

区块链服务

开发指南

文档版本 01
发布日期 2024-01-11



版权所有 © 华为云计算技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目录

1 简介	1
2 链代码开发	3
2.1 开发前准备	3
2.2 开发规范	5
2.3 Go 语言链代码开发	6
2.3.1 链代码结构	6
2.3.2 链代码相关的 API	8
2.3.3 链代码示例（1.4 风格）	8
2.3.4 链代码示例（2.0 风格）	11
2.3.5 链代码调测	13
2.4 Java 语言链代码开发	15
2.4.1 链代码结构	15
2.4.2 链代码相关的 API	16
2.4.3 链代码示例	16
2.4.4 链代码调测	18
3 应用程序开发	22
3.1 概述	22
3.2 开发前准备	22
3.3 应用程序开发	22
4 示例 Demo	24
4.1 GO SDK Demo	24
4.2 Java SDK Demo	30
4.3 Gateway Java Demo	37
4.4 REST API Demo	38
5 区块链中间件接口	48
5.1 概述	48
5.2 链代码调用（公测）	49
5.3 链代码管理（公测）	52
5.3.1 获取 Token	52
5.3.2 安装链代码	53
5.3.3 实例化链代码	56
5.3.4 获取安装的链码列表	59

5.3.5 查询指定链码版本信息.....	63
5.3.6 查询链代码安装信息.....	66
5.3.7 查询链代码实例化信息.....	68
5.3.8 查询应用链信息.....	71
5.3.9 查询区块列表.....	75
5.3.10 查询交易列表.....	77
5.3.11 查询交易总数.....	80
5.3.12 查询区块交易列表.....	82
5.3.13 查询交易详情.....	86
5.3.14 查询节点状态.....	89
5.3.15 删除链代码.....	92
5.3.16 下载报告.....	94
5.4 分布式身份（公测）.....	96
5.4.1 概述.....	96
5.4.2 分布式身份(DID)管理.....	98
5.4.2.1 企业身份注册(带有 service).....	99
5.4.2.2 注册 DID.....	102
5.4.2.3 更新 DID.....	104
5.4.2.4 查询 DID.....	108
5.4.3 可验证凭证(VC)管理.....	112
5.4.3.1 发布可验证凭证的模板.....	112
5.4.3.2 查询凭证模板.....	116
5.4.3.3 申请可验证凭证.....	118
5.4.3.4 签发者确认凭证已签发.....	120
5.4.3.5 查询凭证申请订单.....	122
5.4.3.6 查询待处理的申请订单.....	124
5.4.3.7 签发可验证凭证.....	127
5.4.3.8 根据索引查询可验证凭证.....	130
5.4.3.9 验证凭证.....	133
5.5 可信数据交换（公测）.....	135
5.5.1 概述.....	135
5.5.2 数据集管理.....	138
5.5.2.1 发布数据集.....	138
5.5.2.2 删除数据集.....	143
5.5.2.3 关闭数据集.....	145
5.5.2.4 查询指定数据集.....	147
5.5.2.5 查询数据集列表.....	150
5.5.2.6 主动分享数据集.....	154
5.5.2.7 获取数据解密后的明文.....	160
5.5.2.8 提取文件中的暗水印.....	162
5.5.2.9 查询指定的数据集分享流程.....	163
5.5.2.10 查询指定流程创建者的所有流程.....	166

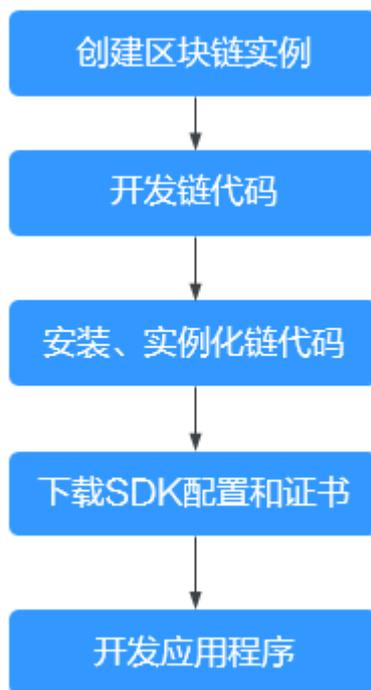
5.5.3 数据订单管理.....	169
5.5.3.1 申请数据集.....	169
5.5.3.2 授权数据集.....	172
5.5.3.3 修改订单状态.....	174
5.5.3.4 删除订单.....	176
5.5.3.5 查询指定订单.....	178
5.5.3.6 查询订单列表.....	181
5.5.4 属性加密的密钥管理.....	184
5.5.4.1 初始化 ABE 主密钥.....	184
5.5.4.2 更新 ABE 主密钥.....	186
5.5.4.3 查询 ABE 主密钥.....	188
5.5.4.4 申请 ABE 用户密钥.....	190
5.5.4.5 授权 ABE 用户密钥.....	193
5.5.4.6 查询 ABE 用户密钥申请.....	196
5.5.4.7 ABE 用户密钥解密数据.....	198
6 附录.....	201
6.1 国密加密.....	201
6.1.1 概述.....	201
6.1.2 SDK 的使用.....	202
6.1.3 附录.....	202
6.2 同态加密.....	203
6.2.1 概述.....	204
6.2.2 同态加密库的使用.....	204
6.2.3 AHE Lib 库接口.....	206
6.2.4 Chaincode 库接口.....	210
6.2.5 IDChaincode.....	211
6.2.6 链代码示例.....	212
6.2.7 应用示例.....	213
6.2.8 同态加密交易验证 Demo.....	216
6.3 区块链应用低代码开发功能.....	229
6.3.1 概述.....	229
6.3.2 低代码合约开发.....	230
6.3.3 业务流程管理.....	230
6.3.4 使用说明.....	231
6.4 错误码.....	235

1 简介

在使用区块链服务时，您需要开发自己的链代码和应用。本文档主要介绍链代码的开发及其应用配置，支持具备Go/Java开发经验的开发人员使用。

区块链服务使用流程如下：

图 1-1 使用流程



1. 购买区块链实例
区块链实例支持在CCE集群上部署，具体可参见[基于CCE集群](#)。
2. 开发链代码
链代码是用Go、Java或Node.js语言编写的程序，主要用于操作账本上的数据，具体可参见[链代码开发](#)。
3. 安装、实例化链代码

区块链服务为您提供界面化链代码管理功能，包括链代码安装、实例化等，具体可参见[链代码管理](#)。

4. 下载SDK配置和证书

应用程序开发前，您需要获取对应实例的SDK配置文件和证书，具体可参见[下载SDK配置和证书](#)。

5. 开发应用程序

您需要自行开发应用程序业务逻辑代码。应用开发过程中可使用BCS提供的[国密加密SDK](#)，也可使用Fabric官方社区提供的和您自身的实例版本匹配的SDK，具体可参见[应用程序开发](#)。

另外，区块链实例提供同态加密库供您使用，相关资料及资源请参见[同态加密](#)。

 **说明**

如果您对业务链代码和客户端APP的设计和开发有需求，可以联系华为云区块链合作伙伴提供进一步服务，华为云区块链合作伙伴会结合您的业务以及华为云的优势和特点为您提供完善的解决方案，联系邮箱如下：sales@huaweicloud.com

2 链代码开发

2.1 开发前准备

链代码（Chaincode）又称智能合约，是用Go、Java或Node.js语言编写的程序，主要用于操作账本上的数据。链代码是运行在区块链上的、特定条件下自动执行的代码逻辑，是用户利用区块链实现业务逻辑的重要途径。基于区块链特点，智能合约的运行结果是可信的，其结果是无法被伪造和篡改的。

在使用区块链服务BCS时，用户需要开发自己的链代码和应用程序。用户的应用程序通过区块链网络中的Peer节点调用链代码，用户链代码通过区块链网络的Peer节点来操作账本数据。

说明

智能合约由用户自行编写上传并保证安全，请务必注意命令注入等相关安全问题。

开发环境准备

请根据自身业务选择Go或Java开发环境。

Go开发环境准备：

1. 安装Go开发环境。安装包下载地址为：<https://go.dev/dl/>。（请选择1.9.2之后的版本）

各个系统对应的包名（以1.11.12版本为例）：

操作系统	包名
Windows	go1.11.12.windows-amd64.msi
Linux	go1.11.12.linux-amd64.tar.gz

- Windows下您可以使用.msi后缀的安装包来安装。默认情况下.msi文件会安装在“C:\Go”目录下。您可以将“C:\Go\bin”目录添加到Path环境变量中。添加后您需要重启命令窗口才能生效。
- Linux下，您需要将下载的二进制包解压至/usr/local目录。将/usr/local/go/bin目录添加至Path环境变量：

```
export PATH=$PATH:/usr/local/go/bin
```

安装完go语言后可以通过命令**go version**查看版本信息，以及通过**go env**命令来查看相关路径配置。

2. 安装Go编辑器。编辑器可自行选择，推荐使用Goland：<https://www.jetbrains.com/go/download>。

Java开发环境准备：

仅适用于Fabric架构版本的区块链实例。

1. 安装Java开发环境。下载JDK并安装（建议选择最新版本）：<https://www.oracle.com/technetwork/java/javase/downloads/index.html>。

各个系统对应的包名（以15.0.2版本为例）：

操作系统	包名
Windows	jdk-15.0.2_windows-x64_bin.exe
Linux	jdk-15.0.2_linux-x64_bin.tar.gz

- Windows下您可以使用 .exe后缀的安装包来安装。
- Linux下，您需要将下载的二进制包解压至/usr/local目录。

```
export PATH=$PATH:/usr/local/go/bin
```

配置环境变量（若无则新建）：

- JAVA_HOME为jdk安装目录如“C:\Program Files (x86)\Java\jdk1.8.0_91”或“/usr/java/jdk1.8.0_91”（以下均略去双引号）；
- CLASSPATH为“.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;”；
- 在Path中新增两条“%JAVA_HOME%\bin”和“%JAVA_HOME%\jre\bin”。

安装完jdk后，可以通过命令**java -version**查看版本信息。

2. 安装Java编辑器。编辑器可自行选择，推荐使用IntelliJ IDEA。

下载源码包

下载Fabric源码包作为三方库。

请根据实际需求，选择下载对应版本的Fabric源码包：

<https://github.com/hyperledger/fabric/tree/release-2.2>

📖 说明

Fabric源码包选择和创建的区块链实例版本对应，即如果创建区块链实例时，Hyperledger Fabric增强版内核是v2.2（4.X.X版本），则Fabric源码包对应选择2.2版本。

2.2 开发规范

防止出现 panic 后链代码容器异常

📖 说明

该内容仅适用于Go语言链代码开发。

为避免出现panic异常时链代码容器异常重启，找不到日志，导致问题无法及时定位，可在Invoke函数入口处添加defer语句时，出现panic异常时返回错误给客户端。

```
// 定义命名返回值，发生panic在defer里面赋值，确保客户端可以收到返回值
// 使用debug.PrintStack()将错误的堆栈信息打印到标准输出，方便问题定位
func (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface) (pr pb.Response) {
    defer func() {
        if err:=recover(); err != nil {
            fmt.Println("recover from invoke:", err)
            debug.PrintStack()
            pr = shim.Error(fmt.Sprintf("invoke panic. err: %v",err))
        }
    }()

    fmt.Println("ex02 Invoke")
    function, args := stub.GetFunctionAndParameters()
    if function == "invoke" {
        // Make payment of X units from A to B
        return t.invoke(stub, args)
    } else if function == "delete" {
        // Deletes an entity from its state
        return t.delete(stub, args)
    } else if function == "query" {
        // the old "Query" is now implemented in invoke
        return t.query(stub, args)
    }

    pr = shim.Error("Invalid invoke function name. Expecting \"invoke\" \"delete\" \"query\"")
    return pr
}
```

分批次查询数据

查询账本数据时，如果在一次查询中返回过多的数据，会导致资源占用过多，接口延时较长（超过30s时peer会中断任务），应预先估计数据量，分批次进行查询。

修改或删除账本数据调用链代码时，同样也需要根据数据量大小确定是否采取分批次操作的方式处理。

合理使用索引（账本数据存储方式为 CouchDB）

对于CouchDB来说，使用索引功能，写入数据时会消耗时间，但可明显提高数据查询速度。因此可以根据业务需要，合理的在某些字段上建立索引。

添加权限验证

对智能合约执行者的权限进行验证，防止无权限的用户执行链代码。

📖 说明

如果业务上不要求确定的某个组织进行背书，为确保链代码上的数据不被任意组织恶意修改（自己安装非法链代码，操作数据等），建议至少两个或两个以上组织共同参与背书。

参数校验

参数（包括入参和代码中定义的各种参数）在使用前需对其个数、类型、长度、取值范围等做校验，验证其合法性，防止出现数组越界等问题。

日志处理

开发时需要对业务逻辑复杂、容易出错的地方，使用fmt打印日志，便于调测。但是在开发调测结束后，需要将其删除，因为fmt会消耗时间和资源。

依赖配置

📖 说明

该内容仅适用于Java语言链代码开发。

请使用Gradle或Maven构建管理工具组织链代码项目。若链代码项目中包含非本地依赖，请确保对应区块链实例的节点均绑定了弹性ip。若链代码容器将运行在受限网络环境，请确保项目中的所有依赖已配置为本地依赖。示例链代码获取方法：登录区块链服务BCS控制台，进入“应用案例”，单击“Java示例Demo-Java SDK Demo”中“Chaincode_Java_Local_Demo”的“下载”按钮。

2.3 Go 语言链代码开发

2.3.1 链代码结构

本章以Go语言为例来介绍。链代码即一个Go文件，创建好文件后进行函数开发等操作。

当前支持两种方式编写链码：1.4风格（使用shim包）和2.2风格（使用fabric-contract-api-go包）。

📖 说明

区块链服务BCS支持使用两种风格编写的链代码。

链代码接口

- 链代码启动必须通过1.4风格（调用shim包）中的Start函数，入参为shim包中定义的Chaincode接口类型。实际开发中，您需要自行定义一个结构体，实现Chaincode接口。

```
type Chaincode interface {
    Init(stub ChaincodeStubInterface) pb.Response
    Invoke(stub ChaincodeStubInterface) pb.Response
}
```

- 2.2风格（使用fabric-contract-api-go包）的链代码实际开发中，您需要自行定义一个结构体，实现Chaincode接口。

```
type Chaincode interface {
    Init(ctx contractapi.TransactionContextInterface, args...) error
    Invoke(ctx contractapi.TransactionContextInterface, args...) error
}
```

链代码结构

- 1.4风格Go语言的链代码结构如下:

```
package main

// 引入必要的包
import (
    "github.com/hyperledger/fabric/core/chaincode/shim"
    pb "github.com/hyperledger/fabric/protos/peer"
)

// 声明一个结构体
type SimpleChaincode struct {}

// 为结构体添加Init方法
func (t *SimpleChaincode) Init(stub shim.ChaincodeStubInterface) pb.Response {
    // 在该方法中实现链代码初始化或升级时的处理逻辑
    // 编写时可灵活使用stub中的API
}

// 为结构体添加Invoke方法
func (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface) pb.Response {
    // 在该方法中实现链代码运行中被调用或查询时的处理逻辑
    // 编写时可灵活使用stub中的API
}

//主函数，需要调用shim.Start()方法
func main() {
    err := shim.Start(new(SimpleChaincode))
    if err != nil {
        fmt.Printf("Error starting Simple chaincode: %s", err)
    }
}
```

- 2.2风格Go语言的链代码结构如下:

```
package main

// 引入必要的包
import (
    "github.com/hyperledger/fabric/plugins/fabric-contract-api-go/contractapi"
)

// 声明一个结构体
type Chaincode struct {
    contractapi.Contract
}

// 为结构体添加Init方法
func (ch *Chaincode) Init(ctx contractapi.TransactionContextInterface, args...) error {
    // 在该方法中实现链代码初始化或升级时的处理逻辑
}

// 为结构体添加Invoke方法
func (ch *Chaincode) Invoke(ctx contractapi.TransactionContextInterface, args...) error {
    // 在该方法中实现链代码运行中被调用或查询时的处理逻辑
}

//主函数
func main() {
    cc, err := contractapi.NewChaincode(new(ABStore))
    if err != nil {
        panic(err.Error())
    }
    if err := cc.Start(); err != nil {
        fmt.Printf("Error starting ABStore chaincode: %s", err)
    }
}
```

2.3.2 链代码相关的 API

Fabric源码包中的shim包提供了如下几种类型的接口，您可以参考使用：

- 参数解析API：调用链代码时需要给被调用的目标函数/方法传递参数，该API提供解析这些参数的方法。
- 账本状态数据操作API：该API提供了对账本数据状态进行操作的方法，包括对状态数据的查询及事务处理等。
- 交易信息获取API：获取提交的交易信息的相关API。
- 对PrivateData操作的API：Hyperledger Fabric在1.2.0版本中新增的对私有数据操作的相关API。
- 其他API：其他的API，包括事件设置、调用其他链代码操作。

2.3.3 链代码示例（1.4 风格）

如下是一个账户转账的链代码示例（1.4风格）仅供安装实例化，若您需要调测请参考[Fabric官方示例](#)中的链代码。

```
package main

import (
    "fmt"
    "strconv"
    "github.com/hyperledger/fabric/core/chaincode/shim"
    pb "github.com/hyperledger/fabric/protos/peer"
)

type SimpleChaincode struct {
}

// 初始化数据状态，实例化/升级链代码时被自动调用
func (t *SimpleChaincode) Init(stub shim.ChaincodeStubInterface) pb.Response {
    // println函数的输出信息会出现在链代码容器的日志中
    fmt.Println("ex02 Init")

    // 获取用户传递给调用链代码的所需参数
    _, args := stub.GetFunctionAndParameters()

    var A, B string // 两个账户
    var Aval, Bval int // 两个账户的余额
    var err error

    // 检查合法性，检查参数数量是否为4个，如果不是，则返回错误信息
    if len(args) != 4 {
        return shim.Error("Incorrect number of arguments. Expecting 4")
    }

    A = args[0] // 账户A用户名
    Aval, err = strconv.Atoi(args[1]) // 账户A余额
    if err != nil {
        return shim.Error("Expecting integer value for asset holding")
    }

    B = args[2] // 账户B用户名
    Bval, err = strconv.Atoi(args[3]) // 账户B余额
    if err != nil {
        return shim.Error("Expecting integer value for asset holding")
    }

    fmt.Printf("Aval = %d, Bval = %d\n", Aval, Bval)

    // 将账户A的状态写入账本中
    err = stub.PutState(A, []byte(strconv.Itoa(Aval)))
    if err != nil {
```

```
        return shim.Error(err.Error())
    }

    // 将账户B的状态写入账本中
    err = stub.PutState(B, []byte(strconv.Itoa(Bval)))
    if err != nil {
        return shim.Error(err.Error())
    }

    return shim.Success(nil)
}

// 对账本数据进行操作时(query, invoke)被自动调用
func (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface) pb.Response {
    fmt.Println("ex02 Invoke")
    // 获取用户传递给调用链代码的函数名称及参数
    function, args := stub.GetFunctionAndParameters()

    // 对获取到的函数名称进行判断
    if function == "invoke" {
        // 调用 invoke 函数实现转账操作
        return t.invoke(stub, args)
    } else if function == "delete" {
        // 调用 delete 函数实现账户注销
        return t.delete(stub, args)
    } else if function == "query" {
        // 调用 query 实现账户查询操作
        return t.query(stub, args)
    }
    // 传递的函数名出错, 返回 shim.Error()
    return shim.Error("Invalid invoke function name. Expecting \"invoke\" \"delete\" \"query\"")
}

// 账户间转账
func (t *SimpleChaincode) invoke(stub shim.ChaincodeStubInterface, args []string) pb.Response {
    var A, B string // 账户A和B
    var Aval, Bval int // 账户余额
    var X int // 转账金额
    var err error

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

    A = args[0] // 账户A用户名
    B = args[1] // 账户B用户名

    // 从账本中获取A的余额
    Avalbytes, err := stub.GetState(A)
    if err != nil {
        return shim.Error("Failed to get state")
    }
    if Avalbytes == nil {
        return shim.Error("Entity not found")
    }
    Aval, _ = strconv.Atoi(string(Avalbytes))

    // 从账本中获取B的余额
    Bvalbytes, err := stub.GetState(B)
    if err != nil {
        return shim.Error("Failed to get state")
    }
    if Bvalbytes == nil {
        return shim.Error("Entity not found")
    }
    Bval, _ = strconv.Atoi(string(Bvalbytes))

    // X为转账金额
    X, err = strconv.Atoi(args[2])
```

```
if err != nil {
    return shim.Error("Invalid transaction amount, expecting a integer value")
}
// 转账
Aval = Aval - X
Bval = Bval + X
fmt.Printf("Aval = %d, Bval = %d\n", Aval, Bval)

// 更新转账后账本中A余额
err = stub.PutState(A, []byte(strconv.Itoa(Aval)))
if err != nil {
    return shim.Error(err.Error())
}

// 更新转账后账本中B余额
err = stub.PutState(B, []byte(strconv.Itoa(Bval)))
if err != nil {
    return shim.Error(err.Error())
}

return shim.Success(nil)
}

// 账户注销
func (t *SimpleChaincode) delete(stub shim.ChaincodeStubInterface, args []string) pb.Response {
    if len(args) != 1 {
        return shim.Error("Incorrect number of arguments. Expecting 1")
    }

    A := args[0] // 账户用户名

    // 从账本中删除该账户状态
    err := stub.DelState(A)
    if err != nil {
        return shim.Error("Failed to delete state")
    }

    return shim.Success(nil)
}

// 账户查询
func (t *SimpleChaincode) query(stub shim.ChaincodeStubInterface, args []string) pb.Response {
    var A string
    var err error

    if len(args) != 1 {
        return shim.Error("Incorrect number of arguments. Expecting name of the person to query")
    }

    A = args[0] // 账户用户名

    // 从账本中获取该账户余额
    Avalbytes, err := stub.GetState(A)
    if err != nil {
        jsonResp := "{\"Error\":\"Failed to get state for " + A + "\"}"
        return shim.Error(jsonResp)
    }

    if Avalbytes == nil {
        jsonResp := "{\"Error\":\"Nil amount for " + A + "\"}"
        return shim.Error(jsonResp)
    }

    jsonResp := "{\"Name\":\"" + A + "\",\"Amount\":\"" + string(Avalbytes) + "\"}"
    fmt.Printf("Query Response:%s\n", jsonResp)
    // 返回转账金额
    return shim.Success(Avalbytes)
}
```

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

2.3.4 链代码示例（2.0 风格）

如下是一个账户转账的链代码示例（2.0风格）仅供安装实例化，若您需要调测请参考[Fabric官方示例](#)中的链代码。

```
package main

import (
    "errors"
    "fmt"
    "strconv"

    "github.com/hyperledger/fabric-contract-api-go/contractapi"
)

// 链码实现
type ABStore struct {
    contractapi.Contract
}

// 初始化链码数据，实例化或者升级链码时自动调用
func (t *ABStore) Init(ctx contractapi.TransactionContextInterface, A string, Aval int, B string, Bval int) error {
    // 使用println函数输出的信息会记录在链码容器日志中
    fmt.Println("ABStore Init")
    var err error

    fmt.Printf("Aval = %d, Bval = %d\n", Aval, Bval)
    // 将状态数据写入账本
    err = ctx.GetStub().PutState(A, []byte(strconv.Itoa(Aval)))
    if err != nil {
        return err
    }

    err = ctx.GetStub().PutState(B, []byte(strconv.Itoa(Bval)))
    if err != nil {
        return err
    }

    return nil
}

// A转账X给B
func (t *ABStore) Invoke(ctx contractapi.TransactionContextInterface, A, B string, X int) error {
    var err error
    var Aval int
    var Bval int

    // 从账本获取状态数据
    Avalbytes, err := ctx.GetStub().GetState(A)
    if err != nil {
        return fmt.Errorf("Failed to get state")
    }
    if Avalbytes == nil {
        return fmt.Errorf("Entity not found")
    }
    Aval, _ = strconv.Atoi(string(Avalbytes))

    Bvalbytes, err := ctx.GetStub().GetState(B)
    if err != nil {
        return fmt.Errorf("Failed to get state")
    }
    if Bvalbytes == nil {
```

```
        return fmt.Errorf("Entity not found")
    }
    Bval, _ = strconv.Atoi(string(Bvalbytes))

    // 执行转账
    Aval = Aval - X
    Bval = Bval + X
    fmt.Printf("Aval = %d, Bval = %d\n", Aval, Bval)

    // 将状态数据重新写回账本
    err = ctx.GetStub().PutState(A, []byte(strconv.Itoa(Aval)))
    if err != nil {
        return err
    }

    err = ctx.GetStub().PutState(B, []byte(strconv.Itoa(Bval)))
    if err != nil {
        return err
    }

    return nil
}

// 账户注销
func (t *ABstore) Delete(ctx contractapi.TransactionContextInterface, A string) error {

    // 从账本中删除账户状态
    err := ctx.GetStub().DelState(A)
    if err != nil {
        return fmt.Errorf("Failed to delete state")
    }

    return nil
}

// 账户查询
func (t *ABstore) Query(ctx contractapi.TransactionContextInterface, A string) (string, error) {
    var err error
    // 从账本获取状态数据
    Avalbytes, err := ctx.GetStub().GetState(A)
    if err != nil {
        jsonResp := "{\"Error\":\"Failed to get state for " + A + "\"}"
        return "", errors.New(jsonResp)
    }

    if Avalbytes == nil {
        jsonResp := "{\"Error\":\"Nil amount for " + A + "\"}"
        return "", errors.New(jsonResp)
    }

    jsonResp := "{\"Name\":\"" + A + "\",\"Amount\":\"" + string(Avalbytes) + "\"}"
    fmt.Printf("Query Response:%s\n", jsonResp)
    return string(Avalbytes), nil
}

func main() {
    cc, err := contractapi.NewChaincode(new(ABstore))
    if err != nil {
        panic(err.Error())
    }
    if err := cc.Start(); err != nil {
        fmt.Printf("Error starting ABstore chaincode: %s", err)
    }
}
```

2.3.5 链代码调测

链代码调测主要是使用shim包中的MockStub进行单元测试，所以您需要在测试代码中引入shim包。您可参考下文提供的示例测试代码，也可参考[Fabric官方示例](#)中的其它测试代码。

编写测试代码

[账户转账链代码示例](#)的测试代码内容如下：

```
package main

import (
    "fmt"
    "testing"

    "github.com/hyperledger/fabric/core/chaincode/shim"
)

func checkInit(t *testing.T, stub *shim.MockStub, args [][]byte) {
    res := stub.MockInit("1", args)
    if res.Status != shim.OK {
        fmt.Println("Init failed", string(res.Message))
        t.FailNow()
    }
}

func checkState(t *testing.T, stub *shim.MockStub, name string, value string) {
    bytes := stub.State[name]
    if bytes == nil {
        fmt.Println("State", name, "failed to get value")
        t.FailNow()
    }
    if string(bytes) != value {
        fmt.Println("State value", name, "was not", value, "as expected")
        t.FailNow()
    }
}

func checkQuery(t *testing.T, stub *shim.MockStub, name string, value string) {
    res := stub.MockInvoke("1", [][]byte{{[]byte("query"), []byte(name)}})
    if res.Status != shim.OK {
        fmt.Println("Query", name, "failed", string(res.Message))
        t.FailNow()
    }
    if res.Payload == nil {
        fmt.Println("Query", name, "failed to get value")
        t.FailNow()
    }
    if string(res.Payload) != value {
        fmt.Println("Query value", name, "was not", value, "as expected")
        t.FailNow()
    }
}

func checkInvoke(t *testing.T, stub *shim.MockStub, args [][]byte) {
    res := stub.MockInvoke("1", args)
    if res.Status != shim.OK {
        fmt.Println("Invoke", args, "failed", string(res.Message))
        t.FailNow()
    }
}

func TestExample02_Init(t *testing.T) {
    scc := new(SimpleChaincode)
    stub := shim.NewMockStub("ex02", scc)
```

```
// Init A=123 B=234
checkInit(t, stub, [][]byte{[]byte("init"), []byte("A"), []byte("123"), []byte("B"), []byte("234")})

checkState(t, stub, "A", "123")
checkState(t, stub, "B", "234")
}

func TestExample02_Query(t *testing.T) {
    scc := new(SimpleChaincode)
    stub := shim.NewMockStub("ex02", scc)

    // Init A=345 B=456
    checkInit(t, stub, [][]byte{[]byte("init"), []byte("A"), []byte("345"), []byte("B"), []byte("456")})

    // Query A
    checkQuery(t, stub, "A", "345")

    // Query B
    checkQuery(t, stub, "B", "456")
}

func TestExample02_Invoke(t *testing.T) {
    scc := new(SimpleChaincode)
    stub := shim.NewMockStub("ex02", scc)

    // Init A=567 B=678
    checkInit(t, stub, [][]byte{[]byte("init"), []byte("A"), []byte("567"), []byte("B"), []byte("678")})

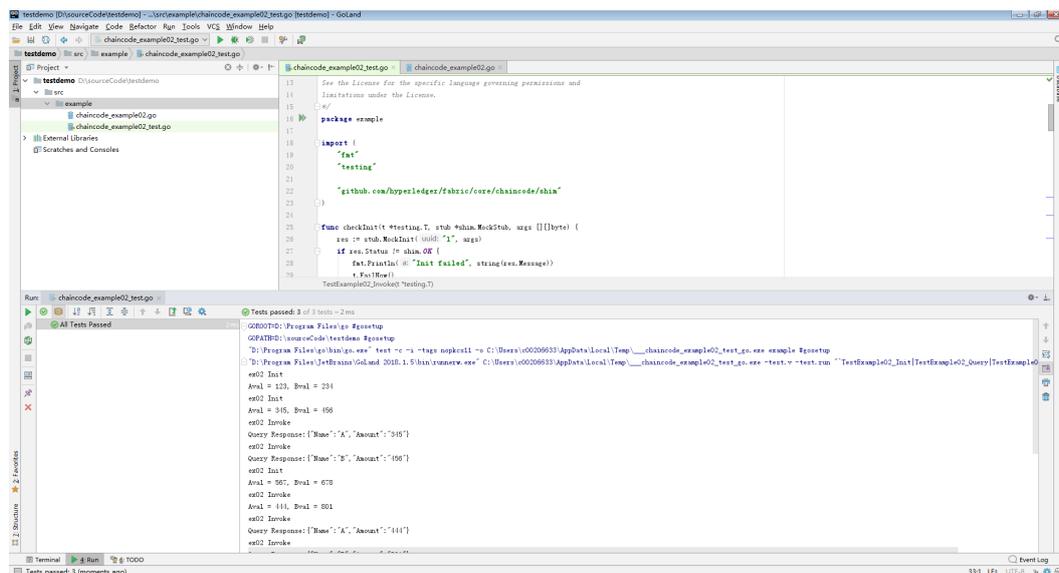
    // Invoke A->B for 123
    checkInvoke(t, stub, [][]byte{[]byte("invoke"), []byte("A"), []byte("B"), []byte("123")})
    checkQuery(t, stub, "A", "444")
    checkQuery(t, stub, "B", "801")

    // Invoke B->A for 234
    checkInvoke(t, stub, [][]byte{[]byte("invoke"), []byte("B"), []byte("A"), []byte("234")})
    checkQuery(t, stub, "A", "678")
    checkQuery(t, stub, "B", "567")
    checkQuery(t, stub, "A", "678")
    checkQuery(t, stub, "B", "567")
}
```

执行调测

调测时在IDE中单击执行对应的测试函数即可。

图 2-1 执行调测



2.4 Java 语言链代码开发

2.4.1 链代码结构

本章以Java语言为例来介绍。链代码即一个Java项目，创建好文件后进行函数开发等操作。

约束与限制

Java链代码仅支持Fabric 2.2及以上版本。

链代码接口

链代码启动必须通过调用shim包中的start方法。实际开发中，您需要自行定义一个类，来继承ChaincodeBase。以下为继承时必须重写的方法：

```
public class SimpleChaincodeSimple extends ChaincodeBase {
    @Override
    public Response init(ChaincodeStub stub) {
    }

    @Override
    public Response invoke(ChaincodeStub stub) {
    }
}
```

- **init方法**：在链代码实例化或升级时被调用，完成初始化数据的工作。
- **Invoke方法**：更新或查询账本数据状态时被调用，需要在此方法中实现响应调用或查询的业务逻辑。

链代码结构

Java语言的链代码结构如下：

```
package main
```

```
// 引入必要的包，系统自动操作，只要在maven或gradle中配置即可
import org.hyperledger.fabric.shim.ChaincodeBase;
import org.hyperledger.fabric.shim.ChaincodeStub;

public class SimpleChaincodeSimple extends ChaincodeBase {
    @Override
    public Response init(ChaincodeStub stub) {
        // 在该方法中实现链代码初始化或升级时的处理逻辑
        // 编写时可灵活使用stub中的API
    }

    @Override
    public Response invoke(ChaincodeStub stub) {
        // 在该方法中实现链代码运行中被调用或查询时的处理逻辑
        // 编写时可灵活使用stub中的API
    }

    //主函数，需要调用shim.Start()方法
    public static void main(String[] args) {
        new SimpleChaincode().start(args);
    }
}
```

2.4.2 链代码相关的 API

Fabric源码包中的shim包提供了如下几种类型的接口，您可以参考使用：

- 参数解析API：调用链代码时需要给被调用的目标函数/方法传递参数，该API提供解析这些参数的方法。
- 账本状态数据操作API：该API提供了对账本数据状态进行操作的方法，包括对状态数据的查询及事务处理等。
- 交易信息获取API：获取提交的交易信息的相关API。
- 其他API：其他的API，包括事件设置、调用其他链代码操作。

2.4.3 链代码示例

如下是一个读写数据的链代码示例，您也可以参考[Fabric官方示例](#)中其他链代码。

📖 说明

新建Java项目时，您可以选择新建maven或者gradle项目，以导入依赖包。本示例以gradle项目为例。

```
/* 导入此段代码到项目的build.gradle前，请删除或注释build.gradle中原有内容 */
buildscript {
    repositories {
        mavenLocal()
        maven{ url "https://mirrors.huaweicloud.com/repository/maven/" }
    }
    dependencies {
        classpath 'com.github.jengelman.gradle.plugins:shadow:2.0.4' }
}

apply plugin: 'com.github.johnrengelman.shadow'
apply plugin: 'java'

sourceCompatibility = 1.8

// 依赖包所用的代码仓库，将从所选地址搜寻并下载依赖包
repositories {
    mavenLocal()
    maven{ url "https://mirrors.huaweicloud.com/repository/maven/" }
    maven{ url "https://jitpack.io" }/* 若Json-Schema无法导入，可以尝试加入此库 */
}
```

```
// 引入代码所需的依赖包
dependencies {
    compile group: 'org.hyperledger.fabric-chaincode-java', name: 'fabric-chaincode-shim', version: '2.2.+
    testCompile group: 'junit', name: 'junit', version: '4.12' // testCompile表明仅在调测时使用此包
    testCompile 'org.mockito:mockito-core:2.4.1'
}

shadowJar {
    baseName = 'chaincode'
    version = null
    classifier = null
    manifest {
        // 需要与继承ChaincodeBase类接口的类路径保持一致
        attributes 'Main-Class': 'org.hyperledger.fabric.example.SimpleChaincode'
    }
}
// 以下为SimpleChaincode类内中的内容
package org.hyperledger.fabric.example;//根据链代码文件实际位置编写，一般由系统自动生成

// 引入必要的包，系统自动操作，只要在maven或gradle中配置即可
import java.util.List;

import com.google.protobuf.ByteString;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.hyperledger.fabric.shim.ChaincodeBase;
import org.hyperledger.fabric.shim.ChaincodeStub;

import static java.nio.charset.StandardCharsets.UTF_8;

// SimpleChaincode example simple Chaincode implementation
public class SimpleChaincode extends ChaincodeBase {

    private static Log logger = LogFactory.getLog(SimpleChaincode.class);

    @Override
    public Response init(ChaincodeStub stub) {
        logger.info("Init");
        return new SuccessResponse();
    }

    @Override
    public Response invoke(ChaincodeStub stub) {
        try {
            logger.info("Invoke java simple chaincode");
            String func = stub.getFunction();
            List<String> params = stub.getParameters();
            if (func.equals("insert")) {
                return insert(stub, params);
            }
            if (func.equals("query")) {
                return query(stub, params);
            }
            return new ErrorResponse("Invalid invoke function name. Expecting one of: [\"insert\", \"query\"]");
        } catch (Throwable e) {
            return new ErrorResponse(e);
        }
    }

    // The Insert method implements the data storage function and stores the key-value on the chain
    private Response insert(ChaincodeStub stub, List<String> args) {
        if (args.size() != 2) {
            return new ErrorResponse("Incorrect number of arguments. Expecting 2");
        }
        String key = args.get(0);
        String val = args.get(1);

        stub.putState(key, ByteString.copyFrom(val, UTF_8).toByteArray());
    }
}
```

```
        return newSuccessResponse();
    }

    // The Query method implements the data query function by invoking the API to query the value of the
    // key
    // API to query the value corresponding to a key
    private Response query(ChaincodeStub stub, List<String> args) {
        if (args.size() != 1) {
            return newErrorResponse("Incorrect number of arguments. Expecting name of the person to
            query");
        }
        String key = args.get(0);
        // Get the value of key
        String val = stub.getStringState(key);
        if (val == null) {
            String jsonResp = "{\"Error\":\"Null val for " + key + "\"}";
            return newErrorResponse(jsonResp);
        }
        logger.info(String.format("Query Response:\nkey: %s, val: %s\n", key, val));
        return newSuccessResponse(val, ByteString.copyFrom(val, UTF_8).toByteArray());
    }

    public static void main(String[] args) {
        new SimpleChaincode().start(args);
    }
}
```

2.4.4 链代码调测

对链代码进行调测，主要是使用MockStub进行单元测试。本章中测试的链代码获取方法：登录区块链服务管理控制台，进入“应用案例”，单击“Java示例Demo-Java SDK Demo”中“Chaincode_Java_Local_Demo”的“下载”按钮。

添加依赖

使用mock()方法，需要添加mockito相关依赖。

- gradle版本：
在build.gradle文件中的dependencies内添加如下配置依赖，注意不是buildscript内的dependencies：
testCompile 'org.mockito:mockito-core:2.4.1'
- maven版本：
在pom.xml文件中的dependencies（若无则添加）内添加如下配置依赖：

```
<dependency>
  <groupId>org.mockito</groupId>
  <artifactId>mockito-core</artifactId>
  <version>2.4.1</version>
</dependency>
```

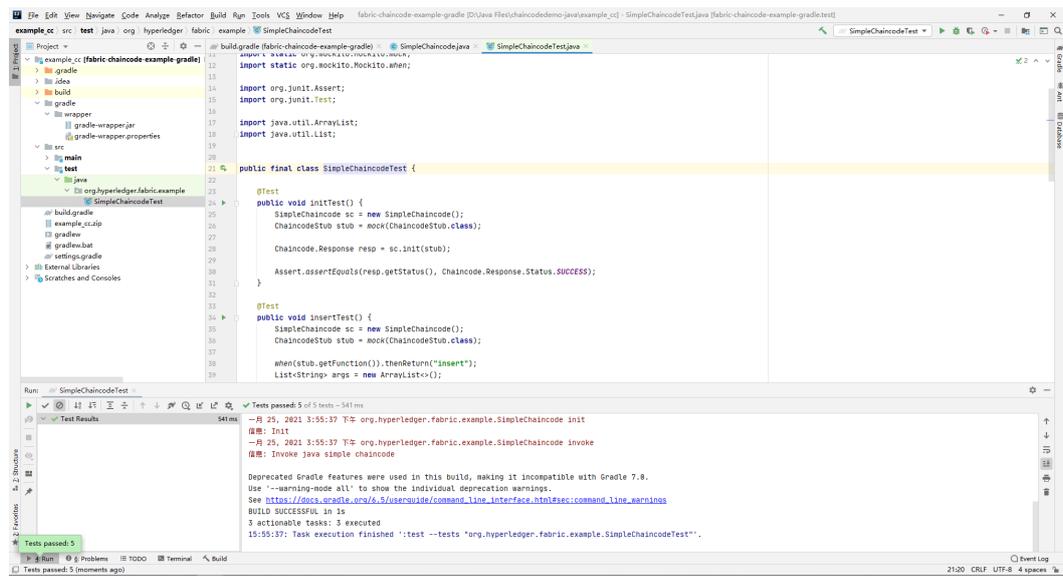
编写测试代码

若创建项目时没有test文件夹，在src下新建文件夹，并如图在Gradle Source Sets里面选择“test\java”，然后创建测试文件：SimpleChaincodeTest.java，如图所示：

图 2-2 创建测试文件



图 2-3 测试文件



SimpleChaincodeTest.java测试代码内容:

```
import org.hyperledger.fabric.example.SimpleChaincode;
import org.hyperledger.fabric.shim.Chaincode;
import org.hyperledger.fabric.shim.ChaincodeStub;
import org.junit.Assert;
import org.junit.Test;

import java.util.ArrayList;
import java.util.List;

import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

public final class SimpleChaincodeTest {

    @Test
    public void initTest() {
        SimpleChaincode sc = new SimpleChaincode();
        ChaincodeStub stub = mock(ChaincodeStub.class);
        Chaincode.Response resp = sc.init(stub);
        Assert.assertEquals(resp.getStatus(), Chaincode.Response.Status.SUCCESS);
    }

    @Test
    public void insertTest() {
        SimpleChaincode sc = new SimpleChaincode();
        ChaincodeStub stub = mock(ChaincodeStub.class);
        when(stub.getFunction()).thenReturn("insert");
    }
}
```

```
List<String> args = new ArrayList<>();
args.add("a");
args.add("100");
when(stub.getParameters()).thenReturn(args);
Chaincode.Response resp = sc.invoke(stub);
Assert.assertEquals(resp.getStatus(), Chaincode.Response.Status.SUCCESS);
}

@Test
public void insertTooManyArgsTest() {
    SimpleChaincode sc = new SimpleChaincode();
    ChaincodeStub stub = mock(ChaincodeStub.class);
    when(stub.getFunction()).thenReturn("insert");
    List<String> args = new ArrayList<>();
    args.add("a");
    args.add("100");
    args.add("b");
    args.add("100");
    when(stub.getParameters()).thenReturn(args);
    Chaincode.Response resp = sc.invoke(stub);
    Assert.assertEquals(resp.getMessage(), "Incorrect number of arguments. Expecting 2");
}

@Test
public void queryTest() {
    SimpleChaincode sc = new SimpleChaincode();
    ChaincodeStub stub = mock(ChaincodeStub.class);
    when(stub.getFunction()).thenReturn("query");
    List<String> args = new ArrayList<>();
    args.add("a");
    when(stub.getParameters()).thenReturn(args);
    when(stub.getStringState("a")).thenReturn("100");
    Chaincode.Response resp = sc.invoke(stub);
    Assert.assertEquals(resp.getMessage(), "100");
}

@Test
public void queryNotExistTest() {
    SimpleChaincode sc = new SimpleChaincode();
    ChaincodeStub stub = mock(ChaincodeStub.class);
    when(stub.getFunction()).thenReturn("query");
    List<String> args = new ArrayList<>();
    args.add("a");
    when(stub.getParameters()).thenReturn(args);
    when(stub.getStringState("a")).thenReturn(null);
    Chaincode.Response resp = sc.invoke(stub);
    Assert.assertEquals(resp.getMessage(), "{\"Error\": \"Null val for a\"}");
}
}
```

执行调测

在SimpleChaincodeTest.java中，单击SimpleChaincodeTest方法前的“Run Test”按钮，执行测试。

图 2-4 执行测试

```
16
17     import java.util.ArrayList;
18     import java.util.List;
19
20
21     Run Test final class SimpleChaincodeTest {
22
23         @Test
24         public void initTest() {
25             SimpleChaincode sc = new SimpleChaincode();
26             ChaincodeStub stub = mock(ChaincodeStub.class);
27
28             Chaincode.Response resp = sc.init(stub);
29
30             Assert.assertEquals(resp.getStatus(), Chaincode.Response.Status.SUCCESS);
31         }

```

执行成功如图所示，表示链代码调测无问题：

图 2-5 执行成功

```
Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.5/userguide/command\_line\_interface.html#sec:command\_line\_warnings
BUILD SUCCESSFUL in 1s
3 actionable tasks: 1 executed, 2 up-to-date
16:12:23: Task execution finished ':test --tests "org.hyperledger.fabric.example.SimpleChaincodeTest"'.

