区块链服务

开发指南

文档版本 01

发布日期 2025-12-09

华为云计算技术有限公司

商标声明


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

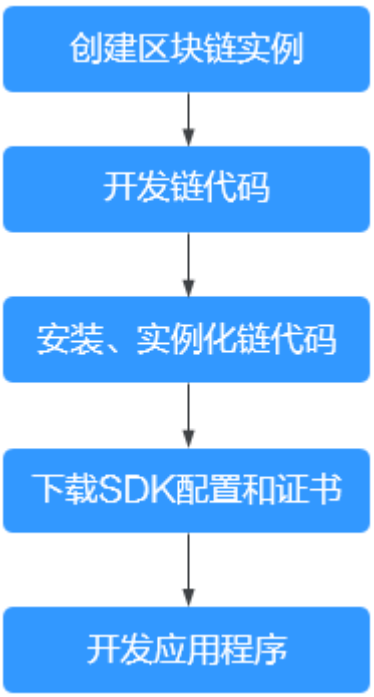
1 Hyperledger Fabric 增强版管理

1.1 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

   Fabric架构版本的区块链实例支持在CCE集群和边缘集群上部署，具体可参见**基于CCE集群**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**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**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。针对Fabric架构版本的区块链实例，应用开发过程中可使用BCS提供的**国密加密SDK**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，Fabric架构版本的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**

1.2 链代码开发

1.2.1 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/节点调用链代码，用户链代码通过区块链网络的Peer节点/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。
- 为了确保代码在不同用户之间的一致性运行，区块链服务参考了成熟的开源社区方案（如Hyperledger Fabric）。用户在部署智能合约时，使用预配置的容器镜像进行处理，通过预配置的开发/编译工具（例如：javac、cpp、gcc等）减少环境差异带来的问题，确保智能合约能够在区块链网络上正确运行。区块链服务通过资源隔离等方式，降低了由此带来的安全风险。
- 为了保障链上数据的机密性，建议采用以下手段进行防护：
  - 数据加密：对链上存储的数据进行加密，只允许授权的参与方解密和查看数据。常用的加密方法包括对称加密和非对称加密。
  - 访问控制：实施严格的访问控制机制，确保只有被授权的用户或节点才能访问敏感数据。例如：不接入公网或使用VPC安全组等网络隔离手段限制访问。
  - 链下数据存储：对于特别敏感的数据，可以采用链下存储的方式，将数据存储在链下，并在区块链上记录其哈希值等摘要信息，以便在需要时验证数据的一致性和完整性。

开发环境准备

请根据自身业务选择Go或Java开发环境。推荐使用**CloudIDE**（支持在线、快速地构建链代码开发环境）。

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。仅适用于Fabric架构版本的区块链实例。

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。

1.2.2 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Fabric架构版本的区块链实例的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 painc. 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

说明

该内容仅适用于Fabric架构版本的区块链实例。

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Fabric架构版本的区块链实例的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。

1.2.3 Go 语言链代码开发

1.2.3.1 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

Fabric架构版本的区块链实例：

- 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

- Fabric架构版本的区块链实例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)
   }
}
```

- Fabric架构版本的区块链实例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)
    }
}
```

1.2.3.2 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。

1.2.3.3 链代码示例（1.4 风格）

Fabric架构版本的区块链实例：

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\"")
}

// 账户间转钱
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)
    }
}
```

1.2.3.4 链代码示例（2.0 风格）

Fabric架构版本的区块链实例：

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)
        }
}
```

1.2.3.5 链代码调测

对链代码进行调测，可以使用CloudIDE进行单元测试，具体请参见以下操作。

操作步骤

步骤1 进入CloudIDE首页。

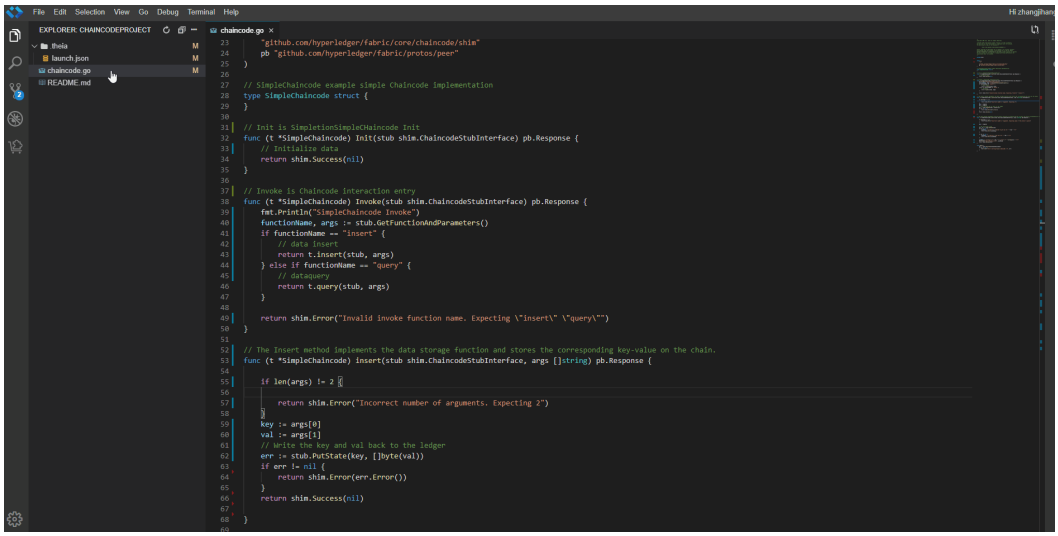
步骤2 创建链代码工程，具体请参见**创建并启动IDE实例**。

图 1-2 创建链代码工程

步骤3 链代码调测。

说明

以下截图对1.4风格（使用shim包）的链代码进行调测。



----结束

1.2.4 Java 语言链代码开发

1.2.4.1 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。

Java链代码仅适用于Fabric架构版本的区块链实例。

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);
    }
}
```

1.2.4.2 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。

1.2.4.3 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 newSuccessResponse();
    }

    @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 newErrorResponse("Invalid invoke function name. Expecting one of: [\"insert\", \"query\"]");
        } catch (Throwable e) {
            return newErrorResponse(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 newErrorResponse("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);
    }
}
```

1.2.4.4 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 1-3 创建测试文件

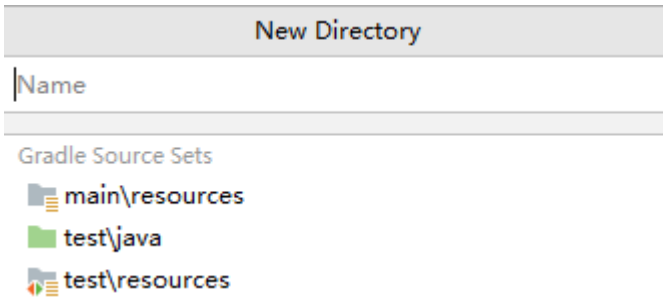
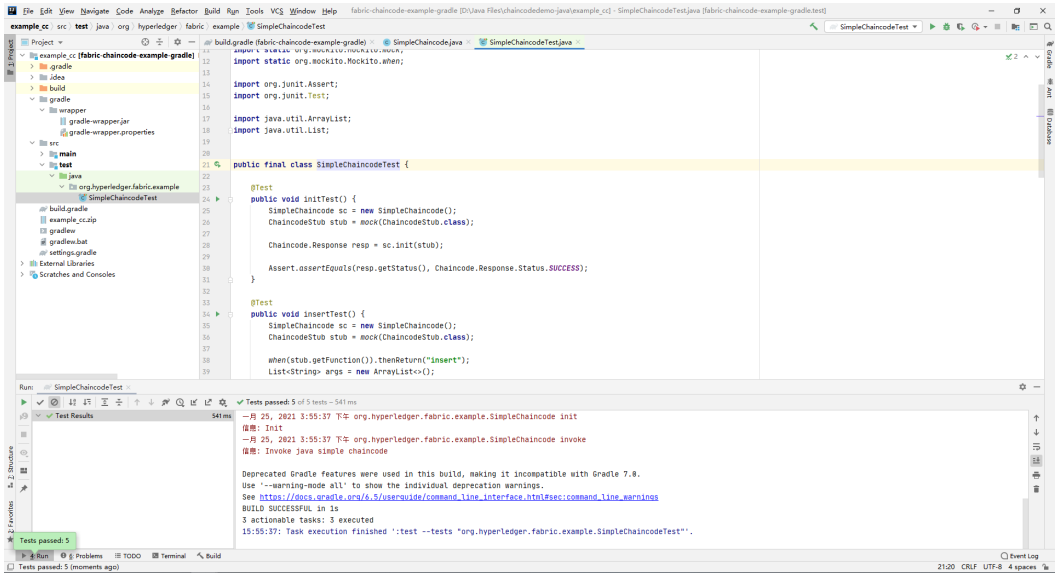


图 1-4 测试文件



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 queryNoExistTest() {
        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 1-5 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      }
```

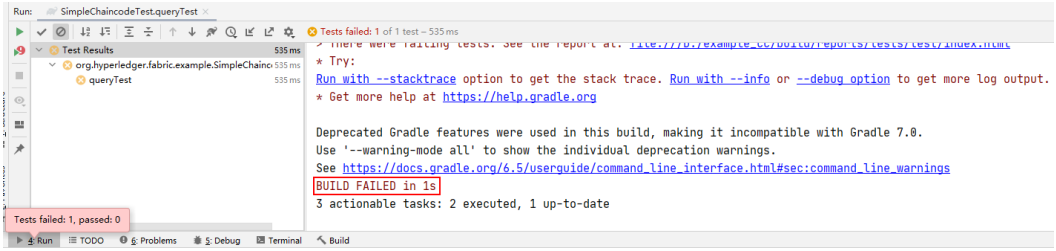
执行成功如图所示，表示链代码调测无问题：

图 1-6 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 1-7 执行失败




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

1.3 应用程序开发

1.3.1 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**。
- Java语言，请参考**Java SDK使用指导**。
- Python语言，请参考**Python使用指导**。
- Node.js语言，请参考**Node.js SDK使用指导**。
- Go语言，请参考**Go SDK使用指导**。

1.3.2 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。

1. 您需要购买区块链实例。

   Fabric架构版本的区块链实例支持在CCE集群和边缘集群上部署，具体可参见基于CCE集群**基于CCE集群**。

2. 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**下载SDK配置和证书**。

1.3.3 应用程序开发

您需要自行开发应用程序业务逻辑代码。针对Fabric架构版本的区块链实例，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，Fabric架构版本的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

Fabric架构版本的区块链实例：

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 1-8 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。

1.4 示例 Demo

1.4.1 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。

说明

本Demo仅适用于Hyperledger Fabric增强版的区块链实例。

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。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**。

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，选择“Hyperledger Fabric增强版”页签，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

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:

  ```
  deffab.Options{ConfigFile: configFile, LoggerFa
  logging.LoggerProvider(), UserName: sysadmin}
  ```

  ConfigFile是SDK配置文件的路径。LoggerFactory是可选的，不提供的话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.TransactionProposlResponse, 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
QueryWithOpts	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
ExecuteTx	链码调用接口，用于链码的调用。	request ExecuteTxRequest	TransactionID, error

接口名称	描述	参数值	返回值
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 ExecuteTxRequestopt 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
UpgradeCCWithOpts	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, optsUpgradeCCOpts	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
DeleteCCWithOpts	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,opts DeleteCCOpts	error
JoinChannel	Peers加入Channel的接口，用于peers加入Channel。	channelID string	error
JoinChannelWithOpts	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method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 1-1 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。

1.4.2 Java SDK Demo

本节提供一个基于Java SDK的Demo，帮助用户开发自己的Java客户端应用程序。

说明

本Demo仅适用于Hyperledger Fabric增强版的区块链实例。

准备工作

1. **开通CloudIDE**（或者用户自己准备JDK、maven和eclipse/IntelliJ IDEA）。

   CloudIDE是软件开发生产线 CodeArts的云端开发环境服务，向开发者提供按需配置、快速获取的工作空间（包含编辑器和运行环境）。

   在CloudIDE上创建一个空的Java工程，如**图1-9**所示。

图 1-9 CloudIDE 上创建一个空的 Java 工程

2. 下载Java SDK示例源码，获取方法：登录区块链服务管理控制台，进入“应用案例”，在“Java示例Demo-Java SDK Demo”下方，单击“App_Java_Src_Demo”中Java项目源码的“下载”按钮。
3. 购买区块链实例、安装链代码及实例化链代码操作，请参见《快速入门》。请参见**《快速入门》**。

部署应用

1. 下载SDK和证书。
   a. 在“实例管理”页面，选择“Hyperledger Fabric增强版”页签，单击对应实例卡片上的“获取客户端配置”。
   b. 勾选“SDK文件”，SDK配置参数如下：

参数名称	说明
链代码名称	chaincodedemo
证书存放路径	/home/user/javasdkdemo_src/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”。
   c. 单击“下载”，下载SDK配置文件、demo-orderer组织的管理员证书和organization组织的管理员证书。
2. 复制并解压。
   a. 先下载工程源代码javasdkdemo_src.zip文件并解压。

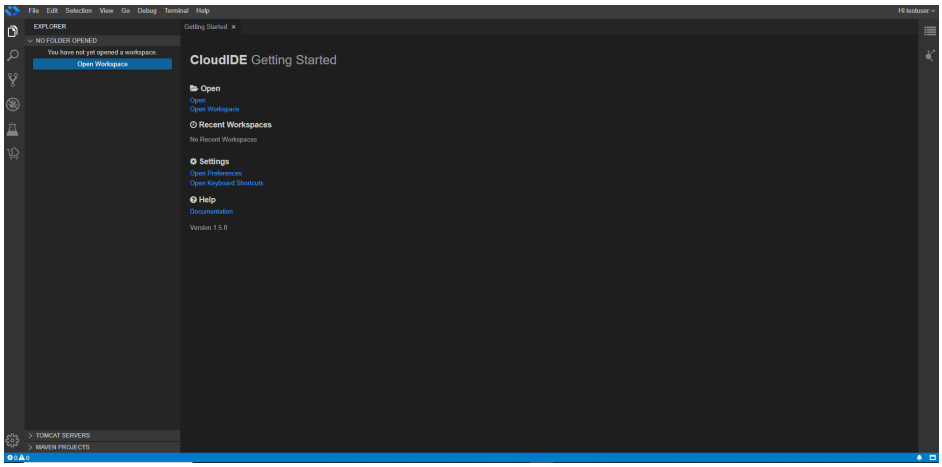
   获取方法：登录区块链服务管理控制台，进入“应用案例”，在“Java示例Demo-Java SDK Demo”下方，单击“App_Java_Src_Demo”中Java项目源码的“下载”按钮。

b. 将1步骤中的zip文件解压，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javasdkdemo_src目录下的config目录下。然后再将javasdkdemo_src目录压缩成javasdkdemo_src.zip包。

3. 部署应用。

a. 将新压缩的javasdkdemo_src工程源代码javasdkdemo_src.zip文件上传到CloudIDE开发环境中。

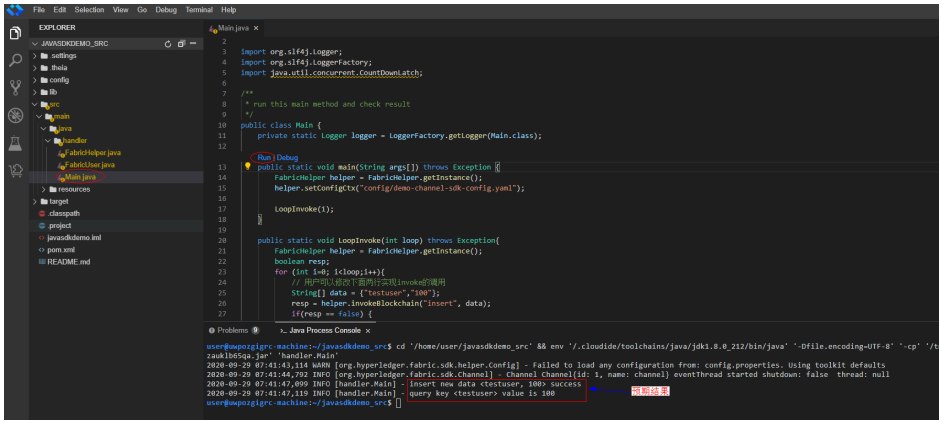
CloudIDE编译工程如下所示：



b. 打开工程后等待一会，会自动下载工程依赖的包，然后按下图操作执行Run，就会得到预期结果。

⚠ 注意

- javasdkdemo工程中config目录下**demo-channel-sdk-config.yaml**文件，必须与javasdkdemo/src/main/java/handler/Main.java文件中helper.setConfigCtx("config/**demo-channel-sdk-config.yaml**")代码路径相同，保证可以正常运行Main.java。
- 客户端app交易的时候，如果指定了未实例化的组织和peer，那么首次交易会超时失败，请您重新运行即可正常交易。



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。

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。

常用接口

Fabric-sdk-java的使用主要分为：加密套件配置，通道操作，链码操作。

加密套件配置部分，根据配置文件内容，设置SDK里面的加密套件服务提供者、 加密类型、安全级别等。

- 通道操作包括：通道查询、通道创建、通道加入、通道删除等。
- 链码操作包括：链码安装、链码实例化、链码查询、链码调用等。

使用Fabric-sdk-java主要会使用HFClient和Channel两个类，其导入路径分别为：org.hyperledger.fabric.sdk.HFClient和org.hyperledger.fabric.sdk.Channel。

更多的api接口请参考Fabric官网。

1.4.3 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。

说明

本Demo仅适用于Hyperledger Fabric增强版的区块链实例。

准备工作

1. **开通CloudIDE**（或者用户自己准备JDK、maven和eclipse/IntelliJ IDEA）。

   CloudIDE是软件开发生产线 CodeArts的云端开发环境服务，向开发者提供按需配置、快速获取的工作空间（包含编辑器和运行环境）。

   在CloudIDE上创建一个空的Java工程，如**图1-10**所示。

图 1-10 CloudIDE 上创建一个空的 Java 工程

2. 下载Java SDK示例源码，获取方法：登录区块链服务管理控制台，进入“应用案例”，在“Java示例Demo-Java SDK Demo”下方，单击“App_Gateway_Java_Demo”中Java项目源码的“下载”按钮。
3. 购买区块链实例、安装链代码及实例化链代码操作，请参见《快速入门》。请参见**《快速入门》**。

部署应用

1. 下载SDK和证书。
   a. 在“实例管理”页面，选择“Hyperledger Fabric增强版”页签，单击对应实例卡片上的“获取客户端配置”。
   b. 勾选“SDK文件”，SDK配置参数如下：

参数名称	说明
链代码名称	chaincodedemo
证书存放路径	/home/user/gatewayjavademo/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”。

   c. 单击“下载”，下载SDK配置文件、demo-orderer组织的管理员证书和organization组织的管理员证书。
2. 复制并解压。
   a. 先下载工程源代码gatewayjavademo.zip文件并解压。

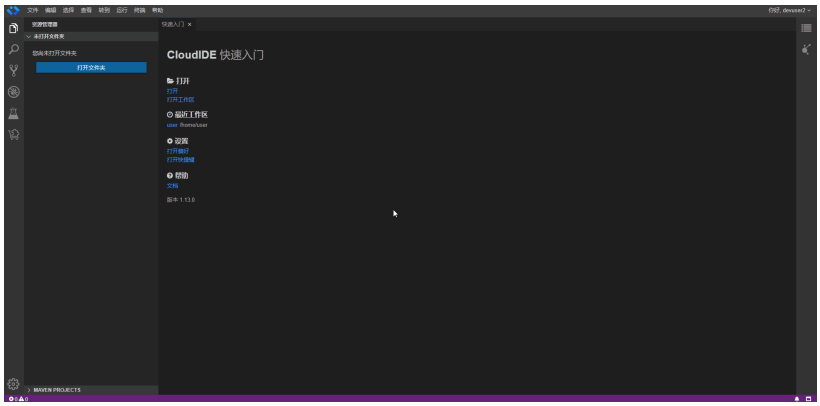
   获取方法：登录区块链服务管理控制台，进入“应用案例”，在“Java示例Demo-Java SDK Demo”下方，单击“App_Gateway_Java_Demo”中Java项目源码的“下载”按钮。

b. 将1步骤中的zip文件解压，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gatewayjavademo目录下的config目录下。然后再将gatewayjavademo目录压缩成gatewayjavademo.zip包。

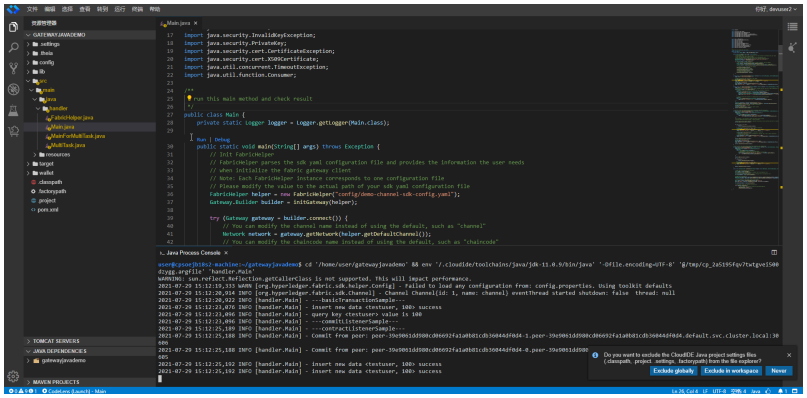
3. 部署应用。

a. 将新压缩的gatewayjavademo工程源代码gatewayjavademo.zip文件上传到CloudIDE开发环境中。

CloudIDE编译工程如下所示：



b. 打开工程后等待一会，会自动下载工程依赖的包，然后按下图操作执行Run，就会得到预期结果。



注意

- gatewayjavademo工程中config目录下**demo-channel-sdk-config.yaml**文件，必须与gatewayjavademo/src/main/java/handler/Main.java和MainForMultiTask文件中helper.setConfigCtx("config/**demo-channel-sdk-config.yaml**")代码路径相同，保证可以正常运行Main.java。
- 客户端app交易的时候，如果指定了未实例化的组织和peer，那么首次交易会超时失败，请您重新运行即可正常交易。

每次成功执行Main.java，将会执行basicTransactionSample、commitListenerSample、contractListenerSample、blockListenerSample四个方法，向区块链存入多对键值对；可以通过区块链浏览器可以查看交易记录。

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

1.4.4 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，用户可以不需要学习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。

仅适用于Hyperledger Fabric增强版的区块链服务。

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1-2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 1-2 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1-3所示。

表 1-3 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
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1-4所示。

表 1-4 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
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。

参数	示例
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
添加RESTful API支持	请选择“是”。若您选择了“否”，则需要在实例创建完毕后，执行以下步骤安装RESTful API： 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

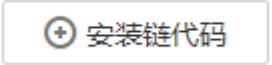
步骤1 登录区块链服务管理控制台。

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

参数	值
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。
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

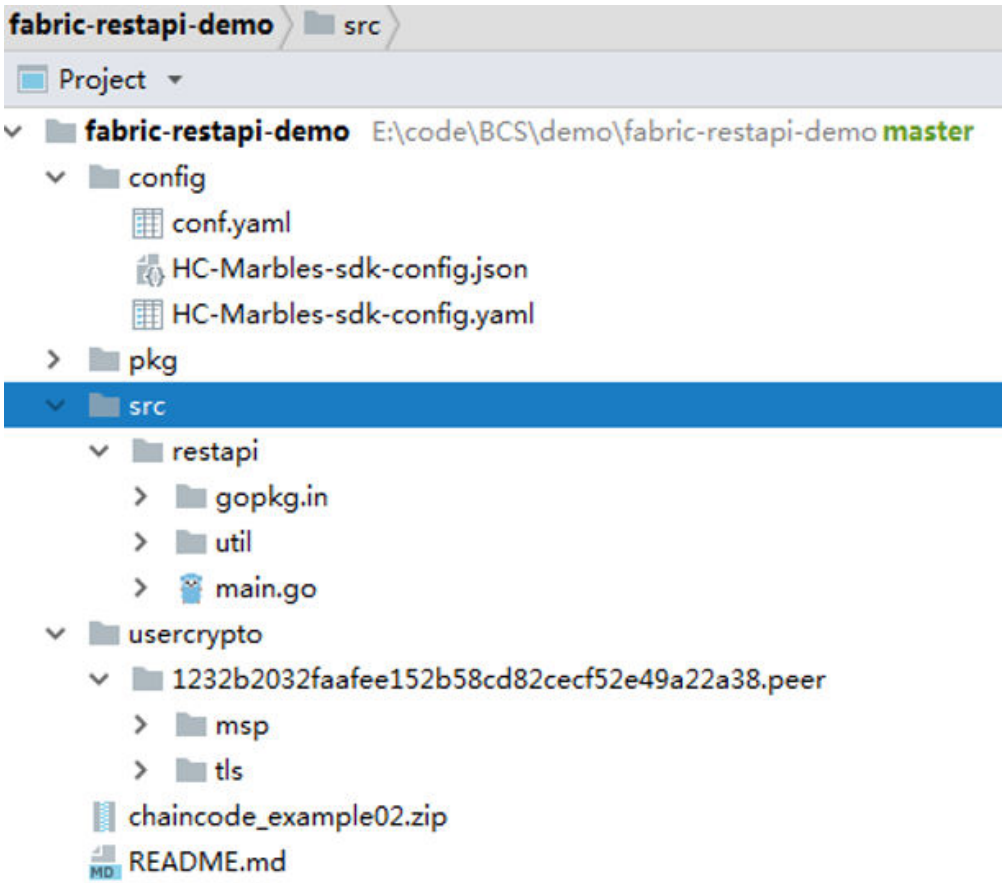
配置应用

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”。

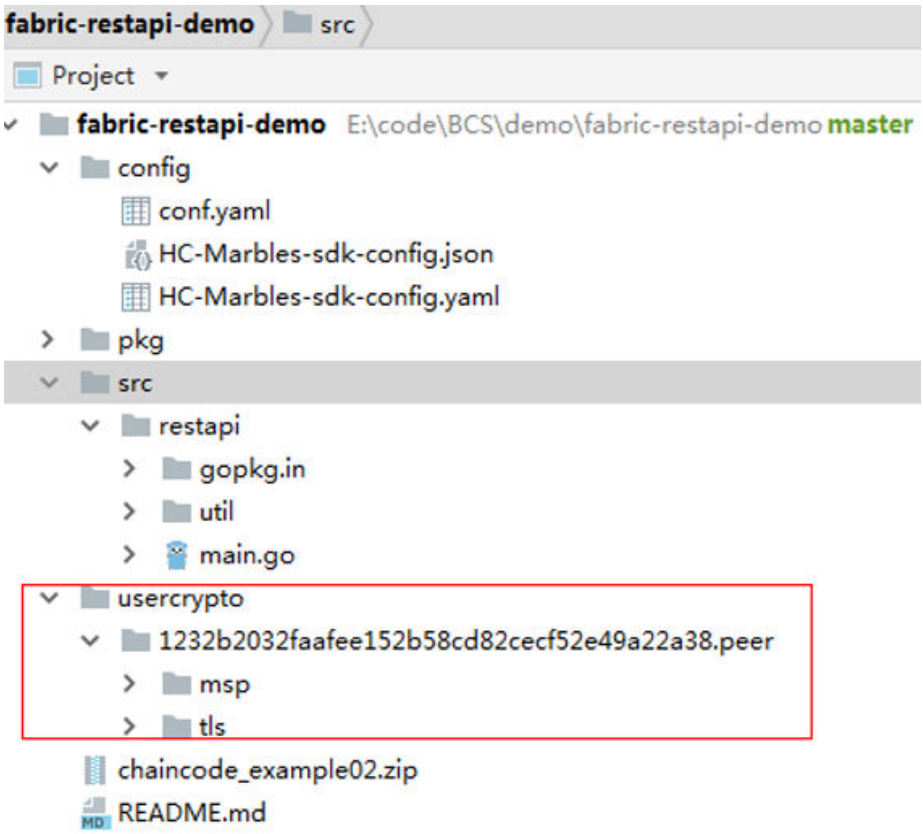
步骤2 勾选“Peer节点证书”，指定节点组织选择organization，勾选“用户证书”并下载。

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。

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

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 1-5 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 1-6 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"
Chaincode Version	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)
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
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

1.5 区块链中间件接口

1.5.1 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1-7**：

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**》。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_addrs字段的值获取，请求示例：https://192.168.0.90:30603/v2/agent/apis/tokens。

表 1-7 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
分布式身份（公测）	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。 **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”。在安装插件页面，开启分布式身份接口。	32621
可信数据交换（公测）	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。 **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”。在安装插件页面，开启可信数据交换接口。	32621

1.5.2 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 1-8 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

参数	是否必选	参数类型	描述
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
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**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1-9**。

表 1-9 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。

名称	是否必选	描述
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。

说明

x-bcs-signature-sign：为了保证只允许有权限的调用端才能够进行合法的链码调用，需要使用**下载用户证书**中下载的用户私钥以ECDSA椭圆曲线的加密方式对整个请求消息体的SHA256摘要进行加密签名，x-bcs-signature-sign值即为签名结果。

下载用户证书

进行API调用前，需要下载区块链服务中已经配置生成的用户证书。

步骤1 登录华为云区块链服务控制台。

步骤2 进入“实例管理”界面，可以看到已经购买的区块链实例卡片。

步骤3 单击卡片上的“获取客户端配置”，勾选“Peer节点证书”。选择指定节点组织，勾选“用户证书”。

步骤4 单击“下载”，下载相应组织的用户证书。

步骤5 解压证书，其中msp文件夹中，keystore文件夹存储的是组织用户私钥，signcerts文件夹存储的是用户证书（公钥）。

----结束

响应消息

- 当opmethod为invoke时，返回值是base64加密的transactionID。
- 当opmethod为query时，返回值是base64加密的链代码的返回值。

示例

下面是调用invoke类型的链代码示例。

- 请求示例

