请求示例

```
DELETE /v1/ethereum/account/0x42328C24028Bb92366039b9Cc78dcC320461Cd69?networkUuid=f5696c47-002d-40fe-89b7-8cc202181a7f HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 10:24:02 GMT
```

查询合约列表

🔗 查询合约列表

接口描述 查询指定网络下的通过合约编辑器部署的合约列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/ethereum/contracts?networkUuid={NetworkUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```


- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	描述	位置	是否必须
networkUuid	String	网络ID	Query参数	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
contractUuid	String	合约Uuid
networkUuid	String	网络Uuid
name	String	合约名称
type	String	合约类型，platform为平台基础库合约，user为用户自定义合约
address	String	合约地址
abi	String	合约abi
sourceCode	String	合约源代码
ownerAddress	String	合约发布者地址
status	String	合约状态，包含None Deployed、Deployed、Deploying和Deployment Failed四种，分别为未部署、运行中，部署中，部署失败
createTime	String	创建时间
updateTime	String	更新时间

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/ethereum/contracts?networkUuid=f5696c47-002d-40fe-89b7-8cc202181a7f HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T07:12:40Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-29T07:12:40Z/1800/host/d9dbd7c7f901daf2c1aa7fd71f162f5783c57d07c3e8b896bbf64b7ed968bd9b
```

- 响应示例

```
200
Content-Length: 1553
Content-Type: application/json; charset=utf-8
Date: Tue, 29 Oct 2019 07:14:27 GMT

{
  "data": {
    "contracts": [{
      "contractUuid": "12316185-7d21-4709-90bd-803c4afd7490",
      "networkUuid": "5c307678-99eb-9ff8-9626-c89c3f3d43e5",
      "name": "RBAC",
      "description": "基于角色的权限访问控制（Role-Based Access Control）作为传统访问控制（自主访问，强制访问）的有前景的代替受到广泛的关注。",
      "type": "platform",
      "address": "0x72C6A151B3045827ead039A6aa6789e6Cb7A301A",
      "abi": null,
      "sourceCode": null,
      "ownerAddress": "0x9896880e5815f2A64b64b9e1D239Be9E56b7d224",
      "status": "Deployed",
      "createdTime": "2019-10-29T15:14:18+08:00",
      "updatedAt": "2019-10-29T15:14:22+08:00"
    }],
    {
      "contractUuid": "aa16ff1a-9e66-42cd-9cd4-d196bbf7c14b",
      "networkUuid": "5c307678-99eb-9ff8-9626-c89c3f3d43e5",
      "name": "Simple",
      "description": "",
      "type": "user",
      "address": "0xf9b94c4c4e2e1be7d3f8d8d043229ffe88a6e6ae",
      "abi": null,
      "sourceCode": null,
      "ownerAddress": "0xf3cfd0f15cbde02252b15e19c74397c771c89267",
      "status": "Deployed",
      "createdTime": "2019-10-29T15:10:41+08:00",
      "updatedAt": "2019-10-29T15:10:41+08:00"
    }
  ]
}
```

获取合约详情

🔗 获取合约详情

接口描述 查询指定网络下的某合约详情。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/ethereum/contract?networkUuid={NetworkUuid}&contractUuid={ContractUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	描述	位置	是否必须
networkUuid	String	网络ID	Query参数	必须
contractUuid	String	合约Uuid	Query参数	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
contractUuid	String	合约Uuid
networkUuid	String	网络Uuid
name	String	合约名称
description	String	合约描述
type	String	合约类型，platform为平台基础库合约，user为用户自定义合约
address	String	合约地址
abi	String	合约abi
sourceCode	Object	合约源代码
ownerAddress	String	合约发布者地址
status	String	合约状态，包含None Deployed、Deployed、Deploying和Deployment Failed四种，分别为未部署、运行中，部署中，部署失败
createTime	String	创建时间
updateTime	String	更新时间

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/ethereum/contract?contractUuid=aa16ff1a-9e66-42cd-9cd4-d196bbf7c14b&networkUuid=5c307678-99eb-9ff8-9626-c89c3f3d43e5 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T07:12:40Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-29T07:12:40Z/1800/host/d9dbd7c7f901daf2c1aa7fd71f162f5783c57d07c3e8b896bbf64b7ed968bd9b
```

- 响应示例

```

200
Content-Length: 2312
Content-Type: application/json; charset=utf-8
Date: Tue, 29 Oct 2019 07:14:27 GMT

{
  "data": {
    "contractUuid": "aa16ff1a-9e66-42cd-9cd4-d196bbf7c14b",
    "networkUuid": "5c307678-99eb-9ff8-9626-c89c3f3d43e5",
    "name": "Simple",
    "description": "",
    "type": "user",
    "address": "0xf9b94c4c4e2e1be7d3f8d8d043229ffe88a6e6ae",
    "abi": "[{"constant":false,"inputs":[{"name":"i","type":"int64"},
{"name":"","s","type":"string"}],"name":"set","outputs":
[],"payable":false,"stateMutability":"nonpayable"},"anonymous":false,"inputs":
[{"indexed":true,"name":"from","type":"address"},{"indexed":true,"name":"i","type":"int64"},
{"indexed":true,"name":"s","type":"string"},{"indexed":false,"name":"h","type":"bytes32"},
{"indexed":false,"name":"m","type":"string"}],"name":"ChangeEvent","type":"event"},
{"constant":true,"inputs":[{"name":"","get","outputs":[{"name":"i","type":"int64"},
{"name":"","s","type":"string"}],"payable":false,"stateMutability":"view","type":"function"},
{"constant":true,"inputs":[{"name":"","storedl","outputs":
[{"name":"","","type":"int64"}],"payable":false,"stateMutability":"view","type":"function"},
{"constant":true,"inputs":[{"name":"","storedS","outputs":
[{"name":"","","type":"string"}],"payable":false,"stateMutability":"view","type":"function"}]",
    "sourceCode": {
      "browser/ballot_test.sol": "pragma solidity >=0.4.24;\n/**\n * @title Simple Storage with events\n * @dev Read
and write values to the chain\n */\ncontract Simple {\n  int64 public storedl;\n  string public storedS;\n\n  event
ChangeEvent (\n    address indexed from,\n    int64 indexed i,\n    string indexed s,\n    bytes32 h,\n    string m\n);\n\n  /**\n   * @dev Set the value\n   * @param i The new value integer\n   * @param s The new value string\n
*/\n  function set(int64 i, string memory s) public {\n    storedl = i;\n    storedS = s;\n    emit ChangeEvent(msg.sender,
i, s, keccak256(bytes(s)), s);\n  }\n\n  /**\n   * @dev Get the value\n   */\n  function get() public view returns (int64
i, string memory s) {\n    return (storedl, storedS);\n  }\n}"
    },
    "ownerAddress": "0xf3cfd0f15cbde02252b15e19c74397c771c89267",
    "status": "Deployed",
    "createdTime": "2019-10-29T15:10:41+08:00",
    "updatedTime": "2019-10-29T15:10:41+08:00"
  }
}

```

销毁合约

销毁合约

接口描述 查询指定网络下的某合约详情。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求 (Request)

请求语法

```

DELETE /v1/ethereum/contract?networkUuid={NetworkUuid}&contractUuid={ContractUuid}&ownerKey={OwnerKey}
HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>

```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	描述	位置	是否必须
networkUuid	String	网络ID	Query参数	必须
contractUuid	String	合约Uuid	Query参数	必须
ownerKey	String	合约部署者的私钥	Query参数	选填，用户自定义合约必填，平台基础库合约可为空

响应 (Response)

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数
无
失败响应参考错误码部分。

示例

- 请求示例

```
DELETE /v1/ethereum/contract?
ownerKey=35bdbbc608b03d404f5966123a4fb80183fa15cf42978b822c719163c8a05a551&contractUuid=aa16ff1a-
9e66-42cd-9cd4-d196bbf7c14b&networkUuid=5c307678-99eb-9ff8-9626-c89c3f3d43e5 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T07:12:40Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-
29T07:12:40Z/1800/host/d9dbd7c7f901daf2c1aa7fd71f162f5783c57d07c3e8b896bbf64b7ed968bd9b
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Tue, 29 Oct 2019 07:14:27 GMT
```

合约网关相关接口

🔗 接口说明

简介 开发以太坊Dapp，与智能合约交互通常使用web3，这种方式需要开发者管理Nonce、构建交易、签名交易、解析合约返回数据等，并且在调用过程中容易出现各种错误，没有很好的提示和处理机制，对开发人员来说，并不是特别友好。基于此，我们开发的以太坊合约网关旨在为用户提供企业级的合约管理服务，使用传统的RESTful API设计让应用开发人员聚焦于自身的业务逻辑和用户体验，将复杂的合约事物提交、Nonce管理等交由合约网关来处理。

使用流程 以太坊合约网关使用流程如下图所示。



服务域名 以太坊合约网关接口服务域名地址详见BBE中所创建的以太坊网络详情。[查看以太坊网络](#)

认证机制 以太坊合约网关接口采用Basic auth认证方式。 其中用户名和密码详见所创建的以太坊网络。[查看以太坊网络](#)

幂等性

当调用创建接口时如果遇到了请求超时或服务器内部错误，用户可能会尝试重发请求，这时用户通过clientToken参数避免创建出比预期要多的资源，即保证请求的幂等性。

幂等性基于clientToken，clientToken是一个长度不超过64位的ASCII字符串，通常放在query string里。

如果用户使用同一个clientToken值调用创建接口，则服务端会返回相同的请求结果。因此用户在遇到错误进行重试的时候，可以通过提供相同的clientToken值，来确保只创建一个资源。

clientToken的有效期为24小时，以服务端最后一次收到该clientToken为准。

交易结果查询接口

交易结果查询接口

接口地址

接口描述

根据交易Hash，查询交易的上链结果。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建的以太坊网络。[查看以太坊网络](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回 401 Unauthorized，错误信息：401 Authorization Required。

服务域名 以太坊合约网关接口服务域名地址 <ContractGateWayHost>详见BBE中所创建的以太坊网络详情。[查看以太坊网络](#)

请求结构

```
GET /v2/ethereum/tx/{txId}
Host: <ContractGateWayHost>
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

名称	类型	位置	描述	是否必须
txId	String	URL参数	交易Hash	必须

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
requestId	String	用户请求Id
result	JsonObject	交易结果数据
status	Integer	交易状态，1表示成功，0表示失败
from	String	合约调用者地址
contract	String	合约地址
ethValue	Integer	向合约发送的以太币数量
gas	Integer	交易消耗的gas
transactionHash	String	交易hash
nonce	Integer	nonce值
blockNumber	Integer	交易所在区块高度
blockHash	String	区块Hash
currBlockNumber	Integer	当前区块高度
logs	Array	交易输出日志

请求示例

```
GET /v2/ethereum/tx/0x208140b148622f260f354c38d9424722949cb1b31f542cd33125630dfa6c9809 HTTP/1.1
Host: 106.12.189.XXX:8080
```

交易已打包到区块中响应示例

[illegible]

交易在区块中不存在响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 07:59:04 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 91

{
  "requestId": "1bd2ed0d-8bfc-4d6f-ac7a-c550430a3ead",
  "result": {}
}
```

调用合约方法详见：[调用合约方法接口](#)

调用合约方法接口

调用合约方法接口

[illegible]

接口描述

调用以太坊智能合约方法，包括call和transaction类型方法。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建的以太坊网络。[查看以太坊网络](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回 401 Unauthorized，错误信息：401 Authorization Required。

服务域名 以太坊合约网关接口服务域名地址 <ContractGateWayHost>详见BBE中所创建的以太坊网络详情。[查看以太坊网络](#)

请求结构

```
POST /v2/ethereum/tx/{contract}/{funcName}?clientToken={clientToken} HTTP/1.1
Host: <ContractGateWayHost>
Content-Type: application/json
```

请求头域

头域	说明	是否必须
Content-Type	值为application/json	必须

请求参数

名称	类型	位置	描述	是否必须
contract	String	URL参数	合约地址	必须
funcName	String	URL参数	请求的合约方法	必须
clientToken	String	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串	非必须
from	String	RequestBody参数	合约调用者地址	必须
jsonrpcUrl	String	RequestBody参数	节点JSON-RPC地址	非必须
callbackUrl	String	RequestBody参数	交易成功回调地址	非必须
privateKey	String	RequestBody参数	合约调用者地址私钥	选填，call类型可为空，transaction类型必填
invokeParameters	JsonObject	RequestBody参数	合约方法参数	选填，根据合约方法所需参数填写
optConfigs	OptConfigs	RequestBody参数	合约通用参数，可选包括gasLimit、gasPrice、nonce、ethValue	选填

使用限制

- 支持合约方法输入参数: intX/uintX, string, address, array, slice, bytes, bytes32
- 支持合约方法输出参数: intX/uintX, string, address, slice, bytes, bytes32

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
requestId	String	用户请求ID
result	JsonObject	请求返回结果数据
txId	String	transaction类型请求时返回的交易Hash

Transaction类型请求示例

说明：

- optConfigs中的参数为可选，如不指定或值为0，合约网关将根据合约方法计算nonce、gasLimit和gasPrice。
- callbackUrl回调参数为transaction类型交易上链成功后，合约网关会将上链信息以POST请求形式推送给回调地址，推送信息详见[交易结果查询接口](#)

```
POST /v2/ethereum/tx/0x134bfd7a3d43ca3b4599718045156ada22ec8591/setValue?clientToken=8dbe1955-c616-4d60-97e8-e48bd2063906 HTTP/1.1
Host: 106.12.189.XXX:8080
Content-Type: application/json

{
  "from": "0x7244Ea287849942bAAC280B1912A575aEeCCc792",
  "privateKey": "3BAB234C5FC756C317BB1CF12F3F5C*****114B0F2B2033BDBA41AA22F6734E4",
  "jsonrpcUrl": "",
  "callbackUrl": "",
  "invokeParameters": {
    "i": 111,
    "s": "baidu"
  },
  "optConfigs": {
    "ethValue": 0,
    "gasLimit": 0,
    "gasPrice": 0,
    "nonce": 0
  }
}
```

Transaction类型请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 07:07:08 GMT
Content Type: application/json; charset=utf-8
Content-Length: 153