```

执行失败示例如图所示，请根据失败提示修改链代码或者检查调测代码的逻辑：

图 2-6 执行失败

```
Run: SimpleChaincodeTest.queryTest
Tests failed: 1 of 1 test - 535 ms
There were failing tests. See the report at: file:///C:/Users/.../reports/TESTS/TEST\_RESULTS.html
* Try:
  Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.
  Get more help at https://help.gradle.org

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.5/userguide/command\_line\_interface.html#sec:command\_line\_warnings
BUILD FAILED in 1s
3 actionable tasks: 2 executed, 1 up-to-date

```

```
expected:<[{"Error":"Null val for a"}]> but was:<[100]>
Expected :{"Error":"Null val for a"}
Actual   :100
<Click to see difference>

```

3 应用程序开发

3.1 概述

用户的应用程序通过链代码与账本数据进行交互。应用程序开发可使用的语言比较广泛，如Golang、Solidity、Java、C++、Python、Node.js等。应用程序和链代码开发语言无强对应关系，只要应用程序能通过SDK来调用链代码即可。

📖 说明

- Hyperledger Fabric增强版对应用程序开放的接口均为gRPC协议，与开源版本保持一致，通常使用SDK进行调用，详情可参考[Hyperledger Fabric增强版SDK接口定义](#)。

3.2 开发前准备

用户的应用程序通过链代码与账本数据进行交互。应用程序开发可使用的语言比较广泛，如Golang、Solidity、Java、C++、Python、Node.js等。应用程序和链代码开发语言无强对应关系，只要应用程序能通过SDK来调用链代码即可。

- 您需要购买区块链实例。
区块链实例支持在CCE集群上部署，具体可参见[基于CCE集群](#)。
- 您需要获取对应实例的SDK配置文件，具体可参见[下载SDK配置和证书](#)。

3.3 应用程序开发

您需要自行开发应用程序业务逻辑代码。应用开发过程中可使用BCS提供的[国密加密SDK](#)，也可使用Fabric官方社区提供的和您自身的实例版本匹配的SDK。

另外，区块链实例提供同态加密库供您使用，相关资料及资源请参见[同态加密](#)。

📖 说明

Fabric源码包选择和创建的区块链实例版本对应，即如果创建区块链实例时，Hyperledger Fabric增强版内核是v2.2（4.X.X版本），则Fabric源码包对应选择2.2版本。

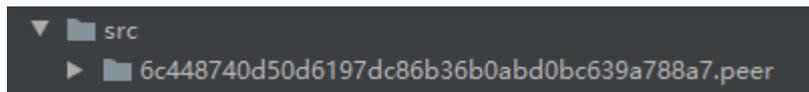
配置组织 ID

您需要修改应用程序中配置实例组织ID的相关代码，下载证书文件解压后的peer文件包括目录名和对应组织ID。

如下图所示，仅供示例参考，请以实际操作的证书文件为准。

证书文件解压后目录名是6c448740d50d6197dc86b36b0abd0bc639a788a7.peer，组织ID为6c448740d50d6197dc86b36b0abd0bc639a788a7。

图 3-1 证书文件解压



配置 SDK 文件

1. 您需要修改应用程序中SDK配置文件相关代码，如下面示例所示，您需要填写正确的SDK配置文件绝对路径。

```
var (  
    configFile = "/root/gosdkdemo/config/go-sdk-demo-channel-sdk-config.yaml"  
    org = "6c448740d50d6197dc86b36b0abd0bc639a788a7"  
)
```

2. 如果您下载SDK配置文件时填写的证书存放路径与实际不符，您需要修改SDK配置文件中所有证书相关路径。

4 示例 Demo

4.1 GO SDK Demo

本节提供了一个基于Go SDK的Demo，帮助用户开发自己的Go客户端应用程序。

准备工作

- 准备弹性云服务器。
- 在弹性云服务器上安装golang环境，Go版本要求：1.12及以上，1.16以下（ ≥ 1.12 ， < 1.16 ）。
- 获取Go SDK源码，获取方法：登录区块链服务管理控制台，进入“应用案例”，单击“GO示例Demo-GO SDK Demo”中Go应用程序源码的“下载”按钮。

购买区块链实例

购买区块链实例，具体请参见实例部署-[基于CCE集群](#)。

安装及实例化链代码

本示例使用链代码文件获取方法：登录区块链服务管理控制台，进入“应用案例”，单击“GO示例Demo-GO SDK Demo”中Go语言示例链代码的“下载”按钮。

参考章节：[用户指南-区块链管理-链代码管理](#)。

下载 SDK 和证书

步骤1 登录区块链服务管理控制台。

步骤2 在“实例管理”页面，单击对应实例卡片上的“获取客户端配置”。

步骤3 在新打开的页面，勾选“SDK文件”，SDK配置参数如下：

参数	值
链代码名称	chaincode 说明 链代码名称需要和链代码安装&实例化时的一致。
证书存放路径	/root/gosdkdemo/config
通道名称	channel
组织&Peer节点	保持系统默认

勾选“共识节点证书”。

勾选“Peer节点证书”，指定节点组织选择organization，勾选“管理员证书”。

步骤4 单击“下载”。下载SDK配置文件、orderer组织的管理员证书和organization组织的管理员证书。

----结束

部署应用

1. 将Go SDK源码下载至准备的弹性云服务器“/root”路径下并解压。
下载方法：登录区块链服务管理控制台，进入“应用案例”，单击“GO示例 Demo-GO SDK Demo”中Go应用程序源码的“下载”按钮。
2. 将**下载SDK和证书**步骤中的zip文件解压后，把configs文件夹中的orderer文件夹、peer文件夹、sdk-config.json、sdk-config.yaml文件全部复制到/root/gosdkdemo/config/目录下。
3. 在代码中找到“/gosdkdemo/src/main.go”文件，进行以下修改：

- a. 将configFile中的值修改为实际的SDK配置文件名称，例如：demo-channel-sdk-config.yaml。
- b. 将org的值修改为organization对应的组织哈希值。
在通道管理页面，单击“查看节点”获取组织的哈希值（MSP标识去掉“MSP”后缀即为对应组织的哈希）。

```
var (
    configFile = "/root/gosdkdemo/config/go-sdk-demo-channel-sdk-config.yaml"
    org = " 9103f17cb6b4f69d75982eb48bececcc51aa3125"
)
```

4. 使用go mod方式配置GOPATH路径，请根据实际安装路径进行配置。

- a. 设置环境变量GO111MODULE为on。
export GO111MODULE=on
- b. go.mod文件如图所示，用户需要根据实际安装路径修改replace代码。

```
module main
go 1.15
// 指定导入的依赖包及其版本
require (
    github.com/bitly/go-simplejson v0.5.0
    github.com/bmizerany/assert v0.0.0-20160611221934-b7ed37b82869// indirect
    github.com/ghodss/yaml v1.0.0
    github.com/hyperledger/fabric-sdk-go v1.0.0
    github.com/pkg/errors v0.9.1
    github.com/spf13/viper v1.7.1
)
```

```
// 以项目路径为/root/gosdkdemo/src为例
replace github.com/hyperledger/fabric-sdk-go => /root/gosdkdemo/src/github.com/hyperledger/
fabric-sdk-go
```

5. 找到“gosdkdemo/src”路径下的main.go文件，执行如下命令：
go run main.go

```
[root@cluster-bcs-luya-401r src]# go run main.go
[fabsdk/fab] 2020/09/21 03:15:16 UTC - fab.(*EndpointConfig).LoadPrivateKeyFromConfig -> WARN private key was not encrypted, please consider encrypt your private key
[fabsdk/fab] 2020/09/21 03:15:16 UTC - fab.detectDeprecatedNetworkConfig -> WARN Getting orderers from endpoint config channels.orderer is deprecated, use entity matchers to override orderer configuration
[fabsdk/fab] 2020/09/21 03:15:16 UTC - fab.detectDeprecatedNetworkConfig -> WARN visit https://github.com/hyperledger/fabric-sdk-go/blob/master/test/fixtures/config/overrides/local_entity_matchers.yaml for samples
insert new data <testuser,100> success
query key <testuser> value is 100
[root@cluster-bcs-luya-401r src]#
```

通过内存传入私钥

如果用户需要对私钥文件进行加密，并在demo中解密后传入FabricSDK。

1. 对于TLS私钥

在main.go文件的initializeSdk函数中，按如下方式调用函数：

```
encryptedTlsKey,err := GetTlsCryptoKey(org) //从配置文件指定路径下读取加密过的TLS私钥
//用户按照自行规定的加解密方法对encryptedTlsKey进行解密得到decryptedTlsKey字符串
SetClientTlsKey(decryptedTlsKey) //将解密后的TLS私钥传入Fabric-SDK，实现通过内存传入TLSKey
```

2. 对于MSP私钥

在main.go文件的insert函数中，按如下方式调用函数：

```
encryptedByteKey,_ := GetPrivateKeyBytes(org) //从配置文件指定路径下读取加密过的MSP私钥
//用户按照自行规定的加解密方法对encryptedByteKey进行解密得到decryptedKey字符串
SetPrivateKey(decryptedKey) //将解密后的MSP私钥赋值给全局变量privateKey并通过该变量传入
```

常用 API 接口

Fabric-sdk-go的主要入口是FabricSDK类，这个可以通过NewSDK()方法分别可以生成。

Fabric-sdk-go的常用操作基本都可以用这四个client实现：FabricClient，ChannelClient，ChannelMgmtClient，ResourceMgmtClient。

- **FabricSDK**

FabricSDK在pkg\fabsdk\fabsdk.go中，通过New()方法生成object。New()方法支持可变参数Option，以下是生成FabricSDK的例子：

```
var opts []fabsdk.Option
opts = append(opts, fabsdk.WithOrgid(org))
opts = append(opts, fabsdk.WithUserName("Admin"))
sdk, err = fabsdk.New(config.FromFile(configFile), opts...)
```

configFile是SDK配置文件的路径。OrgId是SDK配置文件中的组织id。

FabricSDK在def/fabapi/fabapi.go中，通过NewSDK()方法生成object。NewSDK()方法有一个Options参数，以下是生成Options参数的例子：

```
def.Options{ConfigFile: configFile, LoggerFactory: logging.LoggerProvider(), Username: sysadmin}
```

ConfigFile是SDK配置文件的路径。LoggerFactory是可选的，不提供的話default会log到console。

- **FabricClient**

FabricClient主要有以下常用的接口。

接口名称	描述	参数值	返回值
CreateChannel	创建Channel的接口，用于创建channel。	request CreateChannelRequest	txn.TransactionID, error
QueryChannelInfo	查询Channel的接口，用于查询Channel的信息。	name string, peers []Peer	Channel, error
InstallChaincode	安装链码的接口，安装链码到区块链中。	request InstallChaincodeRequest	[]*txn.TransactionProposalResponse, string, error
InstallChaincode	安装链码的接口，安装链码到区块链中。	request InstallChaincodeRequest	[]*txn.TransactionProposalResponse, string, error
QueryChannels	查询channel的接口，查询区块链中已创建的通道。	peer Peer	*pb.ChannelQueryResponse, error
QueryInstalledChaincodes	查询已安装链码的接口，查询区块链中已安装的链码。	peer Peer	*pb.ChaincodeQueryResponse, error

- **ChannelClient**

ChannelClient主要包括链码查询和链码调用两类接口。

接口名称	描述	参数值	返回值
Query	链码查询接口，调用链码进行查询。	request QueryRequest	[]byte, error
QueryWithOptions	带options的链码查询接口，与Query类似，但是可以通过QueryOpts指定notifier, peers,和timeout。	request QueryRequest, opt QueryOpts	[]byte, error

接口名称	描述	参数值	返回值
ExecuteTx	链码调用接口，用于链码的调用。	request ExecuteTxRequest	TransactionID, error
ExecuteTxWithOpts	带options的链码调用接口，与ExecuteTx类似，但是可以通过ExecuteTxOpts指定notifier, peers,和timeout。	request ExecuteTxRequest ExecuteTxOpts	TransactionID, error

- **ChannelMgmtClient**

ChannelMgmtClient 只有两个接口SaveChannel(req SaveChannelRequest) error 和SaveChannelWithOpts(req SaveChannelRequest, opts SaveChannelOpts) error 这两个接口是用于创建channel用的，这两个接口里面具体实现会调用到FabricClient里createChannel()接口。

- **ResourceMgmtClient**

ResourceMgmtClient主要就是与链码生命周期相关的接口和一个peer加入通道的接口。

 **说明**

链码的删除接口为BCS增加的接口，目前只实现了删除链码安装包的功能。

接口名称	描述	参数值	返回值
InstallCC	安装链码，用于安装链码。	reqInstallCCRequest	[]InstallCCResponse, error
InstallCCWithOpts	带options的链码安装，与InstallCC类似，但是可以通过InstallCCOpts指定peers。	reqInstallCCRequest,opts InstallCCOpts	[]InstallCCResponse, error
InstantiateCC	实例化链码接口，用于实例化链码。	channelID string,reqInstantiateCCRequest	error
InstantiateCCWithOpts	带options的链码实例化，与InstantiateCC类似，但是可以通过InstantiateCCOpts指定peers和timeout。	channelID string,reqInstantiateCCRequest, optsInstantiateCCOpts	error
UpgradeCC	升级链码，用于链码的升级。	channelID string,reqUpgradeCCRequest	error

接口名称	描述	参数值	返回值
UpgradeCCWith Opts	带options的链码升级，与UpgradeCC类似，但是可以通过UpgradeCCOpts指定peers和timeout。	channelID string, reqUpgradeCCRequest, optsUpgradeCC Opts	error
DeleteCC	删除链码，用于链码的删除，目前只有删除安装包的功能。	channelID string, reqDeleteCCRequest	error
DeleteCCWith Opts	带options的链码删除，与DeleteCC类似，但是可以通过DeleteCCWithOpts指定peers和timeout。	channelID string, reqDeleteCCRequest, optsDeleteCCOpts	error
JoinChannel	Peers加入Channel的接口，用于peers加入Channel。	channelID string	error
JoinChannelWith Opts	带options的Peers加入Channel的接口，与JoinChannel类似，但是可以通过JoinChannelOpts指定peers。	channelID string, optsJoinChannelOpts	error

📖 说明

带options的接口都可以指定peers，peers可通过def/fabapi/pkgfactory.go 里的 NewPeer(userName string, orgName string, url string, certificate string, serverHostOverride string, config config.Config) (fab.Peer, error) 生成。这个方法比原生的NewPeer多两个参数userName, orgName, 这两个参数用于peer双向tls找到对应的tls证书。

调用合约

Main.go是一个简单的客户端应用示例程序，主要是为了方便用户熟悉客户端开发的流程，主要包含以下步骤：

```
//1.导入相关包：Sdk包中提供了一些API，以便用户的应用程序能够访问链代码。
import (
    "fmt"
    "github.com/hyperledger/fabric-sdk-go/pkg/client/channel"
    "github.com/hyperledger/fabric-sdk-go/pkg/fabsdk" .....
)
//2.创建文件配置：这部分封装了应用开发必要的一些公共配置，包括sdk配置文件路径、组织名
var (
    configFile = "/root/fabric-go-demo/config/go-sdk-demo-channel-sdk-config.yaml"
    org = "9103f17cb6b4f69d75982eb48bececcc51aa3125"
```

```

.....
)
//3.加载配置文件
loadConfig()
//4. 初始化sdk
initializeSdk()
//5. 执行链代码，将数据写入账本，key = "testuser", value= "100"
insert("insert",[] []byte{
    []byte("testuser"),
    []byte("100"),
})
//6.查询链代码，输出查询结果，key = "testuser"
query("query",
[] []byte{
[]byte("testuser"),
})

```

表 4-1 调用函数介绍

函数名	说明
getOptsToInitializeSDK	解析配置文件，创建并返回fabsdk.Option对象。
GetDefaultChaincodeId	解析配置文件，返回chaincodeID。
GetDefaultChannel	解析配置文件，返回channelID。
UserIdentityWithOrgAndName	用户身份验证，输入为组织名和用户名，返回为验证结果。
ChannelClient	创建*channel.Client对象，输入为组织名、用户名以及通道ID，返回*channel.Client对象。
insert	将数据写入账本，输入参数为链码的对应方法名称以及要插入的键值对，返回为写入的结果。
query	查询链上信息，输入参数为链码的对应方法名称以及要查询的数据，返回为查询的结果。

4.2 Java SDK Demo

以上章节是对链代码的开发、调测及其应用配置的介绍，本章节通过提供一个基于Java SDK的Demo（支持国密加密算法），以Java SDK Demo为例，让您了解区块链的相关知识以及体验应用开发的流程，助您快速上手并使用区块链服务。

说明

只用于场景体验，不用于实际应用。

准备工作

操作指导	说明
安装开发工具 (IDE)	本地需提前安装JDK、maven和eclipse (可以使用您习惯的IDE)。 JDK需要安装1.8版本64位, 如果本地已安装JDK, 可以通过cmd命令java -version来查看JDK版本。

购买区块链实例

- 步骤1** 登录区块链服务管理控制台。
- 步骤2** 进入“实例管理”页面, 单击Hyperledger Fabric增强版的“购买”按钮。
- 步骤3** 根据界面提示, 配置区块链基本信息, 参数如表4-2所示。

须知

为了保证示例Demo成功运行, 请在参数配置时按照表格中的参数值填写。

表 4-2 基本信息配置

参数	参数值
计费模式	按需计费
企业项目	选择已创建的企业项目, 例如: default。 如果您没有开通企业管理服务, 将无法看到企业项目选项。
区块链实例名称	java-sdk-demo
版本类型	基础版
区块链类型	私有链
Hyperledger Fabric增强版内核	v2.2
共识策略	Raft(CFT)
资源初始密码	请自行设置
资源初始密码确认	-

- 步骤4** 单击“下一步: 资源配置”, 进行资源配置, 参数如表4-3所示。

表 4-3 资源配置

参数	示例
环境资源	选择“自定义环境”。
集群	创建新的CCE集群
可用区	请自行选择
多可用区	否
云主机规格	4核/8GB
云主机个数	1
高可用	否
虚拟私有云	系统自动创建VPC
所在子网	系统自动创建子网
云主机登录方式	密码
root密码	如果填写该项，则以填写值为准，如果不填写，则以资源初始密码为准。
确认密码	-
是否使用CCE集群节点弹性IP	是
弹性IP计费方式	使用默认规格
弹性IP带宽	5 Mbit/s

步骤5 单击“下一步：区块链配置”，进行区块链配置，参数如表4-4所示。

表 4-4 区块链配置

参数	示例
区块链配置	选择“自定义配置”。
区块链管理初始密码	如果填写该项，则以填写值为准，如果不填写，则以资源初始密码为准。
区块链管理确认密码	-
存储卷类型	极速文件存储卷，也可以根据界面提示配置。
节点组织存储容量(GB)	使用默认规格。
账本数据存储方式	选择“文件数据库 (GoLevelDB)”
peer节点组织	系统已默认创建1个节点组织，名称为organization，将节点数量修改为1。

参数	示例
通道配置	organization节点组织已默认添加进至通道中，保持默认即可。
共识节点数量	使用默认。
安全机制	ECDSA 须知 安全机制支持选择国密算法。若选择了国密算法，则Demo的部署需要做其他修改，请注意下文中的相关描述。
区块生成配置	否
添加RESTful API支持	否

步骤6 单击“下一步：确认规格”。

步骤7 确认配置信息无误后，根据界面提示购买区块链实例。

请等待数分钟，安装页面提示安装成功，查看实例状态变为“正常”后，表示区块链实例部署完成。

----结束

安装及实例化链代码

步骤1 登录区块链服务管理控制台。

步骤2 单击左侧导航栏中的“实例管理”。

步骤3 在新创建的实例卡片中，单击“区块链管理”，登录链代码管理页面。

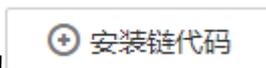


步骤4 在登录页面输入用户名、密码，单击“登录”。

📖 说明

用户名为admin，密码为您在创建区块链实例时设置的区块链管理初始密码，如果没有设置区块链管理初始密码，则以资源初始密码为准。

步骤5 在链代码管理页面，单击页面左上角的



安装参数如下：

表 4-5 安装参数

参数	值
链代码名称	chaincode
链代码版本	2.0
账本数据存储方式	文件数据库(goleveldb)
选择全部Peer节点	勾选
组织&Peer节点	peer-0
链代码语言	Golang
链代码文件	Chaincode_Go_Demo, 获取方法: 登录区块链服务管理控制台, 进入“应用案例”, 单击“Java示例Demo-Java SDK Demo”中Go语言示例链代码的“下载”按钮。
链代码描述	根据需要填写相关描述。
代码安全检查	链代码语言选择Golang, 该功能才会显示。选择是否开启链代码安全检查。

步骤6 单击“安装”完成链代码安装。

步骤7 链代码安装完成后, 在链代码列表的“操作”列, 单击“实例化”。

实例化参数如下:

表 4-6 实例化参数

参数	值
链代码名称	chaincode
实例化通道	channel
链代码版本	2.0
初始化函数	init
链代码参数	a,200,b,250
背书策略	下列任意组织背书
背书组织列表	organization
隐私保护配置	否

步骤8 单击“实例化”, 完成链代码在当前通道上的实例化。

稍等2-3分钟后刷新界面, 单击“实例化”列的“查看更多”, 查看链代码实例化进度。

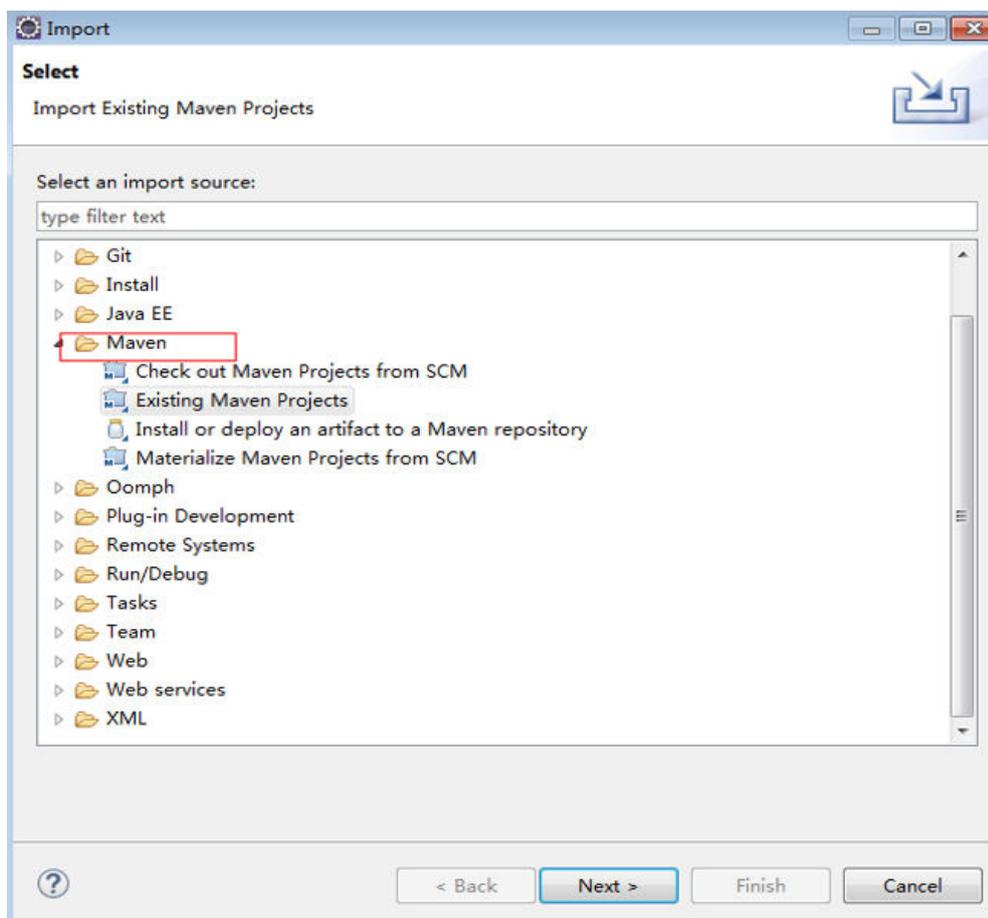
----结束

配置应用

步骤1 导入项目工程

获取项目代码并解压，获取方法：登录区块链服务管理控制台，进入“应用案例”，单击“Java示例Demo-Java SDK Demo”中Java项目源码的“下载”按钮。

在eclipse界面中右键选择import，将项目代码导入eclipse中（javasdkdemo为Maven工程）。



步骤2 下载SDK配置文件和证书。

1. 在“实例管理”界面，在实例卡片中，单击“获取客户端配置”。
2. 勾选“SDK文件”，SDK配置参数如下：

表 4-7 SDK 配置参数

参数名称	说明
链代码名称	chaincode 须知 链代码名称需要和链代码安装&实例化时的一致。

参数名称	说明
证书存放路径	此处配置为下载的javasdkdemo工程的config文件夹路径。 须知 将windows的反斜杠“\”换为“/”。例如，正确的写法为D:/javasdkdemo/config
通道名称	channel
组织&Peer节点	选择通道中所有节点组织

勾选“共识节点证书”。

勾选“Peer节点证书”，指定节点组织选择organization，勾选“管理员证书”。

- 单击“下载”，下载SDK配置文件、java-sdk-demo-orderer组织的管理员证书、organization组织的管理员证书，下载至demo工程的config路径中。

步骤3 复制并解压。

步骤4 将下载的demo-config.zip文件解压，将java-sdk-demo-orderer-admin-cert、organization-admin-cert和sdk-config文件夹中的内容复制至demo工程的config路径下。

----结束

部署应用

步骤1 在Maven工程中找到/javasdkdemo/src/main/java/handler/Main.java，将Main类中如下代码中的文件路径，修改为java-sdk-demo-sdk-config.yaml文件的绝对路径，路径请参见**步骤2.3**。需将windows的反斜杠“\”改为“/”。

```
helper.setConfigCtx("E:/yourdir/huawei.yaml");
例如修改为：helper.setConfigCtx("D:/javasdkdemo/config/java-sdk-demo-channel-sdk-config.yaml");
```

须知

若您在订购区块链实例时安全机制选择“国密算法”，还需要做如下修改：国密算法需要引用华为自己的fabric-sdk依赖包，故需要在pom.xml中添加对工程lib文件夹“fabric-sdk-java-1.4.1-jar-with-dependencies.jar”中的依赖，如下图所示去掉该依赖的注释即可，否则可能会导致运行错误。

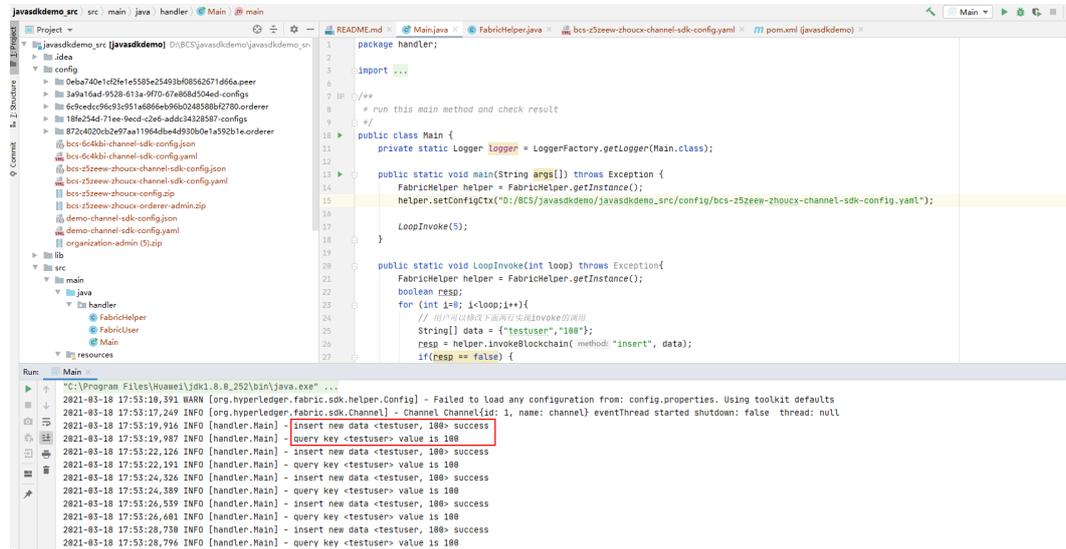
图 4-1 文件详情

```
58     </dependency>
59     <dependency>
60         <groupId>org.hyperledger.fabric-sdk-java</groupId>
61         <artifactId>fabric-sdk-java</artifactId>
62         <version>1.4.1</version>
63     <!--
64     <scope>system</scope>
65     <systemPath>${project.basedir}/lib/fabric-sdk-java-1.4.1-jar-with-dependencies.jar</systemPath>
66     -->
67 </dependency>
68 </dependency>
```

步骤2 运行main函数。

每成功执行一次，表示向区块链存入一对键值对，<testuser,100>; 在区块链上查询键值为testuser的value值为100。也可以通过区块链浏览器可以查看交易记录。

图 4-2 交易记录信息



----结束

通过内存传入私钥

如果用户需要对私钥文件进行加密，并在demo中解密后传入FabricSDK。

对于MSP私钥：在FabricHelper文件的genFabricUser函数中，按如下方式调用函数：

```
//从配置文件指定路径下读取加密过的MSP私钥
String adminPrivateKeyString = extractPemString(msp, "keystore");
//对adminPrivateKeyString 进行解密得到decryptedKey字符串
//将解密后的MSP私钥重新赋值给变量adminPrivateKeyString
String adminPrivateKeyString = decryptedKey;
```

说明

当前不支持通过内存传入TLS私钥。

4.3 Gateway Java Demo

本节提供一个基于Fabric Gateway Java的Demo，Fabric Gateway Java对Java SDK进行了封装，简化了代码量，帮助用户开发自己的Java客户端应用程序。

常用接口

使用Fabric-Gateway-Java发起交易和查询，主要用到Network和Contract两类的接口，[更多的api接口请参考Fabric官网](#)。

- Network

主要有以下常用的接口：

接口名称	描述	参数值	返回值
getContract	获取Contract实例的接口	String chaincodeId	Contract
addBlockListener	设置监听器的接口，监听区块事件	Consumer<org.hyperledger.fabric.sdk.BlockEvent> listener	Consumer<org.hyperledger.fabric.sdk.BlockEvent>
getChannel	获取与network相关联channel的接口	/	org.hyperledger.fabric.sdk.Channel
removeBlockListener	移除监听器的接口	Consumer<org.hyperledger.fabric.sdk.BlockEvent> listener	void

- **Contract**

主要有以下常用的接口：

接口名称	描述	参数值	返回值
submitTransaction	发起交易的接口，需要输入调用方法与参数	String name, String... args	byte[]
evaluateTransaction	发起查询的接口，需要输入调用方法与参数	String name, String... args	byte[]
createTransaction	创建交易的接口，需要submit以发起交易	String name	Transaction
addContractListener	设置监听器的接口，监听已经提交的交易发出的事件	Consumer<ContractEvent> listener	Consumer<ContractEvent> listener
removeContractListener	移除监听器的接口	Consumer<ContractEvent> listener	void

4.4 REST API Demo

华为云区块链提供了REST API服务来简化用户访问区块链的学习成本。通过REST API服务，用户不需要学习fabric-go-sdk, fabric-Java-sdk, fabric-nodejs-sdk等，只需要开发的应用支持RESTful接口，就可以轻松访问区块链。本Demo通过一个go语言的客户端来演示如何使用REST API服务调用链代码，供您学习参考。

 说明

只用于场景体验，不用于实际应用。

创建区块链实例

- 步骤1** 登录区块链服务管理控制台。
- 步骤2** 单击Hyperledger Fabric增强版卡片上的“购买”。
- 步骤3** 根据界面提示，配置区块链基本信息，参数如表4-8所示。

须知

为了保证示例Demo成功运行，请在参数配置时按照表格中的参数值填写。

表 4-8 基本信息配置

参数	参数值
计费模式	按需计费
区域	使用默认区域
企业项目	default
区块链实例名称	demo
版本类型	基础版
区块链类型	私有链
Hyperledger Fabric增强版内核	v2.2
共识策略	Raft(CFT)
资源初始密码	请自行设置
资源初始密码确认	请自行设置

- 步骤4** 单击“下一步：资源配置”，进行资源配置，参数如表4-9所示。

表 4-9 资源配置

参数	示例
环境资源	选择“自定义环境”。
集群	创建新的CCE集群
可用区	请自行选择
云主机规格	4核/8GB
云主机个数	1

参数	示例
高可用	否
虚拟私有云	系统自动创建VPC
所在子网	系统自动创建子网
云主机登录方式	密码
root密码	如果填写该项，则以填写值为准，如果不填写，则以资源初始密码为准。
确认密码	-
是否使用CCE集群节点弹性IP	是
弹性IP计费方式	使用默认规格
弹性IP带宽	5 Mbit/s

步骤5 单击“下一步：区块链配置”，进行区块链配置，参数如表4-10所示。

表 4-10 区块链配置

参数	示例
区块链配置	选择“自定义配置”。
区块链管理初始密码	如果填写该项，则以填写值为准，如果不填写，则以资源初始密码为准。
区块链管理确认密码	-
存储卷类型	极速文件存储卷
节点组织存储容量 (GB)	使用默认规格。
账本数据存储方式	选择“文件数据库 (GoLevelDB) ”
peer节点组织	系统已默认创建1个节点组织，名称为organization，将节点数量修改为1。
通道配置	organization节点组织已默认添加进至通道中，保持默认即可。
共识节点数量	使用默认值。
安全机制	ECDSA 须知 安全机制仅支持选择ECDSA。
区块生成配置	否

参数	示例
添加RESTful API支持	<p>请选择“是”。若您选择了“否”，则需要实例创建完毕后，执行以下步骤安装RESTful API：</p> <ol style="list-style-type: none"> 1. 左侧导航栏选择“插件管理”。 2. 在“插件仓库”页签下，鼠标移至baas-restapi插件卡片右上角。 3. 单击“安装”，选择已购买的区块链实例，安装baas-restapi插件。

步骤6 单击“下一步：确认订单”。

步骤7 确认配置信息无误后，根据界面提示购买区块链实例。

请等待数分钟，安装页面提示安装成功，查看实例状态变为“正常”后，表示区块链实例部署完成。

----结束

安装及实例化链代码

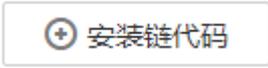
步骤1 登录区块链服务管理控制台。

步骤2 在新创建的实例卡片中，单击“区块链管理”，登录链代码管理页面。

步骤3 在登录页面输入用户名、密码，单击“登录”。

📖 说明

用户名为admin，密码为您在创建区块链实例时设置的区块链管理初始密码，如果没有设置区块链管理初始密码，则以资源初始密码为准。

步骤4 在链代码管理页面，单击页面左上角的 。

安装参数如下：

参数	值
链代码名称	bcsysq
链代码版本	1.0
账本数据存储方式	文件数据库(goleveldb)
选择全部Peer节点	勾选
组织&Peer节点	peer-0
链代码语言	Golang
链代码文件	下载示例链代码文件： chaincode_example02.zip 。
链代码描述	根据需要填写相关描述。

参数	值
代码安全检查	链代码语言选择Golang，该功能才会显示。选择是否开启链代码安全检查。

步骤5 单击“安装”完成链代码安装。

步骤6 链代码安装完成后，在链代码列表的“操作”列，单击“实例化”。

实例化参数如下：

参数	值
实例化通道	channel
链代码版本	1.0
初始化函数	init
链代码参数	a,200,b,250
背书策略	任意组织背书
背书组织列表	organization
隐私保护配置	否

----结束

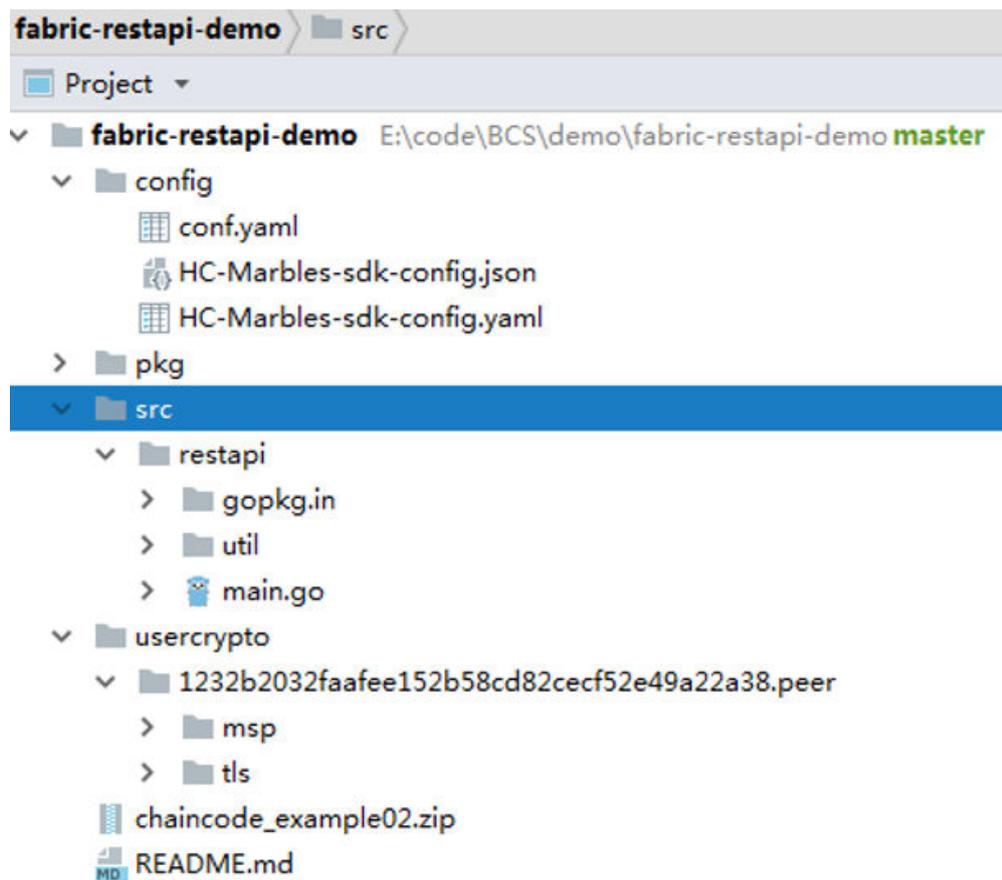
配置应用

步骤1 在“实例管理”界面，在实例卡片中，单击“获取客户端配置”。

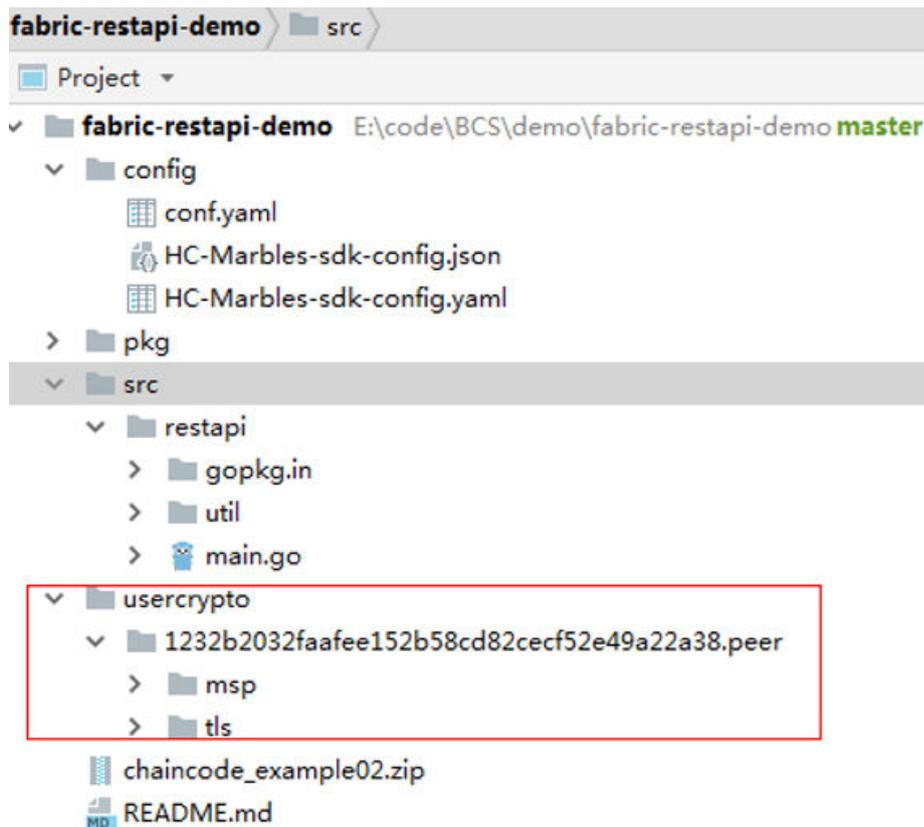
步骤2 勾选“Peer节点证书”，指定节点组织选择organization，勾选“用户证书”并下载。

步骤3 下载Demo项目工程：[fabric-restapi-demo.zip](#)，并将Demo项目代码工程包下载解压到本地并使用IDE打开。

本Demo是一个golang编写的REST客户端，通过RESTAPI服务来调用链代码，实现链代码a给b转账的功能，请用GoLand等个人喜欢的IDE打开。项目的内容如下图所示：



步骤4 将下载的用户证书解压到项目的usercrypto目录下。如图所示：



步骤5 修改参数配置。

1. 修改config目录下conf.yaml中的各项参数，参考如下截图及表格。

```
Endpoint: "https://10.154.116.39:32021"
Peer: "/org1/chaincode/operation"
CryptoMethod: "qa"

SignCert: "usercrypto/1232b2032faafee152b58cd82cecf52e49a22a38.peer/msp/signcerts/User1g1232b2032faafee152b58cd82cecf52e49a22a38.peer-1232b2032faafee152b58cd82cecf52e49a22a38.default.svc.cluster.local-cert.pem"
PrivateKey: "usercrypto/1232b2032faafee152b58cd82cecf52e49a22a38.peer/msp/keystore/850a9b5-c937-442c-833c-6c0aee9300a_us"

InvokeReq:
- invoke01:
  Significance: 0
  ChannelId: "channel"
  ChaincodeId: "testcode"
  ChaincodeVersion: "1.0"
  UserId: "User1"
  OrgId: "1232b2032faafee152b58cd82cecf52e49a22a38"
  OpMethod: "invoke"
  Args: ["invoke", "a", "b", "100"]
- invoke2:
  Significance: 0
  ChannelId: "channel"
  ChaincodeId: "testcode"
  ChaincodeVersion: "1.0"
  UserId: "User1"
  OrgId: "1232b2032faafee152b58cd82cecf52e49a22a38"
  OpMethod: "invoke"
  Args: ["invoke", "a", "b", "100"]
```