```
{
  "channelId": "testchannel",
  "chaincodeId": "zmmcode",
  "chaincodeVersion": "1.0",
  "userId": "User1",
  "orgId": "7258adda1803f4137eff4813e7aba323018200c5",
  "orgPeers": "[{\"orgId\":\"7258adda1803f4137eff4813e7aba323018200c5\",\"peerDomainName
\":\"peer-7258adda1803f4137eff4813e7aba323018200c5-0.peer-7258adda1803f4137eff4813e7aba323
018200c5.default.svc.cluster.local\"}]",
  "opmethod": "invoke",
  "args": "[\"invoke\",\"a\",\"b\",\"1\"]",
  "timestamp": "2018-10-31T17:28:16+08:00",
  "cert": "-----BEGIN CERTIFICATE-----
\nMIIDBzCCAq2gAwIBAgIQEXPZlMsReamxVtVNnKwCCzAKBggqhkjOPQQDAjCCAQQx
\nDjAMBgNVBAYTBUNISU5BMRAwDgYDVQQIEwdCRUlKSU5HMRAwDgYDVQQHEwdCRUlK
\nSU5HMXkwdwYDVQQKE3A3MjU4YWRkYTE4MDNmNDEzN2VmZjQ4MTNlN2FiYTMyMzAx
```


\nODIwMGM1LnBlZXItNzI1OGFkZGExODAzZjQxMzdlZmY0ODEzZTdhYmEzMjMwMTgy
\nMDBjNS5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsMVMwUQYDVQQDE0pjYS5wZWVy
\nLTcyNThhZGRhMTgwM2Y0MTM3ZWZmNDgxM2U3YWJhMzIzMDE4MjAwYzUuZGVmYXVs
\ndC5zdmMuY2x1c3Rlci5sb2NhbDAeFw0xODEwMzAwMjQ5MjZaFw0yODEwMjcwMjQ5\nMjZaMIG1M
Q4wDAYDVQQGEwVDSElOQTEQMA4GA1UECBMHQkVJSklORzEQMA4GA1UE\nBxMHQkVJSklORzF/
MH0GA1UEAwx2VXNlcjFANzI1OGFkZGExODAzZjQxMzdlZmY0\nODEzZTdhYmEzMjMwMTgyMDBjNS5
wZWVyLTcyNThhZGRhMTgwM2Y0MTM3ZWZmNDgx
\nM2U3YWJhMzIzMDE4MjAwYzUuZGVmYXVsdC5zdmMuY2x1c3Rlci5sb2NhbDBZMBMG
\nByqGSM49AgEGCCqGSM49AwEHA0IABPMrzoJL/MHeSFPFOJWLqnJ0sqB0it7wDIOq\n
+eTSvvPpGk1BIDmb2n13K5V04RO8xNezDQ7I6rW4LF2elq14eH+jTTBLMA4GA1Ud\nDwEB/
wQEAwIHgDAMBgNVHRMBAf8EAjAAMCsGA1UdIwQkMCKAIFBXQ5TC4acFeTlT
\nJuDZg62XkXCdnOfvbejSeKI2TXoIMAoGCCqGSM49BAMCA0gAMEUCIQCadHIKl0Mk
\nYn0WZizyDZYR4rT2q0nzjFaiW+YfV5FBjAIgNalKUe3rIwXJvXORV4ZXurEua2Ag\nQmhcjRnVwPTjpTE=
\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。

1.5.3 链代码管理（公测）

1.5.3.1 获取 Token

功能介绍

根据区块链浏览器用户名密码获取Token。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

GET /v2/agent/apis/tokens

请求参数

表 1-10 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 1-11 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 1-12 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=kaIglHeZLTu6UeIu5YRcXTlpUPXylEG3
ESNu0vIukrikG6SVhO393ds8rG+xxxZ+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

状态码	描述
400	Bad Request

错误码

请参见**错误码**。

1.5.3.2 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

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v2/agent/apis/chaincode/install

请求参数

表 1-13 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 1-14 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
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

参数	是否必选	参数类型	描述
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
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 1-15 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 1-16 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/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:22222222chaincode_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**。

1.5.3.3 实例化链代码

功能介绍

实例化链代码

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v2/agent/apis/chaincode/instantiate

请求参数

表 1-17 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 1-18 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
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

参数	是否必选	参数类型	描述
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：[{"name":"kvstore-collection","policy":"OR('9b4ea44913e8eed42cb056177b46191e1d316678MSP.member','9b4ea44913e8eed42cb056177b46191e1d316678MSP.member')","requiredPeerCount":0,"maxPeerCount":2,"blockToLive":0,"memberOnlyRead":true}]

表 1-19 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 1-20 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 1-21 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 1-22 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','8cc7155fb1f26ebac1743f481386c14223b
e511fMSP.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**。

1.5.3.4 获取安装的链码列表

功能介绍

获取已经安装的链码列表

调试

您可以在API Explorer中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

GET /v2/agent/apis/chaincodes

表 1-23 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 1-24 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 1-25 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 1-26 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(,)分隔
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 1-27 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

表 1-28 instantiated_channel

参数	参数类型	描述
error	Array of **CCInstantiatedChannelError** 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 1-29 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 1-30 instantiated_info

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

表 1-31 channels

参数	参数类型	描述
channel_id	String	通道名称
orgs	Array of **orgs** objects	通道内组织信息
versions	Array of strings	版本信息

表 1-32 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 1-33 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/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**。

1.5.3.5 查询指定链码版本信息

功能介绍

查询指定链码版本信息

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

GET /v2/agent/apis/chaincode/versions

表 1-34 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

请求参数

表 1-35 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 1-36 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 1-37 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 1-38 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 1-39 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" : "1473b4807fe9f970d1ba56192e41d39c7d621d07d80e603cf75ed3982b81034d",
    "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**。

1.5.3.6 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

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

GET /v2/agent/apis/chaincode/install

表 1-40 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

请求参数

表 1-41 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 1-42 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 1-43 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 1-44 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**。

1.5.3.7 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

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

GET /v2/agent/apis/chaincode/instantiate

表 1-45 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

请求参数

表 1-46 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 1-47 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 1-48 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
version	String	链代码版本
orgs_info	Array of **OrgInfo** objects	通道组织信息

参数	参数类型	描述
has_private_data	Integer	是否有隐私数据，1表示有，0表示无

表 1-49 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 1-50 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 1-51 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" : "65cfb1c760f24058c865ffcfd8ce1cdb690bf2a3"
    }, {
      "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**。

1.5.3.8 查询应用链信息

功能介绍

通道概要信息查询

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

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

表 1-52 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 1-53 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 1-54 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 1-55 bph

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

表 1-56 bpm

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

表 1-57 tph

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

表 1-58 tpm

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

表 1-59 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 1-60 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 1-61 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

```
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**。

1.5.3.9 查询区块列表

功能介绍

查询区块列表

调试

您可以在API Explorer中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

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

表 1-62 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 1-63 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
offset	否	Integer	偏移量，默认为0
limit	否	Integer	每页显示的条目数量，默认为10，取值范围为1-50

请求参数

表 1-64 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 1-65 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 1-66 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 1-67 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/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+3R95mvdlI8mH1dvmTSTylPwwzMsbY",
    "transaction_count" : 1,
    "data_hash" : "+3B6RFfbQisE8zt2BeWTvCwP1JbmQLIPeQoiKxoCQVA",
    "previous_hash" : "FlEClYDQJ4cHaEslex9usupte0EqbHyymQ+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**。

1.5.3.10 查询交易列表

功能介绍

查询交易列表

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

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

表 1-68 路径参数

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

表 1-69 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
offset	否	Integer	偏移量，默认为0
limit	否	Integer	每页显示的条目数量，默认为10，取值范围为1-50

请求参数

表 1-70 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 1-71 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 1-72 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 1-73 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**。

1.5.3.11 查询交易总数

功能介绍

查询交易总数

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

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

表 1-74 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 1-75 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 1-76 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_num	Integer	交易数量

状态码： 400

表 1-77 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**。

1.5.3.12 查询区块交易列表

功能介绍

查询区块交易列表

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

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

表 1-78 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 1-79 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 1-80 响应 Body 参数

参数	参数类型	描述
[数组元素]	Array<Array<**ShowTransactionDetailRes**>>	Success

表 1-81 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 1-82 KVRead

参数	参数类型	描述
key	String	读键

参数	参数类型	描述
version	**version** object	读集键的版本

表 1-83 version

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

表 1-84 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 1-85 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 1-86 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/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,
    "lscc" : [ {
      "key" : "kvtest",
      "version" : {
        "block_num" : 2
      }
    } ]
  },
  "write_set" : {
    "kvtest" : [ {
      "key" : "a1",
      "is_delete" : false,
      "value" : "1"
    } ],
    "lscc" : [ ]
  }
} ]
```

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**。

1.5.3.13 查询交易详情

功能介绍

查询交易详情

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

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

表 1-87 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 1-88 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 1-89 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"
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 1-90 KVRead

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

表 1-91 version

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

表 1-92 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 1-93 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 1-94 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" : "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,
  "lscc" : [ {
   "key" : "kvtest",
   "version" : {
    "block_num" : 2
   }
  } ]
 },
 "write_set" : {
  "kvtest" : [ {
   "key" : "a1",
   "is_delete" : false,
   "value" : "1"
  } ],
  "lscc" : [ ]
 }
}
```


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**。

1.5.3.14 查询节点状态

功能介绍

查询节点状态

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

GET /v2/agent/apis/peers

请求参数

表 1-95 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 1-96 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 1-97 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 1-98 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/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**。

1.5.3.15 删除链代码

功能介绍

删除区块链节点上的链代码

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

DELETE /v2/agent/apis/chaincode/uninstall

请求参数

表 1-99 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 1-100 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
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

参数	是否必选	参数类型	描述
target_peers	是	Array of **TargetPeer** objects	卸载链代码peer信息

表 1-101 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 1-102 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 1-103 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**。

1.5.3.16 下载报告

功能介绍

下载链代码安全检查报告

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v2/agent/apis/chaincode/report

请求参数

表 1-104 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 1-105 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 1-106 响应 Body 参数

参数	参数类型	描述
-	File	

状态码： 400

表 1-107 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**。

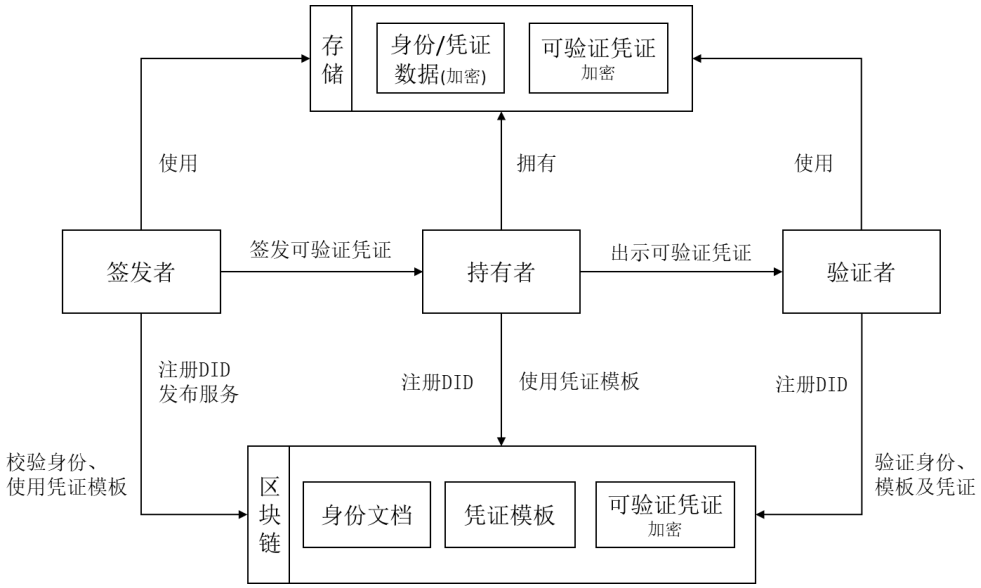
1.5.4 分布式身份（公测）

1.5.4.1 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1-11**、**图1-12**、**图1-13**和**图1-14**。

图 1-11 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

   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。

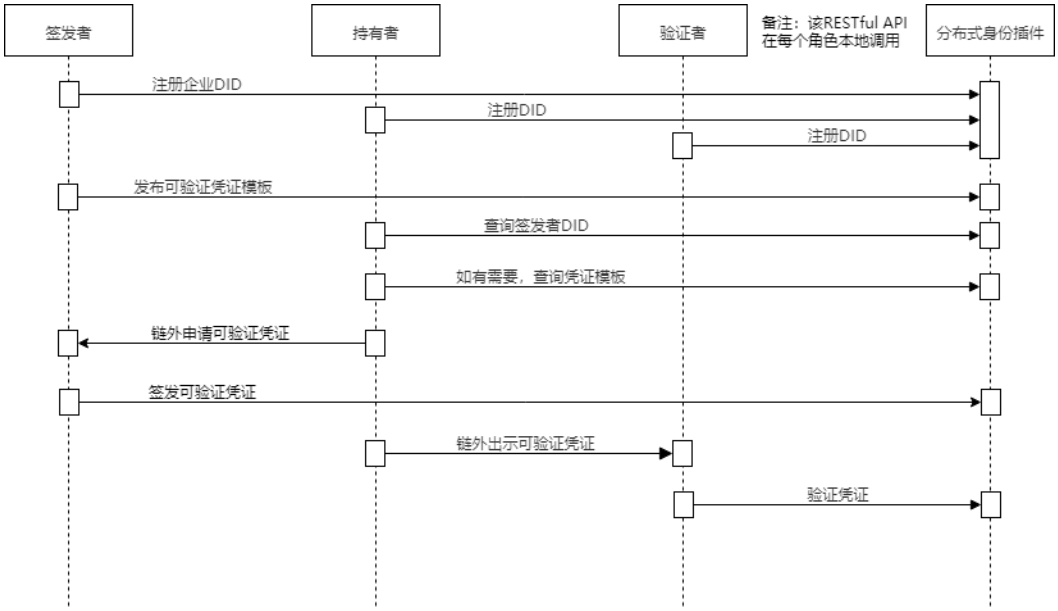
说明

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**《区块链服务用户指南》常见问题中的“获取fabric用户私钥及证书的方法”章节。

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 1-12 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 1-13 分布式身份使用时序图(链上申请-在线申请模式)

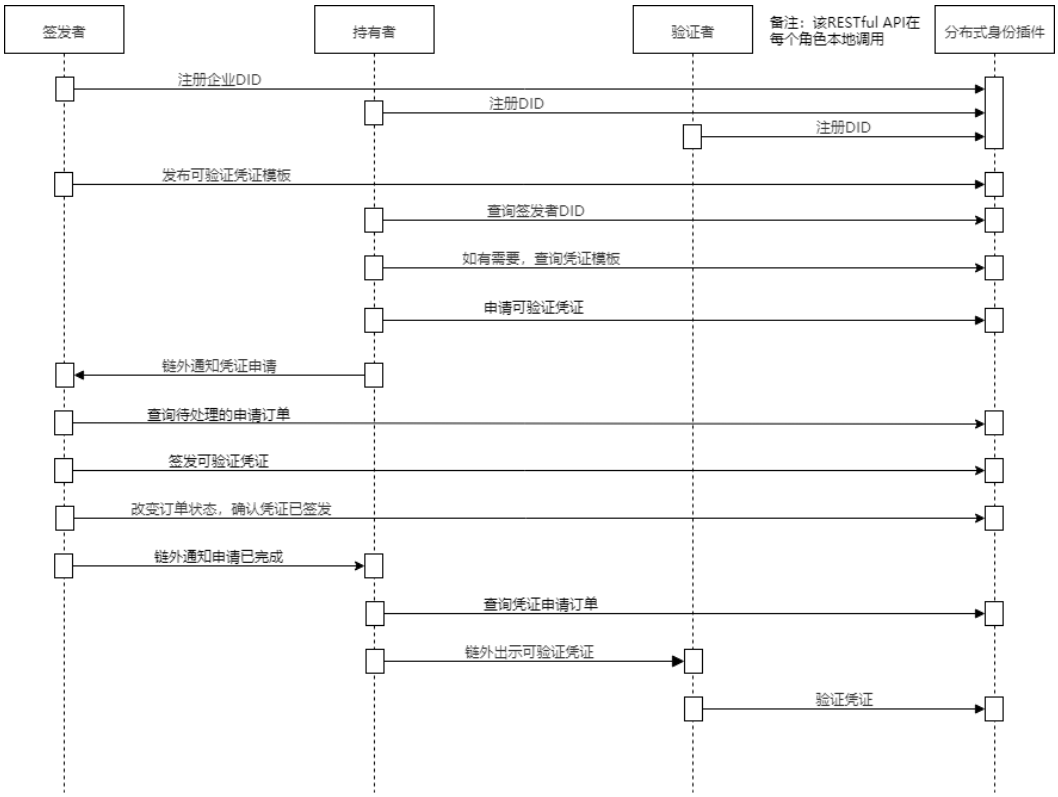
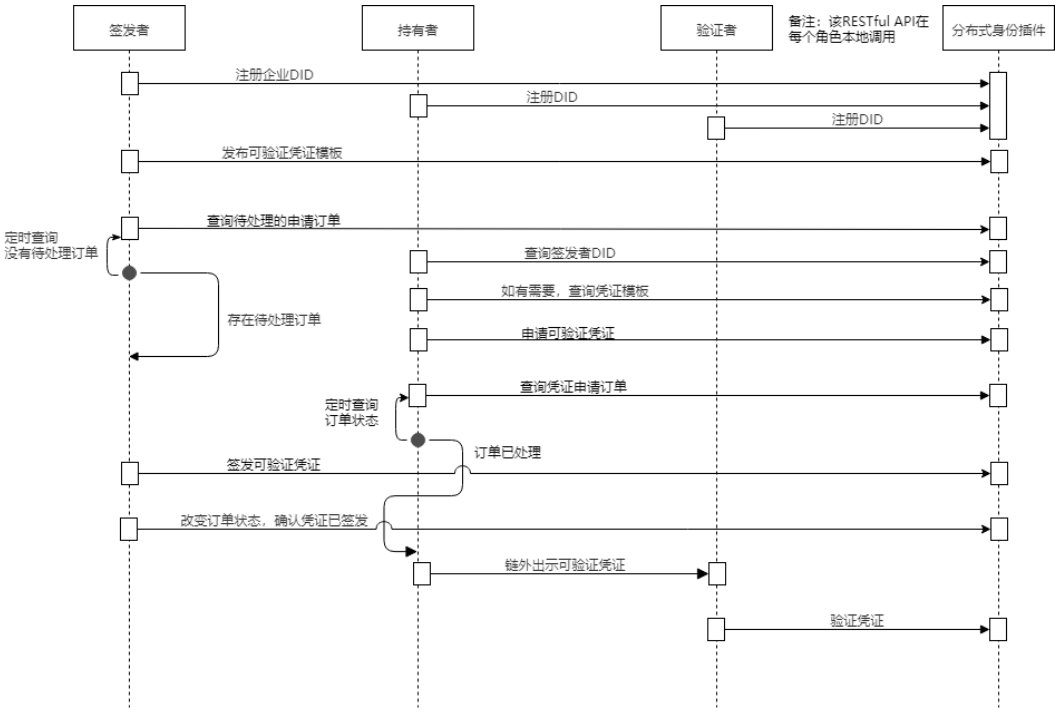