{
  "requestId": "47026c75-008f-4fe5-be1f-807f8244df6f",
  "result": {
    "txId": "0x208140b148622f260f354c38d9424722949cb1b31f542cd33125630dfa6c9809"
  }
}
```

说明：根据txId获取交易上链结果详见：[交易结果查询接口](#)

Call类型请求示例

```
POST /v2/ethereum/tx/0x134bfd7a3d43ca3b4599718045156ada22ec8591/getValue HTTP/1.1
Host: 106.12.189.XXX:8080
Content-Type: application/json

{
  "from": "0x7244Ea287849942bAAC280B1912A575aEeCCc792",
  "jsonrpcUrl": "",
  "invokeParameters": {}
}
```

Call类型请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 07:16:36 GMT
Content Type: application/json; charset=utf-8
Content-Length: 105

{
  "requestId": "a8739881-8eb0-411c-bbbb-5bd5cf0eba65",
  "result": {
    "i": 111,
    "s": "bbe"
  }
}
```

🔗 Event日志检索接口

Event日志检索接口

□ □ □ □ □ □

接口描述

Event（事件）触发后形成日志记录在区块里，本接口提供根据区块范围和Event主题对日志进行检索功能。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建的以太坊网络。[查看以太坊网络](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回401 Unauthorized，错误信息：401 Authorization Required。

服务域名 以太坊合约网关接口服务域名地址<ContractGateWayHost>详见BBE中所创建的以太坊网络详情。[查看以太坊网络](#)

请求结构

```
POST /v2/ethereum/log/{contract}/{eventName} HTTP/1.1
Host: <ContractGateWayHost>
Content-Type: application/json
```

请求头域

头域	说明	是否必须
Content-Type	值为application/json	必须

请求参数

名称	类型	位置	描述	是否必须
contract	String	URL参数	合约地址	必须
eventName	String	URL参数	所请求的事件名称	必须
fromBlock	Integer	RequestBody参数	起始区块高度，默认为0	非必须
toBlock	Integer	RequestBody参数	结束区块高度，默认为最新区块高度	非必须
invokeParameters	JsonObject	RequestBody参数	根据指定合约事件topic过滤，如未指定，则默认返回全部	选填

使用限制

- 事件检索支持主题参数类型: intX/uintX, string, address, bytes, bytes32
- 使用indexed修饰的事件主题输出不支持string和动态类型数组

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
requestId	String	用户请求携带的X-Request-Id
result	JsonObject	日志检索返回结果
<i>contract</i>	String	日志所在的合约地址
<i>data</i>	JsonObject	检索的日志信息，包括indexed修饰的topics数据和未修饰的data数据
<i>blockNumber</i>	Integer	发生本事件的区块高度
<i>blockHash</i>	String	发生本事件的区块哈希
<i>txHash</i>	String	发生本事件的交易哈希
<i>txIndex</i>	Integer	发生本事件的交易所在区块交易的位置
<i>index</i>	Integer	本日志在收据中的位置
<i>removed</i>	Bool	日志是否被回退

请求示例

说明：以太坊支持对indexed修饰的topic进行多条件过滤，即指定多个topic关键字则会按照多个关键字进行过滤。如本请求示例中使用address类型的from和int64类型的i进行日志检索。

```
POST /v2/ethereum/log/0x134bfd7a3d43ca3b4599718045156ada22ec8591/ChangeEvent HTTP/1.1
Host: 106.12.189.XXX:8080
Content-Type: application/json

{
  "fromBlock":0,
  "toBlock":123439,
  "invokeParameters":{
    "from":["0x7244ea287849942baac280b1912a575aecccc792"],
    "i":[111,222]
  }
}
```

请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 07:08:06 GMT
Content Type: application/json; charset=utf-8
Content-Length: 1007
{
  "requestId": "5d7b994a-9485-4d01-a2e1-3bc8f141db28",
  "result": [
    {
      "contract": "0x134bfd7a3d43ca3b4599718045156ada22ec8591",
      "data": {
        "from": "0x7244ea287849942baac280b1912a575aecccc792",
        "i": 222,
        "s": "baidu"
      },
      "blockNumber": 101467,
      "txHash": "0xa6791480e9546170965860c90e27ed6f17c19227734c58708956b34a8a6d5c8b",
      "txIndex": 0,
      "blockHash": "0xcaf81f10e0874a8490b85df95b223b04dd83f3054dd96989149df3e95fc03ee1",
      "index": 0,
      "removed": false
    },
    {
      "contract": "0x134bfd7a3d43ca3b4599718045156ada22ec8591",
      "data": {
        "from": "0x7244ea287849942baac280b1912a575aecccc792",
        "i": 111,
        "s": "bbe"
      },
      "blockNumber": 101467,
      "txHash": "0x208140b148622f260f354c38d9424722949cb1b31f542cd33125630dfa6c9809",
      "txIndex": 1,
      "blockHash": "0xcaf81f10e0874a8490b85df95b223b04dd83f3054dd96989149df3e95fc03ee1",
      "index": 1,
      "removed": false
    }
  ]
}
```

🔗 Event监听注销接口

Event监听注销接口

□□□□□□

接口描述

本接口提供对已注册Event监听的注销。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建的以太坊网络。[查看以太坊网络](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回 401 Unauthorized，错误信息：401 Authorization Required。

服务域名 以太坊合约网关接口服务域名地址 <ContractGateWayHost> 详见BBE中所创建的以太坊网络详情。[查看以太坊网络](#)

请求结构

```
DELETE /v2/ethereum/event/{subscribelId}?clientToken={clientToken} HTTP/1.1
Host: <ContractGateWayHost>
```

请求头域

除公共头域外，无其它特殊头域。

请求参数

名称	类型	位置	描述	是否必须
subscribelId	String	URL参数	事件监听ID	必须
clientToken	String	Query参数	幂等性Token，是一个长度不超过64位的ASCII字符串	非必须

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
requestId	String	用户请求携带的X-Request-Id
data	JsonObject	请求返回数据
result	String	事件监听注销结果

请求示例

```
DELETE /v2/ethereum/event/6cb77e8d-b21b-4d5d-a446-b858af5b26af?clientToken=47026c75-008f-4fe5-be1f-807f8244df6f HTTP/1.1
Host: 106.12.189.XXX:8080
```

请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 07:48:59 GMT
Content Type: application/json; charset=utf-8
Content-Length: 77
{
  "requestId": "434e0977-7114-4a5b-be30-a3db80374e43",
  "result": "Done"
}
```

Event监听注册接口

□□□□□□

接口描述

Event（事件）触发后形成日志记录在区块里，通过对Event的监听，可以掌握交易的上链确认情况。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建的以太坊网络。[查看以太坊网络](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回401 Unauthorized，错误信息：401 Authorization Required。

服务域名 以太坊合约网关接口服务域名地址 <ContractGateWayHost>详见BBE中所创建的以太坊网络详情。[查看以太坊网络](#)

请求结构

```
POST /v2/ethereum/event/{contract}/{eventName} HTTP/1.1
Host: <ContractGateWayHost>
Content-Type: application/json
```

请求头域

头域	说明	是否必须
Content-Type	值为application/json	必须

请求参数

名称	类型	位置	描述	是否必须
contract	String	URL参数	合约地址	必须
eventName	String	URL参数	Event名称	必须
subscribeUrl	String	RequestBody参数	接收事件消息地址	必须
invokeParameters	JsonObject	RequestBody参数	根据指定合约事件topic过滤，如未指定，则默认返回全部	可选

使用限制

- 事件监听支持主题参数类型: intX/uintX, string, address, bytes, bytes32
- 使用indexed修饰的事件主题输出不支持string和动态类型数组
- 同一以太坊网络最多支持注册20个Event监听

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
requestId	String	用户请求Id
result	JsonObject	请求返回数据
subscribeId	String	事件监听ID

请求示例

说明：

- 以太坊支持对indexed修饰的topic进行多条件过滤，即指定多个topic关键字则会按照多个关键字进行过滤。如本请求示例中使用address类型的from和int64类型的i进行事件日志过滤，只有触发的事件同时满足过滤条件时，事件数据才会被推送给用户的接收地址。
- 监听事件触发后，推送事件内容详见：[交易结果查询接口](#)

```
POST /v2/ethereum/event/0x134bfd7a3d43ca3b4599718045156ada22ec8591/ChangeEvent HTTP/1.1
Host: 106.12.189.XXX:8080
Content-Type: application/json

{
  "subscribeUrl": "http://XXX",
  "invokeParameters": {
    "i": [111, 222]
  }
}
```

请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 07:23:55 GMT
Content Type: application/json; charset=utf-8
Content-Length: 134
{
  "requestId": "fbd08f7f-2ee7-46d1-b210-cb2ede56030e",
  "result": {
    "subscribelId": "fbd08f7f-2ee7-46d1-b210-cb2ede56030e"
  }
}
```

🔗 Event监听查询接口

Event监听查询接口

接口描述

本接口提供对已注册Event监听的查询。

权限说明

API权限认证采用Basic Auth方式，请求发起人需要具有合法的Username和Password才能发起请求，其中Username和Password详见所创建的以太坊网络。[查看以太坊网络](#)

注意事项

如果请求中没有用户认证信息（即匿名访问），返回401 Unauthorized，错误信息：401 Authorization Required。

服务域名 以太坊合约网关接口服务域名地址<ContractGateWayHost>详见BBE中所创建的以太坊网络详情。[查看以太坊网络](#)

请求结构

```
GET /v2/ethereum/event?contract={contract}&subscribelId={subscribelId} HTTP/1.1
Host: <ContractGateWayHost>
```


请求头域

除公共头域外，无其它特殊头域。

请求参数

名称	类型	位置	描述	是否必须
subscribeId	String	Query参数	事件监听ID	选填
contract	String	Query参数	事件所在合约地址	选填

响应头域

除公共头域外，无其它特殊头域。

响应参数

名称	类型	描述
requestId	String	用户请求Id
result	JsonObject	事件监听注册查询结果
count	Integer	事件监听注册个数
subscribeList	Array	事件监听注册列表
<i>subscribeId</i>	String	事件监听注册ID
<i>contract</i>	String	事件所在合约地址
<i>subscribeUrl</i>	String	用户接收事件数据URL
<i>invokeDescription</i>	InvokeDescription	事件订阅描述
<i>name</i>	String	事件名称
<i>params</i>	Array	事件监听过滤条件
<i>subscribeTime</i>	String	事件监听注册时间

请求示例

说明：可根据事件订阅ID和事件所在合约地址进行检索过滤，如下方请求示例中，使用合约地址所以查询条件。

```
GET /v2/ethereum/event?contract=0x134BFd7a3D43ca3b4599718045156ADa22EC8591 HTTP/1.1
Host: 106.12.189.XXX:8080
```

请求响应示例

```
HTTP/1.1 200 OK
Date: Thu, 19 Sep 2019 07:38:37 GMT
Content Type: application/json; charset=utf-8
Content-Length: 1029
{
  "requestId": "a34272bb-652c-4d29-97e6-1a412fc12493",
  "result": {
    "count": 2,
    "subscribeList": [
      {
        "subscribeId": "fbd08f7f-2ee7-46d1-b210-cb2ede56030e",
        "contract": "0x134BFd7a3D43ca3b4599718045156ADa22EC8591",
        "subscribeUrl": "http://XXX",
        "invokeDescription": {
          "name": "ChangeEvent",
          "params": {
            "i": [
              111,
              222
            ]
          }
        },
        "subscribeTime": "2019-09-19T07:23:55.435083031Z"
      },
      {
        "subscribeId": "6cb77e8d-b21b-4d5d-a446-b858af5b26af",
        "contract": "0x134BFd7a3D43ca3b4599718045156ADa22EC8591",
        "subscribeUrl": "http://XXX",
        "invokeDescription": {
          "name": "ChangeEvent",
          "params": {
            "from": [
              "0x7244ea287849942baac280b1912a575aeeccc792"
            ]
          }
        },
        "subscribeTime": "2019-09-19T07:38:34.777865918Z"
      }
    ]
  }
}
```

fabric联盟链相关接口

组织管理

🔗 创建组织

创建组织

接口描述 创建组织。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
POST /v1/fabric/consortiums/orgs HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgName	String	Body参数	组织名称 支持大小写字母、数字、中文以及_/.特殊字符，不能以特殊字符开头，长度1-20	必须
englishName	String	Body参数	组织英文名称 支持小写字母、数字以及-，必须以字母开头，长度1-16	必须
description	String	Body参数	组织描述	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	{"orgUuid": "b1f47f52-497f-c66e-b93b-a37029427646"}

失败响应参考错误码部分。

示例

- 请求示例

```
POST /v1/fabric/consortiums/orgs HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
{
  "orgName": "test-org",
  "englishName": "test-org",
  "description": "test-org"
}
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "orgUuid": "b1f47f52-497f-c66e-b93b-a37029427646"
  }
}
```

🔗 查询组织详细信息

查询组织详细信息

接口描述 查询组织详细信息。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/orgs?orgUuid={orgUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query	组织Uuid	是

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"organization":organization}

organization:

名称	类型	描述
orgName	String	机构名称
englishName	String	机构英文名
consortiums	[]String	所在联盟列表
canDelete	Bool	是否可删除
uuid	String	唯一码
userId	String	用户ID
createTim	String	创建时间
previlige	String	特权
encryptionTpye	json	加密算法

示例

- 请求示例

```
GET /v1/fabric/consortiums/account/orgs?orgUuid=2cd9e3d2-438a-f579-ebef-46056b1cb10e HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "organization":{
      "canDelete": true,
      "consortiums": [],
      "createTime": "2019-10-30T12:28:58+08:00",
      "description": "d",
      "encryptionTpye": "ECDSA",
      "englishName": "d"
      "orgName": "d",
      "previlige": "admin",
      "uuid": "f5891400-5fb2-f2da-a08f-78dfab2dfaef",
      "userId": "25891400-5fb2-f2da-a08f-78dfab2dfaef"
    }
  }
}
```

 查询组织列表

查询组织列表

接口描述 查询组织列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/account/orgs HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数
无

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"organizations":[]organization}

organization:

名称	类型	描述
orgName	String	机构名称
englishName	String	机构英文名
consortiums	[]String	所在联盟列表
description	String	描述
uuid	String	唯一码
userId	String	用户ID
previlige	String	特权
canDelete	Bool	是否可删除
encryptionTpye	json	加密算法
createTime	String	创建时间

示例

- 请求示例