2. 修改src/restapi目录下的main.go文件，参考如下截图及表格。

```
for _, v := range GlobalConfig.InvokeReq {
    for _, req := range v {
        orgPeer1 := OrgPeer {
            OrgId: "b67710cb6bee8bacdce095cac73dc8bee82248da",
            PeerDomainName: "peer-b67710cb6bee8bacdce095cac73dc8bee82248da-0-peer-b67710cb6bee8bacdce095cac73dc8bee82248da.default.svc.cluster.local",
        }
        orgPeers := []OrgPeer{orgPeer1}
        orgPeersByte, _ := json.Marshal(orgPeers)
        req.OrgPeers = string(orgPeersByte)
    }
}
```

 说明

针对需要参与背书的每个peer节点，对其构造一个OrgPeer结构体，将组织ID和peer节点的域名传入，并将该结构体添加进OrgPeer类型的数组中，经json.Marshal()方法转换为字节数组，最后会转换成字符串类型传入。其中OrgPeer结构体定义如下：

```
type OrgPeer struct {
    OrgId string `json:"orgId"`
    PeerDomainName string `json:"peerDomainName"`
}
```

表 4-11 参数表

参数	说明
Endpoint	RESTAPI服务的访问IP和端口。具体获取方法如下： 1. 在已购买区块链实例卡片中，单击“容器集群”名称，进入云容器引擎CCE页面。 2. 单击实例所在集群名称，进入集群信息页面。在左侧导航栏，单击“工作负载”。 3. 在“无状态负载”页签，单击baas-restapi工作负载名称，进入详情页面，然后单击实例所在节点跳转至节点管理页面获取节点的弹性公网ip，该弹性公网ip为RESTAPI服务访问IP，端口固定为32621。
Path	RESTAPI服务的访问路径，保持不变。
CryptoMethod	加密算法，如果是ECDSA算法填写“SW”。
SignCert	用户下载的证书中的签名证书路径。
PrvKey	用户下载证书中的签名私钥。
InvokeReq	请求body的参数，请按照部署的链码实际情况填写，可参考如下InvokeReq参数表。
QueryReq	与InvokeReq类似，按照部署链码的实际情况填写。

表 4-12 InvokeReq 参数表

参数	说明	范例
SignGzipSwitch	Sign是否选择Gzip压缩。0表示否，1表示是。	"1"
ChannelId	区块链通道名称。	"channel"
ChaincodeId	链代码名称。	"testcode"
ChaincodeVersion	链代码版本。	"1.0"

参数	说明	范例
UserId	由组织CA签发的用户ID，目前区块链实例默认生成的都为User1。	"User1"
OrgId	区块链组织ID。 说明 在“通道管理”页面中，单击通道名称后的“查看节点”，查看“MSP标识”，去掉MSP即为区块链组织ID。例如：MSP标识为"1232b2032faafee152b58cd82cecf52e49a22a38MSP"，区块链组织ID为"1232b2032faafee152b58cd82cecf52e49a22a38"。	"1232b2032faafee152b58cd82cecf52e49a22a38"
OrgPeers	各个Peer节点的组织ID和域名。	"[{OrgId:"1232b2032faafee152b58cd82cecf52e49a22a38", PeerDomainName:"peer-1232b2032faafee152b58cd82cecf52e49a22a38-0.peer-1232b2032faafee152b58cd82cecf52e49a22a38.default.svc.cluster.local"}]"
Opmethod	区块链链码调用类型，目前只有两类：invoke和query。	"invoke"
Args	链码调用参数。	'["invoke","a","b","100"]'（第一个参数为invoke的调用函数，可以是其他值如：move，query等等）

步骤6 配置完成后，构建并运行main()来运行该Demo项目。

代码中将读取conf.yaml以及main.go中的参数QueryReq和InvokeReq请求，并调用RESTAPI的接口"/v1/chaincode/operation"来调用链代码，实现a给b转账功能。运行结果如下：

```
go build main.go (2) x
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The result resp of query is "MTAwMDA="
After query the count of a is 10000 b is 9000
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The result resp of query is "OTAwMA=="
After query the count of a is 10000 b is 9000
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The result resp of query is "MTAwMDA="
After query the count of a is 10000 b is 9000
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Process finished with exit code 0
```

说明

本Demo用一个简单的REST客户端调用RESTAPI服务实现了调用链代码，返回的invoke结果为一个base64加密的TransactionID，query结果为base64加密的数据值。代码仅供参考，可以通过该项目代码理解如何调用RESTAPI服务。

----**结束**

5 区块链中间件接口

5.1 概述

欢迎使用区块链中间件功能，它可以帮助您快速集成区块链能力，提供易用、标准化的接口，支撑上层应用的开发。详细说明请参见[表5-1](#)：

本章节主要介绍数据面的API接口，管理面的API接口请参见《[API接口参考](#)》。

数据面请求EndPoint可以通过管理面的查询服务实例详细信息接口返回结果中basic_info->agent_portal_addr字段的值获取，请求示例：<https://192.168.0.90:30603/v2/agent/apis/tokens>。

表 5-1 概览

类型	描述	所属中间件	端口
链代码调用 (公测)	通过标准接口的方式访问区块链系统，完成链代码方法（invoke和query）的调用。 说明 该功能为公测特性（Beta）。	baas-restapi插件 安装方法： 登录区块链服务管理控制台，在页面左侧选择“插件管理”，在“插件仓库”页签下，在baas-restapi插件的卡片上，单击“安装”。	32621
链代码管理 (公测)	通过标准接口的方式实现链代码管理能力，包括链代码安装、实例化、删除、查询等。 说明 该功能为公测特性（Beta）。	baas-agent(默认已安装)	30603

<p>分布式身份 (公测)</p>	<p>通过标准接口的方式实现分布式身份, 包括分布式身份(DID)和可验证凭证(VC)的生成、申请、签发等管理能力。</p> <p>说明 该功能为公测特性 (Beta)。</p>	<p>baas-restapi插件 安装方法: 登录区块链服务管理控制台, 在页面左侧选择“插件管理”, 在“插件仓库”页签下, 在baas-restapi插件的卡片上, 单击“安装”。在安装插件页面, 开启分布式身份接口。</p>	<p>32621</p>
<p>可信数据交换 (公测)</p>	<p>通过标准接口的方式实现基于区块链的可信数据交换能力。包括数据的发布、授权、分享、解密等。</p> <p>说明 该功能为公测特性 (Beta)。</p>	<p>baas-restapi插件 安装方法: 登录区块链服务管理控制台, 在页面左侧选择“插件管理”, 在“插件仓库”页签下, 在baas-restapi插件的卡片上, 单击“安装”。在安装插件页面, 开启可信数据交换接口。</p>	<p>32621</p>

5.2 链代码调用 (公测)

功能介绍

对已经部署并已经实例化的区块链链代码进行调用 (invoke) 和查询 (query)。

URI

POST /v1/chaincode/operation

请求消息

表 5-2 请求参数

参数	是否必选	参数类型	描述
channelId	是	String	区块链通道ID
chaincodeId	是	String	链代码ID
chaincodeVersion	否	String	链代码版本
userId	是	String	由组织CA签发的用户ID, 目前区块链服务默认生成的都为User1
orgId	是	String	区块链组织ID

参数	是否必选	参数类型	描述
orgPeers	是	String	由组织中每个节点的组织ID和域名组成的数组，形如： [{"orgId":"7258adda1803f4137eff4813e7aba323018200c5","peerDomainName":"peer-7258adda1803f4137eff4813e7aba323018200c5-0.peer-7258adda1803f4137eff4813e7aba323018200c5.default.svc.cluster.local"}]
opmethod	是	String	区块链链码调用类型，目前只有两类调用方法：invoke和query。
args	是	String	链码调用参数，形如： ["invoke","a","b","1"]
timestamp	是	String	格式为2018-10-31T17:28:16+08:00
cert	是	String	用户的证书文件，以字符串形式上传

📖 说明

以上参数获取方式，详细参见用户指南中的[链代码管理](#)和[区块浏览器](#)章节。

- 在链代码管理页面中，单击链代码名称前的 ，展开链代码详细信息，您可以查看当前链代码的版本列表、安装列表和实例化情况。
- 在区块浏览器页面中，在通道下拉框中选择一个通道，下方的数据即可实时刷新供您查看多项数据。区块链相关信息的查询功能，包括区块数量、交易数量、区块详细信息、交易详细信息、性能数据及节点状态等。
- 为了保证交易安全性，需要使用Fabric用户证书（证书获取方式请见[下载用户证书](#)）中的私钥对请求消息体进行签名（目前只支持椭圆曲线，暂不支持国密等其他加密算法），并将签名结果放到消息头部x-bcs-signature-sign字段。

链码REST API自定义了一些消息头，请参见[表5-3](#)。

表 5-3 自定义消息头

名称	是否必选	描述
x-bcs-signature-sign	是	链码调用请求消息体签名。
x-bcs-signature-method	是	加密类型，目前固定是SW。
x-bcs-signature-sign-gzip	是	Sign是否选择Gzip压缩。0表示否，1表示是。


```
\n-----END CERTIFICATE-----\n"
}
```

- 响应示例

```
After invoke the count of a is 188 b is 262
```

错误码

请参见[错误码](#)页面。

5.3 链代码管理（公测）

5.3.1 获取 Token

功能介绍

根据区块链浏览器用户名密码获取Token。

URI

GET /v2/agent/apis/tokens

请求参数

表 5-4 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-User	是	String	用户名
X-Auth-Pass	是	String	密码

响应参数

状态码： 200

表 5-5 响应 Body 参数

参数	参数类型	描述
user	String	用户名
token	String	token
expire_time	Long	失效时间

状态码： 400

表 5-6 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

GET https://192.168.0.90:30603/v2/agent/apis/tokens

响应示例

状态码： 200

Success

```
{
  "user": "admin",
  "token":
  "ACiUxRlk2Jcu2d5bp2SyfP7abHV1nVKlo9Mm2AxI9dDN7mdHXXHBdg=kaIglHeZLTu6Uelu5YRcXTlpUPXyIEG3
  ESNu0vlukrikG6SVhO393ds8rG+aRnZ+mMyookApP",
  "expire_time": 1605867860701
}
```

状态码： 400

Bad Request

```
{
  "error_code": "BCS.4000013",
  "error_message": "request body is too large"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.2 安装链代码

功能介绍

在区块链节点上安装链代码，部分场景只支持go语言链码

URI

POST /v2/agent/apis/chaincode/install

请求参数

表 5-7 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

表 5-8 FormData 参数

参数	是否必选	参数类型	描述
chaincode_name	是	String	链代码名称，以小写字母开头，支持小写字母和数字 最小长度：6 最大长度：25
chaincode_version	是	String	链代码版本，只允许使用数字、点(.)、横杠(-)，必须以数字开头和结尾，且点和横杠不能相邻 最小长度：1 最大长度：14
target_peers	是	String	链代码安装Peer列表信息，例如： [{"org_id":"9fb42c91458763990a45b62af92546a21f168dae", "org_name":"organization3", "peer_id":"peer-9fb42c91458763990a45b62af92546a21f168dae-0.peer-9fb42c91458763990a45b62af92546a21f168dae.default.svc.cluster.local", "peer_name":"peer-0"}]
description	否	String	链代码描述
chaincode_language	是	String	链代码编程语言，例如 golang
db_type	否	String	账本数据存储方式，请按照实例创建时的账本数据存储方式填写。例如 goleveldb, couchdb

参数	是否必选	参数类型	描述
security_check	否	String	链代码安全检查选项，目前只对golang语言链代码有效，true表示开启，false表示关闭，默认为false
file	是	File	链代码zip文件

响应参数

状态码： 200

表 5-9 响应 Body 参数

参数	参数类型	描述
total_peer_num	Integer	链代码操作Peer总数
success_peer_num	Integer	操作成功Peer数量
fail_peer_num	Integer	操作失败Peer数量
fail_peers	Array of strings	操作失败Peer信息

状态码： 400

表 5-10 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

POST https://192.168.0.90:30603/v2/agent/apis/chaincode/install

```
{chaincode_name:gochaincode3chaincode_version:2.0target_peers:
[{"org_id":"9802af57cfab764dc12b860c44b01969575e83c9","org_name":"organization","peer_id":"peer-9802af57cfab764dc12b860c44b01969575e83c9-1.peer-9802af57cfab764dc12b860c44b01969575e83c9.default.svc.cluster.local","peer_name":"node-1"}]description:2222222chaincode_language:golangdb_type:goleveldbsecurity_check:true}
```

响应示例

状态码： 200

Success

```
{
  "total_peer_num" : 4,
  "success_peer_num" : 4,
  "fail_peer_num" : 0,
  "fail_peers" : [ ]
}
```

状态码： 400

Bad Request

```
{
  "error_code" : "BCS.4000013",
  "error_message" : "request body is too large"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.3 实例化链代码

功能介绍

实例化链代码

URI

POST /v2/agent/apis/chaincode/instantiate

请求参数

表 5-11 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

表 5-12 请求 Body 参数

参数	是否必选	参数类型	描述
chaincode_name	是	String	链代码名称，以小写字母开头，支持小写字母和数字，长度6-25位 最小长度：6 最大长度：25
chaincode_version	是	String	链代码版本，只允许使用数字、点(.)、横杠(-)，必须以数字开头和结尾，且点和横杠不能相邻
channel_name	是	String	实例化通道名称
endorsement_policy	是	Policy object	背书策略
init_variable	是	InitArgs object	初始化函数及参数
private_data	否	String	隐私保护配置数据，json数据的string格式，json数据结构举例： <pre>[{"name":"kvstore-collection","policy":"OR('9b4ea44913e8eed42cb056177b46191e1d316678MSP.member','9b4ea44913e8eed42cb056177b46191e1d316678MSP.member')","requiredPeerCount":0,"maxPeerCount":2,"blockToLive":0,"memberOnlyRead":true}]</pre>

表 5-13 Policy

参数	是否必选	参数类型	描述
operation	是	String	操作符。OR：任意组织背书 AND：全部组织背书
group	是	Array of OrgPolicy objects	组织背书成员

表 5-14 OrgPolicy

参数	是否必选	参数类型	描述
org_id	是	String	组织ID
category	否	String	成员类型：member、admin

表 5-15 InitArgs

参数	是否必选	参数类型	描述
func_name	是	String	初始化函数名
args	是	Array of strings	初始化参数列表

响应参数

状态码： 400

表 5-16 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

```
POST https://192.168.0.90:30603/v2/agent/apis/chaincode/instantiate
{
  "chaincode_name": "gochaincode2",
  "chaincode_version": "1.0",
  "channel_name": "channel001",
  "endorsement_policy": {
    "operation": "AND",
    "group": [ {
      "org_id": "8cc7155fb1f26ebac1743f481386c14223be511f",
      "category": "member"
    } ]
  },
  "init_variable": {
    "func_name": "init",
    "args": [ "a", "200", "b", "100" ]
  },
  "private_data": "[{"name": \"kvstore-collection\", \"policy\": \"OR('8cc7155fb1f26ebac1743f481386c14223be511fMSP.member', '8cc7155fb1f26ebac1743f481386c14223be511fMSP.member')\", \"requiredPeerCount\": 0, \"maxPeerCount\": 2, \"blockToLive\": 0, \"memberOnlyRead\": true}]"
}
```

响应示例

状态码： 400

Bad Request

```
{
  "error_code" : "BCS.4000013",
  "error_message" : "request body is too large"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.4 获取安装的链码列表

功能介绍

获取已经安装的链码列表

URI

GET /v2/agent/apis/chaincodes

表 5-17 Query 参数

参数	是否必选	参数类型	描述
offset	否	Integer	查询链代码列表的起始位置，默认为0
limit	否	Integer	查询链代码列表的数量，默认为10

请求参数

表 5-18 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

响应参数

状态码： 200

表 5-19 响应 Body 参数

参数	参数类型	描述
count	Integer	链码总数
chaincodes	Array of ChaincodeInfo objects	链码列表

表 5-20 ChaincodeInfo

参数	参数类型	描述
chaincode_name	String	链码名称
chaincode_language	String	链码开发语言
update_time	String	链码更新时间
chaincode_version	String	链码版本，多个链码之间以逗号(,)分割
install_org_infos	Array of PeerInfo objects	链码的安装信息
instantiated_channel	instantiated_channel object	链码通道信息
instantiated_info	instantiated_info object	实例化信息

表 5-21 PeerInfo

参数	参数类型	描述
org_name	String	组织名称
org_id	String	组织id
peer_name	String	节点名称
peer_id	String	节点id
status	String	节点状态

参数	参数类型	描述
channels	Array of strings	未实例化的peer信息
url	String	Peer节点的url信息
peer	String	Peer节点的内部域名

表 5-22 instantiated_channel

参数	参数类型	描述
error	Array of CCInstantiateChannelError objects	实例化错误信息
success	Array of strings	成功的通道
inprogress	Array of strings	实例化进度

表 5-23 CCInstantiatedChannelError

参数	参数类型	描述
channel_name	String	错误通道名
error_detail	String	错误详情

表 5-24 instantiated_info

参数	参数类型	描述
channels	Array of channels objects	通道信息

表 5-25 channels

参数	参数类型	描述
channel_id	String	通道名称

参数	参数类型	描述
orgs	Array of orgs objects	通道内组织信息
versions	Array of strings	版本信息

表 5-26 orgs

参数	参数类型	描述
org_name	String	组织名
org_id	String	组织id

状态码： 400

表 5-27 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

GET https://192.168.0.90:30603/v2/agent/apis/chaincodes

响应示例

状态码： 200

Success

```
{
  "count": 12,
  "chaincodes": [ {
    "chaincode_name": "test001",
    "chaincode_version": "1.0",
    "update_time": "2021-01-12T11:32:04.193358708+08:00",
    "instantiated_info": {
      "channels": [ {
        "channel_id": "channel",
        "versions": [ "1.0" ]
      }, {
        "channel_id": "testchannel",
        "orgs": null,
        "versions": [ "1.0" ]
      }
    ]
  }
}
```

```

    },
    "chaincode_language": "golang"
  }
}

```

状态码： 400

Bad Request

```

{
  "error_code": "BCS.4000013",
  "error_message": "request body is too large"
}

```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.5 查询指定链码版本信息

功能介绍

查询指定链码版本信息

URI

GET /v2/agent/apis/chaincode/versions

表 5-28 Query 参数

参数	是否必选	参数类型	描述
chaincode_name	是	String	链代码名称，以小写字母开头，支持小写字母和数字，长度6-25位 最小长度：6 最大长度：25

请求参数

表 5-29 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

响应参数

状态码： 200

表 5-30 响应 Body 参数

参数	参数类型	描述
versions	Array of ChaincodeVersion objects	链码版本信息

表 5-31 ChaincodeVersion

参数	参数类型	描述
version	String	链码版本
hash_code	String	链码版本哈希值
description	String	链码版本描述
install_time	String	链码版本安装时间
update_time	String	链码版本更新时间
instantiate_status	Boolean	链码版本实例化状态
security_check_status	Integer	链代码安全检查状态（0：不存在，1：运行中，2：完成，3：失败）
uninstantiated_peer_infos	Array of PeerInfo objects	未实例化的peer信息

表 5-32 PeerInfo

参数	参数类型	描述
org_name	String	组织名称

参数	参数类型	描述
org_id	String	组织id
peer_name	String	节点名称
peer_id	String	节点id
status	String	节点状态
channels	Array of strings	未实例化的peer信息
url	String	Peer节点的url信息
peer	String	Peer节点的内部域名

状态码： 400

表 5-33 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

GET https://192.168.0.90:30603/v2/agent/apis/chaincode/versions?chaincode_name=chaincode

响应示例

状态码： 200

Success

```
{
  "versions": [ {
    "version": "1.0",
    "hash_code": "1473b4807fe9f70d1ba56192e41d39c7d621d07d80e603cf75ed3982b81034d",
    "description": "",
    "install_time": "2021-01-11T11:27:12.093454567+08:00",
    "update_time": "2021-01-11T11:27:12.093454789+08:00",
    "instantiate_status": false,
    "uninstantiated_peer_infos": [ {
      "org_name": "organization",
      "org_id": "57e7914450b098771f5106acaf02be8a61894fae",
      "peer_name": "peer-0",
      "peer_id":
      "peer-57e7914450b098771f5106acaf02be8a61894fae-0.peer-57e7914450b098771f5106acaf02be8a61894fae.default.svc.cluster.local"
    }, {
      "org_name": "organization",
```

```

"org_id" : "57e7914450b098771f5106acaf02be8a61894fae",
"peer_name" : "peer-1",
"peer_id" :
"peer-57e7914450b098771f5106acaf02be8a61894fae-1.peer-57e7914450b098771f5106acaf02be8a61894fae
.default.svc.cluster.local"
}],
"security_check_status" : 2
}]
}
    
```

状态码： 400

Bad Request

```

{
"error_code" : "BCS.4000013",
"error_message" : "request body is too large"
}
    
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.6 查询链代码安装信息

功能介绍

查询某个链代码在节点上的安装信息

URI

GET /v2/agent/apis/chaincode/install

表 5-34 Query 参数

参数	是否必选	参数类型	描述
chaincode_name	是	String	链代码名称，以小写字母开头，支持小写字母和数字，长度6-25位 最小长度：6 最大长度：25

请求参数

表 5-35 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

响应参数

状态码： 200

表 5-36 响应 Body 参数

参数	参数类型	描述
result	Array of PeerInstallInfo objects	节点链代码安装信息

表 5-37 PeerInstallInfo

参数	参数类型	描述
org_name	String	节点所在组织名称
org_id	String	节点所在组织ID
peer_name	String	节点名称
peer_id	String	节点ID
install_status	String	链代码安装情况： installed / uninstalled
version	String	链代码版本号

状态码： 400

表 5-38 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

```
GET https://192.168.0.90:30603/v2/agent/apis/chaincode/install?chaincode_name=chaincode
```

响应示例

状态码： 200

Success

```
{
  "result": [ {
    "org_name": "org1",
    "org_id": "65cfb1c760f24058c865ffcfd8ce1cdb690bf2a3",
    "peer_name": "peer-0",
    "peer_id":
"peer-65cfb1c760f24058c865ffcfd8ce1cdb690bf2a3-0.peer-65cfb1c760f24058c865ffcfd8ce1cdb690bf2a3.default.svc.cluster.local",
    "install_status": "uninstalled",
    "version": ""
  }, {
    "org_name": "org1",
    "org_id": "65cfb1c760f24058c865ffcfd8ce1cdb690bf2a3",
    "peer_name": "peer-1",
    "peer_id":
"peer-65cfb1c760f24058c865ffcfd8ce1cdb690bf2a3-1.peer-65cfb1c760f24058c865ffcfd8ce1cdb690bf2a3.default.svc.cluster.local",
    "install_status": "installed",
    "version": "1.0"
  } ]
}
```

状态码： 400

Bad Request

```
{
  "error_code": "BCS.4000013",
  "error_message": "request body is too large"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.7 查询链代码实例化信息

功能介绍

查询某个链代码在区块链通道上的实例化信息

URI

GET /v2/agent/apis/chaincode/instantiate

表 5-39 Query 参数

参数	是否必选	参数类型	描述
chaincode_name	是	String	链代码名称，以小写字母开头，支持小写字母和数字，长度6-25位 最小长度：6 最大长度：25

请求参数

表 5-40 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

响应参数

状态码： 200

表 5-41 响应 Body 参数

参数	参数类型	描述
result	Array of ChannelInstantiateInfo objects	通道链代码实例化信息

表 5-42 ChannelInstantiateInfo

参数	参数类型	描述
channel_name	String	通道名称
instantiate_info	InstantiateInfo object	实例化信息
endorsement_policy	String	背书策略

参数	参数类型	描述
version	String	链代码版本
orgs_info	Array of OrgInfo objects	通道组织信息
has_private_data	Integer	是否有隐私数据，1表示有，0表示无

表 5-43 InstantiateInfo

参数	参数类型	描述
status	String	实例化状态，取值有 CHAINCODE_INSTANTIATED（实例化成功）,CHAINCODE_INSTANTIATION_INPROGRESS（实例化进行中）,CHAINCODE_INSTANTIATION_FAILED（实例化失败）
code	String	实例化结果编码
reason	String	实例化结果理由
detail	String	实例化结果详情

表 5-44 OrgInfo

参数	参数类型	描述
org_name	String	组织名称
org_id	String	组织ID

状态码： 400

表 5-45 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

```
GET https://192.168.0.90:30603/v2/agent/apis/chaincode/instantiate?chaincode_name=chaincode
```

响应示例

状态码： 200

Success

```
{
  "result": [ {
    "channel_name": "channel",
    "instantiate_info": {
      "status": "CHAINCODE_INSTANTIATED",
      "code": "1000",
      "reason": "1000",
      "detail": ""
    },
    "endorsement_policy": "OR,org1,org2",
    "version": "2.0",
    "orgs_info": [ {
      "org_name": "org1",
      "org_id": "65cfb1c760f24058c865ffcf8ce1cdb690bf2a3"
    }, {
      "org_name": "org2",
      "org_id": "a48c4ed995238eceaee3fe738f1871b2e58db350"
    } ],
    "has_private_data": 0
  } ]
}
```

状态码： 400

Bad Request

```
{
  "error_code": "BCS.4000013",
  "error_message": "request body is too large"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.8 查询应用链信息

功能介绍

通道概要信息查询

URI

GET /v2/agent/apis/channel/{channel_name}/summary

表 5-46 路径参数

参数	是否必选	参数类型	描述
channel_name	是	String	通道名称

请求参数

表 5-47 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token。通过调用获取TokenAPI获取Token。

响应参数

状态码： 200

表 5-48 响应 Body 参数

参数	参数类型	描述
peer_num	Number	peer数量
chaincodes	Number	合约数量
block_num	Number	区块数量
transaction_num	Number	交易数量
bph	Array of bph objects	区块产生数量统计，按小时统计
bpm	Array of bpm objects	区块产生数量统计，按每5分钟统计
tph	Array of tph objects	合约交易数量统计，按小时统计
tpm	Array of tpm objects	合约交易数量统计，按每5分钟统计
orgs_map	Array of orgs_map objects	组织交易统计

参数	参数类型	描述
peers_list	Array of peers_list objects	peer列表

表 5-49 bph

参数	参数类型	描述
block	Number	区块数量
time	String	时间

表 5-50 bpm

参数	参数类型	描述
block	Number	区块数量
time	String	时间

表 5-51 tph

参数	参数类型	描述
tx	Number	交易数量
time	String	时间

表 5-52 tpm

参数	参数类型	描述
tx	Number	交易数量
time	String	时间

表 5-53 orgs_map

参数	参数类型	描述
org_name	String	组织名称
tx_num	Number	交易数量

表 5-54 peers_list

参数	参数类型	描述
org_name	String	组织名称
org_id	String	组织ID
peer	String	Peer名称
url	String	PeerURL
status	String	Peer状态

状态码： 400

表 5-55 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

GET https://192.168.0.90:30603/v2/agent/apis/channel/channel/summary

响应示例

状态码： 400

Bad Request

```
{
  "error_code" : "BCS.4000013",
  "error_message" : "request body is too large"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.9 查询区块列表

功能介绍

查询区块列表

URI

GET /v2/agent/apis/channel/{channel_name}/blocks

表 5-56 路径参数

参数	是否必选	参数类型	描述
channel_name	是	String	通道名称，名称长度限制：4-24，不能与系统通道名称（testchainid）相同

表 5-57 Query 参数

参数	是否必选	参数类型	描述
is_pagination	否	Boolean	是否分页，默认为否 缺省值： false
offset	否	Integer	偏移量，默认为0 最小值： 0 缺省值： 0
limit	否	Integer	每页显示的条目数量，默认为10，取值范围为1-50 最小值： 1 最大值： 50 缺省值： 10

请求参数

表 5-58 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

响应参数

状态码： 200

表 5-59 响应 Body 参数

参数	参数类型	描述
count	Integer	区块总数
data	Array of BlockHeader objects	区块

表 5-60 BlockHeader

参数	参数类型	描述
block_number	Integer	区块号
block_hash	String	区块哈希
transaction_count	Integer	交易总数
data_hash	String	数据哈希
previous_hash	String	前一个区块哈希
timestamp	String	时间戳
organization_maps	Map<String,Integer>	key:creatorMSP, value:数量

状态码： 400

表 5-61 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

- 默认模式请求

```
GET https://192.168.0.90:30603/v2/agent/apis/channel/channel/blocks
```

- 分页模式请求

```
GET https://192.168.0.90:30603/v2/agent/apis/channel/channel/blocks?is_pagination=true&offset=1&limit=50
```

响应示例

状态码： 200

Success

```
{
  "count" : 1,
  "data" : [{
    "block_number" : 0,
    "block_hash" : "Yux2Ea0RNZM95+3R95mvdll8mH1dvmTSTylPwwzMsby",
    "transaction_count" : 1,
    "data_hash" : "+3B6RFFbQisE8zt2BeWTVcWp1JbmQLIPeQoiKxoCQVA",
    "previous_hash" : "FIECLYDQJ4cHaEslex9usupte0EqbHyymQ+zUaQcJyE",
    "timestamp" : "2021-01-20T14:38:39+08:00",
    "organization_maps" : "{\"d565e95144ae53b9b3f556de613513f257e720ecMSP\":49}"
  }]
}
```

状态码： 400

Bad Request

```
{
  "error_code" : "BCS.4000013",
  "error_msg" : "request body is too large"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.10 查询交易列表

功能介绍

查询交易列表

URI

GET /v2/agent/apis/channel/{channel_name}/transactions

表 5-62 路径参数

参数	是否必选	参数类型	描述
channel_name	是	String	通道名称，名称长度限制：4-24，不能与系统通道名称（testchainid）相同

表 5-63 Query 参数

参数	是否必选	参数类型	描述
is_pagination	否	Boolean	是否分页，默认为否 缺省值: false
offset	否	Integer	偏移量，默认为0 最小值: 0 缺省值: 0
limit	否	Integer	每页显示的条目数量，默认为10，取值范围为1-50 最小值: 1 最大值: 50 缺省值: 10

请求参数

表 5-64 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

响应参数

状态码: 200

表 5-65 响应 Body 参数

参数	参数类型	描述
count	Integer	交易总数
data	Array of TransactionSummary objects	交易

表 5-66 TransactionSummary

参数	参数类型	描述
organization_name	String	创建者组织
type	String	交易类型
transaction_id	String	交易id
chaincode_name	String	链码名称
timestamp	String	时间戳
channel_name	String	通道名称
creator_msp	String	身份信息
chaincode_version	String	链码版本
block_number	Integer	区块号

状态码： 400

表 5-67 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

- 默认模式请求
GET https://192.168.0.90:30603/v2/agent/apis/channel/channel/transactions
- 分页模式请求
GET https://192.168.0.90:30603/v2/agent/apis/channel/channel/transactions?is_pagination=true&offset=1&limit=50

响应示例

状态码： 200

Success

```
{
  "count": 1,
```

```
"data": [{
  "block_number": 0,
  "transaction_id": "",
  "channel_name": "channel",
  "creator_msp": "e784724be5ed75f59b2809e4f0965a10679ae113MSP",
  "type": "CONFIG",
  "chaincode_name": "",
  "chaincode_version": "",
  "timestamp": "2021-01-20T14:38:27+08:00",
  "organization_name": "orderer"
}]
}
```

状态码: 400

Bad Request

```
{
  "error_code": "BCS.4000013",
  "error_msg": "request body is too large"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.11 查询交易总数

功能介绍

查询交易总数

URI

GET /v2/agent/apis/channel/{channel_name}/transactions/count

表 5-68 路径参数

参数	是否必选	参数类型	描述
channel_name	是	String	通道名称

请求参数

表 5-69 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

响应参数

状态码： 200

表 5-70 响应 Body 参数

参数	参数类型	描述
channel_id	String	通道id
block_height	Integer	区块高度
transaction_number	Integer	交易数量

状态码： 400

表 5-71 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

GET https://192.168.0.90:30603/v2/agent/apis/channel/channel/transactions/count

响应示例

状态码： 200

Success

```
{
  "channel_id": "channel",
  "block_height": 2,
  "transaction_num": 2
}
```

状态码：400

Bad Request

```
{
  "error_code": "BCS.4000013",
  "error_message": "request body is too large"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.12 查询区块交易列表

功能介绍

查询区块交易列表

URI

GET /v2/agent/apis/channel/{channel_name}/blocks/{block_num}/transactions

表 5-72 路径参数

参数	是否必选	参数类型	描述
channel_name	是	String	通道名称，名称长度限制：4-24，不能与系统通道名称（testchainid）相同
block_num	是	String	区块号

请求参数

表 5-73 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

响应参数

状态码： 200

表 5-74 响应 Body 参数

参数	参数类型	描述
[数组元素]	Array of ShowTransactionDetailRes objects	Success

表 5-75 ShowTransactionDetailRes

参数	参数类型	描述
read_set	Map<String,Array< KVRead >>	读集 "map[string][]KVRead key:chaincode value:键值对数组"
write_set	Map<String,Array< KVWrite >>	写集 "map[string][]KVWrite key:chaincode value:键值对数组"
validation_code	String	验证代码
endorser_organizations	Array of strings	背书组织
proposal_hash	String	请求数据哈希
transaction_summary	Object	交易详情

表 5-76 KVRead

参数	参数类型	描述
key	String	读键
version	version object	读集键的版本

表 5-77 version

参数	参数类型	描述
block_num	Integer	区块数

参数	参数类型	描述
tx_num	Integer	交易数

表 5-78 KVWrite

参数	参数类型	描述
key	String	写键
is_delete	Boolean	是否删除
value	String	写值

表 5-79 TransactionSummary

参数	参数类型	描述
organization_name	String	创建者组织
type	String	交易类型
transaction_id	String	交易id
chaincode_name	String	链码名称
timestamp	String	时间戳
channel_name	String	通道名称
creator_msp	String	身份信息
chaincode_version	String	链码版本
block_number	Integer	区块号

状态码： 400

表 5-80 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述

参数	参数类型	描述
error_msg	String	错误描述

请求示例

```
GET https://192.168.0.90:30603/v2/agent/apis/channel/channel/blocks/1/transactions
```

响应示例

状态码： 200

Success

```
[ {
  "transaction_summary": {
    "block_number": 29,
    "transaction_id": "6d704b217e17e16de71029b70f17a1ced35c055279f655dfd096bebf978a0546",
    "channelName": "channel",
    "creator_msp": "282f3c713ea1cec646aa7c640defca9c4f64bd88MSP",
    "type": "ENDORSER_TRANSACTION",
    "chaincode_name": "kvtest",
    "chaincode_version": "1.0",
    "timestamp": "2021-01-20T19:30:28+08:00",
    "organization_name": "organization"
  },
  "validation_code": "VALID",
  "endorser_organizations": [ "282f3c713ea1cec646aa7c640defca9c4f64bd88MSP" ],
  "proposal_hash": "k1h2ewweWGrWNmmcu7UvzJ8Aw2G190SQzV+lBAAl4gw=",
  "read_set": {
    "kvtest": null,
    "lsc": [ {
      "key": "kvtest",
      "version": {
        "block_num": 2
      }
    } ]
  },
  "write_set": {
    "kvtest": [ {
      "key": "a1",
      "is_delete": false,
      "value": "1"
    } ],
    "lsc": [ ]
  }
} ]
```

状态码： 400

Bad Request

```
{
  "error_code": "BCS.4000013",
  "error_msg": "request body is too large"
}
```

状态码

状态码	描述
200	Success

状态码	描述
400	Bad Request

错误码

请参见[错误码](#)。

5.3.13 查询交易详情

功能介绍

查询交易详情

URI

GET /v2/agent/apis/channel/{channel_name}/transactions/{transaction_id}/detail

表 5-81 路径参数

参数	是否必选	参数类型	描述
channel_name	是	String	通道名称
transaction_id	是	String	交易id号

请求参数

表 5-82 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

响应参数

状态码： 200

表 5-83 响应 Body 参数

参数	参数类型	描述
read_set	Map<String,Array< KVRead >>	读集 "map[string][]KVRead key:chaincode value:键值对数组"

参数	参数类型	描述
write_set	Map<String,Array<KVWrite>>	写集 "map[string][]KVWrite key:chaincode value:键值对数组"
validation_code	String	验证代码
endorser_organizations	Array of strings	背书组织
proposal_hash	String	请求数据哈希
transaction_summary	Object	交易详情

表 5-84 KVRead

参数	参数类型	描述
key	String	读键
version	version object	读集键的版本

表 5-85 version

参数	参数类型	描述
block_num	Integer	区块数
tx_num	Integer	交易数

表 5-86 KVWrite

参数	参数类型	描述
key	String	写键
is_delete	Boolean	是否删除
value	String	写值

表 5-87 TransactionSummary

参数	参数类型	描述
organization_name	String	创建者组织
type	String	交易类型
transaction_id	String	交易id
chaincode_name	String	链码名称
timestamp	String	时间戳
channel_name	String	通道名称
creator_msp	String	身份信息
chaincode_version	String	链码版本
block_number	Integer	区块号

状态码： 400

表 5-88 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

GET https://192.168.0.90:30603/v2/agent/apis/channel/channel/transactions/1111111/detail

响应示例

状态码： 200

Success

```
{
  "transaction_summary": {
    "block_number": 29,
    "transaction_id": "6d704b217e17e16de71029b70f17a1ced35c055279f655dfd096bebf978a0546",
    "channelName": "channel",
    "creator_msp": "282f3c713ea1cec646aa7c640defca9c4f64bd88MSP",
    "type": "ENDORSE_TRANSACTION",
  }
}
```

```

"chaincode_name": "kvtest",
"chaincode_version": "1.0",
"timestamp": "2021-01-20T19:30:28+08:00",
"organization_name": "organization"
},
"validation_code": "VALID",
"endorser_organizations": [ "282f3c713ea1cec646aa7c640defca9c4f64bd88MSP" ],
"proposal_hash": "k1h2ewweWGrWNmmcu7UvzJ8Aw2G190SQzV+lBAAl4gw=",
"read_set": {
  "kvtest": null,
  "lsc": [ {
    "key": "kvtest",
    "version": {
      "block_num": 2
    }
  }
]
},
"write_set": {
  "kvtest": [ {
    "key": "a1",
    "is_delete": false,
    "value": "1"
  } ],
  "lsc": [ ]
}
}

```

状态码: 400

Bad Request

```

{
  "error_code": "BCS.4000013",
  "error_message": "request body is too large"
}

```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.14 查询节点状态

功能介绍

查询节点状态

URI

GET /v2/agent/apis/peers

请求参数

表 5-89 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

响应参数

状态码： 200

表 5-90 响应 Body 参数

参数	参数类型	描述
peers	Map<String,PeerInfo>	key:节点域名, value:节点详情

表 5-91 PeerInfo

参数	参数类型	描述
org_name	String	组织名称
org_id	String	组织id
peer_name	String	节点名称
peer_id	String	节点id
status	String	节点状态
channels	Array of strings	未实例化的peer信息
url	String	Peer节点的url信息
peer	String	Peer节点的内部域名

状态码： 400

表 5-92 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述

参数	参数类型	描述
error_msg	String	错误描述

请求示例

```
GET https://192.168.0.90:30603/v2/agent/apis/peers
```

响应示例

状态码： 200

Success

```
{
  "peers": {
    "peer-a9e940a9e947e8af0c9c2fb98d0129e56210d6b5-0.peer-
a9e940a9e947e8af0c9c2fb98d0129e56210d6b5.default.svc.cluster.local": {
      "org_name": "organization",
      "org_id": "a9e940a9e947e8af0c9c2fb98d0129e56210d6b5",
      "peer": "peer-a9e940a9e947e8af0c9c2fb98d0129e56210d6b5-0.peer-
a9e940a9e947e8af0c9c2fb98d0129e56210d6b5.default.svc.cluster.local",
      "peer_name": "peer-a9e940a9e947e8af0c9c2fb98d0129e56210d6b5-0",
      "url": "100.95.146.117:30610",
      "channels": [ "channel" ],
      "status": "running"
    },
    "peer-a9e940a9e947e8af0c9c2fb98d0129e56210d6b5-1.peer-
a9e940a9e947e8af0c9c2fb98d0129e56210d6b5.default.svc.cluster.local": {
      "org_name": "organization",
      "org_id": "a9e940a9e947e8af0c9c2fb98d0129e56210d6b5",
      "peer": "peer-a9e940a9e947e8af0c9c2fb98d0129e56210d6b5-1.peer-
a9e940a9e947e8af0c9c2fb98d0129e56210d6b5.default.svc.cluster.local",
      "peer_name": "peer-a9e940a9e947e8af0c9c2fb98d0129e56210d6b5-1",
      "url": "100.95.146.117:30611",
      "channels": [ "channel" ],
      "status": "running"
    }
  }
}
```

状态码： 400

Bad Request

```
{
  "error_code": "BCS.4000013",
  "error_message": "request body is too large"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.15 删除链代码

功能介绍

删除区块链节点上的链代码

URI

DELETE /v2/agent/apis/chaincode/uninstall

请求参数

表 5-93 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

表 5-94 请求 Body 参数

参数	是否必选	参数类型	描述
chaincode_name	是	String	链代码名称,以小写字母开头,支持小写字母和数字,长度6-25位 最小长度: 6 最大长度: 25
chaincode_version	是	String	链代码版本,只允许使用数字、点(.)、横杠(-),必须以数字开头和结尾,且点和横杠不能相邻
target_peers	是	Array of TargetPeer objects	卸载链代码peer信息

表 5-95 TargetPeer

参数	是否必选	参数类型	描述
org_id	是	String	peer所属组织ID
peer_id	是	String	Peer ID

响应参数

状态码： 200

表 5-96 响应 Body 参数

参数	参数类型	描述
total_peer_num	Integer	链代码操作Peer总数
success_peer_num	Integer	操作成功Peer数量
fail_peer_num	Integer	操作失败Peer数量
fail_peers	Array of strings	操作失败Peer信息

状态码： 400

表 5-97 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

```
DELETE https://192.168.0.90:30603/v2/agent/apis/chaincode/uninstall
{
  "chaincode_name": "chaincode1",
  "chaincode_version": "1.0",
  "target_peers": [ {
    "org_id": "9802af57cfab764dc12b860c44b01969575e83c9",
    "peer_id":
"peer-9802af57cfab764dc12b860c44b01969575e83c9-1.peer-9802af57cfab764dc12b860c44b01969575e83c9
.default.svc.cluster.local"
  } ]
}
```

响应示例

状态码： 200

Success

```
{
  "total_peer_num": 4,
  "success_peer_num": 4,
```

```
"fail_peer_num" : 0,
"fail_peers" : [ ]
}
```

状态码： 400

Bad Request

```
{
"error_code" : "BCS.4000013",
"error_message" : "request body is too large"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

5.3.16 下载报告

功能介绍

下载链代码安全检查报告

URI

POST /v2/agent/apis/chaincode/report

请求参数

表 5-98 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户Token

表 5-99 请求 Body 参数

参数	是否必选	参数类型	描述
chaincode_name	是	String	链代码名称，与安装链代码时填写一致，需开启链代码检查选项
chaincode_version	是	String	链代码版本，与安装链代码时填写一致