图 1-14 分布式身份使用时序图(链上申请-离线申请模式)



1.5.4.2 分布式身份(DID)管理

1.5.4.2.1 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。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/firm-did

请求参数

表 1-108 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

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	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 1-109 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 1-110 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 1-111 Attribute

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

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

响应参数

状态码： 200

表 1-112 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 1-113 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",
  "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**。

1.5.4.2.2 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。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/did

请求参数

表 1-114 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	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 1-115 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 1-116 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...\n-----END CERTIFICATE-----",
  "sk" : "------BEGIN PUBLIC KEY-----\n...\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: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**。

1.5.4.2.3 更新 DID

功能介绍

更新DID文档中发布的服务。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

PUT /v1/identity/did

请求参数

表 1-117 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	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 1-118 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 1-119 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 1-120 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 1-121 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 1-122 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 1-123 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 1-124 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 1-125 Attribute

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

表 1-126 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 1-127 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",
  "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**。

1.5.4.2.4 查询 DID

功能介绍

查询指定DID的文档

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/query-did

请求参数

表 1-128 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

响应参数

状态码： 200

表 1-129 响应 Body 参数

参数	参数类型	描述
context	String	context

参数	参数类型	描述
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 1-130 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 1-131 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 1-132 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 1-133 Attribute

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

表 1-134 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 1-135 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**

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" : "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",
  "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**。

1.5.4.3 可验证凭证(VC)管理

1.5.4.3.1 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。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/credential-schema

请求参数

表 1-136 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

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
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 1-137 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 1-138 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
credentialSchema	**CredentialSchema** object	CredentialSchema

表 1-139 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

参数	参数类型	描述
attributes	Array of **Attribute** objects	属性信息
version	Integer	版本

表 1-140 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 1-141 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",
  "title" : "string",
  "identifier" : "string",
  "attributes" : [ {
    "name" : "name",
    "type" : "someType",
    "description" : "Identity number"
  } ],
  "issuer" : "did:example:ebfeb1f712ebc6f1c276e12ec21"
}
```

响应示例

状态码： 200

VCSchemaResponeParams

```
{
  "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	VCSchemaResponeParams
500	失败响应

错误码

请参见**错误码**。

1.5.4.3.2 查询凭证模板

功能介绍

根据索引查询凭证模板。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/query-credential-schema

请求参数

表 1-142 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 1-143 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 1-144 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 1-145 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...\n-----END CERTIFICATE-----",
  "sk" : "------BEGIN PRIVATE KEY-----\n...\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

状态码	描述
500	失败响应

错误码

请参见**错误码**。

1.5.4.3.3 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。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/apply-vc

请求参数

表 1-146 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
applyer	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 1-147 响应 Body 参数

参数	参数类型	描述
orderIndex	String	订单索引

状态码： 500

表 1-148 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...\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",
  "serviceID" : "did:example:ebfeb1f712ebc6f1c276e12ec21#service1",
  "data" : "abcdefg"
}
```

响应示例

状态码： 200

VCOrderResponeParams Information

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

状态码： 500

失败响应

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

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

状态码

状态码	描述
200	VCOrderResponeParams Information
500	失败响应

错误码

请参见**错误码**。

1.5.4.3.4 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。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/vc-order

请求参数

表 1-149 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
orderIndex	是	String	订单索引
vcIndex	是	String	凭证索引

响应参数

状态码： 200

表 1-150 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 1-151 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",
  "orderIndex" : 1,
  "vcIndex" : 0
}
```

响应示例

状态码： 200

ConfirmOrderResponeParams 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	ConfirmOrderResponeParams Information
500	失败响应

错误码

请参见**错误码**。

1.5.4.3.5 查询凭证申请订单

功能介绍

查询凭证的申请订单

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/query-vc-order

请求参数

表 1-152 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 1-153 响应 Body 参数

参数	参数类型	描述
applyer	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
vcIndex	String	凭证索引

状态码： 500

表 1-154 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",
  "orderIndex" : 1
}
```

响应示例

状态码： 200

VCOrder 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	VCOrder Information
500	失败响应

错误码

请参见**错误码**。

1.5.4.3.6 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

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/query-vc-orders

请求参数

表 1-155 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 1-156 响应 Body 参数

参数	参数类型	描述
items	Array of **VCOrder** objects	列表

表 1-157 VCOrder

参数	参数类型	描述
applyer	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

参数	参数类型	描述
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
vcIndex	String	凭证索引

状态码： 500

表 1-158 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",
  "orderIndex" : 1
}
```

响应示例

状态码： 200

UntreatedVCOrder Information List

```
{
  "items" : [ {
    "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	UntreatedVCOrder Information List
500	失败响应

错误码

请参见**错误码**。

1.5.4.3.7 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。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/issue-vc

请求参数

表 1-159 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	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
credentialInfo	是	CredentialSubjectInfo object	凭证主题信息

表 1-160 CredentialSubjectInfo

参数	是否必选	参数类型	描述
applyer	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 1-161 响应 Body 参数

参数	参数类型	描述
vcIndex	String	凭证索引

状态码： 500

表 1-162 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" : "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:fdsafre767f8a3hr773j4h1jkhr",
  "sequence" : "10025469331",
  "schemaIndex" : "did:example:ebfeb1f712ebc6f1c276e12ec21_IDCard",
  "data" : "{\"name\": \"xm\",\"age\": 18}"
}
```

响应示例

状态码： 200

VCResponeParams Information

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

状态码： 500

失败响应

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

状态码

状态码	描述
200	VCResponeParams Information
500	失败响应

错误码

请参见**错误码**。

1.5.4.3.8 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

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/query-vc

请求参数

表 1-163 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 1-164 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

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

表 1-165 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 1-166 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 1-167 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",
  "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" : "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b85"
  },
  "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**。

1.5.4.3.9 验证凭证

功能介绍

验证可验证凭证是否存在以及合法性。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/identity/verify-vc

请求参数

表 1-168 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 1-169 响应 Body 参数

参数	参数类型	描述
isOwned	Boolean	是否拥有

状态码： 400

表 1-170 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 1-171 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",
  "vcIndex" : 1,
```

```
  "owner" : "did:example:ebfeb1f712ebc6f1c276e12ec21"
}
```

响应示例

状态码： 200

VCVerifyResponeParams 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	VCVerifyResponeParams Information
400	失败响应
500	失败响应

错误码

请参见**错误码**。

1.5.5 可信数据交换（公测）

1.5.5.1 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。

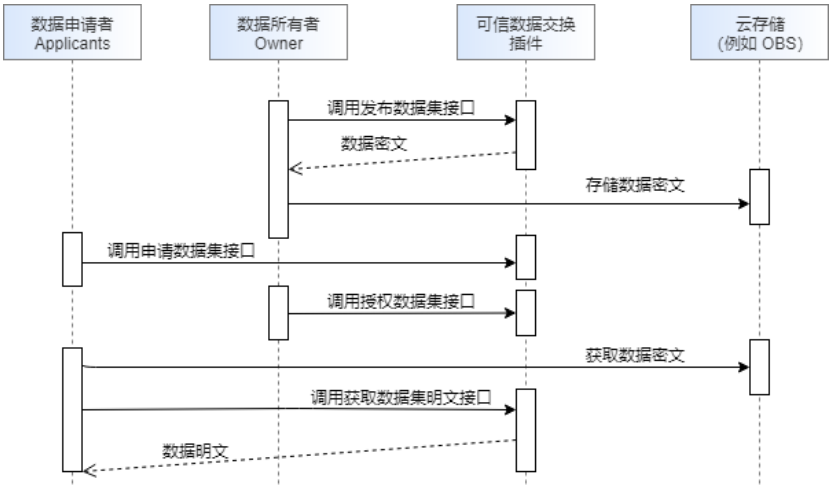
角色定义

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1-15**。
  - 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 1-15 申请-授权模式使用流程

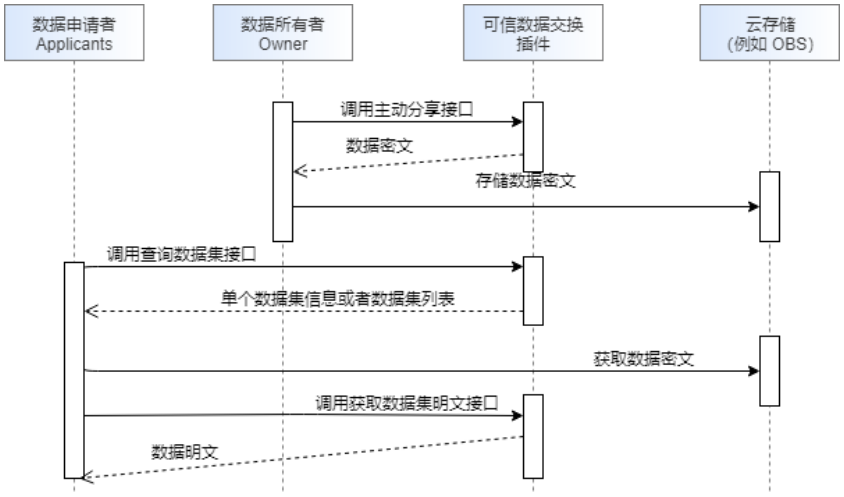
  

- 主动分享模式，使用流程请参见**图1-16**。

  “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 1-16 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1-17。

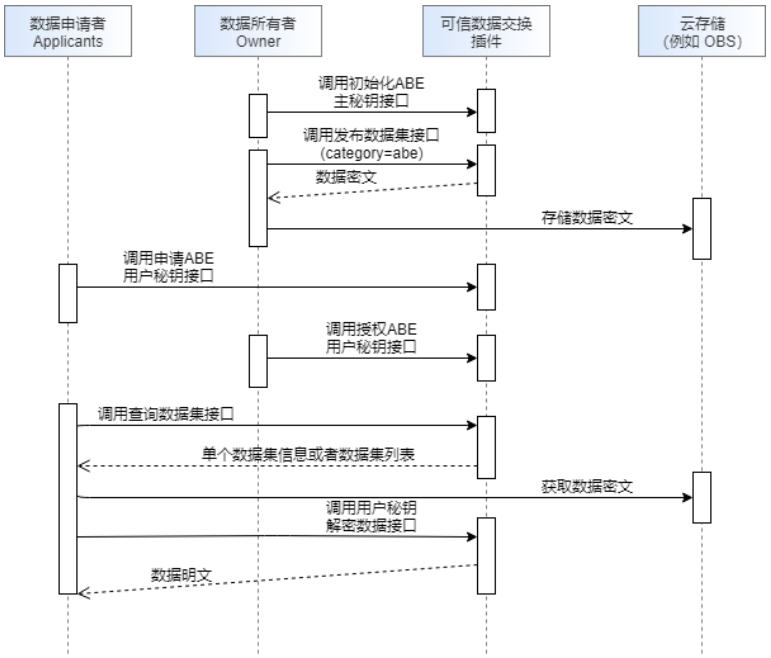
  说明

  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 1-17 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）：数据拥有者所有，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。

1.5.5.2 数据集管理

1.5.5.2.1 发布数据集

功能介绍

发布数据集。水印功能只支持“华北-北京四”区域

调试

您可以在API Explorer中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/dataset

请求参数

表 1-172 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

参数	是否必选	参数类型	描述
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
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 1-173 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 1-174 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 1-175 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 1-176 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

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 1-177 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 1-178 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

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:DHkJjyD5wZwya6sd6BNBnG",
  "providerName" : "test",
  "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:DHkJjyD5wZwya6sd6BNBnG",
  "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+qvk6ZH0c7wP0aS48fSoNtecG79aFpFx0dg7rFdVYXWWzgeyI03eD3gFdXlQ/
ovpxKJG5ALK39OCazUqDrawZHSDGyIlw0hGh88Q+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**。

1.5.5.2.2 删除数据集

功能介绍

删除数据集

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

DELETE /v1/datashare/dataset

请求参数

表 1-179 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	数据集提供者身份标识
productID	是	String	数据产品id

响应参数

状态码： 200

表 1-180 响应 Body 参数

参数	参数类型	描述
result	String	操作结果

状态码： 500

表 1-181 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:DHkJjyD5wZwya6sd6BNBnG",
```

```
  "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**。

1.5.5.2.3 关闭数据集

功能介绍

关闭数据集

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

PUT /v1/datashare/dataset

请求参数

表 1-182 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 1-183 响应 Body 参数

参数	参数类型	描述
result	String	操作结果

状态码： 500

表 1-184 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:DHkJjyD5wZwya6sd6BNBnG",
```

```
  "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**。

1.5.5.2.4 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。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/query-dataset

请求参数

表 1-185 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 1-186 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

参数	参数类型	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 1-187 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 1-188 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",
  "provider" : "did:example:DHkJjyD5wZwya6sd6BNBnG",
  "productID" : "product2"
}
```

响应示例

状态码： 200

操作结果

```
{
  "provider" : "did:example:DHkJjyD5wZwya6sd6BNBnG",
  "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+qvk6ZH0c7wP0aS48fSoNtecG79aFpFx0dg7rFdVYXWWzgeyI03eD3gFdXlQ/
ovpxKJG5ALK39OCazUqDrawZHSDGyIlw0hGh88Q+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**。

1.5.5.2.5 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

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/query-datasets

请求参数

表 1-189 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
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 1-190 响应 Body 参数

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

表 1-191 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
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

参数	参数类型	描述
sampleFiles	Array of **DataFile** objects	样例文件列表
category	String	加密类型

表 1-192 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 1-193 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 1-194 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:DHkJjyD5wZwya6sd6BNBnG",
    "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+qvk6ZH0c7wP0aS48fSoNtecG79aFpFx0dg7rFdVYXWWzgeyI03eD3gFdXlQ/
ovpxKJG5ALK39OCazUqDrawZHSDGyIlw0hGh88Q+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**。

1.5.5.2.6 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。水印功能只支持“华北-北京四”区域

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/dataset/share

请求参数

表 1-195 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

参数	是否必选	参数类型	描述
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。
file	否	String	加水印的文件，当对文件加水印时，plainData无效果
productIDKeywords	否	Array of **productIDKeywordsJson** 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

参数	是否必选	参数类型	描述
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 1-196 productIDKeywordsJson

参数	是否必选	参数类型	描述
value	否	String	关键字值

表 1-197 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 1-198 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 1-199 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 1-200 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“”
encryptData	String	base64编码的数据密文

状态码： 500

表 1-201 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

```
{
  "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:DHkJjyD5wZwya6sd6BNBnG",
  "providerName" : "test",
  "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:DHkJjyD5wZwya6sd6BNBnG",
  "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**。

1.5.5.2.7 获取数据解密后的明文

功能介绍

获取数据解密后的明文。水印功能只支持“华北-北京四”区域

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/dataset/query-plaintext

请求参数

表 1-202 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

参数	是否必选	参数类型	描述
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 1-203 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 1-204 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:DHkJjyD5wZwya6sd6BNBnG",
  "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**。

1.5.5.2.8 提取文件中的暗水印

功能介绍

提取文件中的暗水印。水印功能只支持“华北-北京四”区域

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/dataset/watermark/extract

请求参数

表 1-205 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 1-206 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 1-207 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**。

1.5.5.2.9 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

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/dataset/query-process

请求参数

表 1-208 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 1-209 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 1-210 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 1-211 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-process

{
  "orgID" : "{{orgID}}",
  "channelID" : "{{channelID}}",
  "cryptoMethod" : "{{cryptoMethod}}",
  "cert" : "{{cert}}",
  "sk" : "{{sk}}",
  "timestamp" : "{{timestamp}}",
  "creatorDID" : "did:example:8sAvsS4tB3NYgMJ4uqbVYj",
  "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**。

1.5.5.2.10 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

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/dataset/query-processes

请求参数

表 1-212 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
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 1-213 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 1-214 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 1-215 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 1-216 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 1-217 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

```
/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**。

1.5.5.3 数据订单管理

1.5.5.3.1 申请数据集

功能介绍

申请数据集

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/dataset/dataset-order

请求参数

表 1-218 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
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 1-219 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 1-220 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",
  "provider" : "did:example:DHkJjyD5wZwya6sd6BNBnG",
  "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:DHkJjyD5wZwya6sd6BNBnG",
  "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**。

1.5.5.3.2 授权数据集

功能介绍

授权数据集

调试

您可以在API Explorer中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/dataset/authorize-dataset

请求参数

表 1-221 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 1-222 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

参数	参数类型	描述
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 1-223 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

订单信息

```
{
  "consumer" : "did:example:3TMWx8owKHARgNwbj4ywmG",
  "consumerName" : "Tyler",
  "orderSeq" : "1",
  "provider" : "did:example:DHkJjyD5wZwya6sd6BNBnG",
  "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**。

1.5.5.3.3 修改订单状态

功能介绍

修改订单状态

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

PUT /v1/datashare/dataset/order

请求参数

表 1-224 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

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
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 1-225 响应 Body 参数

参数	参数类型	描述
result	String	操作结果

状态码： 500

表 1-226 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",
  "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**。

1.5.5.3.4 删除订单

功能介绍

删除订单

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

DELETE /v1/datashare/dataset/order

请求参数

表 1-227 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
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 1-228 响应 Body 参数

参数	参数类型	描述
result	String	操作结果

状态码： 500

表 1-229 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**。

1.5.5.3.5 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。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/dataset/query-order

请求参数

表 1-230 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
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 1-231 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 1-232 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:DHkJjyD5wZwya6sd6BNBnG",
  "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**。

1.5.5.3.6 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。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/dataset/query-orders

请求参数

表 1-233 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 1-234 响应 Body 参数

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

表 1-235 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 1-236 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 1-237 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",
  "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:DHkJjyD5wZwya6sd6BNBnG",
    "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" : ""
  } ],
  "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**。

1.5.5.4 属性加密的密钥管理

1.5.5.4.1 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。

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/abe-setup

请求参数

表 1-238 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
owner	是	String	密钥生成者的身份标识
keyManagerMode	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 1-239 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 1-240 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

```
{
  "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**。

1.5.5.4.2 更新 ABE 主密钥

功能介绍

更新ABE主密钥

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/abe-update

请求参数

表 1-241 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
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 1-242 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 1-243 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**。

1.5.5.4.3 查询 ABE 主密钥

功能介绍

查询ABE主密钥

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/abekey

请求参数

表 1-244 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
owner	是	String	密钥生成者的身份标识
keyManagerMode	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 1-245 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 1-246 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**。

1.5.5.4.4 申请 ABE 用户密钥

功能介绍

申请ABE用户密钥

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/abekey-order

请求参数

表 1-247 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
applyer	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 1-248 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 1-249 响应 Body 参数

参数	参数类型	描述
applyer	String	申请者的身份标识

参数	参数类型	描述
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
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 1-250 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”,
  "applyer": "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


```
{
  "applyer" : "did:example:Mb4SshJeN5ukWXkbMJK8xC",
  "provider" : "did:example:Mb4SshJeN5ukWXkbMJK8xC",
  "applyTime" : "1622166512",
  "status" : "ready",
  "attributesJson" : "{\"att1\": \"YXR0MW5hbWU=\",\"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**。

1.5.5.4.5 授权 ABE 用户密钥

功能介绍

授权ABE用户密钥

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

PUT /v1/datashare/abekey-order

请求参数

表 1-251 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
applyer	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 1-252 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 1-253 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
price	Integer	价格

参数	参数类型	描述
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 1-254 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",
  "applyer" : "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

```
{
  "applyer" : "did:hwid:mfqqdiW8V64JbPFgQsoiv",
  "applyerName" : "",
  "provider" : "did:hwid:FahQr32NgQZWjGRiCZc37C",
  "providerName" : "",
  "service" : "",
  "price" : 0,
  "applyTime" : "",
  "encryptedABEKey" : "",
  "status" : "ready",
  "reason" : "",
```

```
  "lockProof" : "",
  "attributesJson" : "[{\"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**。

1.5.5.4.6 查询 ABE 用户密钥申请

功能介绍

查询ABE用户密钥申请

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/query-abekey

请求参数

表 1-255 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

参数	是否必选	参数类型	描述
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 1-256 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
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 1-257 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

```
{
  "applyer" : "did:hwid:mfqqdiW8V64JbPFgQsoiv",
  "applyerName" : "",
  "provider" : "did:hwid:FahQr32NgQZWjGRiCZc37C",
  "providerName" : "",
  "service" : "",
  "price" : 0,
  "applyTime" : "1672985584",
  "encryptedABEKey" : "",
  "status" : "ready",
  "reason" : "",
  "lockProof" : "",
  "attributesJson" : "{\"att1\":\"YXR0MW5hbWU=\",\"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**。

1.5.5.4.7 ABE 用户密钥解密数据

功能介绍

ABE用户密钥解密数据

调试

您可以在**API Explorer**中调试该接口，支持自动认证鉴权。API Explorer可以自动生成SDK代码示例，并提供SDK代码示例调试功能。

URI

POST /v1/datashare/abe-decrypt

请求参数

表 1-258 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
applyer	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。
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 1-259 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 1-260 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",
  "applyer" : "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**。

1.6 附录

1.6.1 国密加密

1.6.1.1 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。

说明

国密加密仅适用于Fabric架构版本的区块链实例。

资源下载

表 1-261 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 • 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。

1.6.1.2 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
  ```

1.6.1.3 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

序号	包名
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

1.6.2 同态加密

1.6.2.1 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。

  说明

  同态加密仅适用于Fabric架构版本的区块链实例。

资源下载

表 1-262 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
• 同态加密链代码下载 IDChaincode.go • Chaincode库接口文件下载 api_ahe_cc.tar.gz		
须知 • 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。		

1.6.2.2 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

1.6.2.3 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： • 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。

Encypt

- **接口原型**

  func Encypt (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
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

参数名	类型	描述
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。

1.6.2.4 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, err 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。

1.6.2.5 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。

1.6.2.6 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. 校验证明有效性。

```
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
}
```

1.6.2.7 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          idendity 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")
  -I, --idcc string         idendity 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")
  -I, --idcc string         idendity 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())
```