```
GET /v1/fabric/consortiums/account/orgs HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "organizations":[{"orgName": "zh-test",
      "englishName": "zh-test",
      "consortiums": [
        "esb-test"
      ],
      "description": "zh-test",
      "uuid": "14c293e0-8c6c-81cf-e507-467b73f8241c",
      "userId": "24c293e0-8c6c-81cf-e507-467b73f8241c",
      "previlige": "admin",
      "canDelete": false,
      "encryptionTpye": "ECDSA",
      "createTime": "2019-11-11T11:46:37+08:00"
    }]
  }
}
```

🔗 删除组织

删除组织

接口描述 删除组织。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
DELETE /v1/fabric/consortiums/orgs?orgUuid={orgUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	query参数	组织Uuid	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	{"status": "success"}

失败响应参考错误码部分。

示例

- 请求示例

```
DELETE /v1/fabric/consortiums/orgs?orgUuid=2cd9e3d2-438a-f579-ebef-46056b1cb10e HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "status": "success"
  }
}
```

联盟管理

邀请加入联盟

邀请加入联盟

接口描述 邀请加入联盟。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
POST /v1/fabric/consortiums/join/apply HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Body	组织Uuid	是
consortiumUuid	String	Body	联盟Uuid	是
objectUserId	String	Body	邀请的目标用户id	是

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。

● 响应参数

名称	类型	描述
data	json	返回结果：{"applicationUuid":"testId"}

示例

● 请求示例

```
POST /v1/fabric/consortiums/join/apply HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
{
  "orgUuid":"testid",
  "consortiumUuid":"testid",
  "objectUserId":"testid"
}
```

● 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "applicationUuid":"testid"
  }
}
```

🔗 查询某联盟通道列表

查询某联盟通道列表

接口描述 查询某联盟通道列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

● 请求语法

```
GET /v1/fabric/consortiums/channels?orgUuid={orgUuid}&consortiumUuid={consortiumUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

● 请求头域 除公共头域外，无其它特殊头域。

● 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query	组织Uuid	是
consortiumUuid	String	Query	联盟Uuid	是

响应 (Response)

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"channels":Channel}

Channel:

名称	类型	描述
channelName	String	通道名称
englishName	String	通道英文名称
networkUuid	String	私有链uuid
consortiumUuid	String	联盟Uuid
peerOrgUuidList	[]String	通道中peer组织uuid
orgs	[]String	通道中组织uuid列表
participants	[]Participant	通道中组织详情列表
state	String	状态
userId	String	用户ID
uuid	String	uuid
description	String	描述
createTime	String	创建时间
initiator	String	发起人

Participant:

名称	类型	描述
OrgUuid	String	组织uuid
OrgName	String	组织中文名
OrgEnglishName	String	组织英文名
Description	String	组织描述
JoinTime	String	加入通道时间

示例

- 请求示例

```
GET /v1/fabric/consortiums/channels?orgUid=2cd9e3d2-438a-f579-ebef-46056b1cb10e&consortiumUid=3b1a289b-da2f-1bc0-518e-8de037cd79e2 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data": {
    "channels": [
      {
        "channelName": "ym-channel",
        "englishName": "ym-channel",
        "networkUid": "",
        "consortiumUid": "764b436f-2772-3a2b-a94c-70bd39a2da20",
        "participants": [
          {
            "orgUid": "ec98e5b1-a381-b306-f0d1-66fe1f717d2a",
            "orgName": "ym-org",
            "orgEnglishName": "ym-org",
            "description": "desc",
            "joinTime": "2019-12-20T10:22:20+08:00"
          }
        ],
        "peerOrgUidList": [
          "ec98e5b1-a381-b306-f0d1-66fe1f717d2a"
        ],
        "orgs": [
          "ym-org"
        ],
        "description": "ym-channel",
        "userId": null,
        "uuid": "9f3f492e-c949-eda1-a94c-70bd39a2da20",
        "createTime": "2019-12-20T10:22:20+08:00",
        "state": "RUNNING",
        "initiator": "ec98e5b1-a381-b306-f0d1-66fe1f717d2a"
      }
    ]
  }
}
```

查询某联盟详细信息

查询某联盟详细信息

接口描述 查询某联盟详细信息。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/query?orgUuid={orgUuid}&consortiumUuid={consortiumUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query	组织Uuid	是
consortiumUuid	String	Query	联盟Uuid	是

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"consortium":Consortium}

Consortium:

名称	类型	描述
uuid	String	唯一ID
name	String	名称
channels	[]ChannelDetail	联盟中通道
orgs	[]OrgDetail	机构详情
ordererOrg	String	盟主组织
ordererSize	int	orderer数量
createTime	String	创建时间
state	String	状态
ordererAddress	String	orderer地址
UserId	[]String	用户ID
errorMessage	String	错误信息
channelPolicy	String	通道策略
englishName	String	英文名称
ownerPeerSize	int	盟主peer数量
deleted	Bool	是否可删除
description	String	描述
consensus	String	共识算法
spec	String	预留
type	String	预留

ChannelDetail :

名称	类型	描述
channelName	String	通道名称
deleted	Bool	是否删除
consortiumUuid	String	联盟Uuid
ordererOrgUuid	String	盟主组织Uuid
peerOrgUuidList	[]String	通道中peer组织uuid
state	String	状态
userId	String	用户ID
uuid	String	uuid
description	String	描述
createTime	String	创建时间
initiator	String	发起人

OrgDetail:

名称	类型	描述
orgName	String	机构名称
deleted	Bool	是否删除
userId	String	用户ID
uuid	String	uuid
englishName	String	英文名称
createTime	String	创建时间
joinTime	String	加入时间
size	String	size
type	[]String	类型
role	String	角色
forConsortium	Bool	是否为联盟链组织
description	String	描述

示例

- 请求示例

```
GET /v1/fabric/consortiums/query?orgUuid=2cd9e3d2-438a-f579-ebef-46056b1cb10e&consortiumUuid=3b1a289b-da2f-1bc0-518e-8de037cd79e2 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
```

```

"data":{
  "consortium":{
    "channelPolicy":"any",
    "channels":[
      {
        "channelName":"mychannel",
        "consortiumUuid":"1888b534-a2df-e63d-7ecf-4b5cf381d7f7",
        "createTime":"2019-10-30T17:51:01+08:00",
        "description":"d",
        "initiator":"f5891400-5fb2-f2da-a08f-78dfab2dfaef",
        "networkUuid":"",
        "orgs":[
          "d"
        ],
        "peerOrgUuidList":[
          "f5891400-5fb2-f2da-a08f-78dfab2dfaef"
        ],
        "state":"RUNNING",
        "userId":null,
        "uuid":"21437b42-b68d-94a2-7ffd-e3a63220692f"
      }
    ],
    "consensus":"kafka",
    "createTime":"2019-10-30T17:05:47+08:00",
    "deleted":false,
    "description":"first",
    "englishName":"first",
    "errorMessage":null,
    "name":"first",
    "ordererAddress":"orderer0.1888b534-a2df-e63d-7ecf-4b5cf381d7f7",
    "ordererOrg":"f5891400-5fb2-f2da-a08f-78dfab2dfaef",
    "ordererSize":4,
    "orgs":[
      {
        "ccIdeUrl":"http://106.13.113.83:8081",
        "createTime":null,
        "deleted":null,
        "description":null,
        "englishName":"d",
        "explorerUrl":"",
        "forConsortium":null,
        "joinTime":"2019-10-30T17:05:47+08:00",
        "orgName":"d",
        "role":"OWNER",
        "size":"2",
        "type":"ON_CLOUD",
        "userId":"3cec085510884e4488a431c426eac95f",
        "uuid":"f5891400-5fb2-f2da-a08f-78dfab2dfaef"
      }
    ],
    "ownerPeerSize":null,
    "spec":null,
    "state":"RUNNING",
    "type":"Internal",
    "userId":"3cec085510884e4488a431c426eac95f",
    "uuid":"1888b534-a2df-e63d-7ecf-4b5cf381d7f7"
  }
}

```

查询某联盟组织列表

接口描述 查询某联盟组织列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/orgs/list?orgUuid={orgUuid}&consortiumUuid={consortiumUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query	组织Uuid	是
consortiumUuid	String	Query	联盟Uuid	是

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"organizations":Organization}

Organization:

名称	类型	描述
orgName	String	机构名称
englishName	String	机构英文名
consortiums	[]String	所在联盟列表
canDelete	Bool	是否可删除
uuid	String	唯一码
createTim	String	创建时间
previlige	String	特权
encryptionTpye	json	加密算法

示例

- 请求示例

```
GET /v1/fabric/consortiums/orgs/list?orgUid=2cd9e3d2-438a-f579-ebef-46056b1cb10e&consortiumUid=3b1a289b-da2f-1bc0-518e-8de037cd79e2 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

● 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "organizations":[{
      "canDelete": true,
      "consortiums": [],
      "createTime": "2019-10-30T12:28:58+08:00",
      "description": "d",
      "encryptionTpye": "ECDSA",
      "englishName": "d"
      "orgName": "d",
      "previlige": "admin",
      "uuid": "f5891400-5fb2-f2da-a08f-78dfab2dfaef"
    }]
  }
}
```

🔗 查看联盟列表

查看联盟列表

接口描述 查看联盟列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

● 请求语法

```
GET /v1/fabric/consortiums/list?orgUid={orgUid}&page={page}&perPage={perPage} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

● 请求头域 除公共头域外，无其它特殊头域。

● 请求参数

名称	类型	位置	描述	是否必须
orgUid	String	Query	组织Uuid	是
page	int	Query	页数，用于分页	是
perPage	int	Query	每页返回的联盟条数	是

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"consortiums":[]Consortium}

Consortium:

名称	类型	描述
uuid	String	唯一ID
name	String	名称
channels	[]ChannelDetail	联盟中通道
orgs	[]OrgDetail	机构详情
ordererOrg	String	盟主组织
ordererSize	int	orderer数量
createTime	String	创建时间
state	String	状态
ordererAddress	String	orderer地址
UserId	[]String	用户ID
errorMessage	String	错误信息
channelPolicy	String	通道策略
englishName	String	英文名称
ownerPeerSize	int	盟主peer数量
deleted	Bool	是否可删除
description	String	描述
consensus	String	共识算法
spec	String	预留
type	String	预留

ChannelDetail :

名称	类型	描述
channelName	String	通道名称
deleted	Bool	是否删除
consortiumUuid	String	联盟Uuid
ordererOrgUuid	String	盟主组织Uuid
peerOrgUuidList	[]String	通道中peer组织uuid
state	String	状态
userId	String	用户ID
uuid	String	uuid
description	String	描述
createTime	String	创建时间
initiator	String	发起人

OrgDetail:

名称	类型	描述
orgName	String	机构名称
deleted	Bool	是否删除
userId	String	用户ID
uuid	String	uuid
englishName	String	英文名称
createTime	String	创建时间
joinTime	String	加入时间
size	String	size
type	[]String	类型
role	String	角色
forConsortium	Bool	是否为联盟链组织
description	String	描述
ccldeUrl	String	编辑器地址
explorerUrl	String	浏览器地址

示例

- 请求示例

```
GET /v1/fabric/consortiums/list?orgUuid=2cd9e3d2-438a-f579-ebef-46056b1cb10e&page=12&perPage=10
HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
```

```
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "consortiums":[
      {
        "channelPolicy":"any",
        "channels":[
          {
            "channelName":"mychannel",
            "consortiumUuid":"1888b534-a2df-e63d-7ecf-4b5cf381d7f7",
            "createTime":"2019-10-30T17:51:01+08:00",
            "description":"d",
            "initiator":"f5891400-5fb2-f2da-a08f-78dfab2dfaef",
            "networkUuid":"",
            "orgs":[
              "d"
            ],
            "peerOrgUuidList":[
              "f5891400-5fb2-f2da-a08f-78dfab2dfaef"
            ],
            "state":"RUNNING",
            "userId":null,
            "uuid":"21437b42-b68d-94a2-7ffd-e3a63220692f"
          }
        ],
        "consensus":"kafka",
        "createTime":"2019-10-30T17:05:47+08:00",
        "deleted":false,
        "description":"first",
        "englishName":"first",
        "errorMessage":"",
        "name":"first",
        "ordererAddress":"orderer0.1888b534-a2df-e63d-7ecf-4b5cf381d7f7",
        "ordererOrg":"f5891400-5fb2-f2da-a08f-78dfab2dfaef",
        "ordererSize":4,
        "orgs":[
          {
            "ccldeUrl":"http://106.13.113.83:8081",
            "createTime":"2019-10-30T12:28:58+08:00",
            "deleted":false,
            "description":"d",
            "englishName":"d",
            "explorerUrl":"http://106.12.48.46:8080",
            "forConsortium":true,
            "joinTime":null,
            "orgName":"d",
            "role":null,
            "size":null,
            "type":"",
            "userId":"3cec085510884e4488a431c426eac95f",
            "uuid":"f5891400-5fb2-f2da-a08f-78dfab2dfaef"
          }
        ],
        "ownerPeerSize":null,
        "spec":"",
        "state":"RUNNING",
        "type":"Internal",
        "userId":"3cec085510884e4488a431c426eac95f",
        "uuid":"1888b534-a2df-e63d-7ecf-4b5cf381d7f7"
      }
    ]
  }
}
```

```
}  
}
```

退出联盟

退出联盟

接口描述 退出联盟。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
POST /v1/fabric/consortiums/quit HTTP/1.1  
Host: bbe.bj.baidubce.com  
x-bce-date: <x-bce-date>  
x-bce-request-id: <x-bce-request-id>  
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Body	组织Uuid	是
consortiumUuid	String	Body	联盟Uuid	是

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"taskUuid":"testId"}

示例

- 请求示例

```
POST /v1/fabric/consortiums/quit HTTP/1.1  
Host: bbe.bj.baidubce.com  
x-bce-date: 2019-10-23T06:26:02Z  
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c  
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8  
{  
  "orgUuid":"testid",  
  "consortiumUuid":"testid"  
}
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "taskUuid":"testId"
  }
}
```

删除联盟

删除联盟

接口描述 删除联盟。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
DELETE /v1/fabric/consortiums?orgUuid={orgUuid}&consortiumUuid={consortiumUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query	组织Uuid	是
consortiumUuid	String	Query	联盟Uuid	是