响应参数

状态码： 200

表 5-100 响应 Body 参数

参数	参数类型	描述
-	File	Success

状态码： 400

表 5-101 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码
error_message	String	错误描述
error_msg	String	错误描述

请求示例

POST https://192.168.0.90:30603/v2/agent/apis/chaincode/report

```
{
  "chaincode_name": "chaincode1",
  "chaincode_version": "1.0"
}
```

响应示例

状态码： 400

Bad Request

```
{
  "error_code": "BCS.4002068",
  "error_message": "failed to get report, not existed"
}
```

状态码

状态码	描述
200	Success
400	Bad Request

错误码

请参见[错误码](#)。

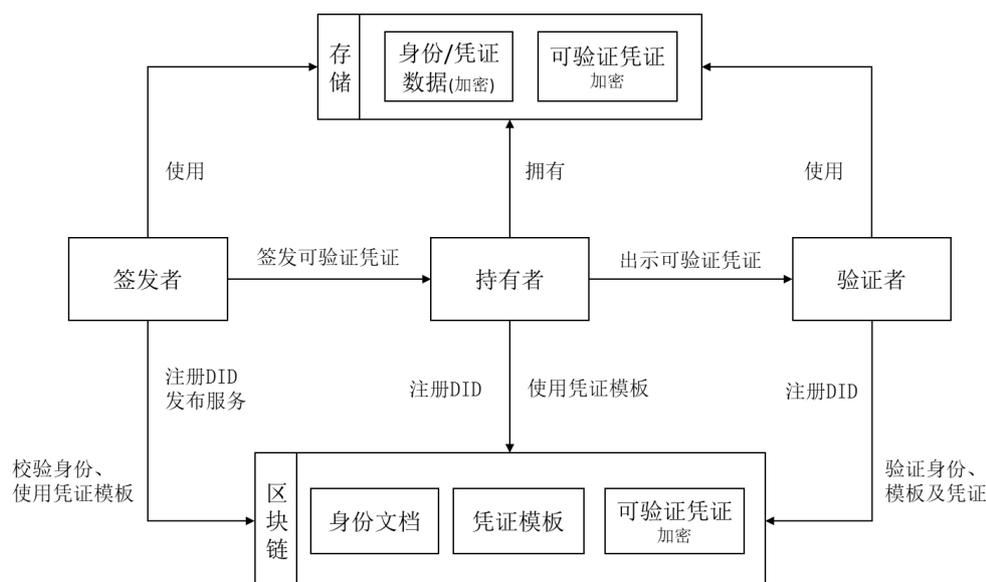
5.4 分布式身份（公测）

5.4.1 概述

分布式身份是一种基于区块链的分布式身份管理技术，提供用户身份的创建、可验证凭证的注册、签发、验证等功能，该特性基于W3C分布式身份(Decentralized Identifiers, DIDs)和可验证凭证(Verifiable Credentials, VC)的标准实现，为个人和企业用户提供统一的、可自解释的、移植性强的分布式身份标识，有效解决跨部门、跨企业、跨地域的身份认证难和隐私泄露等问题。

本文为您介绍分布式身份（DID）管理的实现流程和使用方式。详细请参见[图5-1](#)、[图5-2](#)、[图5-3](#)和[图5-4](#)。

图 5-1 分布式身份实现架构图



实现流程

1. 各个角色可通过企业身份注册(带有service)和注册DID生成完全由自己控制的分布式身份，并将身份文档发布到区块链上完成身份的注册。企业用户的DID身份中可以包含其所能提供的服务信息，以支持多样的应用场景。

📖 说明

1. 主要有三种角色：签发者、持有者和验证者，其中每个角色都可以是设备、应用、个人或者组织。
2. 具备基础的身份标识之后，通过可验证凭证架设起身份与身份之间的认证体系。凭证的模板会由相关主体注册发布到区块链上，并持续维护。持有者便可以向签发者发起认证申请，获得凭证后出示给验证者完成校验。
3. 验证者可以通过接口验证持有者出示的“可验证凭证”，确保其是否有权限和资质开展后续业务。

使用方式

分布式身份中间件是部署在用户侧的一套微服务，简化用户调用区块链相关接口的复杂操作。因此接口调用时需要传入用户私钥和被fabric组织根证书签名的证书。

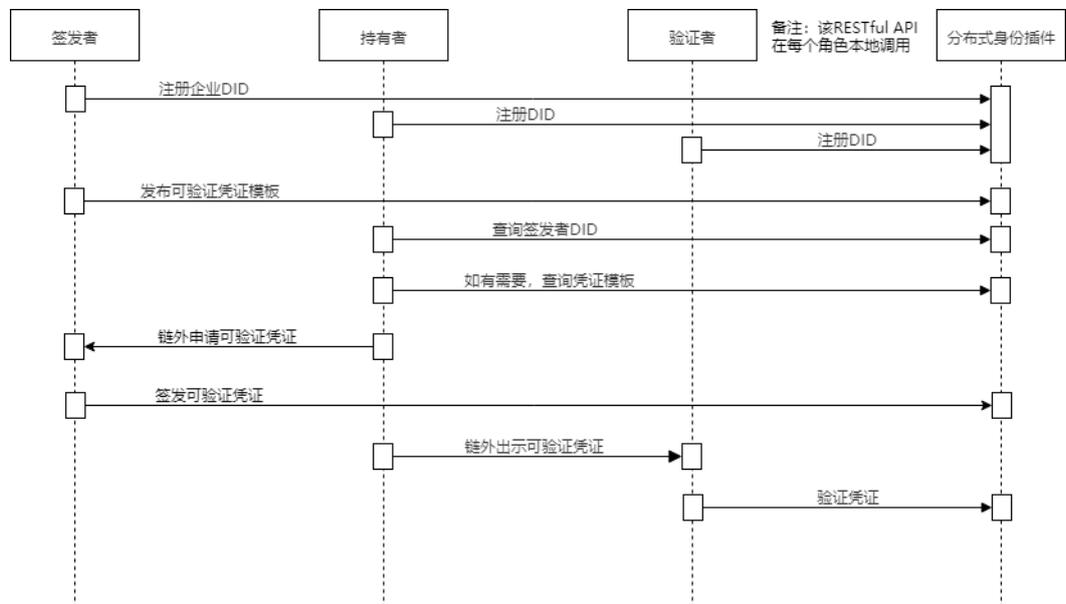
说明

获取用户私钥和证书的方式有两种，BCS区块链管理界面下载和使用openssl工具生成，详细方法请参见[获取fabric用户私钥及证书的方法](#)。

根据持有者申请可验证凭证的方式，将分布式身份服务的使用分为链外申请模式和链上申请模式。

- 链外申请模式中，持有者将申请可验证凭证的身份/凭证数据直接发送给签发者。
- 链上申请模式中，持有者将申请可验证凭证的身份/凭证数据加密存储于区块链。

图 5-2 分布式身份使用时序图(链外申请模式)



链上申请模式中，根据持有者与签发者之间是否需要通信信道，分为在线申请和离线申请。

图 5-3 分布式身份使用时序图(链上申请-在线申请模式)

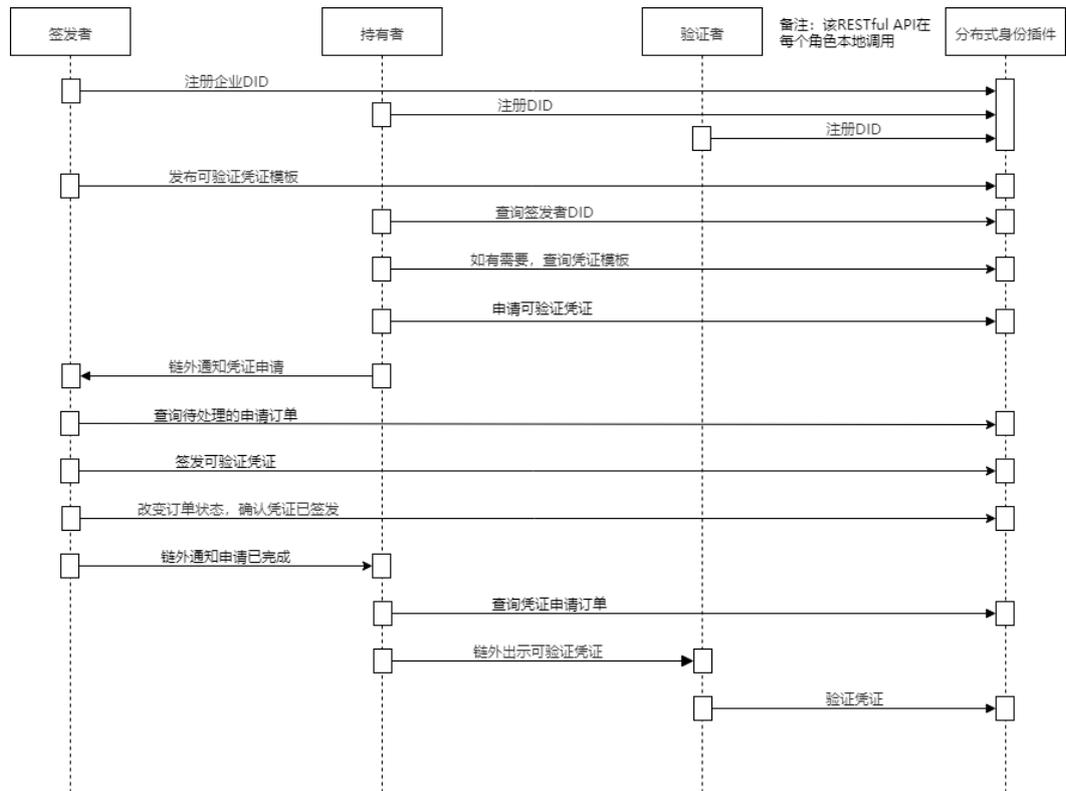
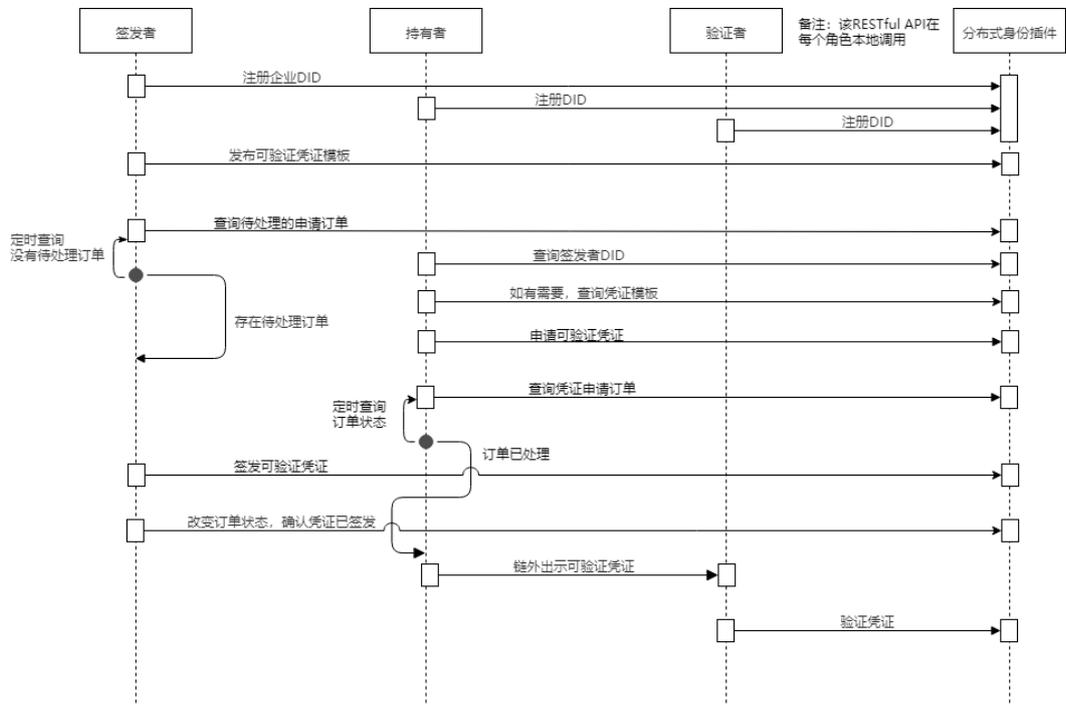


图 5-4 分布式身份使用时序图(链上申请-离线申请模式)



5.4.2 分布式身份(DID)管理

5.4.2.1 企业身份注册(带有 service)

功能介绍

分布式身份注册方法。在使用该方法前需要先使用openssl工具生成每个用户的私钥和被fabric组织根证书签名的证书(或通过BCS区块链管理界面下载用户证书)。注册时需声明可提供的服务列表。

URI

POST /v1/identity/firm-did

请求参数

表 5-102 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法, 目前固定为SW
cert	是	String	用户证书, 每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥, 每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
service	是	Array of DIDService objects	提供的服务

表 5-103 DIDService

参数	是否必选	参数类型	描述
type	是	String	类型
serviceEndpoint	是	String	接入点
credentialApplySchema	否	CredentialApplySchema object	申请凭证所需数据的Schema

表 5-104 CredentialApplySchema

参数	是否必选	参数类型	描述
type	否	String	类型
name	否	String	名称
description	否	String	描述信息
attributes	否	Array of Attribute objects	属性列表

表 5-105 Attribute

参数	是否必选	参数类型	描述
name	否	String	名称
type	否	String	类型
description	否	String	描述信息

响应参数

状态码： 200

表 5-106 响应 Body 参数

参数	参数类型	描述
did	String	分布式身份标识

状态码： 500

表 5-107 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度： 8 最大长度： 36
errorMsg	String	错误描述 最小长度： 2 最大长度： 512

请求示例

```
{
  "orgID" : "4f1439758ebb41f7411b5f684b67713c08b89198",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "service" : [ {
    "type" : "VerifiableCredentialService",
    "serviceEndpoint" : "https://example.com/vc/",
    "credentialApplySchema" : {
      "type" : "file",
      "name" : "Test Enterprise Certification",
      "description" : "this is test apply info",
      "attributes" : [ {
        "name" : "bob",
        "type" : "string",
        "description" : "Attribute's description"
      } ]
    }
  } ]
}
```

响应示例

状态码： 200

分布式身份标识

```
{
  "did" : "did:example:2THjdfbKMLDVcoYvkiepr9"
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "BCS.5002033",
  "errorMsg" : "Service Type and ServiceEndpoint Can not Null"
}
```

状态码

状态码	描述
200	分布式身份标识
500	失败响应

错误码

请参见[错误码](#)。

5.4.2.2 注册 DID

功能介绍

DID快速注册方法。可以方便的注册发布一个DID，该DID不提供服务，只拥有一个公钥。

URI

POST /v1/identity/did

请求参数

表 5-108 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳

响应参数

状态码： 200

表 5-109 响应 Body 参数

参数	参数类型	描述
did	String	分布式身份标识

状态码： 500

表 5-110 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度：8 最大长度：36
errorMsg	String	错误描述 最小长度：2 最大长度：512

请求示例

```
{
  "orgID" : "4f1439758ebb41f7411b5f684b67713c08b89198",
  "channelID" : "channel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PUBLIC KEY-----\n...\n-----END PUBLIC KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00"
}
```

响应示例

状态码： 200

分布式身份标识

```
{
  "did" : "did:example:2HjdffbKMLDVcoYvkiepr9"
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "BCS.5002033",
  "errorMsg" : "Service Type and ServiceEndpoint Can not Null"
}
```

状态码

状态码	描述
200	分布式身份标识
500	失败响应

错误码

请参见[错误码](#)。

5.4.2.3 更新 DID

功能介绍

更新DID文档中发布的服务。

URI

PUT /v1/identity/did

请求参数

表 5-111 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
did	是	String	分布式身份标识
service	否	Array of DIDService objects	服务列表

表 5-112 DIDService

参数	是否必选	参数类型	描述
type	是	String	类型
serviceEndpoint	是	String	接入点
credentialApplySchema	否	CredentialApplySchema object	申请凭证所需数据的Schema object

表 5-113 CredentialApplySchema

参数	是否必选	参数类型	描述
type	否	String	类型
name	否	String	名称
description	否	String	描述信息
attributes	否	Array of Attribute objects	属性列表

表 5-114 Attribute

参数	是否必选	参数类型	描述
name	否	String	名称
type	否	String	类型
description	否	String	描述信息

响应参数

状态码： 200

表 5-115 响应 Body 参数

参数	参数类型	描述
context	String	context
id	String	分布式身份标识
publicKey	Array of DocPublicKey objects	公钥列表
authentication	Array of strings	did主公钥标识
recovery	String	备用公钥标识，可用于修改主密钥
service	Array of Service objects	服务列表
proof	Proof object	证明结构，可为空
created	String	创建时间
updated	String	更新时间

参数	参数类型	描述
status	String	状态

表 5-116 DocPublicKey

参数	参数类型	描述
id	String	公钥标识
type	String	公钥类型
controller	String	公钥的控制者标识
publicKeyPem	String	公钥证书

表 5-117 Service

参数	参数类型	描述
id	String	服务标识
type	String	服务类型
serviceEndpoint	String	服务介绍网址
credentialApplySchema	CredentialApplySchema object	申请凭证所需数据的Schema

表 5-118 CredentialApplySchema

参数	参数类型	描述
type	String	类型
name	String	名称
description	String	描述信息
attributes	Array of Attribute objects	属性列表

表 5-119 Attribute

参数	参数类型	描述
name	String	名称
type	String	类型
description	String	描述信息

表 5-120 Proof

参数	参数类型	描述
creator	String	创建者身份标识
type	String	签名类型
created	String	签名创建时间
signatureValue	String	签名值

状态码： 500

表 5-121 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度： 8 最大长度： 36
errorMsg	String	错误描述 最小长度： 2 最大长度： 512

请求示例

```
{
  "orgID": "4f1439758ebb41f7411b5f684b67713c08b89198",
  "channelID": "mychannel",
  "cryptoMethod": "SW",
  "cert": "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk": "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp": "2020-10-27T17:28:16+08:00",
  "did": "did:example:ebfeb1f712ebc6f1c276e12ec21",
  "service": [ {
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/",
    "credentialApplySchema": {
      "type": "file",
      "name": "Test Enterprise Certification",

```

```

    "description" : "this is test apply info",
    "attributes" : [{
      "name" : "bob",
      "type" : "string",
      "description" : "Attribute's description"
    }]
  }
}]
}

```

响应示例

状态码： 500

失败响应

```

{
  "errorCode" : "BCS.5002033",
  "errorMsg" : "Service Type and ServiceEndpoint Can not Null"
}

```

状态码

状态码	描述
200	分布式身份文档结构体
500	失败响应

错误码

请参见[错误码](#)。

5.4.2.4 查询 DID

功能介绍

查询指定DID的文档

URI

POST /v1/identity/query-did

请求参数

表 5-122 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW

参数	是否必选	参数类型	描述
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
did	是	String	分布式身份标识

响应参数

状态码： 200

表 5-123 响应 Body 参数

参数	参数类型	描述
context	String	context
id	String	分布式身份标识
publicKey	Array of DocPublicKey objects	公钥列表
authentication	Array of strings	did主公钥标识
recovery	String	备用公钥标识，可用于修改主密钥
service	Array of Service objects	服务列表
proof	Proof object	证明结构，可为空
created	String	创建时间
updated	String	更新时间
status	String	状态

表 5-124 DocPublicKey

参数	参数类型	描述
id	String	公钥标识
type	String	公钥类型

参数	参数类型	描述
controller	String	公钥的控制者标识
publicKeyPem	String	公钥证书

表 5-125 Service

参数	参数类型	描述
id	String	服务标识
type	String	服务类型
serviceEndpoint	String	服务介绍网址
credentialApplySchema	CredentialApplySchema object	申请凭证所需数据的Schema

表 5-126 CredentialApplySchema

参数	参数类型	描述
type	String	类型
name	String	名称
description	String	描述信息
attributes	Array of Attribute objects	属性列表

表 5-127 Attribute

参数	参数类型	描述
name	String	名称
type	String	类型
description	String	描述信息

表 5-128 Proof

参数	参数类型	描述
creator	String	创建者身份标识
type	String	签名类型
created	String	签名创建时间
signatureValue	String	签名值

状态码： 500

表 5-129 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度： 8 最大长度： 36
errorMsg	String	错误描述 最小长度： 2 最大长度： 512

请求示例

```
{
  "orgID" : "4f1439758ebb41f7411b5f684b67713c08b89198",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "did" : "did:example:ebfeb1f712ebc6f1c276e12ec21"
}
```

响应示例

状态码： 200

分布式身份文档结构体

```
{
  "context" : "https://www.w3.org/ns/did/v1",
  "id" : "did:example:ebfeb1f712ebc6f1c276e12ec21",
  "publicKey" : [ {
    "id" : "string",
    "type" : "string",
    "controller" : "string",
    "publicKeyPem" : "string"
  } ],
  "authentication" : [ "did:example:ebfeb1f712ebc6f1c276e12ec21#key-0" ],
  "recovery" : "string",
}
```

```

"service": [ {
  "id": "did:example:ebfeb1f712ebc6f1c276e12ec21#xdi",
  "type": "XdiService",
  "serviceEndpoint": "https://xdi.example.com/8377464",
  "credentialApplySchema": {
    "type": "LegalCitizen",
    "name": "LegalCitizen",
    "description": "Certified Chinese citizens",
    "attributes": [ {
      "name": "name",
      "type": "someType",
      "description": "Identity number"
    } ]
  }
} ],
"proof": {
  "creator": "did:example:ebfeb1f712ebc6f1c276e12ec21",
  "type": "RsaSignature2018",
  "created": "1606720551",
  "signatureValue": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0I
PAYuNzVBAh4vGHSrQyHUdBBPM"
},
"created": "1588921521",
"updated": "",
"status": "active"
}

```

状态码： 500

失败响应

```

{
  "errorCode": "BCS.5002034",
  "errorMsg": "request timed out or been cancelled"
}

```

状态码

状态码	描述
200	分布式身份文档结构体
500	失败响应

错误码

请参见[错误码](#)。

5.4.3 可验证凭证(VC)管理

5.4.3.1 发布可验证凭证的模板

功能介绍

发布凭证模板数据结构。签发professional类型的凭证时会使用，因为字段较多容易不统一，需要提前有模板来约束和标明。

URI

POST /v1/identity/credential-schema

请求参数

表 5-130 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
title	否	String	名称
identifier	是	String	标识
attributes	否	Array of Attribute objects	属性信息
issuer	是	String	签发者身份标识

表 5-131 Attribute

参数	是否必选	参数类型	描述
name	否	String	名称
type	否	String	类型
description	否	String	描述信息

响应参数

状态码： 200

表 5-132 响应 Body 参数

参数	参数类型	描述
schemaIndex	String	模板存储在链上的索引

参数	参数类型	描述
credentialSchema	CredentialSchema object	CredentialSchema

表 5-133 CredentialSchema

参数	参数类型	描述
creator	String	创建者身份标识
title	String	名称
identifier	String	凭证模板标识
attributes	Array of Attribute objects	属性信息
version	Integer	版本

表 5-134 Attribute

参数	参数类型	描述
name	String	名称
type	String	类型
description	String	描述信息

状态码： 500

表 5-135 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度： 8 最大长度： 36
errorMsg	String	错误描述 最小长度： 2 最大长度： 512

请求示例

```
{
  "orgID" : "4f1439758ebb41f7411b5f684b67713c08b89198",
```

```
"channelID" : "mychannel",
"cryptoMethod" : "SW",
"cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
"sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
"timestamp" : "2020-10-27T17:28:16+08:00",
"title" : "string",
"identifier" : "string",
"attributes" : [ {
  "name" : "name",
  "type" : "someType",
  "description" : "Identity number"
} ],
"issuer" : "did:example:ebfeb1f712ebc6f1c276e12ec21"
}
```

响应示例

状态码： 200

VCSchemaResponseParams

```
{
  "schemaIndex" : "string",
  "credentialSchema" : {
    "creator" : "string",
    "title" : "string",
    "identifier" : "string",
    "attributes" : [ {
      "name" : "name",
      "type" : "someType",
      "description" : "Identity number"
    } ],
    "version" : 0
  }
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "BCS.5002035",
  "errorMsg" : "Schema Already Exist"
}
```

状态码

状态码	描述
200	VCSchemaResponseParams
500	失败响应

错误码

请参见[错误码](#)。

5.4.3.2 查询凭证模板

功能介绍

根据索引查询凭证模板。

URI

POST /v1/identity/query-credential-schema

请求参数

表 5-136 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
schemaIndex	是	String	schema索引

响应参数

状态码： 200

表 5-137 响应 Body 参数

参数	参数类型	描述
creator	String	创建者身份标识
title	String	名称
identifier	String	凭证模板标识
attributes	Array of Attribute objects	属性信息
version	Integer	版本

表 5-138 Attribute

参数	参数类型	描述
name	String	名称
type	String	类型
description	String	描述信息

状态码： 500

表 5-139 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度： 8 最大长度： 36
errorMsg	String	错误描述 最小长度： 2 最大长度： 512

请求示例

```
{
  "orgID" : "4f1439758ebb41f7411b5f684b67713c08b89198",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "schemaIndex" : "1"
}
```

响应示例

状态码： 200

CredentialSchema Information

```
{
  "creator" : "string",
  "title" : "string",
  "identifier" : "string",
  "attributes" : [ {
    "name" : "name",
    "type" : "someType",
    "description" : "Identity number"
  } ],
  "version" : 0
}
```

状态码： 500

失败响应

```
{
  "errorCode": "stringst",
  "errorMsg": "string"
}
```

状态码

状态码	描述
200	CredentialSchema Information
500	失败响应

错误码

请参见[错误码](#)。

5.4.3.3 申请可验证凭证

功能介绍

申请可验证凭证。根据在服务中声明的必要字段，申请者需要提供相关数据。

URI

POST /v1/identity/apply-vc

请求参数

表 5-140 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
applier	是	String	申请者的身份标识
serviceID	是	String	服务的标识符。服务提供者did中声明的service的id
data	否	String	data

响应参数

状态码： 200

表 5-141 响应 Body 参数

参数	参数类型	描述
orderIndex	String	订单索引

状态码： 500

表 5-142 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度： 8 最大长度： 36
errorMsg	String	错误描述 最小长度： 2 最大长度： 512

请求示例

```
{
  "orgID" : "4f1439758ebb41f7411b5f684b67713c08b89198",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "applier" : "did:example:ebfeb1f712ebc6f1c276e12ec21",
  "serviceID" : "did:example:ebfeb1f712ebc6f1c276e12ec21#service1",
  "data" : "abcdefg"
}
```

响应示例

状态码： 200

VCOOrderResponseParams Information

```
{
  "orderIndex" : "string"
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "stringst",
}
```

```
"errorMsg" : "string"
}
```

状态码

状态码	描述
200	VCOOrderResponseParams Information
500	失败响应

错误码

请参见[错误码](#)。

5.4.3.4 签发者确认凭证已签发

功能介绍

签发者确认申请订单，将签发的凭证索引更新到订单中。

URI

POST /v1/identity/vc-order

请求参数

表 5-143 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
orderIndex	是	String	订单索引
vcIndex	是	String	凭证索引

响应参数

状态码： 200

表 5-144 响应 Body 参数

参数	参数类型	描述
orderIndex	String	订单索引
result	String	结果

状态码： 500

表 5-145 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度： 8 最大长度： 36
errorMsg	String	错误描述 最小长度： 2 最大长度： 512

请求示例

```
{
  "orgID" : "4f1439758ebb41f7411b5f684b67713c08b89198",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "orderIndex" : 1,
  "vcIndex" : 0
}
```

响应示例

状态码： 200

ConfirmOrderResponseParams Information

```
{
  "orderIndex" : "string",
  "result" : "string"
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "stringst",
  "errorMsg" : "string"
}
```

状态码

状态码	描述
200	ConfirmOrderResponseParams Information
500	失败响应

错误码

请参见[错误码](#)。

5.4.3.5 查询凭证申请订单

功能介绍

查询凭证的申请订单

URI

POST /v1/identity/query-vc-order

请求参数

表 5-146 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
orderIndex	是	String	订单索引

响应参数

状态码： 200

表 5-147 响应 Body 参数

参数	参数类型	描述
applier	String	申请者身份标识
orderSeq	String	订单序列号
applyTime	String	申请时间
price	String	价格
status	String	状态
service	String	服务
reason	String	理由信息
dataUri	String	凭证数据URI
encryptedAesKey	String	加密的AES密钥
uriType	String	URI类型
dataHash	String	数据的hash
lockProof	String	锁定证明
vclIndex	String	凭证索引

状态码： 500

表 5-148 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度： 8 最大长度： 36
errorMsg	String	错误描述 最小长度： 2 最大长度： 512

请求示例

```
{
  "orgID" : "4f1439758ebb41f7411b5f684b67713c08b89198",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
}
```

```
{
  "orderIndex" : 1
}
```

响应示例

状态码： 200

VCOOrder Information

```
{
  "applyer" : "string",
  "orderSeq" : "string",
  "applyTime" : "string",
  "price" : "string",
  "status" : "string",
  "service" : "string",
  "reason" : "string",
  "dataUri" : "string",
  "encryptedAesKey" : "string",
  "uriType" : "string",
  "dataHash" : "string",
  "lockProof" : "string",
  "vcIndex" : "string"
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "stringst",
  "errorMsg" : "string"
}
```

状态码

状态码	描述
200	VCOOrder Information
500	失败响应

错误码

请参见[错误码](#)。

5.4.3.6 查询待处理的申请订单

功能介绍

根据服务标识符，查询待处理的凭证申请订单，仅有服务提供者有权限

URI

POST /v1/identity/query-vc-orders

请求参数

表 5-149 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
serviceID	否	String	服务的标识符

响应参数

状态码： 200

表 5-150 响应 Body 参数

参数	参数类型	描述
items	Array of VCOder objects	列表

表 5-151 VCOder

参数	参数类型	描述
applier	String	申请者身份标识
orderSeq	String	订单序列号
applyTime	String	申请时间
price	String	价格
status	String	状态
service	String	服务
reason	String	理由信息
dataUri	String	凭证数据URI

参数	参数类型	描述
encryptedAesKey	String	加密的AES密钥
uriType	String	URI类型
dataHash	String	数据的hash
lockProof	String	锁定证明
vclIndex	String	凭证索引

状态码： 500

表 5-152 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度： 8 最大长度： 36
errorMsg	String	错误描述 最小长度： 2 最大长度： 512

请求示例

```
{
  "orgID" : "4f1439758ebb41f7411b5f684b67713c08b89198",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "orderIndex" : 1
}
```

响应示例

状态码： 200

UntreatedVCOOrder Information List

```
{
  "items" : [ {
    "applier" : "string",
    "orderSeq" : "string",
    "applyTime" : "string",
    "price" : "string",
    "status" : "string",
    "service" : "string",
    "reason" : "string",
    "dataUri" : "string",
  }
]
```

```
"encryptedAesKey" : "string",
"uriType" : "string",
"dataHash" : "string",
"lockProof" : "string",
"vcIndex" : "string"
}]
}
```

状态码： 500

失败响应

```
{
"errorCode" : "stringst",
"errorMsg" : "string"
}
```

状态码

状态码	描述
200	UntreatedVCOrder Information List
500	失败响应

错误码

请参见[错误码](#)。

5.4.3.7 签发可验证凭证

功能介绍

签发可验证凭证,签发时间默认为当前时间。

URI

POST /v1/identity/issue-vc

请求参数

表 5-153 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n

参数	是否必选	参数类型	描述
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
credentialInfo	是	CredentialSubjectInfo object	凭证主题信息

表 5-154 CredentialSubjectInfo

参数	是否必选	参数类型	描述
applier	是	String	申请者身份标识
issuer	是	String	签发者身份标识
sequence	是	String	凭证的序列号
schemaIndex	是	String	凭证模板索引
expirationDate	否	String	过期时间
data	是	String	凭证数据明文

响应参数

状态码： 200

表 5-155 响应 Body 参数

参数	参数类型	描述
vclIndex	String	凭证索引

状态码： 500

表 5-156 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度： 8 最大长度： 36

参数	参数类型	描述
errorMsg	String	错误描述 最小长度：2 最大长度：512

请求示例

```
{
  "orgID" : "org1",
  "channelID" : "mychannel",
  "cryptoMethod" : "this is a demo",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "applyer" : "did:example:ebfeb1f712ebc6f1c276e12ec21",
  "issuer" : "did:example:fdsafre767f8a3hr773j4h1jkh",
  "sequence" : "10025469331",
  "schemaIndex" : "did:example:ebfeb1f712ebc6f1c276e12ec21_IDCard",
  "data" : "{\"name\": \"xm\", \"age\": 18}"
}
```

响应示例

状态码：200

VCResponseParams Information

```
{
  "vcIndex" : "string"
}
```

状态码：500

失败响应

```
{
  "errorCode" : "stringst",
  "errorMsg" : "string"
}
```

状态码

状态码	描述
200	VCResponseParams Information
500	失败响应

错误码

请参见[错误码](#)。

5.4.3.8 根据索引查询可验证凭证

功能介绍

根据索引查询可验证凭证

URI

POST /v1/identity/query-vc

请求参数

表 5-157 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
vcIndex	是	String	凭证索引

响应参数

状态码： 200

表 5-158 响应 Body 参数

参数	参数类型	描述
context	String	内容
sequence	String	颁发机构对应凭证的序列号
type	Array of strings	可验证凭证类型
issuer	String	签发者身份标识
issuanceDate	String	签发日期
expirationDate	String	凭证有效期

参数	参数类型	描述
credentialSubject	CredentialSubject object	凭证主题
revocation	Revocation object	撤销

表 5-159 CredentialSubject

参数	参数类型	描述
owner	String	申请者的身份标识
type	String	凭证类型
schemaID	String	schema ID
dataURI	String	数据URI
encryptedAesKey	String	加密对称密钥
uriType	String	数据索引类型
dataHash	String	数据hash值

表 5-160 Revocation

参数	参数类型	描述
id	String	撤销API或者撤销列表的url
type	String	撤销类型

状态码： 500

表 5-161 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度： 8 最大长度： 36
errorMsg	String	错误描述 最小长度： 2 最大长度： 512

请求示例

```
{
  "orgID" : "4f1439758ebb41f7411b5f684b67713c08b89198",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "vcIndex" : 0
}
```

响应示例

状态码： 200

VerifiableCredential Information

```
{
  "context" : "https://www.w3.org/2018/credentials/v1",
  "sequence" : "x00123456",
  "type" : [ "VerifiableCredential", "AlumniCredential" ],
  "issuer" : "https://example.edu/issuers/565049",
  "issuanceDate" : "1606720551",
  "expirationDate" : "1606720551",
  "credentialSubject" : {
    "owner" : "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "type" : "professional",
    "schemaID" : "did:example:ebfeb1f712ebc6f1c276e12ec21_IDCard",
    "dataURI" : "string",
    "encryptedAeskey" : "string",
    "uriType" : "index",
    "dataHash" : "e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b85"
  },
  "revocation" : {
    "id" : "string",
    "type" : "string"
  }
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "stringst",
  "errorMsg" : "string"
}
```

状态码

状态码	描述
200	VerifiableCredential Information
500	失败响应

错误码

请参见[错误码](#)。

5.4.3.9 验证凭证

功能介绍

验证可验证凭证是否存在以及合法性。

URI

POST /v1/identity/verify-vc

请求参数

表 5-162 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	用户证书，每行末尾均需要增加显式换行符\n
sk	是	String	用户私钥，每行末尾均需要增加显式换行符\n
timestamp	是	String	时间戳
vcIndex	是	String	凭证索引
owner	是	String	凭证所有者身份标识

响应参数

状态码： 200

表 5-163 响应 Body 参数

参数	参数类型	描述
isOwned	Boolean	是否拥有

状态码： 400

表 5-164 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度：8 最大长度：36
errorMsg	String	错误描述 最小长度：2 最大长度：512

状态码：500

表 5-165 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码 最小长度：8 最大长度：36
errorMsg	String	错误描述 最小长度：2 最大长度：512

请求示例

```
{
  "orgID" : "4f1439758ebb41f7411b5f684b67713c08b89198",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "vcIndex" : 1,
  "owner" : "did:example:ebfeb1f712ebc6f1c276e12ec21"
}
```

响应示例

状态码：200

VCVerifyResponseParams Information

```
{
  "isOwned" : true
}
```

状态码：400

失败响应

```
{
  "errorCode" : "BCS.4002030",
}
```

```
{
  "errorMsg": "Owner(bol) does not have credential ..."
}
```

状态码： 500

失败响应

```
{
  "errorCode": "BCS.5002014",
  "errorMsg": "Internal Server Error"
}
```

状态码

状态码	描述
200	VCVerifyResponseParams Information
400	失败响应
500	失败响应

错误码

请参见[错误码](#)。

5.5 可信数据交换（公测）

5.5.1 概述

在商业实践中，数据是重要的生产要素。基于区块链的可信数据交换，实现了分布式场景中业务数据的隐私保护与可信共享，有效打破“数据孤岛”，最大化数据价值。可信数据共享中间件集成在Rest API插件中，可快速插拔，支持弹性伸缩。用户可通过RESTful API的方式访问区块链系统，快速集成，实现数据的发布、授权、分享、加密、解密、细粒度访问控制等能力。

功能介绍

- 可信数据交换基于分布式身份实现，参与数据交换的用户需要注册分布式身份，详细方法请参见[分布式身份（公测）](#)。
- 可信数据涉及两个主要数据结构：数据集和数据订单。数据集包括数据描述信息、访问控制信息（属性加密中的策略）等。数据订单包括数据申请、审核信息等。
- 可信数据交换支持三种模式：申请-授权、主动分享和细粒度访问控制。详细请参见[模式介绍](#)。

📖 说明

可信交换中的数据加密后支持多种存储服务，用户可以根据业务需要自己选择。调用者负责将密文数据存储到公开可访问的存储设备中。

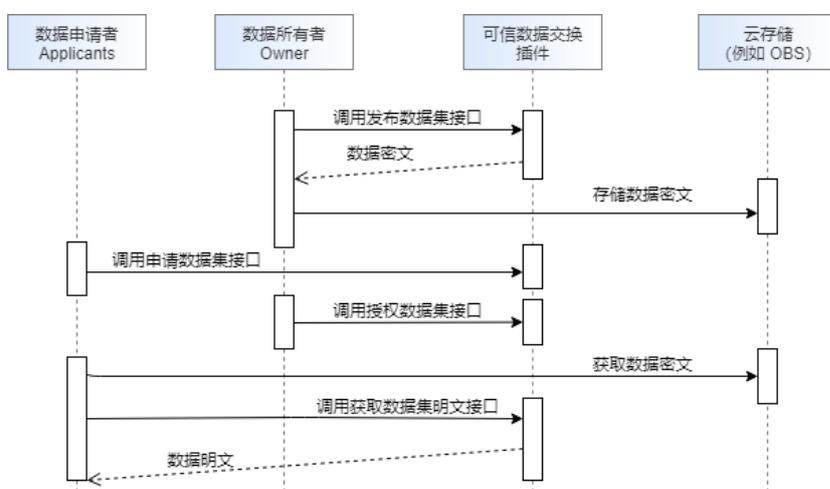
角色定义

数据所有者Owner和数据申请者Applicants，每一个用户既可以是数据所有者角色，也可以是数据申请者角色。

模式介绍

- 申请-授权模式，授权流程图请参见图5-5。
 - 数据所有者通过“发布数据集”接口完成用户明文数据的加密和数据描述等信息的注册发布。
 - 数据申请者可通过“申请数据集”接口调用链代码触发申请-授权流程。
 - 数据所有者可根据申请信息和申请者的did、vc等信息决定授权或者拒绝。

图 5-5 申请-授权模式使用流程

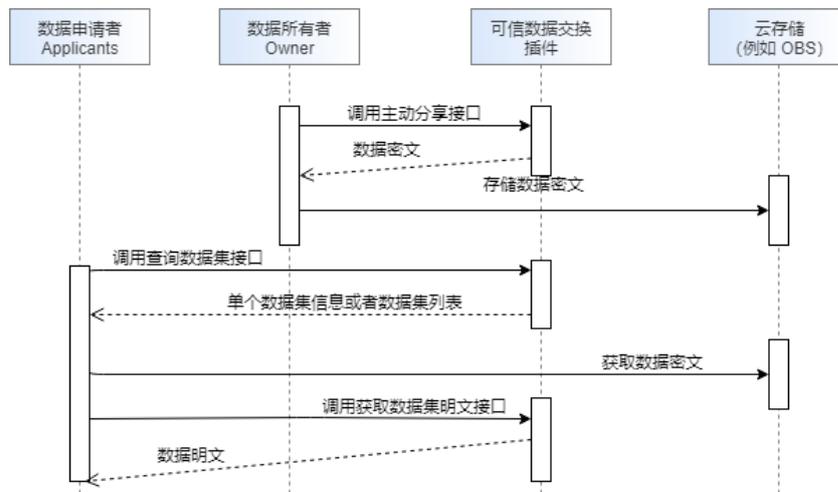


- 主动分享模式，使用流程请参见图5-6。

“主动分享数据集”接口相当于“发布数据集”接口和“授权数据集”接口的组合。数据所有者将数据集发布到区块链，同时授权某申请者解密数据权限，被授权者可以直接解密数据集。此时，其他参与者均可以通过“查询指定数据集”和“查询数据集列表”接口获得数据相关描述信息，并通过申请-授权模式获取数据解密权限。

接口使用方法请参考数据集管理和数据订单管理。

图 5-6 主动分享模式使用流程



- 细粒度访问控制模式，使用流程请参见图5-7。

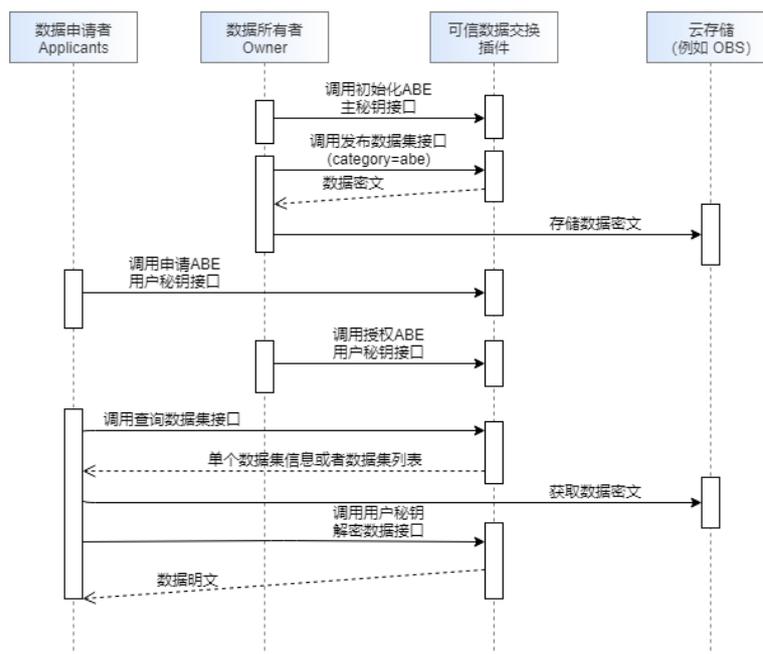
说明

基于属性加密 (Attribute-Based Encryption, ABE) 细粒度控制的数据交换模式，通过为每个数据集配置合理的、自定义的共享策略，实现对数据申请者属性级别的访问控制，即拥有某些属性组合的申请者可以访问密文。