1.6.2.8 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

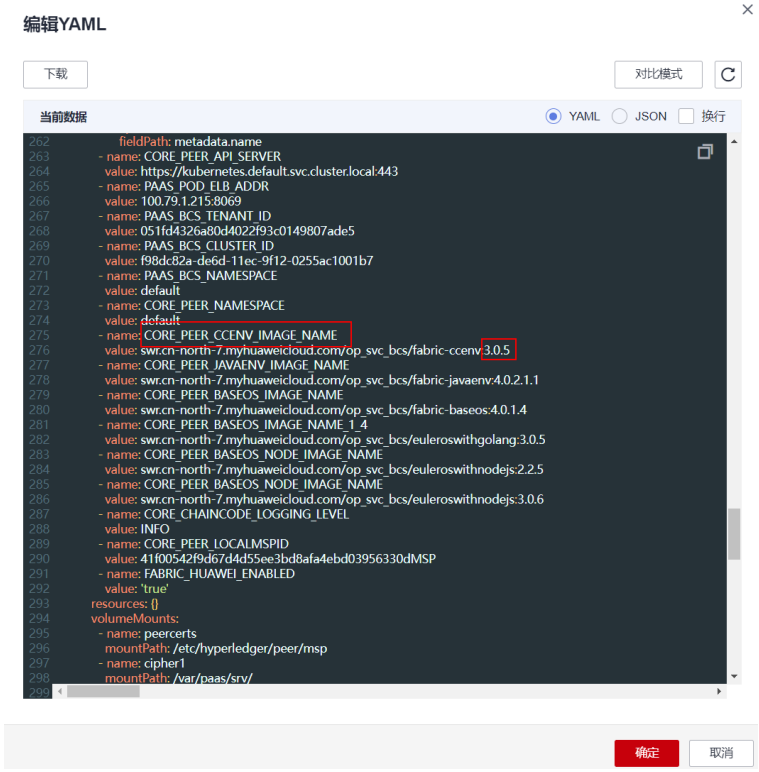
步骤1 订购BCS实例。

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**。

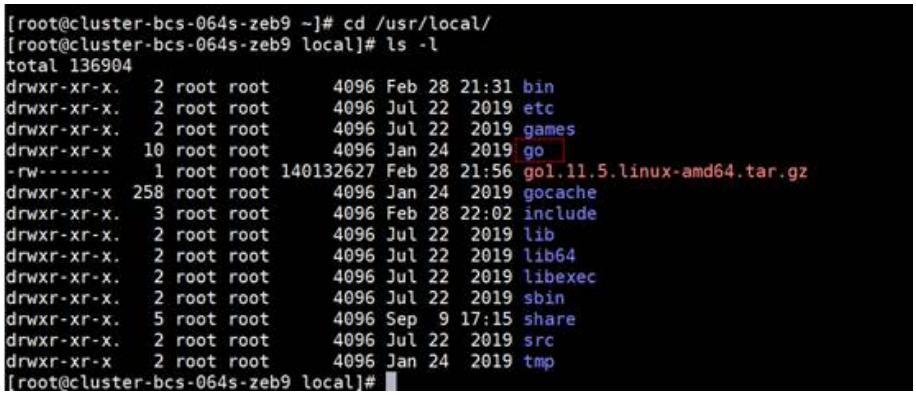
步骤4 在“实例管理”界面，在实例卡片中，单击“获取客户端配置”。

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/ahe/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/ahe/PSW/example/appdemo/”目录，执行go build，编译appdemo文件。

```
[root@cluster-bcs-064s-zeb9 appdemo]# cd /opt/gopath/src/ahe/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。

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。

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。

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-2cf8802066c1c5011fd396c54c91
26a17c9cfcc9.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
```

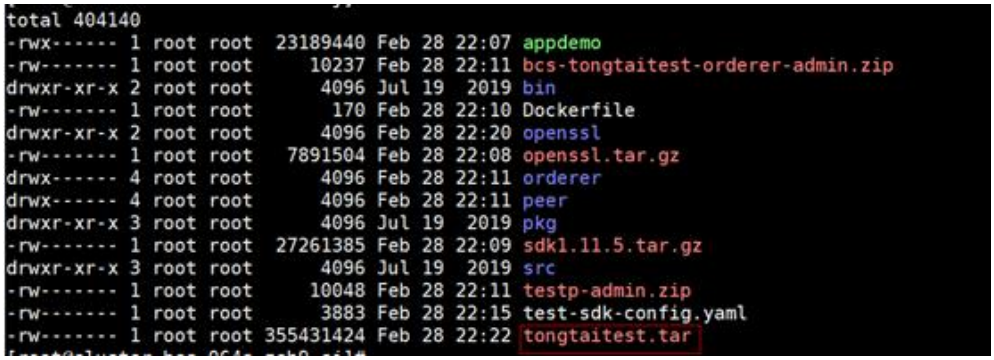
```
 ordererorg:
    mspID: "2cf8802066c1c5011fd396c54c9126a17c9cfcc9MSP"

    cryptoPath: /home/paas/orderer/msp
orderer-eip: 49.4.81.160
orderers:


orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9-0.orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9.default.svc.cluster.local:
  url: grpcs://49.4.81.160:30805

  grpcOptions:
    ssl-target-name-override:
orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9-0.orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9.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: grpcs://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>tongtaitest.tar
```

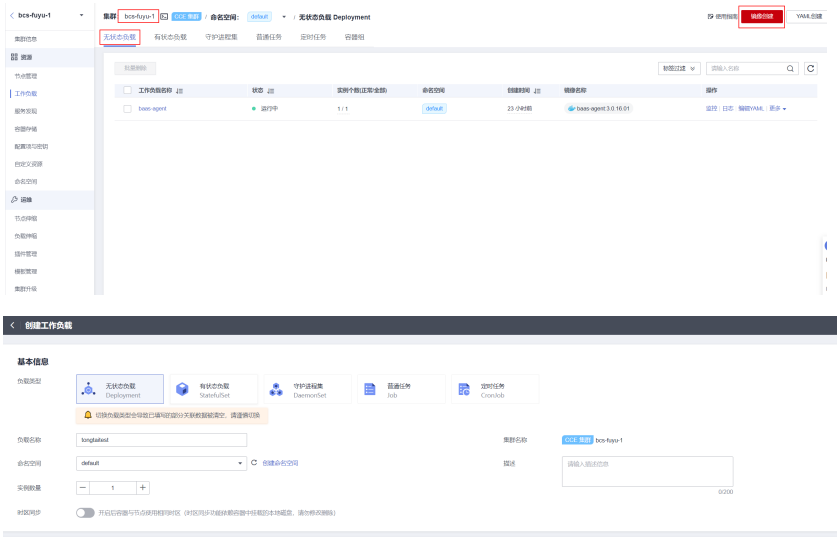


说明

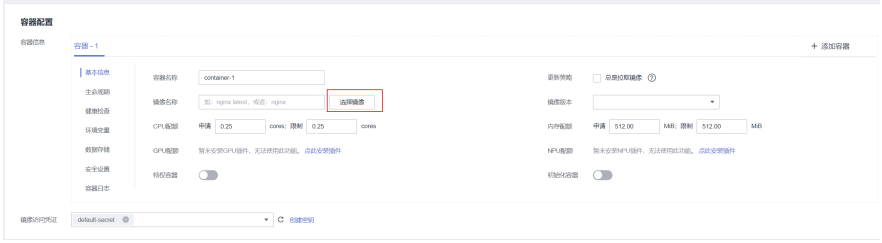
请确保本地有euleros：2.2.5镜像，否则将会打包失败。

步骤10 上传镜像。

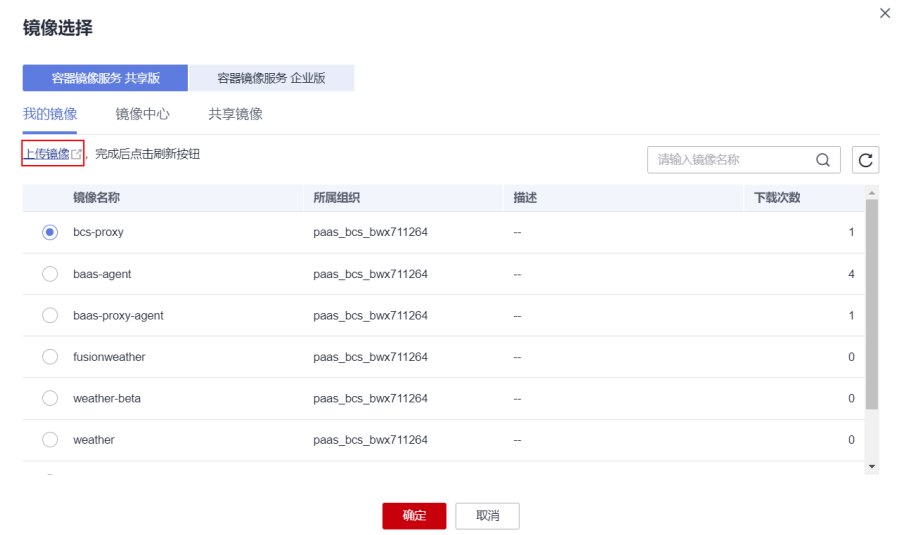
1. 将打包完成的镜像下载到本地，登录华为云，进入云容器引擎页面，在部署了BCS实例对应的集群下，创建无状态工作负载，实例数量选择1个。如果界面布局不一致，请在右上角切换新版cce界面。



2. 在“容器配置”页签，选择镜像。

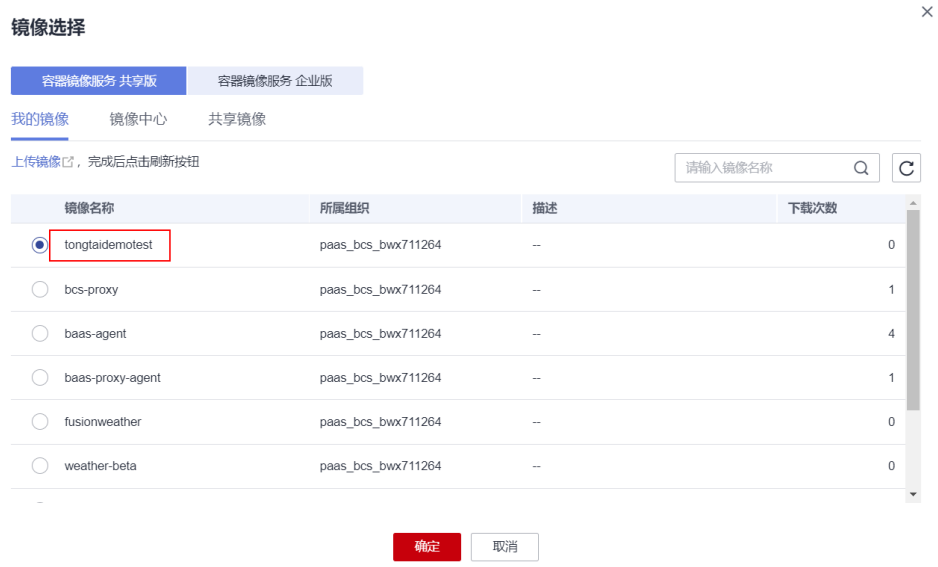


3. 单击“上传镜像”，跳转到总览页面，上传镜像。



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

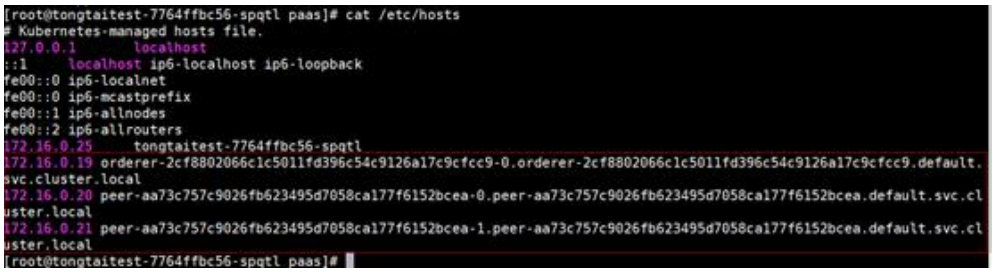
7. 提交后，无状态工作负载创建成功。

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
   ```

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：



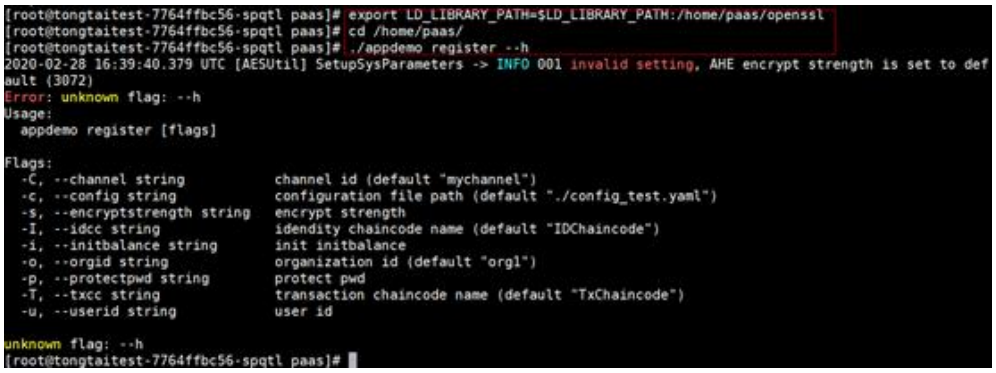

```
x.x.x.x
orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9-0.orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9.default.svc.cluster.local
x.x.x.x peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local
x.x.x.x peer-aa73c757c9026fb623495d7058ca177f6152bcea-1.peer-aa73c757c9026fb623495d7058ca177f6152bcea.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。



----结束

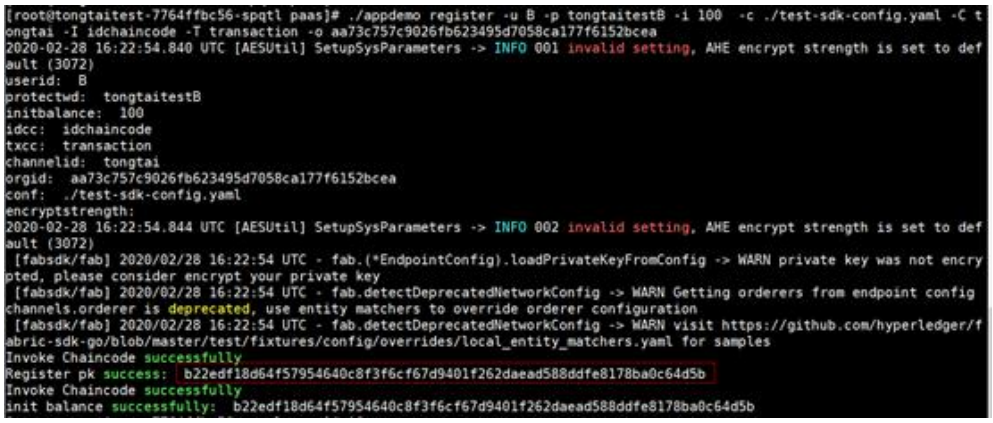
示例 1：注册账户

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。

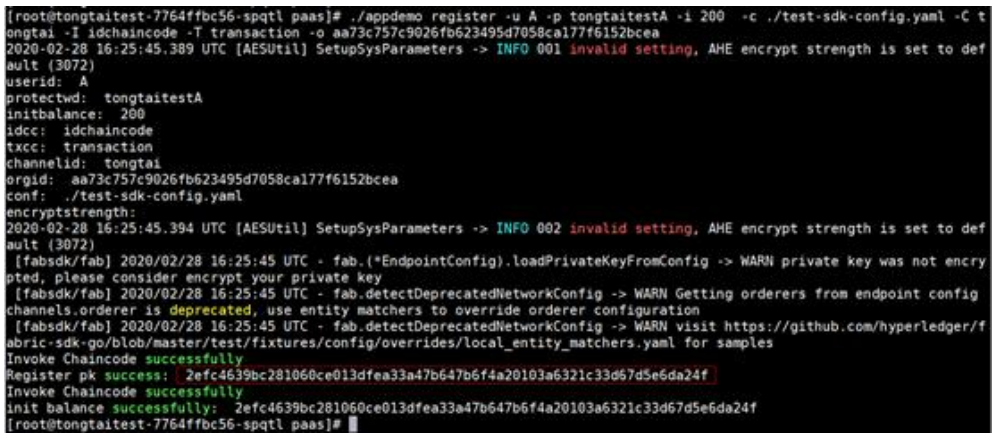


返回值为一串加密地址，示例：

```
b22edf18d64f57954640c8f3f6cf67d9401f262daead588ddfexxxxx
```

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
```



返回值为一串加密地址，示例：

```
2efc4639bc281060ce013dfea33a47b647b6f4a20103a6321c33dxxxxxx
```

----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 XXXXXXA -b b22edf18d64f57954640c8f3f6cf67d9401f262daead588ddfe8178xxxx -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-spqtl paas]# ./appdemo transaction -u A -p tongtaitestA -b b22edf18d64f57954640c8f3f6cf67d9
401f262daead588ddfe8178ba0c64d5b -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:  b22edf18d64f57954640c8f3f6cf67d9401f262daead588ddfe8178ba0c64d5b
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-spqtl paas]#
```

返回值为A账户的地址：
2efc4639bc281060ce013dfea33a47b647b6f4a20103a6321c33d67d5xxxx

----结束

示例 3：查询账户余额

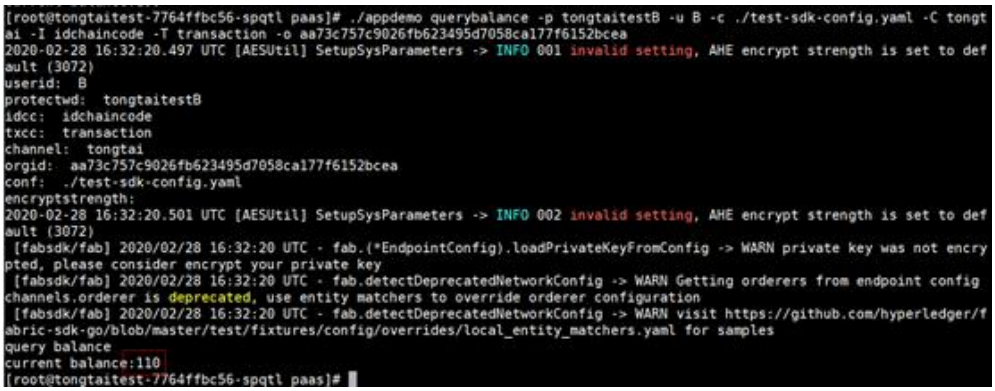
步骤1 执行如下命令查询A账户的余额，返回值为A账户余额。

```
./appdemo querybalance -p XXXXXX -u A -c ./test-sdk-config.yaml -C tongtai -I idchaincode -T transaction -
o aa73c757c9026fb623495d7058ca177f6152bcea
```

```
[root@tongtaitest-7764ffbc56-spqtl paas]# ./appdemo querybalance -p tongtaitestA -u A -c ./test-sdk-config.yaml -C tongt
ai -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 def
ault (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 def
ault (3072)
 [fabsdk/fab] 2020/02/28 16:31:39 UTC - fab.(*EndpointConfig).loadPrivateKeyFromConfig -> WARN private key was not encry
pted, 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/f
abric-sdk-go/blob/master/test/fixtures/config/overrides/local_entity_matchers.yaml for samples
query balance
current balance:190
[root@tongtaitest-7764ffbc56-spqtl paas]#
```

步骤2 执行如下命令查询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
```

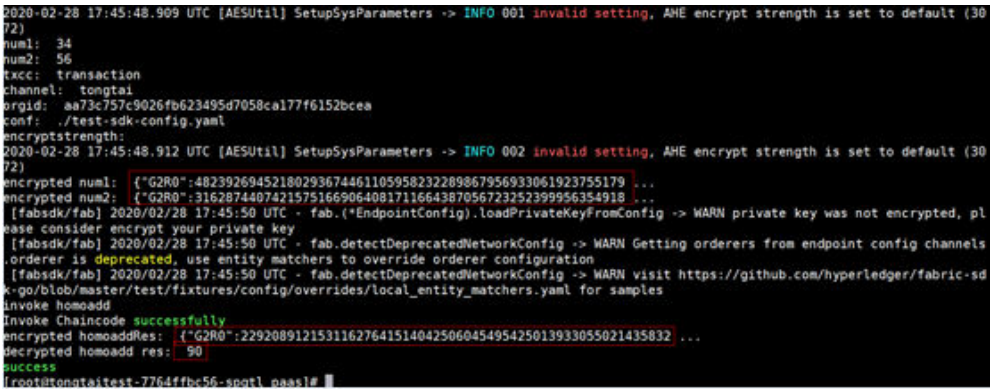


----结束

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
```



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:  a1e954cd9c712864130acaf9c63906d06e15877f
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 -> WAR
 [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

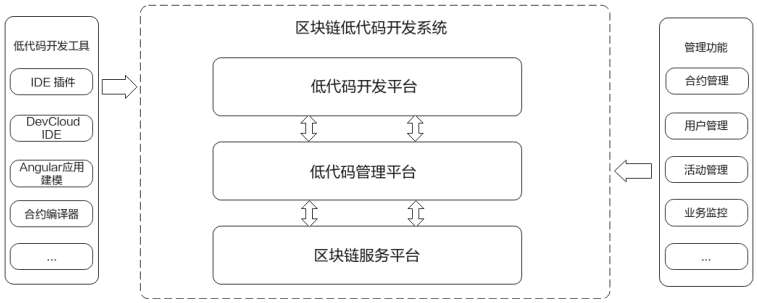
1.6.3 区块链应用低代码开发功能

1.6.3.1 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同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 1-18 系统逻辑



说明

若需要使用该功能，请联系技术支持工程师。

1.6.3.2 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。

1.6.3.3 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进行筛选过滤，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）。

1.6.3.4 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。

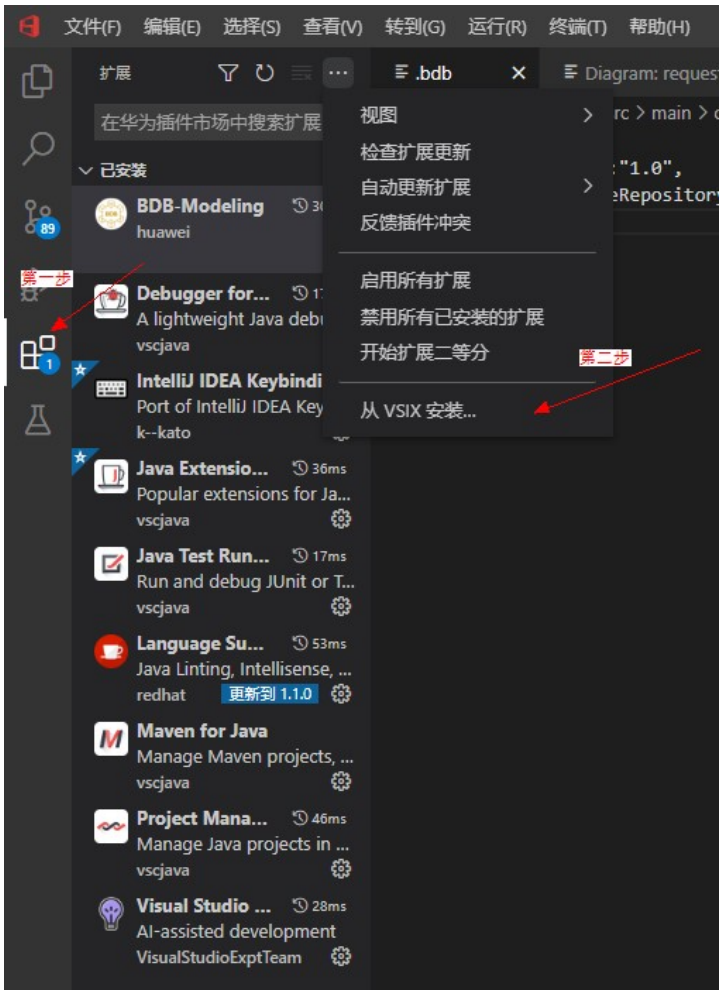
1. 登录云容器引擎CCE控制台。
2. 创建CCE集群，购买Ubuntu系统CPU16核内存32G机器。
3. 选择集群节点，绑定弹性公网IP并设置节点安全组规则，添加入方向规则TCP 9096端口，以及开放Kubernetes服务端口，设置为1-32767。
4. 将License放入虚机目录/下，解压工具包BCS-BPMN.zip放至/root目录，运行一键部署脚本/root/BCS-BPMN/.build_config/one_step_deploy.sh。