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"taskUuid":"testId"}

示例

- 请求示例

```
DELETE /v1/fabric/consortiums?orgUuid=2cd9e3d2-438a-f579-ebef-46056b1cb10e&consortiumUuid=3b1a289b-
da2f-1bc0-518e-8de037cd79e2 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-
23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "taskUuid":"testId"
  }
}
```

通道管理

🔗 申请创建通道

申请创建通道 接口描述：申请创建通道，发起申请后会发送审批，全部审批通过后不会立即创建，需要用户再点击“创建”按钮进行实际创建

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

- 请求语法

```
POST /v1/fabric/consortiums/channels/create/apply HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	string	Body参数	联盟链uuid	必须
orgUuid	string	Body参数	创建通道的组织uuid	必须
name	string	Body参数	通道名称 支持大小写字母、数字、中文以及-_/特殊字符，不能以特殊字符开头，长度1-20	必须
englishName	string	Body参数	通道英文名称 支持小写字母、数字以及-。必须以字母开头，长度1-16	必须
description	string	Body参数	通道描述	必须
orgs	[]string	Body参数	邀请加入组织uuid列表	必须

响应（Response）

- 响应头域：除公共头域外，无其它特殊头域。

- 响应参数：

名称	类型	描述
channelUuid	string	通道Uuid
applicationUuid	string	申请Uuid

失败响应参考错误码部分。

示例

- 请求示例

```
POST /v1/fabric/consortiums/channels/create/apply HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:25:20Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af

{
  "consortiumUuid": "6167a91e-0409-4038-a74c-22d6bad61f91",
  "orgUuid":"bc197ef9-0c47-403a-829a-d537c7ee1fa4",
  "name":"贸易通道",
  "englishName":"test-channel-liuxiong",
  "description":"open api test",
  "orgs": ["bc197ef9-0c47-403a-829a-d537c7ee1fa4"]
}
```

- 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "channelUuid": "1206a499-14c2-4dbf-b0ba-e6716b46c169",
    "applicationUuid": "1206a499-14c2-4dbf-b0ba-e6716b46c169"
  }
}
```

创建通道

创建通道 接口描述：创建通道流程完成后调用此方法执行实际的创建通道逻辑

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

- 请求语法

```
POST /v1/fabric/consortiums/channels HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	string	Body参数	联盟链uuid	必须
channelUuid	string	Body参数	通道uuid	必须
orgUuid	string	Body参数	发起操作的机构uuid	必须

响应（Response）

- 响应头域：除公共头域外，无其它特殊头域。
- 响应参数：

名称	类型	描述
channelUuid	string	通道Uuid
taskUuid	string	部署任务uuid

失败响应参考错误码部分。

示例

- 请求示例

```
POST /v1/fabric/consortiums/channels HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:25:20Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af

{
  "consortiumUuid": "6167a91e-0409-4038-a74c-22d6bad61f91",
  "orgUuid": "bc197ef9-0c47-403a-829a-d537c7ee1fa4",
  "channelUuid": "1206a499-14c2-4dbf-b0ba-e6716b46c169"
}
```

- 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "channelUuid": "1206a499-14c2-4dbf-b0ba-e6716b46c169",
    "taskUuid": "43320e8d-d1e1-46f1-be88-de0ca6cfe3b3"
  }
}
```

🔗 申请加入通道

申请加入通道 接口描述：申请加入某个通道，需要该通道内已有组织全部审批通过

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

- 请求语法

```
POST /v1/fabric/consortiums/channels/join/apply HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	string	Body参数	联盟链uuid	必须
channelUuid	string	Body参数	通道uuid	必须
orgUuid	string	Body参数	申请加入通道的组织uuid	必须

响应（Response）

- 响应头域：除公共头域外，无其它特殊头域。
- 响应参数：

名称	类型	描述
channelUuid	string	通道Uuid
applicationUuid	string	申请Uuid

失败响应参考错误码部分。

示例

- 请求示例

```
POST /v1/fabric/consortiums/channels/join/apply HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:25:20Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af

{
  "consortiumUuid": "6167a91e-0409-4038-a74c-22d6bad61f91",
  "orgUuid": "bc197ef9-0c47-403a-829a-d537c7ee1fa4",
  "channelUuid": "1206a499-14c2-4dbf-b0ba-e6716b46c169",
}
```

- 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "channelUuid": "1206a499-14c2-4dbf-b0ba-e6716b46c169",
    "applicationUuid": "1206a499-14c2-4dbf-b0ba-e6716b46c169"
  }
}
```

加入通道

加入通道 接口描述：申请加入某个通道的流程完成后，调用此方法执行实际的加入通道逻辑

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

- 请求语法

```
POST /v1/fabric/consortiums/channels/join HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	string	Body参数	联盟链uuid	必须
channelUuid	string	Body参数	通道uuid	必须
orgUuid	string	Body参数	加入通道的组织uuid	必须
applicationUuid	string	Body参数	关联申请的uuid	必须

响应（Response）

- 响应头域：除公共头域外，无其它特殊头域。
- 响应参数：

名称	类型	描述
channelUuid	string	通道Uuid
taskUuid	string	部署任务uuid

失败响应参考错误码部分。

示例

- 请求示例

```
POST /v1/fabric/consortiums/channels/join HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:25:20Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af

{
  "consortiumUuid": "6167a91e-0409-4038-a74c-22d6bad61f91",
  "channelUuid": "1206a499-14c2-4dbf-b0ba-e6716b46c169",
  "orgUuid": "bc197ef9-0c47-403a-829a-d537c7ee1fa4",
  "applicationUuid": "bc197ef9-0c47-403a-829a-d537c7ee1fa4"
}
```

● 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "channelUuid": "1206a499-14c2-4dbf-b0ba-e6716b46c169",
    "taskUuid": "04b1b1b8-a0d3-402c-bd74-81b2f54d718e"
  }
}
```

🔗 修改通道描述

修改通道描述 接口描述：修改某个通道的描述，只有创建通道的组织可以进行此操作

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

● 请求语法

```
PUT /v1/fabric/consortiums/channels HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。

● 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	string	Body参数	联盟链uuid	必须
channelUuid	string	Body参数	通道uuid	必须
orgUuid	string	Body参数	发起操作的机构uuid	必须
description	string	Body参数	描述	必须

响应（Response）

- 响应头域：除公共头域外，无其它特殊头域。

- 响应参数：

名称	类型	描述
channelUuid	string	通道Uuid

失败响应参考错误码部分。

示例

- 请求示例

```
PUT /v1/fabric/consortiums/channels HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:25:20Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af

{
  "consortiumUuid": "6167a91e-0409-4038-a74c-22d6bad61f91",
  "channelUuid": "1206a499-14c2-4dbf-b0ba-e6716b46c169",
  "orgUuid": "bc197ef9-0c47-403a-829a-d537c7ee1fa4",
  "description": "test"
}
```

- 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "channelUuid": "1206a499-14c2-4dbf-b0ba-e6716b46c169"
  }
}
```

🔗 查询通道详情

查询通道详情 接口描述：查询联盟中通道的详情，包括通道中组织的详细信息

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/channel/detail:channelUuid HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
channelUuid	string	Path参数	通道uuid	必须

响应（Response）

- 响应头域：除公共头域外，无其它特殊头域。
- 响应参数：

名称	类型	描述
channel	FabricChannel	通道详情

Channel:

名称	类型	描述
channelName	String	通道名称
networkUuid	String	网络uuid
consortiumUuid	String	联盟Uuid
peerOrgUuidList	[]String	通道中peer组织uuid
orgs	[]String	通道中组织uuid列表
participants	String	通道中组织详情列表
state	String	状态
userId	String	用户ID
uuid	String	uuid
description	String	描述
createTime	String	创建时间
initiator	String	发起人

Participant:

名称	类型	描述
OrgUuid	String	组织uuid
OrgName	String	组织中文名
OrgEnglishName	String	组织英文名
Description	String	组织描述
JoinTime	String	加入通道时间

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/fabric/consortiums/channel/detail?channelUuid=9f3f492e-c949-eda1-a94c-70bd39a2da20 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-12-20T06:55:23Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af
```

- 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT
{
  "data": {
    "channel": {
      "channelName": "ym-channel",
      "networkUuid": "",
      "consortiumUuid": "764b436f-2772-3a2b-a94c-70bd39a2da20",
      "participants": [
        {
          "orgUuid": "ec98e5b1-a381-b306-f0d1-66fe1f717d2a",
          "orgName": "ym-org",
          "orgEnglishName": "ym-org",
          "description": "desc",
          "joinTime": "2019-12-20T10:22:20+08:00"
        }
      ],
      "peerOrgUuidList": [
        "ec98e5b1-a381-b306-f0d1-66fe1f717d2a"
      ],
      "orgs": [
        "ym-org"
      ],
      "description": "ym-channel",
      "userId": null,
      "uuid": "9f3f492e-c949-eda1-a94c-70bd39a2da20",
      "createTime": "2019-12-20T10:22:20+08:00",
      "state": "RUNNING",
      "initiator": "ec98e5b1-a381-b306-f0d1-66fe1f717d2a"
    }
  }
}
```

🔗 查询通道出块配置

查询通道出块配置 接口描述

查询联盟中通道的配置，包括出块大小和出块超时时间

权限说明

请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项

如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

● 请求语法

```
GET v1/fabric/consortiums/channel/batch HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

● 请求头域

除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	string	Path参数	联盟uuid	必须
orgUuid	string	Path参数	组织uuid	必须
channelUuid	string	Path参数	通道uuid	必须

响应（Response）

- 响应头域

除公共头域外，无其它特殊头域。

- 响应参数：

名称	类型	描述
batch	Batch	出块配置

Batch:

名称	类型	描述
batchSize	BatchSize	出块大小配置
batchTimeout	BatchTimeout	出块超时时间配置

BatchSize:

名称	类型	描述
maxMessageCount	String	区块最多包含交易数
absoluteMaxBytes	String	区块最多包含交易字节数
preferredMaxBytes	String	当一笔交易大小超过该值时，该交易将单独打包

BatchTimeout:

名称	类型	描述
timeout	String	区块打包超时时间

失败响应错误码，见[错误码](#)。

示例

- 请求示例

```
GET /v1/fabric/consortiums/channel/batch?consortiumUuid=c5127af7-7c22-2eed-a94c-70bd39a2da21&orgUuid=f771b844-7a70-fae0-a94c-70bd39a2da21&channelUuid=9f3f492e-c949-eda1-a94c-70bd39a2da20 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-12-20T06:55:23Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af
```

- 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT
{
  "data": {
    "batch": {
      "batchSize": {
        "maxMessageCount": 10,
        "absoluteMaxBytes": 103809024,
        "preferredMaxBytes": 524288
      },
      "batchTimeout": {
        "timeout": "1000"
      }
    }
  }
}
```

🔗 修改通道出块配置

修改通道出块配置 接口描述

修改某个通道的出块，只有创建通道的组织可以进行此操作

权限说明

请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项

如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

- 请求语法

```
POST v1/fabric/consortiums/channel/batch HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域

除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	string	Body参数	联盟链uuid	必须
channelUuid	string	Body参数	通道uuid	必须
orgUuid	string	Body参数	发起操作的机构uuid	必须
batchConfig	BatchConfig	Body参数	通道具体出块配置	必须

BatchConfig:

名称	类型	描述
batchSize	BatchSize	出块大小配置
batchTimeout	BatchTimeout	出块超时时间配置

BatchSize:

名称	类型	描述
maxMessageCount	String	区块最多包含交易数
absoluteMaxBytes	String	区块最多包含交易字节数
preferredMaxBytes	String	当一笔交易大小超过该值时，该交易将单独打包

BatchTimeout:

名称	类型	描述
timeout	String	区块打包超时时间

响应（Response）

- 响应头域

除公共头域外，无其它特殊头域。

- 响应参数

无返回结果

失败响应错误码，见[错误码](#)。

示例

- 请求示例

```
PUT /v1/fabric/consortiums/channel/batch HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:25:20Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af

{
  "consortiumUuid" : "c5127af7-7c22-2eed-a94c-70bd39a2da21",
  "orgUuid" : "f771b844-7a70-fae0-a94c-70bd39a2da21",
  "channelUuid" : "9f3f492e-c949-eda1-a94c-70bd39a2da20",
  "batchConfig": {
    "batchSize": {
      "maxMessageCount": 20,
      "absoluteMaxBytes": 103809024,
      "preferredMaxBytes": 524288
    },
    "batchTimeout": {
      "timeout": "1000"
    }
  }
}
```

- 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT
```

申请、审批和通知管理

通过审批

通过审批 接口描述：通过某个审批

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求 (Request)

- 请求语法

```
POST /v1/fabric/consortiums/approvals/approve HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
approvalUuid	string	Body参数	审批记录Uuid	必须

响应 (Response)

- 响应头域：除公共头域外，无其它特殊头域。

- 响应参数：无

示例

- 请求示例

```
POST /v1/fabric/consortiums/approvals/approve HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:25:20Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af

{
  "approvalUuid": "89fa78c0-fedc-ecc9-b30e-9ae72060b8c1-0"
}
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
```

🔗 查看通知列表

查看通知列表 接口描述：查看发送给用户的所有组织的事件通知

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/events/list?tag={tag}&state={state} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
tag	int	Query参数	事件类型（1 联盟，2 channel，3 chaincode）	必须
state	int	Query参数	通知状态（0 未读，1 已读）	必须
consortiumName	String	Query参数	联盟名	可选
channelName	String	Query参数	通道名	可选

响应（Response）

- 响应头域：除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
Events	[]Event	事件通知列表， 结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/fabric/consortiums/events/list?tag=2&state=0 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:50:23Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:50:23Z/1800/host/6fafe75cbcc9424e7d176a1ad8bcc94be081749ac8c3809f069f6a23ae290499
```