细粒度访问控制模式采用 CP-ABE (基于密文策略的属性加密, Ciphertext-Policy ABE) 方式实现。将策略嵌入到密文中，属性嵌入到用户密钥中。

- 每个数据所有者都只需初始化一次自己的主公钥和私钥。
- 数据申请者需要使用某数据所有者数据时，需要向其申请用户密钥，当属性没有变化的情况下只需申请一次。
- 当拥有用户密钥且属性满足密文访问策略时，数据申请者可以异步的，随时解密数据所有者发布的所有相应数据。
- 接口使用方法请参考属性加密的密钥管理。

图 5-7 细粒度访问控制模式使用流程



📖 说明

基本概念介绍：

- 属性 (Attribute)：描述一个实体的性质与实体之间关系的统称。
- 策略 (Policy)：策略是属性集合与逻辑关系的组合，由数据所有者制定，将嵌入到数据密文中。例如 “age>26 && gender=man” 代表策略要求年龄大于26岁且性别是男性。
- ABE主密钥包含主公钥 (Master Public Key, MPK) 和主私钥 (Master Secret Key, MSK)：数据所有者所有，用于加密密文和生成用户 (数据申请者) 密钥。
- 用户密钥 (User Key)：数据申请者通过提交属性列表，向数据所有者申请获得。密钥中包含用户的属性信息，用于解密密文。

5.5.2 数据集管理

5.5.2.1 发布数据集

功能介绍

发布数据集。不支持水印功能

URI

POST /v1/datashare/dataset

请求参数

表 5-166 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
provider	是	String	数据发布者身份标识
providerName	否	String	数据发布者名称
productName	是	String	产品名称
productID	是	String	产品ID
sampleUrl	是	String	样例数据存放地址
sampleSize	否	String	样例数据大小

参数	是否必选	参数类型	描述
sampleType	否	String	样例数据类型
sampleName	否	String	样例数据名称
fileType	否	String	数据文件类型
dataUrl	是	String	数据存放地址
dataHash	否	String	数据哈希
dataSize	否	String	数据大小
dataName	否	String	数据名称
description	否	String	数据集描述信息，在使用abe加密时，需要对加密策略进行详细描述
plainData	是	String	base64编码的明文数据
category	否	String	加密类型，abe/symmetric，默认为symmetric，如果使用abe加密，则policy必填
watermarkType	否	String	水印类型，visible明水印，blind暗水印，嵌入水印时，必须填写；不填写时默认不嵌入水印。嵌入的水印内容为：发布人 did_productID。
file	否	File	加水印的文件，当对文件加水印时，plainData无效果
policy	否	policy object	abe策略

表 5-167 policy

参数	是否必选	参数类型	描述
Threshold	是	Integer	策略需要满足的属性阈值
Children	是	Array of policy-children objects	子属性列表

表 5-168 policy-children

参数	是否必选	参数类型	描述
name	是	String	属性名

参数	是否必选	参数类型	描述
type	是	String	属性类型 (plain, comparable 和 policy)
value	是	policy-children-comparablevalue object	属性值, 根据type确定具体参数。当type为plain时, value为string类型属性值。当type为policy时, value为子policy。当type为comparable时, value包含了op、value和maxValue三部分。

表 5-169 policy-children-comparablevalue

参数	是否必选	参数类型	描述
op	否	String	type为comparable时填写, 比较类型符号 (>,<或=)
value	是	String	比较类型属性值, 必须为整数
maxValue	否	String	type为comparable时填写, 比较类型中value上限值

响应参数

状态码: 200

表 5-170 响应 Body 参数

参数	参数类型	描述
provider	String	数据集提供者身份标识
providerName	String	数据集提供者名称
productName	String	数据集产品名称
productID	String	数据集产品id
sampleUrl	String	样例数据url
sampleSize	String	样例数据大小
sampleType	String	样例数据类型
sampleName	String	样例数据名称
fileType	String	文件类型
dataUrl	String	数据url

参数	参数类型	描述
dataHash	String	数据哈希值
dataSize	String	数据大小
dataName	String	数据名称
description	String	数据描述
price	String	数据价格
encryptedAesKey	String	密钥
status	String	状态
publishTime	String	数据发布时间
dataFiles	Array of DataFile objects	数据文件列表
sampleFiles	Array of DataFile objects	样例文件列表
category	String	加密类型
encryptData	String	加密后的数据

表 5-171 DataFile

参数	参数类型	描述
fileType	String	文件类型
dataUrl	String	数据url
dataHash	String	数据哈希
dataSize	String	数据大小
dataName	String	数据名称

状态码： 500

表 5-172 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

abe示例策略（policy）。该示例策略中，解密者需要符合属性att1=="hello"或符合子策略才能解密；解密者需要属性att3的值大于16才能符合子策略。

```
{
  "orgID": "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID": "mychannel",
  "cryptoMethod": "SW",
  "cert": "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk": "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp": "2020-10-27T17:28:16+08:00",
  "provider": "did:example:DHkJyD5wZwya6sd6BNBnG",
  "providerName": "huawei",
  "productName": "prodname",
  "productID": "product2",
  "sampleUrl": "http://hwcloud.com/sample.com/prodname2",
  "sampleSize": "10KB",
  "sampleType": "csv",
  "sampleName": "data_sub1",
  "fileType": "csv",
  "dataUrl": "http://hwcloud.com/prodname2",
  "dataHash": "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
  "dataSize": "100MB",
  "dataName": "mydata1",
  "description": "this is my second prod",
  "plainData": "base64 encoding string",
  "category": "abe",
  "watermarkType": "string",
  "file": "string",
  "policy": "{ \"threshold\": 1, \"children\": [ { \"name\": \"att1\", \"type\": \"plain\", \"value\": \"hello\" },
  { \"name\": \"policy\", \"type\": \"policy\", \"value\": { \"Threshold\": 1, \"Children\": [ { \"name\": \"att3\",
  \"type\": \"comparable\", \"value\": { \"op\": \">\", \"value\": \"16\", \"maxValue\": \"10000\" } } ] } } ] } }
```

响应示例

状态码： 200

数据集信息

```
{
  "provider": "did:example:DHkJyD5wZwya6sd6BNBnG",
  "providerName": "aws",
  "productName": "prodname2",
  "productID": "product2",
  "sampleUrl": "http://hwcloud.com/sample.com/prodname2",
  "sampleSize": "10KB",
  "sampleType": "csv",
  "sampleName": "data_sub1",
  "fileType": "csv",
  "dataUrl": "http://hwcloud.com/prodname2",
  "dataHash": "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
  "dataSize": "100MB",
  "dataName": "mydata",
  "description": "this is second prod",
  "price": "0",
  "encryptedAesKey": "BA4Ub3t3IskN8uKcEMa+4cbtsDS8OzF4V/qqb4OcPMeMvp7lL+HClzAbL6lPnhbDg/AnrStBlf0qFzRj+qvk6ZH0c7wP0aS48fSoNtecG79aFpFx0dg7rFdVYXWWzgey103eD3gFdXlQ/ovpxKJG5ALK39OCazUqDrawZHSdGyllw0hGh88Q+GVORVSp+6V5Ag==",
  "status": "ready",
  "publishTime": "1607157244",
  "dataFiles": [ {
    "fileType": "csv",
    "dataUrl": "http://hwcloud.com/prodname2",
    "dataHash": "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
    "dataSize": "100MB",
    "dataName": "mydata"
  } ]
}
```

```

    }],
    "sampleFiles": [ {
      "fileType": "csv",
      "dataUrl": "http://hwcloud.com/prodname2",
      "dataHash": "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
      "dataSize": "100MB",
      "dataName": "mydata"
    } ],
    "category": "string",
    "encryptData": "string"
  }
}

```

状态码： 500

失败响应

```

{
  "errorCode": "BCS.5002046",
  "errorMsg": "Incorrect number of arguments"
}

```

状态码

状态码	描述
200	数据集信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.2.2 删除数据集

功能介绍

删除数据集

URI

DELETE /v1/datashare/dataset

请求参数

表 5-173 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书

参数	是否必选	参数类型	描述
sk	是	String	私钥
timestamp	是	String	时间戳
provider	是	String	数据集提供者身份标识
productID	是	String	数据产品id

响应参数

状态码： 200

表 5-174 响应 Body 参数

参数	参数类型	描述
result	String	操作结果

状态码： 500

表 5-175 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "provider" : "did:example:DHkJyD5wZwya6sd6BNBnG",
  "productID" : "product1"
}
```

响应示例

状态码： 200

操作结果

```
{
  "result" : "success"
}
```

状态码： 500

失败响应

```
{
  "errorCode": "BCS.5002046",
  "errorMsg": "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	操作结果
500	失败响应

错误码

请参见[错误码](#)。

5.5.2.3 关闭数据集

功能介绍

关闭数据集

URI

PUT /v1/datashare/dataset

请求参数

表 5-176 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
provider	是	String	数据集提供者身份标识
productID	是	String	数据产品id

响应参数

状态码： 200

表 5-177 响应 Body 参数

参数	参数类型	描述
result	String	操作结果

状态码： 500

表 5-178 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "provider" : "did:example:DHkJyD5wZwya6sd6BNbNg",
  "productID" : "product1"
}
```

响应示例

状态码： 200

操作结果

```
{
  "result" : "success"
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "BCS.5002046",
  "errorMsg" : "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	操作结果
500	失败响应

错误码

请参见[错误码](#)。

5.5.2.4 查询指定数据集

功能介绍

查询指定数据集，根据数据集发布者身份标识和数据产品id信息查询。

URI

POST /v1/datashare/query-dataset

请求参数

表 5-179 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
provider	是	String	数据集发布者身份标识
productID	是	String	数据集产品id

响应参数

状态码： 200

表 5-180 响应 Body 参数

参数	参数类型	描述
provider	String	数据集提供者身份标识
providerName	String	数据集提供者名称
productName	String	数据集产品名称
productID	String	数据集产品id
sampleUrl	String	样例数据url
sampleSize	String	样例数据大小
sampleType	String	样例数据类型
sampleName	String	样例数据名称
fileType	String	文件类型
dataUrl	String	数据url
dataHash	String	数据哈希值
dataSize	String	数据大小
dataName	String	数据名称
description	String	数据描述
price	String	数据价格
encryptedAesKey	String	密钥
status	String	状态
publishTime	String	数据发布时间
dataFiles	Array of DataFile objects	数据文件列表
sampleFiles	Array of DataFile objects	样例文件列表
category	String	加密类型

表 5-181 DataFile

参数	参数类型	描述
fileType	String	文件类型
dataUrl	String	数据url

参数	参数类型	描述
dataHash	String	数据哈希
dataSize	String	数据大小
dataName	String	数据名称

状态码： 500

表 5-182 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "provider" : "did:example:DHkJyD5wZwya6sd6BNBnG",
  "productID" : "product2"
}
```

响应示例

状态码： 200

操作结果

```
{
  "provider" : "did:example:DHkJyD5wZwya6sd6BNBnG",
  "providerName" : "aws",
  "productName" : "prodname2",
  "productID" : "product2",
  "sampleUrl" : "http://hwcloud.com/sample.com/prodname2",
  "sampleSize" : "10KB",
  "sampleType" : "csv",
  "sampleName" : "data_sub1",
  "fileType" : "csv",
  "dataUrl" : "http://hwcloud.com/prodname2",
  "dataHash" : "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
  "dataSize" : "100MB",
  "dataName" : "mydata",
  "description" : "this is second prod",
  "price" : "0",
  "encryptedAesKey" : "BA4Ub3t3IskN8uKcEMa+4cbtsDS8OzF4V/qqb4OcPMeMvp7IL+HClzAbL6lPnhbDg/AnrStBlf0qFzRj+qvk6ZH0c7wP0aS48fSoNtecG79aFpFxDg7rFdVYXWWzgeyl03eD3gFdXlQ/ovpxKJG5ALK39OCazUqDrawZHSDGyllw0hGh88Q+GVORVSp+6V5Ag==",
  "status" : "ready",
  "publishTime" : "1607157244",
  "dataFiles" : [ {
```

```

"fileType" : "csv",
"dataUrl" : "http://hwcloud.com/prodname2",
"dataHash" : "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
"dataSize" : "100MB",
"dataName" : "mydata"
}],
"sampleFiles" : [ {
"fileType" : "csv",
"dataUrl" : "http://hwcloud.com/prodname2",
"dataHash" : "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
"dataSize" : "100MB",
"dataName" : "mydata"
}],
"category" : "string"
}
    
```

状态码： 500

失败响应

```

{
"errorCode" : "BCS.5002046",
"errorMsg" : "Incorrect number of arguments"
}
    
```

状态码

状态码	描述
200	操作结果
500	失败响应

错误码

请参见[错误码](#)。

5.5.2.5 查询数据集列表

功能介绍

查询数据集，支持分页和按条件过滤查询

URI

POST /v1/datashare/query-datasets

请求参数

表 5-183 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id

参数	是否必选	参数类型	描述
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
currentPage	否	String	分页参数：当前页码（默认1）
pageSizeNum	否	String	分页参数：每页条数(默认100)
provider	否	String	过滤条件：发布者身份标识
searchText	否	String	过滤条件：匹配关键字（基于数据集中的产品名称、产品描述信息）
status	否	String	过滤条件：数据集状态（ready、closed）

响应参数

状态码： 200

表 5-184 响应 Body 参数

参数	参数类型	描述
items	Array of DatasetResponse objects	列表
pagination	PaginationResponse object	分页信息

表 5-185 DatasetResponse

参数	参数类型	描述
provider	String	数据集提供者身份标识
providerName	String	数据集提供者名称
productName	String	数据集产品名称
productID	String	数据集产品id
sampleUrl	String	样例数据url
sampleSize	String	样例数据大小

参数	参数类型	描述
sampleType	String	样例数据类型
sampleName	String	样例数据名称
fileType	String	文件类型
dataUrl	String	数据url
dataHash	String	数据哈希值
dataSize	String	数据大小
dataName	String	数据名称
description	String	数据描述
price	String	数据价格
encryptedAesKey	String	密钥
status	String	状态
publishTime	String	数据发布时间
dataFiles	Array of DataFile objects	数据文件列表
sampleFiles	Array of DataFile objects	样例文件列表
category	String	加密类型

表 5-186 DataFile

参数	参数类型	描述
fileType	String	文件类型
dataUrl	String	数据url
dataHash	String	数据哈希
dataSize	String	数据大小
dataName	String	数据名称

表 5-187 PaginationResp

参数	参数类型	描述
currentPage	Integer	当前页码
pageSizeNum	Integer	每页条数
totalItems	Integer	总条数

状态码： 500

表 5-188 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "currentPage" : "string",
  "pageSizeNum" : "string",
  "provider" : "string",
  "searchText" : "string",
  "status" : "string"
}
```

响应示例

状态码： 200

数据集分页信息

```
{
  "items" : [ {
    "provider" : "did:example:DHkJyD5wZwya6sd6BNbNc",
    "providerName" : "aws",
    "productName" : "prodname2",
    "productID" : "product2",
    "sampleUrl" : "http://hwcloud.com/sample.com/prodname2",
    "sampleSize" : "10KB",
    "sampleType" : "csv",
    "sampleName" : "data_sub1",
    "fileType" : "csv",
    "dataUrl" : "http://hwcloud.com/prodname2",
    "dataHash" : "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
    "dataSize" : "100MB",
    "dataName" : "mydata",
    "description" : "this is second prod",
    "price" : "0",
  }
]
```

```

"encryptedAesKey" : "BA4Ub3t3lSkN8uKcEMa+4cbtsDS8OzF4V/qqb4OcpMeMvp7IL+HClzAbL6lPnhbDg/
AnrStBlf0qFzRj+qvk6ZH0c7wP0a548fSoNtecG79aFpFxDdg7rFdVYXWWzgeyl03eD3gFdXlQ/
ovpxKJG5ALK39OCazUqDrawZHSDGyllw0hGh88Q+GVORVSp+6V5Ag==",
"status" : "ready",
"publishTime" : "1607157244",
"dataFiles" : [ {
"fileType" : "csv",
"dataUrl" : "http://hwcloud.com/prodname2",
"dataHash" : "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
"dataSize" : "100MB",
"dataName" : "mydata"
}],
"sampleFiles" : [ {
"fileType" : "csv",
"dataUrl" : "http://hwcloud.com/prodname2",
"dataHash" : "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
"dataSize" : "100MB",
"dataName" : "mydata"
}],
"category" : "string"
}],
"pagination" : {
"currentPage" : 1,
"pageSizeNum" : 100,
"totalItems" : 10
}
}

```

状态码： 500

失败响应

```

{
"errorCode" : "BCS.5002046",
"errorMsg" : "Incorrect number of arguments"
}

```

状态码

状态码	描述
200	数据集分页信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.2.6 主动分享数据集

功能介绍

发布数据集并创建已授权的订单。不支持水印功能

URI

POST /v1/datashare/dataset/share

请求参数

表 5-189 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
provider	是	String	数据发布者身份标识
providerName	否	String	数据发布者名称
productName	是	String	产品名称
productID	是	String	产品ID
sampleUrl	是	String	样例数据存放地址
sampleSize	否	String	样例数据大小
sampleType	否	String	样例数据类型
sampleName	否	String	样例数据名称
fileType	否	String	数据文件类型
dataUrl	是	String	数据存放地址
dataHash	否	String	数据哈希
dataSize	否	String	数据大小
dataName	否	String	数据名称
description	否	String	数据集描述信息，在使用abe加密时，需要对加密策略进行详细描述
plainData	是	String	base64编码的明文数据
consumer	是	String	订单申请者身份标识
orderSeq	否	String	订单序列号，为空则随机生成
watermarkType	否	String	水印类型，visible明水印，blind暗水印，嵌入水印时，必须填写；不填写时默认不嵌入水印。嵌入的水印内容为：发布人 did_productID。

参数	是否必选	参数类型	描述
file	否	String	加水印的文件，当对文件加水印时，plainData无效果
productIDKey words	否	Array of productIDKey words Json objects	产品ID包含的索引关键字，用于按条件查询订单时使用，需要为设定的json格式
onChainStore	否	String	是否在链上存储加密后的数据，可设置为“true”或“false”，默认为“false”。当为“true”时，sampleUrl和dataUrl可以自动填充，不需要填写。水印功能暂不支持链上存储。
consumerName	否	String	使用者名称
creatorDID	否	String	数据集分享流程的创建者DID，action为“new”时，不需要填入
processID	否	String	数据集分享流程的流程ID，action为“new”时，不需要输入
stageName	否	String	本次数据集分享阶段的阶段名称
action	否	String	流程动作，新建：“new”、追加：“append”、不涉及：“”，该字段不输入时视为不涉及
category	否	String	加密类型，symmetric/abe，默认为symmetric，如果使用abe加密，则policy必填
policy	否	policy object	abe策略

表 5-190 productIDKeywordsJson

参数	是否必选	参数类型	描述
value	否	String	关键字值

表 5-191 policy

参数	是否必选	参数类型	描述
Threshold	是	Integer	策略需要满足的属性阈值

参数	是否必选	参数类型	描述
Children	是	Array of policy-children objects	子属性列表

表 5-192 policy-children

参数	是否必选	参数类型	描述
name	是	String	属性名
type	是	String	属性类型 (plain, comparable 和policy)
value	是	policy-children-comparablevalue object	属性值，根据type确定具体参数。当type为plain时，value为string类型属性值。当type为policy时，value为子policy。当type为comparable时，value包含了op、value和maxValue三部分。

表 5-193 policy-children-comparablevalue

参数	是否必选	参数类型	描述
op	否	String	type为comparable时填写，比较类型符号 (>,<或=)
value	是	String	比较类型属性值，必须为整数
maxValue	否	String	type为comparable时填写，比较类型中value上限值

响应参数

状态码： 200

表 5-194 响应 Body 参数

参数	参数类型	描述
consumer	String	订单消费者身份标识
consumerName	String	订单消费者名称

参数	参数类型	描述
orderSeq	String	订单序列号
provider	String	订单提供者身份标识
providerName	String	订单提供者名称
productID	String	数据集产品id
productName	String	数据集产品名称
price	String	订单价钱
applyTime	String	订单申请时间
encryptedAesKey	String	密钥
status	String	订单状态
reason	String	订单申请原因
lockProof	String	订单锁定证明
creatorDID	String	流程创建者DID，如果没有加入任何流程，为“ ”
processID	String	当前订单所属流程ID，如果没有加入任何流程，为“ ”
encryptData	String	base64编码的数据密文

状态码： 500

表 5-195 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "provider" : "did:example:DHkJyD5wZwya6sd6BNBnG",
  "providerName" : "huawei",
  "productName" : "prodname",
  "productID" : "product2",
  "sampleUrl" : "http://hwcloud.com/sample.com/prodname2",
}
```

```

"sampleSize" : "10KB",
"sampleType" : "csv",
"sampleName" : "data_sub1",
"fileType" : "csv",
"dataUrl" : "http://hwcloud.com/prodname2",
"dataHash" : "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
"dataSize" : "100MB",
"dataName" : "mydata1",
"description" : "this is my second prod",
"plainData" : "base64 encoding string",
"consumer" : "did:example:3TMWx8owKHARgNwbj4ywmG",
"orderSeq" : "1",
"watermarkType" : "string",
"file" : "string",
"productIDkeywords" : "[{"value":"taiyuan"},{"value":"renmin_hospital"},{"value":"medicine"}]",
"onChainStore" : "string",
"consumerName" : "string"
}

```

响应示例

状态码： 200

订单信息

```

{
  "consumer" : "did:example:3TMWx8owKHARgNwbj4ywmG",
  "consumerName" : "Tyler",
  "orderSeq" : "1",
  "provider" : "did:example:DHkJyD5wZwya6sd6BNBnG",
  "providerName" : "hw",
  "productID" : "product1",
  "productName" : "prodname1",
  "price" : "0",
  "applyTime" : "1607332359",
  "encryptedAesKey" : "BNGhPwjaTgpM+V7czzw1i4mH21KKN+XLKXHLqVsRifybUCncqZNfomkRfzX4WEHj
+oty1X9oCd4h6xMnRvs8BWE5Tvg6BJ6QTW/km9EO/FSYqzJf2GqQzAleAcLJrTBZ3LRbPaF87CgJ114ae7R
+VK9VvfXQ8exuH2KMRD305dXieGpM4VPVv9u1BbL15Jpd/g==",
  "status" : "ready",
  "reason" : "I want product1",
  "lockProof" : "",
  "encryptData" : "base64 encoding string"
}

```

状态码： 500

失败响应

```

{
  "errorCode" : "BCS.5002046",
  "errorMsg" : "Incorrect number of arguments"
}

```

状态码

状态码	描述
200	订单信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.2.7 获取数据解密后的明文

功能介绍

获取数据解密后的明文。不支持水印功能

URI

POST /v1/datashare/dataset/query-plaintext

请求参数

表 5-196 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
consumer	是	String	订单申请者身份标识
orderSeq	是	String	订单序列号
encryptData	否	String	数据密文，当onChainStore设置为“true”时，可不输入
watermarkType	否	String	水印类型，visible明水印，blind暗水印，嵌入水印时，必须填写本字段。如果发布或主动分享数据集的时候嵌入的是暗水印，则无法再次嵌入暗水印。嵌入的水印内容为：使用人 did_orderID。
onChainStore	否	String	数据密文是否在链上存储，可设置为“true”或“false”，默认为“false”。如果设置为“true”，则不需要输入 encryptData，可自动在链上获取数据密文

响应参数

状态码： 200

表 5-197 响应 Body 参数

参数	参数类型	描述
provider	String	订单提供者身份标识
productID	String	数据集产品id
plaintext	String	base64编码的数据明文

状态码： 500

表 5-198 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "consumer" : "did:example:3TMWx8owKHARgNwbj4ywmG",
  "orderSeq" : "1",
  "encryptData" : "base64 encoding string",
  "watermarkType" : "string",
  "onChainStore" : "string"
}
```

响应示例

状态码： 200

订单信息。

```
{
  "provider" : "did:example:DHkJyD5wZwya6sd6BNBnG",
  "productID" : "product1",
  "plaintext" : "base64 encoding string"
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "BCS.5002046",
```

```
"errorMsg" : "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	订单信息。
500	失败响应

错误码

请参见[错误码](#)。

5.5.2.8 提取文件中的暗水印

功能介绍

提取文件中的暗水印。不支持水印功能

URI

POST /v1/datashare/dataset/watermark/extract

请求参数

表 5-199 FormData 参数

参数	是否必选	参数类型	描述
file	是	File	需要提取暗水印的文档

响应参数

状态码： 200

表 5-200 响应 Body 参数

参数	参数类型	描述
watermark	String	文件中嵌入的暗水印内容，不嵌入暗水印时内容为空。

状态码： 500

表 5-201 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

无

响应示例

状态码： 200

提取暗水印的返回内容

```
{
  "watermark" : "string"
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "BCS.5002046",
  "errorMsg" : "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	提取暗水印的返回内容
500	失败响应

错误码

请参见[错误码](#)。

5.5.2.9 查询指定的数据集分享流程

功能介绍

查询指定的数据集分享流程

URI

POST /v1/datashare/dataset/query-process

请求参数

表 5-202 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
creatorDID	是	String	流程创建者身份标识
processID	是	String	流程ID

响应参数

状态码： 200

表 5-203 响应 Body 参数

参数	参数类型	描述
creatorDID	String	流程创建者身份标识
processID	String	流程ID
stages	Array of StageInProcess objects	流程中的阶段信息

表 5-204 StageInProcess

参数	参数类型	描述
stageName	String	阶段名称
createTime	String	阶段信息上链时间戳
consumer	String	消费者身份标识
orderSeq	String	订单序列号

状态码： 500

表 5-205 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
/v1/datashare/dataset/query-process
{
  "orgID": "{{orgID}}",
  "channelID": "{{channelID}}",
  "cryptoMethod": "{{cryptoMethod}}",
  "cert": "{{cert}}",
  "sk": "{{sk}}",
  "timestamp": "{{timestamp}}",
  "creatorDID": "did:example:8sAvs54tB3NYgMJ4uqbVYj",
  "processID": "a779dd88-f7a3-4ac9-bf1c-a0ed1d827632"
}
```

响应示例

状态码： 200

查询指定的数据集分享流程响应参数

```
{
  "creatorDID": "did:example:YLgvmFcukyigJpRsqRbFMn",
  "processID": "25f46489-29bb-4f58-8f4c-e12da0a4bd66",
  "stages": [ {
    "stageName": "transaction1",
    "createTime": "1640574210",
    "consumer": "did:example:My8PRB5dKDvVBKXT76oJoB",
    "orderSeq": "8zLQUpyswA8kpiHEsAUhZN"
  }, {
    "stageName": "transaction2",
    "createTime": "1640574304",
    "consumer": "did:example:T1kFDUQAqo2z2X7hJWiRtQ",
    "orderSeq": "3oKCPSKLGvaebT6z3a4PVY"
  }, {
    "stageName": "transaction2",
    "createTime": "1640587323",
    "consumer": "did:example:T1kFDUQAqo2z2X7hJWiRtQ",
    "orderSeq": "GPSta7mTScEEaQVRB62wUF"
  }, {
    "stageName": "transaction2",
    "createTime": "1640680918",
    "consumer": "did:example:T1kFDUQAqo2z2X7hJWiRtQ",
    "orderSeq": "8kzahSLi2kBxY8GGBZdLhp"
  }
  ]
}
```

状态码

状态码	描述
200	查询指定的数据集分享流程响应参数
500	失败响应

错误码

请参见[错误码](#)。

5.5.2.10 查询指定流程创建者的所有流程

功能介绍

查询指定流程创建者的所有流程

URI

POST /v1/datashare/dataset/query-processes

请求参数

表 5-206 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
currentPage	否	String	分页参数：当前页码（默认1）
pageSizeNum	否	String	分页参数：每页条数(默认100)
creatorDID	是	String	流程创建者身份标识

响应参数

状态码： 200

表 5-207 响应 Body 参数

参数	参数类型	描述
items	Array of DatasetShareProcessResponseBody objects	数据集分享流程列表
pagination	PaginationResp object	分页信息

表 5-208 DatasetShareProcessResponseBody

参数	参数类型	描述
creatorDID	String	流程创建者身份标识
processID	String	流程ID
stages	Array of StageInProcess objects	流程中的阶段信息

表 5-209 StageInProcess

参数	参数类型	描述
stageName	String	阶段名称
createTime	String	阶段信息上链时间戳
consumer	String	消费者身份标识
orderSeq	String	订单序列号

表 5-210 PaginationResp

参数	参数类型	描述
currentPage	Integer	当前页码
pageSizeNum	Integer	每页条数
totalItems	Integer	总条数

状态码： 500

表 5-211 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
/v1/datashare/dataset/query-processes
{
  "orgID": "{{orgID}}",
  "channelID": "{{channelID}}",
  "cryptoMethod": "{{cryptoMethod}}",
  "cert": "{{cert}}",
  "sk": "{{sk}}",
  "timestamp": "{{timestamp}}",
  "creatorDID": "did:example:8sAvsS4tB3NYgMJ4uqbVYj",
  "currentPage": "1",
  "pageSizeNum": "100"
}
```

响应示例

状态码： 200

查询指定流程创建者的所有流程响应参数

```
{
  "items": [ {
    "creatorDID": "did:example:8sAvsS4tB3NYgMJ4uqbVYj",
    "processID": "442a6b42-82b7-415a-a0e6-deaaee59f582",
    "stages": [ {
      "stageName": "Seconde transaction",
      "createTime": "1639824526",
      "consumer": "did:example:8sAvsS4tB3NYgMJ4uqbVYj",
      "orderSeq": "N6UhsPZ5cQY7NtHsxuFTZ"
    } ]
  }, {
    "creatorDID": "did:example:8sAvsS4tB3NYgMJ4uqbVYj",
    "processID": "a779dd88-f7a3-4ac9-bf1c-a0ed1d827632",
    "stages": [ {
      "stageName": "First transaction",
      "createTime": "1639824239",
      "consumer": "did:example:8sAvsS4tB3NYgMJ4uqbVYj",
      "orderSeq": "FKCx1Cfatj7RRsKMWPQ7wQ"
    }, {
      "stageName": "",
      "createTime": "1639826471",
      "consumer": "did:example:8sAvsS4tB3NYgMJ4uqbVYj",
      "orderSeq": "8PuuWWg521bZDXadzwPdMn"
    }, {
      "stageName": "Seconde transaction",
      "createTime": "1639826898",
      "consumer": "did:example:8sAvsS4tB3NYgMJ4uqbVYj",
      "orderSeq": "Sckh1rQ6aD5SYTYkojn44W"
    } ]
  }, {
    "creatorDID": "did:example:8sAvsS4tB3NYgMJ4uqbVYj",
    "processID": "d15b5213-5cb8-4af4-97dc-143379369f35",
    "stages": [ {
      "stageName": "564econde transaction",
      "createTime": "1639827681",

```

```

"consumer": "did:example:8sAvsS4tB3NYgMJ4uqbVYj",
"orderSeq": "XCXtVZsdKFDLaErzpSZYAN"
  }],
  }],
"pagination": {
  "currentPage": 1,
  "pageSizeNum": 100,
  "totalItems": 3
}
}

```

状态码

状态码	描述
200	查询指定流程创建者的所有流程响应参数
500	失败响应

错误码

请参见[错误码](#)。

5.5.3 数据订单管理

5.5.3.1 申请数据集

功能介绍

申请数据集

URI

POST /v1/datashare/dataset/dataset-order

请求参数

表 5-212 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
consumer	是	String	订单申请者身份标识

参数	是否必选	参数类型	描述
orderSeq	是	String	订单序列号
provider	是	String	数据集发布者身份标识
productID	是	String	数据集产品id
reason	否	String	申请原因
consumerName	否	String	数据集申请者名称

响应参数

状态码： 200

表 5-213 响应 Body 参数

参数	参数类型	描述
consumer	String	订单消费者身份标识
consumerName	String	订单消费者名称
orderSeq	String	订单序列号
provider	String	订单提供者身份标识
providerName	String	订单提供者名称
productID	String	数据集产品id
productName	String	数据集产品名称
price	String	订单价钱
applyTime	String	订单申请时间
encryptedAesKey	String	密钥
status	String	订单状态
reason	String	订单申请原因
lockProof	String	订单锁定证明
creatorDID	String	流程创建者DID，如果没有加入任何流程，为“ ”
processID	String	当前订单所属流程ID，如果没有加入任何流程，为“ ”

状态码： 500

表 5-214 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "consumer" : "did:example:3TMWx8owKHARgNwbj4ywmG",
  "orderSeq" : "1",
  "provider" : "did:example:DHkJyD5wZwya6sd6BNBnG",
  "productID" : "product2",
  "reason" : "apply dataset for AI",
  "consumerName" : "user1"
}
```

响应示例

状态码： 200

订单信息

```
{
  "consumer" : "did:example:3TMWx8owKHARgNwbj4ywmG",
  "consumerName" : "Tyler",
  "orderSeq" : "1",
  "provider" : "did:example:DHkJyD5wZwya6sd6BNBnG",
  "providerName" : "hw",
  "productID" : "product1",
  "productName" : "prodname1",
  "price" : "0",
  "applyTime" : "1607332359",
  "encryptedAesKey" : "BNGhPwjaTgpM+V7czzw1i4mH21KKN+XLKXHLqVsRIfybUCncqZNfomkRfzX4WEHj+oty1X9oCd4h6xMnRvs8BWE5Tvg6BJ6QTW/km9EO/FSYqzJf2GqQzAleAcLJrTBZ3LRbPaF87CgJ114ae7R+VK9VvfXQ8exuH2KMRD305dXieGpM4VPVv9u1BbL15Jpd/g==",
  "status" : "ready",
  "reason" : "I want product1",
  "lockProof" : ""
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "BCS.5002046",
  "errorMsg" : "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	订单信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.3.2 授权数据集

功能介绍

授权数据集

URI

POST /v1/datashare/dataset/authorize-dataset

请求参数

表 5-215 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
consumer	是	String	订单申请者身份标识
orderSeq	是	String	订单序列号

响应参数

状态码： 200

表 5-216 响应 Body 参数

参数	参数类型	描述
consumer	String	订单消费者身份标识
consumerName	String	订单消费者名称
orderSeq	String	订单序列号
provider	String	订单提供者身份标识
providerName	String	订单提供者名称
productID	String	数据集产品id
productName	String	数据集产品名称
price	String	订单价钱
applyTime	String	订单申请时间
encryptedAesKey	String	密钥
status	String	订单状态
reason	String	订单申请原因
lockProof	String	订单锁定证明
creatorDID	String	流程创建者DID，如果没有加入任何流程，为“ ”
processID	String	当前订单所属流程ID，如果没有加入任何流程，为“ ”

状态码： 500

表 5-217 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
}
```

```
{
  "consumer": "did:example:3TMWx8owKHARgNwbj4ywmG",
  "orderSeq": "1"
}
```

响应示例

状态码： 200

订单信息

```
{
  "consumer": "did:example:3TMWx8owKHARgNwbj4ywmG",
  "consumerName": "Tyler",
  "orderSeq": "1",
  "provider": "did:example:DHkJyD5wZwya6sd6BNBnG",
  "providerName": "hw",
  "productID": "product1",
  "productName": "prodname1",
  "price": "0",
  "applyTime": "1607332359",
  "encryptedAesKey": "BNGhPwjaTgpM+V7czzw1i4mH21KKN+XLKXHLqVsRIfybUCncqZNfomkRfzX4WEHj+oty1X9oCd4h6xMnRvs8BWE5Tvg6BJ6QTW/km9EO/FSYqzJf2GqQzAleAcLJrTBZ3LRbPaF87CgJ114ae7R+VK9VvfXQ8exuH2KMRD305dXieGpM4VPVv9u1BbL15Jpd/g==",
  "status": "ready",
  "reason": "I want product1",
  "lockProof": ""
}
```

状态码： 500

失败响应

```
{
  "errorCode": "BCS.5002046",
  "errorMsg": "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	订单信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.3.3 修改订单状态

功能介绍

修改订单状态

URI

PUT /v1/datashare/dataset/order

请求参数

表 5-218 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
consumer	是	String	订单申请者身份标识
orderSeq	是	String	订单序列号
orderStatus	是	Integer	设置订单状态（0完成，1失败，2取消）
reason	否	String	原因

响应参数

状态码： 200

表 5-219 响应 Body 参数

参数	参数类型	描述
result	String	操作结果

状态码： 500

表 5-220 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
```

```
"cryptoMethod" : "SW",
"cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
"sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
"timestamp" : "2020-10-27T17:28:16+08:00",
"consumer" : "did:example:3TMWx8owKHARgNwbj4ywmG",
"orderSeq" : "1",
"orderStatus" : 0,
"reason" : "string"
}
```

响应示例

状态码： 200

操作结果

```
{
  "result" : "success"
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "BCS.5002046",
  "errorMsg" : "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	操作结果
500	失败响应

错误码

请参见[错误码](#)。

5.5.3.4 删除订单

功能介绍

删除订单

URI

DELETE /v1/datashare/dataset/order

请求参数

表 5-221 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
consumer	是	String	订单申请者身份标识
orderSeq	是	String	订单序列号

响应参数

状态码： 200

表 5-222 响应 Body 参数

参数	参数类型	描述
result	String	操作结果

状态码： 500

表 5-223 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "consumer" : "did:example:3TMWx8owKHARgNwbj4ywmG",
}
```

```
"orderSeq" : "1"
}
```

响应示例

状态码： 200

操作结果

```
{
  "result" : "success"
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "BCS.5002046",
  "errorMsg" : "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	操作结果
500	失败响应

错误码

请参见[错误码](#)。

5.5.3.5 查询指定订单

功能介绍

查询指定订单，根据订单消费者did和订单序列号查询。

URI

POST /v1/datashare/dataset/query-order

请求参数

表 5-224 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW

参数	是否必选	参数类型	描述
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
consumer	是	String	订单申请者身份标识
orderSeq	是	String	订单序列号

响应参数

状态码： 200

表 5-225 响应 Body 参数

参数	参数类型	描述
consumer	String	订单消费者身份标识
consumerName	String	订单消费者名称
orderSeq	String	订单序列号
provider	String	订单提供者身份标识
providerName	String	订单提供者名称
productID	String	数据集产品id
productName	String	数据集产品名称
price	String	订单价钱
applyTime	String	订单申请时间
encryptedAesKey	String	密钥
status	String	订单状态
reason	String	订单申请原因
lockProof	String	订单锁定证明
creatorDID	String	流程创建者DID，如果没有加入任何流程，为“ ”
processID	String	当前订单所属流程ID，如果没有加入任何流程，为“ ”

状态码： 500

表 5-226 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID": "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID": "mychannel",
  "cryptoMethod": "SW",
  "cert": "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk": "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp": "2020-10-27T17:28:16+08:00",
  "consumer": "did:example:3TMWx8owKHARgNwbj4ywmG",
  "orderSeq": "1"
}
```

响应示例

状态码： 200

订单信息

```
{
  "consumer": "did:example:3TMWx8owKHARgNwbj4ywmG",
  "consumerName": "Tyler",
  "orderSeq": "1",
  "provider": "did:example:DHkJyD5wZwya6sd6BNBnG",
  "providerName": "hw",
  "productID": "product1",
  "productName": "prodname1",
  "price": "0",
  "applyTime": "1607332359",
  "encryptedAesKey": "BNGhPwjaTgpM+V7czzw1i4mH21KKN+XLKXHLqVsRifybUCncqZNfomkRfzX4WEHj+oty1X9oCd4h6xMnRvs8BWE5Tvg6BJ6QTW/km9EO/FSYqzJf2GqQzAleAcLjrTBZ3LRbPaF87CgJ114ae7R+VK9VvfXQ8exuH2KMRD305dXieGpM4VPVv9u1BbL15Jpd/g==",
  "status": "ready",
  "reason": "I want product1",
  "lockProof": ""
}
```

状态码： 500

失败响应

```
{
  "errorCode": "BCS.5002046",
  "errorMsg": "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	订单信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.3.6 查询订单列表

功能介绍

查询订单，支持分页和按条件过滤查询。

URI

POST /v1/datashare/dataset/query-orders

请求参数

表 5-227 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法，目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
currentPage	否	String	分页参数：当前页码（默认1）
pageSizeNum	否	String	分页参数：每页条数(默认100)
provider	否	String	过滤条件：订单中数据集发布者身份标识
searchText	否	String	过滤条件：匹配关键字（订单的产品名称）
status	否	String	过滤条件：订单状态（ready、finished、failed、canceled）
consumer	否	String	过滤条件：订单消费者身份标识

响应参数

状态码： 200

表 5-228 响应 Body 参数

参数	参数类型	描述
items	Array of DataOrderResponse objects	列表
pagination	PaginationResponse object	分页信息

表 5-229 DataOrderResponse

参数	参数类型	描述
consumer	String	订单消费者身份标识
consumerName	String	订单消费者名称
orderSeq	String	订单序列号
provider	String	订单提供者身份标识
providerName	String	订单提供者名称
productID	String	数据集产品id
productName	String	数据集产品名称
price	String	订单价钱
applyTime	String	订单申请时间
encryptedAesKey	String	密钥
status	String	订单状态
reason	String	订单申请原因
lockProof	String	订单锁定证明
creatorDID	String	流程创建者DID，如果没有加入任何流程，为“ ”
processID	String	当前订单所属流程ID，如果没有加入任何流程，为“ ”

表 5-230 PaginationResp

参数	参数类型	描述
currentPage	Integer	当前页码
pageSizeNum	Integer	每页条数
totalItems	Integer	总条数

状态码： 500

表 5-231 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "currentPage" : "string",
  "pageSizeNum" : "string",
  "provider" : "string",
  "searchText" : "string",
  "status" : "string",
  "consumer" : "did:example:3TMWx8owKHARgNwbj4ywmG"
}
```

响应示例

状态码： 200

订单分页信息

```
{
  "items" : [ {
    "consumer" : "did:example:3TMWx8owKHARgNwbj4ywmG",
    "consumerName" : "Tyler",
    "orderSeq" : "1",
    "provider" : "did:example:DHkJyD5wZwya6sd6BNBnG",
    "providerName" : "hw",
    "productID" : "product1",
    "productName" : "prodname1",
    "price" : "0",
    "applyTime" : "1607332359",
    "encryptedAesKey" : "BNGhPwjaTgpM+V7czzw1i4mH21KKN+XLKXHLqVsRlfbUCncqZNfomkRfzX4WEHj+oty1X9oCd4h6xMnRvs8BWE5Tvg6BJ6QTW/km9EO/FSYqzJf2GqQzAleAcLJrTBZ3LRbPaF87CgJ114ae7R+VK9VvfXQ8exuH2KMRD305dXieGpM4VPVv9u1BbL15Jpd/g==",
    "status" : "ready",
    "reason" : "I want product1",
  } ]
}
```

```

    "lockProof" : ""
  },
  "pagination" : {
    "currentPage" : 1,
    "pageSizeNum" : 100,
    "totalItems" : 10
  }
}

```

状态码： 500

失败响应