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。

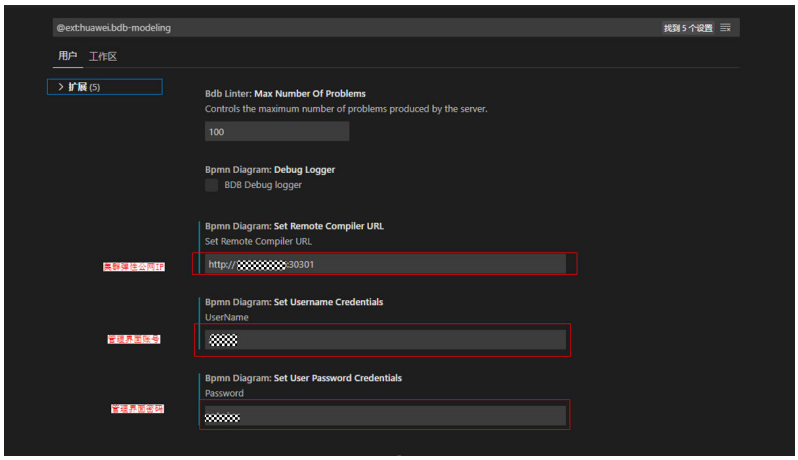
步骤3 安装工具包。

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。



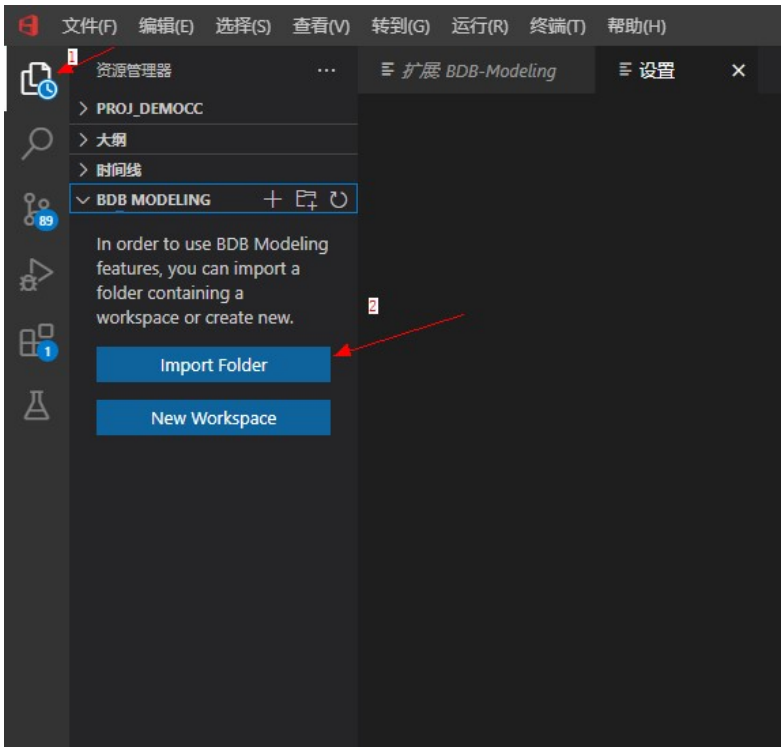
步骤4 配置插件信息。

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。



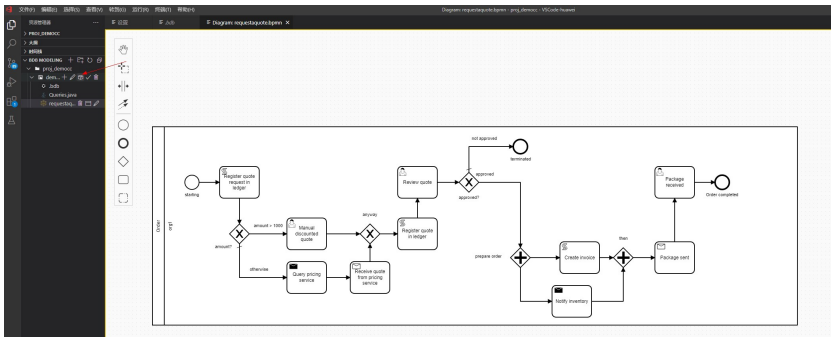
步骤5 导入示例demo。

单击资源管理器 -> 选择BDB MODELING -> Import Folder -> 选择BCS-BPMN\demo\projects\democc\modeller-workspace\proj_democc。



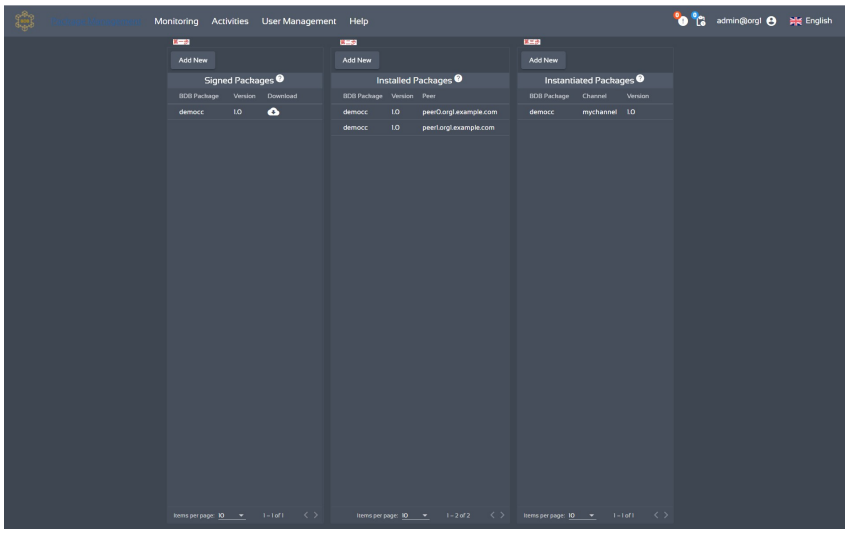
步骤6 编译bpmn，生成合约。

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。

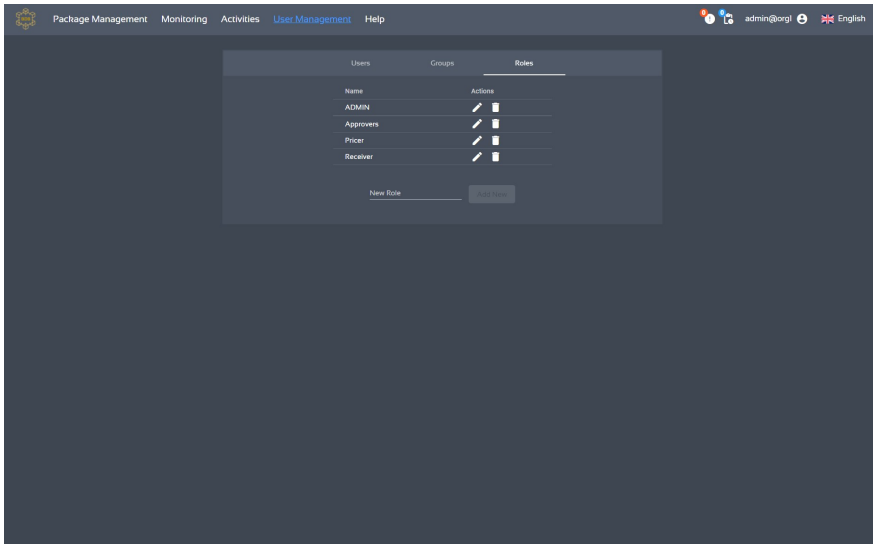


步骤8 设置角色、群组和用户。

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）。

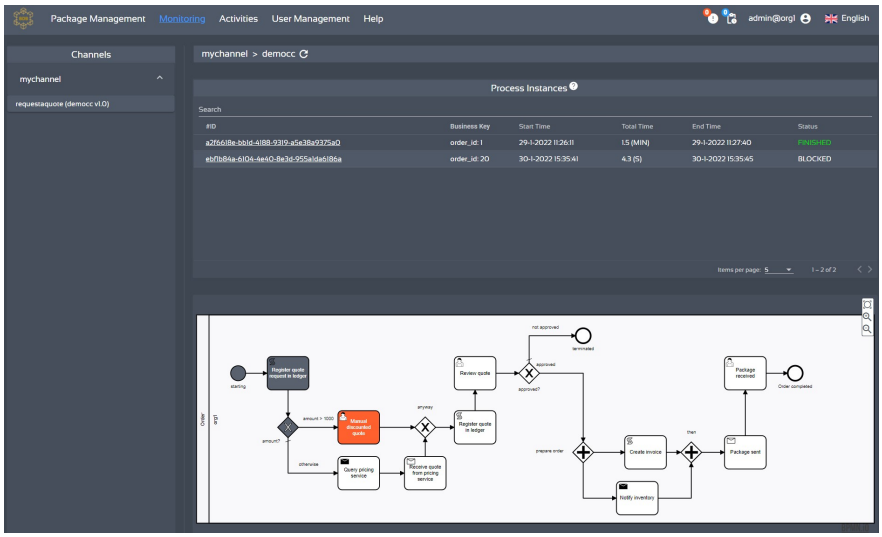
说明

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

1.6.4 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。
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。

状态码	错误码	错误信息	描述	处理措施
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.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，可能原因为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.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，可能原因为CCE集群异常或者ETCD连接问题	登录CCE控制台，查看CCE服务是够正常；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ECTD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ECTD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，可能原因为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.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，集群无可用节点，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，恢复正常后再重试。
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。

状态码	错误码	错误信息	描述	处理措施
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。
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。

状态码	错误码	错误信息	描述	处理措施
400	BCS.4001104	Failed to delete the deployment template because the CCE cluster status may be abnormal.	删除失败，部署模板删除失败，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，部署模板删除超时，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。
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。

状态码	错误码	错误信息	描述	处理措施
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服务是够正常；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服务是够正常；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.	保存部署模板失败，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.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.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.	账号尚未进行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.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.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。

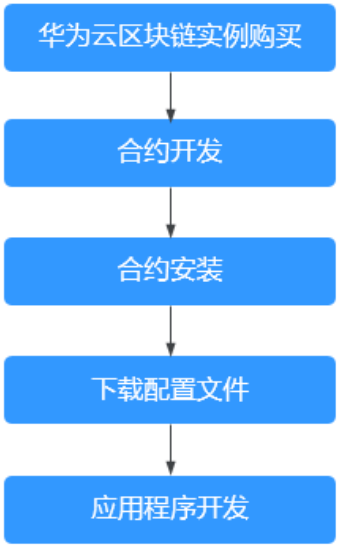
状态码	错误码	错误信息	描述	处理措施
400	BCS.4009100	System error. Check the system or network status.	系统错误，请检查系统或网络状态是否正常	将界面从BCS服务控制台切换至CCE等其他服务界面查看，若多个服务都出现系统异常，则是环境整体公共组件类的故障导致的，等故障恢复后重试，若是只有BCS服务出现异常，则可能是BCS服务后台节点发生了故障。参考应急预案中关于“BCS服务节点故障”指导进行处理。
400	BCS.4009101	The request URL does not exist. Check whether the 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

2 华为云区块链引擎管理

2.1 简介

本指导文档主要针对具备Go/Java开发经验的人员进行开发指导，其中合约与应用程序需客户自行开发，整体开发流程如下：



本文档主要包含以下内容：

- **合约开发**，Go语言、Java合约、Solidity合约。
- **SDK介绍**，主要介绍Java、Golang语言SDK。
- **应用程序开发**，介绍Java、Golang语言客户端开发流程与示例Demo。

说明

开发流程中的其他操作，请参考：

- **华为云区块链实例购买**
- **合约安装**
- **下载配置文件**

2.2 合约开发

2.2.1 概述

合约主要用于操作账本上的数据。作为运行在区块链上的、特定条件下自动执行的代码逻辑，合约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，合约的运行结果是可信的，其结果是无法被伪造和篡改的。

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。
- 为了确保代码在不同用户之间的一致性运行，华为云区块链引擎服务参考了成熟的开源社区方案（如Hyperledger Fabric）。用户在部署智能合约时，使用预配置的容器镜像进行处理，通过预配置的开发/编译工具（例如：javac、cpp、gcc等）减少环境差异带来的问题，确保智能合约能够在区块链网络上正确运行。华为云区块链引擎服务通过资源隔离等方式，降低了由此带来的安全风险。
- 为了保障链上数据的机密性，建议采用以下手段进行防护：
  - 数据加密：对链上存储的数据进行加密，只允许授权的参与方解密和查看数据。常用的加密方法包括对称加密和非对称加密。
  - 访问控制：实施严格的访问控制机制，确保只有被授权的用户或节点才能访问敏感数据。例如：不接入公网或使用VPC安全组等网络隔离手段限制访问。
  - 链下数据存储：对于特别敏感的数据，可以采用链下存储的方式，将数据存储在链下，并在区块链上记录其哈希值等摘要信息，以便在需要时验证数据的一致性和完整性。

基于合约SDK开发合约的流程如下：

步骤1 下载对应的合约SDK文件。

步骤2 配置SDK文件到本地合约项目。

步骤3 基于SDK提供的库函数进行合约开发。

----结束

华为云区块链提供Go、Solidity类型的合约示例供开发者使用，示例下载和开发参考如下：

表 2-1 华为云区块链引擎不同合约类型开发指南

合约类型	合约SDK下载	开发指南
Go	链接	参考文档
Solidity	不涉及	参考文档
Java	<dependency> <groupId>com.huawei.huaweichain</groupId> <artifactId>contract-sdk</artifactId> <version>0.2.2</version> </dependency>	参考文档

2.2.2 Go 合约开发

2.2.2.1 SDK 配置

表 2-2 Go 类型的合约 SDK

合约类型	SDK下载	备注
Go	链接	具体使用可参考合约示例和合约安装。

2.2.2.2 SDK 接口

合约SDK提供如下API接口，可以在合约文件中进行调用。这些API按照功能可以划分为：

表 2-3 stub 接口

接口	说明
FuncName() string	获取智能合约请求中指定的智能合约函数名称。
Parameters() [][]byte	获取请求参数。
ChainID() string	获取智能合约所在链ID。
ContractName() string	获取智能合约名称。
TxTimestamp() time.Time	获取本次交易的时间戳。

表 2-4 ContractStub 接口

接口	说明
GetKV(key string) ([]byte, error)	功能：获取状态数据库中某个key对应的value。 入参：某个键值对的key信息，不可为空。 返回值：返回[]byte类型的value值；当key不存在时，value为nil。 error：当网络出错，状态数据库出错，返回error信息。
PutKV(key string, value []byte) error	功能：写状态数据库操作，将key、value形成写集，打包到交易中，当交易排序、出块、并校验通过之后，将key/value写入到状态数据库中。入参：要写入的键值对要求key != ""，并且value != nil。 error：入参错误。

接口	说明
PutKVCommon(key string, value interface{}) error	功能：写状态数据库操作，与PutKV功能相同；与PutKV接口的不同之处在于 value不是[]byte类型，而是一个实现了Marshal(v interface{}) ([]byte, error)接口的数据，接口内部，会将value通过Marshal接口序列化，然后再形成写集。 入参：要写入的键值对，要求key != ""，并且value实现了Marshal接口，可以序列化为[]byte。 error：入参错误。
DelKV(key string) error	功能：删除状态数据库中的key及其对应的value，此接口只是将待删除的key放入写集，打包到交易中，当交易排序、出块、并校验通过之后，将key删除。 入参：要删除的key要求key != ""。 error：入参错误。
GetIterator(startKey, endKey string) (Iterator, error)	功能：查询状态数据库中，按字典序，以startKey开头，以endKey结尾的所有状态数据，结果以迭代器的形式呈现；查询范围是左闭右开的，[startKey, endKey)。 入参：startKey是待查询状态数据的按字典序的起始key，startKey != ""，endKey是待查询的状态数据的按字典序的结束key，endKey!= ""。 返回值：Iterator是查询结果的迭代器，可以通过此迭代器，按顺序读取查询结果。 error：入参或网络错误。
GetKeyHistoryIterator(key string) (HistoryIterator, error)	功能：查询一个key对应的所有历史的value 入参：key是待查询历史value值的key信息，key != ""。 返回值：HistoryIterator是按顺序返回包含历史value结果的迭代器结构体变量。 error：入参或网络错误。
SaveComIndex(indexName string, attributes []string, objectKey string) error	功能：为objectKey保存索引信息，indexName_attributes_objectKey构成索引信息，注意，此处只是形成索引信息的写集，只有当含有此写集的交易经过排序、出块，并校验通过后，才会写入状态数据库。 入参：indexName 索引标记，indexName != ""，attributes需要当做索引的属性，至少包含一个属性信息，objectKey 待索引的key值，objectKey != ""。 error：入参错误。

接口	说明
GetKVByComIndex(indexName string, attributes []string) (Iterator, error)	功能：通过索引信息，查找满足某种查询条件的key/value，key/value以迭代器的形式输出。 入参：indexName 索引标记，indexName != ""，attributes需要当做索引的属性，至少包含一个属性信息 返回值：满足索引条件的key/value的迭代器变量。 error：入参或网络错误。
DelComIndexOneRow(indexName string, attributes []string, objectKey string) error	功能：删除objectKey的某个索引，indexName_attributes_objectKey构成索引信息，注意，此处只是形成索引信息的写集，只有当含有此写集的交易经过排序、出块，并校验通过后，才会写入状态数据库。 入参：indexName 索引标记，indexName != ""，attributes需要当做索引的属性，至少包含一个属性信息，objectKey 待索引的key值，objectKey != ""。 error：入参错误。
SplitComKey(comKey string) (string, []string, error)	功能：将查询到的复合键分离为objectkey和对应的attributes。 返回值：objectkey，和attributes字符串数组。
GetKVByPartialComKey(objectType string, attributes []string) (Iterator, error)	功能：部分复合键查询。 返回值：Iterator包含查询返回信息，支持迭代获取。

表 2-5 HistoryIterator 迭代器接口

接口	说明
Version() (uint64, int32)	获取当前迭代位置（某笔交易）的BlockNum 和 TxNum。
TxHash() []byte	获取当前迭代位置（某笔交易）的hash。
IsDeleted() bool	被查询的key，当前是否已经在状态数据库中被删除。
Timestamp() uint64	返回当前迭代位置（某笔交易）的时间戳。

2.2.2.3 合约结构

Go语言合约由合约文件及依赖包构成，包含包声明、依赖包导入、智能合约的结构体定义和方法定义。

说明

合约文件中，用户可自定义结构体以及合约函数。以下内容不可更改：

- package名：package usercontract
- 函数的签名：NewSmartContract()、Init(stub contractapi.ContractStub)、Invoke(stub contractapi.ContractStub)

合约的结构如下：

```
package usercontract

// 引入必要的包
import (
    "git.huawei.com/poissonsearch/wienerchain/contract/docker-container/contract-go/contractapi"
)

// 声明合约的结构体
type example01 struct {}

// 创建合约
func NewSmartContract() contractapi.Contract {
    return &example01{}
}

// 合约的初始化（Init）接口。将合约启动时，需要首先执行且只需要执行一次的逻辑放到此方法中。
func (e *example01) Init(stub contractapi.ContractStub) ([]byte, error) {
    // 编写时可灵活使用stub中的API
}

// 合约被调用（invoke）接口。将主要的合约执行逻辑，放到此方法内，供合约使用者调用。
func (e *example01) Invoke(stub contractapi.ContractStub) ([]byte, error) {
    // 编写时可灵活使用stub中的API
}
```

2.2.2.4 合约示例

须知

1. 合约开发需要使用go mod，因此请确保GO111MODULE为on、镜像源配置。请确保可正常访问**华为云镜像网站**，环境设置命令如下

   ```
   go env -w GO111MODULE=on
   go env -w GOPROXY=https://repo.huaweicloud.com/repository/goproxy/
   go env -w GONOSUMDB=*
   ```

2. Go合约SDK文件导入本地项目后，若路径变红，可在Go Land编译器中启用 go mod:

   File->Settings->GO->Go Module->勾选 Enable go modules integration

Go语言合约开发和调测可参考合约示例，使用步骤如下：

步骤1 单击链接获取Go合约示例文件[**链接**]。

步骤2 单击链接Go合约SDK文件[**链接**]。

步骤3 解压Go合约SDK文件，添加到Go合约示例目录。

----结束

示例目录如下：

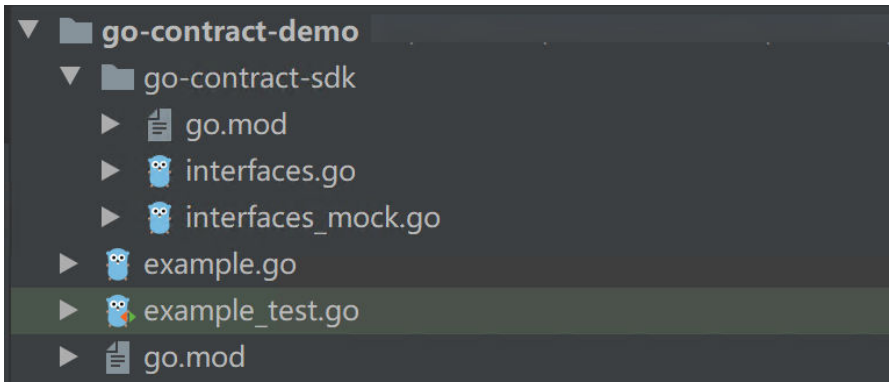


表 2-6 合约示例目录结构

目录	介绍
go-contract-sdk	包含Go合约SDK文件(interfaces.go)、SDK接口的mock，用于实现合约逻辑的接口。
example.go	用户自定义合约文件。示例文件以合约结构为基础，提供简单场景、复杂场景(弹珠游戏)两类合约操作函数。

2.2.2.5 合约安装

Go语言合约安装步骤如下：

步骤1 将已完成开发的合约文件，压缩成zip格式(可参考Go合约示例文件[链接]，示例文件可直接安装)。

步骤2 合约安装，可参考用户指南->**安装合约**。

----结束

说明

- 合约压缩文件格式：需确保格式为*.zip，且合约文件位于压缩包文件中的一级目录。
- 合约压缩文件中，禁止包含Go SDK文件，避免因包含Go SDK文件影响交易操作。

2.2.3 Wasm 合约开发（AssemblyScript）

2.2.3.1 合约结构

AssemblyScript语言合约主要包括index.ts和contract.ts两个文件，其中index.ts为开发智能合约文件(contract.ts)依赖的合约SDK，合约涉及的业务相关开发仅在contract.ts文件，智能合约文件contract.ts需要根据实际业务进行开发。

- 合约SDK(index.ts)主要结构如下：

```
// 引入智能合约文件
import { invoke, init } from "./contract";


// 合约的初始化（wasm_init）接口。包含合约文件的init()接口，合约启动时，需要首先执行且只需要执行一次的逻辑放到合约文件init()接口中。
```



```
export function wasm_init(buffer_offset: i32, size: i32):void{
    // 实际调用合约文件的init()接口
}

// 合约被调用（wasm_invoke）接口。包含合约文件的invoke()接口，供合约使用者通过SDK的wasm_invoke接口调用。
export function wasm_invoke(buffer_offset: i32, size: i32):void{
    // 实际调用合约文件的invoke()接口
}

// 合约被调用（wasm_prepare）接口，保持为空即可。
export function wasm_prepare():void{

}
```

- 智能合约文件(contract.ts)主要结构如下：

```
// 引入合约SDK方法
import { FuncName, smlog, Str2ArrayBuffer, Parameters, PutKV, ArrayBuffer2Str, GetKV, DelKV,
MakeErrRes,MakeSuccessRes, Response,IteratorNew, IteValue, IteKey, IteNext, IteratorFree } from "./index"


// 智能合约的初始化（init）接口的实现。
export function init(txid:string):Response{

}

// 智能合约被调用（invoke）接口的实现。
export function invoke(txid:string):Response{

}
```

2.2.3.2 合约相关的 API

合约SDK(index.ts)提供如下API接口，可以在合约文件中进行调用。这些API按照功能可以划分为：

表 2-7 辅助功能

接口	说明
FuncName(txid :string) :string	获取智能合约请求中指定的智能合约函数名称。
Parameters(txid: string):Array<ArrayBuffer>	获取请求参数。

表 2-8 账本数据操作

接口	说明
GetKV(txid:string, key:string):ArrayBuffer	获取某个键对应的值。
PutKV(txid:string, key:string, value:ArrayBuffer):void	添加或更新一对键值。
IteNext(itor : i64):boolean	返回当前迭代器指针是否存在下一个指针。

接口	说明
DelKV(txid:string, key:string):i32	删除一对键值。
IteratorFree(itor : i64):i32	释放迭代器指针。
IteratorNew(txid:string, beginKey:string, endKey:string) :i64	新建范围为[beginKey, endKey]的迭代器。
IteKey(itor : i64):string	从迭代器中获取key。
IteValue(itor : i64):ArrayBuffer	返回迭代器指向的值。

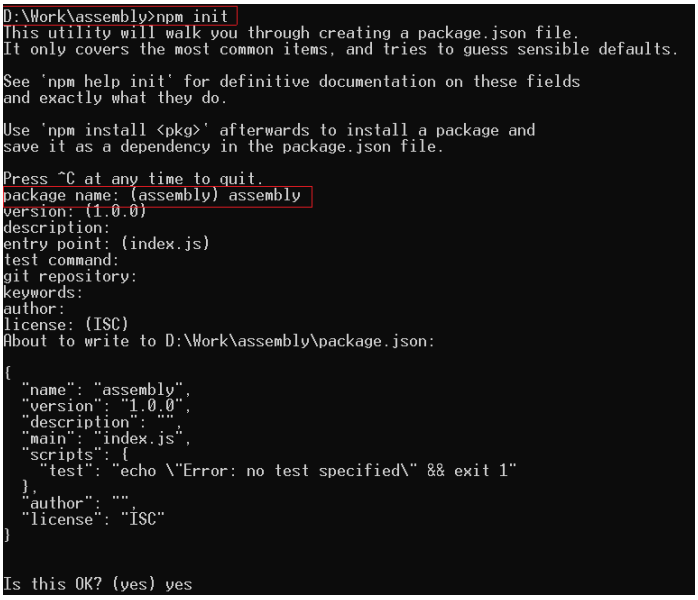
2.2.3.3 示例 Demo

2.2.3.3.1 合约编译

1. 下载Node.js软件并安装，安装成功后，执行如下命令查看对应版本(软件对应版本无强制要求)。

```
node -v
npm -v
```

2. 设置新目录assembly，在该目录下执行 npm init 命令，其中package name输入为assembly(目录名、package name建议保持一致，具体名称无强制要求，可自行定义)。



3. 执行如下命令通过npm安装加载器和编译器

```
npm install --save @assemblyscript/loader
npm install --save-dev assemblyscript
```

```
D:\Work\assembly>npm install --save @assemblyscript/loader

added 1 package in 557ms

D:\Work\assembly>npm install --save-dev assemblyscript

added 6 packages in 1s

1 package is looking for funding
  run `npm fund` for details
```

说明

若安装过程中出现“ idealTree:assembly: sill idealTree buildDeps”，请确认npm使用镜像源可正常访问。

- 查看npm镜像源配置命令
  npm config get registry
- 设置npm镜像源配置命令
  npm config set registry 国内镜像源地址

4. 执行如下命令，利用编译器提供的脚手架设置新项目
   npx asinit .