- 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "events": [
```

```

    {
      "uuid": "c324986a-e305-4f21-bd40-18c2f65a1643-agree-approval-0",
      "fromUuid": "edaefb5b-3344-4321-88b0-bc38ecb3764e",
      "fromName": "fab-con-573-auto-test",
      "toUuid": "bc197ef9-0c47-403a-829a-d537c7ee1fa4",
      "consortiumName": "fab-con-573-auto-test",
      "channelName": "cha-jw2",
      "tag": 2,
      "state": 0,
      "message": "亲爱的审批者，通道创建审批成功",
      "createTime": "2019-10-29T16:31:16+08:00"
    },
    {
      "uuid": "89fa78c0-fedc-ecc9-b30e-9ae72060b8c1-event-0",
      "fromUuid": "edaefb5b-3344-4321-88b0-bc38ecb3764e",
      "fromName": "fab-org2-195-auto-test",
      "toUuid": "bc197ef9-0c47-403a-829a-d537c7ee1fa4",
      "consortiumName": "fab-con-573-auto-test",
      "channelName": "cha-jw2",
      "tag": 2,
      "state": 0,
      "message": "组织fab-org2-195-auto-test申请创建通道cha-jw2",
      "createTime": "2019-10-29T10:36:17+08:00"
    },
    {
      "uuid": "9124e692-8c58-4de4-9a64-d9aa80b6b7c1-agree-approval-0",
      "fromUuid": "6580c04b-fd52-4905-ba0a-467b260f4fa1",
      "fromName": "fab-con-742-auto-test",
      "toUuid": "bb696d38-8594-44da-aad9-6ea0fb37d56d",
      "consortiumName": "fab-con-742-auto-test",
      "channelName": "fab-cha-304-auto-test",
      "tag": 2,
      "state": 0,
      "message": "组织6580c04b-fd52-4905-ba0a-467b260f4fa1加入通道",
      "createTime": "2019-10-28T20:20:17+08:00"
    },
    {
      "uuid": "e0f809e5-64a1-4435-85f7-5b28b37040b9-event-0",
      "fromUuid": "6580c04b-fd52-4905-ba0a-467b260f4fa1",
      "fromName": "fab-org2-738-auto-test",
      "toUuid": "bb696d38-8594-44da-aad9-6ea0fb37d56d",
      "consortiumName": "fab-con-742-auto-test",
      "channelName": "fab-cha-304-auto-test",
      "tag": 2,
      "state": 0,
      "message": "组织fab-org1-880-auto-test申请加入通道fab-cha-304-auto-test",
      "createTime": "2019-10-28T20:20:16+08:00"
    },
    {
      "uuid": "28a2d1c7-9d6d-490a-a8d4-f2912b1b4a43-agree-sender-15",
      "fromUuid": "6580c04b-fd52-4905-ba0a-467b260f4fa1",
      "fromName": "fab-con-742-auto-test",
      "toUuid": "bb696d38-8594-44da-aad9-6ea0fb37d56d",
      "consortiumName": "fab-con-742-auto-test",
      "channelName": "fab-cha-304-auto-test",
      "tag": 2,
      "state": 0,
      "message": "亲爱的用户，通道创建审批成功",
      "createTime": "2019-10-28T20:20:02+08:00"
    }
  ]
}

```

```
}

```

查看审批列表

查看审批列表 接口描述：查询某个人及其所有组织的审批记录，包括联盟创建方邀请用户加入联盟，channel创建方申请创建channel，channel创建方邀请加入channel，同联盟组织申请加入channel

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/approvals/list?tag={tag}&state={state} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
tag	int	Query参数	事件类型（1 联盟，2 channel，3 chaincode）	必须
state	bool	Query参数	是否已审批（true: 已审批， false：待审批）	必须
consortiumName	String	Query参数	联盟名	可选
channelName	String	Query参数	通道名	可选

响应（Response）

- 响应头域：除公共头域外，无其它特殊头域。

- 响应参数

名称	类型	描述
approvals	[]Approval	申请结果列表， 结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/fabric/consortiums/approvals/list?tag=1&state=true HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:25:20Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af
```

- 响应示例

```
200
Content-Type: application/json; charset=utf-8
{
  "data": {
    "approvals": [
```

```

{
  "uuid": "3cfceeaf-cb06-4470-9d3c-dcd924df8fbf-0",
  "applicationUuid": "3cfceeaf-cb06-4470-9d3c-dcd924df8fbf",
  "orgName": "fab-org1-716-auto-test",
  "message": "fab-org1-716-auto-test邀请用户eca97e148cb74e9683d7b7240829d1ff加入联盟fab-con-539-
auto-test",
  "approverUuid": "eca97e148cb74e9683d7b7240829d1ff",
  "approverName": "eca97e148cb74e9683d7b7240829d1ff",
  "consortiumName": "fab-con-539-auto-test",
  "consortiumUuid": "fd657c8f-2815-4cae-8e6c-1b611f8f0433",
  "channelName": "",
  "channelUuid": "",
  "tag": 1,
  "state": "PASS",
  "feedback": "",
  "createTime": "2019-10-17T19:07:52+08:00",
  "finishTime": "2019-10-17T19:07:53+08:00",
  "objectState": "CONFIRMED"
},
{
  "uuid": "c31b19d4-dd7b-4d7f-a936-388ef203592f-0",
  "applicationUuid": "c31b19d4-dd7b-4d7f-a936-388ef203592f",
  "orgName": "fab-org1-316-auto-test",
  "message": "fab-org1-316-auto-test邀请用户eca97e148cb74e9683d7b7240829d1ff加入联盟fab-con-573-
auto-test",
  "approverUuid": "eca97e148cb74e9683d7b7240829d1ff",
  "approverName": "eca97e148cb74e9683d7b7240829d1ff",
  "consortiumName": "fab-con-573-auto-test",
  "consortiumUuid": "6167a91e-0409-4038-a74c-22d6bad61f91",
  "channelName": "",
  "channelUuid": "",
  "tag": 1,
  "state": "PASS",
  "feedback": "",
  "createTime": "2019-10-28T18:33:01+08:00",
  "finishTime": "2019-10-28T18:33:08+08:00",
  "objectState": "CONFIRMED"
},
{
  "uuid": "578ef15b-262f-4f97-bf0e-1a23a873a292-0",
  "applicationUuid": "578ef15b-262f-4f97-bf0e-1a23a873a292",
  "orgName": "fab-org1-880-auto-test",
  "message": "fab-org1-880-auto-test邀请用户eca97e148cb74e9683d7b7240829d1ff加入联盟fab-con-742-
auto-test",
  "approverUuid": "eca97e148cb74e9683d7b7240829d1ff",
  "approverName": "eca97e148cb74e9683d7b7240829d1ff",
  "consortiumName": "fab-con-742-auto-test",
  "consortiumUuid": "985161d4-2842-45ae-b199-830f31f5faee",
  "channelName": "",
  "channelUuid": "",
  "tag": 1,
  "state": "PASS",
  "feedback": "",
  "createTime": "2019-10-28T19:56:54+08:00",
  "finishTime": "2019-10-28T19:56:54+08:00",
  "objectState": "CONFIRMED"
}
]
}
}

```

🔗 查询审批流

查询审批流 接口描述：查询某个申请的审批情况

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/applications?applicationUid={applicationUid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
applicationUid	string	Query参数	申请Uuid	必须

响应（Response）

- 响应头域：除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
approvals	[]Approval	某申请的审批情况, 结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/fabric/consortiums/applications?applicationUid=ae9cf072-5757-42a8-b910-eb2c194bbf22 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:19:39Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:19:39Z/1800/host/b9002c887dc1cbf075d4277024afc2f4eb36f8925185d06904a646c258353216
```

- 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "approvals": [
      {
        "uuid": "ae9cf072-5757-42a8-b910-eb2c194bbf22-master",
        "applicationUuid": "ae9cf072-5757-42a8-b910-eb2c194bbf22",
        "orgName": "fab-org2-738-auto-test",
        "message": "组织fab-org2-738-auto-test申请创建通道fab-cha-304-auto-test",
        "approverUuid": "6580c04b-fd52-4905-ba0a-467b260f4fa1",
        "approverName": "fab-org1-880-auto-test",
        "consortiumName": "fab-con-742-auto-test",
        "consortiumUuid": "985161d4-2842-45ae-b199-830f31f5faee",
        "channelName": "fab-cha-304-auto-test",
        "channelUuid": "ae9cf072-5757-42a8-b910-eb2c194bbf22",
        "tag": 2,
        "state": "PASS",
        "feedback": "",
        "createTime": "2019-10-28T20:20:00+08:00",
        "finishTime": "2019-10-28T20:20:02+08:00",
        "objectState": ""
      }
    ]
  }
}
```

更新通知状态

更新通知状态 接口描述：用户查看通知后，更新通知状态为已阅

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

- 请求语法

```
POST /v1/fabric/consortiums/events/updateState HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
eventUuid	string	Body参数	事件通知的uuid	必须

响应（Response）

- 响应头域：除公共头域外，无其它特殊头域。
- 响应参数：无

示例

- 请求示例

```
POST /v1/fabric/consortiums/events/updateState HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:25:20Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af

{
  "eventUuid": "28a2d1c7-9d6d-490a-a8d4-f2912b1b4a43-agree-sender-15"
}
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
```

🔗 驳回审批

驳回审批 接口描述：驳回某个审批

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

- 请求语法

```
POST /v1/fabric/consortiums/approvals/reject HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域：除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
approvalUuid	string	Body参数	审批记录Uuid	必须
reason	string	Body参数	驳回原因	必须

响应（Response）

- 响应头域：除公共头域外，无其它特殊头域。
- 响应参数：无

示例

- 请求示例

```
POST /v1/fabric/consortiums/approvals/reject HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T08:25:20Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T08:25:20Z/1800/host/56a2a8096f99d5b0fb38f7815b61f43f28b662f0bb49c3407386f932ab6550af

{
  "approvalUuid": "fa78c089-fedc-ecc9-b30e-9ae72060b8c1-0"
}
```

● 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
```

🔗 查询申请列表

查询申请列表 接口描述：查询某个用户所有组织的申请记录：包括邀请用户加入联盟，申请并邀请组织创建channel，申请加入channel

权限说明：请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项：如果请求中没有用户验证信息（即匿名访问），返回**403 Forbidden**，错误信息：**AccessDenied**。

请求（Request）

● 请求语法

```
GET /v1/fabric/consortiums/applications/list?tag={tag}&state={state} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

● 请求头域：除公共头域外，无其它特殊头域。

● 请求参数

名称	类型	位置	描述	是否必须
tag	int	Query参数	事件类型（1 联盟，2 channel，3 chaincode）	必须
state	bool	Query参数	申请状态(true: 已完成， false: 未完成)	必须
consortiumName	String	Query参数	联盟名	可选
channelName	String	Query参数	通道名	可选

响应（Response）

● 响应头域：除公共头域外，无其它特殊头域。

● 响应参数

名称	类型	描述
applications	[]Application	申请结果列表， 结构具体定义见附录说明

失败响应参考错误码部分。

示例

• 请求示例

```
GET /v1/fabric/consortiums/applications/list?tag=2&state=false HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-29T07:55:43Z
Authorization: bce-auth-v1/e38d78f7b74841ec92727531369834d8/2019-10-29T07:55:43Z/1800/host/c5bc425b81bc737709131964c96e3c10e3faec3c2163e6fb2e16348f26ad08df
```

• 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data": {
    "applications": [
      {
        "uuid": "ae9cf072-5757-42a8-b910-eb2c194bbf22",
        "orgUuid": "bb696d38-8594-44da-aad9-6ea0fb37d56d",
        "orgName": "fab-org2-738-auto-test",
        "consortiumName": "fab-con-742-auto-test",
        "consortiumUuid": "985161d4-2842-45ae-b199-830f31f5faee",
        "channelName": "fab-cha-304-auto-test",
        "channelUuid": "ae9cf072-5757-42a8-b910-eb2c194bbf22",
        "tag": 2,
        "type": 20,
        "message": "组织fab-org2-738-auto-test申请创建通道fab-cha-304-auto-test",
        "state": "PASS",
        "approveCnt": 0,
        "rejectCnt": 0,
        "totalCnt": 0,
        "targetRate": 1.0E-6,
        "createTime": "2019-10-28T20:20:00+08:00",
        "finishTime": "2019-10-28T20:20:02+08:00",
        "objectState": ""
      }
    ]
  }
}
```

链码管理

🔗 链码安装

安装Chaincode

接口描述

对已上传的chaincode进行安装。

权限说明

请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项

如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
POST /v1/fabric/chaincode/:chaincodeUuid/install HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域
除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	描述	是否必须
chaincodeUuid	String	chaincodeId	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数 无。失败响应参考错误码部分

示例

- 请求示例

```
POST /v1/fabric/chaincode/fd80b80c-a51a-4ae3-b0e8-2808ff3cf901/install HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 10:24:02 GMT
```

链码实例化

实例化Chaincode

接口描述

对已安装的chaincode进行实例化。

权限说明

请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项

如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
POST /v1/fabric/chaincode/:chaincodeUuid/instantiate HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域

除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	描述	是否必须
chaincodeUuid	String	chaincodeId	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数 无。失败响应参考错误码部分。

示例

- 请求示例

```
POST /v1/fabric/chaincode/fd80b80c-a51a-4ae3-b0e8-2808ff3cf901/instantiate HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 10:24:02 GMT
```

链码升级

升级Chaincode

接口描述

对已实例化的chaincode进行升级。

权限说明

请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项

如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
PUT /v1/fabric/chaincode/:chaincodeUuid HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域
除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	描述	是否必须
chaincodeUuid	String	chaincodeId	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数 无。失败响应参考错误码部分。

示例

- 请求示例

```
PUT /v1/fabric/chaincode/fd80b80c-a51a-4ae3-b0e8-2808ff3cf901 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 10:24:02 GMT
```

🔗 查询Chaincode详情

查询Chaincode详情

接口描述

根据chaincode uuid查询某个chaincode详细信息。

权限说明

请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项

如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：`AccessDenied`。

请求（Request）

- 请求语法

```
GET /v1/fabric/chaincode/:chaincodeUuid HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域
除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	描述	是否必须
chaincodeUuid	String	chaincode uuid	必须