```

{
  "errorCode" : "BCS.5002046",
  "errorMsg" : "Incorrect number of arguments"
}

```

状态码

状态码	描述
200	订单分页信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.4 属性加密的密钥管理

5.5.4.1 初始化 ABE 主密钥

功能介绍

初始化ABE主密钥，如果owner未初始化过ABE主密钥，则自动生成并存储在链上。如果owner已有ABE主密钥，不会覆盖。

URI

POST /v1/datashare/abe-setup

请求参数

表 5-232 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法,目前固定为SW

参数	是否必选	参数类型	描述
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
owner	是	String	密钥生成者的身份标识
keyManager Mode	否	String	abe系统首次使用时，需选择中心模式“central”或者多中心模式“distributed”，该模式仅可选择一次，默认为多中心模式。

响应参数

状态码： 200

表 5-233 响应 Body 参数

参数	参数类型	描述
secretJson	String	ABE主私钥
publicKeyJson	String	ABE主公钥

状态码： 500

表 5-234 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "owner" : "did:example:8poVEtNVCry9ecfHSDeQaR"
}
```

响应示例

状态码： 200

ABE主密钥信息

```
{
  "secretJson": "{}",
  "publicKeyJson": "{}"
}
```

状态码： 500

失败响应

```
{
  "errorCode": "BCS.5002046",
  "errorMsg": "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	ABE主密钥信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.4.2 更新 ABE 主密钥

功能介绍

更新ABE主密钥

URI

POST /v1/datashare/abe-update

请求参数

表 5-235 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法,目前固定为SW
cert	是	String	证书

参数	是否必选	参数类型	描述
sk	是	String	私钥
timestamp	是	String	时间戳
secretKeyJson	否	String	json格式的abe主私钥
publicKeyJson	否	String	json格式的abe主公钥
owner	是	String	密钥生成者的身份标识

响应参数

状态码： 200

表 5-236 响应 Body 参数

参数	参数类型	描述
oldSecretJson	String	旧ABE主私钥
oldPublicKeyJson	String	旧ABE主公钥
newSecretJson	String	新ABE主私钥
newPublicKeyJson	String	新ABE主公钥

状态码： 500

表 5-237 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "owner" : "did:example:8poVEtnVCry9ecfHSDeQaR",
  "secretKeyJson" : "string",
```

```
"publicKeyJson" : "string"
}
```

响应示例

状态码： 200

原有及更新后的ABE主密钥信息

```
{
  "oldSecretJson": "{}",
  "oldPublicKeyJson": "{}",
  "newSecretJson": "{}",
  "newPublicKeyJson": "{}"
}
```

状态码： 500

失败响应

```
{
  "errorCode": "BCS.5002046",
  "errorMsg": "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	原有及更新后的ABE主密钥信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.4.3 查询 ABE 主密钥

功能介绍

查询ABE主密钥

URI

POST /v1/datashare/abekey

请求参数

表 5-238 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id

参数	是否必选	参数类型	描述
channelID	是	String	通道id
cryptoMethod	是	String	加密方法,目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
owner	是	String	密钥生成者的身份标识
keyManager Mode	否	String	abe系统首次使用时, 需选择中心模式“central”或者多中心模式“distributed”, 该模式仅可选择一次, 默认为多中心模式。

响应参数

状态码： 200

表 5-239 响应 Body 参数

参数	参数类型	描述
secretKeyJson	String	json格式的abe主私钥
publicKeyJson	String	json格式的abe主公钥

状态码： 500

表 5-240 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "owner" : "did:example:8poVETnVCry9ecfHSDeQaR"
}
```

响应示例

状态码： 200

ABE主密钥信息

```
{
  "secretKeyJson" : "string",
  "publicKeyJson" : "string"
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "BCS.5002046",
  "errorMsg" : "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	ABE主密钥信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.4.4 申请 ABE 用户密钥

功能介绍

申请ABE用户密钥

URI

POST /v1/datashare/abekey-order

请求参数

表 5-241 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法,目前固定为SW
cert	是	String	证书

参数	是否必选	参数类型	描述
sk	是	String	私钥
timestamp	是	String	时间戳
applier	是	String	申请者的身份标识
provider	是	String	授权者的身份标识
attrJson	是	Array of attribute objects	属性列表

表 5-242 attribute

参数	是否必选	参数类型	描述
name	是	String	属性名
type	是	String	属性类型 (plain, comparable)
value	是	String	属性值 (当type为plain时, value为属性值。当type为comparable时, value必须是整数。)
maxValue	否	String	属性值上限, value可能取到的最大值。只在type为comparable时可选使用

响应参数

状态码： 200

表 5-243 响应 Body 参数

参数	参数类型	描述
applier	String	申请者的身份标识
applierName	String	申请者的名称
provider	String	授权者的身份标识
providerName	String	授权者的名称
service	String	授权者的服务名
price	Integer	价格
applyTime	String	申请时间

参数	参数类型	描述
encryptedABE Key	String	被加密的ABE密钥
status	String	申请状态, request表示未授权; ready表示申请已处理
reason	String	原因
lockProof	String	证明
attributesJson	String	属性

状态码: 500

表 5-244 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID": "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID": "mychannel",
  "cryptoMethod": "SW",
  "cert": "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk": "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp": "2020-10-27T17:28:16+08:00",
  "applier": "did:example:Mb4SshJeN5ukWXkbMJK8xC",
  "provider": "did:example:Mb4SshJeN5ukWXkbMJK8xC",
  "attrJson": "[{\"name\":\"att1\",\"type\":\"plain\",\"value\":\"att1name\"},{\"name\":\"att2\",\"type\":\"plain\",\"value\":\"att2name\"},{\"name\":\"att3\",\"type\":\"plain\",\"value\":\"5\"}]"
}
```

响应示例

状态码: 200

ABE用户密钥订单信息

```
{
  "applier": "did:example:Mb4SshJeN5ukWXkbMJK8xC",
  "provider": "did:example:Mb4SshJeN5ukWXkbMJK8xC",
  "applyTime": "1622166512",
  "status": "ready",
  "attributesJson": "[\"att1\": \"YXR0Mm5hbWU=\", \"att2\": \"YXR0Mm5hbWU=\", \"att3\": \"NQ==\"]"
}
```

状态码: 500

失败响应

```
{
  "errorCode": "BCS.5002046",
}
```

```
"errorMsg" : "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	ABE用户密钥订单信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.4.5 授权 ABE 用户密钥

功能介绍

授权ABE用户密钥

URI

PUT /v1/datashare/abekey-order

请求参数

表 5-245 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法,目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
applier	是	String	申请者的身份标识
provider	是	String	授权者的身份标识
attrJson	否	Array of attribute objects	属性列表

表 5-246 attribute

参数	是否必选	参数类型	描述
name	是	String	属性名
type	是	String	属性类型 (plain, comparable)
value	是	String	属性值 (当type为plain时, value为属性值。当type为comparable时, value必须是整数。)
maxValue	否	String	属性值上限, value可能取到的最大值。只在type为comparable时可选使用

响应参数

状态码： 200

表 5-247 响应 Body 参数

参数	参数类型	描述
applier	String	申请者的身份标识
applierName	String	申请者的名称
provider	String	授权者的身份标识
providerName	String	授权者的名称
service	String	授权者的服务名
price	Integer	价格
applyTime	String	申请时间
encryptedABE Key	String	被加密的ABE密钥
status	String	申请状态, request表示未授权; ready表示申请已处理
reason	String	原因
lockProof	String	证明
attributesJson	String	属性

状态码： 500

表 5-248 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID" : "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID" : "mychannel",
  "cryptoMethod" : "SW",
  "cert" : "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk" : "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp" : "2020-10-27T17:28:16+08:00",
  "applier" : "did:example:Mb4SshJeN5ukWXkbMJK8xC",
  "provider" : "did:example:Mb4SshJeN5ukWXkbMJK8xC",
  "attrJson" : "[{\n  \"name\": \"att1\", \"type\": \"plain\", \"value\": \"att1 name\", \"type\": \"plain\", \"value\": \"att2 name\", \"type\": \"att3\", \"type\": \"plain\", \"value\": \"5\"}]"]
}
```

响应示例

状态码： 200

ABE用户密钥订单信息

```
{
  "applier" : "did:hwid:mfqqdiW8V64JbPFgQsoiv",
  "applierName" : "",
  "provider" : "did:hwid:FahQr32NgQZwJGRiCZc37C",
  "providerName" : "",
  "service" : "",
  "price" : 0,
  "applyTime" : "",
  "encryptedABEKey" : "",
  "status" : "ready",
  "reason" : "",
  "lockProof" : "",
  "attributesJson" : "[{\n  \"name\": \"att3\", \"type\": \"comparable\", \"value\": \"3\", \"maxValue\": \"1000\"}]"]
}
```

状态码： 500

失败响应

```
{
  "errorCode" : "BCS.5002046",
  "errorMsg" : "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	ABE用户密钥订单信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.4.6 查询 ABE 用户密钥申请

功能介绍

查询ABE用户密钥申请

URI

POST /v1/datashare/query-abekey

请求参数

表 5-249 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法,目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
provider	是	String	授权者的身份标识
applyer	是	String	申请者的身份标识

响应参数

状态码： 200

表 5-250 响应 Body 参数

参数	参数类型	描述
applyer	String	申请者的身份标识
applyerName	String	申请者的名称
provider	String	授权者的身份标识
providerName	String	授权者的名称
service	String	授权者的服务名

参数	参数类型	描述
price	Integer	价格
applyTime	String	申请时间
encryptedABEKey	String	被加密的ABE密钥
status	String	申请状态, request表示未授权; ready表示申请已处理
reason	String	原因
lockProof	String	证明
attributesJson	String	属性

状态码: 500

表 5-251 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

无

响应示例

状态码: 200

ABE用户密钥订单信息

```
{
  "applier": "did:hwid:mfqqdiW8V64JbPFgQsoiv",
  "applierName": "",
  "provider": "did:hwid:FahQr32NgQZWjGRiCZc37C",
  "providerName": "",
  "service": "",
  "price": 0,
  "applyTime": "1672985584",
  "encryptedABEKey": "",
  "status": "ready",
  "reason": "",
  "lockProof": "",
  "attributesJson": "{\"att1\": \"YXR0Mm5hbWU=\", \"att2\": \"YXR0Mm5hbWU=\", \"att3\": \"NQ==\"}"
}
```

状态码: 500

失败响应

```
{
  "errorCode": "BCS.5002046",
  "errorMsg": "Incorrect number of arguments"
}
```

状态码

状态码	描述
200	ABE用户密钥订单信息
500	失败响应

错误码

请参见[错误码](#)。

5.5.4.7 ABE 用户密钥解密数据

功能介绍

ABE用户密钥解密数据

URI

POST /v1/datashare/abe-decrypt

请求参数

表 5-252 请求 Body 参数

参数	是否必选	参数类型	描述
orgID	是	String	组织id
channelID	是	String	通道id
cryptoMethod	是	String	加密方法,目前固定为SW
cert	是	String	证书
sk	是	String	私钥
timestamp	是	String	时间戳
encryptData	是	String	数据密文, 当onChainStore设置为“true”时, 可不输入
applier	是	String	申请者的身份标识
provider	是	String	授权者的身份标识
orderSeq	否	String	订单序列号, 当onChainStore为true时, 订单序列号必填。

参数	是否必选	参数类型	描述
onChainStore	否	String	数据密文是否在链上存储，可设置为“true”或“false”，默认为“false”。如果设置为“true”，则不需要输入encryptData，可自动在链上获取数据密文

响应参数

状态码： 200

表 5-253 响应 Body 参数

参数	参数类型	描述
plainData	String	base64处理过的解密后数据

状态码： 500

表 5-254 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码
errorMsg	String	错误描述

请求示例

```
{
  "orgID": "ce0ac69b0c8648cd25b44a551780409767c8890b",
  "channelID": "mychannel",
  "cryptoMethod": "SW",
  "cert": "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
  "sk": "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----",
  "timestamp": "2020-10-27T17:28:16+08:00",
  "encryptData": "string",
  "onChainStore": "false",
  "applier": "did:example:Mb4SshJeN5ukWXkbMJK8xC",
  "provider": "did:example:Mb4SshJeN5ukWXkbMJK8xC"
}
```

响应示例

状态码： 200

base64编码的ABE解密后数据

```
{
  "plainData": "aGVsbG8sdGhpcyBpcyBhbiBleGFtcGxlIGZvciBhYmU="
}
```

状态码：500

失败响应

```
{  
  "errorCode": "BCS.5002046",  
  "errorMsg": "Incorrect number of arguments"  
}
```

状态码

状态码	描述
200	base64编码的ABE解密后数据
500	失败响应

错误码

请参见[错误码](#)。

6 附录

6.1 国密加密

6.1.1 概述

国密是国家商用密码的简称，商用密码是指对不涉及国家秘密内容的信息进行加密保护或者安全认证所使用的密码技术和密码产品。

国密算法是国家密码管理局制定的自主可控的国产算法，可提高加密强度和加解密性能。使用国密加密，可以满足政府机构、事业单位、大型国企、金融银行等行业的改造和国密算法的需求。

华为云区块链服务提供国密加密算法SDK供您使用，供用户开发客户端程序以及对私钥文件进行加密保护。

资源下载

表 6-1 SDK 列表

配套社区Hyperledger Fabric版本	语言	下载链接
Fabric 1.4.0、Fabric 2.2	Go	登录区块链服务管理控制台，进入“应用案例”，单击“国密加密SDK”中 Fabric_SDK_Go的“下载”按钮。
	Java	登录区块链服务管理控制台，进入“应用案例”，单击“国密加密SDK”中 Fabric_SDK_Gateway_Java或 Fabric_SDK_Java的“下载”按钮。
说明 <ul style="list-style-type: none">Go版本要求：1.12及以上，1.16以下（≥ 1.12，< 1.16）。国密SDK涵盖了普通SDK的所有功能，并在此基础上增加了对国密算法的支持。Fabric_SDK_Gateway_Java对SDK的部分接口进行了封装，涵盖Fabric_SDK_Java同时更加简便易用，推荐选用。		

将下载的压缩包解压后，得到如下目录，目录的功能如下表：

目录	说明
src (仅Go)	存放Go SDK的源码文件。
jar (仅Java)	存放Java SDK的Jar包。

6.1.2 SDK 的使用

安装 SDK

如何获取GO、JAVA压缩包、Jar文件请参考[概述](#)。

- GO：将下载的压缩包解压到用户的\$GOPATH目录下。
- Java：将下载的压缩包中的Jar文件添加到项目的依赖中，可按以下方式添加：

- a. 将下载的SDK Jar包注册至Maven本地仓库，可参考以下命令：

```
Mvn install:install-file -Dfile=fabric-sdk-java-2.2.6-jar-with-dependencies.jar -DgroupId=org.hyperledger.fabric-sdk-java -DartifactId=fabric-sdk-java -Dversion=2.2.6-BCS -Dpackaging=jar
```

- b. 在项目中依赖SDK，可参考以下代码：

```
<dependency>  
  <groupId>org.hyperledger.fabric-sdk-java</groupId>  
  <artifactId>fabric-sdk-java</artifactId>  
  <version>2.2.6-BCS</version>  
</dependency>
```

开发 Client 程序

您需要自行开发应用程序业务逻辑代码。国密SDK与FabricSDK的使用方式一致，若需要使用国密加密算法，仅需在创建实例时选择国密加密，并在客户端中将FabricSDK替换为国密SDK即可。

运行 Client 程序

Client程序运行时一般要设定其使用的配置文件路径、通道名称、链代码名称、组织ID等。

- 配置文件路径即用户下载配置文件的存放路径。
- 通道名称即BCS实例中的通道名称。
- 链代码名称即BCS实例中安装链代码时设定的名称。
- 组织ID，以如下示例配置文件内容为例，组织ID为02f23ab00f6e1ffcde8a27bfd3ac2290edc18127

```
client:  
organization: 02f23ab00f6e1ffcde8a27bfd3ac2290edc18127
```

6.1.3 附录

fabric-sdk-client/go依赖的第三方包列表：

序号	包名
1	github.com/Knetic/govaluate
2	github.com/VividCortex/gohistogram
3	github.com/cloudflare/cfssl
4	github.com/go-kit/kit
5	github.com/golang/mock
6	github.com/golang/protobuf
7	github.com/hashicorp/hcl
8	github.com/hyperledger/fabric-config
9	github.com/hyperledger/fabric-lib-go
10	github.com/hyperledger/fabric-protos-go
11	github.com/magiconair/properties
12	github.com/miekg/pkcs11
13	github.com/mitchellh/mapstructure
14	github.com/pelletier/go-toml
15	github.com/pkg/errors
16	github.com/prometheus/client_golang
17	github.com/spf13/afero
18	github.com/spf13/cast
19	github.com/spf13/jwalterweatherman
20	github.com/spf13/pflag
21	github.com/spf13/viper
22	github.com/stretchr/testify
23	github.com/tjfoc/gmsm
24	google.golang.org/grpc
25	gopkg.in/yaml.v2

6.2 同态加密

6.2.1 概述

华为云区块链服务提供同态加密库供您使用，方便您进行开发。同态加密是一类具有特殊自然属性的加密方法，与一般加密算法相比，同态加密除了基本加密外，还能实现密文间的多种计算功能，对于保护信息的安全具有重要意义。利用同态加密技术可以实现无密钥方对密文的计算，密文计算无须经过密钥方，既可以减少通信代价，又可以转移计算任务，可平衡各方的计算代价。利用同态加密技术可以实现让解密方只能获知最后的结果，而无法获得每一个密文的消息，可以提高信息的安全性。

BCS提供客户端库和Chaincode库，该库主要用于交易类的密文运算服务，达到用户交易的隐私保护。

- 客户端库：用于在client端提供加法同态功能和生成交易金额的证明信息。
- 同态加密链代码IDChaincode.go：在同态加密的场景下，用户在部署应用前需要下载安装并且实例化此链代码至区块链实例。
- Chaincode库：提供零知识证明功能，用于在密文条件下，校验用户交易的证明，并生成交易后的数据，使背书者无需解密用户交易的数据，达到余额范围的判断。

资源下载

表 6-2 库列表

配套社区 Hyperledger Fabric 版本	库版本	下载链接
Fabric 1.1.0、Fabric 1.4.0、Fabric 2.2	1.8.5	同态加密库
	1.9.2	同态加密库
	1.11.5	同态加密库
<ul style="list-style-type: none"> • 同态加密链代码下载 IDChaincode.go • Chaincode库接口文件下载 api_ahe_cc.tar.gz 		
<p>须知</p> <ul style="list-style-type: none"> • 需要选择与本地编译环境相一致版本的包。例如本地使用的go编译器为1.8.5，则下载1.8.5版本的库。 • 使用同态加密库需要提前安装好国密SDK。 • api_ahe_cc.tar.gz包仅用于本地编译。 		

6.2.2 同态加密库的使用

介绍同态加密库的使用。

操作步骤

步骤1 实例订购及文件下载。

订购BCS实例时，安全机制选择使用“ECDSA”同态加密。完成订购后，下载同态加密库ahex.x.x.tar.gz，客户端开发套件国密SDK包sdkx.x.x.tar.gz，peer与orderer用户证书，配置文件等。

步骤2 安装客户端SDK库。

将下载的国密SDK包sdkx.x.x.tar.gz解压到用户的\$GOPATH目录下。

步骤3 安装同态加密库。

将下载的ahex.x.x.tar.gz解压到用户的\$GOPATH目录下。

步骤4 安装依赖库（仅针对Fabric1.1）。

依赖的库文件位于同态加密库目录下，当同态加密库被解压到用户的\$GOPATH目录后，该文件位于\$GOPATH/src/ahe/PSW/deps/lib，请将该目录下的所有文件复制至本地的/usr/local/include/openssl/目录下（如果没有该目录，请自行创建该目录），然后设置环境变量：

```
export LD_LIBRARY_PATH=/usr/local/include/openssl:$LD_LIBRARY_PATH
//以项目路径为/usr/local/include/openssl为例，若自行创建目录，请以实际创建目录为准
```

📖 说明

若提示缺少libgmp，则需要用户安装gmp库。安装方法可以参考linux操作系统包管理工具（例如ubuntu系统可以使用apt-get install libgmp10命令来安装，也可在<https://gmplib.org/>下载源码进行编译安装）。

步骤5 开发Client程序与链代码。

请参考[AHE Lib库接口](#)和[Chaincode库接口](#)进行具体的应用和链代码（智能合约）的开发。

例如App客户端和链代码端的逻辑过程参考如下：

- App客户端的典型逻辑过程是：
 - a. 注册用户
注册用户时可以调用密钥生成函数为用户生成公私钥。
 - b. 初始化余额
初始化余额时可以调用初始余额准备函数生成具有隐私保护的初始余额信息。
 - c. 发起交易
交易时可以调用交易准备函数生成具有隐私保护的交易数据。
- 链代码端对应的逻辑过程是：
 - a. 保存用户公钥与地址的映射关系。
 - b. 验证初始余额的有效性并生成初始交易。
 - c. 验证交易数据的有效性并生成交易结果。

📖 说明

- 链码端可以通过调用初始余额校验函数来验证初始余额的有效性。
- 调用交易校验函数来验证交易数据的有效性。

步骤6 安装链代码。

通过管理界面，将开发的链代码安装到用户订购的BCS实例中并实例化。

步骤7 部署应用。

用户根据业务需求开发app应用，app应用可以调用同态加密库来对交易信息进行隐私保护。开发完成后，将app部署在购买的服务服务器上。app部署完成后，确保环境中具有配置文件、证书文件（peer和orderer的用户证书）、openssl库。

----结束

6.2.3 AHE Lib 库接口

提供给用户静态库文件，用于用户开发客户端应用时集成使用同态加密功能。

库的引用路径为：`import "ahe/PSW/api/ahelib"`

GenerateKey

- **接口原型**
func GenerateKey(pwd string) (privKeyStr string, pubKeyStr string, err error)
- **功能描述**
生成同态加密的密钥对。
- **输入说明**

参数名	类型	描述	是否必须
pwd	string	用于加密保护生成的同态私钥串，采用AES-128加密。 pwd至少满足如下要求： 1. 长度至少6个字符。 2. 必须包含如下至少两种字符的组合： <ul style="list-style-type: none"> • 至少一个小写字母； • 至少一个大写字母； • 至少一个数字； • 至少一个特殊字符：`~!@#\$\$%^&*()-_+=+ [{];:","<.>/?` 和空格 	是（不能为空）

- **输出说明**

参数名	类型	描述
privKeyStr	string	使用pwd加密的同态加密私钥。
pubKeyStr	string	同态加密公钥串
err	error	错误

- **注意事项**
保护密钥的复杂度取决于用户实际的应用场景，作为底层库的调用，不去严格限制。

Encrypt

- **接口原型**
func Encrypt (secretNumStr string, pubKeyStr string) (ciphertext string, err error)

- **功能描述**
同态加密。

- **输入说明**

参数名	类型	描述	是否必须
secretNumStr	string	需要加密的数值，只支持大于等于0的正整数，如果数字有小数，需要扩大倍数传入。	是
pubKeyStr	String	同态加密的公钥	是

- **输出说明**

参数名	类型	描述
ciphertext	string	加密后的数据
err	error	返回错误

- **注意事项**
无。

Decrypt

- **接口原型**
func Decrypt(ciphertext string, privKeyStr string, pwd string) (plainText *big.Int, err error)

- **功能描述**
同态解密。

- **输入说明**

参数名	类型	描述	是否必须
ciphertext	string	需要解密的密文	是
privKeyStr	string	同态加密的私钥串（被pwd加密保护	是
pwd	string	用于保护私钥的字符串	否

- 输出说明

参数名	类型	描述
plainText	*big.int	解密后的数据
err	error	返回错误

- 注意事项

无。

Add

- 接口原型

func Add(cipher1, cipher2 string) (cipher string, err error)

- 功能描述

同态加法。

- 输入说明

参数名	类型	描述	是否必须
cipher1	string	被加密的密文1	是
cipher2	string	被加密的密文2	是

- 输出说明

参数名	类型	描述
cipher	string	相加后的数据
err	error	返回错误

- 注意事项

无。

InitBalance

- 接口原型

func InitBalance(balanceStr string, pubKey string) (string, error)

- 功能描述

初始化余额。

- 输入说明

参数名	类型	描述	是否必须
pubKey	string	公钥串	是

参数名	类型	描述	是否必须
balanceStr	string	初始余额，只支持大于等于0的正整数，如果数字有小数，需要扩大倍数传入	是

- 输出说明

参数名	类型	描述
balanceInfo	string	交易金额信息
err	error	返回错误

- 注意事项
无。

PrepareTxInfo

- 接口原型

```
func PrepareTxInfo(cipherBalanceA string, transNumStr string, pubKeyA, pubKeyB string, privKeyA string, pwd string) (string, error)
```

- 功能描述

交易准备。

- 输入说明

参数名	类型	描述	是否必须
cipherBalanceA	string	A的当前余额（密文），取链上当前A的余额	是
transNumStr	string	转账金额（明文）	是
pubKeyA	string	A的公钥串	是
pubKeyB	string	B的公钥串	是
PrivKeyA	string	A的私钥串	是
pwd	string	用于加密保护的字符串	否

- 输出说明

参数名	类型	描述
Txinfo	string	交易准备数据
err	error	返回错误

- **注意事项**
无。

6.2.4 Chaincode 库接口

该静态库集成在BCS实例中。用户在开发链代码时，可以使用BCS提供的API接口文件对开发中的链码进行本地编译。

先将API接口文件下载（下载链接参见[资源下载](#)）并解压到本地的GOPATH目录中，按照4.2.6章节的链代码示例代码来引用同态库。当链码开发完成后，将链码安装到BCS中时，链码会自动链接到BCS中的库代码，实现对链码端同态加密库的调用。

链码中调用同态加密库的引用路径为：`import "ahe/PSW/api/ChainCode"`

说明

请确保使用该引用路径，否则链码端调用同态加密库会失败。

ValidateInitBalance

- **接口原型**
`func ValidateInitBalance(BalanceInfo, PubKey string) (InitBalance string, error error)`
- **功能描述**
校验sdk.InitBalance生成的balanceinfo中余额证明的有效性。
- **输入说明**

参数名	类型	描述	是否必须
BalanceInfo	string	初始余额数据	是
Pubkey	string	余额加密的公钥信息	是

- **输出说明**

参数名	类型	描述	是否必须
InitBalance	string	初始余额密文	是
err	error	错误信息	是

- **处理说明**
验证余额有效后，返回余额的密文。
- **注意事项**
这里用户余额真实性由用户的app逻辑保证，Chaincode端无法验证该用户的真实金额，只能验证该金额是大于0的范围。

ValidateTxInfo

- **接口原型**

ValidateTxInfo(txInfo, cipherBalanceA, cipherBalanceB string)
(newCipherBalanceA,newCipherBalanceB,newCipherTxA,newCipherTxB
string,err error)

- **功能描述**

校验PrepareTxInfo生成的Txinfo中交易证明的有效性。

- **输入说明**

参数名	类型	描述	是否必须
txinfo	string	交易证明数据 说明 交易信息数据包含有交易密文数据和交易证明数据，该信息来源于sdk.PrepareTxInfo返回的txinfo信息。	是
cipherBalanceA	string	A当前的交易余额（密文）	是
cipherBalanceB	string	B当前的交易余额（密文）	是

- **输出说明**

参数名	类型	描述
newCipherBalanceA	string	交易后待更新的A的余额
newCipherBalanceB	string	交易后待更新的B的余额
newCipherTxA	string	交易金额（A的同态加密公钥加密）
newCipherTxB	string	交易金额（B的同态加密公钥加密）
err	error	错误

- **注意事项**

这里A是转账方，B是收款方。

6.2.5 IDChaincode

用于保存用户的公钥和账户，新生成用户的同态密钥对时，需要将公钥注册到IDchaincode上，便于后续根据账户能查询到收款方的同态公钥。链代码IDChaincode.go的下载请参见[资源下载](#)。

📖 说明

IDChaincode.go由华为云BCS服务提供，不建议用户修改，若进行修改将与[链代码示例](#)逻辑不一致。

注册 Register

账户地址是通过公钥hash计算转16进制字符串得到。

查询 Query

使用账户地址查询公钥。

6.2.6 链代码示例

交易链码是用户实现其业务逻辑的链码，这里给出的示例代码Transaction Chaincode完成用户间的转账操作。在转账数据的验证过程中使用同态加密库对密文交易数据进行合法性校验，确保没有非法操作。示例中实现了余额初始化，余额查询，转账交易三个功能函数，具体功能实现参考如下。

Init 初始化余额

当用户注册成功后，需要初始化该用户的余额，默认为0。

- 输入说明

参数名	类型	描述	是否必须
PubKey	string	公钥	是
Balance	string	初始金额	是

- 处理

调用ValidateInitBalance，校验balance范围有效性，完成余额背书。

```
PubKey := string(args[0])
BalanceInfo := string(args[1])
//PubKey := string(args[2])

hashPubkey, err := t.calcAddr(PubKey)
logger.Debug("encrypt initial balance")
//validate balance
cipherBalance,err := pswapi_cc.ValidateInitBalance(BalanceInfo,PubKey)
if err != nil {
    logger.Error("fail to Validate InitBalance ")
    return shim.Error("fail toValidate InitBalance")
}
```

- 输出说明

参数名	类型	描述
newCipherBalance	string	余额密文

QueryBalance 查询余额

查询余额功能比较简单，根据key来查询value，即根据账户地址来查询当前余额，具体可见transaction_demo的queryBalance接口。

queryBalance接口参数如下：

-	参数名	类型	描述
输入	addr	string	账户地址
输出	cipherbalance	string	账户余额密文
输出	err	error	错误信息

Transfer 转账

- 输入说明

参数名	类型	描述	是否必须
AddrA	string	A-转账者地址	是
AddrB	string	B-收账者地址	是
txinfo	String	交易信息PrepareTxInfo	是

- 处理说明

a. 根据账户地址获取账本中A,B的当前余额cipherBalanceAKeyABlock, cipherBalanceBKeyBBlock。

b. 校证明有效性。

```
newCipherBalanceA,newCipherBalanceB,newCipherTxA,newCipherTxB, err :=
pswapi_cc.ValidateTxInfo(txInfo, cipherBalanceAKeyABlock, cipherBalanceBKeyBBlock)
if err != nil {
    logger.Error("fail to validate trans information")
    return shim.Error("fail to validate trans information")
}
```

c. 输出更新后的余额（密文）

业务的账本内容需要用户定制，将上面加密的金额合入到用户的账本中保存，demo中定义了一个存储结构，保存完后通过json序列化为一个交易记录对象进行保存。

```
type TransRecord struct {
    FromAddr string
    ToAddr string
    TXType string
    Balance string
    TX string
    remark string
}
```

6.2.7 应用示例

为了说明同态加密库的具体使用方法，提供一个应用示例代码和对应的链码示例代码。该应用的主要功能是实现用户间相互转账，同时使用同态加密库保护用户的转账交易信息。

该应用的使用包括三个步骤：注册用户（同时会初始化用户余额），用户间转账，查询用户余额。

应用使用命令行的方式进行业务操作，具体过程如下。

注意事项

在下载的sdk.yaml文件中查询orderer和peer的域名信息，将EIP+orderer域名和EIP+peer域名补充进“/etc/hosts”文件中。使用Fabric1.1版本时，peer域名中的EIP需要配置为公网IP，使用Fabric1.4版本时，peer域名中的EIP需要配置为私网IP，否则网络不通将会导致交易验证失败。

在“/etc/hosts”文件中增加orderer节点和peer节点的域名映射时，操作方法如下：

```
X.X.X.X orderer-e5ad7831affbe8e6e2e7984b098f92406930a688-0.orderer-
e5ad7831affbe8e6e2e7984b098f92406930a688.default.svc.cluster.local
X.X.X.X peer-d524b0817aaed85490368776cb88042a149a56b5-0.peer-
d524b0817aaed85490368776cb88042a149a56b5.default.svc.cluster.local
X.X.X.X peer-d524b0817aaed85490368776cb88042a149a56b5-1.peer-
d524b0817aaed85490368776cb88042a149a56b5.default.svc.cluster.local
```

📖 说明

X.X.X.X为示例EIP，请替换为实际EIP。

注册

```
Usage:
  appdemo register [flags]
Flags:
  -C, --channel string    channel id (default "mychannel")
  -c, --config string     configuration file path (default "./config_test.yaml")
  -I, --idcc string       identity chaincode name (default "IDChaincode")
  -i, --initbalance string  init initbalance
  -o, --orgid string       organization id (default "org1")
  -p, --protectpwd string  protect pwd
  -T, --txcc string       transaction chaincode name (default "TxChaincode")
  -u, --userid string     user id
./appdemo register -u A -p test -i 100
```

📖 说明

部分参数不配置时使用的为默认值，见Flags中的描述。如果用户配置的与默认值不同，需要在参数中显示指定。

- 为用户生成一对同态公私钥

这里假设有个用户表示userid，用于区分用户，新用户注册的时候为这个用户生成一对公私钥，这里demo为每个用户将密钥对写到了本地的\${userid}.data文件。

```
privKeyStr, pubKeyStr, err := pswapi_sdk.GenerateKey(propwd)
check(err)
fmt.Println("key is nil")
userdata.PubKey = pubKeyStr
userdata.PriKey = privKeyStr
```

- 注册公钥

使用fabric SDK的接口向IDChaincode 注册，参数为生成的公钥。

📖 说明

注册公钥是为了使大家的同态公钥共享，这里利用了区块链本身链码特性完成该功能。主要作用是在转账的时候，使用转账者的公钥加密交易信息，保护交易隐私，只有私钥持有者才可以解密交易内容。

```
res, err := sdk_client.Invoke(setup, "Register", [][]byte{[]byte(userdata.PubKey), []byte(senderAddr)})
if err != nil {
    fmt.Println("Fail to register user pk ", err.Error())
} else {
    addrByte := res[0].ProposalResponse.GetResponse().Payload
    fmt.Println("Register addr: ", string(addrByte))
}
```

- 注册初始余额
 - a. 使用sdk.InitBalance来初始余额，进行加密生成初始化balanceinfo。
 - b. 发送balanceinfo到交易Chaincode。

```
balanceInfo, err := pswapi_sdk.InitBalance(initbalance, userdata.PubKey) check(err)
setup.ChainCodeID = txchaincode
_, err = sdk_client.Invoke(setup, "init", [][]byte{[]byte(userdata.PubKey), []byte(balanceInfo)})
if err != nil {
    fmt.Println("Initbalance error for user: ", senderAddr, err.Error())
} else {
    fmt.Println("init balance successfully: ", senderAddr)
}
check(err)
```

交易

```
Usage:
  appdemo transaction [flags]
Flags:
  -b, --AddrB string      B' addr
  -t, --Tx string         Transaction Num
  -C, --channel string    channel id (default "mychannel")
  -c, --config string    configuration file path (default "./config_test.yaml")
  -l, --idcc string      identity chaincode name (default "IDChaincode")
  -o, --orgid string     organization id (default "org1")
  -p, --protectpwd string protect pwd
  -T, --txcc string      transaction chaincode name (default "TxChaincode")
  -u, --userid string    user id
./appdemo transaction -u A -p test -b
a0760184f7ed24e0d86f5b2df40a973a9e1b5da9a1ae886532ac9cd634b59d59 -t 10
//说明：其中-b后的字符串为B的账户地址（B注册成功时回显的地址）。
```

查询

```
Usage:
  appdemo querybalance [flags]
Flags:
  -C, --channel string    channel id (default "mychannel")
  -c, --config string    configuration file path (default "./config_test.yaml")
  -l, --idcc string      identity chaincode name (default "IDChaincode")
  -o, --orgid string     organization id (default "org1")
  -p, --protectpwd string protect pwd
  -T, --txcc string      transaction chaincode name (default "TxChaincode")
  -u, --userid string    user id
./appdemo querybalance -p test -u A
```

查询代码：

1. 调用sdk提供的InitBalance接口初始化余额。
2. 调用fabirc sdk发送交易。

```
get balance
setup.ChainCodeID = txchaincode
transRec := sdk_client.TransRecord{}

fmt.Println("query balance")

resps, err := sdk_client.Query(setup, "QueryBalance", [][]byte{[]byte(addrA)})
if err != nil {
```

```

fmt.Println("Fail to query balance :", err.Error())
return err
}

err = json.Unmarshal(resps[0].ProposalResponse.GetResponse().Payload, &transRec)
if err != nil {
    fmt.Println("unmarshal query result error: ", err.Error())
    return err
}

decrypt balance

curbalance, err := pswapi_sdk.Decrypt(transRec.Balance, userdata.PriKey, propwd)
if err != nil {
    fmt.Println("sdk Decrypt error: ", err.Error())
    return err
}

fmt.Println("current balance:" + curbalance.String())
    
```

6.2.8 同态加密交易验证 Demo

介绍同态加密交易验证Demo的使用方法。

📖 说明

只用于场景体验，不用于实际应用。

操作步骤

步骤1 订购BCS实例。

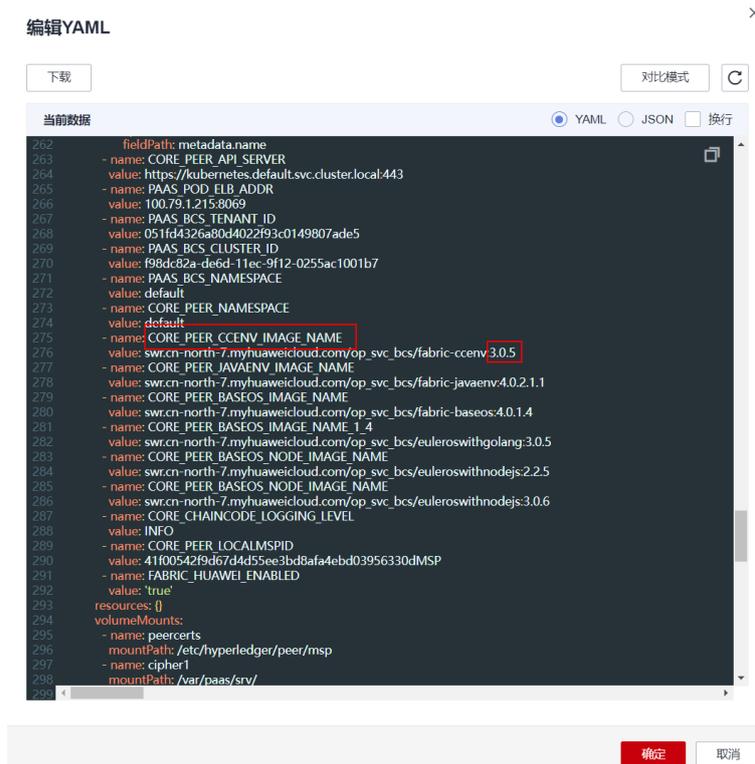
版本为4.X.X（对应社区Hyperledger Fabric 2.2版本），区块链实例名称推荐使用全英文字符，安全机制选择ECDSA，组织数量和名称使用默认配置。

步骤2 修改链码容器版本。

1. 选择实例管理，单击打算安装同态加密链代码的实例的容器集群。
2. 单击工作负载页签，切换到对应集群，编辑peer负载的yaml文件。



3. 修改CORE_PEER_CCENV_IMAGE_NAME的版本号为3.0.5。



步骤3 安装并实例化链代码。

安装示例链代码：[transaction.zip](#)、[IDChaincode.zip](#)，并实例化链代码。

📖 说明

- 为了方便后续操作，建议安装的链代码名称固定为 **transaction**、**idchaincode**。
- 示例链代码中提供交易的脚本中已将链代码版本固定为1.0，安装链代码时，链代码版本号必须为1.0，链代码语言为go。
- 如果自己开发链代码，可以使用Chaincode库接口文件：[api_ahe_cc.tar.gz](#)。

步骤4 在“实例管理”界面，在实例卡片中，单击“获取客户端配置”。

步骤5 勾选需要下载的内容，参数请与如下内容完全保持一致：

- 勾选“SDK文件”：
 - 链代码名称：transaction
 - 证书存放路径：/home/paas
 - 通道名称：channel
 - 选择成员：organization
- 勾选“共识节点证书”。
- 勾选“Peer节点证书”，指定节点组织保持默认值，勾选“管理员证书”。

步骤6 单击“下载”，下载SDK配置文件、共识节点证书和Peer节点证书。

步骤7 在本地服务器安装golang。

1. 下载安装包：[go1.11.5.linux-amd64.tar.gz](#)，上传到本地服务器“/usr/local”目录下并解压。

```
tar -zxvf go1.11.5.linux-amd64.tar.gz
```

```
[root@cluster-bcs-064s-zeb9 ~]# cd /usr/local/
[root@cluster-bcs-064s-zeb9 local]# ls -l
total 136904
drwxr-xr-x. 2 root root 4096 Feb 28 21:31 bin
drwxr-xr-x. 2 root root 4096 Jul 22 2019 etc
drwxr-xr-x. 2 root root 4096 Jul 22 2019 games
drwxr-xr-x 10 root root 4096 Jan 24 2019 go
-rw----- 1 root root 140132627 Feb 28 21:56 go1.11.5.linux-amd64.tar.gz
drwxr-xr-x 258 root root 4096 Jan 24 2019 gocache
drwxr-xr-x. 3 root root 4096 Feb 28 22:02 include
drwxr-xr-x. 2 root root 4096 Jul 22 2019 lib
drwxr-xr-x. 2 root root 4096 Jul 22 2019 lib64
drwxr-xr-x. 2 root root 4096 Jul 22 2019 libexec
drwxr-xr-x. 2 root root 4096 Jul 22 2019 sbin
drwxr-xr-x. 5 root root 4096 Sep 9 17:15 share
drwxr-xr-x. 2 root root 4096 Jul 22 2019 src
drwxr-xr-x 2 root root 4096 Jan 24 2019 tmp
[root@cluster-bcs-064s-zeb9 local]#
```

2. 将以下环境变量配置到“/etc/profile”文件中。

```
export GOROOT=/usr/local/go
export PATH=$PATH:$GOROOT/bin
export GOPATH=/opt/gopath
export PAAS_CRYPTO_PATH=/opt/hao
export PAAS_SSL_ROOT=/opt/hao
```

3. 执行如下命令使环境变量生效。

```
source /etc/profile
```

步骤8 编译appdemo。

1. 进入“/opt/gopath”目录下（如果没有gopath目录，请手动创建），上传sdk库（[sdk1.11.5.tar.gz](#)）、同态加密库（[ahelib1.11.6.tar.gz](#)）和openssl库（[openssl.tar.gz](#)），均解压至当前目录，命令：

```
tar -zxvf xxx
```

```
[root@cluster-bcs-064s-zeb9 local]# cd /opt/gopath
[root@cluster-bcs-064s-zeb9 gopath]# ls -l
total 48148
-rw----- 1 root root 14129773 Feb 28 22:00 ahelib1.11.5.tar.gz
drwxr-xr-x 2 root root 4096 Jul 19 2019 bin
drwxr-xr-x 2 root root 4096 Jul 19 2019 openssl
-rw----- 1 root root 7891504 Feb 28 22:00 openssl.tar.gz
drwxr-xr-x 3 root root 4096 Jul 19 2019 pkg
-rw----- 1 root root 27261385 Feb 28 22:00 sdk1.11.5.tar.gz
drwxr-xr-x 4 root root 4096 Jul 19 2019 src
[root@cluster-bcs-064s-zeb9 gopath]#
```

2. 进入“/opt/gopath/src/ahelib/PSW/deps/lib”目录，把路径里的文件复制到“/usr/local/include/openssl/”目录中（如果没有“openssl”目录则新建）。
3. 进入“/opt/gopath/src/ahelib/PSW/example/appdemo/”目录，执行go build，编译appdemo文件。

```
[root@cluster-bcs-064s-zeb9 appdemo]# cd /opt/gopath/src/ahelib/PSW/example/appdemo/
[root@cluster-bcs-064s-zeb9 appdemo]# go build
```