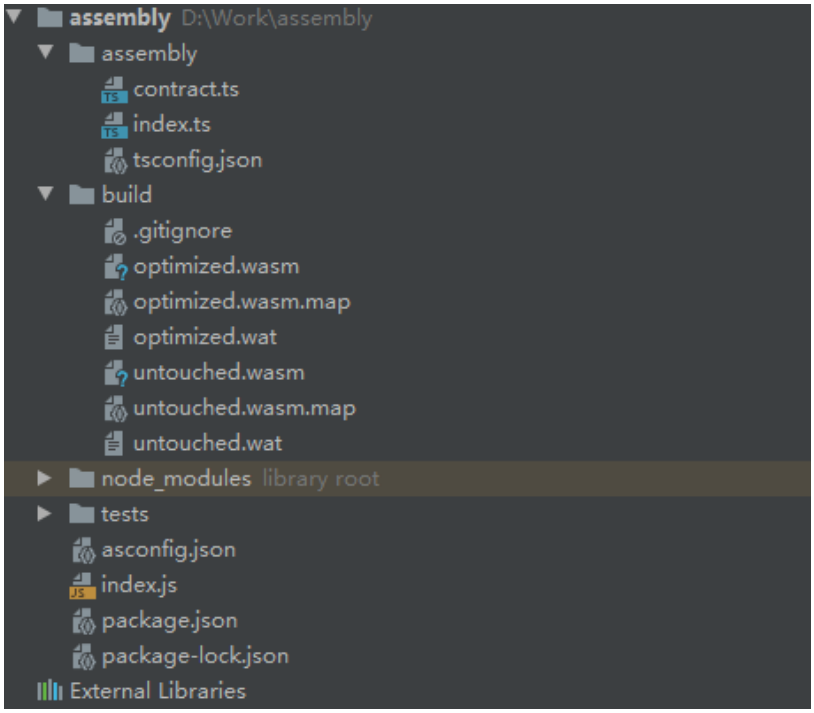
```
D:\Work\assembly>npx asinit .
Version: 0.19.20

This command will make sure that the following files exist in the project
directory 'D:\Work\assembly':

  ./assembly
  Directory holding the AssemblyScript sources being compiled to WebAssembly.

  ./assembly/tsconfig.json
  TypeScript configuration inheriting recommended AssemblyScript settings.

  ./assembly/index.ts
  Example entry file being compiled to WebAssembly to get you started.

  ./build
  Build artifact directory where compiled WebAssembly files are stored.

  ./build/.gitignore
  Git configuration that excludes compiled binaries from source control.

  ./asconfig.json
  Configuration file defining both a 'debug' and a 'release' target.

  ./package.json
  Package info containing the necessary commands to compile to WebAssembly.

  ./index.js
  Main file loading the WebAssembly module and exporting its exports.

  ./tests/index.js
  Example test to check that your module is indeed working.

The command will try to update existing files to match the correct settings
for this instance of the compiler in 'D:\Work\assembly\node_modules\assemblyscript'.

Do you want to proceed? [Y/n] y
```

5. 参考**示例Demo**完成合约文件contract.ts编写与合约SDK文件index.ts引用后，执行build命令编译AssemblyScript类型的合约文件，编译成功后在build目录下生成optimized.wasm字节码文件（该字节码文件可重新命名）。
   npm run asbuild

```
D:\Work\assembly>npm run asbuild

> assembly@1.0.0 asbuild
> npm run asbuild:untouched && npm run asbuild:optimized


> assembly@1.0.0 asbuild:untouched
> asc assembly/index.ts --target debug


> assembly@1.0.0 asbuild:optimized
> asc assembly/index.ts --target release
```

2.2.3.3.2 Demo 工程目录

合约开发和调测可参考合约示例Demo，单击链接获取AssemblyScript语言**合约工程Demo**。



1. build目录下optimized.wasm为合约编译后对应的wasm字节码文件(最终合约类型文件为optimized.wasm压缩成的*.zip包，**合约示例Demo**)。
2. assembly/index.ts 为开发智能合约文件(contract.ts)依赖的合约SDK。
3. assembly/contract.ts 为智能合约文件，本Demo中合约仅进行简单展示(实际合约文件contract.ts需自行开发)。
4. 合约文件的安装请参考**合约管理**。

说明

AssemblyScript语言类型合约不支持查询指定键的历史数据。

2.2.4 Solidity 合约开发

基本方案

1. 使用自研的容器合约环境集成Solidity的智能合约。
2. 外部搭建Solidity Web IDE，生成测试需要的合约安装字节码和合约调用字节码。
3. 将字节码二进制通过Hex编码字符串作为合约调用参数。

须知

1. Solidity合约默认名称为：NATIVE_CUSTOM_EVM。
2. 暂不支持event api。
3. Solidity事件机制实现需要依赖Fabric的event api，当前未做兼容，需要空实现。

2.2.4.1 SDK 接口

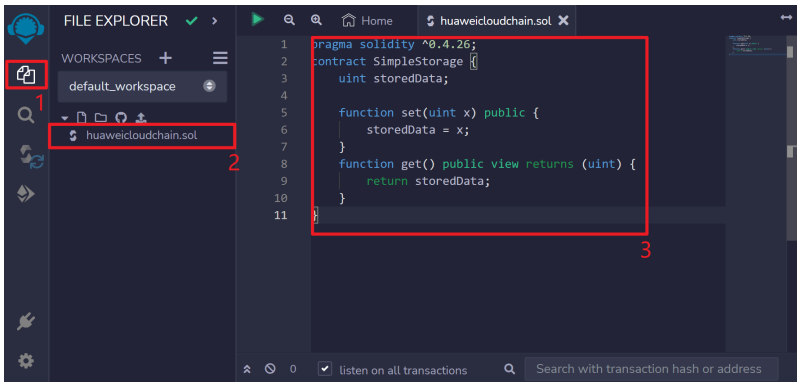
表 2-9 合约接口

接口	说明
Account	account分为contract account和user account，该account接口为获取user account address的接口。
Bytecode	通过contract address 可以获取到实际运行时执行的字节码（ runtime bytecode ），参见“合约管理->合约文件哈希值”。
Call	进行数据获取，更新，删除操作。 第一个入参为contract_address，第二个入参为一串bytecode，bytecode由function+parameter组成，SET= 60fe47b1，GET=6d4ce63c，以SET为例，设置value为000000000000000000000000000000000000000000000000000000000000002a，得到bytecode=60fe47b1000000000000000000000000000000000000000000000000000000000000002a。

2.2.4.2 合约示例

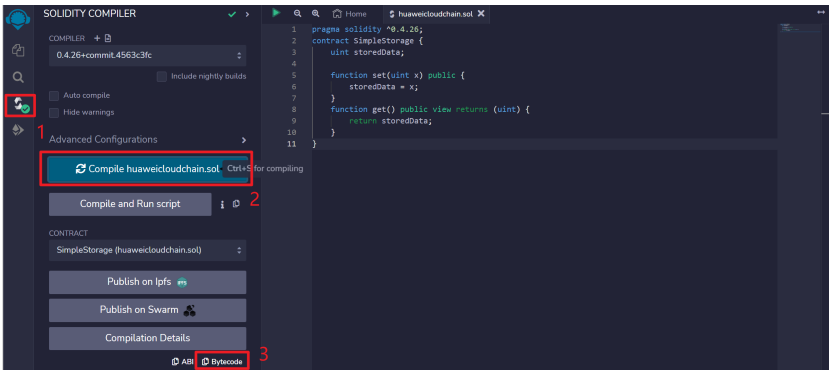
步骤1 进入在线编译器 **https://remix.ethereum.org**

步骤2 创建solidity合约文件，并粘贴示例合约。




```
pragma solidity ^0.4.26;
contract SimpleStorage {
    uint storedData;
    function set(uint x) public {
        storedData = x;
    }
    function get() public view returns (uint) {
        return storedData;
    }
}
```

步骤3 编译solidity合约，并复制bytecode结构体中的object值为合约字节码。




```
{
"linkReferences": {},
"object":"***",
"opcodes": "****",
"sourceMap": "****"
}
```

----结束

2.2.4.3 合约安装

Solidity语言合约安装步骤如下:

步骤1 完成solidity合约编译后，复制object对应的value（示例合约字节码可直接安装）。

```
608060405234801561001057600080fd5b5060df8061001f6000396000f3006080604052600436106049576000357c0100000000000000000000000000000000000000000000000000000000900463ffffffff16806360fe47b1146604e5780636d4ce63c146078575b600080fd5b348015605957600080fd5b506076600480360381019080803590602001909291905050506a0565b005b348015608357600080fd5b50608a60aa565b604051808281526020019150506040518091039f35b80600081905550565b60008054905090565600a165627a7a723058206ee3e59b37412c5e64cfc7227fe4151c945d2f5a6994eb945f126e8ce70082980029
```

步骤2 合约安装，可参考用户指南->**安装合约**。

----结束

2.2.5 JAVA 合约开发

2.2.5.1 SDK 配置

引用SDK的步骤如下:

步骤1 打开项目中的pom.xml文件。

步骤2 粘贴如下代码引入镜像仓。

```
<repositories>
    <repository>
        <id>maven-proxy</id>
        <url>https://repo.huaweicloud.com/repository/maven/huaweicloudsdk</url>
        <releases>
            <enabled>true</enabled>
        </releases>
        <snapshots>
            <enabled>true</enabled>
            <updatePolicy>always</updatePolicy>
            <checksumPolicy>fail</checksumPolicy>
        </snapshots>
    </repository>
</repositories>
```


步骤3 粘贴如下代码引用SDK。

步骤4 配置依赖

```
<dependencies>
   <dependency>
      <groupId>com.huawei.huaweichain</groupId>
      <artifactId>contract-sdk</artifactId>
      <version>0.2.2</version>
   </dependency>
</dependencies>
```

步骤5 等待自动拉取依赖。

----结束

说明

对于企业内部需要使用代理访问外网的情况，可以在用户目录（windows中如C:\Users\xxx\）下的.m2目录中settings.xml（用户配置）或maven安装目录下的conf目录中settings.xml（系统全局配置）里配置代理来实现。

找到settings.xml文件中的标签对，在其内配置代理信息，参考如下样例:

```
<proxies>
   <proxy>
      <id>myProxy</id>
      <active>true</active>
      <protocol>http</protocol>
      <username>xxx</username>
      <password>xxx</password>
      <host>xxx</host>
      <port>xxx</port>
      <nonProxyHosts>*xxx*.com</nonProxyHosts>
   </proxy>
</proxies>
```

2.2.5.2 SDK 接口

表 2-10 ContractStub 合约信息&状态数据库访问接口

接口	说明
String funcName()	功能：获取智能合约请求中指定的智能合约函数名称 入参：无 返回值：智能合约函数名称
byte[][] parameters()	功能： 获取请求参数 入参：无 返回值：用户执行智能合约逻辑时传入的多个参数，每个参数以[]byte表示
String chainId()	功能：获取智能合约所在链ID 入参：无 返回值：链ID

接口	说明
String contractName()	功能：获取智能合约名称 入参：无 返回值：智能合约名称
byte[] getKv(String key) throws ContractException	功能：获取状态数据库中某个key对应的value； 入参：某个键值对的key信息，只支持string类型，不可为空 返回值：value值，目前只支持[]byte类型；当key不存在时，value为null 抛出异常：当网络出错，消息超时等，抛出异常
void putKv(String key, byte[] value) throws ContractException	功能：写状态数据库操作，此接口只是将key、value形成写集，打包到交易中，只有当交易排序、出块、并校验通过之后，才会将key/value写入到状态数据库中 入参：要写入的键值对，要求key != ""，并且value != null 抛出异常：当网络出错，消息超时等，抛出异常
void delKv(String key) throws ContractException	功能：删除状态数据库中的key及其对应的value，此接口只是将待删除的key放入写集，打包到交易中，只有当交易排序、出块、并校验通过之后，才会将key删除 入参：要删除的key，要求key != "" 抛出异常：当网络出错，消息超时等，抛出异常

接口	说明
Iterator getIterator(String startKey, String endKey) throws ContractException	功能：查询状态数据库中，按字典序，以startKey开头，以endKey结尾的所有状态数据，结果以迭代器的形式呈现；注意，查询范围是左闭右开的，[startKey, endKey) 例如：startKey="11"，endKey="14"，所有key都是整数的话，则查询的结果中，key值包括："11","12","13"，不包括"14" 入参：startKey是待查询状态数据的按字典序的起始key，startKey != ""，endKey是待查询的状态数据的按字典序的结束key，endKey!= ""； 返回值：Iterator是查询结果的迭代器，可以通过此迭代器，按顺序读取查询结果 抛出异常：当网络出错，消息超时等，抛出异常
HistoryIterator getKeyHistoryIterator(String key) throws ContractException	功能：查询一个key对应的所有历史的value 例如：一个key的value曾经为1,2,3，当前value为4，则返回的迭代器结果中按顺序包含了1,2,3,4 入参：key是待查询历史value值的key信息，key != "" 返回值：HistoryIterator是按顺序包含了历史value结果的迭代器结构体变量 抛出异常：当网络出错，消息超时等，抛出异常

接口	说明
void saveComIndex(String indexName, String[] attributes, String objectKey) throws ContractException	功能：为objectKey保存索引信息，indexName_attributes_objectKey构成索引信息，注意，此处只是形成索引信息的写集，只有当含有此写集的交易经过排序、出块，并校验通过后，才会写入状态数据库 例如：存储key/value信息，key="zhangsan"，value={height=175, sex="male"}，如果以sex="male"作为查询条件，查询所有的key/value， 则需要反序列化所有的value，性能损耗较大，因此，为当前key/value建立一个sex相关的索引：indexName="sex"，attributes=[]string{"male"}，objectKey="zhangsan， 入参：indexName 索引标记，indexName != ""，attributes需要当做索引的属性，至少包含一个属性信息，objectKey 待索引的key值，objectKey != "" 抛出异常：当网络出错，消息超时等，抛出异常
Iterator getKvByComIndex(String indexName, String[] attributes) throws ContractException	功能：通过索引信息，查找满足某种查询条件的key/value，key/value以迭代器的形式输出 入参：indexName 索引标记，indexName != ""，attributes需要当做索引的属性，至少包含一个属性信息 返回值：满足索引条件的key/value的迭代器变量 抛出异常：当网络出错，消息超时等，抛出异常
void delComIndexOneRow(String indexName, String[] attributes, String objectKey) throws ContractException	功能：删除objectKey的某个索引，indexName_attributes_objectKey构成索引信息，注意，此处只是形成索引信息的写集，只有当含有此写集的交易经过排序、出块，并校验通过后，才会写入状态数据库 入参：indexName 索引标记，indexName != ""，attributes需要当做索引的属性，至少包含一个属性信息，objectKey 待索引的key值，objectKey != "" 抛出异常：当网络出错，消息超时等，抛出异常

表 2-11 Iterator 迭代器接口

接口	说明
boolean next() throws ContractException	功能：检查迭代器中是否还有下一个key/value 入参：无 返回值：bool值，true代表还有下一个值 抛出异常：当网络出错，消息超时等，抛出异常
String key() throws ContractException	功能：从迭代器中获取key 入参：无 返回值：key值 抛出异常：当网络出错，消息超时等，抛出异常
byte[] value() throws ContractException	功能：从迭代器中获取value 入参：无 返回值：value值类型为字节数组 抛出异常：当网络出错，消息超时等，抛出异常
void close()	功能：使用完迭代器之后，需要关闭迭代器 入参：无 抛出异常：当网络出错，消息超时等，抛出异常

表 2-12 HistoryIterator 查询某个 key 历史 value 信息的迭代器接口

接口	说明
long blockNum();	功能：获取当前迭代位置（某笔交易）的 BlockNum 入参：无
int txNum();	功能：获取当前迭代位置（某笔交易）的 TxNum 入参：无
byte[] txHash();	功能：获取当前迭代位置（（某笔交易））的hash 入参：无
boolean isDeleted();	功能：被查询的key，当前是否已经在状态数据库中被删除 入参：无

接口	说明
long timestamp();	功能：返回当前迭代位置（某笔交易）的时间戳 入参：无

2.2.5.3 合约结构

Java语言合约由合约文件及依赖包构成，包含包声明、依赖包导入、智能合约的方法定义。

说明

合约文件中，用户可自定义合约函数，需要实现Contract的init和invoke方法。

合约的结构如下：

```
package com.huawei.poissonchain;

import com.huawei.huaweichain.contract.Contract;
import com.huawei.huaweichain.contract.ContractException;
import com.huawei.huaweichain.contract.ContractStub;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class ExampleContract implements Contract {
    // 功能：合约的初始化（Init）接口，需要合约开发者在智能合约中实现此接口，供合约使用者在启动合约之后调用。注意，一般将合约启动时，首先需要执行且只需要执行一次的逻辑放到此方法中
    // 入参：stub是智能合约SDK为本次合约执行交易准备的上下文对象，可以通过stub提供的API函数，获取交易请求相关信息、读写状态数据库、写日志等
    // 返回值：需要返回给合约调用者（区块链客户端）的信息，没有信息需要返回时，返回值可以为null
    // 抛出异常：初始化过程的异常信息，可由合约编写者自行设定异常逻辑
    @Override
    public byte[] init(ContractStub stub) throws ContractException {
    }

    // 功能：合约被调用（invoke）接口，需要合约开发者在智能合约中实现此接口，将主要的合约执行逻辑，放到此接口内，供合约使用者调用。
    // 入参：stub是智能合约合约SDK为本次合约执行交易准备的上下文对象，可以通过stub提供的API函数，获取交易请求相关信息、读写状态数据库、写日志等
    // 返回值：需要返回给合约调用者（区块链客户端）的信息，没有信息需要返回时，返回值可以为null
    // 抛出异常：初始化过程的异常信息，可由合约编写者自行设定异常逻辑
    @Override
    public byte[] invoke(ContractStub stub) throws ContractException {
    }
}
```

2.2.5.4 合约示例

Java语言合约开发和调测可参考合约示例，使用步骤如下：

步骤1 **合约开发Demo**。

步骤2 编写Java合约。

步骤3 **Java合约示例**。

----结束

2.2.5.5 合约安装

Java语言合约安装步骤如下：

步骤1 部署合约时，执行mvn package，可在target目录下获取可安装的合约Jar包，文件名为contract.jar。

步骤2 合约安装，可参考用户指南 >区块链管理 > 合约管理。

----结束

说明

- 合约压缩文件格式：需确保格式为*.zip，且合约文件位于压缩包文件中的一级目录。
- 合约压缩文件中，只能包含文件名为contract.jar的文件。

2.3 SDK 介绍

2.3.1 概述

SDK 说明

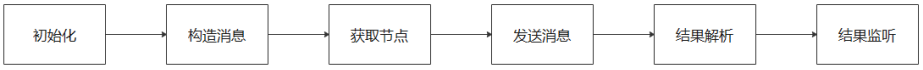
华为云区块链引擎目前提供Java、Golang两种语言SDK，区块链服务启动时会启动一系列grpc接口，监听客户端发送的消息，与客户端交互完成各种请求。在开发客户端时，如果从底层grpc接口开始，进行各种消息封装、消息发送、返回值解析等工作，不仅会导致开发量过大，并且造成重复劳动。

SDK则是将区块链服务提供的各种grpc接口进行封装，同时封装各接口所需类型的消息。在开发客户端时，只需要关注自己的业务逻辑，调用相应接口封装并发送消息即可，不需要关注底层消息发送接收的具体过程。

说明

1. SDK中相关方法的使用示例，可参考**应用程序开发**对应语言的示例Demo。
2. SDK在不同环境下支持的实例安全机制如下：
   - windows环境：ECDSA
   - linux环境：国密算法、ECDSA

基于 SDK 开发流程



1. 初始化：创建SDK实例对象，然后进行初始化。可以通过以下两种方式进行初始化：
   - 使用标准模板配置文件，调用 init 接口。
   - 调用接口自定义设置属性，包括：msp设置、TLS设置、节点信息设置。
2. 构造消息：由于消息类型较多，所以根据消息类型进行了消息类型进行封装，在构造消息之前，必须先获取消息类型对象，然后再基于获取对象中对应的方法封装消息。

3. 获取节点：获取消息发送节点对象。
4. 发送消息：所有的发送接口都封装在节点类中，发送消息时，先获取节点，再调用节点的消息发送对象获取方法。不同的消息对应不同的构建接口，同理，消息发送接口也根据接口类型进行了分装，在发送消息前，必须先获取接口类型对象，然后再基于获取对象中对应的方法发送消息。
5. 结果解析：服务端返回的消息中，均包含交易最后执行结果的标志位，以此判断交易是否执行成功。若结果为不成功，则info字段还包含了错误原因，用于分析定位。
6. 结果监听：对于业务交易和投票类型等需要落盘的交易，即使消息发送成功，后续落盘时还可能产生各种校验失败，导致交易无效。因此还需要监听交易是否最终落盘成功。

SDK 逻辑结构

SDK主要提供消息封装、发送模块及相应的配套组件。配套组件主要包含异常处理、配置文件解析、节点获取等功能接口。

- 消息封装

  由于消息类型较多，因此按类型进行了分类封装。消息封装相关接口均在build这个包下面，包含了ContractRawMessage、QueryRawMessage这几种消息构造的封装类。