响应（Response）

- 响应头域 无特殊Header参数响应
- 响应参数

名称	类型	描述
data	FabricChaincode	chaincode详细信息

示例

- 请求示例

```
GET /v1/fabric/chaincode/8b87c63d-64ca-4ba8-a5da-ddc813732210 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
{
  "data": {
    "name": "chaincode1",
    "uuid": "8b87c63d-64ca-4ba8-a5da-ddc813732210",
    "channelUuid": "c4f4235d-d843-49e3-9e1c-7cc703fa9ab6",
    "networkUuid": "8b87c63d-64ca-4ba8-a5da-ddc813732210",
    "orgName": "org1",
    "version": "1.2.3",
    "state": 2,
    "path": "map",
    "endorsementPolicy": "OR ('hdo5MSP.member')",
    "initArgs": "init",
    "createTime": "2018-12-25T06:52:10Z",
    "instantiateTime": "2018-12-25T16:52:10Z",
    "updatable": true,
    "errMsg": "{ \"org1\": \"success\" }"
  }
}
```

查询channel下Chaincode列表

接口描述

查询某个channel下所有chaincode详细信息。

权限说明

请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项

如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/fabric/chaincode/channel/:channelUuid HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域

除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	描述	是否必须
channelUuid	String	channelId	必须

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。

- 响应参数

名称	类型	描述
data	[]FabricChaincode	chaincode信息列表

示例

- 请求示例

```
GET /v1/fabric/chaincode/channel/c4f4235d-d843-49e3-9e1c-7cc703fa9ab6 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例


```
{
  "data": [
    {
      "name": "chaincode1",
      "uuid": "8b87c63d-64ca-4ba8-a5da-ddc813732210",
      "channelUuid": "c4f4235d-d843-49e3-9e1c-7cc703fa9ab6",
      "networkUuid": "8b87c63d-64ca-4ba8-a5da-ddc813732210",
      "orgName": "org1",
      "version": "1.2.3",
      "state": 2,
      "path": "map",
      "endorsementPolicy": "OR ('hdo5MSP.member')",
      "initArgs": "init",
      "createTime": "2018-12-25T06:52:10Z",
      "instantiateTime": "2018-12-25T16:52:10Z",
      "updatable": true,
      "errMsg": "{\"org1\":\"success\"}"
    },
    {
      "name": "chaincod2",
      "uuid": "c7ea68c4-48c9-4dbc-b3e0-e02933c1bd9b",
      "channelUuid": "c4f4235d-d843-49e3-9e1c-7cc703fa9ab6",
      "networkUuid": "8b87c63d-64ca-4ba8-a5da-ddc813732210",
      "orgName": "org1",
      "version": "1.2.2",
      "state": 5,
      "path": "map",
      "endorsementPolicy": "OR ('hdo5MSP.member')",
      "initArgs": "init",
      "createTime": "2018-12-25T03:24:10Z",
      "instantiateTime": "2018-12-25T05:43:10Z",
      "updatable": true,
      "errMsg": "{\"org1\":\"success\"}"
    }
  ]
}
```

节点管理

🔗 节点列表查询

节点列表查询

接口描述 节点列表查询。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求 (Request)

- 请求语法

```
GET /v1/fabric/consortiums/nodes?orgUuid={orgUuid}&consortiumUuid={consortiumUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query	组织Uuid	是
consortiumUuid	String	Query	联盟Uuid	是

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"nodes":[]Node}

Node:

名称	类型	描述
id	Int	id
uuid	String	唯一ID
consortiumUuid	String	联盟Uuid
orgUuid	String	组织Uuid
nodeName	String	节点名称
nodeType	String	节点类型：orderer/peer
url	String	公网地址
explorerUrl	String	浏览器地址，只有peer类型的节点拥有
deleted	Bool	是否已经删除
createTime	String	创建时间

示例

- 请求示例

```
GET /v1/fabric/consortiums/nodes?orgUuid=2cd9e3d2-438a-f579-ebef-46056b1cb10e&consortiumUuid=3b1a289b-da2f-1bc0-518e-8de037cd79e2 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "nodes":[{
      "id":12,
      "uuid":"f5891400-5fb2-f2da-a08f-78dfab2dfaef",
      "consortiumUuid":"f5891400-5fb2-f2da-a08f-78dfab2dfaef",
      "orgUuid":"f5891400-5fb2-f2da-a08f-78dfab2dfaef",
      "nodeName":"org1-peer0",
      "nodeType":"peer",
      "url":"http://127.0.0.1:7050"
      "explorerUrl":"http://127.0.0.1:8080",
      "deleted":false,
      "createTime":"2019-10-30T12:28:58+08:00"
    }]
  }
}
```

🔗 节点资源列表

节点资源列表

接口描述 节点资源列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/resources?orgUuid={orgUuid}&consortiumUuid={consortiumUuid} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query	组织Uuid	是
consortiumUuid	String	Query	联盟Uuid	是

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"resources":[]Resource}

Resource:

名称	类型	描述
nodeName	String	节点名称
clusterId	String	集群ID
nodeId	String	机器ID
podName	String	pod名称
totalCpu	int	cup总量，单位：m
totalMemory	int	内存总量，单位：byte
totalDisk	int	磁盘总量，单位：byte
usedCpu	int	cup使用量，单位：m
usedMemory	int	内存使用量，单位：byte
usedDisk	int	磁盘使用量，单位：byte

示例

- 请求示例

```
GET /v1/fabric/consortiums/resources?orgUuid=2cd9e3d2-438a-f579-ebef-46056b1cb10e&consortiumUuid=3b1a289b-da2f-1bc0-518e-8de037cd79e2 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "resources":[{
      "nodeName":"org1-peer0",
      "clusterId":"i-cfdsfdf",
      "nodeId":"testid",
      "podName":"org1-peer0-0",
      "totalCpu":"4000",
      "totalMemory":"54543543543",
      "totalDisk":"4244454",
      "usedCpu":"1000",
      "usedMemory":"2222222",
      "usedDisk":"1111111"
    }]
  }
}
```

cpu和内存时序图

cpu和内存时序图

接口描述 cpu和内存时序图。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/metric?clusterId={clusterId}&podName={podName} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
clusterId	String	Query	集群ID	是
podName	String	Query	pod名称	是

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"metric":Metric}

Metric:

名称	类型	描述
cpuUsageHistory	[]PointsHistory	cup使用情况
memoryUsageHistory	[]PointsHistory	内存使用情况

PointsHistory:

名称	类型	描述
timestamp	String	时间
value	long	值，byte/m

示例

- 请求示例

```
GET /v1/fabric/consortiums/metric?clusterId=c-gU68v5Q1&podName=peer0-0 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "metric":{
      "cpuUsageHistory":[{
        "timestamp":"2019-10-22T02:07:00Z",
        "value":1000,
      }],
      "memoryUsageHistory":[{
        "timestamp":"2019-10-22T02:07:00Z",
        "value":2212,
      }]
    }
  }
}
```

证书管理

🔗 查询证书列表

查询证书列表

接口描述

查询证书列表。

权限说明

请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项

如果请求中没有用户验证信息（即匿名访问），返回 403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/msp/list?consortiumUuid={consortiumUuid}&orgUuid={orgUuid}&pageNo={pageNo}&pageSize={pageSize} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
consortiumUuid	String	Query	联盟Uuid	是
orgUuid	String	Query	组织Uuid	是
pageNo	int	Query	当前请求页数	是
pageSize	int	Query	每页请求的数量	是

响应（Response）

- 响应头域
除公共头域外，无其它特殊头域。

- 响应参数

名称	类型	描述
data	json	data 结构如下：

data

名称	类型	描述
totalCount	int	查询证书列表总量
pageNo	String	当前返回页
pageSize	String	每页返回大小
result	[]cert	返回结果集，cert定义如下

cert:

名称	类型	描述
uuid	String	唯一ID
consortiumUuid	String	联盟uuid
orgUuid	String	组织uuid
userName	String	用户名
attrs	String	属性集
type	String	类型：client
encryptionAlgorithm	String	签名算法：ECC
status	String	状态：Normal、Revoke、Expire
deleted	bool	是否已删除
createTime	String	创建时间
expireTime	String	过期时间

失败响应错误码，见[错误码](#)。

示例

- 请求示例

```
GET /v1/fabric/consortiums/msp/list?consortiumUuid=c5127af7-7c22-2eed-a94c-70bd39a2da20&orgUuid=f771b844-7a70-fae0-a94c-70bd39a2da20&pageNo=1&pageSize=1000 HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "totalCount":1,
    "pageNo":1,
    "pageSize":10,
    "result":[{
      "uuid":"ca420577-cb0d-4044-8d66-b4e0168aff9f",
      "consortiumUuid":"81bf8c67-84cb-4202-a928-bfb7018930a8",
      "orgUuid":"f6ee7f89-4f50-d801-a94c-70bd39a2da20",
      "userName":"Bob",
      "secret":"PLpTxKjrauQz",
      "attrs":"role=admin,permission=1",
      "type":"client",
      "encryptionAlgorithm":"ECC",
      "status":"Normal",
      "deleted":false,
      "createTime":"2020-06-17T11:54:44+08:00",
      "expireTime":"2021-06-17T11:54:44+08:00"
    }]
  }
}
```

🔗 申请证书

申请证书

接口描述

申请证书，当前只支持申请用户证书用于sdk调用链码。

权限说明

请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项

如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
POST /v1/fabric/consortiums/msp/create HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域

除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Body	组织Uuid	是
consortiumUuid	String	Body	联盟Uuid	是
userName	String	Body	用户名	是
attrs	String	Body	属性集	是

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
data	json	返回结果：{"mspUuid":""}，mspUuid 该申请证书唯一ID

示例

- 请求示例

```
POST /v1/fabric/consortiums/msp/create HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
{
  "consortiumUuid":"81bf8c67-84cb-4202-a928-bfb7018930a8",
  "orgUuid":"f6ee7f89-4f50-d801-a94c-70bd39a2da20",
  "userName":"Bob",
  "attrs":"role=admin,permission=1"
}
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
{
  "data":{
    "mspUuid":"4dae5286-578e-47fc-90db-d6a5e20ac2a5"
  }
}
```

📄 下载证书

下载证书

接口描述

下载证书，格式为tar.gz

权限说明

请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项

如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/fabric/consortiums/msp/download?orgUuid={orgUuid}&consortiumUuid={consortiumUuid}&mspUuid={mspUuid}&userName={userName} HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	位置	描述	是否必须
orgUuid	String	Query	组织Uuid	是
consortiumUuid	String	Query	联盟Uuid	是
mspUuid	String	Query	证书Uuid	是
userName	String	Query	用户名	是

响应（Response）

- 响应头域
除公共头域外，无其它特殊头域。
- 响应参数 二进制流

示例

- 请求示例

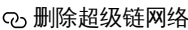
```
GET /v1/fabric/consortiums/msp/download?orgUuid=2cd9e3d2-438a-f579-ebef-46056b1cb10e&consortiumUuid=3b1a289b-da2f-1bc0-518e-8de037cd79e2&mspUuid=ca420577-cb0d-4044-8d66-b4e0168aff9f&userName=Bob HTTP/1.1
Host: bbe.bj.baidubce.com
x-bce-date: 2019-10-23T06:26:02Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/98cdcde27d8e464d895222fd8023a029/2019-10-23T06:26:02Z/1800/host/1e88313cae40a47550bdf0e3bfa591c06e4443c1ed690d510032e084676b00f8
```

- 响应示例

```
200
x-bce-request-id: 4e6bba7f-13c4-4e09-b9ba-030524713358~~~~
Content-Type: application/octet-stream;charset=UTF-8
Date: Wed, 23 Oct 2019 06:26:02 GMT
二进制流
```

超级链相关接口

删除超级链网络



接口描述 根据提供的超级链网络UUID，删除该区块链网络。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求 (Request)

- 请求语法

```
DELETE /v1/xchain/network/:networkUuid
Host: bbe.bj.baidubce.com
Date: <Date>
Authorization: <Authorization_String>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数 除公共头域外，无其它特殊头域。

响应 (Response)

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数 无。失败响应参考错误码部分。

示例

- 请求示例

```
DELETE /v1/xchain/network/2b031568-5073-4967-bc86-bff6a242cd11 HTTP/1.1

content-type: application/json
host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 10:24:02 GMT
```

获取超级链网络全量信息

🔗 获取超级链网络全量信息

接口描述 根据提供的超级链网络UUID，获取该网络的全量信息。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求 (Request)

- 请求语法

```
GET /v1/xchain/information
Host: bbe.bj.baidubce.com
x-bce-date: <x-bce-date>
x-bce-request-id: <x-bce-request-id>
Authorization: <Authorization_String>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	描述	是否必须
networkUuid	String	超级链网络的UUID	必须

响应（Response）

- 响应头域

除公共头域外，无其它特殊头域。

- 响应参数

名称	类型	描述
information	String	全量信息
networkUuid	String	超级链网络ID
networkName	String	超级链网络名称
practiceUrl	String	控制台地址

失败响应参考错误码部分。

示例

- 请求示例

```
GET bbe.bj.baidubce.com/v1/xchain/information?networkUuid=2b031568-5073-4967-bc86-bff6a242cd11
Host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 3153
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 07:24:02 GMT

{
  "data":{
    "information":"网络信息\n名称,UUID,状态,集群,控制台,出块奖励,衰减高度,衰减系数\ntest-lql,2b031568-5073-4967-bc86-bff6a242cd11,RUNNING,c-QOUyL2yh,http://10.107.245.134:8000,1000,1000,0.600000\n账户信息\n角色,地址,私钥,公钥\n\nsupernode,QTzQsgxfZ3ANDWBkCUSN1u1xhjW3AH7vb,eyJDdXJ2bmFtZSI6IiAtMjU2liwiWCI6MTAyODMOMTQ1OTkyMTk2NTQ",
    "networkUuid":"2b031568-5073-4967-bc86-bff6a242cd11",
    "networkName":"test-lql",
    "practiceUrl":"http://10.107.245.134:8000",
    "clusterUuid":"c-QOUyL2yh",
    "clusterName":"lql"
  }
}
```

获取超级链部署列表

🔗 获取超级链部署列表

接口描述 获取用户所有超级链网络列表。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求 (Request)