```
[root@cluster-bcs-064s-zeb9 appdemo]# ls -l
total 22664
-rwx----- 1 root root 23189440 Feb 28 22:03 appdemo
-rwxr-xr-x 1 root root 269 Jul 19 2019 appdemo.go
drwxr-xr-x 2 root root 4096 Jul 19 2019 cmd
drwxr-xr-x 2 root root 4096 Jul 19 2019 sdk_client
drwxr-xr-x 3 root root 4096 Jul 19 2019 vendor
[root@cluster-bcs-064s-zeb9 appdemo]#
```

步骤9 打包镜像。

1. 在“/home/paas/”下创建一个文件夹用来打包镜像，例如：`mkdir cj`
2. 将步骤步骤5中下载的包解压，在“/home/paas/cj”路径下，上传共识节点管理员证书和peer节点的管理员证书，并将步骤6.3中编译好的appdemo文件、openssl库和sdk库以及Dockerfile复制至当前目录下并解压。

```
-rwx----- 1 root root 23189440 Feb 28 22:07 appdemo
-rw----- 1 root root 10237 Feb 28 22:11 bcs-tongtaitest-orderer-admin.zip
-rw----- 1 root root 170 Feb 28 22:10 Dockerfile
-rw----- 1 root root 7891504 Feb 28 22:08 openssl.tar.gz
-rw----- 1 root root 27261385 Feb 28 22:09 sdk1.11.5.tar.gz
-rw----- 1 root root 10048 Feb 28 22:11 testp-admin.zip
-rw----- 1 root root 355431424 Feb 28 22:22 tongtaitest.tar
[root@cluster-bcs-064s-zeb9 cj]#
```

说明

解压后的openssl文件夹下缺少两个lib库文件（libltdl.so.7和libltdl.so.7.3.0），请将这两个lib文件复制至“/home/paas/cj/openssl”目录下。

3. 解压共识节点管理员证书和peer节点的管理员证书：**unzip xxx.zip**，并将解压出来的证书文件分别移至“/home/paas/cj/orderer”文件夹和“/home/paas/cj/peer”文件夹（如果没有则创建）。如下图：

```
[root@cluster-bcs-064s-zeb9 cj]# ls -l peer orderer
orderer:
total 8
drwx----- 7 root root 4096 Feb 28 22:11 msp
drwx----- 2 root root 4096 Feb 28 22:11 tls

peer:
total 8
drwx----- 7 root root 4096 Feb 28 22:11 msp
drwx----- 2 root root 4096 Feb 28 22:11 tls
[root@cluster-bcs-064s-zeb9 cj]#
```

4. 在本地解压步骤步骤5中下载的包，从sdk-config文件夹中获取yaml文件，并修改其中证书路径的配置。

例如：

- a. 将所有路径中的节点域名地址删除哈希值前缀，修改为如下图的peer地址。

```
organizations:
  aa73c757c9026fb623495d7058ca177f6152bcea:
    mspid: aa73c757c9026fb623495d7058ca177f6152bceaMSP
    cryptoPath: /home/paas/peer/msp
    tlsCryptoKeyPath: /home/paas/peer/tls/server.key
    tlsCryptoCertPath: /home/paas/peer/tls/server.crt
  peers:
    - peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30605
    - peer-aa73c757c9026fb623495d7058ca177f6152bcea-1.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30606
  certificateAuthorities:
    - ca-org1
  ordererorg:
    mspID: "2cf8802066c1c5011fd396c54c9126a17c9cfc9MSP"
    cryptoPath: /home/paas/orderer/msp
```

- b. 排查解压后的证书相关路径，修改完成后，sdk配置文件中不存在带hash值的路径。

```
Search "/home/paas/" (7 hits in 1 file)
C:\Users\c00206633\Desktop\test-sdk-config\yaml (7 hits)
Line 81: cryptoPath: /home/paas/peer/msp
Line 82: tlsCryptoKeyPath: /home/paas/peer/tls/server.key
Line 83: tlsCryptoCertPath: /home/paas/peer/tls/server.crt
Line 97: cryptoPath: /home/paas/orderer/msp
Line 109: path: /home/paas/peer/msp/tlscacerts/tlsca.2cf8802066c1c5011fd396c54c9126a17c9cfc9-cert.pem
Line 124: path: /home/paas/peer/msp/tlscacerts/tlsca.aa73c757c9026fb623495d7058ca177f6152bcea-cert.pem
Line 137: path: /home/paas/peer/msp/tlscacerts/tlsca.aa73c757c9026fb623495d7058ca177f6152bcea-cert.pem
```

- c. 去掉sdk配置文件中certificateAuthorities部分代码块（如果没有则跳过），保存后，将该文件上传至“/home/paas/cj”路径下。

```
certificateAuthorities:
  ca-org1:
    url: https://ca_peerOrg1:7054
    httpOptions:
      verify: true
    tlsCACerts:
      path: $GOPATH/src/github.com/hyperledger/fabric-sdk-go/test/api-server/tls/fabricca/certs/ca_root.pem
      client:
        keyfile: $GOPATH/src/github.com/hyperledger/fabric-sdk-go/test/api-server/tls/fabricca/certs/client/client_fabric_client-key.pem
        certfile: $GOPATH/src/github.com/hyperledger/fabric-sdk-go/test/api-server/tls/fabricca/certs/client/client_fabric_client.pem
    registrar:
      enrollid: admin
      enrollSecret: adminpw
      caName: ca-org1
```

如下为全部修改好后的sdk配置文件示例:

```
name: "global-trade-network"

x-type: "hlfv1"
x-loggingLevel: info

description: "The network to be in if you want to stay in the global trade business"

version: 1.0.0

client:

  organization: aa73c757c9026fb623495d7058ca177f6152bcea

  logging:
    level: info

  peer:
    timeout:
      connection: 10s
      queryResponse: 45s
      executeTxResponse: 120s
  eventService:
    timeout:
      connection: 10s
      registrationResponse: 50s
  orderer:
    timeout:
      connection: 10s
      response: 45s

  cryptoconfig:
    path: /opt/gopath/src/github.com/hyperledger/fabric

  credentialStore:
    path: "/tmp/hfc-kvs"

  cryptoStore:
    path: /tmp/msp

  wallet: wallet-name

  BCCSP:
    security:
      enabled: true
      default:
        provider: "SW"
        hashAlgorithm: "SHA2"
        softVerify: true
        ephemeral: false
        level: 256

  channels:

    tongtai:
      orderers:
        -
          orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9-0.orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9.default.svc.cluster.local

    peers:

      peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30605:
        endorsingPeer: true
        chaincodeQuery: true
        ledgerQuery: true
        eventSource: true
```

```
peer-aa73c757c9026fb623495d7058ca177f6152bcea-1.peer-
aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30606:
  endorsingPeer: true
  chaincodeQuery: true
  ledgerQuery: true
  eventSource: true

chaincodes:
- transaction:1.0

organization:

aa73c757c9026fb623495d7058ca177f6152bcea:
  mspid: aa73c757c9026fb623495d7058ca177f6152bceaMSP

  cryptoPath: /home/paas/peer/msp
  tlsCryptoKeyPath: /home/paas/peer/tls/server.key
  tlsCryptoCertPath: /home/paas/peer/tls/server.crt

peers:

- peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-
aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30605

- peer-aa73c757c9026fb623495d7058ca177f6152bcea-1.peer-
aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30606

certificateAuthorities:
- ca-org1

ordererorg:
  mspID: "2cf8802066c1c5011fd396c54c9126a17c9cfcc9MSP"

  cryptoPath: /home/paas/orderer/msp
orderer-eip: 49.4.81.160
orderers:

orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9-0.orderer-2cf8802066c1c5011fd396c54c91
26a17c9cfcc9.default.svc.cluster.local:
  url: grpcs://49.4.81.160:30805

  grpcOptions:
    ssl-target-name-override:
orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9-0.orderer-2cf8802066c1c5011fd396c54c91
26a17c9cfcc9.default.svc.cluster.local
    grpc-max-send-message-length: 15

  tlsCACerts:
    path: /home/paas/orderer/msp/tlscacerts/
  tlsca.2cf8802066c1c5011fd396c54c9126a17c9cfcc9-cert.pem

peers:

peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-
aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30605:

  url: grpcs://49.4.81.160:30605

  grpcOptions:
    ssl-target-name-override: peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-
aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local
    grpc.http2.keepalive_time: 15

  tlsCACerts:
    path: /home/paas/peer/msp/tlscacerts/tlsca.aa73c757c9026fb623495d7058ca177f6152bcea-
```

```
cert.pem

peer-aa73c757c9026fb623495d7058ca177f6152bcea-1.peer-
aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30606:

url: grpc://49.4.81.160:30606

grpcOptions:
  ssl-target-name-override: peer-aa73c757c9026fb623495d7058ca177f6152bcea-1.peer-
aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local
  grpc.http2.keepalive_time: 15

tlsCACerts:
  path: /home/paas/peer/msp/tlscacerts/tlsca.aa73c757c9026fb623495d7058ca177f6152bcea-
cert.pem
```

5. 修改Dockerfile文件：将yaml文件名称修改为下载的sdk文件解压出来的yaml名称。

```
[root@cluster-bcs-064s-zeb9 cj]# cat Dockerfile
FROM euleros:2.2.5
COPY test-sdk-config.yaml /home/paas
COPY peer /home/paas/peer
COPY openssl /home/paas/openssl
COPY orderer /home/paas/orderer
COPY appdemo /home/paas
[root@cluster-bcs-064s-zeb9 cj]#
```

6. 在“/home/paas/cj/”路径下，执行如下命令将所有文件打包成镜像（请自行安装docker）。

```
docker build -t tongtaidemotest:byl1 .
docker save tongtaidemotest:byl1>tongtatest.tar
```

```
total 404140
-rwx----- 1 root root 23189440 Feb 28 22:07 appdemo
-rw----- 1 root root 10237 Feb 28 22:11 bcs-tongtatest-orderer-admin.zip
drwxr-xr-x 2 root root 4096 Jul 19 2019 bin
-rw----- 1 root root 170 Feb 28 22:10 Dockerfile
drwxr-xr-x 2 root root 4096 Feb 28 22:20 openssl
-rw----- 1 root root 7891504 Feb 28 22:08 openssl.tar.gz
drwx----- 4 root root 4096 Feb 28 22:11 orderer
drwx----- 4 root root 4096 Feb 28 22:11 peer
drwxr-xr-x 3 root root 4096 Jul 19 2019 pkg
-rw----- 1 root root 27261385 Feb 28 22:09 sdk1.11.5.tar.gz
drwxr-xr-x 3 root root 4096 Jul 19 2019 src
-rw----- 1 root root 10048 Feb 28 22:11 testp-admin.zip
-rw----- 1 root root 3883 Feb 28 22:15 test-sdk-config.yaml
-rw----- 1 root root 355431424 Feb 28 22:22 tongtatest.tar
```

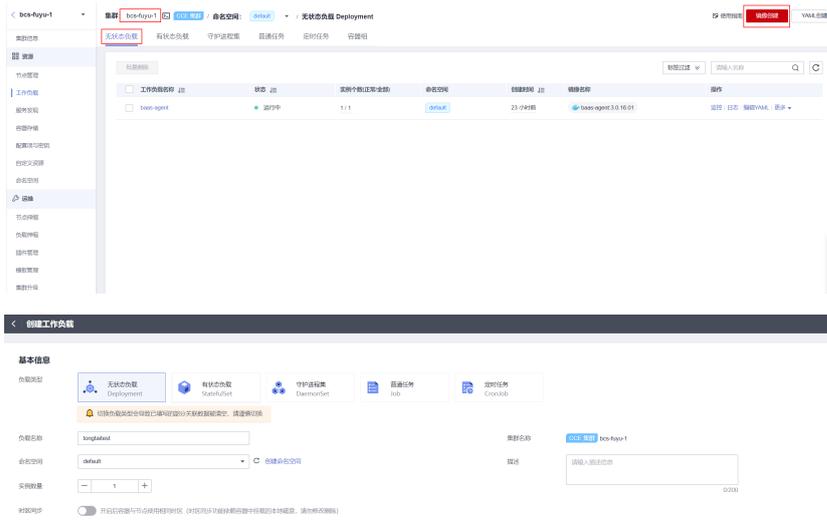
📖 说明

请确保本地有euleros: 2.2.5镜像，否则将会打包失败。

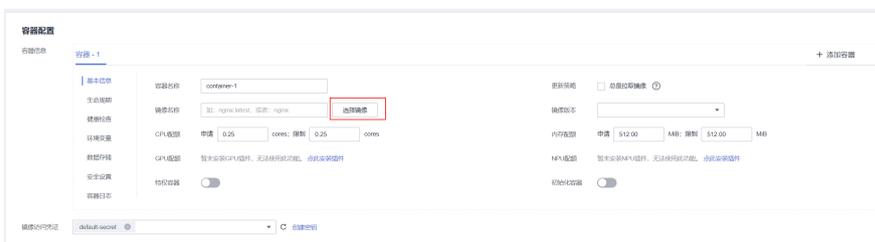
```
[root@cluster-bcs-064s-zeb9 ~]# docker images | grep euleros
swr.cn-north-1.myhuaweicloud.com/op_svc_bcs/euleroswithnodejs
3.0.5 f22d1f510c4e 3 months ago 999MB
swr.cn-north-1.myhuaweicloud.com/op_svc_bcs/euleroswithgolang
3.0.4 c8b11902bfc1 6 months ago 320MB
euleros
2.2.5 b0f6bcd0a2a0 2 years ago 289MB
[root@cluster-bcs-064s-zeb9 ~]#
```

步骤10 上传镜像。

1. 将打包完成的镜像下载到本地，登录华为云，进入云容器引擎页面，在部署了BCS实例对应的集群下，创建无状态工作负载，实例数量选择1个。如果界面布局不一致，请在右上角切换新版cce界面。

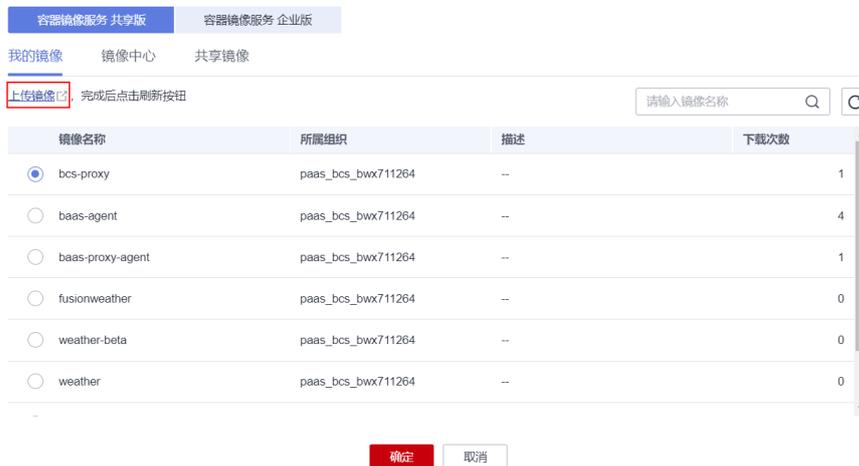


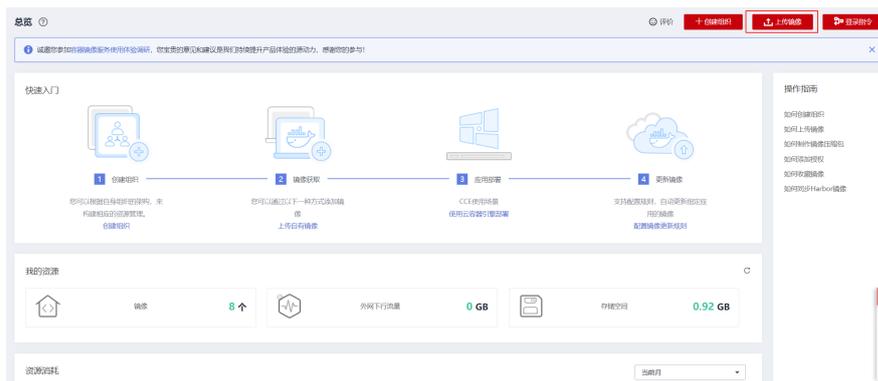
2. 在“容器配置”页签，选择镜像。



3. 单击“上传镜像”，跳转到总览页面，上传镜像。

镜像选择

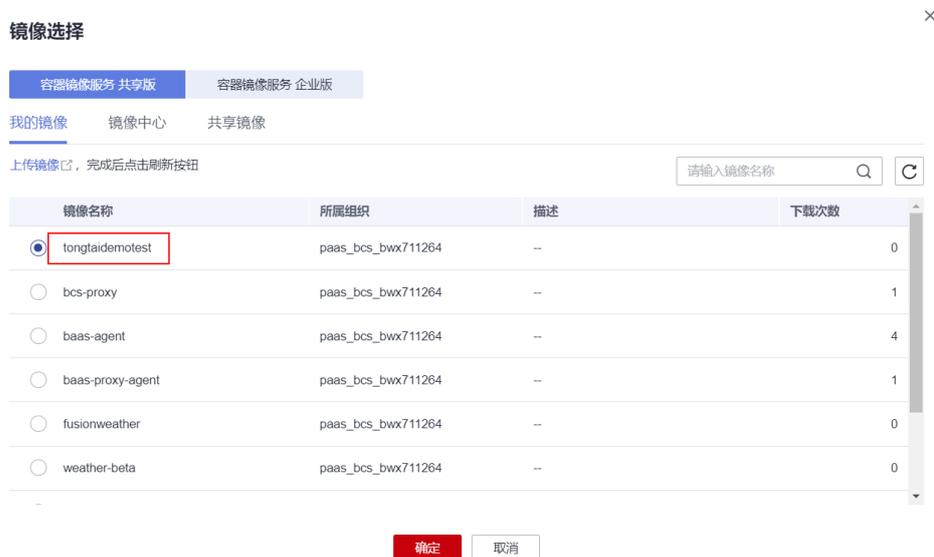




4. 选择打包好的镜像文件，等待上传完毕。



5. 镜像上传完成后，回到镜像选择界面，选择对应镜像，单击“确定”。



6. 在“生命周期”页签，设置启动命令。

- 运行命令：/bin/sh
- 运行参数：
 - c
 - sleep 10000



7. 提交后，无状态工作负载创建成功。



步骤11 交易验证。

1. 登录集群弹性云服务器后台，执行如下命令查看应用容器是否正常。
docker ps -a | grep tongtai | grep container

```
[root@cluster-bcs-064s-zeb9 ~]# docker ps -a | grep tongtai | grep container
39d5db30f097      swr.cn-north-1.myhuaweicloud.com/blockchain-test/tongtaidemotest   "/bin/sh -c 'sleep 1...' 2 hours ago
Up 2 hours
a37-11ea-ac86-fa163e971d0a_0      k8s_container-0_tongtaitest-7764ffbc56-spgtl_default_7800e6cc-5
```

2. 可以看到部署的应用名，进入容器内部，命令如下：
docker exec -it 容器id bash

3. 配置/etc/hosts文件，增加order节点和peer节点的域名映射。

在下载的sdk.yaml文件中查询orderer和peer的域名信息，将ip+orderer域名和ip+peer域名补充在/etc/hosts文件的最后，如下所示：

```
[root@tongtaitest-7764ffbc56-spgtl paas]# cat /etc/hosts
# Kubernetes-managed hosts file.
127.0.0.1      localhost
::1           localhost ip6-localhost ip6-loopback
fe00::0      ip6-localnet
fe00::0      ip6-mcastprefix
fe00::1      ip6-allnodes
fe00::2      ip6-allrouters
172.16.0.25   tongtaitest-7764ffbc56-spgtl
172.16.0.19   orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9-0.orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9.default.svc.cluster.local
172.16.0.20   peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local
172.16.0.21   peer-aa73c757c9026fb623495d7058ca177f6152bcea-1.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local
```

```
x.x.x.x
orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9-0.orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9.default.svc.cluster.local
x.x.x.x peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local
x.x.x.x peer-aa73c757c9026fb623495d7058ca177f6152bcea-1.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local
```

4. 执行命令配置环境变量，并查看注册命令。
export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:/home/paas/openssl
cd /home/paas
./appdemo register -h

📖 说明

- 如果后续同态demo交易中登出容器，再次登录后需重新执行配置环境变量命令。
- 其中./appdemo register -h是用来查看注册命令参数。

```
[root@tongtaitest-7764ffbc56-spgtl paas]# export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/paas/openssl
[root@tongtaitest-7764ffbc56-spgtl paas]# cd /home/paas/
[root@tongtaitest-7764ffbc56-spgtl paas]# ./appdemo register --h
2020-02-28 16:39:40.379 UTC [AESUtil] SetupSysParameters -> INFO 001 invalid setting, AHE encrypt strength is set to default (3072)
Error: unknown flag: --h
Usage:
  appdemo register [flags]

Flags:
  -c, --channel string          channel id (default "mychannel")
  -c, --config string          configuration file path (default "./config_test.yaml")
  -s, --encryptstrength string encrypt strength
  -I, --idcc string            identity chaincode name (default "IDChaincode")
  -i, --initbalance string     init initbalance
  -o, --orgid string           organization id (default "org1")
  -p, --protectpwd string      protect pwd
  -T, --txcc string            transaction chaincode name (default "TxChaincode")
  -u, --userid string          user id

unknown flag: --h
[root@tongtaitest-7764ffbc56-spgtl paas]#
```

----结束

示例 1：注册账户

步骤1 执行如下命令注册B账户，账户金额为100。

```
./appdemo register -u B -p XXXXXX -i 100 -c ./test-sdk-config.yaml -C tongtai -I idchaincode -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
```

📖 说明

-u: 为注册的用户名B、-p: 为B用户密码XXXXXX，密码必须包含大写、小写、数字、特殊字符中的至少两种，-c: 为sdk配置文件名，-C: 为安装链代码的通道名，-I: 为安装示例链代码IDChaincode的实际安装链代码名，-T: 为安装示例链代码Transaction的实际安装链代码名，-o: 为peer节点组织的ID，可在通道管理界面查询。参数下同。

```
[root@tongtaitest-7764ffbc56-spgtl paas]# ./appdemo register -u B -p tongtaitestB -i 100 -c ./test-sdk-config.yaml -C tongtai -I idchaincode -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
2020-02-28 16:22:54.840 UTC [AESUtil] SetupSysParameters -> INFO 001 invalid setting, AHE encrypt strength is set to default (3072)
userid: B
protectpwd: tongtaitestB
initbalance: 100
idcc: idchaincode
txcc: transaction
channelid: tongtai
orgid: aa73c757c9026fb623495d7058ca177f6152bcea
conf: ./test-sdk-config.yaml
encryptstrength:
2020-02-28 16:22:54.844 UTC [AESUtil] SetupSysParameters -> INFO 002 invalid setting, AHE encrypt strength is set to default (3072)
[fab:sdk/fab] 2020/02/28 16:22:54 UTC - fab.(*EndpointConfig).loadPrivateKeyFromConfig -> WARN private key was not encrypted, please consider encrypt your private key
[fab:sdk/fab] 2020/02/28 16:22:54 UTC - fab.detectDeprecatedNetworkConfig -> WARN Getting orderers from endpoint config channels.orderer is deprecated, use entity matchers to override orderer configuration
[fab:sdk/fab] 2020/02/28 16:22:54 UTC - fab.detectDeprecatedNetworkConfig -> WARN visit https://github.com/hyperledger/fabric-sdk-go/blob/master/test/fixtures/config/overrides/local_entity_matchers.yaml for samples
Invoke Chaincode successfully
Register pk success: b22edf18d64f57954640c8f3f6cf67d9401f262daead588ddf8178ba0c64d5b
Invoke Chaincode successfully
init balance successfully: b22edf18d64f57954640c8f3f6cf67d9401f262daead588ddf8178ba0c64d5b
```

返回值为一串加密地址，示例：

```
b22edf18d64f57954640c8f3f6cf67d9401f262daead588ddf8178ba0c64d5b
```

步骤2 以相同的方法注册A账户，账户金额为200。

```
./appdemo register -u A -p XXXXXX -i 200 -c ./test-sdk-config.yaml -C tongtai -I idchaincode -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
```

```
[root@tongtaitest-7764ffbc56-spgtl paas]# ./appdemo register -u A -p tongtaitestA -i 200 -c ./test-sdk-config.yaml -C t
ongtai -I idchaincode -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
2020-02-28 16:25:45.389 UTC [AESUtil] SetupSysParameters -> INFO 001 invalid setting, AHE encrypt strength is set to def
ault (3072)
userid: A
protectwd: tongtaitestA
initbalance: 200
idcc: idchaincode
txcc: transaction
channelid: tongtai
orgid: aa73c757c9026fb623495d7058ca177f6152bcea
conf: ./test-sdk-config.yaml
encryptstrength:
2020-02-28 16:25:45.394 UTC [AESUtil] SetupSysParameters -> INFO 002 invalid setting, AHE encrypt strength is set to def
ault (3072)
[fabsdk/fab] 2020/02/28 16:25:45 UTC - fab.(*EndpointConfig).loadPrivateKeyFromConfig -> WARN private key was not encry
pted, please consider encrypt your private key
[fabsdk/fab] 2020/02/28 16:25:45 UTC - fab.detectDeprecatedNetworkConfig -> WARN Getting orderers from endpoint config
channels.orderer is deprecated, use entity matchers to override orderer configuration
[fabsdk/fab] 2020/02/28 16:25:45 UTC - fab.detectDeprecatedNetworkConfig -> WARN visit https://github.com/hyperledger/f
abric-sdk-go/blob/master/test/fixtures/config/overrides/local_entity_matchers.yaml for samples
Invoke Chaincode successfully
Register pk success: 2efc4639bc281060ce013dfea33a47b647b6f4a20103a6321c33d67d5e6da24f
Invoke Chaincode successfully
init balance successfully: 2efc4639bc281060ce013dfea33a47b647b6f4a20103a6321c33d67d5e6da24f
[root@tongtaitest-7764ffbc56-spgtl paas]#
```

返回值为一串加密地址，示例：

```
2efc4639bc281060ce013dfea33a47b647b6f4a20103a6321c33dxxxxx
```

----结束

示例 2：A 向 B 转账

步骤1 执行如下命令，A向B转账，金额为10。

```
./appdemo transaction -u A -p XXXXXXXX -b
b22edf18d64f57954640c8f3f6cf67d9401f262daead588ddf8178xxxx -t 10 -c ./test-sdk-config.yaml -C
tongtai -I idchaincode -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
```

说明

其中-b后的参数为接收方的地址。示例中为注册B用户时的返回值，即B用户注册数据的地址信息。

```
[root@tongtaitest-7764ffbc56-spgtl paas]# ./appdemo transaction -u A -p tongtaitestA -b b22edf18d64f57954640c8f3f6cf67d9
401f262daead588ddf8178ba0c64d5b -t 10 -c ./test-sdk-config.yaml -C tongtai -I idchaincode -T transaction -o aa73c757c90
26fb623495d7058ca177f6152bcea
2020-02-28 16:29:07.468 UTC [AESUtil] SetupSysParameters -> INFO 001 invalid setting, AHE encrypt strength is set to def
ault (3072)
userid: A
protectwd: tongtaitestA
AddrB: b22edf18d64f57954640c8f3f6cf67d9401f262daead588ddf8178ba0c64d5b
Tx: 10
idcc: idchaincode
txcc: transaction
channel: tongtai
orgid: aa73c757c9026fb623495d7058ca177f6152bcea
conf: ./test-sdk-config.yaml
encryptstrength:
2020-02-28 16:29:07.471 UTC [AESUtil] SetupSysParameters -> INFO 002 invalid setting, AHE encrypt strength is set to def
ault (3072)
[fabsdk/fab] 2020/02/28 16:29:07 UTC - fab.(*EndpointConfig).loadPrivateKeyFromConfig -> WARN private key was not encry
pted, please consider encrypt your private key
[fabsdk/fab] 2020/02/28 16:29:07 UTC - fab.detectDeprecatedNetworkConfig -> WARN Getting orderers from endpoint config
channels.orderer is deprecated, use entity matchers to override orderer configuration
[fabsdk/fab] 2020/02/28 16:29:07 UTC - fab.detectDeprecatedNetworkConfig -> WARN visit https://github.com/hyperledger/f
abric-sdk-go/blob/master/test/fixtures/config/overrides/local_entity_matchers.yaml for samples
get A's balance successfully
Get B's ID successfully
2020-02-28 16:29:13.068 UTC [psw] PrepareTxInfo -> INFO 003 Prepare TxInfo successfully
prepare transaction information successfully
Invoke Chaincode successfully
Transfer success: 2efc4639bc281060ce013dfea33a47b647b6f4a20103a6321c33d67d5e6da24f
[root@tongtaitest-7764ffbc56-spgtl paas]#
```

返回值为A账户的地址：

```
2efc4639bc281060ce013dfea33a47b647b6f4a20103a6321c33d67d5xxxx
```

----结束

示例 3：查询账户余额

步骤1 执行如下命令查询A账户的余额，返回值为A账户余额。

```
./appdemo querybalance -p XXXXXX -u A -c ./test-sdk-config.yaml -C tongtai -I idchaincode -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
```

```
[root@tongtaitest-7764ffbc56-sqql paas]# ./appdemo querybalance -p tongtaitestA -u A -c ./test-sdk-config.yaml -C tongtai -I idchaincode -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
2020-02-28 16:31:39.906 UTC [AESUtil] SetupSysParameters -> INFO 001 invalid setting, AHE encrypt strength is set to default (3072)
userid: A
protectwd: tongtaitestA
idcc: idchaincode
txcc: transaction
channel: tongtai
orgid: aa73c757c9026fb623495d7058ca177f6152bcea
conf: ./test-sdk-config.yaml
encryptstrength:
2020-02-28 16:31:39.909 UTC [AESUtil] SetupSysParameters -> INFO 002 invalid setting, AHE encrypt strength is set to default (3072)
[fabsdk/fab] 2020/02/28 16:31:39 UTC - fab.(*EndpointConfig).loadPrivateKeyFromConfig -> WARN private key was not encrypted, please consider encrypt your private key
[fabsdk/fab] 2020/02/28 16:31:39 UTC - fab.detectDeprecatedNetworkConfig -> WARN Getting orderers from endpoint config channels, orderer is deprecated, use entity matchers to override orderer configuration
[fabsdk/fab] 2020/02/28 16:31:39 UTC - fab.detectDeprecatedNetworkConfig -> WARN visit https://github.com/hyperledger/fabric-sdk-go/blob/master/test/fixtures/config/overrides/local_entity_matchers.yaml for samples
query balance
current balance:190
[root@tongtaitest-7764ffbc56-sqql paas]#
```

步骤2 执行如下名查询B账户的余额，返回值为B账户余额。

```
./appdemo querybalance -p XXXXXX -u B -c ./test-sdk-config.yaml -C tongtai -I idchaincode -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
```

```
[root@tongtaitest-7764ffbc56-sqql paas]# ./appdemo querybalance -p tongtaitestB -u B -c ./test-sdk-config.yaml -C tongtai -I idchaincode -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
2020-02-28 16:32:20.497 UTC [AESUtil] SetupSysParameters -> INFO 001 invalid setting, AHE encrypt strength is set to default (3072)
userid: B
protectwd: tongtaitestB
idcc: idchaincode
txcc: transaction
channel: tongtai
orgid: aa73c757c9026fb623495d7058ca177f6152bcea
conf: ./test-sdk-config.yaml
encryptstrength:
2020-02-28 16:32:20.501 UTC [AESUtil] SetupSysParameters -> INFO 002 invalid setting, AHE encrypt strength is set to default (3072)
[fabsdk/fab] 2020/02/28 16:32:20 UTC - fab.(*EndpointConfig).loadPrivateKeyFromConfig -> WARN private key was not encrypted, please consider encrypt your private key
[fabsdk/fab] 2020/02/28 16:32:20 UTC - fab.detectDeprecatedNetworkConfig -> WARN Getting orderers from endpoint config channels, orderer is deprecated, use entity matchers to override orderer configuration
[fabsdk/fab] 2020/02/28 16:32:20 UTC - fab.detectDeprecatedNetworkConfig -> WARN visit https://github.com/hyperledger/fabric-sdk-go/blob/master/test/fixtures/config/overrides/local_entity_matchers.yaml for samples
query balance
current balance:110
[root@tongtaitest-7764ffbc56-sqql paas]#
```

----结束

示例 4：测试同态加法

步骤1 执行如下命令测试同态加法。

```
./appdemo homoadd -c ./test-sdk-config.yaml -a 30 -b 60 -C tongtai -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
```

```
2020-02-28 17:45:48.909 UTC [AESUtil] SetupSysParameters -> INFO 001 invalid setting, AHE encrypt strength is set to default (3072)
num1: 34
num2: 56
txcc: transaction
channel: tongtai
orgid: aa73c757c9026fb623495d7058ca177f6152bcea
conf: ./test-sdk-config.yaml
encryptstrength:
2020-02-28 17:45:48.912 UTC [AESUtil] SetupSysParameters -> INFO 002 invalid setting, AHE encrypt strength is set to default (3072)
encrypted num1: {"G2R0":48230269452180293674461105958232289867956933061923755179 ...
encrypted num2: {"G2R0":316287440742315751669064081711664387056723252399956354918 ...
[fabsdk/fab] 2020/02/28 17:45:50 UTC - fab.(*EndpointConfig).loadPrivateKeyFromConfig -> WARN private key was not encrypted, please consider encrypt your private key
[fabsdk/fab] 2020/02/28 17:45:50 UTC - fab.detectDeprecatedNetworkConfig -> WARN Getting orderers from endpoint config channels, orderer is deprecated, use entity matchers to override orderer configuration
[fabsdk/fab] 2020/02/28 17:45:50 UTC - fab.detectDeprecatedNetworkConfig -> WARN visit https://github.com/hyperledger/fabric-sdk-go/blob/master/test/fixtures/config/overrides/local_entity_matchers.yaml for samples
invoke homoadd
Invoke Chaincode successfully
encrypted homoaddRes: {"G2R0":22920891215311627641514042506045495425013933055021435832 ...
decrypted homoadd res: 90
success
[root@tongtaitest-7764ffbc56-sqql paas]#
```

📖 说明

其中-a和-b后的参数为进行同态加的两个数字。

----结束

示例 5：测试同态乘法

步骤1 执行如下命令测试同态乘法。

```
./appdemo homomulti -c ./test-sdk-config.yaml -a 100 -b 5 -C tongtai -T transaction -o  
aa73c757c9026fb623495d7058ca177f6152bcea
```