  - ContractRawMessage：包含交易背书、落盘两阶段消息的构建。
  - QueryRawMessage：包含所有查询相关接口的消息构建，目前支持交易详情查询、链状态查询、区块查询、合约信息查询等接口。

- 消息发送

  同消息封装类接口，按类型进行了分类封装。消息发送相关接口均包含在action这个包下面，包含了ContractAction、QueryAction、EventAction这几种消息发送的封装类。

  - ContractAction：对应ContractRawMessage封装的消息的发送。
  - QueryAction：对应QueryRawMessage中封装的消息的发送。
  - EventAction：主要用于监听消息的最终状态，因为参数仅包含交易ID，消息封装的方法直接内置。同时所有的消息发送接口均提供同步和异步两种接口。

    说明

    同步接口入参均为需要发送的消息，返回值为一个ListenableFuture对象，用于监听消息发送结果。

    异步接口则传入需要发送消息的同时，还需要传入一个StreamObserver对象，用于异步获取消息发送结果。

接口说明

由于实现各种不同的交易发送，需要多个接口互相配合，因此接口说明按照实现功能进行归类介绍，而不是逐个接口类介绍。

所有消息发送方法，都提供同步和异步两种接口，异步接口多一个StreamObserver监听对象，无返回值。

另外SDK还提供其他扩展能力，有需要的用户可以参考。如Java接口：

public SdkClient(String configPath, Function<byte[], byte[]> func)

public SdkClient()

public void setIdentity(String type, byte[] key, byte[] cert)

public void setTls(byte[] key, byte[] cert, byte[][] roots)

public void addWienerChainNode(String name, String host, int port)

public String getTxId(Transaction tx)

public ChainRawMessage getChainRawMessage()

public Block buildGenesisBlock(String chainId, String path)

public Block buildGenesisBlock(String chainId, String path, Function<byte[], byte[]> func)

public Block buildGenesisBlock(String chainId, ChainConfig chainConfig)

public static GenesisConfig createGenesisConfig(String path)

public void addOrganization(String name, byte[] admin, byte[] root, byte[] tls)

public void addConsenter(String name, String org, String host, long port, byte[] cert, byte[] tee)

public ChainConfig getChainConfig(String chainId)

public Block buildGenesisBlock(String chainId, ChainConfig chainConfig)

public void addChainNode(String name, String hostOverride, String host, int port)

public RawMessage buildJoinChainRawMessage(Block genesisBlock)

public RawMessage buildJoinChainRawMessage(ByteString genesisBlock, Entrypoint entrypoint)

public RawMessage buildJoinChainRawMessage(ByteString genesisBlock, ConfigInfo config, Entrypoint entrypoint)

public static Entrypoint readEntrypointFile(String path) throws ConfigException

public ListenableFuture<RawMessage> joinChain(RawMessage rawMessage)

public void joinChain(RawMessage rawMessage, StreamObserver<RawMessage>)

public RawMessage buildQuitChainRawMessage(String chainId)

public ListenableFuture<RawMessage> quitChain(RawMessage rawMessage)

public void quitChain(RawMessage rawMessage, StreamObserver<RawMessage> responseObserver)

public RawMessage buildQueryChainRawMessage()

public ListenableFuture<RawMessage> queryAllChains(RawMessage rawMessage)

public void queryAllChains(RawMessage rawMessage, StreamObserver<RawMessage> responseObserver)

public RawMessage buildQueryChainRawMessage(String chainId)

public ListenableFuture<RawMessage> queryChain(RawMessage rawMessage)

public void queryChain(RawMessage rawMessage, StreamObserver<RawMessage>

public Builder.TxRawMsg buildUpdateChainRawMessage(String chainId, String path)

public Builder.TxRawMsg buildUpdateChainRawMessage(String chainId, ChainConfig chainConfig)

public Builder.TxRawMsg buildUpdateChainRawMessage(String chainId, String path, Function<byte[], byte[]> func)

public TxRawMsg buildUpdateConfPolicyRawMessage(String chainId, String policy)

public TxRawMsg buildUpdateOrgRawMessage(String chainId, String path, Function<byte[], byte[]> func)

public TxRawMsg buildUpdateOrgRawMessage(String chainId, List<ConfigSet.OrgUpdate> orgUpdates)

public RawMessage buildQueryChainUpdateVote(String chainId)

public ListenableFuture<RawMessage> queryVote(RawMessage rawMsg)

public void queryVote(RawMessage rawMsg, StreamObserver<RawMessage> responseObserver)

public RawMessage buildImportRawMessage(Contract contract, String path, String sandbox, String language)

public ListenableFuture<RawMessage> contractImport(RawMessage rawMessage)

public void contractImport(RawMessage rawMessage, StreamObserver<RawMessage> responseObserver)

public Builder.TxRawMsg buildManageRawMessage(String chain, String contract, String option)

public RawMessage buildQueryStateRawMessage(String chain, String contract)

public ListenableFuture<RawMessage> queryState(RawMessage rawMessage)

public Builder.TxRawMsg buildVoteRawMessage(Contract contract, String description, String policy, boolean historySupport)

public Builder.TxRawMsg buildSQLVoteRawMessage(Contract contract, String description, String policy, String schema, boolean isHistorySupport)

public ListenableFuture<RawMessage> transaction(RawMessage rawMessage)

public void transaction(RawMessage rawMessage, StreamObserver<RawMessage> responseObserver)

public RawMessage buildQueryLifecycleVote(String chainId, String contract)

public static ContractInvocation buildContractInvocation(String name, String function, String[] args)

public RawMessage buildContractRawMessage(String chainId, String contract)

public ListenableFuture<RawMessage> queryContractInfo(RawMessage rawMsg)

public void queryContractInfo(RawMessage rawMsg, StreamObserver<RawMessage> responseObserver)

如Go接口：

func NewGatewayClient(configPath string, decrypts ...func(bytes []byte) ([]byte, error)) (*GatewayClient, error)

func GenerateTimestamp() uint64

func (msg *ChainRawMessage) BuildGenesisBlock(chainID string, genesisConfigPath string, decrypts ...func(bytes []byte) (*common.Block, error)

func (msg *ChainRawMessage) BuildJoinChainRawMessage(genesisBlockBytes []byte) (*common.RawMessage, error)

func (msg *ChainRawMessage) BuildJoinMsgWithLatestConf(genesisBlockBytes []byte, latestConf *common.ConfigInfo) (*common.RawMessage, error)

func (msg *ChainRawMessage) BuildJoinMsgWithEntrypoint(genesisBlockBytes []byte, latestConf *common.ConfigInfo, entrypoint *common.Entrypoint) (*common.RawMessage, error)

func (action *ChainAction) JoinChain(rawMsg *common.RawMessage) (*common.RawMessage, error)

func (msg *ChainRawMessage) BuildQuitChainRawMessage(chainID string) (*common.RawMessage, error)

func (action *ChainAction) QuitChain(rawMsg *common.RawMessage) (*common.RawMessage, error)

func (msg *ChainRawMessage) BuildQueryAllChainRawMessage() (*common.RawMessage, error)

func (action *ChainAction) QueryAllChains(rawMsg *common.RawMessage) (*common.RawMessage, error)

func (msg *ChainRawMessage) BuildQueryChainRawMessage(chainID string) (*common.RawMessage, error)

func (action *ChainAction) QueryChain(rawMsg *common.RawMessage) (*common.RawMessage, error)

func (u *UpdateConfig) BuildUpdateConfPolicyRawMessage(chainID string, policy string) (*TxRawMsg, error)

func (u *UpdateConfig) BuildUpdateLifecycleRawMessage(chainID string, policy string) (*TxRawMsg, error)

func (u *UpdateConfig) BuildUpdateOrgRawMessageWithYaml(chainID string, path string, decrypt config.DecryptFunc) (*TxRawMsg, error)

func (u *UpdateConfig) BuildUpdateOrgRawMessage(chainID string, orgUpdates *common.ConfigSet_OrgUpdates) (*TxRawMsg, error)

func (msg *QueryRawMessage) BuildQueryChainUpdateVoteRawMessage(chainID string) (*common.RawMessage, error)

func (action *QueryAction) GetVote(rawMsg *common.RawMessage) (*common.RawMessage, error)

func (msg *LifecycleRawMessage) BuildImportRawMessage(c *Contract, path string, sandbox string, language string) (*common.RawMessage, error)

func (action *ContractAction) ContractImport(rawMsg *common.RawMessage) (*common.RawMessage, error)

func (msg *LifecycleRawMessage) BuildManageRawMessage(chain string, contract string, option string) (*TxRawMsg, error)

func (msg *LifecycleRawMessage) BuildVoteRawMessage(c *Contract, desc string, policy string, historySupport bool) (*TxRawMsg, error)

func (action *ContractAction) Transaction(rawMsg *common.RawMessage) (*common.RawMessage, error)

func (msg *QueryRawMessage) BuildQueryLifecycleVoteRawMessage(chainID string, contract string) (*common.RawMessage, error)

func BuildTxHeader(chainID string, []string domains) *common.TxHeader

func (msg *QueryRawMessage) BuildContractRawMessage(chainID string, contract string) (*common.RawMessage, error)

func (action *QueryAction) GetContractInfo(rawMsg *common.RawMessage) (*common.RawMessage, error)

如rpc接口：

rpc CallSM(stream ContractMsgsWrap) returns (stream ContractMsgsWrap)

rpc CreateChain (common.RawMessage) returns (common.RawMessage)

rpc DeleteChain (common.RawMessage) returns (common.RawMessage)

rpc QueryChainInfo (common.RawMessage) returns (common.RawMessage)

rpc QueryAllChainInfos (common.RawMessage) returns (common.RawMessage)

rpc GetLatestChainState(common.RawMessage) returns (common.RawMessage)

rpc GetBlockByNum(common.RawMessage) returns (common.RawMessage)

rpc GetBlockAndResultByNum(common.RawMessage) returns (common.RawMessage)

rpc GetTxByHash(common.RawMessage) returns (common.RawMessage)

rpc GetTxResultByTxHash(common.RawMessage) returns (common.RawMessage)

rpc GetBlockByTxHash(common.RawMessage) returns (common.RawMessage)

rpc GetContractInfo(common.RawMessage) returns (common.RawMessage)

rpc GetArchiveCheckPoints(common.RawMessage) returns (common.RawMessage)

rpc GetSnapshotMetadata(common.RawMessage) returns (common.RawMessage)

rpc GetArchiveConfigData(common.RawMessage) returns (common.RawMessage)

rpc GetArchiveStateMetadata(common.RawMessage) returns (common.RawMessage)

rpc GetArchiveBlockMetadata(common.RawMessage) returns (common.RawMessage)

rpc Invoke (common.RawMessage) returns (common.RawMessage)

rpc Query (common.RawMessage) returns (common.RawMessage)

rpc Import (common.RawMessage) returns (common.RawMessage)

rpc UnImport (common.RawMessage) returns (common.RawMessage)

rpc QueryState (common.RawMessage) returns (common.RawMessage)

rpc GetTeePubKey (common.RawMessage) returns (common.RawMessage)

rpc RegisterBlockEvent(common.RawMessage) returns (stream common.RawMessage)

rpc RegisterResultEvent(common.RawMessage) returns (stream common.RawMessage)

rpc RegisterBlockAndResultEvent(common.RawMessage) returns (stream common.RawMessage)

rpc RegisterBlockNumEvent(common.RawMessage) returns (stream common.RawMessage)

rpc RegisterTxEvent(stream common.RawMessage) returns (stream common.RawMessage)

rpc SendTransaction(common.RawMessage) returns (common.RawMessage)

rpc Query (common.RawMessage) returns (common.RawMessage)

如rest接口：

GET /node/health

GET /{chain_id}/stat/node

GET /{chain_id}/stat/zones?detail=1

GET /{chain_id}/stat/domains

GET /{chain_id}/{encoded_zone_id}/zone/nodes

GET /{chain_id}/{encoded_zone_id}/zone/linker

GET /{chain_id}/{encoded_zone_id}/zone/isolation

DELETE /{chain_id}/{encoded_zone_id}/zone/node?host={node_host}

POST /{chain_id}/{encoded_zone_id}/zone/node

POST /{chain_id}/{encoded_zone_id}/zone/linker

DELETE /{chain_id}/{encoded_zone_id}/zone/linker?host={node_host}

POST /{chain_id}/{encoded_zone_id}/zone/isolation?host={node_host}

POST /{chain_id}/{encoded_zone_id}/parent/zone

DELETE /{chain_id}/{encoded_zone_id}/parent/zone

POST /{chain_id}/{encoded_zone_id}/zone/id

DELETE /{chain_id}/{encoded_zone_id}/zone/id

POST /{chain_id}/{encoded_zone_id}/zone/id/state

2.3.2 Java SDK 介绍

2.3.2.1 SDK 配置

说明

1. linux环境，SDK的编译、运行，需要手动下载并配置openssl，单击链接下载openssl[**链接**]
   - 解压huaweichain_lib.zip压缩文件，进入解压后的huaweichain_lib目录下
   - 执行脚本文件：bash prepare_lib.sh
   - 执行库的导入命令：

     export ZK_STDLIB=/usr/local/include/zk/stdlib

     export C_INCLUDE_PATH=/usr/local/include/openssl:/usr/local/include/fhe:/usr/local/include/zk

     export LIBRARY_PATH=/usr/local/include/openssl:/usr/local/include/fhe:/usr/local/include/zk

     export LD_LIBRARY_PATH=/usr/local/include/openssl:/usr/local/include/fhe:/usr/local/include/zk
2. Java的项目管理工具有maven和gradle两种，本指导主要以maven为例。

引用SDK的步骤如下：

步骤1 打开项目中的pom.xml文件。

步骤2 粘贴如下代码引入华为镜像仓。

```
<repositories>
    <repository>
        <id>maven-proxy</id>
        <url>https://repo.huaweicloud.com/repository/maven/huaweicloudsdk</url>
        <releases>
            <enabled>true</enabled>
        </releases>
        <snapshots>
            <enabled>true</enabled>
            <updatePolicy>always</updatePolicy>
            <checksumPolicy>fail</checksumPolicy>
        </snapshots>
    </repository>
 </repositories>
```

步骤3 粘贴如下代码引用SDK。

```
<dependencies>
   <dependency>
      <groupId>com.huawei.poissonchain</groupId>
      <artifactId>client-sdk</artifactId>
      <version>0.2.14</version>
   </dependency>
</dependencies>
```

步骤4 等待自动拉取依赖。

----结束

说明

对于企业内部需要使用代理访问外网的情况，可以在用户目录（windows中如C:\Users\xxx\）下的.m2目录中settings.xml（用户配置）或maven安装目录下的conf目录中settings.xml（系统全局配置）里配置代理来实现。

找到settings.xml文件中的标签对，在其内配置代理信息，参考如下样例:

```
<proxies>
   <proxy>
      <id>myProxy</id>
      <active>true</active>
      <protocol>http</protocol>
      <username>xxx</username>
      <password>xxx</password>
      <host>xxx</host>
      <port>xxx</port>
      <nonProxyHosts>*xxx*.com</nonProxyHosts>
   </proxy>
</proxies>
```

2.3.2.2 通用方法

相关类

```
com.huawei.wienerchain.SdkClient
```

SdkClient对象包含获取服务节点、获取各种类型的消息构造器及交易ID等方法，基于SDK开发时，必须先构造该对象。

初始化 SDK 客户端

基于标准配置文件模板初始化SDK。

- 调用方法
  ```
  public SdkClient(String configPath) throws CryptoException, ConfigException, IOException
  ```
- 参数说明

参数	类型	说明
configPath	String	客户端SDK配置文件的绝对路径。

获取节点对象

根据节点名称，获取需要发送交易的节点对象。所有消息发送前，都必须调用该方法，获取发送节点对象，然后再获取对应的消息发送接口对象。

- 调用方法
  ```
  public WienerChainNode getWienerChainNode(String name) throws InvalidParameterException
  ```
- 参数说明

参数	类型	说明
name	String	节点名称。

- 返回值

类型	说明
WienerChainNode	WienerChainNode对象。

获取交易 ID

交易ID作为交易的标识，是交易哈希的十六进制字符串形式。以下获得交易ID的方法是通过计算交易哈希，然后转化为十六进制字符串获得的。

- 调用方法
  ```
  public String getTxId(Transaction tx)
  ```
- 参数说明

参数	类型	说明
tx	Transaction	交易实体。

- 返回值

类型	说明
String	交易ID。

2.3.2.3 利用合约发送交易

步骤1 合约调用信息构建。

- 接口方法

  ContractRawMessage.class
  ```
  public RawMessage buildInvokeRawMsg(String chainId, String name, String function, String[] args)
  ```
- 参数说明

参数	类型	说明
chainId	String	链ID。
name	String	合约名称。
function	String	调用合约中的方法名。
args	String[]	合约方法参数。

- 返回值

类型	说明
RawMessage	消息体，用于合约调用。

步骤2 背书请求消息构建。

- 接口方法

  ContractRawMessage.class
  ```
  public RawMessage getRawMessageBuilder(ByteString payload) throws CryptoException
  ```

- 参数说明

参数	类型	说明
payload	ByteString	合约调用信息，由invocation.toByteString()得到。

- 返回值

类型	说明
RawMessage	消息体，用于背书请求。

步骤3 背书请求消息发送。

- 接口方法

  ContractAction.class
  ```
  public ListenableFuture<RawMessage> invoke(RawMessage rawMessage) throws
  InvalidParameterException
  ```

- 参数说明

参数	类型	说明
rawMessage	RawMessage	背书请求的消息体，步骤2的返回值。

- 返回值

类型	说明
ListenableFuture	future对象，用于获取背书请求结果。

步骤4 落盘消息构建。

- 接口方法

  ContractRawMessage.class
  ```
  public TxRawMsg buildTxRawMsg(RawMessage[] rawMessages) throws
  InvalidProtocolBufferException, TransactionException, CryptoException
  ```

- 参数说明

参数	类型	说明
rawMessages	RawMessage	消息体，步骤3的背书请求结果。

- 返回值

类型	说明
TxRawMsg	交易消息体，包含原始信息和哈希。

步骤5 落盘消息发送。

- 接口方法

  ContractRawMessage.class
  ```
  public ListenableFuture<RawMessage> transaction(RawMessage rawMessage) throws InvalidParameterException
  ```

- 参数说明

参数	类型	说明
rawMessage	RawMessage	消息体，步骤4返回的交易消息体中的原始信息。

- 返回值

类型	说明
ByteString	落盘结果。

----结束

2.3.2.4 利用合约查询数据

步骤1 合约调用信息构建。

- 接口方法

  ContractRawMessage.class
  ```
  public RawMessage buildInvokeRawMsg(String chainId, String name, String function, String[] args)
  ```

- 参数说明

参数	类型	说明
chainId	String	链ID。
name	String	合约名称。
function	String	调用合约中的方法名。
args	String[]	合约方法参数。

- 返回值

类型	说明
RawMessage	消息体，用于合约调用。

步骤2 查询请求消息发送。

- 接口方法

  ContractAction.class
  ```
  public ListenableFuture<RawMessage> invoke(RawMessage rawMessage) throws
  InvalidParameterException
  ```

- 参数说明

参数	类型	说明
rawMessage	RawMessage	消息体，用于查询请求。

- 返回值

类型	说明
ListenableFuture	future对象，用于获取查询结果。

----结束

2.3.2.5 其他查询

2.3.2.5.1 查询块高

步骤1 消息构建。

- 接口方法

  QueryRawMessage.class
  ```
  public RawMessage buildLatestChainStateRawMessage(String chainId) throws CryptoException
  ```

- 参数说明

参数	类型	说明
chainId	String	链ID。

- 返回值

类型	说明
RawMessage	消息体，用于查询块高。

步骤2 消息发送。

- 接口方法

  QueryAction.class
  ```
  public ListenableFuture<RawMessage> queryLatestChainState(RawMessage rawMsg) throws
  InvalidParameterException
  ```

- 参数说明

参数	类型	说明
rawMsg	RawMessage	查询块高的消息体。步骤1返回的消息体

- 返回值

类型	说明
ListenableFuture	future对象，用于获取查询结果。

----结束

2.3.2.5.2 查询区块详情

步骤1 消息构建。

- 接口方法

  QueryRawMessage.class
  ```
  public RawMessage buildBlockRawMessage(String chainId, long blockNum) throws CryptoException
  ```

- 参数说明

参数	类型	说明
chainId	String	链ID。
blockNum	long	区块号。

- 返回值

类型	说明
RawMessage	消息体，用于查询区块详情。

步骤2 消息发送。

- 接口方法

  QueryAction.class
  ```
  public ListenableFuture<RawMessage> queryBlockByNum(RawMessage rawMsg) throws InvalidParameterException
  ```

- 参数说明

参数	类型	说明
rawMsg	RawMessage	查询区块详情需发送的消息。

- 返回值

类型	说明
ListenableFuture	future对象，用于获取查询结果。

----结束

2.3.2.5.3 查询交易执行结果

步骤1 消息构建。

- 接口方法

  QueryRawMessage.class

```
public RawMessage buildTxRawMessage(String chainId, byte[] txHash) throws CryptoException
```

- 参数说明

参数	类型	说明
chainId	String	链ID。
txHash	byte[]	交易哈希。

- 返回值

类型	说明
RawMessage	消息体，用于查询交易执行结果。

步骤2 消息发送。

- 接口方法

  QueryAction.class

```
public ListenableFuture<RawMessage> queryTxResultByTxHash(RawMessage rawMsg) throws InvalidParameterException
```

- 参数说明

参数	类型	说明
rawMsg	RawMessage	根据交易ID查询交易执行结果需发送的消息。

- 返回值

类型	说明
ListenableFuture	future对象，用于获取查询结果。

----结束

2.3.2.5.4 利用交易 ID 查询交易详情

步骤1 消息构建。

- 接口方法

  QueryRawMessage.class
  ```
  public RawMessage buildTxRawMessage(String chainId, byte[] txHash) throws CryptoException
  ```

- 参数说明

参数	类型	说明
chainId	String	链ID。
txHash	byte[]	交易哈希。

- 返回值

类型	说明
RawMessage	消息体，用于查询交易详情。

步骤2 消息发送。

- 接口方法

  QueryAction.class
  ```
  public ListenableFuture<RawMessage> queryTxByHash(RawMessage rawMsg) throws
  InvalidParameterException
  ```

- 参数说明

参数	类型	说明
rawMsg	RawMessage	根据交易ID查询交易详情需发送的消息。

- 返回值

类型	说明
ListenableFuture	future对象，用于获取查询结果。

----结束

2.3.3 Go SDK 介绍

2.3.3.1 SDK 配置

说明

linux环境，SDK的编译、运行，需要手动下载并配置openssl，单击链接下载openssl[**链接**]

1. 解压huaweichain_lib.zip压缩文件，进入解压后的huaweichain_lib目录下
2. 执行脚本文件：bash prepare_lib.sh
3. 执行库的导入命令：

   export ZK_STDLIB=/usr/local/include/zk/stdlib

   export C_INCLUDE_PATH=/usr/local/include/openssl:/usr/local/include/fhe:/usr/local/include/zk

   export LIBRARY_PATH=/usr/local/include/openssl:/usr/local/include/fhe:/usr/local/include/zk

   export LD_LIBRARY_PATH=/usr/local/include/openssl:/usr/local/include/fhe:/usr/local/include/zk

引用SDK的步骤如下：

步骤1 单击链接下载Go SDK[**链接**]。

步骤2 解压到应用程序所在目录，具体可参考Go应用程序开发的**示例Demo**。

----结束

2.3.3.2 通用方法

相关类

- GatewayClient对象包含获取服务节点、获取各种类型的消息构造器及交易ID等方法，基于SDK开发时，必须先构造该对象。
  ```
  client.GatewayClient
  ```
- BsClient对象包含富媒体文件上链、下载、操作记录查询等方法，使用区块链富媒体存储相关功能时，必须先构造该对象。
  ```
  bstore.BsClient
  ```