- 请求语法

```
GET /v1/xchain/list
Host: bbe.bj.baidubce.com
Date: <Date>
Authorization: <Authorization_String>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数

名称	类型	描述	是否必须
marker	String	批量获取列表的查询的起始位置，是一个由系统生成的字符串	否
maxKeys	Integer	每页包含的最大数量，最大数量通常不超过1000，缺省值为1000	否
status	String	bbe状态:状态间”，” 分割,可以不填写,则返回前端显示状态	否

响应 (Response)

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
marker	String	标记查询的起始位置
isTruncated	boolean	true表示后面还有数据，false表示已经是最后一页
nextMarker	String	获取下一页所需要传递的marker值。当isTruncated为false时，该域不出现
maxKeys	Integer	每页包含的最大数量
result	[]NetworkInfo	网络详情列表，结构具体定义见附录说明

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/xchain/list HTTP/1.1
content-type: application/json
host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/84bc3dd2774cf6a6276a86e0cc7680e01710a1b7cb1a87de552af581131c818f
```

- 响应示例

```
200
Content-Length: 6171
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 10:24:02 GMT

{
  "nextMarker": "-1",
  "marker": "-1",
  "maxKeys": 1000,
  "isTruncated": false,
  "result": [
    {
      "blockchainUuid": "1f7833ed-e378-4108-9f71-9d110386b298",
      "clusterUuid": "c-Q0UyL2yh",
      "clusterName": "lql",
      "accountId": "00dc1b52d8354d9193536e4dd2c41ae6",
      "networkUuid": "2b031568-5073-4967-bc86-bff6a242cd11",
      "networkName": "test-lql",
      "contractUuid": "",
      "chaincodeUuid": "",
      "networkType": null,
      "dappUuid": "",
      "taskUuid": "1a204bf5-f0c7-4064-85bb-3b199311fd4b",
      "createTime": 1546500733000,
      "deleteTime": 31536001000,
      "updateTime": null,
      "clusterStatus": "RUNNING",
      "taskStatus": "DONE",
      "channels": "",
      "remixUrl": null,
      "netstatsAddress": "http://10.107.245.134:8000",
      "explorerAddress": null
    }
  ]
}
```

查询预定义超级节点账户

查询预定义超级节点账户

接口描述 查询预定义超级节点账户。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法

```
GET /v1/xchain/network/defaultAccount
Host: bbe.bj.baidubce.com
Date: <Date>
Authorization: <Authorization_String>
```

- 请求头域 除公共头域外，无其它特殊头域。
- 请求参数 无

响应（Response）

- 响应头域 除公共头域外，无其它特殊头域。
- 响应参数

名称	类型	描述
address	String	超级链账户地址

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/xchain/network/defaultAccount HTTP/1.1
content-type: application/json
host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2019-01-06T06:55:42Z/1800/host/0b9f07c38e8f3815424002ef2026b2d66d73439f591d561b7d5bfa3fad8c558d
```

- 响应示例

```
200
Content-Length: 0
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 10:24:02 GMT

{
  "data": [
    {
      "address": "mdmdDqBAp2EBcU4QASG5D64M1SGoGbGNk"
    },
    {
      "address": "d6HjeHPMg9acCE9xs4ztmdEHDUMVps7mk"
    },
    {
      "address": "dppE8C1vyA9iL9ZtbLAF8UeSvSX728Kbd"
    }
  ]
}
```

获取代币总量

🔗 获取代币总量

接口描述 根据给定配置计算超级链网络能生成的超级币总量。

权限说明 请求发起人需要具有合法的AccessKeyID和SecretAccessKey才能发起请求。

注意事项 如果请求中没有用户验证信息（即匿名访问），返回403 Forbidden，错误信息：AccessDenied。

请求（Request）

- 请求语法


```
GET /v1/xchain/token/totalQuota
Host: bbe.bj.baidubce.com
Date: <Date>
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
Authorization: <Authorization_String>
```

- 请求头域

除公共头域外，无其它特殊头域。

- 请求参数

名称	类型	说明	是否必须
initialQuota	int	管理账户初始分配代币数，范围 [100000, 1000000000]，缺省100000，至少为award的100倍	必须
award	int	出块奖励，范围 [1000, 10000000]，缺省1000	必须
awardHeightGap	int	奖励衰减高度，范围 [1000, 10000000]，缺省1000	必须
awardRatio	float	奖励衰减系数，范围[0.5, 0.999]，缺省 0.5	必须

响应 (Response)

- 响应头域

除公共头域外，无其它特殊头域。

- 响应参数

名称	类型	描述
totalQuota	int64	生成的超级币总量

失败响应参考错误码部分。

示例

- 请求示例

```
GET /v1/xchain/token/totalQuota?awardRatio=0.9&awardHeightGap=1000&award=10000&initialQuota=1000000
HTTP/1.1
content-type: application/json
host: bbe.bj.baidubce.com
x-bce-date: 2019-01-06T06:55:42Z
x-bce-request-id: b9485394-c7dc-3cfa-d454-57c34773d08c
authorization: bce-auth-v1/5fded23b03594981872fbfadaad70ef6/2018-12-06T06:55:42Z/3600/host;x-bce-client-ip;x-bce-date;x-bce-request-id;x-bce-security-token/f86ff010bf7e8b45112357332630de7d48b6fed64f74d6b589c2fcd2e9be3c86
```

- 响应示例

```
200
Content-Length: 102
Content-Type: application/json; charset=utf-8
Date: Fri, 04 Jan 2019 10:24:02 GMT