```
[root@tongtaitest-777dd875bd-m485x paas]# ./appdemo homomulti -a 100 -b 5 -c ./conf.yaml -C  
2022-07-04 13:35:47.642 UTC [AESUtil] SetupSysParameters -> INFO 001 invalid setting, AHE enc  
num1: 100  
times: 5  
txcc: transaction  
channel: channel  
orgid: ale954cd9c712864130acaf9c63906d06e15877f  
conf: ./conf.yaml  
encryptstrength:  
2022-07-04 13:35:47.647 UTC [AESUtil] SetupSysParameters -> INFO 002 invalid setting, AHE enc  
encrypted num1: {"G2R0":39359997139242753333582037424879571668866411229463383855 ...  
times of multiplication: 5 ...  
[fabsdk/fab] 2022/07/04 13:35:48 UTC - fab.(*EndpointConfig).loadPrivateKeyFromConfig -> WARN  
[fabsdk/fab] 2022/07/04 13:35:48 UTC - fab.detectDeprecatedNetworkConfig -> WARN Getting ord  
[fabsdk/fab] 2022/07/04 13:35:48 UTC - fab.detectDeprecatedNetworkConfig -> WARN visit https  
invoke homomulti  
Invoke Chaincode successfully  
encrypted homomultiRes: {"G2R0":59047406159742519949166743088090166851246728403868919402 ...  
decrypted homomultiRes: 500  
success
```

📖 说明

其中-a和-b后的参数为进行同态乘的乘数和被乘数，其中-a后的参数会被加密，-b后的参数是明文。

----结束

6.3 区块链应用低代码开发功能

6.3.1 概述

目前区块链智能合约编写门槛高，开发人员需要掌握区块链基础知识以及对底层区块链支持的合约语言有所了解，同时编写出高效和安全的智能合约需要拥有丰富的经验；并且在处理业务合约时，操作人员需实时监控链上交易以获取业务流程进展，增加了业务管理流程和编程人员的开发工作量。

本功能旨在提升区块链的易用性，消除开发人员对区块链知识和智能合约编程的依赖，同时创新性地以业务为维度提供流程管理功能，让区块链操作人员更多地关注于业务本身，提供简单便捷、安全可靠的区块链业务开发和流程管理功能。

图 6-1 系统逻辑



说明

若需要使用该功能，请联系技术支持工程师。

6.3.2 低代码合约开发

传统的智能合约开发需要开发人员使用底层区块链支持的合约语言进行相关业务开发和测试，开发人员不仅要求具备一定的编程能力，还需具备区块链等相关知识以保障合约的安全性。华为区块链提供低代码开发能力，开发人员仅需要根据实际业务画出业务流程建模符号（Business Process Modeling Notation，简称BPMN）和少量辅助功能代码的开发，即可完成安全可靠的智能合约编写。

通过设计以下模块实现低代码开发和部署：

- **BPMN业务图绘制**
BCS提供的BPMN绘制页面简单易用，用户通过拖拽方式将开始事件、活动、网关、连接对象、结束事件、泳道等组合起来，通过实际业务逻辑绘制定制化BPMN业务图。其中，用户仅需编写少量业务需要的代码和查询功能代码即可完成智能合约的制作。通过绘制页面可实时验证BPMN的有效性，验证无误后单击构建，生成智能合约。
- **智能合约部署**
Package Management界面用于智能合约的管理，用户登录业务管理页面后跳转到该界面，可查看和管理已生成的智能合约。通过对生成的智能合约选择背书策略、安装合约的组织等配置，完成合约的安装和实例化，智能合约最终运行在背书节点中的一个Docker容器内。
- **智能合约触发**
实例化后的智能合约，可以通过外部条件来触发合约执行过程，支持事件触发和交易触发的方式，两种模式均会触发背书节点进行一致性共识，避免恶意节点作恶。
- **智能合约变更**
业务更替升级后，用户绘制新的BPMN业务图并改变版本号，构建新版本的智能合约，重新部署合约即完成智能合约的变更。为保证安全性，合约的变更需要一定数量的背书节点达成共识后才能完成。

6.3.3 业务流程管理

常见的区块链业务开发管理需要操作人员监控链上交易数据来掌握该笔业务的流程走向，即使使用区块链浏览器也需操作人员对浏览器中的数据筛选过滤，无疑增加了操作人员的工作量以及需要操作人员无一遗漏的筛选出正确交易才能开展下一步业务流程。

BCS提供以业务为维度的管理流程，操作人员通过界面更加直观地查看该笔业务的当前进展和历史交易数据。并且针对用户管理，提供了以角色和群组的细粒度访问授权（Hyperledger Fabric增强版目前不支持细粒度，功能正在开发中），不同角色或群组的用户授予不同的权限，例如仓库管理员仅可处理货物的发放、货物收件人员仅可确认货物的签收。该功能使用户更多的精力放在处理业务，丰富了区块链上层业务和人员管理，极大地增加了区块链的易用性。

- 用户权限管理

智能合约的发起者作为该合约的管理员，可通过用户管理页面User Management进行用户权限管理。首先根据实际业务设置角色，例如买家、卖家和仓库管理员等角色。然后设置群组，每个群组可包含多个或单个角色。最后添加用户，管理员为其设置用户名和登录密码，并选择对应的群组。

- 业务监控

用户可通过用户名和密码登录到业务监控页面，查看当前业务流程和业务历史交易数据，通过图形化呈现业务进展和下一步流程走向。并且通过状态标识，可直观的查看当前业务是否完成。

- 业务处理

业务处理包括接口调用业务处理、自动业务处理和人工处理三种业务处理方式。接口调用业务处理指用户通过调用合约SDK发起交易进行业务处理；自动业务处理指根据业务逻辑判断，满足一定要求后自动发起交易进行业务处理；人工处理指根据用户角色，不同用户拥有特定的权限执行相应的人工处理业务，以角色为维度进行细粒度访问授权（Hyperledger Fabric增强版目前不支持细粒度，功能正在开发中）。

6.3.4 使用说明

介绍低代码开发功能的使用说明，如需使用请联系技术支持工程师获取工具包BCS-BPMN.zip和License。

步骤1 部署后端服务。

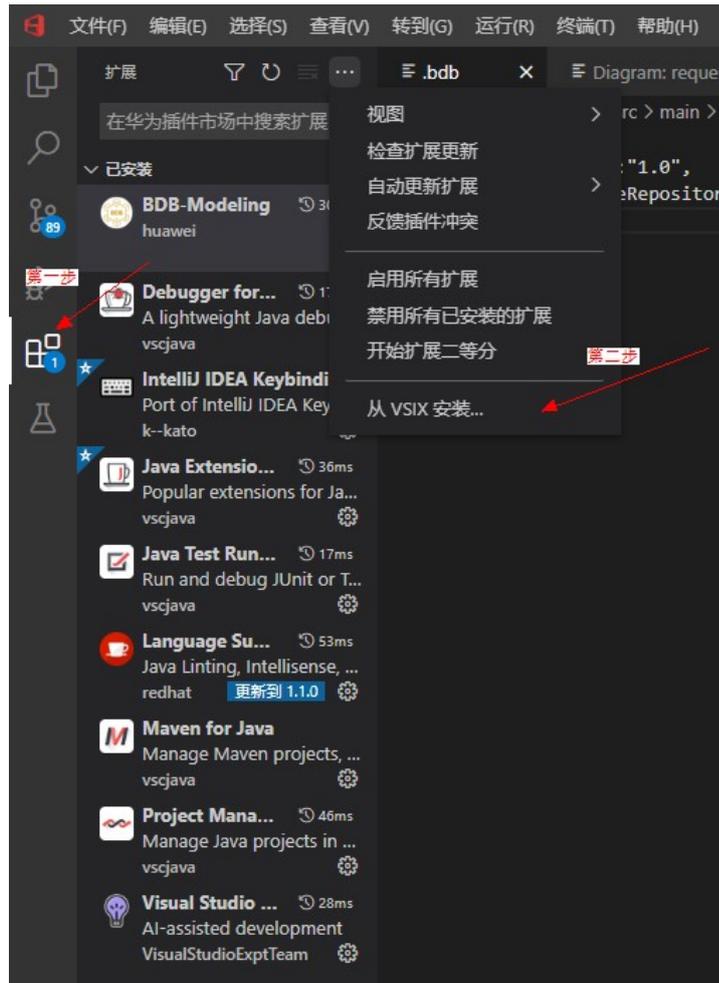
1. 登录云容器引擎CCE控制台。
2. 创建CCE集群，购买Ubuntu系统CPU16核内存32G机器。
3. 选择集群节点，绑定弹性公网IP并设置节点安全组规则，添加入方向规则TCP 9096端口，以及开放Kubernetes服务端口，设置为1-32767。
4. 将License放入虚拟机目录/下，解压工具包BCS-BPMN.zip放置/root目录，运行一键部署脚本/root/BCS-BPMN/.build_config/one_step_deploy.sh。

步骤2 开发环境下解压工具包。

Windows开发环境下解压工具包BCS-BPMN.zip。

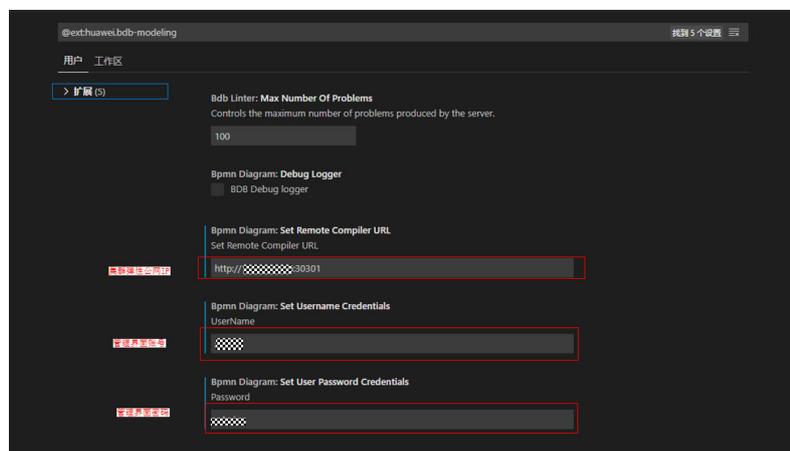
步骤3 安装工具包。

打开VScode，单击左侧扩展框 -> 选择扩展框右上角 -> 单击从VSIX安装 -> 选择BCS-BPMN\extension\BDB-Modeling-0.1.0.vsix。



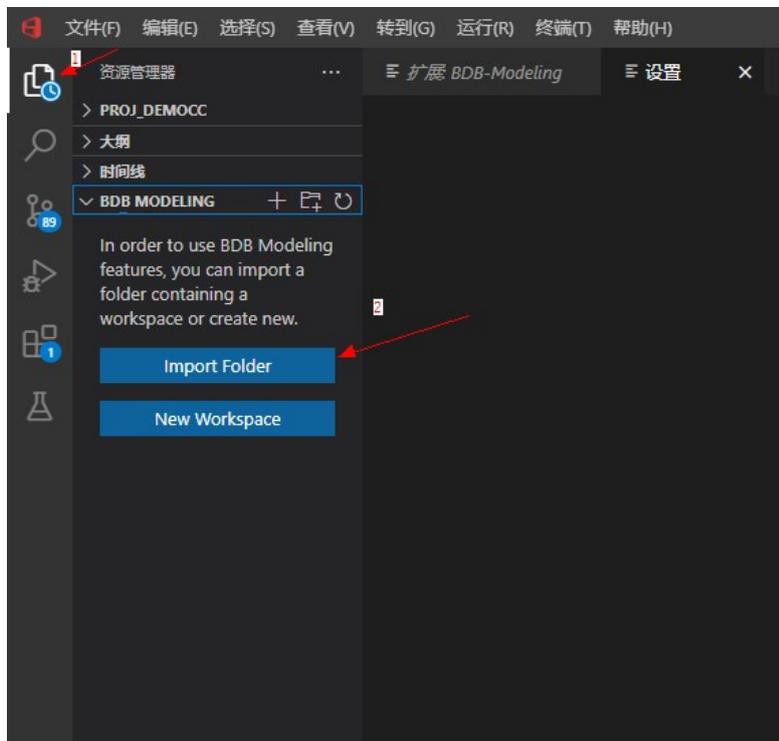
步骤4 配置插件信息。

单击插件扩展设置，修改远端编译器地址http://{租户机器弹性IP}:30301、管理员账号和密码。



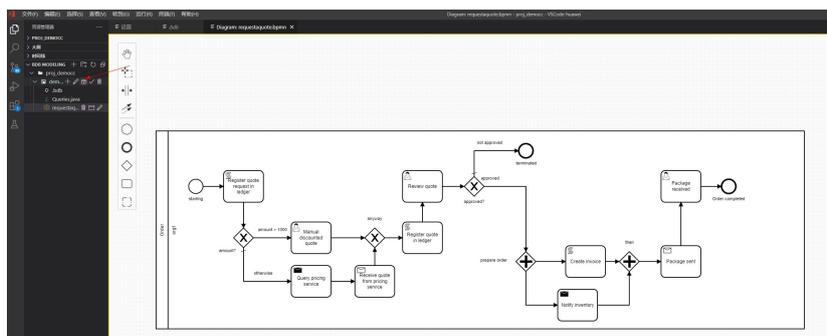
步骤5 导入示例demo。

单击资源管理器 -> 选择BDB MODELING -> Import Folder -> 选择BCS-BPMN\demo\projects\democc\modeller-workspace\proj_democc。



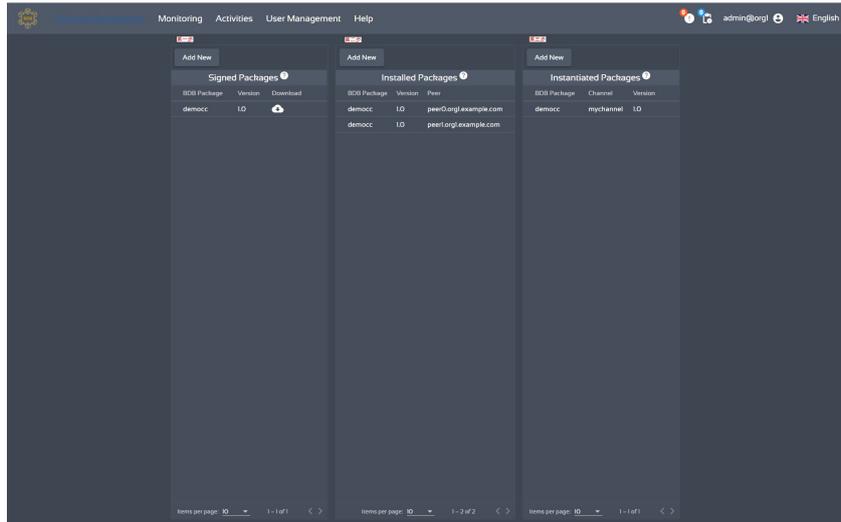
步骤6 编译bpmn，生成合约。

用户可根据示例demo进行修改，生成符合自身业务的bpmn图，然后单击编译合约，等待若干秒后右下角显示编译成功。



步骤7 安装和实例化合约。

登录管理界面<http://{{租户机器弹性IP}}:30300>，单击Package Management，依次签名合约、安装合约以及实例化合约。

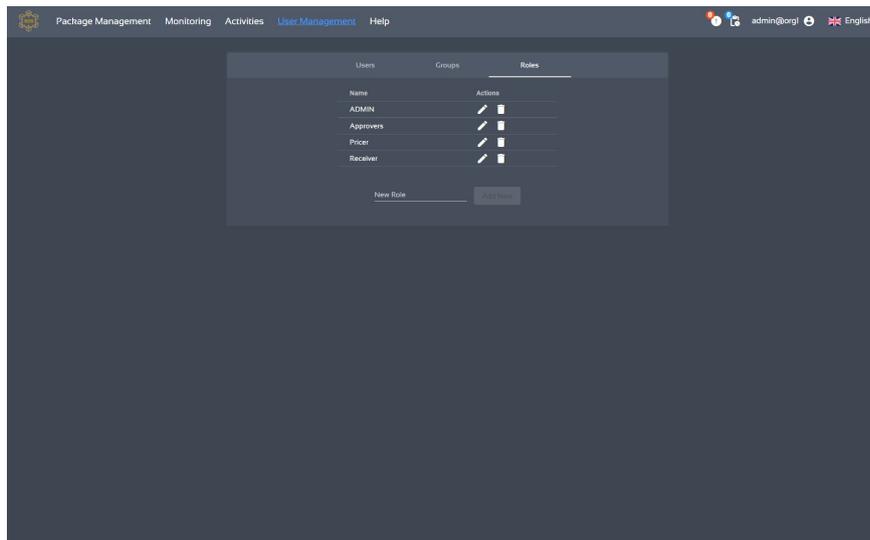


步骤8 设置角色、群组和用户。

管理界面User Management可设置角色（用于细粒度授权人工任务）、群组（每个群组可包含多个角色）以及用户（每个用户可加入多个群组）。

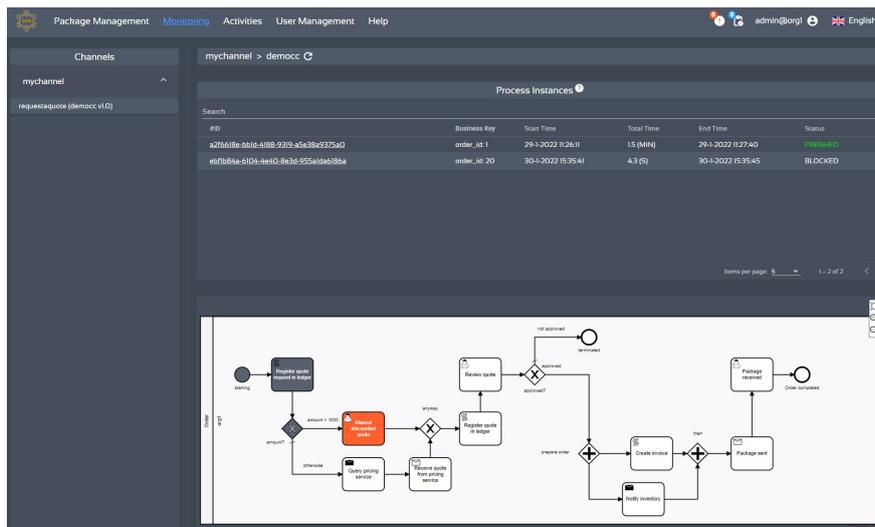
📖 说明

Hyperledger Fabric增强版目前不支持细粒度，功能正在开发中。



步骤9 业务监控界面。

该界面通过图形化呈现业务进展和下一步流程走向，并且通过状态标识，可直观的查看当前业务是否完成。



步骤10 接口服务。

目前支持业务应用通过项目生成的JAVA SDK调用智能合约，后续为了方便用户快速接入区块链系统，将提供快速构建Restful API的方式接入，用户可以在SDK和 Restful API之间选择适合的方式调用智能合约接入区块链系统。

----结束

6.4 错误码

当您调用API时，如果遇到“APIGW”开头的错误码，请参见[API网关错误码](#)进行处理。

状态码	错误码	错误信息	描述	处理措施
100	BCS.4006012	Invalid channel name.	channel 名称不合法	请提供合法的channel 名称
400	BCS.2001001	BCS service modified.	操作成功，BCS服务实例变更成功	可以正常使用服务
400	BCS.4001001	Operation failed. The BCS service is being created.	操作失败。BCS服务正在创建中	等待服务创建完成后操作
400	BCS.4001002	Operation failed. The BCS service is being upgraded.	操作失败。BCS服务正在升级中	等待升级结束即可。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001003	Operation failed. The BCS service is being deleted.	操作失败。BCS服务正在删除中	等待删除结束即可。
400	BCS.4001006	Failed to delete the SFS file system because the CCE API or the cluster status is abnormal.	删除失败，删除SFS网盘相关资源失败，可能原因为CCE网盘操作接口异常或者集群状态异常导致	手动将相应网盘删除即可。具体步骤如下：登录CCE服务控制台，进入存储管理界面，选择“资源管理”->“存储管理”，找到服务对应集群下的网络存储；如果使用的是“文件存储卷”，则直接单击存储卷后面的“删除选项”；如果使用的是“极速文件存储卷”，则单击“解关联”，然后登录弹性文件服务界面，依次将存储卷删除。
400	BCS.4001007	Services billed in the yearly/monthly mode cannot be deleted.	包周期服务不允许删除。	包周期服务不允许删除，请到界面上进行退订。
400	BCS.4001008	Operation failed. Bind an EIP to at least one node in the cluster and try again.	操作失败，集群内至少有一个节点需要绑定弹性IP，请绑定后重新创建	集群内至少有一个节点需要绑定弹性IP，请绑定后重新操作。具体步骤：登录CCE服务控制台，进入集群管理界面，选择“资源管理”中的“节点管理”，过滤服务对应的集群的节点，对任意一个节点做如下操作：单击节点名称，会跳转到ECS界面；在ECS界面单击节点名称后，会跳转到详情界面，切换至“弹性公网IP”页签；单击“绑定弹性公网IP”。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001009	Operation failed. Bind an EIP to each node in the cluster and try again.	操作失败，集群内有节点未绑定弹性IP，请为每个节点绑定后重新创建	给集群每个节点绑定EIP，可参考CCE服务的用户指南，然后重新操作。具体步骤：登录CCE服务控制台，进入集群管理界面，选择“资源管理”中的“节点管理”，过滤服务对应的集群的节点，找出没有绑定弹性IP的节点，并依次对其中每一个节点做如下操作：单击节点名称，会跳转到ECS界面；在ECS界面单击节点名称后，会跳转到详情界面，切换至“弹性公网IP”页签；单击“绑定弹性公网IP”。
400	BCS.4001010	Failed to download the BCS certificate because the etcd connection is abnormal.	下载BCS证书失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001011	Operation failed. The service does not exist or has been deleted. Refresh the page.	操作失败，服务不存在或已被删除，请刷新	等待片刻，重新刷新页面即可。
400	BCS.4001015	Failed to obtain the peer information because the etcd connection is abnormal.	获取Peer节点信息失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001018	Operation failed. A service with the same name already exists. Enter another name and try again.	操作失败，服务名称已经被使用，请修改后重新提交	修改服务名称后重新提交。
400	BCS.4001019	Failed to obtain the image version for upgrade because the required image is not in the image repository or the version configuration is incorrect.	获取升级镜像版本失败，升级版本相关镜像不在镜像库，或者版本配置错误	检查服务版本配置，确定升级版本使用镜像已经在镜像库。
400	BCS.4001020	Operation failed. The service status is abnormal or the service is being processed. Try again later.	操作失败，服务状态异常或者服务正在处理中，请稍后重试	请稍后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001021	Failed to add the peer to the channel because the CCE API, cluster status, or etcd connection is abnormal.	添加Peer到通道失败，可能存在CCE接口、集群状态异常或者ETCD连接异常的情况	登录CCE服务控制台，查看CCE服务是否正常；若服务正常则进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务；按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001025	Invalid EIP. The EIP may have been unbound from the cluster. Check the EIP binding of the cluster.	EIP不合法，可能当前EIP已从集群解绑，请检查确认集群EIP的绑定情况	登录CCE服务控制台，选择“资源管理”-》“集群管理”，单击服务对应的集群，进入“节点管理”界面，查看集群节点是否绑定对应的EIP，如果没有则重新绑定EIP之后再次重试该操作。
400	BCS.4001026	Failed to save the EIP to etcd because the etcd connection is abnormal.	保存EIP到ETCD失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001027	Failed to update the EIP information in the consortium member configuration because the CCE API, cluster status, or etcd connection is abnormal.	保存EIP到联盟成员失败，可能存在CCE接口、集群状态异常或者ETCD连接异常的情况	登录CCE服务控制台，查看CCE服务是否足够正常；若服务正常则进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务；按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001030	Operation failed because the etcd connection check failed.	操作失败，ETCD连接检查失败	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001031	Failed to query the member details because the etcd connection is abnormal.	查询成员详细信息失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001032	You do not have the required permission.	您没有操作权限	按照用户指南-服务部署-前提条件章节，重新配置权限后再进行操作。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001033	Failed to delete the member information because the etcd connection is abnormal.	删除成员信息失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001036	Operation failed. The notification does not exist. Refresh the page.	操作失败，通知信息不存在，请刷新查看	稍后刷新查看。
400	BCS.4001040	Failed to decline the invitation for joining a channel. Refresh the page and check the invitation status later.	拒绝加入联盟通道失败。您已同意加入此通道，请勿再次单击并稍后刷新查看通知状态	您已同意加入此通道，请勿再次单击并稍后刷新查看通知状态。
400	BCS.4001042	Failed to query the member information because the etcd connection is abnormal.	查询成员信息失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001043	Failed to modify the member information because the etcd connection is abnormal.	修改成员信息失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001044	Failed to accept the invitation. The acceptance is being processed. Refresh the page and check the invitation status later.	同意加入失败。同意请求正在处理中，请勿再次单击并稍后刷新查看通知状态	稍后刷新查看处理结果。
400	BCS.4001045	Weak password.	输入密码强度不够	请输入安全性较高的密码
400	BCS.4001047	Failed to modify the etcd data because the etcd connection is abnormal.	修改ETCD数据失败，ETCD连接异常	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001048	Failed to change the password because the CCE cluster is abnormal.	修改配置中密码失败，CCE集群异常导致	登录CCE控制台，进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群节点是否正常，若集群节点状态异常，则参考CCE服务的故障处理指导，待集群恢复后进行操作，若集群节点已经被删除则需要重新订购节点并重新创建BCS服务来使用。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001049	Operation failed. The cluster status is abnormal. Check the cluster status and try again.	操作失败，集群状态异常，请确认集群状态并重试	登录CCE控制台，进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群节点是否正常，若集群节点状态异常，则参考CCE服务的故障处理指导，待集群恢复后进行操作，若集群节点已经被删除则需要重新订购节点并重新创建BCS服务来使用。
400	BCS.4001050	Scaling failed because the CCE API or the cluster status is abnormal.	扩缩容失败，CCE服务接口异常或对应集群状态异常	登录CCE控制台，查看CCE服务是够正常；若CCE服务功能正常则进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务使用。
400	BCS.4001051	Scaling failed due to incorrect request parameters. Check whether the number of peers in the organization has reached the limit (5).	扩缩容失败，请求参数不正确，请检查组织内的的peer节点数目是否达到上限（5个）	重试扩容操作，保证每个组织内的peer节点数目不超过上限值（5个）。
400	BCS.4001052	Operation failed because it timed out.	操作失败，等待超时	稍后查看执行结果。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001053	Failed to query the network storage because the CCE API or the cluster status is abnormal.	网络存储查询失败，可能因为CCE服务接口或对应集群出现异常	登录CCE控制台，查看CCE服务是否正常；若CCE服务功能正常则进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务使用。
400	BCS.4001054	Failed to add the peer to the channel because the CCE cluster status or etcd connection is abnormal.	peer加通道失败，可能因为CCE集群异常或者ETCD连接问题	登录CCE控制台，查看CCE服务是否正常；然后进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务；检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001058	Failed to delete the member information because the etcd connection may be abnormal.	删除成员信息失败，可能出现ETCD连接异常情况	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001062	The maximum number of tenants that can be invited has been reached.	已邀请租户数量达到上限	邀请租户达上限后禁止邀请。
400	BCS.4001063	The maximum number of invitation notifications that can be received has been reached.	已接收邀请通知数达上限	检查已收到邀请数是否达到了上限值100，登录BCS服务控制台，进入“通知管理”界面，查看邀请数目并处理邀请通知。
400	BCS.4001067	Failed to delete the notification. The etcd connection may be abnormal.	删除通知失败。可能出现ETCD连接异常情况	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001069	The chaincode is being instantiated. Try again later.	链代码正在实例化，请稍后再试	登录对应服务的链代码管理界面查看链代码实例化状态，待实例化结束后重新操作。
400	BCS.4001078	Failed to query the channel information because the etcd connection may be abnormal.	查询通道信息失败，可能因为ETCD连接异常	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001080	Operation failed because the CCE cluster information failed to be obtained.	操作失败，CCE集群信息获取失败	检查CCE服务状态，登录CCE服务控制台，检查能否正常登录。登录成功后，选择“资源管理”中的“集群管理”，查找服务使用的集群状态是否正常，待恢复正常后重试

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001081	Failed to obtain the invitee's organization because the etcd is abnormal.	获取被邀请组织失败，ETCD异常	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001086	Permission verification failed.	权限校验失败	按照用户指南-服务部署-前提条件章节，重新配置权限后再进行操作。
400	BCS.4001088	Operation failed due to insufficient quota. Increase the quota and try again.	操作失败，服务配额不足，请扩大配额后重试	申请扩大配额。联系管理员，在ManageOne界面的“组织“->“VDC管理“->“配额管理”中修改。
400	BCS.4001089	Deletion failed. Failed to delete the BCS information.	删除失败。BCS信息删除失败	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001090	Operation failed. No node is available in the cluster. Try again later.	操作失败，集群无可节点，请稍后再试	排查节点是出现了异常还是节点被删除了。登录CCE控制台，进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群下是否有节点，如果没有则节点可能被删除了，需添加节点后重新订购服务；若节点出现异常，则参考CCE服务的故障指导尝试恢复集群节点，恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001095	Failed to delete block configurations due to etcd exceptions.	删除失败，ETCD异常导致区块配置删除失败	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001096	Failed to delete the AgentConfig information due to etcd exceptions.	删除失败，ETCD异常导致AgentConfig信息删除失败	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001097	Failed to delete the CCE cluster occupation information due to etcd exceptions.	删除失败，ETCD异常导致CCE集群占用信息删除失败	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001098	Failed to delete the SDK information due to etcd exceptions.	删除失败，ETCD异常导致SDK信息删除失败	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001099	Failed to delete the channel information due to etcd exceptions.	删除失败，ETCD异常导致通道信息删除失败	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001100	Failed to delete the EIP information due to etcd exceptions.	删除失败，ETCD异常EIP信息删除失败	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001101	Failed to delete the port information due to etcd exceptions.	删除失败，ETCD异常导致Port信息删除失败	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001102	Failed to delete the ipmapping information due to etcd exceptions.	删除失败，ETCD异常导致ipmapping删除失败	检查ETCD状态，若ETCD异常，则按照故障恢复指导书中ETCD异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001104	Failed to delete the deployment template because the CCE cluster status may be abnormal.	删除失败，部署模版删除失败，可能出现CCE集群状态异常	CCE界面检查对应集群状态是否正常。登录CCE控制台，进入集群管理界面，选择“资源管理”中的“集群管理”，查找服务使用的集群状态是否正常，若出现集群状态异常的情况，请参考CCE的用户指南尝试恢复集群，待集群状态正常后再重新操作。
400	BCS.4001105	Deleting the deployment template timed out because the CCE cluster status may be abnormal.	删除失败，部署模版删除超时，可能出现CCE的集群状态异常	CCE界面检查对应集群状态是否正常。登录CCE控制台，进入集群管理界面，选择“资源管理”中的“集群管理”，查找服务使用的集群状态是否正常，若出现集群状态异常的情况，请参考CCE的用户指南尝试恢复集群，待集群状态正常后再重新操作。
400	BCS.4001106	Operation failed. The cluster does not exist. Try again later.	操作失败，集群不存在，请稍后再试	CCE界面检查集群是否存在，登录CCE控制台，进入集群管理界面，选择“资源管理”中的“集群管理”，查找服务使用的集群是否存在，若不存在，需要重新创建集群订购区块链服务。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001109	Operation failed. Failed to obtain the peer organization status. Try again later.	操作失败，获取节点组织状态失败，请稍后再试	稍后重试。
400	BCS.4001110	Failed to create CertBitConf because the network may be faulty.	创建 CertBitConf 失败，可能出现网络问题	稍后重试。
400	BCS.4001120	Failed to delete the service. The member failed to exit the consortium. The cluster or etcd connection is abnormal.	删除失败，退出联盟阶段失败，可能出现集群异常或者 ETCD 连接异常	登录 CCE 控制台，查看 CCE 服务是否正常；然后进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务；检查 ETCD 状态，若 ETCD 异常，则按照故障恢复指导书中 ETCD 异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001123	Failed to delete the service because the cluster has been hibernated.	删除失败，集群已休眠	唤醒集群后重试。登录 CCE 控制台，进入“资源管理”中的“集群管理”，找到服务使用的集群，单击集群下方的“更多”，选择“集群唤醒”，待集群状态恢复正常后再重试删除。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001130	Failed to save the node and port information because the etcd, CCE API, or cluster is abnormal.	保存节点及 port 信息失败，可能出现 ETCD 异常、CCE 接口或者集群异常	登录 CCE 控制台，查看 CCE 服务是否正常；然后进入集群管理界面，找到服务对应的集群，查看集群状态是否正常，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务；检查 ETCD 状态，若 ETCD 异常，则按照故障恢复指导书中 ETCD 异常的处理指导进行恢复，待恢复正常后重试。
400	BCS.4001133	Failed to save the channel information because etcd may be abnormal.	保存通道信息失败，可能出现 ETCD 异常	检查 ETCD 状态，如果发生 ETCD 异常，参考故障处理指导书中关于 ETCD 故障处理的指导步骤进行操作，待恢复正常后重新进行操作。
400	BCS.4001134	Failed to save the BCS information because etcd may be abnormal.	保存 BCS 信息失败，可能出现 ETCD 异常	检查 ETCD 状态，如果发生 ETCD 异常，参考故障处理指导书中关于 ETCD 故障处理的指导步骤进行操作，待恢复正常后重新进行操作。
400	BCS.4001135	Failed to save the certificate because etcd may be abnormal.	保存证书失败，可能出现 ETCD 异常	检查 ETCD 状态，如果发生 ETCD 异常，参考故障处理指导书中关于 ETCD 故障处理的指导步骤进行操作，待恢复正常后重新进行操作。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001136	Failed to save bitconf because etcd may be abnormal.	保存bitconf失败，可能出现ETCD异常	检查ETCD状态，如果发生ETCD异常，参考故障处理指导书中关于ETCD故障处理的指导步骤进行操作，待恢复正常后重新进行操作。
400	BCS.4001137	Failed to save the deployment template because etcd may be abnormal.	保存部署模版失败，可能出现ETCD异常	检查ETCD状态，如果发生ETCD异常，参考故障处理指导书中关于ETCD故障处理的指导步骤进行操作，待恢复正常后重新进行操作。
400	BCS.4001149	Failed to query the cluster details because the CCE service or API is abnormal.	集群详情查询失败，可能出现CCE服务或接口异常	检查CCE及其接口状态。登录CCE服务控制台，查看CCE服务是否足够正常；若服务正常则进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001161	Failed to add the organization because the etcd connection or CCE API may be abnormal.	添加组织失败，可能出现ETCD链接异常或者CCE服务接口异常	登录CCE服务控制台，查看CCE服务是否正常；若服务正常则进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务；按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001171	Weak password.	输入密码强度不够。	请输入安全性较高的密码。
400	BCS.4001196	Operation failed because the current service or services in the consortium are being updated. Try again later.	操作失败，当前服务或者所在联盟中的服务正在更新，请稍后重试	请稍后重试。
400	BCS.4001202	Deletion failed. The service does not exist or has already been deleted. Refresh the page.	删除失败。服务不存在或已被删除，请刷新	服务不存在或已被删除，重新刷新BCS服务界面即可。
400	BCS.4001203	Operation failed. Failed to obtain the deployment task.	操作失败，获取部署任务失败	请再次尝试获取

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001212	Operation failed. The ECS specified by node_flavor in the request cannot be purchased.	操作失败，请求参数中的 node_flavor 指定的机器不可购买	将修 node_flavor 的值修改为指定其他规格的 ECS 机器。
400	BCS.4001234	Operation failed because the BCS service is in the Hibernating or Frozen state.	操作失败，BCS 服务处于休眠或冻结状态	将服务唤醒，单击对应服务右侧的“更多”，单击“唤醒”待服务状态恢复正常后再操作。
400	BCS.4001242	Failed to remove the organization from a channel.	操作失败，组织退出通道失败	请检查通道及 BCS 服务实例状态，待正常后再下发组织退通道请求。
400	BCS.4001301	Failed to query the ROMA instance information because the ROMA instance may have been deleted or its status is abnormal.	查询 ROMA MQS 实例信息失败，可能为 ROMA MQS 实例被删除或者状态异常导致	登录 ROMA 服务控制台，检查对应实例的状态是否正常，待实例状态恢复正常或者更换其他未被使用的状态正常的实例。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001302	Failed to query the cluster information because the cluster has been deleted or is abnormal.	查询集群信息失败，集群已经被删除或者状态异常	登录CCE服务控制台，查看CCE服务是否足够正常；若服务正常则进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群是否足够存在，若不存则说明集群已经被删除，需使用其他状态正常的集群重新订购BCS服务，若集群存在，则检查状态是否正常，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务。
400	BCS.4001304	Failed to query the ROMA instance list because the ROMA MQS API is abnormal.	查询ROMA MQS实例列表失败，ROMA MQS服务接口异常导致	登录ROMA服务控制台，确认服务状态是否正常，检查对应实例的状态是否正常，待服务或实例状态恢复正常后继续使用，或者参考下ROMA的故障处理指导进行恢复。
400	BCS.4001305	Failed to obtain the topic information of the ROMA instance because the ROMA instance is abnormal or has been deleted.	获取ROMA MQS实例 Topic信息失败，ROMA MQS实例异常后者被删除导致	请至ROMA界面检查服务使用的实例状态是否正常，恢复正常后再重试操作。若实例被删除掉了，则需要购买新的实例，并重新创建服务才能使用。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001306	Operation failed because the ROMA instance is currently in use.	操作失败，ROMA MQS实例已被使用	请重新选择未被使用且状态正常的ROMA MQS实例。
400	BCS.4001307	Operation failed because you do not have the permissions to view DMS data. Check your permissions in IAM.	操作失败，当前用户没有权限查看DMS数据，请在IAM检查账号权限	请在IAM内检查账号权限，具体所需账号权限参考用户指南中给出的用户权限，将DMS操作相关的权限添加后再重试。
400	BCS.4001356	Operation failed. No network storage space has been configured for the cluster.	操作失败，集群下无网络存储，请修改后重新提交	将对应集群下绑定好网络存储后重新提交。单击购买界面的“网络存储”下拉选项，选择“创建网络存储卷”。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001401	Failed to update the IPMapping information of the current tenant because etcd, the CCE API, or the cluster is abnormal.	更新当前租户 IPMapping 失败，可能为 ETCD 异常或者 CCE 接口或者集群异常导致	进入 CCE 服务控制台，选择“资源管理”->“集群管理”，找到对应服务使用的集群，确认集群是否正常，若集群异常请参考 CCE 服务的异常处理或者使用其他可用集群订购服务；若集群状态正常，则选择“配置中心”->“配置项 ConfigMap”，按照集群过滤，找到服务对应集群的 IPMapping，查看配置内容是否正确，格式是否完整；若以上正常，按照故障处理指导书中 ETCD 故障处理指导中的步骤，检查 ETCD 状态，若 ETCD 异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001402	Failed to update the access address because the EIP information in the SDK configuration fails to be updated due to an etcd exception.	更新访问地址失败，ETCD 异常导致更新 SDK 配置中的 EIP 相关信息失败	按照故障处理指导书中 ETCD 故障处理指导中的步骤，检查 ETCD 状态，若 ETCD 异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001404	Failed to update the EIP information of other consortium members because etcd may be abnormal.	更新其他联盟方弹性IP信息失败，可能出现ETCD异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001407	Failed to obtain EIP data.	EIP数据获取失败	请重试或查看确保ETCD是否正常。
400	BCS.4001408	The current domain name has been used. Do not use the domain name in different BCS instances.	当前域名已被使用，请勿重复使用在不同BCS实例	请勿重复使用该域名，建议重新申请域名或添加新域名解析集使用。
400	BCS.4001520	Failed to obtain the topology information of the invitee because etcd may be abnormal.	获取被邀请者拓扑信息失败，可能出现ETCD异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001521	Failed to obtain the topology information of the inviting party because etcd may be abnormal.	获取邀请者拓扑信息失败，可能出现ETCD异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001708	Operation failed. The IP address of the IEF node cluster is incorrect.	操作失败，IEF节点集群IP不正确	1、登录CCE服务控制台。2、选择“资源管理 > 集群管理”，单击服务对应的集群，进入集群管理界面。3、单击“节点管理”，进入节点管理界面，查看集群节点绑定对应的EIP。4、设置该部分IEF节点的公网IP为集群节点绑定的EIP。
400	BCS.4001717	Operation failed. The number of IEF nodes is less than that required by the consensus policy.	操作失败，IEF节点数量小于共识策略所需的节点数量	根据所选的共识策略，选购的IEF节点数量大于等于共识策略所需的节点数量。
400	BCS.4001750	Failed to query the quota from ManageOne because the ManageOne API may be abnormal.	从MO查询配额失败，MO接口异常导致	检查MO接口状态。登录Console界面，选择“VDC”，单击“VDC列表”界面中的“配额管理”，若跳转界面能正常加载出相应region的各个服务的配额信息，则说明MO接口正常，否则说明MO接口出现了异常，需要等MO接口功能恢复后再操作。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001751	The BCS service quota is not found because the quota is not registered or the quota configuration is incorrect.	未找到BCS服务配额，可能配额未注册或者配额信息配制错误	MO界面确认BCS配额是否存在，登录Console界面，选择“VDC”，单击“VDC列表”界面中的“配额管理”，若跳转界面能正常加载出相应region的BCS服务的配额信息，则说明BCS服务配额注册正确，否则说明BCS服务配额注册错误或者未注册，不存在或者注册的信息错误则需要管理员重新注册。
400	BCS.4001752	Failed to modify the quota on ManageOne because the ManageOne API may be abnormal.	修改MO配额失败，可能出现MO接口异常	检查MO接口状态。登录Console界面，选择“VDC”，单击“VDC列表”界面中的“配额管理”，若跳转界面能正常加载出相应region的各个服务的配额信息，则说明MO接口正常，否则说明MO接口出现了异常，需要等MO接口功能恢复后再操作。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001753	Failed to query the product from ManageOne because the product does not exist, the product information is abnormal, or the ManageOne API is abnormal.	从MO查询产品失败，可能原因为产品不存在或者产品信息异常，或者MO接口异常导致	检查MO接口状态，检查产品信息。登录Console界面，选择“VDC”，单击“VDC列表”界面中的“配额管理”，若跳转界面能正常加载出相应region的各个服务的配额信息，则说明MO接口正常，检查其中是否有BCS的相关信息，没有的话需要找管理员注册BCS，若界面不能正常显示否则说明MO接口出现了异常，需要等MO接口功能恢复后再操作。
400	BCS.4001754	This operation cannot be performed on the order because the order parameters do not exist.	无法对订单进行对应操作，订单对应参数信息不存在	重新订购。
400	BCS.4001755	Failed to save the order parameters because etcd may be abnormal.	保存订单对应参数失败，可能出现ETCD异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4001756	The order processing cannot proceed because the order parameters have changed.	订单对应的参数发生变化，不支持继续执行订单操作	重新订购。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001757	Operation verification failed. Only create and delete operations are supported.	不支持此操作，操作校验失败，只支持 create、delete 操作	确认是否为 create、delete 操作。
400	BCS.4001999	Failed to accept the invitation. Refresh the page and check the invitation status later.	同意加入失败。您已同意加入此通道，请勿再次单击并稍后刷新查看通知状态	同意加入失败。您已同意加入此通道，请勿再次单击并稍后刷新查看通知状态。
400	BCS.4003007	No authorization from CCE.	帐号尚未进行 CCE 授权	进入 CCE 控制台，确认同意授权
400	BCS.4004001	You cannot select EVS storage because the CCE cluster version is earlier than 1.15.	CCE 集群版本低于 1.15 不支持 EVS	登录 CCE 服务控制台，选择“资源管理” -> “集群管理”，单击服务对应的集群，查看集群版本号是否大于 1.15，如果集群版本号小于 1.15，则新建一个 1.15 以上的 CCE 集群，如果 CCE 不支持创建 1.15 以上版本集群，联系 CCE 服务支撑人员。
400	BCS.4004002	EVS storage is not supported.	BCS 服务不支持使用 EVS 服务	登录工具 CDK 控制台，选择对应 region，“云服务管理” -> “服务升级”，选择 BCS 服务部署所在集群，选择 baas-service 实例进行升级，修改 evs_service_enable 参数为 true 后进行升级，等待升级完成。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4004003	You cannot select EVS storage because the EVS disk and the CCE node are located in different AZs.	EVS可用区和CCE节点可用区不一致，无法使用EVS	登录CCE服务控制台，选择“资源管理”-》“集群管理”，单击服务对应的集群，进入“节点管理”界面，查看集群节点可用区，接着进入“存储管理”，选择“云硬盘存储卷”，单击“购买云硬盘存储卷”，查看云硬盘存储卷的可用区是否和集群节点可用区一致，如果不一致建议购买和云硬盘存储卷可用区一致的CCE集群。
400	BCS.4004004	You cannot select EVS storage when the Fabric version of the BCS service is earlier than 3.0.	Fabric版本低于3.0，BCS不支持使用EVS	创建BCS服务版本信息选择3.0以上即可。
400	BCS.4004005	You cannot change the billing mode of the BCS service to yearly/monthly because the BCS service uses EVS storage.	BCS服务使用EVS存储，不支持转为包周期服务	目前不支持使用EVS存储购买包周期服务，如果使用包周期服务，建议使用SFS网盘存储。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4004006	Failed to query the EVS type.	查询EVS服务类型失败	排查网络是否异常，手动curl evs服务域名，域名不通联系DNS服务人员；登录EVS服务页面，单击购买磁盘，查看页面EVS是否报错，如果出现错误，请联系EVS运维人员；排查BCS服务后端服务是否正常，如果异常baas-service查看相应日志。
400	BCS.400402	Query failed.	查询失败	重新检查输入参数中的实体类型
400	BCS.4004022	Failed to query the monitoring data because the AOM API may be abnormal. Check whether the AOM service is normal and whether the AOM monitoring API can be accessed.	监控数据查询失败，可能原因为AOM接口访问异常，请检查AOM服务以及AOM监控接口访问地址的联通情况	请检查AOM服务以及AOM监控接口访问地址的联通情况。
400	BCS.4004025	Invalid request information.	请求信息无效。	请检查请求信息，确保请求信息正确填写。
400	BCS.4004029	Query failed.	查询失败	重新检查输入参数后重试
400	BCS.4006005	Failed to query the BCS service because the etcd connection may be abnormal.	BCS服务查询失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4006007	Failed to submit the request because no EIP is bound to the CCE cluster or the EIP fails to be queried.	提交请求失败，CCE集群未绑定弹性IP或弹性IP查询失败	具体步骤：登录CCE服务控制台，进入集群管理界面，选择“资源管理”中的“节点管理”，过滤服务对应的集群的节点，查看弹性IP的绑定情况，如果是私有链则需要至少一个节点绑定了弹性IP，如果是联盟链则要求所有节点都绑定了弹性IP。操作绑定的步骤如下：单击节点名称，会跳转到ECS界面；在ECS界面单击节点名称后，会跳转到详情界面，切换至“弹性公网IP”页签；单击“绑定弹性公网IP”。
400	BCS.4006009	Failed to download the SDK configuration because the etcd connection or the CCE cluster status is abnormal.	下载SDK配置失败，可能出现ETCD连接异常或CCE集群状态异常导致	登录CCE服务控制台，查看CCE服务是否足够正常；若服务正常则进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，是否有可用节点，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务；按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4006012	Failed to create the channel because the etcd connection or the CCE cluster status is abnormal.	创建通道失败，可能出现ETCD连接异常或CCE集群状态异常导致	登录CCE服务控制台，查看CCE服务是否正常；若服务正常则进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，是否有可用节点，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务；按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4006013	Failed to update the BCS service instance information because the etcd connection is abnormal.	更新BCS服务实例信息失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4006017	Failed to upgrade the BCS service because the etcd connection or the CCE cluster status is abnormal.	BCS服务升级失败，可能出现ETCD连接异常或CCE集群状态异常导致	登录CCE服务控制台，查看CCE服务是否足够正常；若服务正常则进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，是否有可用节点，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务；按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4006018	Failed to query data because the etcd connection is abnormal.	数据查询失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4006019	Failed to change the password because the etcd connection, the CCE configuration update API, or the cluster status is abnormal.	修改密码失败，可能出现ETCD连接异常或CCE配置更新接口或集群状态异常导致	登录CCE服务控制台，查看CCE服务是否正常；若服务正常则进入集群管理界面，选择“资源管理”中的“集群管理”，找到服务对应的集群，查看集群状态是否正常，若集群异常，待集群恢复后进行操作，或者重新选择使用状态正常的集群创建区块链服务；按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4006025	Invitation failed because the tenant has already been invited or the etcd connection is abnormal.	邀请失败，该租户已经被邀请过或者出现ETCD连接异常	BCS服务控制台界面上，切换至“成员管理”页签，检查该是否有该租户的邀请信息，若邀请信息已经存在，则说明已经邀请过了不能重复邀请，等待对方响应即可，若没有该租户的邀请信息，则可能是发生了ETCD连接异常，可进入下一步。按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4006026	Some invitations failed because the etcd connection is abnormal.	部分邀请失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4006027	Failed to list the members because the etcd connection is abnormal.	查询成员列表信息失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4006032	Failed to download the SDK configuration. The OrderEIP is empty because the etcd connection may be abnormal.	下载SDK配置失败，OrderEIP查询结果是空值，可能出现ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4006034	Failed to obtain the notification information because the etcd connection is abnormal.	获取通知信息失败，ETCD连接异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4006036	Failed to generate the organization certificate because the etcd connection is abnormal.	组织证书生成失败，ETCD连接出现异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4006037	Failed to send the event information because the AOM API or the network connection is abnormal.	Event信息发送失败，可能出现AOM接口异常或网络链接异常	登录AOM服务控制台，检查AOM服务是否正常，相关信息是否上报，若故障了需要等AOM恢复后Event才能正常上报。若AOM服务接口都正常，则可能出现了网络链接波动，稍后重试即可。
400	BCS.4006038	Failed to update the notification information because the etcd connection is abnormal.	通知信息更新失败，ETCD连接出现异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4006039	Failed to submit the request because the cluster is in use.	提交请求失败，集群已被使用	重新选择其他未被占用的集群，重新操作。
400	BCS.4006040	Failed to save the member information because the etcd connection is abnormal.	成员信息保存失败，ETCD连接出现异常	按照故障处理指导书中ETCD故障处理指导中的步骤，检查ETCD状态，若ETCD异常，则按照指导书中的步骤进行恢复，待恢复正常后重试。
400	BCS.4006041	Failed to delete the notification because finished notifications cannot be deleted.	删除通知失败，已完成状态的通知不允许删除	通知状态为已完成，不允许删除。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4009100	System error. Check the system or network status.	系统错误，请检查系统或网络状态是否正常	将界面从BCS服务控制台切换至CCE等其他服务界面查看，若多个服务都出现系统异常，则是环境整体公共组件类的故障导致的，等故障恢复后重试，若是只有BCS服务出现异常，则可能是BCS服务后台节点发生了故障。参考应急预案中关于“BCS服务节点故障”指导进行处理。
400	BCS.4009101	The request URL does not exist. Check whether the URL has been registered.	请求URL不存在，请检查URL是否已在系统中注册	登录APIG注册接口的服务界面，找到BCS相关的接口注册列表，并在其中查找是否有操作报错的对应接口的信息，若不存在，则去BCS服务的安装包中查找APIG接口文档，找到缺失的接口并按照文档进行注册。
400	BCS.4009102	Internal system error. Check the system status or network status.	系统内部异常，请检查系统或网络状态是否正常	将界面从BCS服务控制台切换至CCE等其他服务界面查看，若多个服务都出现系统异常，则是环境整体公共组件类的故障导致的，主要等故障恢复后重试，若是只有BCS服务出现异常，则可能是BCS服务后台节点发生了故障。参考应急预案中关于“BCS服务节点故障”指导进行处理。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4009103	The maximum number (5) of invited members in the channel has been reached.	当前通道中邀请的成员数已超出上限值5个	每个通道邀请成员数不得超出上限值。
401	BCS.4010401	Permission verification failed.	用户权限校验失败	按照用户指南-服务部署-前提条件章节，重新配置权限后再进行操作
403	BCS.4030403	Insufficient permissions.	用户权限不足	按照用户指南-服务部署-前提条件章节，重新配置权限后再进行操作