初始化 SDK 客户端

基于标准配置文件模板初始化Gateway SDK。

- 调用方法
  ```
  func NewGatewayClient(configPath string, decrypts ...func(bytes []byte) ([]byte, error))
  (*GatewayClient, error)
  ```
- 参数说明

参数	类型	说明
configPath	String	必填参数，客户端SDK配置文件的绝对路径。
decrypts	func(bytes []byte) ([]byte, error)	非必填参数，指定证书密文解密算法，默认为明文，无须解密（可变参数）。

初始化 SDK 文件存储客户端

基于已初始化的SDK Gateway客户端模板初始化富媒体存储客户端。

- 调用方法
  ```
  func NewBsClient(gatewayClient *client.GatewayClient, chainId string, endorserName, consenterName string) (*BsClient, error)
  ```
- 参数说明

参数	类型	说明
gatewayClient	*client.GatewayClient	必填参数，已初始化过的Gateway客户端。
chainID	string	链名称。
endorserName	string	背书节点名称，如“node-0.organization1”。
consenterName	string	共识节点名称，如“node-0.organization”。

生成当前时间戳

根据节点名称，获取需要发送交易的节点对象。所有消息发送前，都必须调用该方法，获取发送节点对象，然后再获取对应的消息发送接口对象。

- 调用方法
  ```
  func GenerateTimestamp() uint64
  ```
- 返回值

类型	说明
uint64	生成系统当前的UTC时间戳。

2.3.3.3 利用合约发送交易

步骤1 背书消息构建

- 接口函数

  func (msg *ContractRawMessage) BuildInvokeMessage(chainID string, name string, function string, args []string) (*common.RawMessage, error)
- 参数说明

参数	类型	说明
chainID	string	链ID。
name	string	合约名称。
function	string	调用合约中的方法名。

参数	类型	说明
args	[]string	合约方法参数。

- 返回值

类型	说明
*common.RawMessage	背书请求需发送的消息。
error	发送成功返回类型为nil，反之返回error。

步骤2 背书请求消息发送。

- 接口函数
  ```
  func (action *ContractAction) Invoke(rawMsg *common.RawMessage) (*common.RawMessage, error)
  ```
- 参数说明

参数	类型	说明
rawMsg	*common.RawMessage	上述背书请求需发送的消息。

- 返回值

类型	说明
*common.RawMessage	背书请求需发送的消息。
error	发送成功返回类型为nil，反之返回error。

步骤3 交易消息构建。

- 接口方法
  ```
  func (msg *ContractRawMessage) BuildTxRawMsg(rawMessages []*common.RawMessage)
  (*TxRawMsg, error)
  ```
- 参数说明

参数	类型	说明
rawMessages	[]*common.RawMessage	背书请求返回结果集合。

- 返回值

类型	说明
*TxRawMsg	包含交易hash的交易请求信息，该消息使用transaction接口发送。

类型	说明
error	构建成功返回类型为nil，反之返回error。

步骤4 交易消息发送。

- 接口方法
  ```
  func (action *ContractAction) Transaction(rawMsg *common.RawMessage) (*common.RawMessage, error)
  ```
- 参数说明

参数	类型	说明
rawMsg	*common.RawMessage	上述生成的交易消息。

- 返回值

类型	说明
*common.RawMessage	用于获取包含发送结果的消息。
error	发送成功返回类型为nil，反之返回error。

----结束

2.3.3.4 利用合约查询数据

步骤1 查询请求消息构建

- 接口函数

  func (msg *ContractRawMessage) BuildInvokeMessage(chainID string, name string, function string, args []string) (*common.RawMessage, error)
- 参数说明

参数	类型	说明
chainID	string	链ID。
name	string	合约名称。
function	string	调用合约中的方法名。
args	[]string	合约方法参数。

- 返回值

类型	说明
*common.RawMessage	查询请求需发送的消息。

类型	说明
error	发送成功返回类型为nil，反之返回error。

步骤2 查询请求消息发送。

- 接口函数
  ```
  func (action *ContractAction) Invoke(rawMsg *common.RawMessage) (*common.RawMessage, error)
  ```
- 参数说明

参数	类型	说明
rawMsg	*common.RawMessage	上述查询请求需发送的消息。

- 返回值

类型	说明
*common.RawMessage	查询请求返回的消息。
error	发送成功返回类型为nil，反之返回error。

----结束

2.3.3.5 文件上链

- 接口方法
  ```
  func (bc *BsClient) UploadFile(filePath, fileName string) (*UploadFileResponse, error)
  ```
- 参数说明

参数	类型	说明
filePath	string	待上链文件在本地的路径。当前支持不大于100MB的任意格式文件。
fileName	string	文件在链上的名称。不允许包含 "/"。

- 返回值

类型	说明
*UploadFileResponse	文件上链返回信息。
error	上链成功返回类型为nil，反之返回error。

2.3.3.6 文件下载

- 接口方法
```
func (bc *BsClient) DownloadFile(filePath, fileName string, versionId int) error
```
- 参数说明

参数	类型	说明
filePath	string	文件下载到本地的路径。
fileName	string	待下载文件在链上的名称。不允许包含 "/"
versionId	int	待下载文件的版本号。版本号要求大于等于1，可以通过**查询文件历史版本**获取文件的版本号信息。

- 返回值

类型	说明
error	下载成功返回类型为nil，反之返回error。

2.3.3.7 组织加密

- 接口方法
```
func (client *GatewayClient) EncryptDataWithE2EE(consensusOrgID, encOrgID string, decOrgIDs []string, data string, options ...interface{}) (txID string, err error)
```
- 参数说明

参数	类型	说明
consensusOrgID	string	共识组织ID
encOrgID	string	执行加密操作的组织ID。
decOrgIDs	[]string	除了执行加密操作的组织ID以外，可以解密该消息的组织ID列表。
data	string	需要加密的数据明文。
options	interface{}	其他选项，目前支持输入一个bool类型，用于指定加密后，是否更新群组密钥。

- 返回值

参数	类型	说明
txID	string	加密后返回密文对应的交易ID，解密时输入交易ID可以获得对应明文。
err	error	加密成功返回nil，否则返回error。

2.3.3.8 组织解密

- 接口方法
```
func (client *GatewayClient) DecryptDataWithE2EE(consensusOrgID, decOrgID string, txID string)
(data string, err error)
```
- 参数说明

参数	类型	说明
consensusOrgID	string	共识组织ID
decOrgID	string	执行解密操作的组织ID。
txID	string	密文对应的交易ID。

- 返回值

参数	类型	说明
data	string	解密后的明文信息。
err	error	解密成功返回nil，否则返回error。

2.3.3.9 其他查询

2.3.3.9.1 查询区块块高

步骤1 消息构建。

- 接口方法
```
func (msg *QueryRawMessage) BuildLatestChainStateRawMessage(chainID string)
(*common.RawMessage, error)
```
- 参数说明

参数	类型	说明
chainID	string	链ID。

- 返回值

类型	说明
*common.RawMessage	查询链状态需要发送的消息
error	查询成功返回类型为nil，反之返回error。

步骤2 消息发送。

- 接口方法

  QueryAction.class
  ```
  func (action *QueryAction) GetLatestChainState(rawMsg *common.RawMessage)
  (*common.RawMessage, error)
  ```

- 参数说明

参数	类型	说明
rawMsg	*common.RawMessage	上述查询链状态消息。

- 返回值

类型	说明
*common.RawMessage	用于获取包含发送结果的消息。
error	发送成功返回类型为nil，反之返回error。

----结束

2.3.3.9.2 查询区块详情

步骤1 消息构建。

- 接口方法
  ```
  func (msg *QueryRawMessage) BuildBlockRawMessage(chainID string, blockNum uint64)
  (*common.RawMessage, error)
  ```

- 参数说明

参数	类型	说明
chainID	string	链ID。
blockNum	uint64	区块高度。

- 返回值

类型	说明
*common.RawMessage	根据块高查询区块详情需发送的消息。

类型	说明
error	查询成功返回类型为nil，反之返回error。

步骤2 消息发送。

- 接口方法
```
func (action *QueryAction) GetBlockByNum(rawMsg *common.RawMessage)
(*common.RawMessage, error)
```
- 参数说明

参数	类型	说明
rawMsg	*common.RawMessage	上述根据块高查询区块详情消息。

- 返回值

类型	说明
*common.RawMessage	用于获取包含发送结果的消息。
error	发送成功返回类型为nil，反之返回error。

----结束

2.3.3.9.3 查询交易执行结果

步骤1 消息构建。

- 接口方法
```
func (msg *QueryRawMessage) BuildTxRawMessage(chainID string, txHash []byte)
(*common.RawMessage, error)
```
- 参数说明

参数	类型	说明
chainID	string	链ID。
txHash	[]byte	交易Hash。

- 返回值

类型	说明
*common.RawMessage	查询交易执行结果需发送的消息。
error	构建成功返回类型为nil，反之返回error。

步骤2 消息发送。

- 接口方法

  QueryAction.class

```
func (action *QueryAction) GetTxResultByTxHash(rawMsg *common.RawMessage)
(*common.RawMessage, error)
```

- 参数说明

参数	类型	说明
rawMsg	*common.RawMessage	上述生成的查询指定交易执行结果的消息。

- 返回值

类型	说明
*common.RawMessage	用于获取包含发送结果的消息。
error	发送成功返回类型为nil，反之返回error。

----结束

2.3.3.9.4 利用交易 ID 查询交易详情

步骤1 消息构建。

- 接口方法

```
func (msg *QueryRawMessage) BuildTxRawMessage(chainID string, txHash []byte)
(*common.RawMessage, error)
```

- 参数说明

参数	类型	说明
chainID	string	链ID。
txHash	[]byte	交易哈希。

- 返回值

类型	说明
*common.RawMessage	根据交易ID查询交易详情需发送的消息。
error	构建成功返回类型为nil，反之返回error。

步骤2 消息发送。

- 接口方法

  QueryAction.class


```
func (action *QueryAction) GetTxByHash(rawMsg *common.RawMessage) (*common.RawMessage, error)
```

- 参数说明

参数	类型	说明
rawMsg	*common.RawMessage	上述根据交易Hash查询交易详情消息。

- 返回值

类型	说明
*common.RawMessage	用于获取包含发送结果的消息。
error	发送成功返回类型为nil，反之返回error。

----结束

2.3.3.9.5 查询文件历史版本

- 接口方法

```
func (bc *BsClient) GetFileHistory(fileName string) ([]*bstore.FileHistory, error)
```

- 参数说明

参数	类型	说明
fileName	string	查询的链上文件名。

- 返回值

类型	说明
[]*FileHistory	文件历史版本信息列表，每条历史版本信息包含版本号、文件哈希值、首次上链时间、更新时间、上传者数据。
error	查询成功返回类型为nil，反之返回error。

2.3.3.9.6 查询文件操作记录

- 接口方法

```
func (bc *BsClient) GetFileOperation(fileName, startTime, endTime string) ([]*bstore.StorageEvent, error)
```

- 参数说明

参数	类型	说明
fileName	string	查询的链上文件名。

参数	类型	说明
startTime	string	查询记录的起始时间（秒时间戳）。
endTime	string	查询记录的结束时间（秒时间戳）

- 返回值

类型	说明
[]*StorageEvent	文件操作记录列表，每条操作记录包含操作者、操作类型、时间数据。
error	查询成功返回类型为nil，反之返回error。

2.3.4 SDK 升级与变更

须知

华为云区块链引擎提供Java、Go两种语言的SDK，供开发者使用。

- Java SDK：支持通过配置在线动态更新与升级。
- Go SDK：目前不支持在线更新与升级，需要手动下载并引入应用程序中。
- SDK版本：仅支持向前兼容，建议将SDK升级到最新版本。

2.3.4.1 Java SDK

说明

- Java SDK升级过程中只需修改pom.xml中SDK对应的版本号即可，详细见对应版本的使用配置。
- java SDK需要在项目pom文件中配置华为镜像仓。

```
<repositories>
   <repository>
      <id>maven-proxy</id>
      <url>https://repo.huaweicloud.com/repository/maven/huaweicloudsdk</url>
      <releases>
         <enabled>true</enabled>
      </releases>
      <snapshots>
         <enabled>true</enabled>
         <updatePolicy>always</updatePolicy>
         <checksumPolicy>fail</checksumPolicy>
      </snapshots>
   </repository>
</repositories>
```

表 2-13 Java SDK 版本与变更

版本号	使用配置	说明
0.2.2	<dependency> <groupId>com.huawei.poissonchain</groupId> <artifactId>client-sdk</artifactId> <version>0.2.2</version> </dependency>	适配实例版本1.0.2.1。
0.1.11	<dependency> <groupId>com.huawei.poissonchain</groupId> <artifactId>client-sdk</artifactId> <version>0.1.11</version> </dependency>	适配实例版本1.0.2.0。
2.1.0.6.41	<dependency> <groupId>com.huawei.wienerchain</groupId> <artifactId>wienerchain-java-sdk</artifactId> <version>2.1.0.6.41</version> </dependency>	适配实例版本1.0.1.1-1.0.1.10。 2.1.0.2.52升级至2.1.0.6.41时，合约发送、查询类方法需要适配，详细适配内容如下： • buildInvocation更新为buildInvokeRawMsg，参考**利用合约发送交易** • buildTransactionMessage更新为buildTxRawMsg，参考**利用合约发送交易** • buildInvocation更新为buildInvokeRawMsg，参考**利用合约查询数据**
2.1.0.2.52	<dependency> <groupId>com.huawei.wienerchain</groupId> <artifactId>wienerchain-java-sdk</artifactId> <version>2.1.0.2.52</version> </dependency>	适配实例版本1.0.0.39。

说明

版本查看方法：实例创建成功后，单击实例名称，进入概览页面，在链信息下方查看实例版本信息。

2.3.4.2 Go SDK

Go SDK升级过程如下：

步骤1 删除旧版本huaweichain SDK。

步骤2 下载新版本的SDK文件解压至原有的SDK文件路径。

----结束

表 2-14 Go SDK 版本与变更

版本号	下载	说明
0.4.9	链接	适配实例版本1.0.2.1。
0.2.15	链接	适配实例版本1.0.2.0。
2.1.0.6.41	链接	适配实例版本1.0.1.1-1.0.1.10。 2.1.0.2.52升级至2.1.0.6.41时，合约发送、查询类方法需要适配，详细适配内容如下： BuildTransactionMessage更新为BuildTxRawMsg，参考利用合约发送交易
2.1.0.2.52	不涉及	适配实例版本1.0.0.39。

说明

版本查看方法：实例创建成功后，单击实例名称，进入概览页面，在链信息下方查看实例版本信息。

2.4 应用程序开发

2.4.1 概述

为了能在应用程序中使用区块链服务，可参考本章节完成应用程序的开发。

开发完成后，应用程序可以调用合约将业务数据发送到链上或从链上进行查询，以及查询区块链的块高、查看某个区块的详情和查看某笔交易的详情等。

2.4.2 Java 应用程序开发

须知

windows环境：java客户端仅支持64位操作系统与64位Java SDK。

2.4.2.1 SDK 客户端配置

SDK客户端配置如下：

步骤1 SDK配置，可参考SDK配置章节。

步骤2 下载示例对应的配置文件，可参考用户指南->下载配置文件。

步骤3 客户端初始化，配置内容如下表，配置示例可参考应用程序的示例Demo。

----结束

表 2-15 客户端初始化配置介绍

参数	值
ConfigFilePath	链配置文件中yaml文件所在路径
ContractName	配置为合约安装时填写的合约名称，参考用户指南->**安装合约**。 Solidity合约的默认名称为"NATIVE_CUSTOM_EVM"。
ConsensusNode	实例下共识组织对应的节点，华为云区块链引擎共识组织下共3个共识节点，任选其一即可。
EndorserNodes	根据合约安装时选择的背书策略进行配置。 若为任意组织背书，则配置任意组织下的某一节点即可； 若为全部组织背书，则配置时选择每个组织的一个节点进行配置。
ChainID	链ID，配置为链配置文件中yaml中chain_id字段对应的值。

配置文件yaml示例：

```
chain_id: * # 链ID
client:
  type: *
  identity:
    keyPath: *.key
    certPath: *.crt
  tls:
    enable: true
    keyPath: *.key
    certPath: *.crt
    rootPath:
      - *.crt
nodes:
  node-0.organization-1yoamyube: # 节点名称，共识组织包含3个节点，非共识组织包含2个节点
    hostOverride: *
    host: *
    port: *
```

说明

- SDK客户端的配置与初始化

  可参考**SDK介绍**，配置示例可参考不同语言对应的应用程序示例Demo。
- 配置文件内容

  由证书和yaml配置构成，实例中每个组织对应一个yaml文件，可通过读取不同的yaml生成不同的SDK客户端对象。
- 客户端初始化

  需确保yaml文件中证书文件等路径为证书的实际存放路径。

2.4.2.2 SDK 客户端调用

SDK客户端配置后，即可调用SDK进行区块链相关的业务逻辑开发，调用示例可参考应用程序的示例Demo。

表 2-16 SDK 接口调用

类型	备注
合约接口	提供通过合约进行交易的发送与查询接口，详情可参考**利用合约发送交易**、**利用合约查询数据**
查询接口	提供查询块高、区块详情以及交易查询等常见接口，详情可参考**其他查询**

2.4.2.3 示例 Demo

Java语言示例Demo基于Java SDK开发，主要用于帮助开发人员理解并开发Java客户端应用程序，使用步骤如下：

步骤1 单击链接获取Java客户端示例[**链接**]。

步骤2 完成客户端初始化参数配置(App.java)，进行使用(配置可参考SDK客户端配置)。

----结束

示例Demo项目结构如下：

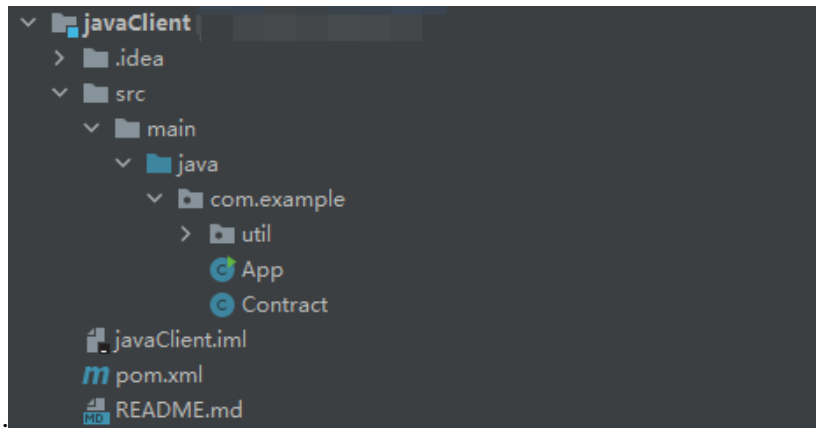


表 2-17 示例 Demo 目录结构

文件	介绍
App.java	业务端调用的示例，客户端的初始化配置。
Contract.java	通过合约对业务数据交互。对数据有修改的操作如插入和删除，需要调用其中的send方法。对数据的查询操作如查询某个键的历史，需要调用其中的query方法。
BlockUtil.java	BlockUtil用于从区块上获取数据。
TxUtil.java	TxUtil用于从交易上获取数据。
pom.xml	配置SDK仓库地址、SDK版本号。
README.md	提供详细的接口调用示例。

2.4.3 Go 应用程序开发

2.4.3.1 SDK 客户端配置

SDK客户端配置如下：

步骤1 SDK配置，可参考**SDK配置**章节。

步骤2 下载示例对应的配置文件，可参考用户指南->**下载配置文件**。

步骤3 客户端初始化，配置内容如下表，配置示例可参考应用程序的示例Demo。

----结束

表 2-18 客户端初始化配置介绍

参数	值
ConfigFilePath	链配置文件中yaml文件所在路径
ContractName	配置为合约安装时填写的合约名称，参考用户指南->**安装合约**。 Solidity合约的默认名称为"NATIVE_CUSTOM_EVM"。
ConsensusNode	实例下共识组织对应的节点，华为云区块链引擎共识组织下共3个共识节点，任选其一即可。
EndorserNodes	根据合约安装时选择的背书策略进行配置。 若为任意组织背书，则配置任意组织下的某一节点即可； 若为全部组织背书，则配置时选择每个组织的一个节点进行配置。
ChainID	链ID，配置为链配置文件中yaml中chain_id字段对应的值。

配置文件yaml示例：

```
chain_id: * # 链ID
client:
  type: *
  identity:
    keyPath: *.key
    certPath: *.crt
  tls:
    enable: true
    keyPath: *.key
    certPath: *.crt
    rootPath:
      - *.crt
nodes:
  node-0.organization-1yoamyube: # 节点名称，共识组织包含3个节点，非共识组织包含2个节点
    hostOverride: *
    host: *
    port: *
```

说明

- SDK客户端的配置与初始化

  可参考**SDK介绍**，配置示例可参考不同语言对应的应用程序示例Demo。

- 配置文件内容

  由证书和yaml配置构成，实例中每个组织对应一个yaml文件，可通过读取不同的yaml生成不同的SDK客户端对象。

- 客户端初始化

  需确保yaml文件中证书文件等路径为证书的实际存放路径。

2.4.3.2 SDK 客户端调用

SDK客户端配置后，即可调用SDK进行区块链相关的业务逻辑开发。

表 2-19 SDK 接口调用

类型	备注
合约接口	提供通过合约进行交易的发送与查询接口，详情可参考**利用合约发送交易**、**利用合约查询数据**。
查询接口	提供查询块高、区块详情以及交易查询等常见接口，详情可参考**其他查询**。

2.4.3.3 示例 Demo

须知

应用程序开发需要使用go mod，因此请确保GO111MODULE为on、镜像源配置。请确保可正常访问**华为云镜像网站**，环境设置命令如下

```
go env -w GO111MODULE=on
go env -w GOPROXY=https://repo.huaweicloud.com/repository/goproxy/
go env -w GONOSUMDB=*
```

Go语言示例Demo基于Go SDK开发，主要用于帮助开发人员理解并开发Go客户端应用程序，使用步骤如下：

步骤1 单击链接下载客户端示例[**链接**]。

步骤2 单击链接下载客户端所使用的Go SDK[**链接**]。

步骤3 添加Go SDK文件到客户端示例Demo目录下，完成Go SDK配置。

步骤4 完成客户端初始化参数配置(utils/config.go)，详情可参考**SDK客户端配置**。

----结束

示例Demo项目结构如下：

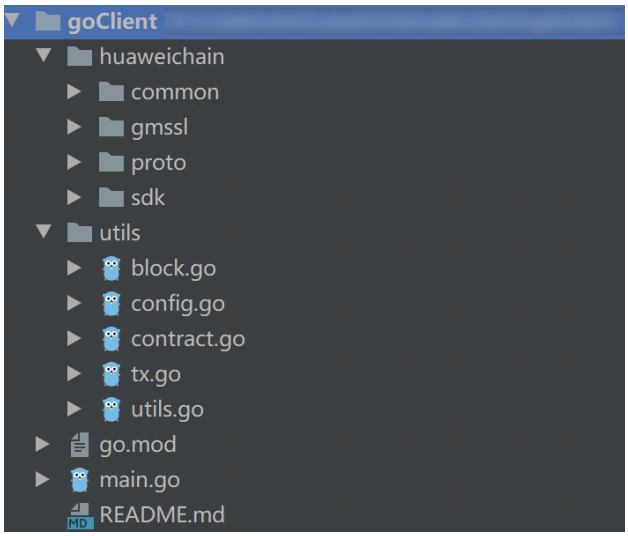


表 2-20 示例 Demo 目录结构

文件	介绍
huaweichain	Go语言客户端SDK，相关介绍请参考**Go SDK介绍**。
main.go	Go语言客户端主程序。
contract.go	通过合约对业务数据交互。对数据有修改的操作如插入和删除，需要调用其中的send方法。对数据的查询操作如查询某个键的历史，需要调用其中的query方法。
block.go	用于从区块上获取数据。
tx.go	用于从交易上获取数据。
config.go	客户端的初始化配置。
readme.md	提供详细的接口调用示例。