{
  "data":{
    "totalQuota":101000000
  }
}
```

附录

公共部分

 类型说明

networkType

名称	描述
xChainExperience	超级链试用网络
fabricConsortiumOwner	创建Fabric联盟网络
fabricConsortiumParticipant	加入Fabric联盟网络
ether	以太坊网络
quorumConsortiumOwner	创建Quorum联盟网络
quorumConsortiumParticipant	加入Quorum联盟网络

subServiceType

名称	描述
XuperChainTrail	创建超级链试用网络
ConsortiumCreateDefaultConf	创建Fabric联盟普通配置
ConsortiumCreateLuxuryConf	创建Fabric联盟豪华配置
ConsortiumJoinDefaultConf	加入Fabric联盟普通配置
ConsortiumJoinLuxuryConf	加入Fabric联盟豪华配置
EthereumDefaultConf	以太坊普通配置
EthereumAdvancedConf	以太坊高级配置
EthereumLuxuryConf	以太坊豪华配置
QuorumDefaultConf	创建或加入Quorum联盟普通配置
QuorumLuxuryConf	创建或加入Quorum联盟豪华配置

以太坊相关

 类型说明

EthereumNetworkInfo

名称	类型	描述
name	String	网络名称
uuid	String	网络id
cceClusterId	String	CCE集群id
networkSize	Long	区块链节点个数
chainId	Long	区块链网络id
targetGasLimit	Long	目标GasLimit
maxPeers	Int	最大peer个数
verbosity	Int	日志等级
syncMode	String	区块链节点同步模式
gasPrice	Long	gas价格
difficulty	Long	初始挖矿难度
gasLimit	Long	初始GasLimit
genesisAccounts	[]EthereumAccount	创世块内初始账户列表
minerAccount	[]EthereumAccount	挖矿账户列表
accounts	[]EthereumAccount	钱包账户列表
gethNodes	EthereumNode	节点列表
state	String	网络状态
startTime	Date	网络创建时间
UserId	String	用户id
netstatsAddress	String	netstats地址
remixAddress	String	remix地址
explorerAddress	String	explorer地址
jsonrpcAddress	String	jsonrpc地址
deleted	String	网络是否被删除
errorMessage	String	错误信息

EthereumAccount

名称	类型	描述
address	String	以太坊账户地址
privateKey	String	以太坊账户私钥
balance	Long	以太坊账户余额

EthereumNode

参数名	类型	说明
ip	String	IP地址
jsonrpcPort	Int	jsonrpc端口
wsrpcPort	Int	wsrpc端口
ipcListenPort	Int	区块链节点ipc监听端口
ipcDiscoveryPort	Int	区块链节点ipc发现端口
role	String	区块链节点角色
desp	String	区块链节点描述

创建接口相关

🔗 类型说明

XChainExperienceForm

参数名	类型	说明
tpl	String	产品线：xplatform
address	String	用户地址
netType	String	网络类型：当前支持4
deployType	String	部署类型，部署方式：当前只支持00000010
techBased	String	技术类型，当前支持1
nodeObj	NodeObj	节点信息
netName	String	网络名称
netNameCn	String	网络中文名称
consensus	String	网络下链的共识机制：tdpos，single
isExplorer	String	是否购买浏览器服务，1，是，0否

FabricConsortiumForm

参数名	类型	说明
name	String	联盟名称
englishName	String	联盟英文名
channelPolicy	String	通道创建策略：all=需要所有人审批，any=只需要盟主审批
consensus	String	共识算法：kafka
ordererOrg	Integer	组织ID
description	Long	联盟描述

JoinFabricConsortiumForm

参数名	类型	说明
consortiumUuid	String	联盟Uuid
orgUuid	String	组织Uuid
approvalUuid	String	审批Uuid

EthereumForm

参数名	类型	说明
networkName	String	网络名称
nodeNumber	Integer	网络节点个数
maxPeers	Integer	最大peer节点连接数
syncMode	String	节点同步模式，包括full和fast两种模式
difficulty	Integer	创始块中的出块初始难度
verbosity	Integer	日志级别，0=silent, 1=error, 2=warn, 3=info, 4=debug, 5=detai
targetGasLimit	Long	最低gas限制
gasLimit	Long	一次交易可使用的Gas的上限
gasPrice	Long	单位gas对应的Token价格

QuorumConsortiumForm

参数名	类型	说明
name	String	联盟名称
domain	String	联盟英文名
orgUuid	String	初始联盟组织Uuid
description	String	联盟描述
nodeNumber	Integer	本组织交易节点数（只同步块而不参与出块的节点）
targetGasLimit	Long	最低gas限制
gasLimit	Long	一次交易可使用的Gas的上限
consensus	String	共识机制，可选IBFT
blockPeriod	Integer	出块周期(s)

JoinQuorumConsortiumForm

参数名	类型	说明
orgUuid	String	加入联盟组织Uuid
applicationUuid	String	申请Uuid
nodeNumber	Integer	本组织交易节点数（只同步块而不参与出块的节点）

BaseCreateOrderRequestVo

参数名	类型	说明
items	[]Item	bbe配置

Item

参数名	类型	说明
config	BbeConfig	bbe配置

BbeConfig

参数名	类型	说明
flavor	[]FlavorItem	flavor配置项

FlavorItem

参数名	类型	说明
name	String	配置项名称
value	String	配置项值

NodeObj

参数名	类型	说明
level	String	机器套餐，默认1
nodes	[]Node	节点列表

Node

参数名	类型	说明
id	String	节点类型
displayName	[]Node	节点名称
count	[]Node	机器数量

Quorum联盟相关

 类型说明

QuorumNetworkInfo

名称	类型	描述
name	String	联盟名称
domain	String	联盟英文名称
uuid	String	联盟Uuid
description	String	联盟描述
status	String	联盟状态，RUNNING表示运行中
createTime	String	联盟创建时间
consensus	String	共识机制
errorMessage	String	错误信息
genesisConfig	GenesisConfig	创世区块配置信息
nodeConfig	NodeConfig	Quorum节点配置信息
members	Members	联盟成员信息

GenesisConfig

名称	类型	描述
config	Object	配置信息
nonce	String	随机数
timestamp	String	UTC时间戳
extraData	String	额外数据
gasLimit	String	初始GasLimit
difficulty	String	初始挖矿难度
mixHash	String	混淆Hash
coinbase	String	初始矿工地址
alloc	Object	初始账户资产配置
number	String	表示区块在整个区块链中的位置
gasUsed	String	创世区块内所有Transaction执行时消耗的Gas总和
parentHash	String	创世区块的父区块的哈希值

NodeConfig

名称	类型	描述
targetGasLimit	Long	目标GasLimit
maxPeer	Integer	最大peer节点连接数
syncMode	String	同步模式
gasPrice	Integer	单位gas对应的Token价格
blockPeriod	Integer	出块周期(s)
istanbulRequestTimeout	Integer	共识请求超时时间
cpuLimit	Integer	节点最大使用CPU，0表示不限制
memoryLimit	Integer	节点最大使用内存，0表示不限制

Members

参数名	类型	说明
uuid	String	联盟成员Uuid
orgUuid	String	联盟成员所在组织Uuid
orgName	String	组织名称
orgDomain	String	组织英文名称
consortiumUuid	String	联盟Uuid
consortiumName	String	联盟名称
consortiumDomain	String	联盟英文名称
userId	String	用户Id
cluster_id	String	集群Id
validatorNumber	Integer	验证节点个数
transactionNodeNumber	Integer	交易节点个数
joinedTime	String	加入联盟时间
status	String	联盟成员状态，RUNNING表示运行中
errMessage	String	错误信息

QuorumNodeInfo

名称	类型	描述
uuid	String	节点Uuid
shortId	String	节点短id
orgUuid	String	组织Uuid
consortiumUuid	String	联盟Uuid
consortiumMemberUuid	String	联盟成员Uuid
userId	String	用户id
cluster_id	String	集群id
type	String	节点类型，validatorNode表示验证节点，transactionNode表示交易节点
nodeConfig	NodeConfig	Quorum节点配置信息
coinbase	String	节点矿工地址
keystore	String	节点矿工地址keystore
contractKey	String	隐私合约交易地址
jsonRPC	String	jsonRPC地址
contractEditorUrl	String	合约编辑器地址
blockChainBrowserUrl	String	区块链浏览器地址
nodeConsoleUrl	String	节点控制台地址
status	String	节点状态，RUNNING表示运行中
createTime	String	节点创建时间

OrgInfo

名称	类型	描述
uuid	String	组织Uuid
name	String	组织名称
domain	String	组织英文名称
description	String	组织描述
userId	String	用户id
cert	String	cert证书
createTime	String	组织创建时间

ApplicationsInfo

名称	类型	描述
uuid	String	申请Uuid
consortiumUuid	String	联盟Uuid
target	String	目标
targetUuid	String	目标Uuid
initiator	String	发起人
initiatorUuid	String	发起人Uuid
userId	String	用户Id
orgUuid	String	组织Uuid
type	String	事件类型
state	String	事件状态，DOING表示进行中
createTime	String	创建时间
done	Bool	事件是否已完成，true表示完成，false表示未完成

ApprovalsInfo

名称	类型	描述
uuid	String	审批Uuid
applicationUuid	String	申请Uuid
target	String	目标
targetUuid	String	目标Uuid
initiator	String	发起人
initiatorUuid	String	发起人Uuid
userId	String	用户Id
orgUuid	String	组织Uuid
orgName	String	组织名称
type	String	事件类型
state	String	事件状态，DOING表示进行中
message	String	消息
createTime	String	创建时间
finishTime	String	结束时间
done	Bool	事件是否已完成，true表示完成，false表示未完成

Fabric联盟相关

🔗 类型说明

🔗 申请信息（Application）

字段	类型	说明
Uuid	string	对象唯一标识
OrgUuid	string	组织uuid
OrgName	string	组织名字
ConsortiumUuid	string	联盟Uuid
ConsortiumName	string	联盟名字
ChannelUuid	string	通道Uuid
ChannelName	string	通道名字
Tag	int	事件类型，枚举，包括以下几种：1 联盟, 2 Channel, 3 Chaincode, etc
Type	int	申请类型，枚举，包括以下几种：10 邀请用户加入联盟, 20 创建通道, 21 申请加入通道
InviteUserId	string	邀请用户加入联盟时用户的id
Message	string	申请信息
State	string	申请状态，枚举，包括以下几种：Processing 进行中, Approved 通过, Rejected 拒绝
ApproveCnt	int	已同意数量
RejectCnt	int	已拒绝数量
TotalCnt	int	总审批数
TargetRate	float32	审批通过水位
CreateTime	datetime	申请创建时间
FinishTime	datetime	申请结束时间
ObjectState	string	申请对象操作状态

审批信息 (Approval)

字段	类型	说明
Uuid	string	对象唯一标识
ApplicationUuid	string	申请uuid
OrgName	string	这个审批相关联的申请发起者的名字
ApproverUuid	string	需要审批的组织uuid或用户的uuid(联盟只能邀请用户加入，审批角色只能是人)
ApproverName	string	审批者的名字
ConsortiumUuid	string	联盟Uuid
ConsortiumName	string	联盟名字
ChannelUuid	string	通道Uuid
ChannelName	string	通道名字
Message	string	申请信息
tag	int	事件类型（1：联盟，2：channel, 3: Chaincode)
State	string	审批状态，枚举，包括以下几种：Processing 进行中, Approved 通过, Rejected 拒绝
Feedback	string	反馈的意见
CreateTime	datetime	审批发起时间
FinishTime	datetime	审批结束时间
ObjectState	string	申请对象操作状态

🔗 事件日志（Event）

字段	类型	说明
Uuid	string	对象唯一标识
FromUuid	string	发起方的uuid（组织或者用户的uuid ---- 联盟邀请用户，用户拒绝时只能用用户ID来通知联盟）
FromName	string	发起方的名字
ToUuid	string	接收方组织的uuid
ConsortiumName	string	联盟名字
ChannelName	string	通道名字
Message	string	事件内容
tag	int	事件类型（1：联盟，2：channel, 3: Chaincode）
State	int	日志状态，枚举，包括以下几种：0 未读, 1 已读
CreateTime	datetime	事件发生时间

🔗 链码信息

参数名	类型	说明
name	String	chaincode名称
uuid	String	chaincode唯一标识id
channelUuid	String	channel唯一标识id
networkUuid	String	channel关联network唯一标识id
orgName	[]String	chaincode关联的机构名列表
version	String	chaincode版本
state	int	chaincode状态码，0: '链码已上传'，1: '正在安装'，2: '安装完成'，3: '安装出错'，4: '正在实例化'，5: '运行中'，6: '实例化出错'，7: '正在升级'，8: '升级时出错'
path	String	链码代码包路径
endorsementPolicy	String	背书策略
initArgs	bool	链码参数
createTime	String	创建时间
instantiateTime	String	实例化时间
updatable	bool	是否可升级
errMsg	String	最近一次chaincode操作的结果，内容为json格式kv值，如成功为{"orgname": "success"}

功能更新记录

🔗 2019-01-09

- openapi alpha 版本

常见问题

🔗 XUPERBAAS常见问题

1. XUPERBAAS 如何处理欠费网络？

XUPERBAAS 会保留欠费网络七天，七天后会自动删除网络。欠费期间网络不可用，充值后网络才可恢复使用。

🔗 Fabric常见问题

1. Fabric 盟主是什么概念？

Fabric 盟主是联盟的创建者，拥有联盟中的orderer组织及其节点。后续加入联盟的成员只有peer节点且连接盟主的orderer节点。

2. Fabric 支持组建联盟吗？成员数量有限制吗？

XUPERBAAS Fabric是支持多用户组建联盟的，而且对联盟的数量没有限制。 [查看详情](#)

3. 一个用户可以组建多组织Fabric联盟吗？

XUPERBAAS Fabric 暂时只支持多用户组建联盟，联盟中只允许包含某个用户一个组织。

4. Fabric 通道创建策略（ANY/ALL）什么作用？

ANY(任一组织同意)：创建新通道时，只需任意参与组织的同意；ALL(全部组织同意)：创建新通道时，需要所有参与组织的同意。

5. Fabric 如何创建一个多组织的通道？

有两种方式，一是在创建通道时选择多个组织加入通道；二是后续组织可以加入已创建的通道。 [查看详情](#)

6. Fabric 链码支持什么语言？如何打包链码？

Fabric 目前只支持golang语言的链码。链码引用的第三方库均放入vendor中，然后将链码外层文件夹打包成tar.gz或zip。

7. Fabric 链码背书策略如何填写？

XUPERBAAS Fabric 链码背书策略完全遵循官方原则只是对书写方式进行了优化：鼠标点击填写框自动出现联盟中组织的名称和角色，您可直接选择。背书策略也可为空，即：联盟中任意成员背书即可。 [查看详情](#)

8. Fabric 如何升级链码？

首先，XUPERBAAS Fabric支持升级链码。您先本地打包好待升级的链码，在链码管理页中上传同名不同版本的链码，安装成功后会提示升级操作。 [查看详情](#)

9. Fabric 如何调用部署的链码？

进入联盟详情页下载SDK，里面包含网络的成员配置文件和msp文件，还有各个语言版本的Demo，您可以仿照Demo编写自己的服务调用链码。

10. Fabric 证书管理如何使用？

Fabric 网络中用户调用链码都需要一套msp client证书，除下载sdk时有一套默认证书（User1）您还可以在证书管理中自定义一套client证书。借助自定义证书可实现abac权限管理功能。 [查看详情](#)

11. Fabric 为何无法删除联盟？

只有联盟成员全部退出联盟后盟主才可删除联盟；如果联盟接入跨链服务，需要解除跨链服务后才可退出或删除。

12. 为什么Fabric 浏览器无法访问？

创建通道后才会自动创建浏览器（数分钟启动时间），请先确认是否已创建通道。

13. 创建好的网络支持扩容数据盘吗

Fabric和Quorum支持数据盘扩容。

🔗 以太坊常见问题

1. XUPERBAAS中的以太坊和公链以太坊有什么区别？

XUPERBAAS中的以太坊是使用公链以太坊的开源代码部署的一条私有链。在数据上，与公链相互隔离；在使用上，与公链一致。

2. 公链上的以太坊账号（地址和私钥）可以在XUPERBAAS的以太坊中使用吗？

可以使用，但该地址在公链中的资产和交易数据在XUPERBAAS的以太坊中不存在。

3. XUPERBAAS中的以太坊如何获取以太币？

XUPERBAAS中，创建以太坊网络时，会创建一些预定义账号并分配有以太币。

4. 如何在以太坊网络中部署合约？

部署合约有两种方式：

- 方式一：通过remix部署

XUPERBAAS为创建的以太坊网络提供了remix IDE，可以在remix中调试、编译和部署智能合约。

- 方式二：通过jsonrpc部署

XUPERBAAS中的以太坊网络对外提供了jsonrpc地址，用户可以通过该地址调用网络，包括部署合约。

5. 以太坊网络支持websocket链接吗？ 不支持。目前只支持http协议的jsonrpc链接。

🔗 Quorum常见问题

1. 什么是Quorum？

J.P. Morgan在2016年启动了开源区块链项目 Quorum。它是在以太坊(Ethereum)的基础上开发的，提供私有智能合约执行方案，并满足企业级的性能要求。

适用于需要高速交易以及高吞吐量处理联盟间进行私有交易的应用场景，主要是为了解决区块链技术在金融及其他行业应用的特殊挑战而设计。

2. XUPERBAAS中的Quorum和开源Quorum有什么区别？

XUPERBAAS中的Quorum在开源Quorum的基础上，引入了自研的网关节点，对内作为一个虚拟的区块链节点只转发交易，对外统一提供本组织入口。

改进后有以下优势：

- 组织间只需网关节点互连即可，即对外只需暴露一个端口，解除两两互连的限制；
- 网关节点间通过双向认证和加密来建立一条安全隧道，通过该隧道实现联盟内跨组织的节点间的间接互通，更加安全、可控；
- 组织纬度的准入控制，节点只需把自身组织的网关节点加入准入列表；
- 联盟链内组织的节点运行在一个网络隔离的内网环境中，从一个组织内的节点上只能够获取到组织内的节点间的网络拓扑，保证Quorum节点网络拓扑结构不泄漏。
- 私有交易以组织为维度，由网关节点进行组织内部私有交易的转发，隐私交易更加便捷。

3. Quorum网络如何进行联盟管理？

XUPERBAAS的Quorum联盟链强化了联盟管理功能，针对联盟链：

- 抽象联盟概念，并支持通过申请审批流加入联盟，通过投票从联盟中剔除组织；

- 抽象组织概念，并支持组织及其成员管理，包括成员的创建、删除、赋权等；
- 支持同平台内构建联盟链；
- 灵活的组织准入CA证书控制。

4. 如何在Quorum网络中部署智能合约？

在Quorum网络中部署智能合约有两种方式：

- 通过智能合约编辑器部署；
- 通过节点jsonrpc部署。

服务等级协议SLA

超级链BaaS平台服务等级协议SLA

协议生效时间：2022年03月01日

本服务等级协议（Service Level Agreement，以下简称“SLA”）规定了百度智能云向客户提供的超级链BaaS平台（Baidu Xuper BaaS）的服务可用性等级指标及赔偿方案。

1. 定义

1.1 服务周期

一个服务周期为一个自然月。

1.2 服务周期总分钟数

按照服务周期内的总天数 * 24（小时）* 60（分钟）计算。

1.3 服务不可用分钟数

因超级链BaaS平台服务系统自身导致已成功创建的XuperChain、Fabric、Quorum、以太坊、IPFS、存证平台和跨链服务等服务异常而无法正常使用，视为服务不可用。不可用分钟数=不可用解决时间-不可用开始时间，按分钟计算，不足1分钟的按1分钟来计算。

1.4 月度服务费用

客户在一个自然月中就单个账号超级链BaaS平台服务所支付的服务费用总额，如果客户一次性支付了多个月份的服务费用，则将按照所购买的月数分摊计算月度服务费用。

2. 服务可用性

2.1 服务可用性计算方式

服务可用性 = (服务周期总分钟数 - 服务不可用分钟数) / 服务周期总分钟数 * 100%

2.2 服务可用性承诺

百度智能云超级链BaaS平台承诺99.9%的服务可用性。如超级链BaaS平台服务未达到上述服务可用性承诺，客户可以根据本协议第三条约定获得赔偿。赔偿范围不包括以下原因所导致的不可用：(1) 百度智能云预先通知用户后进行系统维护所引起的，包括割接、维修、升级和模拟故障演练；(2) 任何百度智能云所属设备以外的网络、设备故障或配置调整引起的；(3) 用户的应用程序或数据信息受到黑客攻击而引起的；(4) 用户维护不当或保密不当致使数据、口令、密码等丢失或泄漏所引起的；(5) 用户的疏忽或由用户授权的操作所引起的；(6) 不可抗力以及意外事件引起的；(7) 超级链BaaS平台服务只赔付超级链BaaS平台自身，不赔付超级链BaaS平台服务所关联的其他云服务；(8) 其他非百度智能云原因所造成的超级链BaaS平台服务无法正常使用。

3.赔偿条款

3.1 赔偿标准

根据超级链BaaS平台月度服务可用性，按照下表中的标准计算赔偿金额，赔偿方式仅限于用于购买超级链BaaS平台产品的代金券，且赔偿总额不超过未达到服务可用性承诺当月客户就该超级链BaaS平台支付的月度服务费用。

服务可用性	赔偿代金券金额
低于99.9%但等于或高于99.00%	月度服务费用的10%
低于99.00%但等于或高于95.00%	月度服务费用的25%
低于95.00%	月度服务费用的100%

3.2 赔偿申请时限

客户可以在每月第五（5）个工作日对上个月没有达到可用性的实例提出赔偿申请。赔偿申请必须限于在超级链BaaS平台服务没有达到可用性的相关月份结束后两（2）个月内提出。超出申请时限的赔偿申请将不被受理。

4.其他说明

- (1) 在法律法规允许的范围内，百度智能云负责对本协议进行解释说明。
- (2) 本协议一经公布立即生效，百度智能云有权对本SLA条款作出修改。如本SLA条款有任何修改，百度智能云将以网站公示或发送邮件的方式通知您。如您不同意百度智能云对SLA所做的修改，您有权停止使用超级链BaaS平台服务，如您继续使用超级链BaaS平台服务，则视为您接受修改后的SLA。
- (3) 协议项下百度智能云对于用户所有的通知均可通过网页公告、站内信、电子邮件、手机短信或其他形式等方式进行；该等通知于发送之日视为已送达收件人。因用户未及时获知百度智能云的服务变更或终止条款遭受损失的，百度智能云不承担任何责任。
- (4) 本协议的订立、执行和解释及争议的解决均应适用中国法律并受中国法院管辖。如双方就本协议内容或其执行发生任何争议，双方应尽量友好协商解决；协商不成时，任何一方均可向北京市海淀区人民法院提起诉讼。
- (5) 本协议构成双方对本协议之约定事项及其他有关事宜的完整协议，除本协议规定的之外，未赋予本协议各方其他权利。
- (6) 如本协议中的任何协议无论因何种原因完全或部分无效或不具有执行力，本协议的其余协议仍应有效并且有约束力。
- (7) 关于用户约束条款，详见《[百度智能云用户服务协议](#)》中的"用户的权利与义务"相关条款内容。
- (8) 关于服务商免责条款，详见《[百度智能云用户服务协议](#)》中的"免责声明"相关条款内容。